The MusicKit V5.5.2

Leigh M. Smith
Oz Music Code LLC. (http://www.ozmusiccode.com)

+61-4-0255-4184
leigh@leighsmith.com_nospam

These are overall comments (incorporating original notes by David A. Jaffe) and serve as the highest level overview of the MusicKit 5.5.2 — the “README”.

1. Description

The MusicKit is an object-oriented software system for building music, sound, signal processing, and MIDI applications. It has been used in such diverse commercial applications as music sequencers, computer games, and document processors. Professors and students in academia have used the MusicKit in a host of areas, including music performance, scientific experiments, computer-aided instruction, and physical modeling. The MusicKit was the first to unify the MIDI and Music V paradigms, thus combining interaction with generality (Music V, written by Max Mathews and others at Bell Labs four decades ago, was the first widely available “computer music compiler”).

The NeXT MusicKit was first demonstrated at the 1988 NeXT product introduction and was bundled in NeXT software releases 1.0 and 2.0. Beginning with NeXT’s 3.0 release, the MusicKit was no longer part of the standard NeXT software release. Instead, it was being distributed and supported as Version 4.0 by the Center for Computer Research in Music and Acoustics (CCRMA) of Stanford University. Versions 5.0 to 5.4.1 were then supported by tomandandy music, porting to several more popular operating systems.

The MusicKit Distribution is a comprehensive package that includes on-line documentation, programming examples, utilities, applications and sample score

documents. The MusicKit is dependent on the SndKit distribution, originally written by Stephen Brandon, and both Framework collections are available at the same distribution site. The SndKit was written to be a complete open source implementation of NeXTs SoundKit. The re-write started and almost finished before the SoundKit itself was released in source code form.

Source code is available for everything, with the exception of the NeXT hardware implementation of the low-level sound and DSP drivers. This means researchers and developers may study the source or even customize the Music Kit and DSP tools to suit their needs. Enhancements can be committed to the CVS repository to have them incorporated for future releases. Commercial software developers may freely incorporate and adapt the software to accelerate development of software products.

1.1. Feature List

The following is a partial list of the highlights of the MusicKit provided by David Jaffe on his web-page:

- Applicable to composers writing real-time computer music applications.
- Applicable to programmers writing cross-platform audio/music applications.
- Extensible, high-level object-oriented frameworks that are a super-set of Music V and MIDI paradigms.
- Written in Objective C and C, using Apple’s OpenStep/Cocoa API, the FoundationKit.
- Using the Python to Objective C bridge PyObjC (http://www.sourceforge.net/projects/pyobjc) enables applications and utilities to be written in Python (http://www.python.org), an interpreted object-oriented language.
- Functionally comparable (although architecturally dissimilar) to JMSL (Java Music Specification Language).
- Representation system capable of depicting phrase-level structure such as legato transitions.
- General time management/scheduling mechanism, supporting synchronization to MIDI time code.
- Efficient real-time synthesis and sound processing, including option for quadraphonic sound.
- Complete support for multiple MIDI inputs and outputs.

- Fully-dynamic DSP resource allocation system with dynamic linking and loading, on multiple DSPs.
- Digital sound I/O from the DSP port with support for serial port devices by all popular vendors.
- Non-real time mode, where the DSP returns data to the application or writes a sound file.
- Suite of applications, including Ensemble — an interactive algorithmic composition and performance environment (including a built-in sampler), and ScorePlayer — a Scorefile and standard MIDI file player.
- Library of instruments, including FM, wavetable, physical modeling and waveshaping synthesis.
- Library of unit generators for synthesis and sound processing.
- Documentation, programming examples, utilities, including a sound file mixer, sample rate converter, etc.
- ScoreFile, a textual scripting language for music.
- Connectable audio processing modules (“plugins”) including standard audio effects such as reverb.
- Sound data held in a specifiable variety of formats, i.e 8, 16, 24 bit or floating point. Allows trading off sample data size vs. processing time.
- MP3 file reading and writing. Decoding of MP3 can be done in a background thread after reading, or on-the-fly during playback, allowing selection of memory consumption versus processor load.
- MP3 and Ogg/Vorbis streaming of audio output to web servers using the libshout (http://www.icecast.org) library. The libshout library license is LGPL, not GPL and so do not compromise the MusicKit license

1.2. References at Other Sites

The MusicKit is listed on the following web sites and open source portals:

- 

sourceforge.net (http://www.sourceforge.net/projects/musickit) hosts the CVS repository, news, mail lists and web site.

-  freshmeat.net (http://www.freshmeat.net/projects/musickit/)
-  advogato.org (http://www.advogato.org/proj/MusicKit%20and%20SndKit/)
-  opensourcedirectory.org (http://www.opensourcedirectory.org/projects/musickit/)
- CCRMA's MusicKit site. (http://ccrma-www.stanford.edu/CCRMA/Overview/hardsoftware.html)
- David Jaffe's (the original author) MusicKit site. (http://www.jaffe.com/mk97.html)

1.3. MusicKit Mail-lists

- Announcements of new releases are sent to the <musickit-announce@lists.sourceforge.net_nospam> mail list. Subscribe to the list by visiting http://lists.sourceforge.net/mailman/listinfo/musickit-announce.
- Discussions on development of the MusicKit and notices of updates committed to the CVS source repository are sent to <musickit-developer@lists.sourceforge.net_nospam> mail list. Subscribe to the list by visiting http://lists.sourceforge.net/mailman/listinfo/musickit-developer.
- Discussions on writing applications using the MusicKit (not developing it) are sent to <musickit-users@lists.sourceforge.net_nospam> mail list. Subscribe to the list by visiting http://lists.sourceforge.net/mailman/listinfo/musickit-users.

