ACT: Audio, Control & Timing.
Signal Characteristics, Functions & Flow

To approach a modular sound synthesis system, you need to become acquainted with signals. To use a modular system intelligently requires making distinctions about measurable signal *characteristics* at module *outputs and* recognizing how signals *function* and *flow* when connected to specific module *inputs.* To excel at synthesis requires skill at imagining manipulations of signals and envisioning how signal characteristics such as frequency, waveform, and amplitude might–or more likely, *might not* correspond to sonic attributes such as pitch, timbre (tone color), and loudness. Unlike votes in Parliament or Congress, the ayes do not have it–the *ears* do. Ultimately, to achieve artistry, we must not forget the original motive: to make art, not to merely become adept at manipulating modules.

This implies a hierarchy with sound and sonic attributes near the top. Indeed, sound *is* the main thing. But that doesn't necessarily mean that it should be the *first* thing, especially when learning sound synthesis. Unfortunately, the simplistic "sound first" teaching approach exemplifies the way sound synthesis is typically introduced: by pointing out obvious sonic attributes (pitch, timbre, loudness) and making facile correspondences with electronic signal characteristics (frequency, waveform, amplitude).

Our classroom student outcomes and other experience persuade that this traditional approach blunts the creativity of its followers, limiting imagination as to how various

modules might be used in a sound synthesis system. Our *schema,* or particular way of thinking about the field, is shaped by extensive experience with learners. Its basis is more experiential than philosophical. You may come to appreciate the merits of this schema, but likely only as its rationale makes itself plain over this entire publication. We hope to convince primarily by example, rather than by argument. The hallmarks of a truly powerful idea are its simplicity–and effortless ability to manifest its utility.

We've opted to start with *signal characteristics*, *signal functions,* and *signal flow* in a modular sound synthesis system. That is, signals before sound, even though we are acutely aware that the natural impulse of most musicians is to orient themselves toward sound and its attributes. Our initial approach is further narrowed here toward signal *functions* and *flow,* treating signal characteristics only tangentially at this juncture. The most useful initial insights about modular sound synthesis systems spring more from focusing on signal functions and flow at *inputs,* rather than signal characteristics at *outputs*. Signal characteristics such as *amplitude, waveform, frequency, period,* and *polarity* are briefly introduced, but solely to illuminate this exposition of signal functions and flow. A few modules are introduced to facilitate this discussion, but their descriptions are kept sketchy as well. The conceptual (mathematical) operation of the signal controlled amplifier (SCA), or *multiplier* is explained in some detail, drawing attention to the need to become aware of rudimentary digital signal processing (DSP) operations. Comprehensive treatments of topics that are only sketched here will appear later. But for now–to

start our show, the curtain rolls back not on sound–but on signals . . .

Above all, we flatly assert that it is worse than misleading to start this exposition by equating *signal:*

(1) *frequency* with pitch;

(2) *waveform* with timbre (tone color); and

(3) *amplitude* with loudness.

Superficially, these seductively simple *correlates,* or mutual correspondences between signal characteristics and sonic attributes might serve us–if *audio* signals were our sole interest. Significantly, fields where audio signals *are* of primary interest (acoustics and audio engineering), are contexts in which these correlates have been formalized and popularized.

At any rate, sound designers consciously manipulate *control* and *timing* signals–as well as *audio* signals. And there's the rub. Signal-sound correlates that may be useful for audio are typically meaningless for control or timing. In contexts where audio-oriented signal-sound correlates are not relevant, referring to them tends to confuse rather than to clarify. Indeed, there are all too many instances in modular sound synthesis where standard signal-sound correlates prove to be *worse* than useless.

Best to start anew and learn "first principles" that don't have to be *unlearned* when actual cases are encountered. However, rare is the reader with no prior exposure to the concept of conventional signal-sound correlates. So, we can't simply ignore this ever present "elephant in the room." Consequently, we intend to put signal-sound correlates in a context unique to the needs of sound designers. And to articulate a simple memorable rule that explains–*without exception*–how signals function in a modular sound synthesis system.

A great deal of real world evidence supports the contrarian propositions that, for modular sound synthesis, signal:

(1) frequency is *not necessarily* pitch;

(2) waveform is *not necessarily* timbre;

(3) amplitude is *not necessarily* loudness.

There is a set of useful terms for signal characteristics, and a different set for sonic attributes. To equate terms across these two sets willy-nilly causes much mischief. Relationships between signal characteristics and sonic attributes certainly *must* be drawn eventually–but this is possible only within the particular contexts of how specific signals *function* and *flow,* as illustrated below. Each apparently "contrarian" proposition (1–3) above is supported by concrete examples below.

OK, signals before sound. But first, what *is* a signal*?* By analogy, it might be instructive to consider devices designed to regulate motor vehicle traffic. An intersection might have a signal or a sign. A stop sign has a single state, one message. An electrical traffic signal has multiple states, several color-coded messages. The sign is passive. The signal is active. If personified, the sign would seem unconscious, involuntary, inert. In contrast, the signal would seem conscious, voluntary, active, changing with purpose. Someone who shows signs of distress is not necessarily asking for assistance. A cry for help is a distress *signal* that actively seeks assistance. Signaling is an attempt to communicate, not to merely represent a static condition or unvarying command such as "stop." In this example, the signal means either "prepare to stop," or "stop," or "proceed." Without delving into the field of semiotics, the study of signs and symbols, let's define a *signal* as any electrical characteristic, gesture, visual indication, graphic display, mechanical configuration, photograph, number, action, computer code, color scheme, image, or sound that conveys information between apparatuses and/or people, based on a set of messages whose meanings have been subject to prior agreement. "One if by land–two if by sea." Now, *there* is a case where a signal was designed to clearly communicate critical information that a few patriots (or rebellious colonials) could spread.

To understand electronic signals specifically, let's look at electrical energy in general. Most electronic devices used in sound synthesis have a direct current (DC) power supply. It may be internal or *external*–that lumpy "wall wart" at one

end of a power cord. Direct current (DC) can be derived from alternating current (AC), *electrical energy* that is generated remotely, distributed by transmission lines, and provided at electrical outlets in various buildings. Alternating current (AC) is so-called “wall,” or *line* current. On the other hand, a *cell* can produce direct current (DC) locally, as a result of electrochemical action. A *cell* is the familiar individual unit–a *battery* is created by connecting several cells. One commonplace DC cell produces a low voltage signal, about 1.5 volts.

Cells connected in series create a battery with a higher *voltage,* the potential difference between electrical levels. The power supply for most modern electronic audio gear features a DC power supply that might be thought of as several cells connected in series. In fact, some equipment of interest actually uses such a battery. On the other hand, cells connected in parallel create a battery with a higher *amperage,* a measure of the flow of electrical current. An automobile starter battery has internal cells called “plates” connected in parallel, and this design provides sufficient power to crank a heavy internal combustion engine, even though such a battery produces a mere 12 volts. Most modern modular systems, samplers, electronic musical instruments, digital computers, etc. have a low voltage, low amperage DC power supply. Elsewhere, we’ll look at electrical energy and power supplies more closely, and it may be surprising how much we can learn that relates to signals and modules we can directly manipulate in a modular sound synthesis system.

To compare DC with AC, respective signal characteristics must be inspected. Let's look at DC first. Direct current (DC) is a signal with a single fixed *magnitude,* or size. Four DC cells @ 1.5 volts each, connected in series, constitute a battery with a signal output of 6.0 volts. We use a specific *number* to describe DC voltage level, or magnitude.

In a *virtual,* or computer software-based sound synthesis system, a signal comprises representations of *numerical* magnitudes, or sizes that we think of as signal *levels* that possibly change *over time* (as time passes). A number with a *fixed* magnitude (representing a single DC-like signal level) is called a *constant*. Despite what its name seems to imply, a constant is *not necessarily* unchangeable. A *constant* is a number that maintains its value during a specified set of mathematical calculations or scientific experiments, or for that matter, within a completed sound design. A constant maintains the same magnitude, or size until that number is changed. Similarly, a battery outputs a DC signal with an unvarying, or constant signal level. We're ignoring the fact that a real battery's level declines as its electrical energy is expended. Our virtual "battery" is an abstraction: it's a mathematically pristine *constant,* and its magnitude can last forever, unlike the voltage level of a real battery.

At first glance, it may seem that a constant is more like a sign than a signal. However, we can change this constant's magnitude, by adding or removing a cell in the "battery," or we could process this battery's output. Also, we could turn a battery signal on-off in various patterns, making multiple

encoded messages apparent, for instance when signaling with a flashlight. The output of a battery can be increased or decreased to create signals having different magnitudes, or a battery could be turned on-off in various patterns to create more than one message. A constant might be *programmed* to take on different magnitudes or *processed* to manifest a variety of messages. A constant has one specific value, or magnitude at any given moment, but that value can be changed.

In terms of musical instruments, the standard tunings for open guitar or violin strings are "constants" during performance. Ideally, those open string tunings remain constant, or *fixed.* (We wish!) However, alternative open string tunings, a technique known as *scordatura,* are used by many performers. In musical terms, stringed instrument players can change the "pitches" of their "open strings." In terms of signals, they can change the "magnitudes" represented by open string frequency "constants."

One way to visualize numerical size, or magnitude is to look at the *real number line* in mathematics. Numbers are arrayed on a horizontal line with zero (0) at its *origin,* or center. Numbers progress in magnitude outward in both directions from the origin, negative (–) to the left, and positive (+) to the right. Each number (except zero) has an *absolute magnitude* (size without regard to sign), as well as a *sign* (±). That is, a real number has a size, and it can be negative (–) or positive (+). A real number is not an *imaginary* number, a mathematical possibility we won't explore.

Any constant of interest to us is a real number, and it embodies both magnitude and *sign.* Plus (+), minus (–), and plus or minus (±) signs indicate *polarity* in the world of signals. (More on signal polarity later). Unique points on the real number line represent unique real numbers, and every real number has a unique position on the real number line. A constant with a specific magnitude and sign is indicated by a corresponding point on the real number line. The real number line is essentially a one dimensional *graph* that displays only numerical magnitude and sign. That is, the real number line does not display *time,* which is another dimension.

But *time* is critically important in music and sound. Max V. Mathews, the revered pioneer of direct digital synthesis (computer music) developed at Bell Labs in the late ‘fifties, pointed out that speaking, playing music, singing, and many other physical or mechanical processes could be represented by an appropriate set of time functions (Mathews and Moore 1970).

When time is displayed, it is usually graphed horizontally, as is the real number line. To graph the magnitude(s) of a signal *over time* (as time passes) requires a graph with *two* dimensions, called *axes* (aks´eez). When we rotate the real number line counterclockwise one-quarter of a circle, or ninety degrees (90°) around its zero origin, and display *time* on the horizontal *axis* (singular of *axes*), we then have a graph with *two* axes. For the moment, it’s not critical to know exactly how numbers are scaled on this new *vertical*

axis, just that this axis continues to represent magnitude and sign. Magnitude might correspond to *level*, *amplitude, sound pressure level (SPL), voltage, current, decibels (dB),* or *power* in our world of signals, and represents signal *size* or *intensity.* Sign (±) corresponds to signal *polarity,* which indicates whether a specific signal level is positive (+) or negative (–) with respect to zero (0). On the horizontal axis, *time* is divided into familiar units such as the second, with subdivisions such as the millisecond (one thousandth, or 1/1000 of a second).

The formal name for such a two dimensional graph or electronic representation, with a single choice from among various relevant measures of magnitude such as amplitude, voltage, etc. on the vertical axis, with *time* on the horizontal axis, is the *time domain.* Some graphic displays or devices that convey information in the time domain include EKG or ECG (electrocardiogram, heart monitor); seismograph (earth tremors indicator); barograph (ambient air pressure recorder); and *oscilloscope* (electronic signal display). Music notation is a variation on this *time domain* theme, with changes of pitch represented by *notes* on a vertical axis known as the *grand staff.* In most cases, *musical* time is *relative*, meaning that horizontal axis elements such as *time signature, measures, notes,* and *rests* are interpreted at a speed, or musical *tempo* chosen by the performer. *Musical* time is *subjective,* based on judgment, feelings, preference, interpretation, imagination, or opinion. Typical Musical Instrument Digital Interface (MIDI) sequencer "player piano roll" graphics mimic music notation by portraying an extended grand staff vertically, with an

explicit display of relative musical time functions such as MIDI *note on* and *note off* messages along the horizontal axis. Similar to an actual player piano paper roll, a sequence of MIDI notes can be played at a selected speed, or *tempo* that represents *relative* musical time. Strictly speaking, the *time domain* represents data during the passing of absolute, or real time. *Absolute,* or *real* time is measured by a clock that subdivides rigidly defined invariant units of time such as days, hours, minutes, seconds, milliseconds, etc.

