# $Id: $

#

# For a "How-To" please refer to the Portaudio documentation at:

# http://www.portaudio.com/trac/wiki/TutorialDir/Compile/CMake

#

CMAKE\_MINIMUM\_REQUIRED(VERSION 2.8)

# Check if the user is building PortAudio stand-alone or as part of a larger

# project. If this is part of a larger project (i.e. the CMakeLists.txt has

# been imported by some other CMakeLists.txt), we don't want to trump over

# the top of that project's global settings.

IF(${CMAKE\_SOURCE\_DIR} STREQUAL ${CMAKE\_CURRENT\_LIST\_DIR})

IF(NOT CMAKE\_BUILD\_TYPE AND NOT CMAKE\_CONFIGURATION\_TYPES)

MESSAGE(STATUS "Setting CMAKE\_BUILD\_TYPE type to 'Debug' as none was specified.")

SET(CMAKE\_BUILD\_TYPE Debug CACHE STRING "Choose the type of build." FORCE)

SET\_PROPERTY(CACHE CMAKE\_BUILD\_TYPE PROPERTY STRINGS "Debug" "Release")

ENDIF()

PROJECT(portaudio)

SET\_PROPERTY(GLOBAL PROPERTY USE\_FOLDERS ON)

IF(WIN32 AND MSVC)

OPTION(PA\_DLL\_LINK\_WITH\_STATIC\_RUNTIME "Link with static runtime libraries (minimizes runtime dependencies)" ON)

IF(PA\_DLL\_LINK\_WITH\_STATIC\_RUNTIME)

FOREACH(flag\_var

CMAKE\_C\_FLAGS CMAKE\_C\_FLAGS\_DEBUG CMAKE\_C\_FLAGS\_RELEASE

CMAKE\_C\_FLAGS\_MINSIZEREL CMAKE\_C\_FLAGS\_RELWITHDEBINFO

CMAKE\_CXX\_FLAGS CMAKE\_CXX\_FLAGS\_DEBUG CMAKE\_CXX\_FLAGS\_RELEASE

CMAKE\_CXX\_FLAGS\_MINSIZEREL CMAKE\_CXX\_FLAGS\_RELWITHDEBINFO)

IF(${flag\_var} MATCHES "/MD")

STRING(REGEX REPLACE "/MD" "/MT" ${flag\_var} "${${flag\_var}}")

ENDIF()

ENDFOREACH()

ENDIF()

ENDIF()

ENDIF()

SET(PA\_PKGCONFIG\_VERSION 19)

# Most of the code from this point onwards is related to populating the

# following variables:

# PA\_PUBLIC\_INCLUDES - This contains the list of public PortAudio header

# files. These files will be copied into /include paths on Unix'y

# systems when "make install" is invoked.

# PA\_PRIVATE\_INCLUDES - This contains the list of header files which

# are not part of PortAudio, but are required by the various hostapis.

# It is only used by CMake IDE generators (like Visual Studio) to

# provide quick-links to useful headers. It has no impact on build

# output.

# PA\_PRIVATE\_INCLUDE\_PATHS - This contains the list of include paths which

# will be passed to the compiler while PortAudio is being built which

# are not required by applications using the PortAudio API.

# PA\_PRIVATE\_COMPILE\_DEFINITIONS - This contains a list of preprocessor

# macro definitions which will be set when compiling PortAudio source

# files.

# PA\_SOURCES - This contains the list of source files which will be built

# into the static and shared PortAudio libraries.

# PA\_NON\_UNICODE\_SOURCES - This also contains a list of source files which

# will be build into the static and shared PortAudio libraries. However,

# these sources will not have any unicode compiler definitions added

# to them. This list should only contain external source dependencies.

# PA\_EXTRA\_SHARED\_SOURCES - Contains a list of extra files which will be

# associated only with the shared PortAudio library. This only seems

# relevant for Windows shared libraries which require a list of export

# symbols.

# Where other PA\_\* variables are set, these are almost always only used to

# preserve the historic SOURCE\_GROUP behavior (which again only has an impact

# on IDE-style generators for visual appearance) or store the output of

# find\_library() calls.

SET(PA\_COMMON\_INCLUDES

src/common/pa\_allocation.h

src/common/pa\_converters.h

src/common/pa\_cpuload.h

src/common/pa\_debugprint.h

src/common/pa\_dither.h

src/common/pa\_endianness.h

src/common/pa\_hostapi.h

src/common/pa\_memorybarrier.h

src/common/pa\_process.h

src/common/pa\_ringbuffer.h

src/common/pa\_stream.h

src/common/pa\_trace.h

src/common/pa\_types.h

src/common/pa\_util.h

)