Don't assume nothing is being done if you don't hear anything for a while! The maintainers are usually adding fixes/code, but full releases are less frequent. The CVS repository can be used to track all updates and obtain them as they are committed by developers.

2. Further Documentation

- Thorough documentation of classes, applications, usage and concepts are found under the directory `Documentation` in the source distribution.
- The file `Documentation/MusicKit_ChangeLog` (http://www.musickit.org/MusicKit_ChangeLog) lists changes and is generated from the CVS logs.

2.1. Online Documentation

- README in PDF (http://www.musickit.org/MusicKit_README.pdf) — This README document in the Adobe Portable Document Format (e.g Acrobat).
- MusicKit and SndKit Concepts (http://www.musickit.org/Concepts) (PDF version) (http://www.musickit.org/MusicKitConcepts.pdf) — Thorough documentation of the MusicKit and SndKit in operation.
- MusicKit Tutorials (http://www.musickit.org/TutorialClasses) (PDF version) (http://www.musickit.org/MusicKitTutorials.pdf) — Tutorial exercises by David A. Jaffe, used in his lecture series.
- Class Documentation (http://www.musickit.org/Frameworks) — Documentation for each module, C function and Objective-C class.

2.1.1. Published Papers

Conference papers and lecture slides in PDF or HTML are listed in the bibliography.

3. Contributors, History and Acknowledgements

- The MusicKit was designed and implemented by David A. Jaffe, and the DSP computer-music and array-processing software was designed and implemented by

Julius O. Smith III while at NeXT. The SoundKit (forerunner of the SndKit) was designed and implemented by Lee Boynton. Their original design appeared in [JaffeBoynton89].

- Michael McNabb brought wave table synthesis to the MusicKit and designed and built a number of `UnitGenerator` and `SynthPatch` subclasses and contributed the Ensemble application.
- Douglas Fulton was responsibile for the documentation and made substantial design improvements in clarifying general protocol and the identity and mechanisms of the classes.
- Gregg Kellogg wrote the DSP, Sound, and MIDI device drivers for the NeXT which were then maintained by Doug Mitchell.
- John Strawn wrote most of the matrix and array processing macros.
- Dana Massie contributed speech coding, sampling-rate conversion, and signal conditioning modules for the Sound Kit.
- Doug Keislar helped with testing, developer support, and demos.
- Mike Minnick helped finish the DSP array processing tools and wrote most of the programming examples.
- Roger Dannenberg influenced both the MusicKit noteTag design and the design of the performance mechanism (using a data flow discrete simulation model).
- Andy Moorer helped shape the `Envelope` strategy, suggested the unit-generator memory-argument scheme, and provided consultation.
- The software of William Schottstaedt and others at CCRMA (Stanford University) served as a model for some of the mechanisms in the MusicKit.
- James A. Moorer, Perry Cook, Rob Poor made code and design contributions also.
- Following NeXT's release of the source to Stanford in 1994, David did the port to Intel NeXTStep and the MPU-401 MIDI and DSP drivers. There were some other bug fix contributors (acknowledged in code comments).
- Stephen Brandon did the initial OpenStep port in early 1998 and the majority of the conversion work. The MusicKit now uses the SndKit, written by Stephen, rather than the SoundKit for its sound processing.
- Leigh M. Smith, <`leigh@leighsmith.com_nospam`> fixed bugs and ported the MusicKit and MIDI drivers to Intel and PowerPC Rhapsody in late 1998 and reorganised the packages and documentation for MacOS X-Server V1.2 and in 2000, to various developer previews of MacOS X. The frameworks were ported to

Windows 98/NT using DirectMusic in 1999. A sourceforge project (http://www.sourceforge.net/projects/musickit) was created in 2000.

- DirectSound interface code for Windows, MP3 encoding, much of the initial sound stream design and FX architecture was contributed by SKoT McDonald.
- Keith Hamel tested and helped in bug fixes for the MacOS X version.
- Matt Rice and Denis Crowdy both made valuable contributions to the Linux/portaudio port.
- Kim Shrier added FreeBSD support.

4. Supported Platforms

The MusicKit currently runs on the platforms described in Table 1:

Table 1. MusicKit Supported Platforms

Platform	MIDI	Sound	DSP
OpenStep 4.2/m68k (NeXT)	Y	Y	Y
OpenStep 4.2/Intel (with ISA 56k card)	Y	Y	Y
Windows XP (using GNUstep)	N	Y	N
MacOS X-Server V1.0-1.2	Y	Y	N
MacOS X	Y	Y	N
Linux/Unix (Using GNUstep, portaudio/portmusic)	N	Y	N

4.1. Portability

All the platform specific stuff is located in the `MKPerformSndMIDI` framework. The MusicKit and SndKit interface to `MKPerformSndMIDI` using a C API which was originally modelled after some NeXT API (although that is now history, particularly for

streaming audio). There are several versions of `MKPerformSndMIDI` underneath `MusicKit/Frameworks/PlatformSpecific`. You only compile one, all the versions are intended to produce one framework named `MKPerformSndMIDI.framework` which the MusicKit and SndKit link to.

The two versions under most active development are `MKPerformSndMIDI_MacOSX` and `MKPerformSndMIDI_portaudio`. The latter uses the multiplatform portaudio (http://www.portaudio.com) (and eventually portmidi) library to map down to the platform specific API and this is the means to interface with Linux and Windows.

4.2. MacOS X support

While there is now MacOS X support in portaudio, historically `MKPerformSndMIDI_MacOSX` existed before it and I've personally continued to use it to ensure MacOS X support is first class. All MIDI access is done using CoreMIDI API calls and all sound I/O is done using CoreAudio.

