Programming in Harmony

By James Mooney

Restoring the History of Digital Music

In 1961, Peter Samson, a student at MIT, programmed the new PDP-1 minicomputer to play polyphonic music. This experiment led to the Harmony Compiler, one of the earliest pieces of music-making software to be distributed to users. It also inspired Peter's later pioneering work in real-time digital sound synthesis.

Nowadays, Peter is working as a docent at CHM. I interviewed him in 2020 in the PDP-1 lab, where he demoed the music software and explained how he developed it.

‘Mozart or Bach’: Peter Samson loads a paper tape of polyphonic music on the PDP-1: https://youtu.be/AuqNHe_QWN8?si=L04I_qYaamTsta_D.

Peter's system was novel in 1961 because it was both real-time and polyphonic: real-time meaning that the computer could play music "live," polyphonic meaning it could play more than one note at a time.

As early as 1951, engineers in Australia programmed the CSIRAC computer to play music in real-time, but only monophonically (one note at a time).[1] At Bell Labs, Max Mathews achieved polyphony with his MUSIC software in 1958, but this worked in "batch mode," not real-time: running a "job" on punched cards produced a digital waveform on tape, which could be played back later via a custom-built digital-to-analog converter.[2]

Closer to home, Ercolino Ferretti was developing experimental sound synthesis programs for MIT's IBM 7094 mainframe, but these too were non-real-time. "Real-time was my thing from the start," Peter told me. "Hands-on interaction versus, you know, submit your job."

Peter Samson demonstrates his PDP-1 music programs at CHM, February 2020:

[1] Paul Doornbusch, ‘Early Computer Music Experiments in Australia and England’, *Organised Sound*, 22.2 (2017), pp. 297–307 (p. 301), doi:10.1017/S1355771817000206.
[2] Curtis Roads and Max Mathews, ‘Interview with Max Mathews’, *Computer Music Journal*, 4.4 (1980), pp. 15–22 (p. 16), doi:10.2307/3679463.

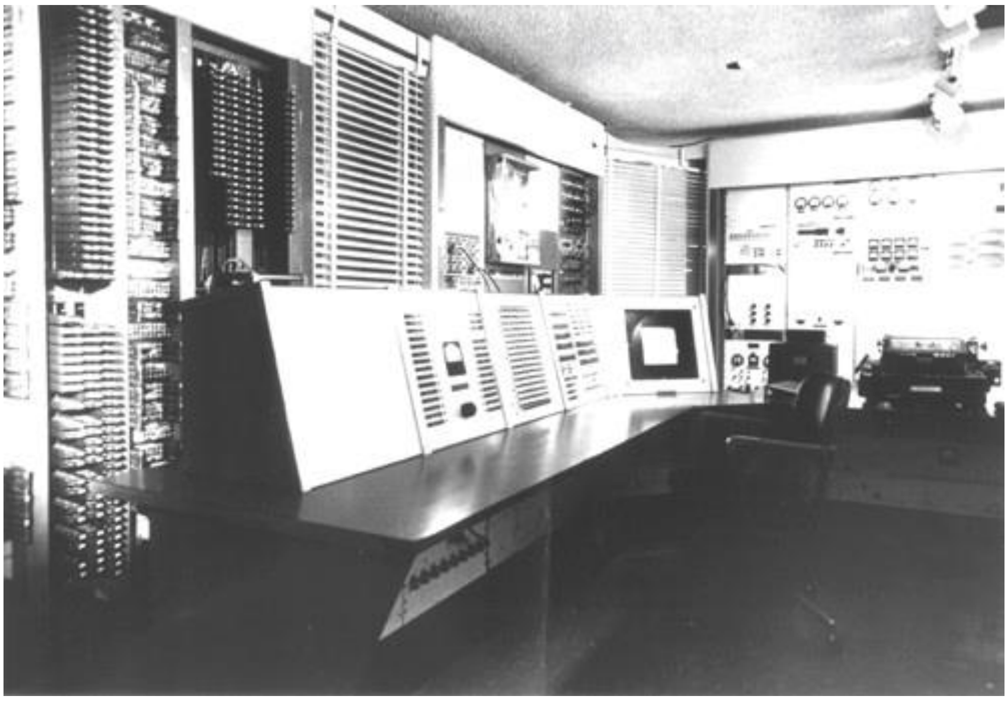
Peter first attempted real-time synthesis in 1960 when he programmed MIT’s experimental TX-0 computer to play monophonic melodies. He did this by writing a program that sent precisely timed pulses to the machine’s built-in speaker: 262 pulses per second for "middle C," 294 pulses per second for "D," and so on. Timing the bits to coincide with the machine’s instruction cycle required clever programming, Peter recalled—but an even greater challenge arose when he considered rewriting the program to play several notes simultaneously.

