Computer Music Journal

Volume 1 Number 4

PIANO and STRING TONE GENERATION

SCORES PRINTED from ANALYSIS of SOUND
produced by traditional MUSIC INSTRUMENTS

LSI-11

MICROCOMPUTER CONTROLLED DIGITAL SOUND SYNTHESIZER

A Portable Digital Sound Synthesis System

H. G. Alles
Bell Laboratories '
Murray Hills New Jersey 07974

A complete real time digital sound synthesis system has been constructed. In one compact unit (42" w x 25" h x 18"d, weighing ~300 lb). The following equipment has been included:

A. Digital Equipment Corp. LSI-11 based general purpose computing system with:

1. Two floppy discs with DMA controllers

2. A 64k word mapable memory for Table and 1/0 buffering

3. An ASCII AT&T gaphics video-terminal with full ASCII keyboard

B. A performer interface that samples and independently filters the position of 256 input devices with ~7 bit resolution (~100-200 different positions) at a 250hz sampling rate. The input devices include:

1. Two 61 key organ type manuals (the position of each key is measured with 7 bit resolution, 250 times/sec)

2. 72 slide levers

3. Four 3-axis joysticks

4. A variety of other things

C. A 16 bit digital synthesizer operating at 30k samples/sec with:

1. 32 FM sinewave oscillators (.002 hz frequency resolution and 14 bit accuracy)

2. 32 FM oscillators that directly generate the first N (l <N <l27)

harmonics of the specified frequency.

3. 32 completely programmable second order digital filters (two pole and two.zeros) that may be signal controlled

4. 32 AM (4 quadrant) multipliers

5. 256 envelope generators (linear or logarithmic)

6. A 2 second (48k word) digital reverberation and/or signal driven lookup table with 64 programmable taps

7. An array of 192 accumulating registers for interconnecting all the devices in any arbitrary way.

8. Four channels of 16 bit D/A output

9. Two channels of 14 bit A/D input

10. An array of 255 independent timers (1 ms resolution) with 16 FIFO's for sorting and storing timing events.

All the devices are bus interfaced to the LSI-11 computer and all the control words appear in LSI-11 address space (6k words). Approximately 1400 IC's are used in the entire system.

All of the system components have been designed to complement each other's capabilities. Special purpose hardware was constructed to perform those tasks which are repetitive and time consuming (timekeeping and performer input filtering).

Since there are no handwired connections between the input devices and the synthesizer hardware, and since synthesizer interconnections are accomplished through program loaded control registers, the whole system may be used in a variety of ways. For example:

A. All the control parameters may be specified in real time and at performance time.

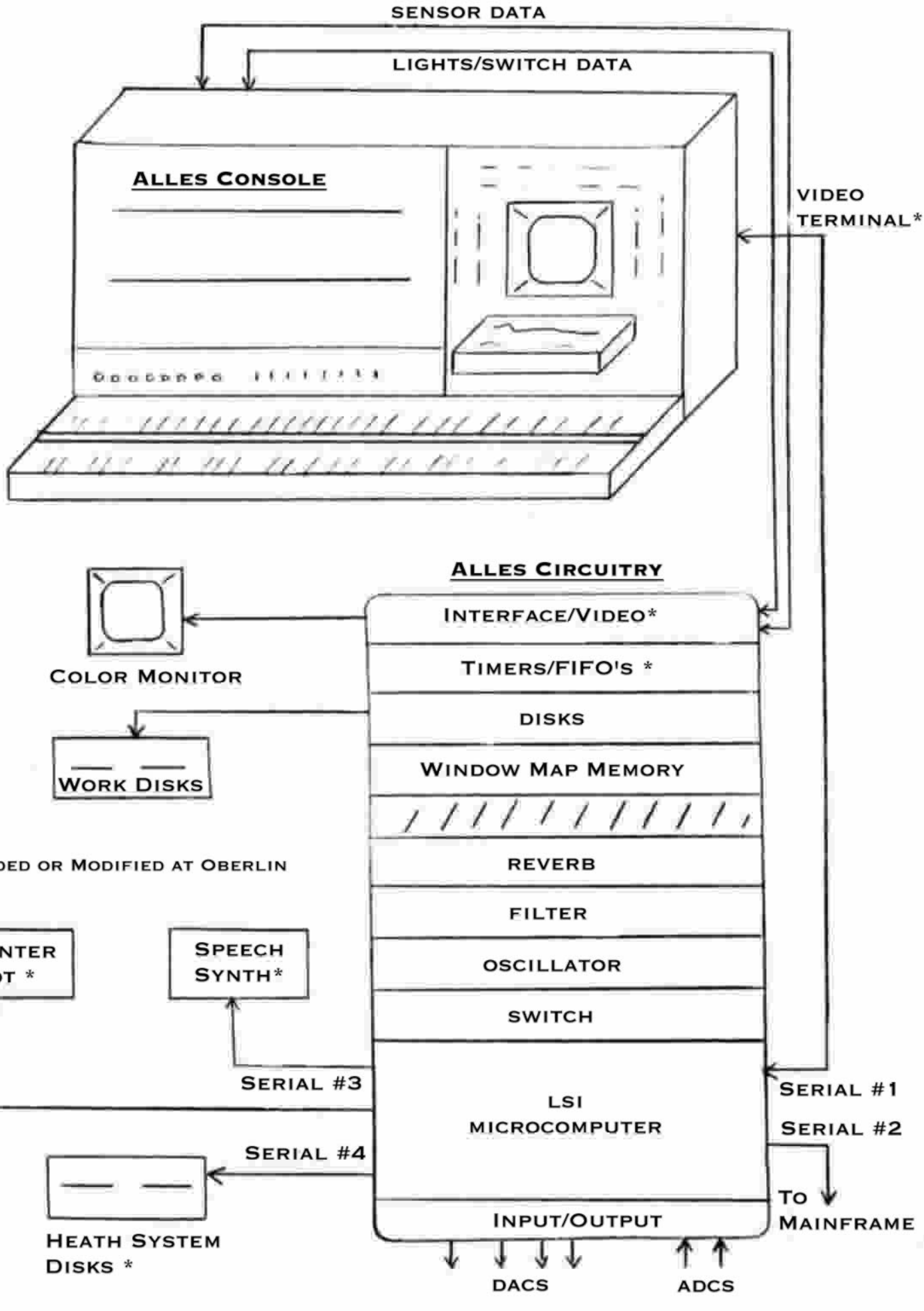
B. Several files may be prepared in real time, but before the

performance. Then at performance time the files may be played with some subset of the control parameters supplied during performance.

C . Files may be prepared and/or edited in nonreal time, incrementally improving the original performance.

The total real time synthesis capacity depends, of course, on the type of synthesis techniques and configuration used. The LSI-11 and floppy disc multiple file system can support ~1000 parameter changes/sec. These parameters may be used to specify frequencies, envelopes, configuration changes, graphic displays, etc. This data rate should be able to generate ~100 reasonably complex notes per second.

This system is perhaps the first representative of a new generation of musical instruments that combines in one relatively portable unit all the hardware and interfaces necessary to produce in real time and in a performance environment sounds approaching the complexity of a modest orchestra.


SENSOR DATA
LIGHTS/SWITCH DATA
ALLES CONSOLE
VIDEO TERMINAL*
ALLES CIRCUITRY
INTERFACE/VIDEO*
TIMERS/FIFO'S *
DISKS
WINDOW MAP MEMORY
REVERB
FILTER
OSCILLATOR
SWITCH
LSI MICROCOMPUTER
INPUT/OUTPUT
COLOR MONITOR
WORK DISKS
*ADDED OR MODIFIED AT OBERLIN
PRINTER PLOT *
SPEECH SYNTH*
SERIAL #3
SERIAL #4
SERIAL #1
SERIAL #2
TO MAINFRAME
HEATH SYSTEM DISKS *
DACS
ADCS