AudioUnits are supported. `SndAudioUnitProcessor` allows loading and using existing AudioUnits as a `SndAudioProcessor`, so you will be able to plug other developers AudioUnits into your SndKit sound processing stream.

4.3. Linux support

Linux/Unix support using the GNUstep (http://www.gnustep.org) library is partially completed and looking for more volunteers. A performance framework (`MKPerformSndMIDI_portaudio.framework`) using the portaudio (http://www.portaudio.com) library has been completed, porting the MusicKit to Intel Linux. Streaming sound works, but MIDI needs further work, please help! Potentially other Unix platforms will work (SGI, Solaris, FreeBSD, AIX etc) if they are supported in the portaudio library and GNUstep.

4.3.1. Installing GNUStep

Install GNUStep on your system in the usual fashion, according to the build guide. The following versions of GNUStep packages are known to work:

gnustep-make-1.6.0.tar.gz (http://www.gnustep.org/gnustep-make-1.6.0.tar.gz).

Install the library using:

```
tar xzvf gnustep-make-1.6.0.tar.gz
cd gnustep-make-1.6.0
sh ./configure --enable-import
sudo make install
```

> **Note:** We continue to use `#import` since it has been reinstated according to this email exchange (http://gcc.gnu.org/ml/gcc/2003-03/msg00269.html). Older **gcc** versions may produce spurious warnings.

gnustep-base-1.8.1.tar.gz (http://www.gnustep.org/gnustep-base-1.8.1.tar.gz).

Install the library using:

```
tar xzvf gnustep-base-1.8.1.tar.gz
cd gnustep-base-1.8.1
sh ./configure
sudo make install
```

gnustep-gui-0.8.5.tar.gz (http://www.gnustep.org/gnustep-gui-0.8.5.tar.gz).

Install the library using:

```
tar xzvf gnustep-gui-0.8.5.tar.gz
cd gnustep-gui-0.8.5
sh ./configure --disable-gsnd
sudo make install
```

4.4. Windows support

While other means of using the MusicKit on Windows is possible, the current recommended approach is using MinGW (http://www.mingw.org), the “Minimal Gnu for Windows”, together with GNUstep (http://www.gnustep.org) and the `MKPerformSndMIDI_portaudio` framework.

Currently the GNUstep AppKit support on Windows is incomplete, so your success running GUI applications will vary. However the MusicKit and SndKit run fine.

These instructions document the success I have had in getting MinGW, GNUstep, SndKit and MusicKit and all the supporting libraries compiled and running on a Windows XP system. Instead of repeating instructions for installation of each package that is better described by the package's documentation, I have listed the order and version number of each package that needs installing and the location of the documentation file. Please report problems you may encounter in getting things to build.

4.4.1. Install MinGW and friends

Install the minimal command line GNU system by running `MSYS-1.0.9.exe`. Install complier, libraries and headers by running `Mingw-3.1.0.exe`. Install into the `C:/msys/mingw` location. Install the CVS and other development tools `msysDTK-1.0.1.exe` Install `gcc-3.3.1.tar.gz`.

When installing MSYS, ensure that the user name or at least the home directory is a single name without a space, as this tends to break **configure**. This is further described in GNUstep's `gnustep/core/make/Documentation/README.MinGW`

4.4.2. Install libraries supporting GNUstep

- libtiff V3.6.0 (ftp://ftp.remotesensing.org/pub/libtiff/tiff-3.6.0-beta2) needs patching to accept ./configure --libdir and --includedir
- I couldn't get a version of libiconv to compile with MinGW, even libiconv 1.9.1 (http://www.gnu.org/software/libiconv-1.9.1.tar.gz) and had to settle for installing a binary version.
- libxml2 V2.5.11 (ftp://xmlsoft.org/libxml2-2.5.11.tar.gz)
- ffcall V1.8d (ftp://ftp.gnustep.org/pub/gnustep/libs/ffcall-1.8d.tar.gz)

4.4.3. Install GNUstep

Read the instructions in `gnustep/core/make/Documentation/README.MinGW`. I installed GNUstep from the head of the CVS tree. If you are working from the head of the CVS tree also use:

```
. /usr/GNUstep/System/Library/Makefiles/GNUstep.sh
```

instead of

```
. /usr/GNUstep/System/Makefiles/GNUstep.sh
```

Configure the `gnustep-gui` (until `libtiff` can install into the GNUstep tree):

```
./configure --with-tiff-library=/usr/local/lib --with-tiff-include=/usr/loca
```

4.4.4. Installation of Libraries Supporting SndKit

In addition to installing the common support libraries, the following steps need to be performed:

- The Microsoft DirectX 9.0b SDK (http://www.microsoft.com/downloads/details.aspx?displaylang=en&familyid=bc7ddedd-af62-493d-8055-5e57bab71e1a) (free) needs to be installed for DirectSound operation.
- The libogg needs patches (http://www.xiph.org/archives/vorbis-dev/200310/0068.html) to compile under MinGW. These patches may soon be incorporated into a libogg distribution newer than 1.0.
- The lame project needs patches (http://sourceforge.net/tracker/index.php?func=detail&aid=809315&group_id=290&atid=300290) to compile under MinGW. These patches may soon be incorporated into a lame distribution newer than 3.93.1.
- The libshout project needs patches (http://www.xiph.org/archives/icecast-dev/0660.html) to compile under MinGW. These patches may soon be incorporated into a libshout distribution newer than 2.0.