"Where are we going to store multiple bits?" he wondered.

Flip-Flop Polyphony

His solution was to use outboard flip-flops. A flip-flop is a digital circuit whose output can be switched "on" (positive voltage) or "off" (no voltage) by an electrical control signal. With such a circuit, it is possible to generate an electrical waveform roughly the shape of a square wave by switching the circuit on and off at regular intervals.

A photograph of the TX-0, c.1956 in CHM's collection, 102622467. Courtesy of Gwen Bell:



The TX-0 had plug-in flip-flop units controlled from the machine’s processor. Peter determined that by using three of these, it would be possible to make the machine play three different musical notes simultaneously. Going a step further, he hypothesized that three-part polyphonic music might be attainable by programming the machine to play a sequence of three-note combinations one after the other. To test that hypothesis, he set about writing a new three-voice music player program.

Achieving three-part polyphony required an appropriate choice of music. Peter chose the "Presto" movement of J.S. Bach’s *Organ Concerto in G Major*, partly because he loved the music, and because the sound of an organ could be reasonably convincingly emulated using the means at his disposal.

"If you’re generous, it sounds like a pipe organ," he reflected.

Programming TX-0 to play "contrapuntal" music—that is, music with multiple melodic lines—required Peter to conceptualize the music in an unconventional way. Musicians tend to think of three-part counterpoint "horizontally," as three simultaneous intertwining melodies. Instead, Peter thought of the music "vertically." He observed that in three-part polyphonic music, at any given instant, there may be up to three notes sounding simultaneously. His chosen piece, for example, begins with a high "G" in the right hand, "B" in the left hand, and low "G" in the pedal part, a situation that persists for the duration of a sixteenth-note. Then, the right hand changes to a high "D" and the left hand to a low "D," while the pedal part holds the low "G" for a further sixteenth-note, and so on. In other words, Peter thought of the music as a sequence of three-note "instants," each consisting of pitch 1, pitch 2, pitch 3, and duration N. An "instant," he explained to me, is an arbitrary duration in which the pitches of all three voices remain constant.

To realize this concept, Peter wrote a music player program that read frequency and duration data, in three-note instants, from punched paper tape into TX-0's magnetic core memory. The program switched the three flip-flops on and off simultaneously at the specified frequencies, holding each three-frequency combination for the specified duration before moving on to the next one. The flip-flops were connected to an electronic filter, amplifier, and loudspeaker, which removed some noise components of the signals and rendered them readily audible as musical notes.

Below is an audio clip of the third movement, "Presto," from J.S. Bach's *Organ Concerto in G Major* on PDP-1.

[Embedded audio is included in the online version.]

The three-part music player program worked well, but there was a problem: entering the data was very time consuming. Rendering a complete piece of music required Peter to manually divide the music up into three-note instants, calculate the frequency of each note and the number of loop iterations required to produce the correct duration for each instant, then painstakingly enter the data on paper tape. Having completed that process for the Bach organ concerto, Peter recalled his frustration.

"I'm not doing that again!" he told me. "I'm going to write a compiler."