Many electronic signals of interest to sound designers are depicted in the time domain, because these signals often have a level or many individual successive magnitudes that change over time. One example of magnitude changes over time is known as signal *waveform,* or waveshape. Waveform embodies several *signal characteristics* that can be displayed, measured, described, discussed, and understood without *hearing* anything, or even thinking about sound. A waveform does *not necessarily* represent sound and its attributes (pitch, timbre, loudness). On the other hand, electronic *waveforms* sometimes *do* represent sounds, and can cause the dynamic physical movements, or *vibrations* that loudspeakers make as they convert, or *transduce* such *audio* signal waveforms into sound. The *oscilloscope* provides dynamic electronic depictions of signal waveforms in the time domain, and can display control or timing signals, as well as audio signals.

Again, a word of caution about those simplistic signal-sound correlates, typically stated as: (1) frequency = pitch;

(2) waveform = timbre; and (3) amplitude = loudness; particularly in relation to how signals *function.* Waveform sometimes reveals specific signal characteristics at sight. However, signal characteristics don't indicate how a signal might *function* in a modular sound synthesis system: as an *audio, control,* or *timing* signal. And, signal function is a primary determining factor for what we actually hear! Even a waveform displayed continuously over time by an oscilloscope indicates nothing about the type of *input* (audio, control, timing) in a modular system to which you have actually *connected* that signal. This much is certain: signal characteristics *do not dictate* how a signal must function (audio, control, timing)– how it is used. Indeed, signal characteristics should not even *imply* limitations on signal functions in the mindset of a creative sound designer, as we shall see. Rather, the proper mindset should be "any output can be connected to any input–get creative!"

Now that we know about representing a signal in the *time domain*, let's revisit that fixed level DC (battery-like) signal. It's a *constant* in software-based (digital) terms, or a *bias* (DC voltage) in older hardware-based (analog) modular sound synthesis system terms. In math terms, magnitude and sign (±) are the elements of interest–we're talking about a constant. For now, any constant's *time,* or *temporal* dimension is assumed to be infinite, unknown, or at least indefinitely lasting, meaning that we can assume that a specific constant's magnitude and sign will continue without change until we alter it. Again, a constant is like that virtual electrical battery that never expends its energy.

In our context, a constant doesn't change of its own accord. A constant is *static,* or fixed–not dynamic.

But music is a *dynamic* art form. Most elements of music change over time–they're not *static,* or even temporarily *fixed* like a constant. It may not be intuitively apparent, or even seem reasonable that a *constant* could have much to do with the *dynamic* duo of sound and music. Nevertheless, to illustrate that signal characteristics do *not necessarily* dictate signal functions (audio, control, timing) in a sound synthesis system, we'll first show an extreme case: how a constant, or direct current (DC) signal with a single level, can function as an *audio* signal. Examples of a constant that functions as a *control* signal, then as a *timing* signal will follow. As stated previously, we're focusing on signal *functions* first.

When we connect a positive (+) constant (DC signal) to the audio input of an active audio monitor, its speaker elements quickly move outward, and in some designs may remain there while the constant is applied. The speakers produce an audible "click" or "bump" each time a constant (DC signal) of significant size is connected to or disconnected from an active audio monitor *signal input.* Perhaps you've noticed such audible "bumps" when powering up some of your audio gear. The sudden introduction or removal of electrical energy (AC) when turning an audio system on or off may cause momentary speaker movements similar to the "bumps" heard when connecting or disconnecting a constant (DC) to an audio monitor's *audio* input. When any loudspeaker element moves a large distance "very quickly"

(albeit not quite instantly), such sound bumps are likely. [If you experiment with connecting and disconnecting a constant to an active audio monitor input in a virtual sound synthesis system, carefully set the final "loudness" or "volume" control on your *real* amplifier to a low value at first. Then raise that value cautiously in small increments as you proceed with the experiment.]

Admittedly, a *constant* that functions as an *audio* signal makes a sound that seems to offer meager artistic potential. However, a creative sound designer might sample, process, and organize such speaker "bumps" into a valid musical composition or soundscape. At any rate, this exercise does show that a constant *can* function as an audio signal. The point is to initiate our ongoing argument in behalf of the *universality* and *equality* of all signals in terms of how they can *function* in a modular sound synthesis system. In a word, the functions of modules and signals in a modular system are not predetermined, despite any appearances or naming conventions to the contrary! *Connection* is everything, as we shall see.

We've adhered to an accepted convention for showing connections of modules depicted in our first block diagram. A *block diagram* is a group of icons that represents modules connected, or "patched" into a particular configuration. In this context, an *icon* is a line drawing that represents a particular module. A *module* is a recognizable hardware or virtual "building block;" a unit with a specific limited capability (oscillator, filter, amplifier, envelope generator, ring modulator, etc.) that can be connected to

others to form various structures or subsystems within a modular system. The block diagram itself is often referred to as a *patch,* even though a block diagram is only a graphic representation of the connections of physical or virtual modules that actually constitute a particular patch. First of all, outputs are connected to inputs! We've depicted the signal output of the constant on the right side of its icon, and the signal input of the audio monitor on the left side of its icon. This conforms to one popular convention for representing input-output (I/O) ports on module icons: *signal output* on the *right,* and *signal input* (where available) on the *left*. Other kinds of inputs, for *control* or *timing* signals usually appear on the *bottom* of an icon, as we shall see shortly. Similar to reading words in English, *signal flow is generally depicted from left to right* in this kind of block diagram.

We've demonstrated that a constant can function as an *audio* signal. How else might a constant function? As a *control* signal. Consider a high-fidelity audio amplifier, guitar amplifier, or recording studio console audio output bus. Each has a master *gain* (amplification or power factor) knob or slide pot *fader* that determines the overall playback signal *amplitude* (size), which happens to be heard in any of these contexts as *loudness.* (Rob Reiner's film *This is Spinal Tap* (1984) features a guitar amp whose master gain knob goes all the way up to "11" (eleven). Now, that must be a very powerful amplifier!) Master gain is typically set *manually* and remains at that setting until changed. A master gain setting is therefore a type of *constant and* is treated as such in most virtual systems.

In the amplifiers cited above, the numerical value of the constant that represents amplifier *gain* is the mathematical *factor* by which the *processed* (amplified) signal is *multiplied* to increase its overall signal *amplitude,* or size. In math, or digital signal processing (DSP) terms, an amplifier is a *multiplier,* where the signal being amplified is multiplied by *another signal,* such as the constant in these examples. Take note, however, that the signal being processed by an amplifier in a modular system does *not necessarily* have to be an *audio* signal, as is the case in this example. An amplifier does *not necessarily* have to function as an *audio* signal processor in a modular sound synthesis system, or elsewhere. Indeed, amplifiers that have absolutely nothing to do with sound are found in various kinds of electronic circuitry. Amplification is always about signal processing (multiplication), but *not necessarily* about sound.

A special kind of amplifier that appeared as a module in analog modular sound synthesis systems, dating from the era (1964) when popular synthesizers were pioneered by Robert A. Moog, is known as a voltage controlled amplifier (VCA), spoken "V-C-A." [Dr. Moog's name rhymes with *rogue* or *vogue;* no need to make a sound like a cow!] In modern digital signal processing (DSP) terms, this module is a *signal controlled amplifier* (SCA), spoken S-C-A. An SCA is a special kind of amplifier, but *any* amplifier is defined in digital signal processing (DSP) terms as a *multiplier.* Multiplication requires two numbers, multiplier

and multiplicand. An SCA *module* multiplies two signals, so it necessarily must have at least *two* inputs.

A *multiplier,* or amplifier is also categorized as a *module,* specifically a *signal processor.* A signal processor has a *signal input* and a *signal output,* so a signal can potentially flow *through* it. (More on signal flow later). SCA *signal input* is apparently one of those *two* inputs needed to facilitate multiplication of signals. The second requisite SCA input is a so-called *control input* to which we might connect our constant. By convention, a *control input* is shown on the *bottom* of a module's icon, which for the signal controlled amplifier (SCA) looks like a triangle. [The control input of an SCA is sometimes shown on the *top* of its icon. We'll show it on the bottom for the time being.]

A constant connected to the SCA control input provides a fixed factor that determines the amount of gain that the SCA subsequently provides. A *factor* is a number by which some other number can be multiplied. *Gain* is the extent of signal amplification, represented by comparing the output (O) signal magnitude to input (I) signal magnitude of a device. Gain is often represented as a simple output to input (O:I) voltage ratio (2:1), that might also be expressed in decibels (+6dB), a power ratio. A *ratio* is a quantitative relationship between two values that indicates the number of times one value is contained within the other. Gain could also be equated simply to the size of the math factor, or *multiplier* at the SCA control input that alters instantaneous levels of the *processed* signal, or *multiplicand* connected to

the SCA *signal input.* An *instantaneous* level has a size, or magnitude that is present during an extremely small, or *infinitesimal* interval of time whose duration is close to, but slightly larger than zero. [In digital terms, an *instantaneous* level could be thought of as a *single sample* value in a sampled sound file. A single sample occupies a small interval of time when played back at any standard sample rate (SR).] The instantaneous level of the multiplicand at the SCA *signal input* is *multiplied* by the instantaneous level of the multiplier at the SCA *control input* during successive instants of time. Each of these two SCA input levels is a mathematical *factor* with respect to the other. The succession of *products* that result from multiplying these two *factors,* or *instantaneous signal levels* from moment to moment, constitute the signal that flows out of the SCA *signal output.* In terms of math and DSP (digital signal processing), amplifier output is a succession of *products* of multiplication of the instantaneous *factors* at the amplifier's two inputs. Amplification might be thought of as multiplication of instantaneous signal levels (analog) or numbers (digital) over time.

Digital signal processing (DSP) is all about *math.* We don't have to actually *do* the math that DSP involves to use a modular sound synthesis system. However, if we at least know *which* math operations (multiplication, division, addition, etc.) are executed by a module, it helps us better understand how that module deals with signals. As we've seen, an amplifier is a *multiplier.* Later, we'll point out some of the math operations that various modules execute, particularly when we discuss module features in detail.

So far, the current patch seems to depict any familiar *audio* amplifier with a knob or fader that sets a fixed gain *manually*–the typical analog audio amplifier "volume" or "loudness" knob setting represents something similar to our constant in a DSP (digital signal processing) environment. However, we were careful to introduce the SCA *not necessarily* as an *audio* amplifier. In generic terms, an SCA changes the *amplitude* (size) of the signal it processes, without regard to where that SCA's *signal output* is connected–*loudness* is *not necessarily* involved. An SCA might be used to alter the amplitude of a control or timing signal, rather than an audio signal. That is, SCA *signal output* might be connected to a control or timing input, rather than an audio input. So, how could we possibly justify categorizing the SCA as an "audio" signal processor, ignoring its other possible functions? Obviously, we shouldn't. So, *we don't,* because the *function* (audio, control, timing) of any SCA (or any other module) remains undefined until its *signal output* is connected to a specific type of *input* (audio, control, timing). Eventually, you'll become adept at using the SCA in various ways in patches. At the moment, we're using this SCA to show how a *constant* can function as a *control signal.*

To provide a convenient shorthand for those "audio, control, timing" inputs and associated signal functions referred to repeatedly, we've coined the acronym ACT, spoken like the word *act:*

ACT stands for "audio, control, timing," indicating the ways a signal can function in a modular sound synthesis system.

Let's assume that our SCA *signal output* is connected to the *signal input* of an active *audio* monitor. This *connection* (nothing more and nothing less) defines this particular SCA as an *audio* signal processor. Neither SCA features nor SCA signal handling characteristics have anything to do with defining how an SCA *functions* in a patch–it's solely about where its *signal output* is ultimately connected! Assume that the *signal output* of a noise generator (NG) is connected to this SCA *signal input.* Because the function of the final, or *ultimate* input destination for the noise generator (NG) signal in this patch is *audio* (via SCA signal input, flowing through the SCA to its signal output), the noise generator (NG) therefore functions as an *audio* signal generator. The noise generator (NG), SCA, audio monitor, and graphic "patch cords" that connect these modules comprise an *audio signal path* (ASP) because the *ultimate,* or final input of this signal path is *audio*.

We mentioned that a signal controlled amplifier (SCA) is a special kind of amplifier. Like most amplifiers, the SCA alters the *amplitude*, or size of the *processed* signal–the one connected to the amplifier's *signal input.* However, a standard SCA has an *internal* default gain factor of zero (0), meaning that *initially* an SCA will *not* pass any signal. A *default* value is one built into the basic design, but typically the user can alter or override this default value. "By default," the typical SCA does *not* pass the signal

connected to its *signal input* to its *signal output,* because *default* SCA gain is set to zero (0) internally. The amplitude of the signal connected to SCA *signal input* is multiplied by this *internal* default gain value, or *constant* of zero (0), hence no output. We can reasonably deduce that this zero (0) default constant must be connected to an unseen internal *control input* within the SCA. [In some designs this internal SCA default constant is represented by an adjustable "bias" slide pot, knob, or "numerical" (rectangular box) with values that range from zero (0) to some positive (+) maximum known as *full scale*. In this case, the SCA can be "biased" (in analog terms) or "offset" (in digital terms) using this built-in slide-pot, knob, or numerical to provide a particular fixed SCA gain, rather than the initial, or *default* of zero (0) gain.]

