The Music Kit on a PC

DAVID A. JAFFE, JULIUS O. SMITH and NICK PORCARRO
david@jaffe.com, jos@ccrma.stanford.edu and nick@ccrma.stanford.edu
Stanford University Office of Technology Licensing and
the Center for Computer Research in Music and Acoustics
Stanford University, Stanford, CA 94305

Abstract

We have recently ported the Music Kit to the Intel PC NEXTSTEP architecture, using inexpensive DSP and MIDI cards. We describe the port, which is soon to be released as Music Kit 5.0. We also introduce SynthBuilder, a graphic Music Kit instrument design and performance system. Finally, we discuss future plans for the Music Kit architecture.

What is NEXTSTEP?

In 1989, NeXT Inc. introduced NEXTSTEP, an operating system environment based on an object-oriented architecture. The promise to developers was an environment that forstered rapid application development with fewer lines of code and higher levels of reusability than ever before. In addition NEXTSTEP is an exceptionally convenient and powerful operating system for users. At that time, NEXTSTEP ran exclusively on NeXT's own hardware, which included a Motorolla DSP56001 signal processor, built-in 16-bit audio output, telephone quality microphone input, and two RS232 serial ports suitable for MIDI. In addition, NEXTSTEP included the Music Kit software package.

What is the Music Kit?

The Music Kit is an object-oriented software system for building music, sound, signal processing, and MIDI applications running on the NEXTSTEP operating system. It has been used in such diverse commercial applications as music sequencers, notation packages, computer games, and document processors. Professors and students in academia have used the Music Kit in a host of areas, such as music performance, scientific experiments, computer-aided instruction, and physical modeling (Jaffe, 1991). The Music Kit is the first to unify the MIDI and Music V paradigms, thus combining interaction with generality. It was developed by NeXT Computer, Inc. from 1986 to 1991 and has been supported since then by Stanford University and developers such as Pinnacle Research, Inc. For further information, see (Jaffe and Boynton, 1989) and (Smith et al., 1989). The Music Kit is available free of charge, as described at the end of this paper.

Is There Life After NeXT Hardware?

In 1993, NeXT announced discontinuation of its hardware line, choosing instead to focus on its software environment, which it has ported to a variety of architectures. The first port of NEXTSTEP was NS486, which runs on Intel 486 and Pentium chips. Soon after came NEXTSTEP for Hewlett-Packard HP-RISC. NeXT has recently announced ports to Sun and DEC architectures.

While we expect owners of NeXT hardware to continue to use the Music Kit on NeXT hardware, clearly the future of the Music Kit is to be found on other hardware architectures. We have chosen the Intel platform as our first port because of the many availability options and because of the price/performance advantages of the PC. We are also considering other ports, as discussed below.

Hardware Architecture

The 5.0 release of the Music Kit features support for both Intel and NeXT hardware. The Intel support is based on the strategy of providing DSP, sound I/O and MIDI via inexpensive external PC cards with which the Music Kit communicates via software device drivers. For ease of porting and backward compatibility, we chose DSP cards with hardware similar to that of the original NeXT. These cards also have features beyond that of the oringinal NeXT hardware, such as more memory, faster processor speed and built-in sound peripherals.

We currently support the Ariel PC56d (Ariel, 1994) and expect to soon support the ILink i56 (Ilink, 1994). Both are based on the Motorolla DSP56001 and support a DSP serial port connection compatible with that of the original NeXT hardware. This allows a variety of external peripherals to be used with the Music Kit, including the Ariel ProPort (high-quality DAC/ADC), the Ariel DatPort and Stealth DAI2400 (digital I/O), the Singular Solutions AD64x (high-quality ADC and digital I/O), etc.

Both the Ariel and ILink cards feature DSPs that run faster than the original NeXT hardware, allowing for more processing or synthesis in real time. In addition, both include built-in CoDecs for DAC/ADC. The following chart summarizes the features of the cards, in comparison with the NeXT hardware.

Hardware:	NeXT	Ariel PC56d	Ilink i56
DSP fast static RAM:	8 K, 32 K or 192 K words	16 K or 64 K words	8 K words
Processor speed:	25 mhz.	27 mhz.	33 mhz.
CoDec:	8 bit mono	14 bit stereo	16 bit stereo
DSP can send IRQ:	Yes (unused)	No	Yes

Both the Ariel and ILink cards are priced at under 500 US$ at the time of this writing.

