**安装msys2**

msys2类似于cygwin（可以在windows上配置linux环境）和mingw（git的bash环境），安装完成之后,先需要把安装目录下的msys2\_shell.cmd中注释掉的rem set MSYS2\_PATH\_TYPE=inherit,改成set MSYS2\_PATH\_TYPE=inherit，主要是将vs的环境继承给msys2；接着打开msys2的shell，安装4个编译工具，

pacman -S make gcc perl diffutils

这时候我们需要在我的电脑中改变一下环境变量，在Path中将D:\cygwin64\bin提到D:\msys64\usr\bin前面，没有的话可以先配置

# **替换yasm**

将下载的yasm-\*\*-win64.exe改成yasm.exe,替换msys安装目录x:\msys64\usr\bin\yasm.exe,可以做个备份把原来的改成yasm.bak

# **安装gas-preprocessor**

将下载的gas-preprocessor.pl放到msys2安装目录下面x:\msys64\usr\bin\gas-preprocessor.pl

从VS的命令窗口重新启动msys2的bash窗口

这一步没有把握正确，编译很容易出错,例如cl is unable to create an executable file.正确的启动步骤是：先去打开VS的工具命令提示符，C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Visual Studio 2015\Visual Studio Tools\Windows Desktop Command Prompts，然后在命名窗口中使用命令符打开msys2的bash窗口，

我安装在D盘，依次执行命令如下：

D:

cd msys64

msys2\_shell.cmd

在bash窗口中先检查，安装环境是否已经正确配置：

$ which cl

/d/Program Files (x86)/Microsoft Visual Studio 14.0/VC/BIN/amd64\_x86/cl

$ which link

/d/Program Files (x86)/Microsoft Visual Studio 14.0/VC/BIN/amd64\_x86/link

$ which yasm

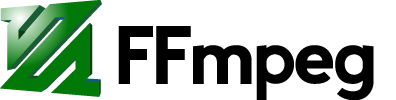
/usr/bin/yasm

$ which cpp

/usr/bin/cpp

$ which gas-preprocessor.pl

/usr/bin/gas-preprocessor.pl

[](https://ffmpeg.org/)

窗体顶端

# **Prerequisites and First Time Setup Instructions**

## **Prerequisites**

* For Windows 10: [​Microsoft Visual Studio 2015](https://dev.windows.com/en-us/downloads/windows-10-developer-tools) (tested with Visual Studio 2015 RTM)
* For Windows 8.1: [​Microsoft Visual Studio 2013](http://www.visualstudio.com/downloads/download-visual-studio-vs) (use MSVC 2013 Update 3 RTM or newer)
* [​MSYS2](http://msys2.github.io/) (GNU Make environment)
* [​YASM](http://yasm.tortall.net/) (x86 assembly code compiler)
* [​gas-preprocessor.pl](https://github.com/FFmpeg/gas-preprocessor) (ARM assembly pre-processor)

### **Windows 10 Setup**

Download and install [​Microsoft Visual Studio 2015](https://dev.windows.com/en-us/downloads/windows-10-developer-tools) on a Windows 10 machine.

### **Windows 8.1 Setup**

Download and install [​Microsoft Visual Studio 2013](http://www.visualstudio.com/downloads/download-visual-studio-vs) on a Windows 8.1 machine

### **MSYS2 Setup**

Download the latest MSYS2 installer from [​http://msys2.github.io/](http://msys2.github.io/) and **follow the installation instruction closely** from the installation webpage. In order for the environment from the Windows side to be inherited on the MSYS2 side, uncomment the following line from msys2\_shell.cmd if it is present: replace rem set MSYS2\_PATH\_TYPE=inherit with set MSYS2\_PATH\_TYPE=inherit this will allow the environment variables for Visual Studio to be transferred to the MSYS2 environment and back.

Once base MSYS2 is successfully installed, get the latest make package by invoking the following command in your MSYS2 shell

pacman -S make

Also get the latest gcc package

pacman -S gcc

Rename or remove link.exe in the MSYS2 usr bin folder (E.g. C:\msys64\usr\bin\link.exe) to prevent conflict with MSVC link.exe

Install perl as it will be needed to run 'gas-preprocessor.pl'

pacman -S perl

Install also diffutils for configure script

pacman -S diffutils

### **YASM Setup**

Download YASM executable from [​http://yasm.tortall.net/Download.html](http://yasm.tortall.net/Download.html). You have to download the "**general use**" binaries and NOT the ones for VS2010. Either Win32 or Win64 binaries support outputting object files for both architectures so that should not matter. The last tested version was yasm-1.3.0-win64.exe.

Rename the downloaded executable to yasm.exe and place it in your MSYS2 path. E.g.C:\msys64\usr\bin\yasm.exe.

### **gas-preprocessor Setup**

Download gas-preprocessor.pl Perl script from [​https://github.com/FFmpeg/gas-preprocessor](https://github.com/FFmpeg/gas-preprocessor)

Place the downloaded Perl script in your MSYS2 path. E.g. C:\msys64\usr\bin\gas-preprocessor.pl

## **Verifying your FFmpeg Environment Setup**

Launch Visual Studio **ARM Cross Tools Command Prompt**. E.g.