The Harmony Compiler

Peter's work on the Harmony Compiler coincided with the arrival at MIT of a new PDP-1 minicomputer, a commercial machine based on the design of the experimental TX-0. Peter wrote the compiler to be cross-compatible with both machines. The compiler enabled him to follow a musical score and enter the note data in a simple symbolic shorthand: one number represented the note's pitch-position on the stave; another, its duration as a quarter note, eighth note, sixteenth note, and so on. The compiler converted these numbers into frequency and loop-iteration values, meaning Peter no longer had to do those calculations manually.

Peter also devised a system of character symbols for musical articulation, and even Baroque ornamentation.

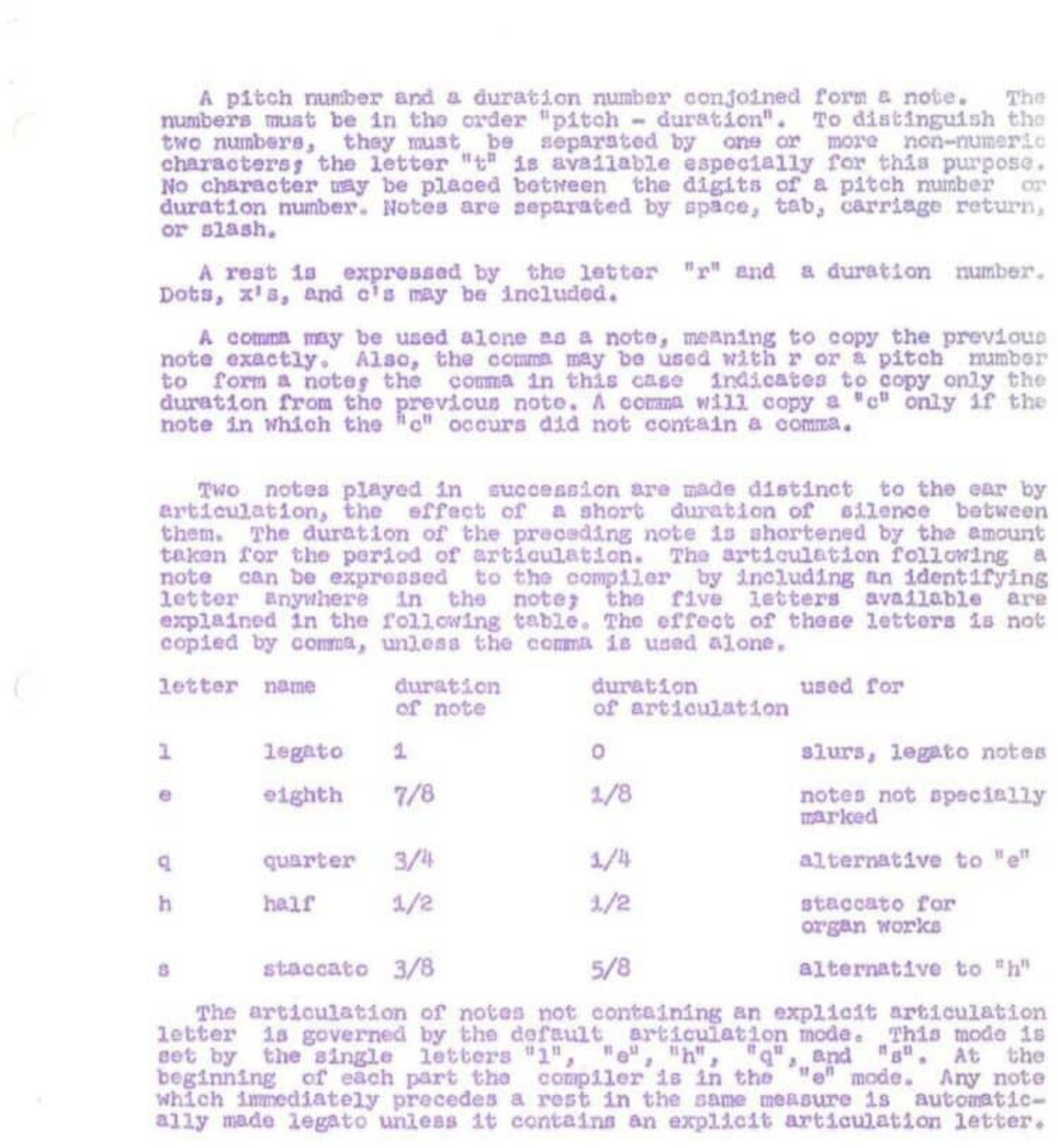
Each contrapuntal part could be entered horizontally: the compiler took care of merging the parts and generating the vertical instants that Peter had previously had to figure out on paper. The compiler accepted as its input a punched paper tape containing pitch, duration, and articulation data in this more user-friendly format, and produced as its output a punched paper tape containing the less human-friendly data expected by the music player program.