At any rate, the SCA seems to be an amplifier designed specifically to *not necessarily* amplify, given its default zero (0) internal control input level! However, the SCA also has *external* control inputs to which one typically connects signals from signal sources found within the system. Any signal connected to an external control input *adds* to, or *sum*s with this internal default signal level of zero (0) in order to control amplifier gain, thereby potentially allowing the SCA to pass its processed signal to its *signal output.* A strange and wonderful amplifier, the SCA. Its initial "no output" zero (0) gain internal default value allows us to subsequently connect strange and wonderful externally available *control* signals in order to change processed signal gain, thereby dynamically *shaping* the amplitude (size) of whatever signal the SCA processes.

More important, we're not stuck with connecting a simple constant that behaves like a fixed knob position or fader setting. We could connect a *dynamic* signal to an SCA external *control* input and control the amplitude of the signal it processes dynamically. In essence, *signal controlled* (SC) modules such as the SCA are "automated" when we connect dynamic signals to their external *control* inputs.

For the moment, we'll defer possibilities of strange and wonderful control signals, and *will* connect our lowly *constant* to this particular SCA *control* input. This *connection* alone substantiates that a *constant* can function as a *control* signal. No description of the signal characteristics of this, or *any* other control signal should be necessary to determine its function. It's simply about connections, and how *you* make them! When we connected a positive (+) constant to the SCA *control input,* SCA gain subsequently became positive (+), which increased the *amplitude* of the signal being processed, noise in this case. So, we *heard* noise, because our SCA's *signal output* is connected to an *audio* input. In many *virtual,* or computer software-based *multiplier* module designs, *either* a positive (+) *or* a negative (–) signal connected to the external *control input* of a multiplier (amplifier) causes an *increase* of gain. [For future reference, compare *multiplier, four quadrant* with *multiplier, two quadrant* in the Learner's Glossary. The SCA in these examples is a classic *two quadrant multiplier,* which responds only to positive (+) signals at its "control" input. In such a device, zero (0) *or* a negative (–) control signal of any magnitude results in

multiplication by zero (0), causing an attendant zero output from the SCA.]

In this patch, the increase of SCA *gain* due to connection to its *control* input of a positive (+) constant allows the noise signal at the SCA *signal input* to flow through the SCA to its *signal output.* For purposes of creating block diagrams, the noise signal flows out of the right side of the noise generator (NG) icon, into the left side of the SCA triangular icon, and out of the right side of the SCA. This signal from the SCA signal output flows to the signal input of the audio monitor, whose icon suggests the shape of a loudspeaker horn. By convention, we show the *control* input of the SCA on the *bottom* of its icon, to which we have connected our constant.

Under these conditions, we hear the noise generator (NG) signal–a chaotic roaring, rushing sound. The unprocessed output of a standard noise generator (NG) provides no clear sense of pitch when heard, because this *noise* signal is *random*–its magnitudes, or instantaneous levels fluctuate unpredictably from moment to moment. The SCA shapes the *amplitude* (overall, or averaged size) of this noise signal. In this context, the SCA is an audio signal processor that we can use, *in conjunction with* the connected constant, to pass the noise signal connected to the SCA signal input. This particular SCA is connected to the *audio* input of an active audio monitor, and we therefore hear noise when a positive (+) constant is connected to the SCA control input. When we *disconnect* the positive (+) constant, or control signal from the SCA external control input, we no longer

hear the noise signal, because SCA gain is once again controlled by its internal zero (0) *default* value. This particular SCA functions as an audio signal processor solely because its *signal output* is connected to the *signal input* of an active audio monitor–an *audio* input.

When a constant is connected to the *control* input of this SCA, that constant functions as a *control* signal. So, a constant *is* a control signal, right? *Not necessarily.* Recall that we made a constant function as an *audio* signal previously, making loudspeakers "go bump in the night." Momentarily, we'll show how a constant can function as a *timing* signal. However, in the patch presently under consideration, the constant *does* function as a *control* signal, because it is connected to a *control* input, as depicted on the bottom of the SCA icon. Previously, the same constant, when connected to an *audio* signal input, functioned as an *audio* signal. Evidently, we've converged on an obvious and disarmingly simple way to determine how signals *function* in a modular sound synthesis system:

The functions of a signal are determined solely by the types of ACT (audio, control, timing) inputs to which that signal is connected.

This is the ACT *dictum,* or rule that lets us determine how any signal in a modular sound synthesis system functions: as an audio, control, or timing (ACT) signal. The ACT dictum is so important, we'll state it repeatedly, in slightly different ways:

The functions of a signal are not dictated by its signal characteristics, but by the functions of inputs to which it is connected.

There is yet another distinction between an SCA (signal controlled amplifier) and an ordinary amplifier. The external control inputs on an SCA can accept signals *other* than the constant we just connected. An ordinary audio amplifier is typically "stuck" with a *static* (fixed) manual control setting–like our constant, that determines amplifier gain. On the other hand, when we connect a *dynamic,* or fluctuating signal to an SCA's external control input, SCA gain is then controlled dynamically. A variety of signals might be connected to the control inputs of an SCA, altering its gain, thereby altering the amplitude (overall size) of the signal being processed. Any *signal controlled* (SC) module such as a signal controlled amplifier (SCA) is so-named to indicate that it can be controlled dynamically by signals *external* to that module–by a connected signal or combination of signals obtained from other modules in the system. A *signal controlled* (SC) module is not restricted solely to its fixed, or *static* manual or internal control features within that module. This (SC) capability for a module to accept externally generated control signals is at the heart of the *dynamism* and versatility of a modular system. In the earliest incarnation of the modern modular sound synthesizer such modules are referred to as *voltage controlled* (VC).

Take note that we speak of SCA *gain,* the measurable factor by which an amplifier multiplies processed signal

amplitude, a signal characteristic. We do not automatically associate SCA gain, and the amplitude of the signal the SCA processes, with *loudness*–a sonic attribute. The point is important: in a sound synthesis environment, a given signal characteristic such as amplitude does *not necessarily* correspond to, or *correlate* with a particular sonic attribute such as loudness. For the sound designer, it's best to think about *signal characteristics* when dealing with modules and signals. Learning to correlate signal characteristics with sonic attributes will come later, after signal functions and flow are understood. Sound is elusive and difficult to describe, but signal characteristics can be measured handily, and signal functions can be determined simply using the ACT dictum. Most amplifiers (read SCA here) deal with *amplitude,* a measurable signal characteristic. But, due to the possibility that a given SCA *signal output* can be connected to various ACT (audio, control, timing) inputs, a particular SCA is *not necessarily* associated with changes of *loudness,* a specific sonic attribute. (An example where SCA gain *is* associated with loudness is provided below). Again, an SCA is *not necessarily* an *audio* signal processor. The signal-sound correlation of the signal characteristic *amplitude* with the sonic attribute *loudness* may *seem* compelling to some, but only because signal characteristics and sonic attributes have been consistently and superficially equated. In particular, the tendency to define signal characteristics solely in terms of *audio* (sonic attributes) limits thinking about how to use modules. In time, you'll learn to use modules such as the SCA to process signal(s) that function in any possible way: audio, control, timing (ACT).

At last, some strange and wonderful control signals! For example, keyboard *note number* might be connected to control SCA gain in steps. Note number is a type of *step* signal, as exemplified by MIDI keyboard note number, keyboard gate, keyboard velocity, "analog" note sequencer output, and constant signals, etc. A step signal features discrete changes of signal size–*steps*. A *discrete* signal has magnitudes limited to specific numerical values. That is, these values are not free to vary continuously, and values *between* two possible adjacent discrete values just can't be expressed by that signal! The *step* signals that represent keyboard note numbers might be thought of as a series of constants whose magnitudes *increase,* typically as progressively higher notes are played on the keyboard. [Keyboard note number *is* defined in the original Musical Instrument Digital Interface (MIDI version 1.0) specification as such a series of constants: middle C = 60, C-sharp = 61, D = 62, and so forth as one progresses up the keyboard.] Now that we've made this particular connection, does playing keys on a keyboard therefore always result in hearing different *loudness* steps? *Not necessarily.* First of all, our prior experience confirms the contrary: a keyboard is typically used to control what we hear as *pitch* in steps. However, it is *stultifying* (look up this word, or we have people who will track you down and stultify you), and confusing to automatically associate a particular *signal characteristic* with a specific *sonic attribute.* Hence our warning up front about adhering to glib signal-sound correlates. A keyboard might be used to control *any* sonic attribute, when we learn to connect

keyboard outputs to appropriate control inputs. Alternatively, a keyboard might also be used solely as a timing device, or even as an audio signal generator!

Of what then, can we be certain in this *seemingly* uncertain world of signal functions where a keyboard does *not necessarily* control pitch, and a constant (DC signal) can function as an audio signal? It's really *very* easy: simply *ACT.* We can be confident that a *particular* constant is a *control* signal whenever that constant is connected to any *control* input. So, keyboard note number functions as a *control* signal when it's connected to any *control* input. While it's possible to discover a good deal about such signals by measuring their characteristics prior to connecting them within the system, only when we *connect* the constant or the keyboard note number signal *or any other signal* to a control input does that signal *function* as a control signal. Not before. The ACT dictum lends clarity by providing a simple means to determine the functions of any signal. What is that ACT dictum again?

The functions of a signal are not determined a priori by inspecting its signal characteristics, but by noting the functions of inputs to which it is connected. [For future reference, see *a priori* in the Learner's Glossary.]

Let's take this to extremes to emphasize the point. Even a physical keyboard shouldn't be called a "controller" prior to connecting at least one of its signal outputs to some *control* input! A keyboard certainly *is* a keyboard, and it is capable of generating signals, but we don't know how it

might *function* until connections of its signal outputs to specific audio, control, and/or timing (ACT) inputs are made. This is not "conventional wisdom" in the field, it is simply better wisdom.

To extend this idea that *signal* functions are determined solely by connections to ACT inputs, it necessarily follows that the functions of *modules* must be categorized the same way. That is, the *signal* flowing out of a module takes on the function of the ACT input to which it is *ultimately* connected. [A signal may pass through various processors on its way to its ultimate, or final ACT input destination.] Therefore:

Any module functions as an audio, control, and/or timing module solely due to its ACT connections.

As Mr. Spock of Star Trek television fame would say, "it is illogical" to call a keyboard a *controller* when it is just hanging around in outer space, not yet connected to any ACT input. So, the ACT dictum also defines how any *module* functions–as well as how any signal functions–by where it is connected. The astute sound designer realizes that any module can function any possible way (ACT), thereby enriching modular connection possibilities. Those who have categorized the functions of modules *a priori,* prior to connection, will not enjoy this expansive mindset.

Previously, we showed that a constant can function as an *audio* signal, making speakers "go bump in the night." Then we showed how a constant can function as a *control*

signal, controlling the gain of a signal controlled amplifier (SCA). Next, a context in which our constant functions as a *timing* signal. Then we'll have our ACT together: *audio, control, timing*–the ways any signal can function in a modular sound synthesis system.

We intuitively expect *timing* to indicate the *now,* or alternatively, the *not now* of some event or sequence of events. This *binary*, or twofold discrete nature of timing seems implicit, and this idea is reinforced by the fact that certain inputs on modules in a modular sound synthesis system *do respond* that way. A *timing input* on a module typically exhibits this *discrete,* or *threshold* response that embodies the generic *binary* form high-low. A timing input *threshold* is set to a specific value, or magnitude, whether by designer or user. *If* the magnitude of the signal you have connected to that *timing* input *exceeds* the existing threshold value, this situation *then* causes that input to respond by executing its "high" status. Typically, a *transition* to this high status tells that module to execute or initiate some action. Following this initial input condition, *if* the connected timing signal's magnitude subsequently does *not* exceed the threshold value, this situation *then* causes a *transition* to the "low" status, which also may execute or initiate some action. Note the use of the words "if" and "then," a tip-off that we're discussing a *conditional* operation. More on such *logical* operations later.

Speaking of logic, it is important to note that the low threshold status, which might be interpreted as a *logical no* (zero) by some, does *not necessarily* mean that "nothing"

or “no action” or the “opposite action” occurs, compared to the high status. It is misleading to think of the discrete high-low timing input status as “on” versus “off.” One or *both* high-low states might be used to *initiate* an action or enable a condition. High versus low *signal* levels at a timing input do *not necessarily* act as bipolar opposites (on versus off), in terms of how the affected module *responds*. Perhaps high causes *this* action, and low causes *that.* Nevertheless, high-low timing input states are indeed *complementary,* and having a *pair* does seem necessary, for without the low we cannot have a transition to the high, and the converse. This is a deep and true saying, grasshopper. (Maybe. It seems true. Or, it may simply reflect Western infatuation with *dualism.* Or perhaps, *duelism* in westerns–with apologies to Tom Mix, Gene Autry, Roy Rogers, John “Duke” Wayne, David Carradine, Clint Eastwood, Johnny Depp, et al.)

