Syntauri limited

THE ALPHASYNTAURI: A KEYBOARD BASED DIGITAL PLAYING AND RECORDING SYSTEM WITH A MICROCOMPUTER INTERFACE

Paper presented at the
67th Convention, Audio Engineering
Society, New York city, Nov. 1, 1980

By Charlie Kellner, Ellen V.B. Lapham
and Laurie Spiegel

3506 waverley street
palo alto, ca 94306
(415) 494-1017

Paper presented at the Audio Engineering Society National Meeting, Nov. 1, 1980

THE ALPHASYNTAURI; A KEYBOARD BASED DIGITAL PLAYING AND RECORDING SYSTEM WITH A MICROCOMPUTER INTERFACE

By Charlie Kellner and Ellen Lapham, Syntauri Ltd., Palo Alto
and Laurie Spiegel, composer, New York City

abstract:
Interfaced to an inexpensive, readily available microcomputer, the alphaSyntauri^{tm} provides musicians and studio technicians with a digital synthesizer. It can be used for sequencing and real time effects, timing, repeats, and timbre. Waveforms and timbres may be created and stored on disk using built-in additive synthesis and analysis programs.

Introduction

The alphaSyntauri is a modular, general purpose microcomputer based instrument intended for music composition and real time performance. The philosophy which guided the system's designer, Charlie Kellner, was based on one fundamental concept:

> A flexible, useful instrument can successfully use software for real time instrument operation and for generating and *controlling timbres and effects.*

Background

Historically, sound synthesizing devices have been hardware based. The programming of early synthesizers was accomplished using either hard-wired equipment or patch bays, such as in the original Moog synthesizer.

Dependence upon circuitry and hardware has inherent limitations. First, the 'programming' is not easily changed. Patch bays can become complex, and on-the-road modifications require hardware expertise, time, and patience. Second, the cost of the instrument increases as a function of size and complexity - so expansions and modifications create cost penalties, and thus have tended to be limited to large, fixed installations.

Today, digital technologies are opening horizons for music exploration and control. Uses of digital technologies for sound synthesis range from dedicated microprocessors to full general purpose computer based systems. Microprogrammed synthesizers have been effectively 'hard wired' at the software level: the user typically does no have access to the microprocessor to make programming and control changes. (The Prophet 5 is an example of the microprogrammed synthesizer.)

Taking advantage of computer system technologies, the Fairlight and Synclavier, for instance, use more sophisticated techniques such as disk storage media, and allow a greated degreee of user control.

The alphaSyntauri uses a general purpose computer system, the Apple II-which is designed simply for information handling for any purpose. Here, *the system software designer puts control in the user's hands through* software commands, letting the user modify - or even create from scratch-his own musical instrument.

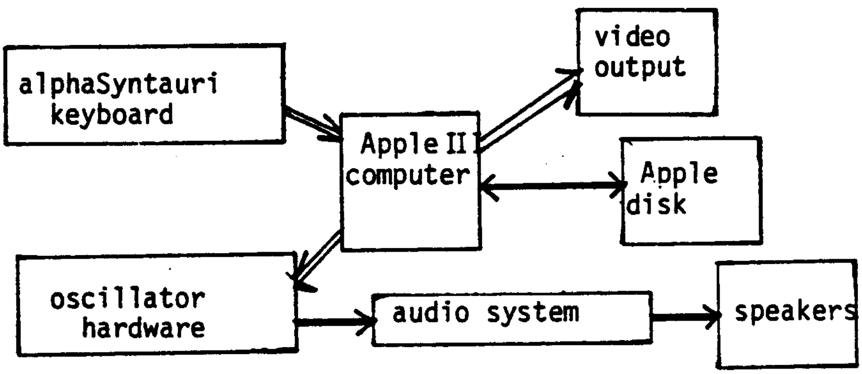
Controlling and accurately replicating sounds and note sequences is accomplished through programming (software). A user can decide, for instance, to additively synthesize ten timbres by using a built-in software module, the additive synthesis program. Saving these timbres to diskette is accomplished using standard Apple II disk utilities called by the program. Once saved, the new instruments are stored in unambiguous digital form.

It is the general purpose nature of the computer which here provides new options to the music industry when working with 'synthesized' sounds. From the instrument definition process to the editing of recorded pieces, the Apple II supports the user with a multitude of software tools, languages, printers, and utilities.

Users may send programs, instruments, and even recorded pieces over phone lines using conventional Apple II peripheral equipment. The computer is completely impartial and accurate - thus an improvisation recorded recorded in Los Angeles can be sent to an alphaSyntauri site in New York along with the instrument definitions and effects for review and editing or mixing.

<u>The alphaSyntauri system structure</u>

Primary system components are the alphaSyntauri keyboard, software, and interface hardware, plus the oscillators (available from various sources); the Apple II computer system; and the audio system. These system components are illustrated in the accompanying diagram.