For MIDI support, we use the standard MPU-401 architecture that is ubiquitous in the PC world. We currently support the MusicQuest MIDI cards. Depending on which version of the card you buy, it may include one or two MIDI inputs and one or two MIDI outputs.

Software Architecture

Portable File Formats

Intel x86 and Motorolla 68x00 processors differ in the order in which bytes are represented within words. Intel orders the bytes from low to high ("little endian"), while Motorolla orders them from high to low ("big endian"). This is no problem for ASCII files such as the Music Kit **.score** file. However, binary files must be carefully handled to allow users to freely move files between systems. The Music Kit uses the following bniary files:

1. **.dsp**—compiled DSP monitor files (load faster than **.lod** files)
2. **.sound**—sound files
3. **.playscore**—compiled score files (load faster than **.score** files)
4. **.midi**—Standard MIDI files

To prevent confusion, these files are *always* stored in big-endian order. This means there is a slight cost in reading a file on a little-endian system, since the bytes must be swapped. However, the swapping can be done quite rapidly and the overhead is not significant.

Portable Applications

Just as users may move files between architectures, they may also move applications. It is desirable for a given copy of a Music Kit application to work on both the NeXT and Intel machines. That way, a user can buy an application for his NeXT hardware and, when he buys an Intel machine, simply copy the file to his new computer.

To make this inter-operability possible, the Music Kit provides "fat libraries" and "fat applications." These are actually copies of the object code that includes both Intel and Motorolla machine code. The NEXTSTEP

operating system then automatically chooses the appropriate code to execute for a given architecture.

Custom DSP Monitors

Another aspect of making it possible to move applications from one computer is for the application to be self-contained—it should have no dependency on the Music Kit run-time library being installed on a given machine. To make this possible, beginning in release 4.0, we have made it possible to include the DSP run-time monitor in the application "wrapper", i.e. the directory containing the application resources. This monitor, with a **.lod** (ASCII) or **.dsp** (binary) file extension, then travels with the application as it is moved from one computer to another.

In addition, the Music Kit supports variable DSP memory configurations. Each memory configuration has its own DSP monitor file. Release 4.0 included monitors for the 32k and 8k configurations of the NeXT DSP, as well as monitors for the hub and satellittes of the Ariel QuintProcessor (Ariel, 1990). (The QuintProcessor is a five-DSP board for the NeXTcube, with the DSP arranged in a hub/satellite configuration. Each DSP has its own bank of fast static RAM and serial ports, and the hub processor has a large pool of DRAM.)

Release 5.0 introduces new monitors for the PC56d and i56 cards, as well as for the 192k expansion memory board for NeXT hardware. The Music Kit's Orchestra object automatically chooses a monitor that is optimal for a given machine. For examle, on NeXT hardware, it probes for the 192K memory expansion card and, if such a card is found, uses the 192k monitor. If it finds no 192k card, it checks for a 32k card. If no 32k card is found, the default 8k monitor is used. On Intel hardware, the monitor is chosen based on the kind of driver that is installed.

The support for different monitors, coupled with the wrapper search convention, makes it possible (and suggested) for an application to include a set of monitors that allows it to run optimally on a wide variety of computers.

Communicating with the DSP

Communication with the DSP on Intel hardware is through a loadable device driver. The Music Kit provides drivers for each of the cards it supports. The user can easily install the driver with the Configure application, which allows I/O ports and interrupt requests to be set from a convenient graphic interface. The Music Kit automatically searches for an installed driver of the type "DSP driver" and assumes there is a matching card installed. The only concern of the user is that he avoid collissions between the IO ports and interrupt requests of the various cards in his computer. This is a familiar headache for PC users. Recent "plug and play" architectures are beginning to address it. We hope to make the process of installing cards easier as the technology evolves.

The Music Kit's Orchestra object supports any number of DSPs, which made it easy to port to the Ariel QuintProcessor (Jaffe and Smith, 1992). On the Intel hardware, at the time of this writing, we support use of one DSP card at a time. This is really only a limitation of the way in which the Music Kit senses the presence or absence of the DSP drivers. It can be easily made more flexible if there is enough interest. We are hoping that PC cards with multiple DSP56001s, similar to the QuintProcessor, will be made soon and we look forward to supporting them. As an example of how we handle such a multi-DSP card, we examine briefly the Ariel QuintProcessor support. Support for multiple cards would follow a similar pattern:

The primary Music Kit class that supports the QuintProcessor is the *ArielQuintProcessor* class, which serves a dual purpose. First, it is a subclass of Orchestra that represents the hub DSP. As such, it can be sent Orchestra messages to allocate UnitGenerators, SynthPatches, etc. But it also represents the QuintProcessor as a whole and provides master control methods. The satellite DSPs are represented by instances of the subsidiary class, *ArielQPSat*, which is also a subclass of Orchestra. Creating an instance of ArielQuintProcessor automatically creates the associated ArielQPSat objects. A developer can chose to allocate DSP resources on a particular DSP by sending the appropriate allocation messages to the appropriate ArielQPSat or ArielQuintProcessor object.