4.4.5. MIDI on Windows

The `MKPerformSndMIDI_portaudio` framework does not currently support MIDI. An earlier Windows only framework `MKPerformSndMIDI_Win32` which implements MIDI with the core layer of DirectMusic is available. The downloadable sound (DLS) capability of DirectMusic (providing a MIDI oriented sample playback API to PC

soundcards) is also supported when playing to a DirectMusic software synthesiser. There is minimal but usable support for downloading new DLS instruments to the software synthesiser. Eventually this functionality will be managed by portmusic (http://www-2.cs.cmu.edu/~music/portmusic/) cross platform in `MKPerformSndMIDI_portaudio.framework`

5. Supported Programming Languages

While the MusicKit and SndKit have been written in Objective C, this does not limit the system's use to that language. In particular, the dynamic binding of Objective C enables quite straightforward bridging to interpreters. This enables a relatively slow but "friendly" (i.e interactive) interpreter to benefit from the speed, multi-threading and feature set of the MusicKit and SndKit architectures.

Examples included in the distribution which are written in other languages include:

scoreinfo

A command-line tool written in Python to list MIDI channels (`MK_midiChan`) used by each part in MIDI or Scorefiles. This requires Python (http://www.python.org) and PyObjC (http://www.sourceforge.net/projects/pyobjc) to be installed.

6. Version numbering of the MusicKit

Version numbering of the MusicKit is as follows.

All versions are numbered in the popular standard GNU nomenclature *`V.R.P`*, referred here as a *version tuple*, referring to *`Version`*, *`Revision`* and *`Patch`* respectively. *`Version`* refers to major milestones in a project, such as a complete rewrite or major internal conversion, major functional improvement etc. *`Revision`*, refers to a minor milestone that can include new functionality that may break external interfacing software, forcing them to be updated (minimally a recompile). *`Patch`*, refers to a singular bug fix to correct operation which does not cause an incompatible change or introduce new behaviour.

Each framework has it's own version tuple. This is encoded in the framework and is used by the dynamic loader to verify the correct versions of applications link against

the correct versions of frameworks. However, to allow new framework with patches to be installed into operational sites without forcing recompiles of the application, the version tuple encoded in the framework is of the form `V.R`. Therefore a new patched framework can be installed over the top of the old same numbered framework with the knowledge it will continue to work.

6.1. CVS Branch Structure

Different software projects adopt different procedures for dealing with *branching* of the code base. The MusicKit project currently has few branches, however this is an attempt to define a policy for developers to conform to and to therefore be able to use when retrieving from the CVS tree. For example, some projects use branches for the "bleeding-edge" or experimental development, with the "head" or "trunk" of the code tree being stable. Positive experiments are then merged back into the head of the main branch of the tree.

The head of the MusicKit can be considered unstable, but should be considered to represent a concensus of development strategies. As explained above, any diverging approaches or experiments will be handled in sub-branch. Likewise, any patches to an existing release other than next impending release will be found in a sub-branch.

This can be counter intuitive. For example, when 5.4.1 is the most recent patch, it is found at the head of the tree. If however, eventually 5.5.0 was released, and then a patch needed to be made to 5.4.1, to be named 5.4.2, this would need to be made in a sub-branch at 5.4.1. In practice, the amount of branching has proven to be nearly zero. If indeed it starts to become necessary to simultaneously support an existing release and an developmental release, this will probably be at the release of a new version (i.e 5.5.0) and we can properly branch there. Comments are welcome!

7. License

7.1. Original MusicKit License

The source code as distributed by Stanford (V4.2) included the following usage message:

Music Kit Usage

The Music Kit software is distributed free of charge. Copyright remains with the owner indicated in each file. The software may be freely incorporated into any NeXTStep commercial application, any academic application, or any musical composition.

We would appreciate your acknowledging the use of the Music Kit in any academic paper, music program notes and application documentation that use the Music Kit.

7.2. License for MusicKit Post-4.2 Code

In regard of version V5.5.2 (consitituting all changes and modifications, porting efforts and documentation beyond V4.2), the current license follows.

The MusicKit software is distributed free of charge. Copyright remains with the owner indicated in each file and copyright of the modifications to each file remains with the contributor of the changes. The software in this distribution may be freely incorporated into any commercial application, any academic application, and public domain application or any musical composition so long as the attribution of authorship of the software is retained in the original source and documentation files.

By sharing your contributions back to the public domain, this helps others as well as yourself. It is not intended that your changes be enforced to be returned to the public domain, but it is hoped that common sense will prevail and that everyone will realise that reward does not mean avoiding openness.

We (the contributors to the MusicKit) would appreciate you acknowledging the use of the MusicKit in any academic paper, music program notes and application documentation that use the MusicKit or SndKit.

8. Downloading and Installing the Distribution

The MusicKit can be installed from a binary distribution package or compiled from the source code distribution tarball. In either case, certain libraries must be preinstalled for the MusicKit to operate.

8.1. Installing Supporting Libraries

The following libraries need to be installed prior to compiling the MusicKit source or before attempting to link against the MusicKit frameworks. These supporting libraries

may be installable using fink (http://www.sourceforge.net/projects/fink), or they can be individually downloaded and compiled as described below. Certain libraries may be optionally omitted (reducing functionality of the MusicKit) if necessary. These are noted below.

> **Note:** If installing on GNUstep, be sure that `PKG_CONFIG_PATH` is set to point to whereever you install the libraries listed below. For example, if you do install these in `/usr/local/lib`, `PKG_CONFIG_PATH` should be set to `/usr/local/lib/pkgconfig`. See **pkg-config** for details.

libogg-1.0.tar.gz (http://www.xiph.org/ogg/vorbis/download/libogg-1.0.tar.gz). Status: optional - disables `SndAudioProcessorMP3Encoder` if omitted.
libvorbis-1.0.tar.gz (http://www.xiph.org/ogg/vorbis/download/libvorbis-1.0.tar.gz). Status: optional - disables `SndAudioProcessorMP3Encoder` if omitted.