Of course, as we’ve seen, *any* signal connected to a timing input, regardless of that signal’s characteristics, *functions* as a timing signal, as per the ACT dictum. However, the timing signal *itself* is *not necessarily* required to have a binary, or two-state *signal characteristic.* This despite the fact that a timing input exhibits such a binary *response* to *any* input signal connected. If you imagine that a *gate input,* or timing input on any module, *dictates* that you connect a two-state “gate” signal, then rethink. That scenario is only an unfortunate cliché. A timing signal need *not necessarily* look like a discrete *step* signal or constant–a characteristic only seemingly implied (for some) by a typical timing input’s *threshold* response. Just as signal

characteristics do not dictate signal functions, input response does not dictate the characteristics of the signal that can be connected. For example, the continuously varying output of a microphone (mic, pronounced like "Mike") functions as a timing signal when connected to any timing input, exactly as the ACT dictum reveals. An imaginative sound designer will *make art* due to this realization, even though this suggested connection probably would not occur to those who restrict possibilities due to convergent thinking when categorizing functions of modules: a microphone as a timing device, imagine that! So, input *response* does not dictate the characteristics of the input signal chosen. This is an idea worth repeating:

Input response does not dictate the characteristics of the signal that might be connected to that input in a modular sound synthesis system.

This principle is illustrated by the suggested connection of a *continuous* microphone output signal to a timing input that has a *non-continuous*, or *discrete* (binary) response. The signal connected to a particular input is *not necessarily* required to have signal characteristics that conform in some simple-minded way to that input's *response.* If the only signal you ever connect to a *gate input* is the *gate output* signal from a keyboard, then your sound designs will be limited accordingly.

Take note that all inputs that have a *threshold* response are *not necessarily* timing inputs. For example, the inputs on *logic gates* (AND, NAND, OR, NOR, etc.) are threshold

(discrete) response inputs, but are *not necessarily* timing inputs. We classify logic gates as signal processors, and as such they can be audio, control, and/or timing modules as per the ACT dictum. That is, the *signal output* of a logic gate can be connected to *any* ACT input, not only to another logic gate input. To further explain, the *ultimate* input destination for a logic gate in a modular sound synthesis system is likely some ACT input. Therefore, a logic gate–like any module, can function as an audio, control, and/or timing module.

However, any *threshold response* type input does *operate* or *respond* the same way in any context, regardless of the role it plays in the module on which it appears. In math terms, *threshold response* is about whether an input signal value is greater than some predetermined threshold value–or not. Threshold response implicitly involves a logical, or *conditional* decision. Furthermore, all threshold response input *systems* or networks are *not necessarily* restricted to having a binary response. One might create, by connecting a number of modules, a threshold response input or *network* of inputs that embody several *different* discrete threshold values, not just a single threshold.

The preceding discussion points out the importance of making distinctions about the *response* and *function* of an input, independent from the signal *characteristics* of the connected signal. We remain free to connect *any* output to *any* input in a modular system, even though this freedom inevitably yields some connections that seem sonically or musically barren. Nevertheless, it remains important to

categorize signal and module functions using a powerful schema (ACT), and to know how inputs actually respond– rather than relying on a repertoire based on "what you're supposed to connect to what to get a certain effect." Deep understanding frees thinking, allowing improbable options to be explored. In sound design, creativity is often about the improbable patch, and the unlikely (signal) path.

Now, to illustrate that a constant can *function* as a *timing* signal, let's introduce another module, the *envelope generator (EG).* We'll focus on the *timing input* found on a standard envelope generator (EG). To set the stage, we connect the *signal output* of this envelope generator (EG) to the control input of the SCA, in place of the constant we first connected. Envelope generator (EG) signal output can then *potentially* control the gain of the SCA. Potentially? Yes, an envelope generator must be prompted, or told *when* to output its signal, as we shall see. This is consistent with the "now" or "not now" description of *timing* that we first discussed.

The typical envelope generator (EG) module has a *timing input,* variously called *gate input, trigger input,* or simply *gate.* Any sufficiently large positive (+) signal connected to this EG *gate input* causes that envelope generator to *start* producing the *first* (Attack) segment of the signal produced. Following this initial high (+) *gate input* condition, at any time the value (for example, (0) zero or (–) negative) of the connected (timing) signal does *not* exceed the threshold value, the EG is thereby signaled to *start* producing the *final* (Release) segment of its signal, in response to this

timing input *low* signal condition. In this context, this *high* versus *low* description of EG module response to the timing signal connected to the EG gate input seems more appropriate than “on” versus “off,” as mentioned previously. The envelope generator responds to *both* of its gate input high-low conditions by *starting* a particular envelope segment. This is an example where *both* high and low timing input states facilitate an “on,” or “start” condition. The initial transition to the high condition at the *gate input* enables the envelope generator to subsequently respond to any ensuing transition back to the original low condition at the input. There might be any number of intervening envelope segments generated by the EG between the first (Attack) and last (Release) segments, at the designer’s discretion. [For future reference, see the *ADSR* (envelope generator) entry in the Learner’s Glossary.]

When this signal is connected to an SCA *control* input, it can increase SCA *gain* dynamically. [This SCA responds to positive (+) *control* signals only. For future reference, compare *multiplier, two quadrant* (SCA), with *multiplier, four quadrant* in the Learner’s Glossary.] In analog terms, the signal produced by an envelope generator (EG) might be thought of as unipolar positive (+ only) fluctuating direct current (DC), with a magnitude that changes relatively slowly over time–similar to articulations of notes heard in music. In the current patch, this positive-only (+) envelope signal allows us to hear the noise generator (NG), as did our previous connection of a positive (+) constant to the SCA control input. This particular SCA is processing an

audio signal, because its *signal output* is connected to an audio monitor.

In a previous patch, we were able to selectively start and stop sound–to *articulate* noise, by alternately connecting, then disconnecting a positive (+) constant directly to the control input of an SCA. So, why do we connect this envelope generator (EG) output to the control input of the SCA to facilitate such articulation, rather than the constant we used originally? Because the signal produced by an envelope generator can be *programmed* flexibly to provide various shapes that are more interesting than the discrete high-low *step* shape we provided when connecting and disconnecting the constant. The general capability provided by the envelope generator (EG) to subtly *alter* various signals *in time* is invaluable.

Envelope is defined historically as *the dynamic changes of the intensity of a sound over time.* Such changes are called *articulation* in music, and are heard as changes of *loudness,* a sonic attribute. As we see here, it takes *both* envelope generator (EG) *and* SCA modules to create an *envelope,* as defined traditionally, even though we call only one of these modules an *envelope generator*. In the present context, the envelope generator controls the *amplitude*, or size of the signal being processed by the SCA, and this signal is an audio signal. This configuration conforms to the traditional definition of "envelope."

However, the signal produced by an envelope generator might be connected to inputs on modules other than the

SCA, in order to dynamically control signal characteristics *other* than amplitude, and signals that are *not necessarily* audio signals. That is, the concept of "envelope" has been broadened by sound designers to include use of the signal produced by an envelope generator (EG) for dynamic control of a variety of signal characteristics other than the *amplitude* of an audio signal. Such uses of EG output are illustrated in later sections.

Terms in the preceding paragraph such as *loudness, intensity,* and *amplitude* are closely related, but each has an appropriate context. *Loudness* is a subjective (personal) evaluation of something we actually hear. Our judgment about loudness is largely influenced by the power of the sound, which can be measured objectively using scientific, or "test" instruments. However, loudness perception also depends *greatly* on the *frequency* of the sound–what we might hear as *pitch*. Obviously, sounds near the upper and lower frequency *limits* of our hearing don't sound loud at all, even when they are quite powerful. *Intensity* is related to the power of a signal and can be quantified objectively in decibels (dB). *Amplitude* is another objective signal characteristic, but a particular signal may or may not relate to sound, depending on how that signal *functions* (as per the ACT dictum). The amplitude of an *audio* signal is related to the intensity of the sound it creates and is therefore somewhat related to the loudness we *might* hear. (Again, our hearing has a limited frequency span). On the other hand, household AC (electrical energy) is a signal that has an *amplitude* (120 or 240 volts USA), but AC is definitely *not* intended to be an *audio* signal, so AC

amplitude does not correlate with loudness. Amplitude is an objective signal characteristic that pertains to many kinds of signals, while loudness is an individual, subjective judgment that pertains to an audio signal that falls within the frequency span of human hearing. *Loudness* is a sonic attribute; *amplitude* and *intensity* are signal characteristics. These terms are not equivalent, and they should not be used interchangeably. Unfortunately, the term *intensity* is often used loosely to describe either objective or subjective elements of a sound. It may not be reasonable to expect newcomers to make such fine distinctions about terminology immediately, but it seems reasonable to alert all readers to the desirability of needing to do so *eventually*.

To make such distinctions in terminology is not academic or pedantic. It simply shows that you know what you're talking about, and that you realize that making such fine distinctions about terminology sharpens understanding of important principles. The size of the Learner's Glossary in this publication underlines the importance the authors place on learning the technical language that music technologists use to communicate. Terms such as "nut, frog," and "bridge" are *not necessarily* interpreted the same way by a violinist, biologist, or a civil engineer. Artists, particularly music technologists, are not exempt from needing to understand relevant terminology. Once you learn the basics, don't just browse this publication's Learner's Glossary, *graze* in it. [However, we don't recommend delving into the Learner's Glossary first thing. Technical terms are typically defined using other technical terms, and this circularity might be frustrating. Understanding any

topic is based on creating a *web* of associations. When basic terms are understood, exploring a well written glossary enhances sophistication and increases technical vocabulary. Basics first. Eventually, you should also explore various other narrowly focused glossaries dealing with music, computer technology, communications media, acoustics, audio engineering, electronics, etc. The scope of our Learner's Glossary is only somewhat broader than the otherwise narrow focus of this publication, which is about modules and their use to synthesize sound.]

Let's complete our discussion about how a constant can function in all possible guises: audio, control, timing (ACT). In the most recent patch, the constant functions as a *timing* signal that tells the envelope generator (EG) *when* to start producing its Attack and Release signal segment(s). *Any* signal connected to the same EG *gate input* might tell the envelope generator *when* to initiate such actions. Envelope generator (EG) output in this patch controls the gain of the signal controlled amplifier (SCA) dynamically. The SCA alters noise signal amplitude, determined by the actual features of the EG output signal, thereby allowing the noise signal to pass through the SCA. Finally, the audio monitor *transduces* (converts) electronic signals into sounds, which causes the loudness changes we perceive. In this patch, we can shape the *envelope* (change of the intensity of a sound over time) by programming specific envelope generator (EG) module parameters such as Attack time and Release time. Programming a *parameter* (EG Attack time, EG Release time, oscillator frequency, filter cutoff frequency, etc.) comprises setting a (virtual)

numerical value or (physical) knob or slide pot setting– typically also calibrated numerically, to determine values or conditions that relate to the signal generated or processed by a module. We use the envelope generator (EG) to dynamically control the amplitude of the noise signal being processed by the SCA. In this context, dynamic signal amplitude is heard as corresponding changes of loudness– articulation of sound, an *envelope* as classically defined.

In this patch, noise is an audio (A) signal, EG output is a control (C) signal, and the constant is a timing (T) signal. The SCA processes the noise signal. The SCA is an audio signal processor solely because the SCA *signal output* is connected to an audio *input.* The envelope generator (EG) functions as a control device due solely to its connection to a control *input*. And the constant is a timing signal due solely due to its connection to a timing *input.* Inspect the respective audio, control, timing (ACT) *input* connections for the *signal output* of each module in a patch to understand how each module *functions.* Noise generator signal output is eventually connected to the audio monitor input, after passing through the SCA. An important point:

The final input, or "ultimate" destination in a signal path determines a signal's ACT function.

So, the noise generator (NG) is part of the audio signal path (ASP), as is the SCA. The envelope generator (EG) constitutes a simple control signal path (CSP). The constant functions as the only signal in a timing signal path (TSP).

To understand such *patches,* or interconnections of modules, it is critical to understand not only signal *functions,* but also signal *flow* between, within, and (in some cases) through modules. In our latest patch, the constant connected to the *timing* input (gate input) on the envelope generator (EG) does *not* pass through the EG module. Any signal connected to the EG *gate input* does *not* appear at the EG *signal output.* That is, the envelope generator (EG) module *does not process* the signal (constant in this example) connected to its *timing* (gate, gate input, gate in) input. First of all, an envelope generator is not a signal processor, it's a signal *generator,* as its name clearly indicates! A signal generator doesn't *process* signals, it *produces* them. The distinction between a generator and a processor can be made prior to making connections, by simply inspecting any module. If a module has a *signal input* via which a signal can potentially pass through that module to an associated *signal output*–it's a processor. A generator has no *signal input,* only a *signal output.* Presence or absence of a *control* input or *timing* input has no bearing on this differentiation between generator and processor. Generators and processors may or may not have control and/or timing inputs.