Sound Input and Output

Our experience implementing sound output on the QuintProcessor paved the way for our approach to sound

output on the Intel hardware. On the QuintProcessor, we implemented direct sound output from the hub DS P serial port, rather than doing sound output the way it was done on the original NeXT hardware, where sound traveled first from the DSP to the main CPU (68040), then from the main CPU to the NeXT sound hardware. This has a number of advantages, including virtually eliminating latencies due to sound buffering, simplifying the software architecture, and reducing the load on the main CPU.

We took the same approach on the PC. The sound output, as well as the sound input, is done via the NeXT-compatible DSP port of the DSP cards. Alternatively, the codecs on the cards themselves may be used. In either case, the main CPU of the controlling computer has no responsibility with respect to sound output and input—the card handles everything.

To use the DSP serial port, the programmer sends sends the Objective-C message `setSerialPortOutput:YES` to the Orchestra object that represents the DSP. No change is necessary to the SynthPatch or UnitGenerator code. Since all SynthPatches already have an output UnitGenerator (such as *Out1aUG*), the DSP system simply routes the sound from this output to the serial port. To use the internal codec of the card, the programmer sends `setSerialPortOutput:NO`.

The *DSPSerialPortDevice* class encapsulates the details about the external device that is plugged into the DSP serial port. The Music Kit provides subclasses of DSPSerialPortDevice support various commercially-available devices. For example, if you have an Ariel **ProPort**, you simply send the message `setSerialPortDevice:[[ArielProPort alloc] init]` to the Orchestra object, which then defers to the *ArielProPort* object to set up the external device. The ArielProPort object also takes care of sending the appropriate commands to the DSP SCI port to set the sampling rate. The Music Kit's DSP system automatically handles half sampling rates. E.g. if the serial port device supports only 44100 and the sampling rate of the music is 22050, the Music Kit will automatically up-sample the sound data. If a hardware designer creates a new serial port device, he need only subclass DSPSerialPortDevice and override a few methods—the Music Kit then automatically supports the new device. One interesting new device is a quadraphonic interface, developed by Fernando Lopez-Lezcano, Stephen A. Davis and Atau Tanaka at CCRMA (Lopez-Lezcano, 1993) (Tanaka, 1991), which allows the Music Kit to generate four-channel sound.

The 4.0 Music Kit also added support for receiving 16-bit sound sent to the serial port from an external ADC such as the AD64x or ProPort, or an external AES/EBU interface such as the Stealth DAI2400. This sound input runs simultaneously with sound output, enabling incoming sound to be processed and sent back out the serial port.

Sound input support is enabled in a manner similar to sound output—you merely send `setSerialSoundInput:YES` to the Orchestra that represents the DSP. To create a sound-processing SynthPatch, you provide a sound input source UnitGenerator. For example, to receive sound input from the left channel, you include an *In1aUG* UnitGenerator object, just as you use an *Out1aUG* to send sound output to the left channel. The output of In1aUG can then be connected to any other UnitGenerators to create a signal processing network. Any number of In1aUGs may be instantiated; each gets a copy of the incoming sound so it is easy to create parallel banks of processing modules.

Since the SSI input and output may be used simultaneously, and since the SSI output path gets rid of buffer latency as described above, a real-time signal processor with no noticeable latency can be implemented using the Music Kit tools we have described.

MIDI

To support MIDI, we have written a device driver that provides the same functionality as the driver for NeXT hardware, including time-stamping of input bytes, timed output of time-stamped bytes, suppot for multiple MIDI cables and synchronization to incoming MIDI time code. The driver supports the standard NeXT MIDI driver API, allowing it to be used by both Music Kit and non-Music Kit applications.

Performance Evaluation

The greater speed of the higher-end Pentium systems, as well as the higher-performance DSP cards allow for an improvement in Music Kit efficiency. In particular, the Pentium speeds up such operations as inverse Fourier transforms, which are used in the Music Kit to convert from Partials objects (frequency-domain representation) to DSP wavetables (time-domain representation). Other operations that are improved include

soundfile mixing using the **mixsounds** command-line utility, parsing of long scorefiles, etc.