SET(PA\_COMMON\_SOURCES

src/common/pa\_allocation.c

src/common/pa\_converters.c

src/common/pa\_cpuload.c

src/common/pa\_debugprint.c

src/common/pa\_dither.c

src/common/pa\_front.c

src/common/pa\_process.c

src/common/pa\_ringbuffer.c

src/common/pa\_stream.c

src/common/pa\_trace.c

)

SOURCE\_GROUP("common" FILES ${PA\_COMMON\_INCLUDES} ${PA\_COMMON\_SOURCES})

SET(PA\_PUBLIC\_INCLUDES include/portaudio.h)

SET(CMAKE\_MODULE\_PATH ${CMAKE\_MODULE\_PATH} ${CMAKE\_CURRENT\_SOURCE\_DIR}/cmake\_support)

SET(PA\_SKELETON\_SOURCES src/hostapi/skeleton/pa\_hostapi\_skeleton.c)

SOURCE\_GROUP("hostapi\\skeleton" ${PA\_SKELETON\_SOURCES})

SET(PA\_SOURCES ${PA\_COMMON\_SOURCES} ${PA\_SKELETON\_SOURCES})

SET(PA\_PRIVATE\_INCLUDE\_PATHS src/common ${CMAKE\_CURRENT\_BINARY\_DIR})

IF(WIN32)

SET(PA\_PRIVATE\_COMPILE\_DEFINITIONS ${PA\_PRIVATE\_COMPILE\_DEFINITIONS} \_CRT\_SECURE\_NO\_WARNINGS)

SET(PA\_PLATFORM\_SOURCES

src/os/win/pa\_win\_hostapis.c

src/os/win/pa\_win\_util.c

src/os/win/pa\_win\_waveformat.c

src/os/win/pa\_win\_wdmks\_utils.c

src/os/win/pa\_win\_coinitialize.c)

SET(PA\_PLATFORM\_INCLUDES

src/os/win/pa\_win\_coinitialize.h

src/os/win/pa\_win\_wdmks\_utils.h)

IF(MSVC)

SET(PA\_PLATFORM\_SOURCES ${PA\_PLATFORM\_SOURCES} src/os/win/pa\_x86\_plain\_converters.c)

SET(PA\_PLATFORM\_INCLUDES ${PA\_PLATFORM\_INCLUDES} src/os/win/pa\_x86\_plain\_converters.h)

ELSE()

SET(PA\_PRIVATE\_COMPILE\_DEFINITIONS ${PA\_PRIVATE\_COMPILE\_DEFINITIONS} \_WIN32\_WINNT=0x0501 WINVER=0x0501)

SET(DEF\_EXCLUDE\_X86\_PLAIN\_CONVERTERS ";")

ENDIF()

SOURCE\_GROUP("os\\win" FILES ${PA\_PLATFORM\_SOURCES} ${PA\_PLATFORM\_INCLUDES})

SET(PA\_SOURCES ${PA\_SOURCES} ${PA\_PLATFORM\_SOURCES})

SET(PA\_PRIVATE\_INCLUDES ${PA\_PRIVATE\_INCLUDES} ${PA\_PLATFORM\_INCLUDES})

SET(PA\_PRIVATE\_INCLUDE\_PATHS ${PA\_PRIVATE\_INCLUDE\_PATHS} src/os/win)

SET(PA\_LIBRARY\_DEPENDENCIES ${PA\_LIBRARY\_DEPENDENCIES} winmm)

# Try to find ASIO SDK (assumes that portaudio and asiosdk folders are side-by-side, see

# http://www.portaudio.com/trac/wiki/TutorialDir/Compile/WindowsASIOMSVC)

FIND\_PACKAGE(ASIOSDK)

IF(ASIOSDK\_FOUND)

OPTION(PA\_USE\_ASIO "Enable support for ASIO" ON)

ELSE()

OPTION(PA\_USE\_ASIO "Enable support for ASIO" OFF)

ENDIF()

IF(PA\_USE\_ASIO)