The Alles Machine Revisited

by

Gary Nelson, Director
and
John Talbert, Music Engineer

TIMARA Program
Conservatory of Music
Oberlin, Ohio
44074

1 Introduction

During the 1979-80 academic year, the Technology in Music and Related Arts Program at Oberlin College underwent a review and evaluation. As is our custom, a team of visiting referees was chosen from experts in the field. That team included Max V. Mathews, Director of Acoustics and Behavioral Research at Bell Laboratories in Murray Hill, New Jersey. A happy consequence of Mathews' visit was Oberlin's aquisition of the Alles Digital Sound Synthesizer.

In June 1980, we went to Bell Laboratories to consult with Mathews and the synthesizer's designer, H. G. Alles. After several weeks of asking questions and taking notes, we gathered up technical documentation, circuit diagrams, and the machine itself and headed back to Ohio to begin a challenging but rewarding period of what the seal of Oberlin College calls "learning and labor".

2 System Architecture

The "Alles Machine" was demonstrated and discussed at the 1977 International Computer Music Conference in San Diego. A description was published shortly thereafter in the COMPUTER MUSIC JOURNAL.

The system contains a DEC LSI-11 microcomputer which controls an array of realtime synthesis modules and user interface devices. The synthesizer was constructed on nine 10.5x8.5 wire-wrapped circuit boards which are housed in a special leaved cage which facilitates access for maintenance. The data registers of the synthesis modules are mapped into the top 8k words of the LSI-11 memory space.

A detailed inventory of system components follows below. Items with a single star indicate additions which have been made at Oberlin. Double stars identify components of the original system which we have substantially modified.

The general purpose computer consists of:

- 16-bit CPU with EIS and FIS
- 24kw of user memory (4kw added*)
- 64kw window-mapped memory
- 4 serial lines (2 lines added*)
- 2 RX-01 disk drives (480kb)*
- 2 DMA disk drives (616kb)**
- ASCII terminal**
- RT-11 operating system*

Special peripherals include:

- Microvox text-to-speech synthesizer*
- TI TMS-9918A video display processor (16kb memory)* with color monitor*
- Strobe 100 incremental plotter*
- Diablo daisy wheel printing terminal*

Synthesizer modules are:

- 4 16-bit digital-to-analog converters
- 2 14-bit analog-to-digital converters
- 32 sinewave oscillators
- 32 "harmonic" oscillators (sum of first N harmonics)
- 32 second-order filters
- 32 adder/multipliers
- 32 envelope/multipliers
- 256 ramp generators (linear or exponential)
- 64 delay lines (64kw memory)
- 640 element patch bay for synthesis modules
- 128 element patch bay for delay lines
- 26 trunk lines between patch bays
- 191 general-purpose patch cords
- 6 patch cords to DACs and ADCs
- 32 constant patch cords
- 1 white noise generator
- 255 down-counting timers with master control**
- 128 256-word ques (fifo) with hardware pointers*
- 256 256-word ques (fifo) for disk I/O buffering*

In the user interface we find:

- 2 61-key keyboards**
- 128 controls (sliders, joysticks, pedals, etc.)**
- 64 function keys**
- 16 switches
- 64 indicator lights (32 added*)

3 Initial Hardware Modifications

It was understood that the instrument was given to us "as is" and that we were to expect the problems associated with a prototype which had led a very rough early life. Our estimate was that the machine was approximately 75% functional on arrival in Oberlin.

Among the more serious hardware problems we addressed was the distribution of loads on two switching power supplies. By rearranging the modules we were able to equalize the loads and partition the system so that the computer could be used for general development without turning on the synthesizer. The ribbon cables which were used for power transmission were replaced with a heavier grade of cable to reduce power loss.

