

Musical Signal Processing with LabVIEW – Analog Synthesis and Modular Synthesizers

By:
Ed Doering

Musical Signal Processing with LabVIEW – Analog Synthesis and Modular Synthesizers

By:

Ed Doering

Online:

< <http://cnx.org/content/col10480/1.1/> >

C O N N E X I O N S

Rice University, Houston, Texas

This selection and arrangement of content as a collection is copyrighted by Ed Doering. It is licensed under the Creative Commons Attribution 2.0 license (<http://creativecommons.org/licenses/by/2.0/>).

Collection structure revised: November 7, 2007

PDF generated: February 4, 2011

For copyright and attribution information for the modules contained in this collection, see p. 12.

Table of Contents

1 Analog Synthesis Modules	1
2 [mini-project] Compose a piece of music using analog synthesizer techniques	7
Index	11
Attributions	12

Chapter 1

Analog Synthesis Modules¹

1.1 Introduction

Analog synthesizers dominated music synthesis technology throughout all but the last 15 years of the 20th century. Early synthesizers were based on vacuum tubes or other electro-mechanical devices, and transistor technology entered the music scene during the early 1960s. Analog synthesizers produce sound waveforms as **continuous** voltages. Oscillators produce basic waveforms such as sinusoids, square waves, and triangle waves, much like a function generator in the electronics laboratory. These waveforms are shaped by time-varying amplifiers to emulate the characteristics of physical instruments, e.g., loud at the beginning transient of a note, softer during the sustained portion of the note.

You probably know that synthesizers defined many of the pop music styles of the 1970s. Watch (and listen!) to the screencast video in Figure 1.1 to learn more about some of the common synthesizer techniques. A real-time graphical signal analyzer is used to visualize the sounds so that you can better understand what you hear.

Image not finished

Figure 1.1: [video] Examples of analog synthesizer sounds, including visualization of waveform and frequency spectra

The history of electronic synthesizers is really fascinating. In particular, the following sites form an excellent starting point:

- SynthMuseum.com²
- EMF Institute³ – Follow the link for "The Big Timeline"
- 120 Years of Electronic Music⁴ – Look for the entries under 1960

¹This content is available online at <<http://cnx.org/content/m15442/1.2/>>.

²<http://www.synthmuseum.com>

³<http://emfinstitute.emf.org/>

⁴http://www.obsolete.com/120_years

1.2 Analog Synthesizer Modules

Everything about an analog synthesizer is analog! For example, a keyboard-based synthesizer uses a **control voltage (CV)** to change the frequency of the oscillator; the oscillator is therefore called a **voltage-controlled oscillator** or **VCO**. The time-varying gain of an amplifier is also controlled by a CV, so the amplifier is called a **voltage-controlled amplifier** or **VCA**. The VCO and VCA are two of many types of synthesizer **modules** that can be interconnected (or **patched** together) in many different ways.

Take a look at the video in Figure 1.2 to find out why interconnected modules are called **patches**, and to learn how to put together a simple patch involving a VCO, VCA, **envelope generator**, and keyboard controller.

Image not finished

Figure 1.2: [video] Origins of the term "patch", and simple example of an analog synthesizer patch using a VCO, VCA, envelope generator, and keyboard

Analog synthesizer modules can be grouped into four categories: **sources**, **processors**, **envelope generators**, and **controllers**; each of these is discussed in detail in the following sections.

1.2.1 Sources

Signal sources include the **VCO** and the noise generator. View the video in Figure 1.3 to learn more, then quiz yourself to check your understanding.

Image not finished

Figure 1.3: [video] VCO and noise generator signal sources

Exercise 1.1

The amplitude of a VCO's waveform can be adjusted (true or false).

(Solution on p. 5.)

Exercise 1.2

How does a VCO interpret its control voltage to produce a desired frequency?

(Solution on p. 5.)

Exercise 1.3

Which VCO has the richest harmonic content?

(Solution on p. 5.)

1.2.2 Processors

Signal processors include the **VCA** and the **voltage-controlled filter (VCF)**. View the video in Figure 1.4 to learn more, then quiz yourself to check your understanding.

Image not finished

Figure 1.4: [video] VCA and VCF signal processors

Exercise 1.4

How does a VCA interpret its control voltage to produce a desired gain?