<u>The system software</u>

The major alphaSyntauri performance program, called ALPHA III, is a software-based real time emulation of the traditional synthesizer. The program performs all the functions of a synthesizer with hardware such as switches, knobs, patch cords and hard circuitry such as envelope generators, mixers, and filters.

In operation, the program calculates what a hardware (analog) synthesizer would be doing evey 20 milliseconds, and controls the output oscillators in real time to achieve this. Because many major features of traditional synthesizers have been incorporated into the ALPHA III program, the processing cycle is complex.

To achieve the required musical results from the alphaSyntauri system, the languages with which the system has been written have been written have been chosen for their specific benefits. The programs embedded in the process loop are written in 6502 native assembly code for speed. The main control program which sets up the loop and allows user-selected paramters to control the cycles, is written in BASIC, a well known and easily modified programming language.

The alphaSyntauri process

Within the alphaSyntauri system, there are multiple processes (or tasks) being done essentially simultaneously. The computer is sufficiently fast that all these tasks, and their attendant decisions, are executed without impairing the sound quality.

System tasks include reading the keyboard input device, determining which key was pressed, reading tables (envelopes, waveforms, etc.) and updating the output oscillators. See the attached process flow diagram.

The oscillator update process is the most fundamental process in the alphaSyntauri system. The update tells the output hardware to produce a sound at a certain volume and frequency, until the next update cycle, when the oscillator control(s) may be changed to reflect new keys pressed, new envelope stages, or the termination of the note.

A keyboard event...the pressing of a key during live play... initiates many processes itself. This event, interestingly enough, is treated as an exception event, as its detection and determination of what to do with the information takes less than 1% of the total process loop time.

Examples of software-controllable parameters and features

The following examples were selected to demonstrate the range of features and control options achievable using the general purpose micro-computer, the Apple II. Extensive use is made of the Apple's utilities, memory handling, and disk operations to extend the alphaSyntauri instrument beyond what would have been possible using dedicated electronic components.

The tradeoffs involved in not selecting possibly higher speed dedicated microprocessors have not materially impacted the audible result: in fact, it is the generality and expandability of the system design which is achieved only through the use of a low-cost commercial computer system.

Keyboard tuning

The alphaSyntauri has been designed, as standard for the first release, to have 12 keys per octave, with a minimum resolvable step of one quarter tone. Thus, there are 24 slots in a pitch translation table (loaded into RAM) which are available for frequency assignment for one octave.

To minimize table size, the system assumes that frequency doubles with each octave, so that the frequency of one octave can be derived by multiplying (or dividing) by two. Computers are especially proficient at this task.

During real time performance and playback, the keyboard can be dynamically tuned using the FC (frequency control) software from the Apple II keyboard. Users may input any number from Ø to (effectively) 180 or so: middle C corresponds to 30 in mid-keyboard, 6 transposes the keyboard down one octave, still in the key of C).