While the Harmony Compiler was cross-platform, the music player program had to be re-written using the PDP-1's instruction set. Peter took this opportunity to augment the program's capabilities to four polyphonic voices, as well as adding routines for rendering the opcodes just mentioned.

Here is an audio clip of J.S. Bach's Two Part Invention No. 1 on PDP-1.

[Embedded audio is included in the online version.]

A page from the Harmony Compiler manual. A "staccato" (shortened) note could be specified by appending the letter "s" to a pitch value:



A pitch number and a duration number conjoined form a note. The numbers must be in the order "pitch - duration". To distinguish the two numbers, they must be separated by one or more non-numeric characters; the letter "t" is available especially for this purpose. No character may be placed between the digits of a pitch number or duration number. Notes are separated by space, tab, carriage return, or slash.

A rest is expressed by the letter "r" and a duration number. Dots, x's, and c's may be included.

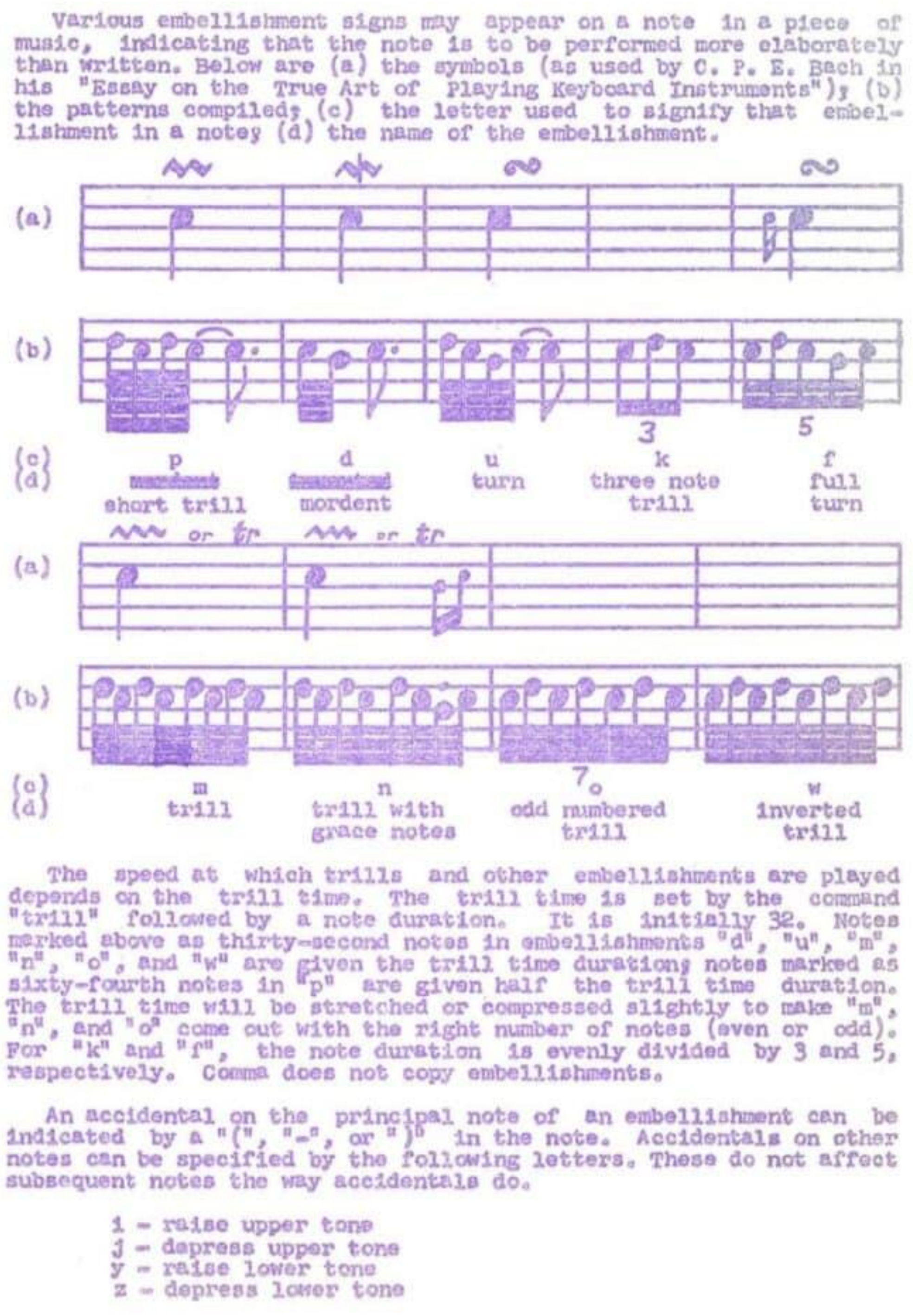
A comma may be used alone as a note, meaning to copy the previous note exactly. Also, the comma may be used with r or a pitch number to form a note; the comma in this case indicates to copy only the duration from the previous note. A comma will copy a "c" only if the note in which the "c" occurs did not contain a comma.