> The Ogg/Vorbis (http://www.xiph.org/ogg/vorbis) libraries provide patent-free Vorbis encoding/decoding of PCM audio to and from Ogg format bitstreams for use by libshout. Newer versions than 1.0 for Ogg and 1.0 for Vorbis will probably just work.
>
> Install the library using:

```
tar xzvf libogg-1.0.tar.gz
cd libogg-1.0
sh ./configure
sudo make install
cd ..
tar xzvf libvorbis-1.0.tar.gz
cd libvorbis-1.0
sh ./configure
sudo make install
```

libsndfile-1.0.11.tar.gz (http://www.mega-nerd.com/libsndfile/libsndfile-1.0.11.tar.gz). Status: Required.

> The libsndfile (http://www.mega-nerd.com/libsndfile) sound file I/O library provides sound file format conversion. Newer versions than 1.0.11 will probably just work.

Install the library using:

```
tar xzvf libsndfile-1.0.11.tar.gz
cd libsndfile-1.0.11
sh ./configure
sudo make install
```

lame-3.93.1.tar.gz (http://prdownloads.sourceforge.net/lame/lame-3.93.1.tar.gz?download). Status: optional - disables `SndAudioProcessorMP3Encoder` if omitted.

The LAME (http://www.sourceforge.net/projects/lame) library provides MP3 encoding/decoding capability. Newer versions than 3.93.1 will probably just work.

Install the library using:

```
tar xzvf lame-3.93.1.tar.gz
cd lame-3.93.1
sh ./configure
sudo make install
```

libshout-2.0.tar.gz (http://www.icecast.org/files/libshout/libshout-2.0.tar.gz). Status: optional - disables `SndAudioProcessorMP3Encoder` if omitted.

The libshout library (part of the IceS distribution) of the icecast (http://www.icecast.org) project provides the capability to stream MP3 or Ogg/Vorbis encoded audio to a broadcast server.

Install the library using:

```
tar xzvf libshout-2.0.tar.gz
cd libshout-2.0
sh ./configure --without-python --disable-dependency-tracking
make
sudo make install
```

pa_snapshot_v19.tar.gz (http://www.portaudio.com/archives/pa_snapshot_v19.tar.gz). Status: Needs to be installed on Windows, Linux and other Unixen except for MacOS X, OpenStep/NeXTStep systems.

At the moment, the portaudio CVS repository version is the only version that works on MinGW and so `MKPerformSndMIDI_portaudio` uses the upcoming V19 release. Either fetch the CVS snapshot, or using CVS, checkout the portaudio version 19 development branch. On Linux/BSD/SGI etc do:

```
cvs -d:pserver:anonymous@www.portaudio.com:/home/cvs co -r v19-devel por
cd portaudio
./configure --libdir=$GNUSTEP_SYSTEM_ROOT/Libraries
 --includedir=$GNUSTEP_SYSTEM_ROOT/Local/Libraries/Headers
sudo make install
```

On Windows XP do:

```
cvs -d:pserver:anonymous@www.portaudio.com:/home/cvs co -r v19-devel por
cd portaudio
./configure --libdir=$GNUSTEP_SYSTEM_ROOT/Libraries
 --includedir=$GNUSTEP_SYSTEM_ROOT/Local/Libraries/Headers
--with-winapi=directx
--with-dxdir=/c/DXSDK
sudo make install
```

libmp3hip-0.4pre1.tar.gz (http://icepick.info/projects/libmp3hip/libmp3hip-0.4pre1.tar.gz). Status: optional - disables `SndMP3` if omitted.

The libmp3hip library of the LAME (http://www.icecast.org) project provides the MP3 decoding

Install the library using:

```
tar xzvf libmp3hip-0.4pre1.tar.gz
cd libmp3hip-0.4pre1
sh ./configure --without-python --disable-dependency-tracking
make
sudo make install
```

8.2. Installing MusicKit Binaries

Binaries for the compiled frameworks for various operating systems reside on sourceforge as tar'ed `.pkg` (Apple/NeXT), or `.rpm` (Linux) packages. The files include the version number, choose the most recent release:

MK-5.5.2.b.MOX.pkg.tar
(http://prdownloads.sourceforge.net/musickit/MK-5.5.2.b.MOX.pkg.tar?download)

> The MacOS X frameworks (including the SndKit), applications, command line tools and documentation.

ZilogSCCMIDI.b.tar.gz
(http://prdownloads.sourceforge.net/musickit/ZilogSCCMIDI.b.tar.gz?download)

> The MacOS X-Server V1.2 MIDI driver binary for SCC UARTs in 8500/8600/9500/9600/G3/G4 PowerMacs.
>
> **Note:** This driver is not needed and will not work on MacOS X, only MacOS X-Server V1.2.

8.2.1. Installation Locations

The Installer program will let you choose a directory in which to place the binaries. If you are on a stand-alone machine, you should be logged in (or running Installer) as root and choose "`/`" for the installation directory (the default).

The frameworks and applications are installed into the appropriate platform locations given in Table 2.

Table 2. MusicKit Install locations

Platform	Frameworks install location	Applications install location
OpenStep 4.2 (m68k,ix86)	`/LocalLibrary/Frameworks`	`/LocalApps`
MacOS X-Server V1.0-1.2	`/Local/Library/Frameworks`	`/Local/Apps`
MacOS X	`/Library/Frameworks`	`/Applications`
GNUStep (on Unixen)	`/usr/GNUstep/Local/Library/Frameworks`	`/usr/GNUstep/Local/Applications`
GNUStep (on Windows)	`C:\GNUstep\Local\Library\Frameworks`	`C:\GNUstep\Local\Applications`

Command line tools and manual pages will install into the `/usr/local` hierarchy on Unix type machines, specifically `/usr/local/bin`. Locations for command line Windows tools have yet to be determined.