SET(PA\_PRIVATE\_INCLUDE\_PATHS ${PA\_PRIVATE\_INCLUDE\_PATHS} ${ASIOSDK\_ROOT\_DIR}/common)

SET(PA\_PRIVATE\_INCLUDE\_PATHS ${PA\_PRIVATE\_INCLUDE\_PATHS} ${ASIOSDK\_ROOT\_DIR}/host)

SET(PA\_PRIVATE\_INCLUDE\_PATHS ${PA\_PRIVATE\_INCLUDE\_PATHS} ${ASIOSDK\_ROOT\_DIR}/host/pc)

SET(PA\_ASIO\_SOURCES src/hostapi/asio/pa\_asio.cpp src/hostapi/asio/iasiothiscallresolver.cpp)

SET(PA\_ASIOSDK\_SOURCES ${ASIOSDK\_ROOT\_DIR}/common/asio.cpp ${ASIOSDK\_ROOT\_DIR}/host/pc/asiolist.cpp ${ASIOSDK\_ROOT\_DIR}/host/asiodrivers.cpp)

SOURCE\_GROUP("hostapi\\ASIO" FILES ${PA\_ASIO\_SOURCES})

SOURCE\_GROUP("hostapi\\ASIO\\ASIOSDK" FILES ${PA\_ASIOSDK\_SOURCES})

SET(PA\_PUBLIC\_INCLUDES ${PA\_PUBLIC\_INCLUDES} include/pa\_asio.h)

SET(PA\_SOURCES ${PA\_SOURCES} ${PA\_ASIO\_SOURCES})

SET(PA\_NON\_UNICODE\_SOURCES ${PA\_NON\_UNICODE\_SOURCES} ${PA\_ASIOSDK\_SOURCES})

SET(PA\_LIBRARY\_DEPENDENCIES ${PA\_LIBRARY\_DEPENDENCIES} ole32 uuid)

ELSE()

# Set variables for DEF file expansion

SET(DEF\_EXCLUDE\_ASIO\_SYMBOLS ";")

ENDIF()

# Try to find DirectX SDK

FIND\_PACKAGE(DXSDK)

IF(DXSDK\_FOUND)

OPTION(PA\_USE\_DS "Enable support for DirectSound" ON)

ELSE()

OPTION(PA\_USE\_DS "Enable support for DirectSound" OFF)

ENDIF()

IF(PA\_USE\_DS)

OPTION(PA\_USE\_DIRECTSOUNDFULLDUPLEXCREATE "Use DirectSound full duplex create" ON)

MARK\_AS\_ADVANCED(PA\_USE\_DIRECTSOUNDFULLDUPLEXCREATE)

IF(PA\_USE\_DIRECTSOUNDFULLDUPLEXCREATE)

SET(PA\_PRIVATE\_COMPILE\_DEFINITIONS ${PA\_PRIVATE\_COMPILE\_DEFINITIONS} PAWIN\_USE\_DIRECTSOUNDFULLDUPLEXCREATE)

ENDIF()

SET(PA\_PRIVATE\_INCLUDE\_PATHS ${PA\_PRIVATE\_INCLUDE\_PATHS} ${DXSDK\_INCLUDE\_DIR})

SET(PA\_DS\_INCLUDES src/hostapi/dsound/pa\_win\_ds\_dynlink.h)

SET(PA\_DS\_SOURCES src/hostapi/dsound/pa\_win\_ds.c src/hostapi/dsound/pa\_win\_ds\_dynlink.c)

SOURCE\_GROUP("hostapi\\dsound" FILES ${PA\_DS\_INCLUDES} ${PA\_DS\_SOURCES})

SET(PA\_PUBLIC\_INCLUDES ${PA\_PUBLIC\_INCLUDES} include/pa\_win\_ds.h)

SET(PA\_PRIVATE\_INCLUDES ${PA\_PRIVATE\_INCLUDES} ${PA\_DS\_INCLUDES})

SET(PA\_SOURCES ${PA\_SOURCES} ${PA\_DS\_SOURCES})

# If we use DirectSound, we need this for the library to be found (if not in VS project settings)

IF(DXSDK\_FOUND)

SET(PA\_LIBRARY\_DEPENDENCIES ${PA\_LIBRARY\_DEPENDENCIES} ${DXSDK\_DSOUND\_LIBRARY})

ENDIF()

ENDIF()

OPTION(PA\_USE\_WMME "Enable support for MME" ON)

IF(PA\_USE\_WMME)