Two notes played in succession are made distinct to the ear by articulation, the effect of a short duration of silence between them. The duration of the preceding note is shortened by the amount taken for the period of articulation. The articulation following a note can be expressed to the compiler by including an identifying letter anywhere in the note; the five letters available are explained in the following table. The effect of these letters is not copied by comma, unless the comma is used alone.

letter	name	duration of note	duration of articulation	used for
l	legato	1	0	slurs, legato notes
e	eighth	7/8	1/8	notes not specially marked
q	quarter	3/4	1/4	alternative to "e"
h	half	1/2	1/2	staccato for organ works
s	staccato	3/8	5/8	alternative to "h"

The articulation of notes not containing an explicit articulation letter is governed by the default articulation mode. This mode is set by the single letters "l", "e", "h", "q", and "s". At the beginning of each part the compiler is in the "e" mode. Any note which immediately precedes a rest in the same measure is automatically made legato unless it contains an explicit articulation letter.

Baroque ornamentation. Trills, mordents, and turns each had character symbols:



Various embellishment signs may appear on a note in a piece of music, indicating that the note is to be performed more elaborately than written. Below are (a) the symbols (as used by C. P. E. Bach in his "Essay on the True Art of Playing Keyboard Instruments"); (b) the patterns compiled; (c) the letter used to signify that embellishment in a note; (d) the name of the embellishment.

(a)
(b)
3
5
or tr
or tr
(a)
(b)

(c)	p	d	u	k	f
(d)	~~mordent~~ short trill	~~trill~~ mordent	turn	three note trill	full turn

(c)	m	n	7 o	w
(d)	trill	trill with grace notes	odd numbered trill	inverted trill

The speed at which trills and other embellishments are played depends on the trill time. The trill time is set by the command "trill" followed by a note duration. It is initially 32. Notes marked above as thirty-second notes in embellishments "d", "u", "m", "n", "o", and "w" are given the trill time duration; notes marked as sixty-fourth notes in "p" are given half the trill time duration. The trill time will be stretched or compressed slightly to make "m", "n", and "o" come out with the right number of notes (even or odd). For "k" and "f", the note duration is evenly divided by 3 and 5, respectively. Comma does not copy embellishments.