On both the NeXT and Pentium hardware, the DSP's "timed message queue" (TMQ), which controls the precise timing of events on the DSP, is of a fixed size and sometimes fills up. However, the larger amount of DSP static RAM on the PC56d allows for a larger timed message queue and thus minimizes the probability of the TMQ filling, in turn leading to better behavior than the NeXT hardware had with its minimum DSP memory configuration. The small size of the TMQ was enough of a problem on the NeXT hardware that we supported extensive buffering of DSP commands in the driver for NeXT hardware. While the larger memory size of the PC56d lessens the need for buffering on the main CPU, it remains to be seen whether it turns out to be necessary to add this extra level of buffering.

Even with a larger TMQ, in extreme situations, the TMQ may fill up. If that occurs, it is important for the driver to detect when room again becomes available in the TMQ. To accomplish this optimally, the DSP should be able to interrupt the main CPU to tell it there is room for more commands However, the PC56d card is unable, to generate interrupts. The DSP driver for the PC56d card must poll the DSP periodically, to see when room in the TMQ becomes available again. This is not only inefficient, it can lead to less-than-ideal response, since there may be some time between the time that room becomes available in the TMQ and the time that the driver wakes up and notices it's time to send more data. Other cards, such as the i56, support interrupts and should improve performance in this area.

SynthBuilder

One of the most exciting developments in the Music Kit project is a new graphic application for creating Music Kit instruments, configuring them to respond to MIDI, and performing them. Synthesizer "patches" are represented by networks consisting of digital signal processing elements called "unit generators" (Music Kit UnitGenerator objects) and MIDI event elements called "note filters" and "note generators" (Music Kit NoteFilter and Performer objects, respectively.) SynthBuilder is based on GraSP, a student project by Eric Jordan, created at Princeton University in 1992, with advisory assistance by the author (Jaffe), who was a visiting faculty. Since that time, it has been extensively developed by Nick Porcarro, in collaboration with the authors.

The graphical interface enables construction of complex patches without having to write a single line of code, and the underlying Music Kit software provides support for real-time DSP synthesis and MIDI. This means there is no "compute/then listen" cycle to slow down the process of developing a patch. It can be tried out immediately on a MIDI keyboard, and unit-generator and note-filter parameters can be adjusted in real time while a note is still sounding.

Sixteen bit stereo sound is synthesized immediately on the NeXT's built-in DSP56001 signal processing chip, and can be controlled from the user interface with software-simulated or physical MIDI devices. In addition to synthesis, the system supports configurations for sound processing via the DSP serial port as well as for sound output to DACs and other digital I/O devices.

MIDI control signals can be mapped to unit generator object control methods, permitting high-level control of patch parameters. For example, a MIDI key number can be readily mapped into frequency, and then mapped into a delay line length via a graphically constructed lookup table. A single MIDI event can be fed to (or through) multiple note filters, each of which can modify the event stream and/or control one or more unit-generator parameters.

Polyphony is handled in SynthBuilder by graphically specifying a voice allocation scheme. Optionally, a Music Kit SynthPatch can be generated and used in another application. Other types of code generation are possible, such as generic C code, or assembly code for another digital signal processor.

Dynamically loadable custom "inspectors" (user interfaces) can be created for patch elements. Dynamic loading promotes easy distribution and sharing of inspector modules, and promotes a fast, efficient development cycle. The process of creating a custom inspector is facilitated by a code generator, which takes a DSP assembly macro and a signal flow/parameter list specification as input, and ouputs working interface code which can then be customized.

As of this writing, SynthBuilder had more than 50 graphical custom inspectors, including an envelope editor, digital filter responses editors, and a MIDI event lookup table.

SynthBuilder is being used by researchers at the CCRMA to explore new synthesis techniques. It is now in

the alpha release stage on both NeXT 68040 based systems and Pentium systems.