SET(PA\_WMME\_SOURCES src/hostapi/wmme/pa\_win\_wmme.c)

SOURCE\_GROUP("hostapi\\wmme" FILES ${PA\_WMME\_SOURCES})

SET(PA\_PUBLIC\_INCLUDES ${PA\_PUBLIC\_INCLUDES} include/pa\_win\_wmme.h)

SET(PA\_SOURCES ${PA\_SOURCES} ${PA\_WMME\_SOURCES})

SET(PA\_LIBRARY\_DEPENDENCIES ${PA\_LIBRARY\_DEPENDENCIES} ole32 uuid)

ENDIF()

IF(MSVC)

OPTION(PA\_USE\_WASAPI "Enable support for WASAPI" ON)

ELSE()

# I was unable to get WASAPI to compile outside of Visual Studio. If

# anyone can figure out how to make this work with MinGW, please fix me.

SET(PA\_USE\_WASAPI OFF)

ENDIF()

IF(PA\_USE\_WASAPI)

SET(PA\_WASAPI\_SOURCES src/hostapi/wasapi/pa\_win\_wasapi.c)

SOURCE\_GROUP("hostapi\\wasapi" FILES ${PA\_WASAPI\_SOURCES})

SET(PA\_PUBLIC\_INCLUDES ${PA\_PUBLIC\_INCLUDES} include/pa\_win\_wasapi.h)

SET(PA\_SOURCES ${PA\_SOURCES} ${PA\_WASAPI\_SOURCES})

SET(PA\_LIBRARY\_DEPENDENCIES ${PA\_LIBRARY\_DEPENDENCIES} ole32 uuid)

IF(NOT MSVC)

SET(PA\_PRIVATE\_INCLUDE\_PATHS ${PA\_PRIVATE\_INCLUDE\_PATHS} src/hostapi/wasapi/mingw-include)

ENDIF()

ELSE()

SET(DEF\_EXCLUDE\_WASAPI\_SYMBOLS ";")

ENDIF()

OPTION(PA\_USE\_WDMKS "Enable support for WDMKS" ON)

IF(PA\_USE\_WDMKS)

SET(PA\_WDMKS\_SOURCES src/hostapi/wdmks/pa\_win\_wdmks.c)

SOURCE\_GROUP("hostapi\\wdmks" FILES ${PA\_WDMKS\_SOURCES})

SET(PA\_PUBLIC\_INCLUDES ${PA\_PUBLIC\_INCLUDES} include/pa\_win\_wdmks.h)

SET(PA\_SOURCES ${PA\_SOURCES} ${PA\_WDMKS\_SOURCES})

SET(PA\_LIBRARY\_DEPENDENCIES ${PA\_LIBRARY\_DEPENDENCIES} setupapi ole32 uuid)

ENDIF()

OPTION(PA\_USE\_WDMKS\_DEVICE\_INFO "Use WDM/KS API for device info" ON)

MARK\_AS\_ADVANCED(PA\_USE\_WDMKS\_DEVICE\_INFO)

IF(PA\_USE\_WDMKS\_DEVICE\_INFO)

SET(PA\_PRIVATE\_COMPILE\_DEFINITIONS ${PA\_PRIVATE\_COMPILE\_DEFINITIONS} PAWIN\_USE\_WDMKS\_DEVICE\_INFO)

ENDIF()

SET(GENERATED\_MESSAGE "CMake generated file, do NOT edit! Use CMake-GUI to change configuration instead.")

CONFIGURE\_FILE(cmake\_support/template\_portaudio.def ${CMAKE\_CURRENT\_BINARY\_DIR}/portaudio\_cmake.def @ONLY)

CONFIGURE\_FILE(cmake\_support/options\_cmake.h.in ${CMAKE\_CURRENT\_BINARY\_DIR}/options\_cmake.h @ONLY)

SET(PA\_PRIVATE\_INCLUDES ${PA\_PRIVATE\_INCLUDES} ${CMAKE\_CURRENT\_BINARY\_DIR}/options\_cmake.h)

SET(PA\_PRIVATE\_COMPILE\_DEFINITIONS ${PA\_PRIVATE\_COMPILE\_DEFINITIONS} PORTAUDIO\_CMAKE\_GENERATED)

SOURCE\_GROUP("cmake\_generated" FILES ${CMAKE\_CURRENT\_BINARY\_DIR}/portaudio\_cmake.def ${CMAKE\_CURRENT\_BINARY\_DIR}/options\_cmake.h)