An accidental on the principal note of an embellishment can be indicated by a "(", "-", or ")" in the note. Accidentals on other notes can be specified by the following letters. These do not affect subsequent notes the way accidentals do.

i	raise upper tone
j	depress upper tone
y	raise lower tone
z	depress lower tone

In 1962, the Harmony Compiler and music player became part of the stock software bundle shipped with new PDP-1 machines. This places Peter's programs amongst the earliest programmable music software packages to be distributed to users.

"DEC paid me $200 to write a version that will run on PDP-1 without the outboard flip-flops," Peter explained.

Developing Digital Synthesis

Peter stopped working with the PDP-1 in 1963. He wrote a six-voice program for the PDP-6 some months later, then left MIT in 1970, but he remained at the forefront of developments in digital synthesis.

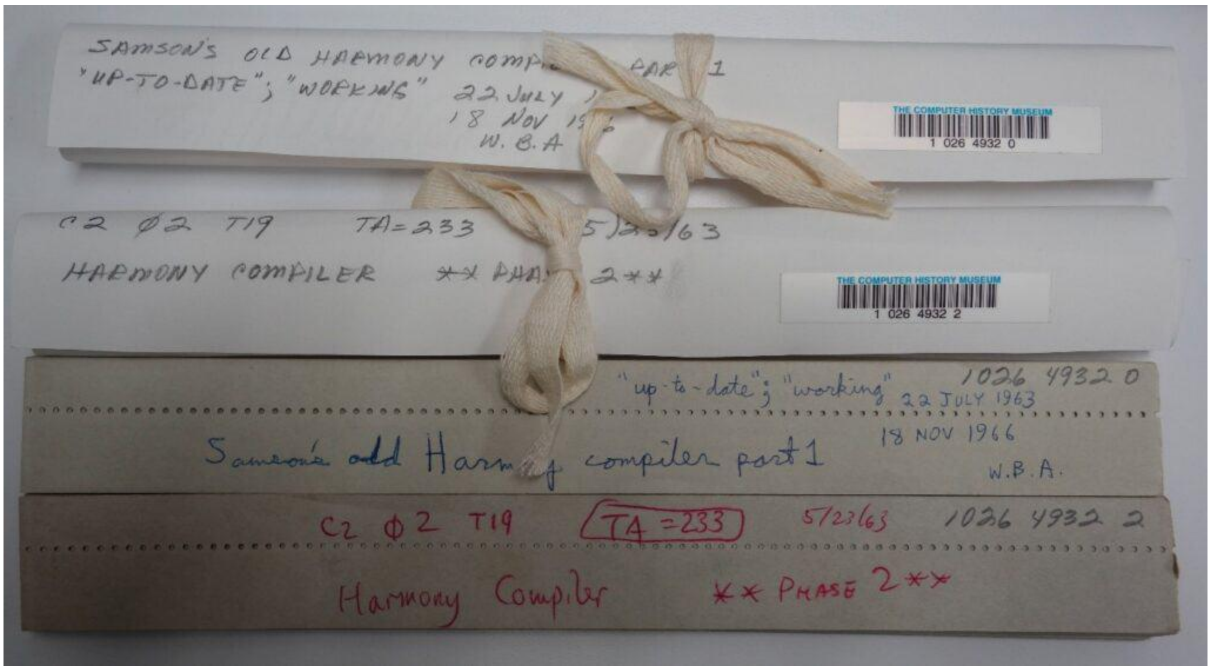
In the 1970s, Peter developed the Systems Concepts Digital Synthesizer, one of the first dedicated real-time digital synthesizers. A radically different design based on hardware rather than software, the Samson Box, as it became known, was motivated by the desire to facilitate, in real-time, the superior synthesis capabilities of non-real-time software systems like Max Mathews’ MUSIC programs. The emphasis on real-time made this an extension into the hardware domain of Peter’s earlier work developing synthesis software for the PDP-1 and PDP-6.