If you are the system administrator for a NFS shared network, you probably want to choose a local directory which can be exported and create symbolic links into that directory on the networked machines. Alternatively, if you have a single server that exports `/Local*` and `/usr/local`, simply install the package there and you're done.

8.3. Installing From The Source Tarball

The source distribution is located at:

MK-5.5.2.s.tar.gz
(http://prdownloads.sourceforge.net/musickit/MK-5.5.2.s.tar.gz?download)

> The source to `MusicKit`, `SndKit`, `MKDSP`, `MKPerformSndMIDI`, `MKUnitGenerators` and `MKSynthPatches` frameworks and all utilites, applications, documentation and example code.
>
> This is the most recent revision and patch. Some older revisions are located at the sourceforge site (http://prdownloads.sourceforge.net/musickit) for regression testing.

ZilogSCCMIDI.s.tar.gz
(http://prdownloads.sourceforge.net/musickit/ZilogSCCMIDI.s.tar.gz?download)

The MacOS X-Server V1.0-1.2 MIDI driver source.

Note: This is not needed for MacOS X.

DriverKitHeadersForMOXS_V1.2.tar.gz
(http://prdownloads.sourceforge.net/musickit/DriverKitHeadersForMOXS_V1.2.tar.gz?download)

NeXTStep V3.3 headers required to compile the driver on MacOS X-Server V1.2.

Note: This is not needed for MacOS X.

8.4. Retrieving Sources From The CVS Repository

The bleeding edge source is obtainable from the CVS repository on sourceforge (http://www.sourceforge.net) and is described in detail by this sourceforge page (http://sourceforge.net/cvs/?group_id=9881).

8.5. Compiling The Sources

The MusicKit installation process is the now nearly the same as most autoconf configured software. Trivially, the following commands will compile and install the MusicKit.

```
./configure
make
sudo make install
```

There are a couple of subtle variations to this. On GNUstep systems, **make** will build the MusicKit sources as expected. On MacOS X, to benefit from the Xcode IDE (the IDE, plus zero-linking), **make** will just run the **xcodebuild** command line tool which does the actual building using the `MusicKit.xcode` project. If **configure** determined some libraries were unavailable, they will be omitted when **xcodebuild** is run. Since **xcodebuild install** builds and then installs, the **sudo make install** suffices, typing a preceding **make** is not actually necessary.

The MusicKit project can also be built within the Xcode IDE GUI. However **configure** must be run before building with Xcode, in order to configure headers to selectively compile sources based on library availability. In addition, the default configuration is to assume if the Xcode GUI is being used, then the MusicKit is being developed, not just installed and that all libraries have been installed. If all of the libraries are not available, the build setting CONFIGURED_LIBS must be changed for the SndKit to remove those libraries which are not installed. In practice, the user must manually modify CONFIGURED_LIBS build setting in the SndKit target to match the setting in `Makefile`.

The `MKUnitGenerators` and `MKSynthPatches` frameworks won't compile on platforms other than OpenStep or NeXTStep because the Motorola **asm56000** and **lnk56000** tools are missing.

8.5.1. Installing a MIDI driver

If necessary, compile and install an appropriate MIDI driver.

To compile the MacOS X-Server V1.2 Zilog SCC MIDI driver, some header files are missing. You can get them from the Darwin distribution of the SoundKit, copy them from a NeXTStep 3.3/OpenStep 4.2 system, or as a convience, they are made available for download.

MacOS X Developer release includes source for a generic USB MIDI driver at `/Developer/Examples/CoreAudio/MIDI/SampleUSBDriver`. There is also a driver binary for the MIDIMan MIDISPORT™ USB interfaces for MacOS X available for download from MIDIMan's web page (http://www.midiman.com) and other vendors may well have drivers available for their hardware.

9. Included Applications and Example Code

9.1. Applications in the distribution

- ScorePlayer (David Jaffe)
- WaveEdit (David Jaffe)

  WaveEdit is an application that makes it possible to display, edit and listen to wave tables.

- Ensemble (Michael McNabb)

  Ensemble combines elements of a sequencer, a voicing application and an algorithmic composition application.

- EnvelopeEd (Fernando Lopez Lezcano)
- PianoRoll (Jonathan Knudsen)
- HosePlayer (Perry Cook)
- SlideFlute (Perry Cook)
- edsnd (Lee Boynton)
- Reich-o-Matic (Brad Garton)
- Looching (Brad Garton)
- ResoLab (Perry Cook)
- ClariNot (Perry Cook)
- ResonSound (David Jaffe)

  Real time processing of sound from the DSP serial port.

- PatchCord (Leigh Smith)

  A system exclusive librarian and patch editor.

9.2. Example code in the distribution

There are two kinds of programming examples. Programs with names entirely in lowercase, such as **playscorefile**, are command-line programs. Programs with names beginning with an uppercase letter, such as PlayNote, are applications (i.e. graphic-interface programs).

The complete set of programs is listed below. For further information on a given programming example, see the README file in its directory.