SET(PA\_EXTRA\_SHARED\_SOURCES ${CMAKE\_CURRENT\_BINARY\_DIR}/portaudio\_cmake.def)

ELSE()

SET(PA\_PRIVATE\_INCLUDE\_PATHS ${PA\_PRIVATE\_INCLUDE\_PATHS} src/os/unix)

SET(PA\_PLATFORM\_SOURCES src/os/unix/pa\_unix\_hostapis.c src/os/unix/pa\_unix\_util.c)

SOURCE\_GROUP("os\\unix" FILES ${PA\_PLATFORM\_SOURCES})

SET(PA\_SOURCES ${PA\_SOURCES} ${PA\_PLATFORM\_SOURCES})

IF(APPLE)

SET(CMAKE\_MACOSX\_RPATH 1)

OPTION(PA\_USE\_COREAUDIO "Enable support for CoreAudio" ON)

IF(PA\_USE\_COREAUDIO)

SET(PA\_COREAUDIO\_SOURCES

src/hostapi/coreaudio/pa\_mac\_core.c

src/hostapi/coreaudio/pa\_mac\_core\_blocking.c

src/hostapi/coreaudio/pa\_mac\_core\_utilities.c)

SET(PA\_COREAUDIO\_INCLUDES

src/hostapi/coreaudio/pa\_mac\_core\_blocking.h

src/hostapi/coreaudio/pa\_mac\_core\_utilities.h)

SOURCE\_GROUP("hostapi\\coreaudio" FILES ${PA\_COREAUDIO\_SOURCES} ${PA\_COREAUDIO\_INCLUDES})

SET(PA\_PUBLIC\_INCLUDES ${PA\_PUBLIC\_INCLUDES} include/pa\_mac\_core.h)

SET(PA\_PRIVATE\_INCLUDES ${PA\_PRIVATE\_INCLUDES} ${PA\_COREAUDIO\_INCLUDES})

SET(PA\_SOURCES ${PA\_SOURCES} ${PA\_COREAUDIO\_SOURCES})

FIND\_LIBRARY(COREAUDIO\_LIBRARY CoreAudio REQUIRED)

FIND\_LIBRARY(AUDIOTOOLBOX\_LIBRARY AudioToolbox REQUIRED)

FIND\_LIBRARY(AUDIOUNIT\_LIBRARY AudioUnit REQUIRED)

FIND\_LIBRARY(CARBON\_LIBRARY Carbon REQUIRED)

MARK\_AS\_ADVANCED(COREAUDIO\_LIBRARY AUDIOTOOLBOX\_LIBRARY AUDIOUNIT\_LIBRARY CARBON\_LIBRARY)

SET(PA\_LIBRARY\_DEPENDENCIES ${PA\_LIBRARY\_DEPENDENCIES} ${COREAUDIO\_LIBRARY} ${AUDIOTOOLBOX\_LIBRARY} ${AUDIOUNIT\_LIBRARY} ${CARBON\_LIBRARY})

SET(PA\_PRIVATE\_COMPILE\_DEFINITIONS ${PA\_PRIVATE\_COMPILE\_DEFINITIONS} PA\_USE\_COREAUDIO)

SET(PA\_PKGCONFIG\_LDFLAGS "${PA\_PKGCONFIG\_LDFLAGS} -framework CoreAudio -framework AudioToolbox -framework AudioUnit -framework Carbon")

ENDIF()

ELSEIF(UNIX)

FIND\_PACKAGE(Jack)

IF(JACK\_FOUND)

OPTION(PA\_USE\_JACK "Enable support for Jack" ON)

ELSE()

OPTION(PA\_USE\_JACK "Enable support for Jack" OFF)

ENDIF()

IF(PA\_USE\_JACK)

SET(PA\_PRIVATE\_INCLUDE\_PATHS ${PA\_PRIVATE\_INCLUDE\_PATHS} ${JACK\_INCLUDE\_DIRS})

SET(PA\_JACK\_SOURCES src/hostapi/jack/pa\_jack.c)

SOURCE\_GROUP("hostapi\\JACK" FILES ${PA\_JACK\_SOURCES})

SET(PA\_PUBLIC\_INCLUDES ${PA\_PUBLIC\_INCLUDES} include/pa\_jack.h)