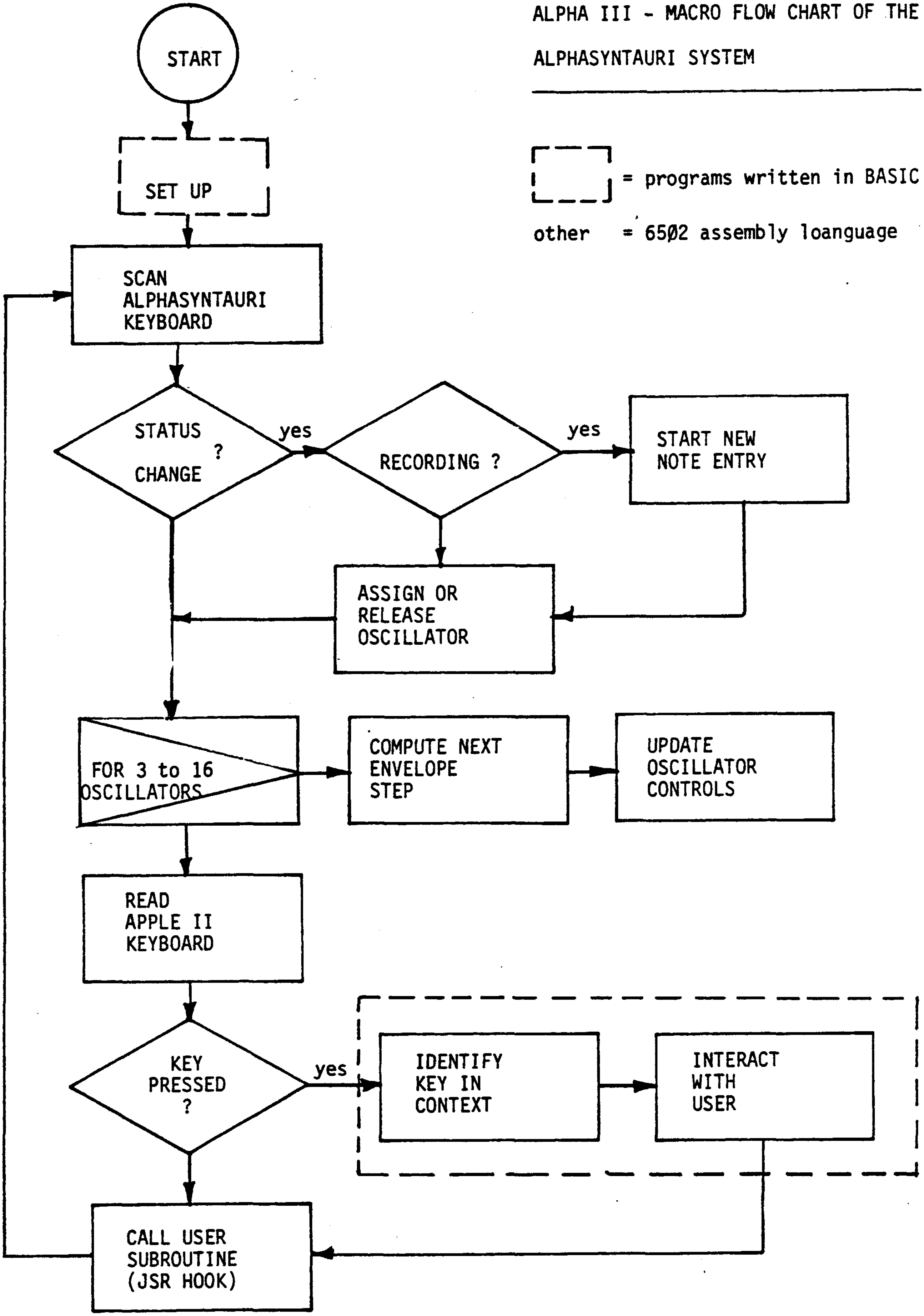
For finer tuning adjustments the user exits real time control to recalculate the entire table from which the oscillator frequency assignments are made. The tuning process is: 1) RUN SCALE III program*, 2) LIST lines Ø to 99, and 3) change the appropriate paramters in the internally documented program to fine tune to within nine place decimal accuracy.

* currently, three types of scales, well tempered, just, and international, are built in to the alphaSyntauri.

ALPHA III - MACRO FLOW CHART OF THE ALPHASYNTAURI SYSTEM

[dashed box] = programs written in BASIC

other = 65Ø2 assembly loanguage



Syntauri Ltd.
AES/Nov. 1, 1980

Software features, continued.

Variable Speed Playback

During the playback loop, there is an additional process called READ APPLE GAME PADDLES. Because of the way they are implemented in hardware, it takes from zero to approximately 20 milliseconds, in a linear fashion, to read the paddles. The read time simply increases the process loop linearly according to the paddle setting. Thus, during a 'read' of the paddle setting, the process loop time increases: the performance/playback process can be slowed down by as much as 50%, resulting in a slower playback.

The variable speed does not affect the instrument definition or in any way affect the sound quality of the piece being played back.

Faster playback is also available. During the playback process, a duration byte which was attached to each note during recording, is counted out in the same way it was recorded.(This duration byte is essentially a length-of-real-time-passage counter.) Pressing the Apple II game paddle 'target' button initiates a fast forward by causing the duration byte to be counted out by twos. This doubles the playback speed.

Normal variable speed playback range is 50 to 100% of the original input (playing of the keyboard) speed, and fast-forward doubles this range from 100 to 200%.

Analysis of waveforms

Waveforms are user created through additive sine synthesis. The program ANALYZER III allows any waveform stored on disk to be separated into its harmonic components. A simple digital filter algorithm is executed for each of the first twenty harmonics(more if desired). The resulting display reports their relative amplitude on an arbitrary scale (roughly from 1 to 200.)

The harmonic analysis can be used, of course, with the built-in WAVE III synthesis program to re-create the analyzed waveform. A more interesting use might be to process the first waveform through a filter program to obtain the equivalent filtered waveform. Two waves could be added harmonically to obtain a third. (All it takes is a little imagination.)

Velocity sensing

Each key on the alphaSyntauri keyboard has two electrical contacts, one which makes about one third of the key down travel, the other which makes at bottom. When a key is pressed, the following occurs:

1. Upper contact closes. Here, a count register in the Apple II's memory which is assigned and maintained for each key is set to zero. No other action takes place.
2. The lower contact closes. The count value which has accumulated in that key's timing register is used as an index into the velocity assignment table. The attack rate and attack target volume of that oscillator are here determined. (note, the psychoacoustics of perceived effects as a function of key velocity have been handled by varying both the attack rate and target volume, where loudness is perceived

to increase not only with absolute loudness increases, but also with the quickness of getting there.)