Such "signal, audio, control, timing" input naming is part of the *nomenclature,* or set of words and symbols used to name system features. Ideal nomenclature would provide information that is not *ambiguous*–having more than one possible meaning or interpretation. Use of the word "signal" throughout this publication to name particular inputs and most module outputs may seem unfortunate,

because it *is* ambiguous. It's certainly reasonable to wonder why *all* inputs in a modular system couldn't be considered, in some sense, "signal" inputs! Yes, technically all inputs *do* accept signals, but in our schema (and that of Bob Moog, who founded the modern synthesizer industry), the word "signal" is reserved for *particular* inputs and outputs, to help us better understand not signal *function,* but signal *flow*. Here's a rule that clarifies:

(1) A signal connected to a signal input can flow through that module to its associated signal output, from which it can be connected to any audio, control, timing (ACT) input.

We can't anticipate, much less accurately determine the *function* (ACT) of a signal connected to a module's *signal input*, until we know where that module's associated *signal output* is actually connected (to some ACT input). Such an input is of necessity given a "generic" name: *signal* input, as giving it an ACT designation *presumes* where the user (you) intend to connect it. Fortune telling, anyone? But at least we can confirm how any signal connected to a *signal* input can potentially *flow:* through the module.

Fortunately, unlike the situation with a *signal* input, the case of any signal connected to a *control* or *timing* input is not ambiguous at all. It is *explicit,* in terms of both *function (ACT)* and *flow.* A second rule for signal *flow* provides another important fact:

(2) A signal connected to a control or timing input does not flow out of that module; a control or timing input does not

have an associated signal output (or "control" or "timing" output!) Outputs can have no ACT designation, because we don't know where an output might be connected.

A signal connected to a control or timing input works *within* that module, but that signal doesn't subsequently flow *out* of the module. A signal connected to a *control input* or a *timing input* does not flow out of an associated *signal output*. Therefore, its *function* is explicit–control or timing.

Confusion about signal *flow* within a module might arise because a particular module has timing and/or control inputs, as well as a signal input. For instance, a noise generator (NG) has no inputs at all, signal, control, or timing. Of course, it must have a signal output–all modules do. Signal flow is obvious in the case of a noise generator (NG), because there really is no choice, and therefore no confusion.

However, a signal controlled amplifier (SCA), or *two quadrant multiplier* has a signal input *and* a control input. Multiplication requires two numbers, so an SCA has two inputs. The question arises as to how signals connected to these two SCA inputs relate to each other, and to the single SCA *signal output.* Here's how: the *control* signal affects the inner workings of the SCA module, causing the SCA to process the signal connected to SCA *signal* input in a particular way. However, the signal connected to the SCA control input does *not* "mix," or *sum* (±) with the signal connected to the SCA signal input. Therefore, the control

signal does *not* appear at SCA *signal output.* The *effect* of a control signal may be quite apparent by ear, but the control signal itself does not appear at the signal output of the SCA to which it is connected. This creates another rule for signal flow:

(3) A signal connected to a control or timing input does not "mix," or sum with the signal flowing out of that module's signal output.

This rule pertains to control and timing inputs found on all processors and generators. Control and timing signals work *within* the modules to which they are connected.

To give an analogy in terms of musical instruments, each string on a violin is potentially an audio signal generator. The finger pressed on the string and wiggling back and forth to make *vibrato* is a control signal. However, we don't *hear* this control signal as such–it does *control* string pitch, but the vibrating string is the *signal output* we hear. We hear the *effect* of this control signal (finger) due to its "connection" to the "control input" of the audio signal (vibrating string).

To merge concepts and recap the principles of signal function and flow in a modular sound synthesis system:

(1) Outputs are connected to inputs.

(2) A signal connected to a signal input can flow out of its associated signal output, and signal function(s) are defined

by the ACT (audio, control, timing) input(s) to which that output signal is ultimately connected;

(3) ACT (audio, control, timing) inputs are "ultimate" destinations for signals, as all ACT inputs lack associated signal outputs within the virtual system.

(4) In the case of audio (A) inputs, the connected signal flows out of the virtual system (ostensibly to any connected audio monitor). In the case of control (C) and timing (T) inputs, the connected signal stays within the module to which it is connected–does not mix with the output signal of that module.

Given rampantly inconsistent nomenclature in the field, experimenting with the inputs and outputs of modules to determine signal *function* and *flow* possibilities is clearly necessary. Having a schema for such experiments is critically important.

Signal function and flow are readily understood, given modest comprehension of the four (4) principles listed above. Reluctance to learn simple ideas causes persistent confusion for those who approach sound design using the *discovery,* or empirical method solely, without recourse to any clear intellectual schema. It's easier, and far more efficient use of your time, to learn how signals *function* and *flow,* rather than to imagine that endlessly twiddling knobs, with no understanding, will somehow magically make you "more creative." There is plenty of room for creativity by discovery, even for *serendipity* (fortunate accidental or

chance discoveries, events, or outcomes), *after* finding out what you are actually doing with modules! Learning how signals function and flow is not the desired *end*–it's merely a *means* intended to facilitate your *artistic* creativity anywhere modules are found. (They're found everywhere, if you look!) You're poised to *recognize* serendipitous events and myriad creative possibilities when you have a schema firmly in mind–as any exploration of the history of invention confirms. It was the scientist Louis Pasteur who observed that ". . . in the fields of observation, chance favors only the prepared mind," which might translate to "good fortune and serendipity come to those who work hard for them." The 17th century French philosopher and mathematician René Descartes (reh nay´ day kart´) might have said, succinctly: "cogitate." Centuries later, the founders of the first digital computer manufacturing giant (IBM) would similarly implore: "THINK."

The preceding discussions also give rise to a categorization of modules, *without* consideration of their possible function(s) due to ACT connection(s), that we can now formalize:

A module is a (1) generator, (2) processor, or (3) terminal.

(1) A *generator* has a signal output, signals do not flow *through* a generator, they flow *out* of it.

(2) A *processor* has a signal input *and* a signal output; signals flow *through* a processor.

(3) A *terminal* is a module that has input(s) *or* output(s) used to route signals between virtual (computer-based) and external worlds.

A terminal is essentially a "transparent" *conduit,* or "pipeline" for signals. Signals flow through a terminal, but are *not necessarily* processed by the terminal, although some terminals provide simple gain (signal size) control. For example, an *audio bus* is a terminal that routes signals *out* of the system to a physical audio monitor. The reason such a terminal might be called an "audio" output, or *audio bus* is due to our assumption that this "audio" output of the system is always directly connected to an audio monitor. In reality, the possible function(s) of a signal on an "audio bus" are restricted only by convergent thinking. The so-called *audio output* of a system can carry a signal that you might choose to use as a control or timing signal elsewhere! You'll see an example when we discuss the *vocoder* (voice coder), where audio is *typically* used as a *control* signal. In the *taxonomy,* or classification system articulated above, at least the generator versus processor distinction is intuitive, due to the *intrinsic*, or basic and essential input and output (I/O) features of respective modules.

Take note that modules are categorized in this context essentially by signal *flow,* not by signal *function.* That is, modules "hanging in space" without being connected to anything, can be categorized as generators, processors, or terminals. However, modules "hanging in space" *can't* be categorized as "audio, control, or timing" (ACT) devices. We can discover a module's signal *flow* possibilities simply

by noting its input-output *features;* we can't possibly know how it *functions* until we connect its *signal output* to some ACT *input.* Generators, processors, terminals, module features, signal characteristics, signal functions, signal flow, input response, sonic attributes: related, but not to be confused with each other! This is the crux of understanding a modular sound synthesis system.

Similarly, categorization of signals as *periodic* or *aperiodic* depends on the *intrinsic* characteristics of such signals, which we discuss more fully later. Signal *characteristics* can be recognized when a *signal* "hangs in space," or remains unconnected to any input. It is the *functions* of signals and their associated modules that depend on ACT connections. During this introduction, you might say that we've been more interested in *inputs* (response, function, flow) than *outputs* (signal characteristics and associated module features). The all-too prevalent *output-oriented* teaching approach leads to self-limiting conclusions about possible signal functions. For example, any schema that categorizes a microphone as an "audio" signal generator (transducer), based on "conventional wisdom" due to its *output signal characteristics,* will tend to preclude that "module's" functional use (ACT) in any other way than audio. It wouldn't even occur to someone operating under such a schema to connect a microphone output to a *timing* or *control* input. Can a microphone be used creatively as a controller or timing device? You bet! It is better to realize that all signals in a modular system can function in *all* available ways (ACT), and ACT on this idea. ACT inputs define how a signal functions–*not* signal characteristics.

We'll turn our attention to signal characteristics (at module outputs) soon enough.

We've shown how a simple *step* signal such as a constant (bias) can *function* as an audio, control, or timing (ACT) signal. Similarly, we've explored some ways an oscillator might function. We've discovered how signals *flow* in a modular system. And, we have an inkling about how certain inputs *respond* to a signal as well–a system design feature typically not subject to our change. Some of the ideas expressed in this exposition presently occupy a *minority* position in the field, as shown by our review of currently available texts, and recollection of our own prior (more traditional) teaching methodologies. We could have sketched the trite traditional schema in a few paragraphs and moved on, assuming easy *recognition* on the part of even marginally experienced learners. The "elephant in the room," or *traditional* approach is found everywhere.

But, any experienced *teacher,* and any perceptive learner knows that being "told" something new is not as effective as being *shown,* and it takes more time, energy, and *ink* to show, rather than to simply tell you a different schema. That's what this exposition and a good bit of this publication are about–convincing you of the *power* of this schema, a way of thinking about modules. But even more powerful than any written narrative is the "doing" by any self-directed and *informed* learner, and that's why we've provided Sound Doilies, Progressive Patches, and other experiences you can *hear* as you play with parameter values of various modules. It's inevitable that we will turn

to the primacy of sound and sonic attributes, given our enterprise. However, for pedagogical purposes, it is not necessarily compelling that we start there.

This initial exposition focuses on signal *functions* and *flow.* However, to extend this understanding, we now take a sidelong glance at another signal *characteristic,* by revisiting our initial look at electrical energy. We began by considering the signal characteristics of direct current (DC). Now let's manipulate some DC steps (signal levels), also known as *constants,* in order to begin to understand alternating current (AC) signal characteristics. To facilitate this discussion, we need to introduce a signal characteristic known as polarity.

Polarity is represented in the virtual world by mathematical sign (±), one of the elements that defines a constant. In the case of real world electricity, direct current (DC) is a signal that flows in only *one* direction on an electrical conductor such as a copper wire. DC is *unipolar*–its signal is either positive (+) *or* negative (–). Although a unipolar waveform has only one polarity (+ or – sign), it may include zero (0) among its values, as zero (0) has no polarity. Zero (0) is the reference against which we determine polarity.

Alternating current (AC) flows on a conductor *alternately,* first in one direction, then the other. Alternating current (AC) therefore has *two* polarities, flowing positive (+) *then* negative (–), and so forth repetitively. Opposite directions of current flow are represented *alternately,* first by a positive (+), then a negative (–) sign, or the converse. AC is

bipolar–its signal has either a negative (–) or positive (+) polarity at any given moment, but not both polarities at the same time. Recall the (bipolar) real number line, with zero (0) at its *origin,* or center, with positive (+) and negative (–) numbers of increasing magnitude in opposite directions. (See above to review).

We really don't need a deep knowledge of electricity to understand signal polarity. Just remember that non-zero numbers representing signals in a virtual system have either a positive (+) or a negative (–) sign. (In cases where sign is not shown explicitly, assume a positive (+) polarity.) A unipolar signal has only one sign, positive (+) *or* negative (–). A bipolar waveform has signal values that cross *through*, and therefore include zero (0) in order to encompass *both* positive and negative (±) signs, (but not both polarities at the same *time*). Signal polarity is very important to a sound designer, as will become apparent a little later.

An unprocessed direct current (DC) signal is a unipolar *constant* in virtual terms, a constant whose sign (±) and magnitude we might change. Now let's imagine a DC signal, or constant that regularly and repetitively *does* change from one polarity (+) to the other (–), in the time of say, one (1) second. The qualifiers *regularly* and *repetitively* tell us that this signal is *periodic.* A classic *periodic* signal has a waveform, or *cycle* that occurs during a specific amount of time, and this cycle theoretically repeats endlessly. We have further specified that this imagined waveform completes its cycle in one (1) second,

with one polarity *transition* per each cycle. We might also say, one (1) *cycle* entails a single change from one polarity to the other, encompassing both (±) possibilities. Let the first value of our periodic signal be positive (+), then the ensuing value negative (–), and so forth *alternately.* Our thought experiment has transformed the unipolar positive (+) DC constant from a previous patch into a *bipolar* (±) periodic signal, featuring a level that alternately changes *polarity.* We now have some kind of bipolar (±) *alternating* signal that has two different levels, or *steps.* This signal is similar to, but not exactly like alternating current (AC), a form of electrical energy, as we shall see below.