SET(PA\_SOURCES ${PA\_SOURCES} ${PA\_JACK\_SOURCES})

SET(PA\_PRIVATE\_COMPILE\_DEFINITIONS ${PA\_PRIVATE\_COMPILE\_DEFINITIONS} PA\_USE\_JACK)

SET(PA\_LIBRARY\_DEPENDENCIES ${PA\_LIBRARY\_DEPENDENCIES} ${JACK\_LIBRARIES})

SET(PA\_PKGCONFIG\_LDFLAGS "${PA\_PKGCONFIG\_LDFLAGS} -ljack")

ENDIF()

FIND\_PACKAGE(ALSA)

IF(ALSA\_FOUND)

OPTION(PA\_USE\_ALSA "Enable support for ALSA" ON)

ELSE()

OPTION(PA\_USE\_ALSA "Enable support for ALSA" OFF)

ENDIF()

IF(PA\_USE\_ALSA)

SET(PA\_PRIVATE\_INCLUDE\_PATHS ${PA\_PRIVATE\_INCLUDE\_PATHS} ${ALSA\_INCLUDE\_DIRS})

SET(PA\_ALSA\_SOURCES src/hostapi/alsa/pa\_linux\_alsa.c)

SOURCE\_GROUP("hostapi\\ALSA" FILES ${PA\_ALSA\_SOURCES})

SET(PA\_PUBLIC\_INCLUDES ${PA\_PUBLIC\_INCLUDES} include/pa\_linux\_alsa.h)

SET(PA\_SOURCES ${PA\_SOURCES} ${PA\_ALSA\_SOURCES})

SET(PA\_PRIVATE\_COMPILE\_DEFINITIONS ${PA\_PRIVATE\_COMPILE\_DEFINITIONS} PA\_USE\_ALSA)

SET(PA\_LIBRARY\_DEPENDENCIES ${PA\_LIBRARY\_DEPENDENCIES} ${ALSA\_LIBRARIES})

SET(PA\_PKGCONFIG\_LDFLAGS "${PA\_PKGCONFIG\_LDFLAGS} -lasound")

ENDIF()

ENDIF()

SET(PA\_PKGCONFIG\_LDFLAGS "${PA\_PKGCONFIG\_LDFLAGS} -lm -lpthread")

SET(PA\_LIBRARY\_DEPENDENCIES ${PA\_LIBRARY\_DEPENDENCIES} m pthread)

ENDIF()

SOURCE\_GROUP("include" FILES ${PA\_PUBLIC\_INCLUDES})

SET(PA\_INCLUDES ${PA\_PRIVATE\_INCLUDES} ${PA\_PUBLIC\_INCLUDES})

IF(WIN32)

OPTION(PA\_UNICODE\_BUILD "Enable Portaudio Unicode build" ON)

IF(PA\_UNICODE\_BUILD)

SET\_SOURCE\_FILES\_PROPERTIES(${PA\_SOURCES} PROPERTIES COMPILE\_DEFINITIONS "UNICODE;\_UNICODE")

ENDIF()

ENDIF()

OPTION(PA\_ENABLE\_DEBUG\_OUTPUT "Enable debug output for Portaudio" OFF)

IF(PA\_ENABLE\_DEBUG\_OUTPUT)

SET(PA\_PRIVATE\_COMPILE\_DEFINITIONS ${PA\_PRIVATE\_COMPILE\_DEFINITIONS} PA\_ENABLE\_DEBUG\_OUTPUT)

ENDIF()

INCLUDE(TestBigEndian)

TEST\_BIG\_ENDIAN(IS\_BIG\_ENDIAN)

IF(IS\_BIG\_ENDIAN)

SET(PA\_PRIVATE\_COMPILE\_DEFINITIONS ${PA\_PRIVATE\_COMPILE\_DEFINITIONS} PA\_BIG\_ENDIAN)

ELSE()

SET(PA\_PRIVATE\_COMPILE\_DEFINITIONS ${PA\_PRIVATE\_COMPILE\_DEFINITIONS} PA\_LITTLE\_ENDIAN)

ENDIF()

ADD\_LIBRARY(portaudio SHARED ${PA\_INCLUDES} ${PA\_COMMON\_INCLUDES} ${PA\_SOURCES} ${PA\_NON\_UNICODE\_SOURCES} ${PA\_EXTRA\_SHARED\_SOURCES})