(Solution on p. 5.)

Exercise 1.5

What types of filter functions can be implemented by a VCF?

(Solution on p. 5.)

Exercise 1.6

What VCF filter parameters can be adjusted by control voltages?

(Solution on p. 5.)

1.2.3 Envelope Generators

An **envelope generator** creates a **CV** to operate other voltage-controlled modules such as the VCA and VCF. View the video in Figure 1.5 to learn more about envelope generators, in particular why they are usually called an **ADSR**.

Image not finished

Figure 1.5: [video] Envelope generators, especially the ADSR-style envelope generator

Exercise 1.7

What is the normal (un-triggered) output of an envelope generator?

(Solution on p. 5.)

Exercise 1.8

What does the acronym **ADSR** mean?

(Solution on p. 5.)

Exercise 1.9

Why is the exponential shape used for envelope generators?

(Solution on p. 5.)

1.2.4 Controllers

A **controller** creates a control voltage (CV) to operate other voltage-controlled modules such as the VCA and VCF. An **interactive controller** offers the musician direct and immediate control of the sound, such as a keyboard, knob, or slider. A **programmed controller** generates a control voltage in some pre-defined way, such as a **low-frequency oscillator (LFO)** to produce vibrato, and a sequencer to produce a repeating pattern of control voltages for the VCO. View the video in Figure 1.6 to learn more, and then quiz yourself to check your understanding.

Image not finished

Figure 1.6: [video] Controllers including keyboard, knobs and sliders, low-frequency oscillator (LFO), and sequencer

Exercise 1.10

(Solution on p. 5.)

Which types of output voltages does a keyboard controller produce?

Exercise 1.11

(Solution on p. 5.)

What is the **portamento** effect, and how is it produced?

Exercise 1.12

(Solution on p. 5.)

Which device would a keyboardist use to automatically play a repeating pattern of notes?

Solutions to Exercises in Chapter 1

Solution to Exercise 1.1 (p. 2)

False

Solution to Exercise 1.2 (p. 2)

One octave per volt

Solution to Exercise 1.3 (p. 2)

Square wave has highest amplitude harmonics, but contains odd harmonics only; triangle wave has the most harmonics (even and odd)

Solution to Exercise 1.4 (p. 3)

Zero volts mean zero gain, one volt mean unit gain

Solution to Exercise 1.5 (p. 3)

Lowpass, highpass, bandpass, etc.

Solution to Exercise 1.6 (p. 3)

Corner (cutoff) frequency, bandwidth, resonance frequency

Solution to Exercise 1.7 (p. 3)

Zero

Solution to Exercise 1.8 (p. 3)

Attack - Decay - Sustain - Release

Solution to Exercise 1.9 (p. 3)

Easy to produce with RC-networks; matches behavior of real instruments

Solution to Exercise 1.10 (p. 4)

A control voltage to control the frequency of a VCO, and a gate voltage to control an envelope generator

Solution to Exercise 1.11 (p. 4)

A **portamento** is a continuous frequency transition from one note to the next; instead of producing a step-change in the control voltage connected to the VCO, the keyboard produces a continuously-varying voltage from the starting note to the ending note

Solution to Exercise 1.12 (p. 4)

Sequencer

Chapter 2

C

compose a piece of music using analog synthesizer techniques][mini-project] Compose a piece of music using analog synthesizer techniques¹

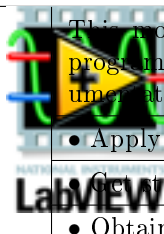
	<p>This module refers to LabVIEW, a software development environment that features a graphical programming language. Please see the LabVIEW QuickStart Guide² module for tutorials and documentation that will help you:</p> <ul style="list-style-type: none"> • Apply LabVIEW to Audio Signal Processing <p>Get started with LabVIEW</p> <ul style="list-style-type: none"> • Obtain a fully-functional evaluation edition of LabVIEW
--	--

Table 2.1

2.1 Objective

This project will give you the opportunity to begin designing sounds in LabVIEW. You will create two subVIs: one to implement an ADSR-style envelope generator and the other to create a multi-voice sound source. You will then create a top-level application VI to render a simple musical composition as an audio file.