Simple command-line programming examples

playpart

Create notes algorithmically and play them.

playscorefile

Read a scorefile into a `MKScore` and play it on the DSP.

playscorefile2

Read a scorefile and play it on the DSP as it is read.

playscorefilemidi

Play scorefile through MIDI out.

mixscorefiles

Mix any number of scorefiles and write the result out.

process_soundfiles_dsp

Process a sound file through the DSP (non-real-time). Includes `MKSynthPatch`es for resonating and enveloping sounds.

mixsounds

Soundfile mixer that shows how to make your own `MKInstrument` (non-real-time).

exampleSynthPatch

Demonstrates how to build a `MKSynthPatch` and play it.

exampleUnitGenerator

Demonstrates how to build a `MKUnitGenerator`.

example1

Simple `MKNote` and `MKScore` generation.

example2

Demonstration of playing `MKNote`s.

example3

Simple algorithmic melody generation.

example4

Simple algorithmic melody generation using `MKPart`.

example5

Demonstration of `MKUnitGenerator`s.

Simple application programming examples

Metronome

Simple note playing.

MidiFilePlayback

Play MIDI files with samples using `MKSamplerInstrument`.

MidiEcho

Take MIDI in, generate echoes, and send to MIDI output.

MidiLoop

Take MIDI input and send it right out MIDI again.

MidiPlay

Take MIDI input and play the DSP.

MidiRecord

Read MIDI input into a `MKScore` obj, write a scorefile.

PerformerExample

Adjust algorithmically-generated music playing on DSP.

PlayNote

Click a button to play and adjust notes on the DSP.

QuintProcessor

Interactive application for the Ariel QuintProcessor.

SineGen

Interactively adjust the frequency of a sine wave.

10. What works?

The frameworks (SndKit, MusicKit) have been exercised fairly extensively and are quite stable.

MIDI recording and playback of scores and computed parts works on all supported platforms. The application ScorePlayer, the examples MidiFilePlayback and **playscorefilemidi**, and the utility **playscore** demonstrate these capabilities.

Sound recording and playback is also stable. The applications Spectro and TwoWaves, and the commands **playsnd**, **recsnd** demonstrate Sound recording/performance.

11. What doesn’t work? / What needs doing?

- The current list of tasks to be done is listed on the sourceforge Task Manager (http://www.sourceforge.net/projects/musickit).
- DSP on non-56K DSP systems. Stephen mentioned he got sound out under OS4.2. However none of the DSP drivers have been ported from NeXTStep V3.3 (as they are all ISA bus cards and I don’t know of a PCI 56K card with published interface specs).

  A more fruitful avenue seems to be to convert the unit generators to the MPEG Layer 4 structured audio language SAOL (http://www.saol.net) [ScheirerVercoe99] and

modify the MKDSP framework to download to a supporting card or do the emulation using the native processor. If emulating with the native processor, it would then be possible to use Apples AltiVec vector libraries for increased performance.

- For this current revision the `.nib`s have been upgraded to work with MacOS X and MacOS X-Server and will not load with OpenStep 4.2. The `.nib`s which are compatible with OpenStep 4.2 are now named *`nibname`*`-openstep.nib`. At the moment you will need to manually symbolically link these to compile for OpenStep 4.2. Eventually Apple's InterfaceBuilder will produce XML based nibs which should reduce the compatibility issue.

12. API Changes in MusicKit V5.X frameworks

The changes from Version 4 to Version 5 of the MusicKit mostly consists of conversion to the OpenStep specification from the older NeXTStep API.

12.1. Frameworks replace libraries

Libraries have now been replaced as the following frameworks:

- MusicKit.framework
- MKDSP.framework
- MKPerformSndMIDI.framework
- MKSynthPatches.framework
- MKUnitGenerators.framework

12.2. Include Files

The root hierarchical include file is now `#import`ed as `<MusicKit/MusicKit.h>` replacing `<musickit/musickit.h>`.

Unit generator headers are included from `MKUnitGenerators.framework`, not `musickit/unitgenerators`.

12.3. Class naming conventions

All the public MusicKit classes are now prefixed with `MK` to match the Foundation/AppKit model, ie `MKNote`, `MKOrchestra` replaces `Note` and `Orchestra`. In a similar manner to the changes in other frameworks when OpenStep-ified, as well as method name changes, there are object allocation changes, generally `+new` is now an appropriate name returning an autoreleased instance, i.e `+score` for `MKScore`, `+note` for `MKNote` etc. `NSStrings` replace `char *` where ever possible.

12.4. Class Specific Changes

12.4.1. `MKMidi`

- `+new` has been renamed `+midi` in keeping with OpenStep conventions.
- `allocFromZone:onDevice:` and `allocFromZone:onDevice:hostName` have been replaced with corresponding `-initOnDevice:` and `-initOnDevice:hostName:` instance methods and `+midiOnDevice:`, `+midiOnDevice:hostName:` class methods to support OpenStep allocation conventions.
- The methods `-noteSenders` and `-noteReceivers` don't return a copy of the array. The parent object is expected to copy it.

13. Contributing Fixes

Please! Changes are maintained via an open CVS system (http://sourceforge.net/cvs/?group_id=9881). If you send me <`leigh@leighsmith.com_nospam`> changes I will incorporate them and check them in. If you are planning to do major work, I can register you as a developer to provide write access to the CVS server.

If you make a MusicKit program that you'd like us to consider for distribution as part of the MusicKit, DSP and SndKit Distribution, please send it to the following address: <`leigh@leighsmith.com_nospam`>. We are interested in applications, unit

generators, synth patches, etc. If you do not have e-mail access, you can send media to the address of the author at the start of this document.

If you have discovered a bug, please report it via the sourceforge bug tracker.

14. Supporting The MusicKit Project by Donations

The MusicKit project is run by volunteers who donate their development time, equipment and personal money instead of working on paying jobs. While we prefer contributions of development time and of code, monetary donations allow us in a very real sense to continue to develop the MusicKit for the benefit of all your projects, including commercial projects.