SET\_PROPERTY(TARGET portaudio APPEND\_STRING PROPERTY COMPILE\_DEFINITIONS ${PA\_PRIVATE\_COMPILE\_DEFINITIONS})

TARGET\_INCLUDE\_DIRECTORIES(portaudio PRIVATE ${PA\_PRIVATE\_INCLUDE\_PATHS})

TARGET\_INCLUDE\_DIRECTORIES(portaudio PUBLIC include)

TARGET\_LINK\_LIBRARIES(portaudio ${PA\_LIBRARY\_DEPENDENCIES})

ADD\_LIBRARY(portaudio\_static STATIC ${PA\_INCLUDES} ${PA\_COMMON\_INCLUDES} ${PA\_SOURCES} ${PA\_NON\_UNICODE\_SOURCES})

SET\_PROPERTY(TARGET portaudio\_static APPEND\_STRING PROPERTY COMPILE\_DEFINITIONS ${PA\_PRIVATE\_COMPILE\_DEFINITIONS})

TARGET\_INCLUDE\_DIRECTORIES(portaudio\_static PRIVATE ${PA\_PRIVATE\_INCLUDE\_PATHS})

TARGET\_INCLUDE\_DIRECTORIES(portaudio\_static PUBLIC include)

TARGET\_LINK\_LIBRARIES(portaudio\_static ${PA\_LIBRARY\_DEPENDENCIES})

IF(WIN32 AND MSVC)

OPTION(PA\_CONFIG\_LIB\_OUTPUT\_PATH "Make sure that output paths are kept neat" OFF)

IF(CMAKE\_CL\_64)

SET(TARGET\_POSTFIX x64)

IF(PA\_CONFIG\_LIB\_OUTPUT\_PATH)

SET(LIBRARY\_OUTPUT\_PATH ${CMAKE\_CURRENT\_BINARY\_DIR}/bin/x64)

ENDIF()

ELSE()

SET(TARGET\_POSTFIX x86)

IF(PA\_CONFIG\_LIB\_OUTPUT\_PATH)

SET(LIBRARY\_OUTPUT\_PATH ${CMAKE\_CURRENT\_BINARY\_DIR}/bin/Win32)

ENDIF()

ENDIF()

SET\_TARGET\_PROPERTIES(portaudio PROPERTIES OUTPUT\_NAME portaudio\_${TARGET\_POSTFIX} FOLDER "Portaudio")

SET\_TARGET\_PROPERTIES(portaudio\_static PROPERTIES OUTPUT\_NAME portaudio\_static\_${TARGET\_POSTFIX} FOLDER "Portaudio")

ELSE()

IF(APPLE AND CMAKE\_VERSION VERSION\_GREATER 3.4.2)

OPTION(PA\_OUTPUT\_OSX\_FRAMEWORK "Generate an OS X framework instead of the simple library" OFF)

IF(PA\_OUTPUT\_OSX\_FRAMEWORK)

SET\_TARGET\_PROPERTIES(portaudio PROPERTIES

FRAMEWORK TRUE

MACOSX\_FRAMEWORK\_IDENTIFIER com.portaudio

FRAMEWORK\_VERSION A

PUBLIC\_HEADER "${PA\_PUBLIC\_INCLUDES}"

VERSION 19.0

SOVERSION 19.0)

ENDIF()

ENDIF()

IF(NOT PA\_OUTPUT\_OSX\_FRAMEWORK)

CONFIGURE\_FILE(cmake\_support/portaudio-2.0.pc.in ${CMAKE\_CURRENT\_BINARY\_DIR}/portaudio-2.0.pc @ONLY)

INSTALL(FILES ${CMAKE\_CURRENT\_BINARY\_DIR}/portaudio-2.0.pc DESTINATION lib/pkgconfig)

INSTALL(FILES ${PA\_PUBLIC\_INCLUDES} DESTINATION include)

INSTALL(TARGETS portaudio DESTINATION lib)

ENDIF()

ENDIF()

# Prepared for inclusion of test files

OPTION(PA\_BUILD\_TESTS "Include test projects" OFF)

IF(PA\_BUILD\_TESTS)

SUBDIRS(test)

ENDIF()

# Prepared for inclusion of test files

OPTION(PA\_BUILD\_EXAMPLES "Include example projects" OFF)

IF(PA\_BUILD\_EXAMPLES)

SUBDIRS(examples)

ENDIF()