2.2 Prerequisite Modules

If you have not done so already, please study the prerequisite module Analog Synthesis Modules (Chapter 1). If you are relatively new to LabVIEW, consider taking the course LabVIEW Techniques for Audio Signal Processing³, which provides the foundation you need to complete this mini-project activity, including working with arrays, creating subVIs, playing an array to the soundcard, and saving an array as a .wav sound file.

2.3 Deliverables

- All LabVIEW code that you develop (block diagrams and front panels)

¹This content is available online at <<http://cnx.org/content/m15443/1.2/>>.

²"NI LabVIEW Getting Started FAQ" <<http://cnx.org/content/m15428/latest/>>

³*Musical Signal Processing with LabVIEW – Programming Techniques for Audio Signal Processing*
<<http://cnx.org/content/col10440/latest/>>

- All generated sounds in .wav format
- Any plots or diagrams requested
- Summary write-up of your results

2.4 Part 1: ADSR Envelope

Create the subVI **ADSR.vi** to generate the ADSR-style envelope specified in Figure 2.1; recall that ADSR stands for "Attack Decay Sustain Release."

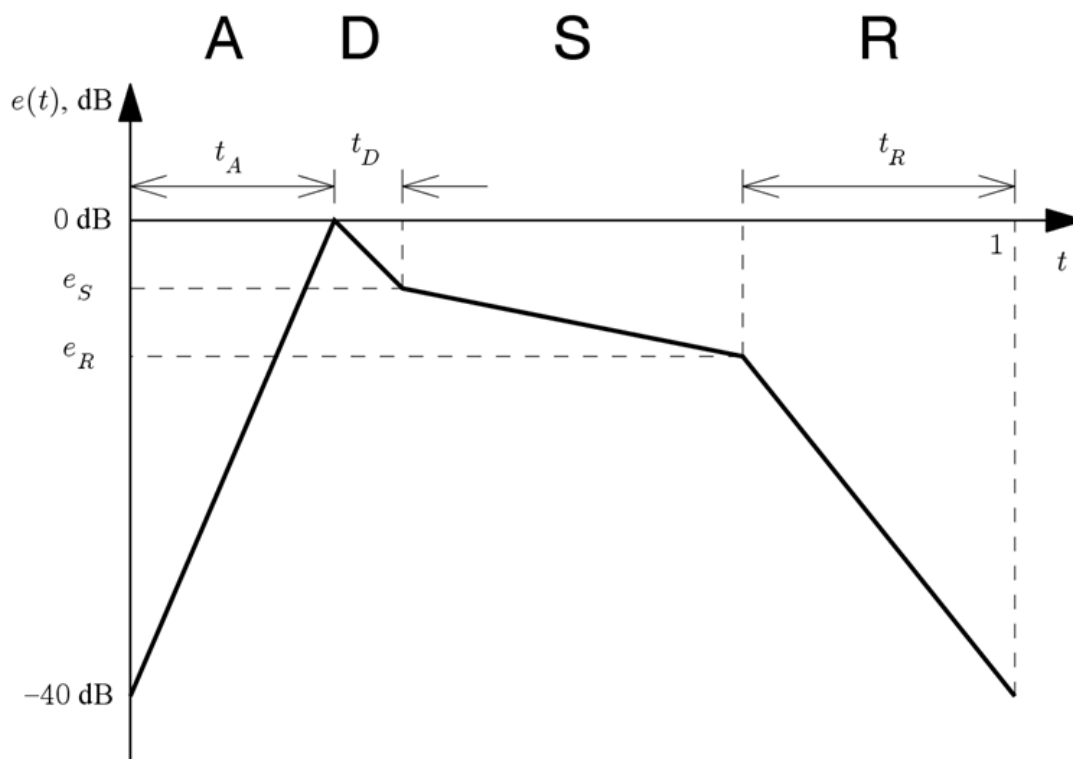


Figure 2.1: Specifications for the ADSR-style envelope

The envelope shape is defined over a normalized time interval of 0 to 1; the envelope stretches or compresses to match the actual duration requested. The three time intervals are likewise expressed in normalized units, so an attack duration (t_A) of 0.2 indicates that the attack time is 20% of the entire envelope duration. The envelope values range from 0 dB (full sound intensity) to -40 dB (close to silence). Once you have defined the envelope shape in terms of straight-line segments, you can create the envelope amplitude waveform by undoing the decibels operation; recall that an amplitude "A" expressed in dB is $20 \log_{10}(A)$.