The sampling rate of the synthesizer was reduced from 30khz to 16khz. The loss of frequency response was offset by gains in several other areas. The higher clock rates of the original system approached the performance limit of wirewrap boards and of some of the TTL chips. In particular, the serial multiplier (AM2514) needed to be culled by the designers at Bell Laboratories to find those chips which would be fast enough for adequate operating margins. Our lower sampling rate reduced the performance requirements of the multipliers and, indeed, the whole system to a more reliable range. At the same time we were able to replace many of the high power Schottky circuits with equivalents requiring one-fifth the power.

We were able to remedy a serious timing problem by deriving the synthesizer's two basic clock frequencies from a single crystal. The 28mhz and 18mhz system clocks in the original design left little room for timing errors. They frequently drifted out of sync and caused general confusion in the communications among the synthesizer modules. The result was distinctly unmusical bursts of noise.

The Alles Machine was conceived as a portable system and therefore was housed in a single cabinet (weighing approximately 300

pounds). However, each time the machine is moved, a period of hardware failure follows. For the present, we have chosen to violate this basic design criterion to achieve a measure of reliability. We have separated the user console from the remaining components and placed the latter in a cabinet which contains additional cooling fans and noise isolation facilities.

4 Operating System

Concurrent with our program of hardware modifications, we confronted the issue of software. At Bell Laboratories, the programming of the Alles Machine was carried out with UNIX. Programs were written in C, compiled on a PDP-11/45, and down-loaded to the LSI-11 for execution. Some work was done directly on the LSI-11 with a version of UNIX which was not available in 1980 for proprietary reasons.

It was clear from the outset that we should adopt a system which conformed as much as possible to existing standards. To this end, we purchased a pair of RX01 disk drives and the RT-11 operating system . Since Oberlin has undergone a PASCAL boom of late, we realized substantial student programming assistance by adding a locally-modified version of that language to RT-11. PASCAL and LSI-11 assembly language now comprise our primary programming tools. A package of programs for communicating with our SIGMA9 and VAX 11/780 mainframes completed our software development package.

5 Subsequent Hardware Modifications

A period of systems analysis and software prototyping followed our initial repairs and modifications. We concluded that a number of additional hardware changes were needed to facilitate programming and to improve the performance of the system.

The original terminal combined text and graphics in an elegant but nonstandard way. We chose to separate these functions by converting the terminal to an ASCII standard and implementing color graphics with a Texas Instruments TMS-9918A video display processor and a separate color monitor. This change greatly simplified communication with the RT-11 operating system and removed the 2kw graphics bit map from the LSI-11 address space.

Several components of the synthesizer proved unreliable. The user interface which included the keyboard and sliders was particularly problematical. Each key and slider was an analog device with its own "personality". It was necessary to calibrate them frequently to produce even marginally predictable results. In the original design [Alles, 1977d] the keyboard and slider positions were converted into variable pulse width signals which were scanned by a high frequency clock. The separation of the user console from the computer and synthesizer required long cables which would not support this high frequency. In our solution, the analog voltages from the sensors are filtered, multiplexed, and transmitted over the long cable to a single analog-to-digital converter. This more direct design used lower frequency clocks with no loss of data speed. A higher degree of accuracy and increased stability were also realized.

The limited address space of the LSI-11 continues to be a serious weakness in the system. We improved our situation somewhat with the addition of 4k words on a MXV11-AA which fit into one of the vacant slots of the LSI-11's card cage. This card also included two additional serial lines which expanded our communication with the mainframes and peripherals.

It is often harder to figure out a complicated and insufficiently documented circuit or program than to design or write

a new one. We found this to be the case with the DMA disk system which came with the Alles Machine. The disks employed a nonstandard format and a controller driven by a ROM program which we were unable to decipher. A series of breakdowns led us to redesign around a more standard controller and Format. We were able to design a DMA (direct memory access) interface in which the disks can communicate directly with a buffer without stealing cycles from the CPU to take over the bus as was done in the original system.