Let's connect this bipolar (±) alternating step signal to the *signal input* of an active *audio* monitor. We shouldn't be surprised to hear a repetitive *series* of "bumps." Previously we connected and disconnected a *step signal* known as a *constant* to the same audio monitor to create similar individual "bumps." If this new *audio* signal's plus-then-minus (±) discrete polarity transitions (steps) alternate quite regularly, and we gradually increase the *frequency* (speed, or rate) of their occurrence, at some point this regularly repeating series of "bumps" is perceived as a *tone* that has a definite *pitch.* Perhaps you created a cardboard "motor" that was struck by your bicycle spokes when you were a kid and noticed how the *pitch* of such a "motor" gets higher the faster you travel. Spokes that "bump" the cardboard more frequently create more cycles per second, and consequently create a higher pitch.

Pitch is one of the sonic attributes of an audible signal whose waveform is periodic. (Once again, take note that *periodicity* is a signal characteristic, *not necessarily* a sonic attribute. *Pitch* is the sonic attribute–what you hear. *Pitch* and *periodic:* related words, but not synonyms! If one positive-negative (±) change in the alternating waveform we've constructed constitutes a cycle, the time that one (1) cycle occupies is known as that waveform's period. *Period,* a measurement of the *time* that *one cycle* of a periodic waveform occupies, is represented using the capital letter *T*. *Frequency* is the number of times such a repeating cycle, or period occurs during a standard unit of time–one (1) second for audio engineering and acoustics. Frequency is measured in *hertz* (Hz), a unit of measurement formerly known as cycles per second (cps). *Frequency,* the number of periodic waveform cycles per second, is represented by the lower case letter *f*. In a musical context, the frequency of the pitch we call "tuning A" (also designated A4) is 440 Hertz, that is: $f = 440$ Hz. A common reference tone in audio engineering is 1000 Hz.

Here's an analogy from the public library: a magazine or journal is classified as a *periodical*, because it is published repeatedly at a regular interval, or *period* of time–weekly, monthly, quarterly, etc. In the case of print publications, the unit of time for which one considers *periodicity* is normally one (1) *year*. A monthly magazine has a *period* of 1/12, or one twelfth of a year. So, the corresponding *frequency* for a monthly publication is 12, meaning twelve per year, literally 12/1 (the *reciprocal,* or *inverse* of 1/12). Period and frequency have a *reciprocal,* or *inverse* relationship

(12/1 and 1/12 are reciprocals of each other). In this context, *frequency (f)* is the number of times, or "cycles" this monthly magazine appears in the specified unit of time (t) of one (1) year. *Period (T)* is the interval of time that a single copy (one cycle) of a magazine or journal occupies, one (1) month in this example, therefore one twelfth (1/12) of a year. The publication date and subsequent revisions or editions of a *book* in this context are *aperiodic,* meaning *not periodic.* A book is not a periodical, because a book is not published (and/or revised) repeatedly at a regular time interval. It's just called a "book" at the public library. Periodicals define themselves due to their ongoing regular appearance. So, in music "tuning A" (A4) has a frequency of 440 Hz (cycles per second), and therefore a period of 1/440 seconds, which is about 0.002272 seconds, or approximately 2.3 milliseconds. A single cycle of a periodic waveform with a frequency of 440 Hz occupies an interval of time of about 2.3 milliseconds. You can make music without knowing such numerical relationships, but if you want to be a music technologist . . . you take an interest in such.

The frequency (f) and period (T) of a periodic waveform are *signal characteristics* we can understand without thinking about *sonic attributes* at all. For instance, a *microwave* signal can be used to transmit data using a *carrier* frequency (f) that is much higher than we can hear. A microwave carrier frequency can't be perceived as having a *pitch,* because its frequency is not within the frequency limits of human hearing. Frequency, period, and other signal characteristics do *not necessarily* relate to

(correlate with) sonic attributes–such as pitch. Signal characteristics embrace a general concept that pertains to many types of signals other than audio signals. On the other hand, sonic attributes pertain strictly to *sound.* Audio signals belong to a very limited subset within the world of signals.

A generator that produces several periodic signals with different waveforms is known as a *periodic function generator,* or due to popular usage, an *oscillator.* An "oscillating" electric fan cycles from side to side periodically, regularly and repetitively during a particular unit of time. In the patch presently under discussion, we have connected a one (1) hertz (Hz) *audio oscillator*–one that slowly bumps the audio monitor's speakers *periodically* (repeatedly and regularly) at a frequency (f) of one (1) Hz. When we increase this audio oscillator's frequency (f) to 440 Hz, meaning 440 cycles per second, we hear *tuning A*, the note "A4" in the octave above middle C on the piano. A constant with an appropriate magnitude (size) connected to this audio oscillator's *control* input provides this particular oscillator's frequency *tuning*. A positive (+) constant connected to an oscillator's *control* input raises oscillator frequency; a negative (–) constant lowers oscillator frequency. This oscillator is evidently a signal controlled oscillator (SCO).

Now let's use a second signal controlled oscillator (SCO), with capabilities identical to the first, but one that remains tuned to one (1) Hz, much lower than the audio oscillator we've presently tuned to 440 Hz. We could connect this

low frequency (1 Hz) SCO signal output to a *second* control input on the 440 Hz SCO, the *audio* oscillator. (Remember, we previously connected a constant to the *first* control input of our audio oscillator, for tuning purposes). The frequency of the periodic step signal (square waveform) audio oscillator is therefore now controlled by two (2) control signals: (1) the audio oscillator's "tuning" constant we connected previously; and (2) this newly connected low frequency (1 Hz) *control* oscillator.

This connection of two (2) control signals to a module's control inputs reveals an important point: *control signals sum.* When several signals are connected to a module's *control* inputs, whether to internal or external inputs, those control signals *add* to control the selected parameter of the module being controlled, which is oscillator frequency in this example.

Multiple signals connected to a module's control inputs sum algebraically (±)*, taking into account both positive and negative polarities.*

The 440 Hz "tuning" of a typical signal controlled oscillator (SCO) is determined by the value of the constant connected to one of its (internal) frequency control inputs. [Many virtual oscillator designs feature one or more tuning constants *built-in* to the module, "coarse" and "fine" tuning for example. That is, tuning constants may *not necessarily* be depicted as signals connected to control inputs on the oscillator, even though they actually are connected that way internally. There may be several *internal,* or built-in control

inputs on various modules typically represented as programmable numerical *constants,* which may be depicted as knobs or rectangular "numerical windows" on the module, rather than *external* control inputs capable of accepting *any* signal available in the system.]

The slow (1 Hz) *modulation*, or repetitive change of audio oscillator frequency is caused by the control oscillator we connected to a second (external) frequency control input on this audio oscillator. The control oscillator's signal causes the repetitive change of audio oscillator pitch, alternately from a note higher (+ positive level of the control oscillator signal) than tuning A (440 Hz), to a note lower (– negative level of the control oscillator signal) than tuning A. The particular waveform and low frequency of this control signal cause a variant of musical *trill,* a repetitive and regular change from one discrete pitch to another. The instantaneous (moment to moment) audio oscillator frequency we hear is caused by the instantaneous *sum* of its *two* control signals: (1) tuning constant; and (2) control oscillator. The *rate* of the pitch changes in our "trill" is determined by the *frequency* of the control oscillator. The frequency (rate) of this control oscillator is determined by the value of yet another constant connected to *its* control input.

Take note that this low frequency (1 Hz) *control* oscillator signal manifests itself in this patch as a *modulation rate,* not as a *pitch. Modulation* might be defined simply as a change–typically a repeating change, of some characteristic of a signal, frequency in this case. The modulation in this

example occurs due to this particular patching and is *heard* as repetitive discrete changes of *pitch*. In musical terms, we've programmed a form of *trill.* However, when we change the frequency of the control oscillator, this causes no change of the two *pitches* heard alternately that constitute the trill. Rather, the *speed* of the trill changes. This clearly illustrates that frequency change is *not necessarily* pitch change. Here, frequency is heard as modulation rate. We hope, at this point, it is becoming clear why we want you to "rethink" signal-sound correlates.

Words possibly complicate these relationships, and that's one reason we have block diagrams that depict connections of modules. You should learn to "read, write, and speak" this nonverbal "language" fluently. The overall *pitch* level of what we hear is largely due to the *frequency* of the audio oscillator, and this frequency is determined by this oscillator's connected (or built-in) frequency control constant. The modulation, or alternating higher and lower changes of audio oscillator *pitch* occur at a *rate,* or *speed* determined by the *frequency* of the control oscillator. The frequency of each oscillator is determined by *its* connected (or built-in) frequency control constant. In the case of the audio oscillator, a change of its *frequency* is heard as a change of the basic *pitch* level. In the case of the control oscillator, a change of its *frequency* is *not* heard as a change of the two pitches heard; it's heard as the *rate of modulation,* or speed at which the two pitches alternate.

Both oscillators have the same features: each generates a periodic signal, and the frequency of each oscillator is

controlled by a constant connected to its frequency *control* input. But these oscillators do not *function* identically, due to their respective connections to different kinds of ACT inputs. One becomes an audio device, the other becomes a control device, solely due to their respective connections. Both oscillators may have *features* completely in common, but each oscillator functions differently due to the specific ACT input to which we have connected its signal output. Note that the low frequency *control oscillator* also has a constant connected to its control input to determine *its* frequency. This completed patch provides independent *frequency* control over *audio* and *control* oscillators: *frequency* changes are perceived as *pitch* (audio) and *modulation (trill) rate* (control) respectively.

To recap our contrarian assertions about signal-sound correlates:

(1) Frequency is not necessarily pitch.

As just illustrated, oscillator frequency might be perceived as pitch in one context, and rate of modulation (trill rate) in another context. In the *audio* signal context, the signal-sound correlate that equates frequency with pitch makes fairly good sense. More on this later. In the *control* signal context, it does *not*–frequency is heard as a rate of modulation in the patch illustrated. Oscillator *frequency* is the *signal characteristic* of interest in *both* cases. But, in terms of sonic attributes, *frequency* is *not* heard as *pitch* in both cases. Note that the musical *interval* (distance between two discrete pitches) created by the modulation (trill) in our

patch doesn't change when we change the control oscillator's frequency–the *rate* of the trill changes. Note also that any change of signal *frequency* in this patch *does* relate to the temporal realm, or *time.* Clearly, both *pitch* and *modulation rate* involve time, and both are bound up with the concept of *frequency*–a count of some event that occurs in a standard unit of time (one second in this case). Our sense of pitch is based *primarily* (but not exclusively) on the frequency of a periodic audio signal. But, just as clearly, a change of signal frequency, as illustrated, is *not necessarily* heard as a change of *pitch* in a sound synthesis system. It depends on how the particular signal in a specific patch *functions* (audio, control, timing) as to how changes of its associated signal characteristics are actually *heard.*

(2) Waveform is not necessarily timbre (tone color).

As a result of our experiments, we now have a patch with two oscillators, one control–the other audio. This is all we need to explore oscillator waveforms, and how they are perceived. Let's add the capability that each oscillator can generate either one of two (2) possible waveforms, a *square* waveform with two discrete steps (as we constructed previously) *or* a *sine* waveform (smoothly changing waveform). In this patch that features oscillators with this enhanced waveform selection, we note that a change of *audio* oscillator waveform is indeed heard as a change of *timbre* (tone color). But a change of *control* oscillator waveform is *not* heard as a change of timbre. Rather it is heard in this context as a change of modulation *shape.* We can change between trill and some form of *vibrato* by

changing the modulation waveform. Clearly, waveform is *not necessarily* timbre, although waveform relates to signal shape in both cases.

(3) Amplitude is not necessarily loudness.

To illustrate this, we'll introduce a module called an attenuator into the preceding patch. An *attenuator* is a signal processor that reduces the *amplitude* (size) of the signal passing through it, without significantly altering any other signal characteristic. Unlike the SCA, an attenuator is a processor that typically provides a *static*, or fixed change of signal amplitude–it is not *signal controlled.* In the case of the *audio* input attenuator in our abridged patch, we hear that a change of signal *amplitude* is indeed heard as a change of *loudness.* But the *control* input attenuator, which processes the *control* oscillator signal's *amplitude,* causes changes in the size (amplitude) of the *musical interval* produced by the modulation ("trill" or "vibrato," depending on selected waveform)–not changes of loudness. Clearly, amplitude is *not necessarily* loudness, although *amplitude* relates to signal *size* in both cases. Once again, we see the importance of interpreting the resulting *sonic attribute* of any *signal characteristic* strictly in the context in which that particular signal presently *functions:* audio, control, or timing (ACT).