The subVI requires the following controls as input parameters (units specified in brackets):

1. **duration [s]** - total duration of envelope
2. **fs [Hz]** - system-wide sampling frequency

3. **t values** - three-value array containing attack (A) interval, decay (D) interval, and release (R) interval. All three values are in the range 0 to 1, so the intervals are to be treated as fractions of the total duration of the envelope. For example, A=0.3 indicates the attack interval is 30% of the envelope's duration.
4. **e values** - two-value array containing the envelope value at the beginning of the sustain interval (eS in the diagram above) and the envelope value at the beginning of the release interval (eR).

The subVI requires the single indicator (output) **envelope**, an array with values in the range 0 to 1.

Show that your ADSR VI functions properly by plotting the envelope for at least two distinct cases. Plot both the dB form of the envelope as well as its non-dB form.

The following screencast video provides coding tips and other suggestions to help you develop your subVI.

Image not finished

Figure 2.2: [video] Coding tips and suggestions for **ADSR.vi**

2.5 Part 2: Sound Generator

Make a subVI called **SoundGen.vi** that combines a signal source with the ADSR envelope generator. Use a **case structure** with an **enumerated data type** control that allows you to choose one of several different "voices" or "instruments." The case structure then defines the ADSR input parameters to choose the type of signal source (or noise source for a percussive instrument).

The subVI requires the following controls as input parameters (units specified in brackets):

1. **duration [s]** - total duration of note
2. **fs [Hz]** - system-wide sampling frequency
3. **fref [Hz]** - reference frequency
4. **note** - note frequency as described by an integer value that denotes the number of semitone intervals from the reference frequency; **note** can be a negative, zero, or positive value
5. **instrument** - enumerated data type with values of your choice; your SoundGen.vi should have at least three different tone-type sounds and at least two different percussive sounds. Use a variety of envelope shapes.

If the discussion in Item 4 for the **note** control seems mysterious, review the module [INSERT NAME AND MODULE LINK] to learn how to calculate the frequency of a semitone interval using **equal temperament**.

Show that your sound generator VI functions properly by developing a top-level VI that plays a chromatic scale. Vary the duration of the notes. Plot the waveform to ensure that the envelope shape makes sense. Save your generated sound to a .wav file to be included with your deliverables.

The screencast video of Figure 2.3 offers coding tips and other suggestions to help you develop your subVI.

Image not finished

Figure 2.3: [video] Coding tips and suggestions for **SoundGen.vi**

2.6 Part 3: Multi-Voice Composition

Compose a simple piece of music using analog synthesis techniques of your choice. Better compositions will include variety, e.g., different envelope parameters at different times, multiple channels (chords), stereo, percussive sounds, and so on. Write a paragraph or two that describes your compositional technique. Better compositions are click-free, i.e., joining the notes together does not produce any pops or clicks.

Hints:

- You may wish to use a familiar melody (remember, you've got an equation that converts a note from the equal-tempered scale into frequency), or you may want to use a musical "experiment" based on an algorithm as the basis for your composition.
- Operate multiple instances of **SoundGen.vi** in parallel (each with a different instrument setting) and add their outputs together.
- Import multiple spreadsheets to define your note lists (use "File IO | Read Spreadsheet"). Spreadsheets are the easiest way to hand-edit the contents of arrays.
- Remember to normalize your finished output before saving it to a .wav file. Use **QuickScale** ("Signal Processing | Signal Operation | Quick Scale") for this purpose.
- View your finished composition as a waveform plot. Confirm that the envelope shapes make sense, that the waveform is symmetrical about the zero axis (it should not contain any constant offset), and that the waveform fits within the ± 1 range.

2.7 Alternative Part 3: Multi-Voice Composition Using MIDI File

The overall objectives of this alternative Part 3 are the same as described above, but the technical approach is different.