Like the disks, the array of timers proved complicated and required reworking into a simpler model. A single master clock was provided to determine the rate at which 255 15-bit slave clocks would decrement. The resolution of the clocks is continuously variable from .25" to .ØØ1". Upon reaching zero, each clock may interrupt. An interrupt handler may read the clock number from a que and a data word from a register associated with that clock.

Queing of events became a central issue in our software strategies. The keyboard/slider interface, the envelope ramps, and the timers already contained hardware-driven ques which greatly facilitated realtime programming. We determined that additional, more general hardware ques would be useful. We were able to implement an array of 128 such ques in the space left on the simplied timer board. Each is a first-in-first-out (fifo) que with 256 entries and pointers maintained by hardware. The pointers were designed to be controllabl by software so that the ques might be used as lookup tables.

We are in the process of improving the performance of the DMA disks by implementing an array of 256 similar ques which contain a double porting architecture which permits access by the disk controller without the use of the LSI-11 bus. Data blocks of 512 bytes will be I/O buffered with simple commands under software control.

6 Programming Techniques

Our initial programming strategies followed very closely the model of Lawson and Mathews [1977]. We have implemented all of the their suggested modes of synthesizer control. These include:

- performance by keyboard player
- playback from a compiled score
- recording of performer input
- mixing of previous recordings with performer input
- algorithmic composition

Interaction with the performer is via keyboards, sliders, joy sticks, foot pedals, and graphics. The software is menu-oriented and entirely driven by interrupts. Operating modes and programs are selected with an array of push buttons. In a "help" mode, the user may select printed messages or messages spoken through a speech synthesizer. In the help mode, pressing buttons and moving control devices evokes an explanation of their function in the currently running program. Verbose messages are available for the beginner and terse messages may be selected by the more experienced user.

Recording and playback functions employ a network of ques, buffers, and floppy disks to maintain a constant rate of data flow. One disk functions as the input score and contains data which have been recorded during a previous run or which have been assembled with a composition program or music editor. The second disk is the output score. It contains the result of a real-time run in which prerecorded material may have been mixed with additional data supplied by the user or by a composition algorithm. Both disks are in a format which is compatible with text editing methods on the LSI-11 or on the mainframe.

Real-time activity in the user interface (keyboards, sliders, etc.) is transmitted through a hardware que which prevents data from being lost due to overhead in the interrupt handler. The data in this que informs the system of the device number and new data value which corresponds to the real-time user action. It is this data along with timing information that is recorded on the output score. Each datum from the que and each time value is a 16-bit word. The capacity of the disks thus permits more than 150,000 such actions to be recorded in the score. On playback, this information may be assigned to hardware timers and ques to simulate the realtime activity which produced it. The timers may interrupt and identify themselves in a que. A data word which is carried by each timer may then specifiy which general que is to be read. The device data is passed to the real-time play procedure and the timing information from the general que is used to restart the timer in preparation for the next event in the data stream.

The data from the user device que is actually two bytes, one for the device number and one for the value. the device number is used to identify a lookup table and the value is used as an index to that table. It is practical, therefore, to translate the user devices into any useful scale of 256 values. Since the lookup tables are not part of the recorded score, they may be modified or replaced. The score information may thus be interpreted differently in subsequent runs.

A primary design goal was to make timbre change dynamic so that orchestration could become part of the performance. (Instrument design remains a non-real-time activity.) Two files are used to this end. An envelope file contains a collection of normalized envelopes. Each envelope in the file is identified by number and each represents a function of time relative to the duration of the note to which it is applied. Data in the envelopes is a series of amplitude levels and transition rates. The rates may represent either real time (for attacks and releases, etc.) or relative time to provide elasticity. Since the prescaling of envelopes intended for frequency changes is somewhat different from that for amplitude, envelopes intended for frequency are flagged with negative envelope numbers. When an envelope file is called up at the beginning of a run, all of the envelopes are read into memory, normalised, converted to logarithmic form, and stored as a linked list for quick access during performance. Envelope scaling is accomplished at performance time through user devices or dynamic processes built into the instrument patch.