We've demonstrated that signal: (1) frequency is *not necessarily* pitch; (2) waveform is *not necessarily* timbre; and (3) amplitude is *not necessarily* loudness. So, if these demonstrations prove that those oft-repeated signal-sound

correlates alluded to in the opening paragraphs do *not necessarily* reliably predict what you actually will *hear* relative to signal characteristics, what does? The key to successful use of modules is to orient yourself first toward signal *characteristics, functions,* and *flow,* even though the natural impulse for most musicians is to think in terms of sound and its *attributes.* Sound is definitely the *ultimate* thing, but we question whether it should be the *first* thing, particularly when learning sound synthesis. Those who persist in defining signal characteristics solely in terms of *audio* (sonic attributes), leaning on standard signal-sound correlates, tend to limit their creativity with modules.

More-sophisticated discussions about signal-sound relationships in this publication will illustrate that, even for *audio* signals, standard signal-sound correlates are neither infallibly sensible nor particularly consistent. Human hearing response is complicated. Our ears don't work like laboratory instruments. Our hearing does not exhibit perfect *linearity* like the markings along the edge of an ordinary ruler. Elements of sound perception interact in complicated ways, due to the *nonlinear* (curved) responses of human hearing. We have "curvy" ears–not just on the outside, but on the inside as well. We'll have more to say about this later.

For now, we can be confident that the two oscillators in our recent patches are interchangeable periodic waveform generators that can produce identical (or different) signals. It was our choice to connect modules, and program their frequency and waveform settings as we did. Oscillators

with identical features can be connected to respective ACT inputs such that one oscillator functions as an audio signal generator, the other as a control signal generator. Which oscillator is selected as the audio oscillator or the control oscillator is your decision, because you will make such connections. That's the beauty of a modular system. You get to make the connections and use system resources as you see fit. If you have 10 oscillators, you could have 9 audio oscillators and 1 control oscillator, or 9 control oscillators and 1 audio oscillator. Or any other possible configuration. The artistic ramifications are extensive.

Could an oscillator function as a *timing* device? Yes, of course. Simply connect the oscillator's *signal output* to any *timing* input. That oscillator then functions as a *timing* device, as per the ACT dictum. For example, in a variation on a patch shown previously, we've replaced the constant connected to the *gate input* of the envelope generator (EG) with an oscillator. We've selected the oscillator previously tuned to a very low (1 Hz) frequency as the *timing* device. We therefore hear noise being enveloped (articulated) periodically (repetitively), starting each time this low frequency timing oscillator starts the positive (+) half of its bipolar (±) waveform, thereby initiating the *attack* (A) segment of the signal produced by the envelope generator. The *release* (R) segment produced by the envelope generator is initiated when the timing oscillator starts the non-positive (negative (–) or zero) part of its waveform. Because Release time is short in this example, we hear silence one half the time (during the negative part of the *timing* oscillator's period). Noise is therefore articulated

periodically, or repetitively and regularly, because the timing signal (oscillator) *is* periodic.

The envelope generator (EG) starts its Attack segment each time this timing oscillator makes a transition to the positive (+), or *high* state of its bipolar (±) oscillator waveform. The envelope generator (EG) starts its Release segment each time this timing oscillator makes a transition to zero (0) or negative (–), the *low* state of its bipolar (±) signal. The signal subsequently produced by the envelope generator (EG) *controls* the gain of the SCA. The SCA processes the noise generator (NG) signal connected to the SCA signal input. The SCA *intermittently* and *periodically* passes the noise signal connected to its signal input to its signal output, onward to the audio monitor where it is heard as noise. Such connections are neither mysterious nor difficult, but signal functions and signal flow must be analyzed systematically in order to find out how modules and signals in a patch interact.

We've illustrated how a constant can *function* as an audio, control, and timing device. We've used an oscillator as an audio, control, and timing device. So, what exactly is the function of a constant or an oscillator? Each is obviously an audio, control, and timing module, depending on the particular input(s) to which it is connected! Each of these signals could even function all three ways at the same time, given simultaneous connections of *signal output* to all three types of ACT (audio, control, timing) inputs. The functions of a constant, an oscillator, and any other module, and their

respective output signals are differentiated solely by the inputs to which their *signal outputs* are connected.

On the other hand, an oscillator–or any other module hanging around in outer space, does have some intrinsic *features* that don't depend on the ACT inputs to which it is connected. The ACT dictum doesn't explain everything we need to know. We'll explore such features fully when we focus on individual modules. At this point, we can say with confidence that an oscillator is a signal generator–not a signal processor. That is, an oscillator has no signal input. (It may or may not have one or more control and/or timing inputs.) We now know that an oscillator produces periodic signals, just as we have learned that a noise generator (NG) produces aperiodic signal(s). These kinds of distinctions among modules (generator, processor, terminal), and among signals (periodic, aperiodic) allow *classifications* that have meaning *prior* to connecting modules. However, only when a connection is presently made can we then determine how any module and its associated signal *functions.* When we look at nomenclature for *inputs,* with descriptive names such as *control* and *timing*, and the necessarily ambiguous name *signal,* we can then determine how signals *flow.* Signal characteristics, functions, and flow are critical elements that help us predict what kinds of sounds a patch might produce. This is what a *professional* in any field does: make accurate short-term predictions based on prior training, experience, and available data.

We've yet to fully describe the signal *characteristics* of alternating current (AC), although some hints about this

form of electrical energy may be suggesting themselves. Household AC has specific signal waveform, frequency, and amplitude characteristics. USA household AC has a sinusoidal (smoothly fluctuating (±) bipolar) periodic waveform with a frequency of 60 Hz, and nominal signal amplitudes of 120 or 240 volts. [For future reference, the Learner's Glossary defines *amplitude* in its various guises.]

In the field of audio engineering, an audio signal with an *amplitude* of 120 volts would be outrageously oversized. The reader should avoid connecting line current (AC) to the signal input of any audio amplifier–or any other module's input! In contrast, the "large" signal represented by zero (0) dB near the upper extremity of the volume unit (VU) meter found on various audio engineering equipment represents a signal with a voltage of approximately 0.775 volts. That's right, a large signal in audio engineering is *less* than one (1.0) volt. In the real world, size (signal amplitude) does matter. Just ask, and *trust* Godzilla on this–size matters. In virtual environments, signals work well with each other, and no connection within a virtual system should cause damage. However, when an external signal is introduced into a virtual system, watch its level in order to protect the system! And protect yourself in the real world, because even household AC must be dealt with using *extreme caution*–electricity can be deadly!

Routing unprocessed AC–an external signal, into any input of a piece of audio equipment would cause a colorful disaster: blue smoke–red fire engines. Even so, alternating current (AC) signal characteristics act as a useful *mnemonic*

(nuh mon´ik), or memory aid, that might help us recall the *general* categories of signal characteristics. Alternating current (AC) has specific *frequency, waveform,* and *amplitude* signal characteristics: 60 Hz (USA), sinusoidal, and 120/240 volts. (We will ignore the concept of *phase* for the time being.) The amplitude of AC is "stepped up" and "stepped down" by devices called transformers, depending on whether the AC is intended for a long distance high voltage transmission line, or is routed locally to your household, where voltage amplitudes are a *relatively* low 120 and/or 240 volts. Many signals of interest to audio engineers and sound designers have the same general *types* of signal characteristics as AC, but with radically different frequencies, waveforms, and amplitudes. Note that a "relatively low" signal amplitude in one context, such as electrical power engineering, can be hugely out of bounds in another context such as audio engineering. The concept of *amplitude* is very important. Recording engineers and sound designers spend much of their time adjusting signal amplitudes (using faders, pan pots, and other attenuators). An *attenuator* is a signal processor that reduces the size (amplitude, level, intensity, power) of a signal, ideally without significantly altering any other signal characteristic such as waveform, frequency, or phase.

Ordinarily, household AC does not function as an audio signal. But even here, some divergent thinking might be instructive. AC *can* become a prevalent part of an audio signal, as unwanted *hum.* In this case, a small amount of AC intrudes into the audio signal of interest, due to user oversights, equipment design deficiencies, or unavoidable

laws of physics. The nominal frequency of hum derives from local alternating current (AC), which is 60 Hz (USA) or 50 Hz (EU). However, real world signals such as AC are messy compared to the idealized signals depicted in textbooks. AC nominally has a sinusoidal (smoothly varying) waveform, meaning that AC should have only one frequency (60 Hz USA or 50 Hz EU). In reality, an AC waveform may also contain reasonably powerful *harmonics* at whole number multiples of its nominal AC frequency. That is, whole number multiples of 60 Hz (USA) such as 120 Hz, 180 Hz, 240 Hz, etc. may also be low amplitude components of hum. AC is an example of a signal whose possibilities have been restricted due to convergent thinking about how AC might function other than as a source of *power.* In fact, low amplitude AC not only inadvertently *does* function as a (typically unwanted) audio signal, but AC can also function as a control signal or a timing signal, even in fields such as audio engineering. We simply can't use raw, or un-attenuated alternating current (AC) as a signal we can *manipulate* in a modular sound synthesis *patch*, and we should *never* introduce AC into the audio signal paths of a recording console or guitar amplifier! This small exercise in divergent thinking is our parting shot for the moment, in this intentionally redundant prodding intended to make you realize that any signal *is* simply what it functions *as.* That is, function is based on *usage*–not signal characteristics. One can *learn* to think divergently. It's not a mysterious talent restricted to a gifted few. Adopting a schema that facilitates divergent thinking is a good start, so ACT!

We don't want to use raw AC as an audio signal, or any other kind of signal in sound synthesis, because of its extremely large amplitude. However, if we connect an oscillator module to an audio monitor, we can learn by experimenting with the AC "hum" frequency. This oscillator becomes an *audio* waveform generator due to this particular ACT connection. When this audio oscillator is tuned to 60 Hz and its *sine* waveform is selected, we hear essentially the (USA) hum frequency. [Caution! Your sound system may be incapable of reproducing a frequency this low! If this is the case, raise the frequency until you can hear it. Also, human ears are insensitive to such low frequency sine waves anyway, and we have a tendency to turn the monitor amp's "loudness" or "volume" control higher to better hear this waveform. Resist doing this. Keep the level low, for a low frequency sine waveform can stress your speakers if played continuously at a sufficiently high level. Keep it barely audible.]

Now, as we slowly turn this audio oscillator's frequency down *below* 20 Hz, this *audio* signal becomes *inaudible*. Just a moment ago, we had a signal whose *audibility* many would presume to be *the* critical and *self-defining* condition that determines how to properly categorize an *audio* signal. But this can't be the case! When we hear nothing at the lower frequency, then what? Does this signal cease to function as an audio signal simply because we changed its frequency, lower *or* higher? Not in our schema. We simply cannot hear this particular *audio* signal, a sine waveform, when it lies within the *infrasonic* frequency range, below 20 Hz. We can't hear *ultrasonic* audio signals either, those

above 20 kHz (20 000 Hz). This *sine* wave infrasonic audio signal is below the lower frequency limit (20 Hz) of normal human hearing. We have an *audio* signal that is *inaudible.* The signal does not cease to *function* as an audio signal as per the ACT dictum, because it remains connected to the *audio* signal input of an active audio monitor. This is an example of an audio signal that does *not* "go bump in the night" at very low (infrasonic) frequencies–it makes no sound at all. If the speaker cones are moving at all under these conditions (infrasonic sine wave audio signal), they are moving in and out slowly and smoothly, and no audible clicks or bumps should be heard. However, a *square* waveform, with its near-instantaneous changes of signal polarity (±) at the *same* infrasonic frequency, would create a slow series of bumps we *can* hear. We heard similar speaker bumps previously when we alternately connected and disconnected a constant to the signal input of an active audio monitor. Previously, we also heard a repetitive *series* of bumps when we connected a 1 Hz square wave oscillator to the audio monitor.

The *audio signal:* now we hear it, now we don't–does this make for a credible schema? Even though signal *audibility* might blink in and out of existence, a signal's ACT (audio, control, timing) *function* doesn't, certainly not due to simple changes of frequency or waveform. Better to say, a signal's function *shouldn't* do this if we want to have a rational, consistent schema that explains how signals function! The ACT dictum accounts for the *objective* side of how signals function. And, this objective world is the only thing a modular sound synthesis system "knows." The

response of the human ear is complicated–signal functions in a modular sound synthesis system are simple. Let's keep signal functions simple: ACT.

Of course, there is also a *subjective* side that relates to what human beings actually *hear*. When we produce a clearly *audible* signal, a profoundly deaf person would not actually *hear* the sound produced. Does such a signal that is clearly *audible* to most, cease to be an audio *signal* to others in such a context? Is a signal an audio signal for one person, and not for another? Should the definition of a signal's function be *subjective,* based on the sensitivity of a particular individual's hearing? Not in our schema. A technical definition or categorization should have an *objective* (measurable or clearly verifiable) basis. So, an audio signal is an audio signal whether it is *audible* or not. Again, the value of the simplicity of the ACT dictum:

A signal's function (e.g. audio) is defined solely by its connection to a particular (e.g. audio) ACT input (or inputs).