Beyond the 56001

Recently-announced new versions in the DSP5600x series promise much greater compute power. Additionally, there are plans to modify SynthBuilder to generate code for a variety of other DSP chips. In addition, ss main CPU speeds continue to increase at their current rate, eventually it is likely that we will be able to run the Music Kit's synthesis and sound processing code on the main CPU itself, rather than on a DSP, using UnitGenerators written entirely in C. The object-oriented nature of the Music Kit makes such a change managable because all references to the DSP are localized in the Orchestra, UnitGenerator and SynthData classes. We have made a prototype of the low-level portion of such a system, with all current Music Kit DSP unit generators translated into C and have been able to use it to do some simple interactive real-time synthesis on a Pentium-based computer. One of the advantages of doing this excercise is that it enabled a comparison between the Pentium and DSP-based solutions. Currently, the DSP came out many times faster and has the advantage of being dedicated exclusively to sound production. Nevertheless, many more hours have gone into optimizing the DSP implementation than the Pentium implementation. If the Pentium unit generators were written in assembly language, they would probably be more efficient. We plan to watch closely the developments in the area of both DSPs and RISC chips and plan our migration path appropriately.

Availability

The Music Kit is available via ftp from **ccrma-ftp.stanford.edu** (email: musickit@ccrma.stanford.edu.) It is also available on CD ROM as part of the *Big Green CD ROM* at P.O. Box 471645, San Francisco, CA 94147 (email: **disc@skylee.org**, fax: 415 474 7896, phone: 415 474 7803.) At the time of this writing, the CD ROM contains version 4.0, which runs only on NeXT hardware. To subscribe to a Music Kit news group, send to **mkdist-request@ccrma.stanford.edu**.

References

(Ariel, 1994) Ariel Corp, 433 River Road, Highland Park, NJ 08904. (201) 249-2900, (201) 249-2123 fax.

(Ariel, 1990) Ariel Corp. *Ariel QuintProcessor Installation, Technical and Programming Manual.* Ariel Corp.

(Ilink, 1994) i*link. Kommunikationssysteme GmbH, Nollendorfstrasse 11-12, 10777, Berlin, Germany. phone: 49 30 - 216 20 48, fax: 49 30 - 215 82 74, mail: info@ilink.de.

(Jaffe, 1993) David A. Jaffe and Julius O. Smith. *Real Time Sound Processing & Synthesis on Multiple DSPs Using the Music Kit and the Ariel QuintProcessor.* Proceedings of the 1993 International Computer Music Conference, Tokyo, Japan.,

(Jaffe, 1990) David A. Jaffe. Efficient Dynamic Resource Management on Multiple DSPs, as Implemented in the NeXT Music Kit. Proceedings of the 1990 International Computer Music Conference, Glasgow, Scotland,Computer Music Assoc., pp. 188-190.

(Jaffe, 1991) David A. Jaffe. Musical and Extra-Musical Applications of the NeXT Music Kit. Proceedings of the 1991 International Computer Music Conference, Montreal, Canada, Computer Music Assoc., pp. 521-524.

(Jaffe, 1989) David A. Jaffe. An Overview of the NeXT Music Kit. Proceedings of the 1989 International Computer Music Conference, Columbus, Ohio, Computer Music Assoc., pp. 135-138.

(Jaffe and Boynton, 1989) David A. Jaffe and Lee Boynton. An Overview of the Sound and Music Kits for the NeXT Computer. Computer Music Journal, MIT Press, 14(2):48-55, 1989. Reprinted in book form in The Well-Tempered Object, ed. Stephen Pope, 1991, MIT Press.

(Jaffe and Smith, 1983) David A. Jaffe and Julius O. Smith. Extensions of the Karplus-Strong Plucked-String Algorithm. D. Jaffe and J. O. Smith. 1983. Computer Music Journal, 7(2):56-69. Reprinted in The Music Machine, ed. Curtis Roads, 1989, MIT Press, pp. 481-494.

(Lopez-Lezcano, 1993) F. Lopez-Lezcano. A four channel dynamic sound location system. Proceedings of the 1993 Japan Music and Computer Society.

(McNabb, 1990) Michael McNabb. Ensemble, An Extensible Real-Time Performance Environment. Proc. 89th Audio Engineering Society Convention, Los Angeles, CA, 1990.

(Smith et al., 1989) Julius O. Smith, David A. Jaffe and Lee Boynton. Music System Architecture on the NeXT Computer. Proceedings of the 1989 Audio Engineering Society Conference, L.A., CA.

(Tanaka, 1991) Atau Tanaka. Implementing Quadraphonic Audio on the NeXT, Proceedings of the 1991 International Computer Music Conference, Montreal, Canada, Computer Music Assoc.

Acknowledgements

Work on SynthBuilder and the corresponding Music Kit support was provided by the Stanford Office of Technology Licensing. Support for the Music Kit was provided by the Stanford Center for Computer Research in Music and Acoustics.