Go to MIDI JamSession⁴ to learn all about a LabVIEW application VI of the same name that reads a standard MIDI file (.mid format) and renders it to an audio file (.wav format) using "instrument" subVIs of your own design. You have already done all of the work necessary to create your own instruments, and it is a simple matter to place them inside a standard subVI wrapper (or "Virtual Musical Instrument," VMI) that MIDI JamSession can use.

⁴"[LabVIEW application] MIDI_JamSession" <<http://cnx.org/content/m15053/latest/>>

Index of Keywords and Terms

Keywords are listed by the section with that keyword (page numbers are in parentheses). Keywords do not necessarily appear in the text of the page. They are merely associated with that section. *Ex.* apples, § 1.1 (1) **Terms** are referenced by the page they appear on. *Ex.* apples, 1

- A** ADSR, § 1(1), 3, 3, § 2(7)
ADSR.vi, 8, 9
analog synthesis, § 1(1), § 2(7)
Analog synthesizers, 1
- C** case structure, 9
continuous, 1
control voltage (CV), 2
controller, 3
controllers, 2
CV, 3
- D** duration [s], 8, 9
- E** e values, 9
enumerated data type, 9
envelope, 9
envelope generator, § 1(1), 2, 3, § 2(7)
envelope generators, 2
equal temperament, 9
- F** fref [Hz], 9
fs [Hz], 8, 9
- I** instrument, 9
interactive controller, 3
- L** LabVIEW, § 2(7)
LFO, 3
low-frequency oscillator, 3
- M** modular synthesis, § 1(1), § 2(7)
modules, 2
- N** note, 9, 9, 9
- P** patched, 2
patches, 2
portamento, 4, 5
processors, 2
programmed controller, 3
- Q** QuickScale, 10
- S** SoundGen.vi, 9, 9, 10
sources, 2
- T** t values, 9
- V** VCA, § 1(1), 2, 2
VCF, § 1(1), 2
VCO, § 1(1), 2, 2
voltage-controlled amplifier, 2
voltage-controlled filter, 2
voltage-controlled oscillator, 2

Attributions

Collection: *Musical Signal Processing with LabVIEW – Analog Synthesis and Modular Synthesizers*

Edited by: Ed Doering

URL: <http://cnx.org/content/col10480/1.1/>

License: <http://creativecommons.org/licenses/by/2.0/>

Module: "Analog Synthesis Modules"

By: Ed Doering

URL: <http://cnx.org/content/m15442/1.2/>

Pages: 1-5

Copyright: Ed Doering

License: <http://creativecommons.org/licenses/by/2.0/>

Module: "[mini-project] Compose a piece of music using analog synthesizer techniques"

By: Ed Doering

URL: <http://cnx.org/content/m15443/1.2/>

Pages: 7-10

Copyright: Ed Doering

License: <http://creativecommons.org/licenses/by/2.0/>

Musical Signal Processing with LabVIEW – Analog Synthesis and Modular Synthesizers

Analog modular synthesizers popular in the 1960s and 1970s produce sound with electronic devices such as oscillators, amplifiers, filters, and envelope generators linked together by cables. A specific cable configuration (or "patch") produces a distinct sound controlled by a keyboard or sequencer. While digital synthesis has largely replaced analog synthesizers, the concepts and techniques of analog synthesis still serve as the basis for many types of synthesis algorithms. Learn about modular synthesizers and use LabVIEW to compose a piece of music by emulating an analog synthesizer. This course is part of the series "Musical Signal Processing with LabVIEW".

About Connexions

Since 1999, Connexions has been pioneering a global system where anyone can create course materials and make them fully accessible and easily reusable free of charge. We are a Web-based authoring, teaching and learning environment open to anyone interested in education, including students, teachers, professors and lifelong learners. We connect ideas and facilitate educational communities.

Connexions's modular, interactive courses are in use worldwide by universities, community colleges, K-12 schools, distance learners, and lifelong learners. Connexions materials are in many languages, including English, Spanish, Chinese, Japanese, Italian, Vietnamese, French, Portuguese, and Thai. Connexions is part of an exciting new information distribution system that allows for **Print on Demand Books**. Connexions has partnered with innovative on-demand publisher QOOP to accelerate the delivery of printed course materials and textbooks into classrooms worldwide at lower prices than traditional academic publishers.