Restoring History

Meanwhile, back at MIT, Peter’s music software was maintained by various programmers until, eventually, the PDP lab closed. That would have been the end of the story, if not for the fact that somewhere along the line, a box of paper tapes, including several pieces of music and various parts of the player and compiler programs, made its way from MIT to the Computer History Museum.

It isn’t clear precisely when that happened, or who made the donation, but by extraordinary coincidence, Peter rediscovered this uncatalogued box of tapes in 2004, when he became involved as a volunteer on CHM’s PDP-1 Restoration Project. Playing music on the restored PDP-1 thus became one of the project’s goals.

Two paper tapes containing parts of the Harmony Compiler:


SAMSON'S OLD HARMONY COMP... PAR 1
"UP-TO-DATE"; "WORKING" 22 JULY 1
18 NOV 19
W. B. A
THE COMPUTER HISTORY MUSEUM
1 026 4932 0
C2 Ø2 T19 TA=233 5/23/63
HARMONY COMPILER ** PHA 2**
THE COMPUTER HISTORY MUSEUM
1 026 4932 2
"up-to-date"; "working" 1026 4932 0
22 JULY 1963
Samson's old Harmony compiler part 1
18 NOV 1966
W.B.A.
C2 Φ2 T19 TA=233 5/23/63 1026 4932 2
Harmony Compiler ** PHASE 2 **

Peter immediately recognized that the programs had been rewritten for various newer systems over the years and would not be able to run on the old PDP-1 hardware. However, he knew the playing algorithm well, and with the salvaged tapes as an aide memoire—plus a flowchart he’d kept that outlined the process for interpreting musical data—he was able to write the player program again from scratch.

Rewriting the compiler forty years after the fact using ambiguously labelled punched paper tapes and a copy of the user manual as a reference would have been a much more onerous task, but in a turn of good fortune, a listing of it turned up in Peter's basement.

"My wife was rummaging around in the basement and found a box of MIT memorabilia my mother had saved, bless her, and there at the bottom was a print-out of the compiler, a listing. A mere 67 pages? I'll type it in again! Which I did. So we have a working compiler as well as a working player."

It is only thanks to this somewhat improbable sequence of contingencies that, in 2020, I was able to receive a demonstration of Peter's PDP-1 music programs—and hear about this important and often overlooked chapter in the history of digital synthesis.

Peter Samson listens to music during the PDP-1 Restoration Project, March 2005:

Members of the PDP-1 Restoration team discuss the project:
https://youtu.be/BHTa3EfzrNE?si=2299Muyr1mVEWQnI

Digital synthesis with computers is commonplace nowadays: anybody with a laptop can try their hand at making music using soft synths running within a digital audio workstation (DAW) package like Logic, Reaper, Ableton, or GarageBand. The range of techniques used to generate sounds digitally has expanded enormously since the early 1960s, of course. Frequency modulation (FM), physical modelling, and granular synthesis algorithms have all been implemented in software, for instance.

Samson's groundbreaking work with TX-0 and PDP-1 predated all these, however, and his programs can perhaps be regarded as the earliest real-time software synthesizers of all.

Acknowledgements

Thank you to Peter Samson, Heidi Hackford, Al Kossow, and Dag Spicer. Research for this article was supported by British Academy/Leverhulme grant SRG19\190060.

About the Author

James Mooney is a historian and sociologist of music technology and associate professor of musicology and music technology at the School of Music, University of Leeds, UK. He is a subject-matter expert for CHM, specializing in electronic sound and computer music, and a research associate of the Science Museum Group (UK), specializing in electronic sound and music technology.