Please consider donating to the MusicKit project to keep releases and bug fixes regularly delivered. You can make online donations via the sourceforge donations system (http://sourceforge.net/project/project_donations.php?group_id=9881).

The MusicKit Project developers are also available for contracting work to add substantial features to the MusicKit, to consult on integration of MusicKit technologies into other projects and to work on other complimentary projects. Please direct questions to the project administrator <leigh@leighsmith.com_nospam>.

Published Documentation

This section cites documentation related to the MusicKit which has been published.

Keith Hamel “NoteAbility: A Music Notation System Combines Musical Intelligence with Graphic Flexibility” *Proceedings of the 1994 International Computer Music Conference* 303-6 1994 International Computer Music Association

David A. Jaffe “An Overview of the NeXT Music Kit” *Proceedings of the 1989 International Computer Music Conference* 135-8 1989 International Computer Music Association

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* 188-90 1990 International Computer Music

Association OrchestraDetails.pdf (http://www.musickit.org/Publications/OrchestraDetails.pdf)

An in depth discussion of the MusicKit `MKOrchestra`'s dynamic allocation system, DSP memory management, and multi-DSP support.

David A. Jaffe "Musical and Extra-Musical Applications of the NeXT Music Kit" *Proceedings of the 1991 International Computer Music Conference* 521-4 1991 International Computer Music Association MusicApplications.pdf (http://www.musickit.org/Publications/MusicApplications.pdf)

Paper by David Jaffe from the 1991 International Computer Music Conference describing some of the uses to which the MusicKit has been put.

David A. Jaffe and Julius O. Smith, III "Real Time Sound Processing & Synthesis on Multiple DSPs Using the Music Kit and the Ariel QuintProcessor" *Proceedings of the 1993 International Computer Music Conference* International Computer Music Association 1993 MusicKit-NWExpo93.pdf (http://www.musickit.org/Publications/MusicKit-NWExpo93.pdf)

Documents multiple DSP operation.

David A. Jaffe and Lee R. Boynton "An Overview of the Sound and Music Kits for the NeXT Computer" *Computer Music Journal* 13(2) 1989 48-55 MIT Press Reprinted in book form in *The Well-Tempered Object* MIT Press 1991 Edited by Stephen Pope

Stephen Brandon and Leigh M. Smith "Next Steps from NeXTSTEP: MusicKit and SoundKit in a New World" *Proceedings of the 2000 International Computer Music Conference* 503-6 2000 International Computer Music Association ICMC2000.pdf (http://www.musickit.org/Publications/ICMC2000.pdf)

Most recent publication describing the OpenStep conversion and cross-platform approach.

Michael McNabb “Ensemble, An Extensible Real-Time Performance Environment” *Proceedings of 89th Audio Engineering Society Convention* 1990 Audio Engineering Society

Julius O. Smith, III, David A. Jaffe, and Lee R. Boynton “Music System Architecture on the NeXT Computer” *Proceedings of the Audio Engineering Society Conference* 1989 Audio Engineering Society SoundMusicDSP.pdf (http://www.musickit.org/Publications/SoundMusicDSP.pdf)

A general (if dated) overview.

Julius O. Smith, III “Computer Music on the DSP56001” *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics* October, 1989 IEEE DSPSoftwareDesign.pdf (http://www.musickit.org/Publications/DSPSoftwareDesign.pdf)

Transparencies for a talk describing the use of the DSP56001 for music synthesis.

Julius O. Smith, III “Unit-Generator Implementation on the NeXT DSP Chip” *Proceedings of the 1989 International Computer Music Conference* 303-6 1989 International Computer Music Association

Atau Tanaka “Implementing Quadraphonic Audio on the NeXT” *Proceedings of the 1991 International Computer Music Conference* 1991 International Computer Music Association

Fernando Lopez-Lezcano “A Four Channel Dynamic Sound Location System” *Proceedings of the 1993 Japan Music and Computer Society* 1993 Japan Music and Computer Society

Suggested Reading

This section indicates where you can get additional information, whether in printed form or on-line.

Signal and Array Processing

Lawrence R. Rabiner and Bernard Gold *Theory and Application of Digital Signal Processing* Prentice-Hall 1975

L. E. Franks *Signal Theory* Prentice-Hall 1969

Gene H. Golub and Charles F. Van Loan *Matrix Computations* 1983 John Hopkins University Press

G. W. Stewart *Introduction to Matrix Computations* Academic Press 1973

Ronald N. Bracewell *The Fourier Transform and Its Applications* McGraw-Hill 1978

Sound

John Backus *The Acoustical Foundations of Music, Second edition* Norton 1977

B.A. Blesser " Digitization of Audio: A Comprehensive Examination of Theory, Implementation and Current Practice " *Journal of the Audio Engineering Society* 26(10) pp. 739-771 1978

Music

Charles Dodge and Thomas A. Jerse *Computer Music* Schirmer Books 1985

Hal Chamberlain *Musical Applications of Microprocessors* Hayden 1980

Hubert S. Howe, Jr. *Electronic Music Synthesis* Norton 1975

Bill Schottstaedt " The Simulation of Natural Instrument Tones using Frequency Modulation with a Complex Modulating Wave " *Computer Music Journal* 1(4) pp. 46-50 1977

Marc Le Brun *Digital Waveshaping Synthesis* Journal of the Audio Engineering Society 18(2) pp. 250-266 1979

Eric D. Scheirer and Barry L. Vercoe “SAOL: The MPEG-4 Structured Audio Orchestra Language” *Computer Music Journal* 23(2) 1999 31-51 MIT Press

SourceForge kindly hosts the MusicKit. (http://sourceforge.net)