Any schema that defines signal functions based on signal characteristics, or by misguided appeals to the subjective nature of human hearing, is ultimately neither consistent nor useful. Signal functions are objective–not contingent upon the aural sensitivity of potential listeners. For certain, signal functions can't be predicted by simply measuring a signal *characteristic* such as *amplitude,* which is so glibly equated with *loudness*, a sonic attribute. An audio signal's *function* does not depend on something as subjective as

audibility, as demonstrated by this simple experiment with a so-called "sub-audio" sine wave. It's actually a *sub-audible,* or *infrasonic* sine wave. *Any signal* remains an audio signal while connected to an audio input, as per the ACT dictum. The necessity for, and utility of this rigid viewpoint will become apparent and *necessary* as we detail a variety of sound synthesis engines later.

Another observation about *audibility:* an experienced sound designer can anticipate what a patch will sound like without actually hearing it. Certainly, *initial* efforts at designing a sound can be done not only in splendid isolation, but in golden silence as well. This is akin to using our musical *inner ear* to "listen to" the sounds that music notation *silently* represents. Notation–for a literate musician, isn't about time signatures, notes, rests, and dynamic markings. It's about *mental* impressions that allow one to perceive inwardly and *inaudibly* the *sounds* that move in time as *represented* by music notation. Likewise, a patch–for the experienced sound designer–isn't about icons, connections, and knob settings. It too, is about *mental* impressions of *sounds*. We can learn to "read" the sounds of modular patches the same way we learn to read the sounds of music notation, through repeated experience. There is certainly a subjective element about such abilities, which means that these considerations don't play a role in defining how a particular signal *functions.* The worlds of objective signal characteristics, and subjective sonic attributes do not always correspond, or *correlate* neatly, as we stated up front. Again, this begs the question why one should begin *learning* sound synthesis as though this were the case.

But we still need a way to think about this great divide between the objective and the subjective when dealing with modules, therefore the ACT dictum:

The functions of a signal are determined solely by the types of ACT (audio, control, timing) inputs to which that signal is connected.

At any rate, we see (and hear) that audio signals do *not necessarily* fall within the frequency span 20 Hz-20 kHz, or what is popularly called the "audio window." Perhaps this so-called "audio" window might more profitably be called the *audible* window. *Audible window* terminology would properly focus on perception of *sound,* which for *homo sapiens* does usually fall within the well known 20 Hz-20 kHz frequency span. In a modular sound synthesis system, audio *signals* do occur outside these frequency limits, but there's not much to argue about with respect to actual *audibility,* as those frequency limits of normal human hearing are well established. However, those frequency limits actually refer to that which is *audible* for human beings–not to *audio* signals *per se*. Cats, elephants, and bats certainly have a different audible window than people. (We've talked to them about this, although it's been quite a strain on our ears, to say nothing of our voices.) An *audio* window should in principle be about signals–*audio* signals, not about the limits of *audibility,* human or otherwise. In this publication we'll try to remember to refer to that which is *heard* as falling within the *audible window.* An *audio signal* has a function that is defined by the ACT dictum,

and as we've illustrated briefly, an audio signal is *not necessarily* audible.

We've taken a brief journey into the land of signals. There is more to say, with progressively greater sophistication. For instance, representation of signals in the *frequency domain,* another way to look at signals, has not even been mentioned in this exposition. A particular frequency domain representation known as a *line spectrum* is presented when we introduce various periodic function generator (oscillator) waveforms. This coupling provides a better understanding of both ideas.

We punctuate this discussion by showing some signals in the time domain, some of which might relate to sound. Note that the scale of the horizontal *time* base and the vertical *magnitude* axis can accommodate a wide range of possible signal types and ranges of values.

A note to the avid learner: look at these and other graphics carefully to see if they make sense. Reread any written passage thoroughly when its meaning is not apparent. Don't expect everything to be immediately apparent–some ideas are inherently more complicated than others. Look up pertinent technical terms in this publication's Learner's Glossary, even when you think you already know what they mean. Look up any other unfamiliar words in your dictionary. Listen to the sound examples, execute the tutorials, and play with the Sound Doilies in this publication. Seek other sources of information–find out about the literature in your field.

Play hard, but also–work hard. There is nothing inherently wrong with playing video games. But, it's one thing to play video games all day, turning yourself into a pudding. It's another to turn yourself into a professional who gets *paid* to create the sonic environments or music *for* those video games. If you aspire to success in any of the arenas of song writing, music synthesis, film scoring, sound design, and allied sonic arts, then you should *move forward and work without ceasing.* Talent and intelligence are certainly assets in such a quest, but our teaching experience persuades us that *sustained effort* often trumps genetic gifts.

This exposition has introduced many of the elements of the schema that is an *idée fixe* in this publication. You'll hear many of these ideas, including examples of ACT in action, more than once. Parts of our schema are conventional–parts are not. In particular, we avoid drawing simplistic parallels between signal characteristics and sonic attributes. That allegedly "simple" approach does get you off the ground more easily, relying on *a priori* categorizations of modules such as "sound sources, sound modifiers, controllers, etc." But that orientation eventually takes its toll, in terms of using modules creatively. The celebrated physicist Albert Einstein reportedly said that an idea should be expressed as simply as possible, but *"not simpler."* The ideas expressed in this publication are *not simpler* than what is practicable, and we haven't even flirted with creating some form of "instant" appeal that panders to people who think of themselves as "dummies" or "complete idiots," as some "educational" book titles would have it. For certain, we

haven't tried to *entertain* you throughout this holistic, plodding, and intentionally redundant introduction to the principles of signal functions and flow. People who need to be entertained at every juncture will populate the *audience.* That's OK, we *need* an audience. However, those who *move forward and work without ceasing* will position themselves to become part of the sonic arts *profession.*

Sophistication in sound design comes only after long hours of experimentation–*playing* with sound. Nobody else can do that work for you. However, at various places in this publication we've optimized opportunities for *constrained* experimentation with sound that virtually assure that learning will take place–but only if you exercise those learning tools diligently and thoughtfully. The mundane task of understanding how signals *function* has been facilitated by the ACT dictum, and a few principles of signal flow that we've made *not simpler* than practicable. Whether you choose to adopt our schema *in toto* is, as the Navajo say: "up to you." This schema happens to be learner-tested and proven effective over *many* years. And it's consistent across various synthesis engines.

The following pages provide an overview of the schema whose principles we will revisit throughout the publication, some elements of which we've only touched on thus far:

SIGNAL CHARACTERISTICS

A SIGNAL CONVEYS INFORMATION

A SIGNAL MAY BE PERIODIC OR APERIODIC

AN APERIODIC SIGNAL MAY BE RANDOM OR NOT RANDOM

SIGNAL CHARACTERISTICS ARE IDENTIFIED AND/OR MEASURED AT MODULE OUTPUTS

SIGNAL CHARACTERISTICS MAY INCLUDE FREQUENCY, PERIOD, WAVEFORM, AMPLITUDE, ENVELOPE, PHASE, POLARITY, AND SPECTRUM

A SIGNAL IS EITHER BIPOLAR (±), UNIPOLAR POSITIVE (+), OR UNIPOLAR NEGATIVE (–)

SIGNAL FUNCTIONS

SIGNAL FUNCTIONS ARE DEFINED SOLELY BY CONNECTIONS TO MODULE INPUTS

A SIGNAL MAY BE CONNECTED TO INPUTS TO HAVE AUDIO, CONTROL, AND/OR TIMING (ACT) FUNCTION(S)

ANY SIGNAL CONNECTED TO AN AUDIO INPUT FUNCTIONS AS AN AUDIO SIGNAL, REGARDLESS OF ITS AUDIBILITY

ANY SIGNAL CONNECTED TO A CONTROL INPUT FUNCTIONS AS A CONTROL SIGNAL

ANY SIGNAL CONNECTED TO A TIMING INPUT FUNCTIONS AS A TIMING SIGNAL

THE FUNCTION OF ANY SIGNAL CONNECTED TO A “SIGNAL” INPUT CANNOT BE DETERMINED UNTIL (ACT) CONNECTIONS WITH ITS ASSOCIATED “SIGNAL” OUTPUT ARE KNOWN

SIGNAL FLOW

OUTPUTS ARE DESIGNED TO BE CONNECTED TO INPUTS

SIGNAL OUTPUT APPEARS ON THE RIGHT SIDE OF AN ICON IN A CONVENTIONAL BLOCK DIAGRAM

SIGNAL INPUT APPEARS ON THE LEFT SIDE OF AN ICON IN A CONVENTIONAL BLOCK DIAGRAM

CONTROL INPUT(S) APPEAR ON THE BOTTOM OF AN ICON IN A CONVENTIONAL BLOCK DIAGRAM

TIMING INPUT(S) APPEAR ON THE BOTTOM OF A MODULE ANGLED AT 45 DEGREES IN THE AUTHORS' REPRESENTATION OF A CONVENTIONAL BLOCK DIAGRAM

A SIGNAL CONNECTED TO A TERMINAL MODULE FLOWS IN/OUT OF THE MODULAR SOUND SYNTHESIS SYSTEM FROM/TO THE EXTERNAL WORLD

A SIGNAL CONNECTED TO A CONTROL OR TIMING INPUT DOES NOT FLOW THROUGH THAT MODULE TO ITS SIGNAL OUTPUT

A SIGNAL CONNECTED TO A CONTROL OR TIMING INPUT DOES NOT MIX WITH THE SIGNAL AT THAT MODULE'S SIGNAL INPUT

SIGNALS CONNECTED TO THE CONTROL INPUTS (INTERNAL AND EXTERNAL) ON A MODULE SUM (ADD (±) ALGEBRAICALLY)

MODULES

A MODULE IS EITHER A GENERATOR, PROCESSOR, OR A TERMINAL

ALL GENERATORS AND PROCESSORS HAVE A SIGNAL OUTPUT

ALL PROCESSORS HAVE A SIGNAL INPUT AS WELL AS A SIGNAL OUTPUT

SOME MODULES HAVE CONTROL INPUTS

SOME MODULES HAVE TIMING INPUTS

MANY MODULES HAVE INTERNALLY CONNECTED CONTROL SIGNALS REPRESENTED AS CONSTANTS OR BIASES

A SIGNAL CONNECTED TO A TERMINAL MODULE FLOWS IN OR OUT OF THE MODULAR SOUND SYNTHESIS SYSTEM FROM OR TO THE EXTERNAL WORLD

INPUT RESPONSE

A PARTICULAR INPUT MAY RESPOND TO A CONNECTED SIGNAL IN EITHER A DISCRETE (TIMING-THRESHOLD DETECTOR) OR CONTINUOUS (CONTROL, AUDIO) MANNER

AN INPUT WITH A CONTINUOUS RESPONSE MAY BE SENSITIVE TO EITHER UNIPOLAR POSITIVE (+ ONLY), UNIPOLAR NEGATIVE (– ONLY), OR BIPOLAR SIGNALS (ALTERNATELY (±) PLUS THEN MINUS)

A comprehensive understanding of all of these schema elements is not critical at the moment. In fact, this list is not even intended to be exhaustive. We've introduced a few key ideas, a lot of terms, and an alphabet soup of acronyms (MIDI, NG, SCA, SCO, EG, ACT, etc). For now, it should be sufficient to recall the four (4) critical principles of signal behavior in a modular sound synthesis system:

(1) The functions of any signal are determined solely by the types of ACT (audio, control, timing) inputs to which that signal is connected.

(2) ACT inputs are ultimate destinations for signals. A signal connected to an ACT input: audio (A), control (C), or timing (T), does not flow out of that module's signal output.

(3) A signal input is not *an ultimate destination for signals. A signal connected to a signal input can potentially flow out of that module's associated signal output, and its signal functions are subsequently defined by the types of ACT (audio, control, timing) inputs to which that output signal is ultimately connected.*

(4) Control signals sum. Multiple signals connected to a module's internal and external control (C) inputs add algebraically (±)*, taking into account both positive and negative signal polarities.*

Focus on these four principles of signal behavior during your early endeavors, and you'll fare quite well. Also, it

might make sense to work through other parts of this publication before returning to the *extended* list of schema elements above. This will reinforce how much you will have learned in the meanwhile. You can eventually absorb the complete schema for modular sound synthesis by: reading this publication thoroughly, exercising the included Progressive Patches Tutorial, Sound Doilies, and other interactive elements; analyzing and modifying others' modular patches; independently designing your own patches from scratch; reading literature from the field; and above all–by *playing with sound.* The road to success is *not necessarily* paved with efforts toward rote memorization of basic principles–experimentation is required.

The stage is now set for you to *move forward and work without ceasing.* Again, as the Navajo say (more-complete saying, loosely translated): "It's up to you. Nobody else can do it for you."

Notes

Mathews, Max, and F. R. Moore. 1970. "GROOVE–a program to compose, edit, and store functions of time." Communications of the Association for Computing Machinery 13(12): 1.

Reiner, Rob, et al. 1984. This is Spinal Tap. 1984.