A timbre file consists of a collection of records which identify each timbre by number and which specify a list of envelopes which are to be applied to the parameters of the synthesis algorithm. Each algorithm or "patch" has its own expectations about the number of envelopes required and the correspondence between components of the patch and the order of envelope numbers presented in the timbre record. Timbre selection is made during performance by means of a slider. Up to 256 timbres may be available to the user. Changes in timbre take place at the beginning of the next note by reassigning envelope pointers. Notes in progress at the time of the timbre change continue with the timbre that was in place at the attack points of those notes.

We have decided for the present not to implement a patch language like MUSIC V or MUSIC 360. Rather, we are coding patches in PASCAL. A set of standard definitions of synthesizer registers and a library of PASCAL procedures seem to meet our needs. In addition, the requirement that our students deal with PASCAL follows the general philosophy of the TIMARA Program that musicians who use computers must attain a high level of programming expertise.

The combination of envelope and timbre files, patch definition in PASCAL, and real-time parameter scaling provides us with a fine

degree of interactive timbral control.

7 Applications Programs

Our principal applications at present derive from a concern for compatibility with programs running on our SIGMA9 mainframe. These programs include 48-voice frequency modulation orchestra (FMORCH), a "timbre tester", and the Musical Program Library (MPL). MPL is described elsewhere [Nelson, 1977]. Briefly, MPL is a package of music functions written in APL. The principle strengths of MPL are score editing and algorithmic composition. Since a large number of students and faculty at Oberlin are already familiar with MPL, we decided to provide a playback facility on the Alles Machine which would represent a subset of the synthesis we do in software on the SIGMA9. (A version of MPL without synthesis capabilities is in preapartion for the VAX 11/78Ø.) Data is transmitted between the mainframe and the floppy disks of the LSI-11 in the score file format mentioned earlier.

The Timbre Tester (TT) program is implemented in software on the SIGMA9 and provides a workable level of interactive instrument design. However, a similar program on the Alles Machine runs in real-time. The data relating to timbre may be transmitted from the Alles Machine to the SIGMA9 where it is translated into a format suitable for generating the same timbre with FMORCH. A future plan is a simulator of the Alles Machine on the mainframe. This would permit us to generate music in which the complexity or density exceeds real-time capabilities of the synthesis hardware.

Other applications include an emulator which is capable of digitizing or computing up to four seconds of sound into the memory of the reverb unit. This sound may then be sampled at various frequencies under keyboard control. Different sampling rates are accomplished by driving the pointers of the reverb delay memories with the phase registers of the hardware oscillators.

We plan to use the filter module to investigate vocoding and linear predictive synthesis. Fourier and other standard analysis techniques are also being implemented.

8 Conclusion

Much work remains before the Alles Machine can become a production system at Oberlin. The task has been difficult, time consuming, and often deeply frustrating. In spite of the confidence expressed by Max Mathews in arranging the gift, we were not entirely ready for the job. However, the oportunities brought about by the presence of the Alles Machine on our campus has fostered a strong working knowledge of such systems. It has enriched the TIMARA Program in ways not yet measurable and given us the means to comprehend and, perhaps, participate in future developments in computer music synthesis.

We plan a series of user manuals and technical reports which will explain our methods in more detail. For the present, we continue our efforts to understand and apply this marvelous tool in our primary vocation, the making of music.

9 References

H. G. Alles. "A Portable Digital Sound Synthesis System", COMPUTER MUSIC JOURNAL, 1977a, I(4), 5-6.

H. G. Alles. "A 256-Channel Performer Input Device", COMPUTER MUSIC JOURNAL, 1977d, I(4), 14-15.

J. Lawson and M. Mathews. "Computer Program to Control a Digital Real-Time Sound Synthesizer", COMPUTER MUSIC JOURNAL, 1977, I(4), 16-21.

G. Nelson. "MPL: A Program Library for Musical Data Processing", CREATIVE COMPUTING, March/April, 1977. 76-81.