3. A new oscillator is then assigned for the key which has been pressed.

To achieve velocity sensing results, attack rate and volume are both used. Typically, the attack rate and volume for a given key are inversely proportional to the time between contact closures. That is, velocity is the inverse of transition time. The actual volume effects, changes to attack rate and volume, are handled by a look up table which is loaded automatically by the ALPHA III software into a specific memory location.

Now, to obtain a stiff or loose keyboard, a parameter value is set in the program which effectively makes the table values more or less linear. To illustrate:

The linear, loosest keyboard has a parameter value of 7.99

The logarithmic, stiffest keyboard, has a parameter of Ø

In the stiffest keyboard, the key has to be struck very fast and sharply to get loud volumes.

The table values used affect both the real and perceived energy differences between a soft or a hard (fast) key depression. For a soft key stroke, there is less energy in the sound. That is, a lower target volume is arrived at much later. It is the first few milliseconds of hearing which determine the perceived energy in the sound, so the note that reaches peak volume first has the most energy.

Flexibility in the velocity sensing results can be further achieved through completely reworking the look up table from which the velocity sensing results are ultimately derived. In addition, being a general design, the parameter(s) of the envelope which are affected by the volocity sensing process may be altered from the attack rate/target volume to, for instance, the attack target volume/decay rate.

<u>User's own subroutine</u>

The main process control loop (see illustration) contains what is called an unconditional JSR (jump to a user-written subroutine). This allows any user familiar with 6502 assembler code to devise his/her own special effects and controls - which are always assessed during the process cycle.

In keeping with the goal of producing realistic and pleasant sound effects, the time taken to perform the user's subroutine should be kept under one millisecond (1000 microseconds).

An example of a JSR might be to process the note information through a table for frequency modulating the sound.

More information may be obtained by contacting Syntauri Ltd, 3506 Waverley Street, Palo Alto Ca 94306. (415) 494-1017.

<u>Frequency Modulation</u> : a modification to the alphaSyntauri music system.
by Laurie Spiegel

In order to provide a means of generating more complex timbres and repeating patterns of pitches from a single keypress, I implemented a software modification which permits the use of a form of digital frequency modulation to achieve these effects. Using dynamically changing variables already present in the system, this approach to more varied musical output is highly economical, adding little time or code to the system.

Various (and varying) values are added or not added to the pitch value currently to be outputted to hardware to provide this modulation. These values are drawn from addresses, storage locations where dynamically changing system variables reside in memory. (For example, the location of one byte of the instantaneous amplitude of the voice whose frequency is to be modulated.)

The address of the storage location from which the frequency offset is to be taken may be changed from the BASIC control program level through a 'poke' command. During any user-triggered (ASCII entry from the Apple keyboard) pass through the BASIC level program, the second of the two values required to produce the modulation is read from one of the Apple's game paddles. This value, an "FM mask", is stored and used to generate the modulating signal by being "AND'ed" with the dynamically changing variable discussed above. The result of the AND is then added to the current keyboard-derived pitch value as a varying offset, producing the frequency modulation.

The range of effects possible with the FM modification to the alphaSyntauri includes the following:

Use the high byte of the current voice's instantaneous amplitude:

1. No effect, normal alphaSyntauri operation, if value read from knob input is zero.
2. Continuous glissando changing of frequency in the same pattern as the envelope (similar to a common hardware analog synthesizer patch) if the knob value is 255.
3. Frequency modulation by any of 253 possible knob-selectable complex waveforms which result from sequences of logical AND's between the knob-defined mask and the changing envelope as updated by the envelope routine for the instrument currently selected.

   (This affords defferent timbres and effects for different envelope 'presets' too, and constitutes one reason why I originally implemented the bank of ten instrument presets loaded into RAM, as opposed to the single one which had previously been resident.)
4. Any of the above effects, but slowed down in tempo into the melodic rather than timbral range. This is achieved by holding down any ASCII key and the Apple's 'repeat' key, thereby forcing a full loop through the BASIC program to be introduced between changes of pitch which are produced by the frequency modulation code.

Due to the polyphonic nature of the instrument, dense textures of such timbres and patterns may be easily created, permitting the alphaSyntauri system to be used for musical effects more characteristic of synthesizers than of more conventional keyboard instruments. This textural and timbral potential provides a rich area of exploration which can be utilized and enjoyed with or without the use of conventional keyboard technique.

The wide range of completely replicable (though difficult to predict) effects are achieved quite economically as this FM modification requires only two additional parameters of control, and adds 26 bytes of 6502 code and two BASIC statements to the alphaSyntauri system.