C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Visual Studio 2015\Visual Studio Tools\Windows Desktop Command Prompts\VS2015 x86 ARM Cross Tools Command Prompt

Open **MSYS2 Shell** from the command prompt above (use the correct drive and location of your MSYS2 installation). Note that the command shell above will close and it may take a while for the MSYS2 shell to launch.

C:\msys64\msys2\_shell.cmd

In the MSYS2 shell verify that all the tools below are setup properly by running the following commands

$ which cl

/c/Program Files (x86)/Microsoft Visual Studio 14.0/VC/BIN/x86\_ARM/cl

$ which link

/c/Program Files (x86)/Microsoft Visual Studio 14.0/VC/BIN/x86\_ARM/link

$ which armasm

/c/Program Files (x86)/Microsoft Visual Studio 14.0/VC/BIN/x86\_ARM/armasm

$ which yasm

/usr/bin/yasm

$ which cpp

/usr/bin/cpp

$ which gas-preprocessor.pl

/usr/bin/gas-preprocessor.pl

Verify that the tools are in the path and point to the right location where MSYS2 and Visual Studio are installed

To keep the source tree clean and the platforms separated, we will have the intermediate files go to the Output\<Platform>\<Architecture> folder under the FFmpeg source tree. We will also have the install files (the files necessary to link and use FFmpeg in your application) go to the Build\<Platform>\<Architecture> folder under the FFmpeg source tree.

# **Compiling for Windows 10**

## **Windows 10 x86**

Launch **VS2015 x86 Native Tools Command Prompt**. E.g.

C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Visual Studio 2015\Visual Studio Tools\Windows Desktop Command Prompts\VS2015 x86 Native Tools Command Prompt

Set the following environment variables in the launched command prompt above

SET LIB=%VSINSTALLDIR%VC\lib\store;%VSINSTALLDIR%VC\atlmfc\lib;%UniversalCRTSdkDir%lib\%UCRTVersion%\ucrt\x86;;%UniversalCRTSdkDir%lib\%UCRTVersion%\um\x86;C:\Program Files (x86)\Windows Kits\NETFXSDK\4.6\lib\um\x86;;C:\Program Files (x86)\Windows Kits\NETFXSDK\4.6\Lib\um\x86

SET LIBPATH=%VSINSTALLDIR%VC\atlmfc\lib;%VSINSTALLDIR%VC\lib;

SET INCLUDE=%VSINSTALLDIR%VC\include;%VSINSTALLDIR%VC\atlmfc\include;%UniversalCRTSdkDir%Include\%UCRTVersion%\ucrt;%UniversalCRTSdkDir%Include\%UCRTVersion%\um;%UniversalCRTSdkDir%Include\%UCRTVersion%\shared;%UniversalCRTSdkDir%Include\%UCRTVersion%\winrt;C:\Program Files (x86)\Windows Kits\NETFXSDK\4.6\Include\um;

Open **MSYS2 Shell** from the command prompt above (use the correct drive and location of your MSYS2 installation). Note that the command shell above will close and it may take a while for the MSYS2 shell to launch.

C:\msys64\msys2\_shell.cmd

In your MSYS2 shell navigate to your cloned FFmpeg folder. E.g.

cd /c/ffmpeg

Invoke the following make commands

mkdir -p Output/Windows10/x86

cd Output/Windows10/x86

../../../configure \

--toolchain=msvc \

--disable-programs \

--disable-d3d11va \

--disable-dxva2 \

--arch=x86 \

--enable-shared \

--enable-cross-compile \

--target-os=win32 \

--extra-cflags="-MD -DWINAPI\_FAMILY=WINAPI\_FAMILY\_APP -D\_WIN32\_WINNT=0x0A00" \

--extra-ldflags="-APPCONTAINER WindowsApp.lib" \

--prefix=../../../Build/Windows10/x86

make

make install

Generated libraries can be found in Build/Windows10/x86 folder specified in --prefix option above

## **Windows 10 x64**

Launch **VS2015 x86 x64 Cross Tools Command Prompt**. E.g.

C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Visual Studio 2015\Visual Studio Tools\Windows Desktop Command Prompts\VS2015 x86 x64 Cross Tools Command Prompt

Set the following environment variables in the launched command prompt above

SET LIB=%VSINSTALLDIR%VC\lib\store\amd64;%VSINSTALLDIR%VC\atlmfc\lib\amd64;%UniversalCRTSdkDir%lib\%UCRTVersion%\ucrt\x64;;%UniversalCRTSdkDir%lib\%UCRTVersion%\um\x64;C:\Program Files (x86)\Windows Kits\NETFXSDK\4.6\lib\um\x64;;C:\Program Files (x86)\Windows Kits\NETFXSDK\4.6\Lib\um\x64

SET LIBPATH=%VSINSTALLDIR%VC\atlmfc\lib\amd64;%VSINSTALLDIR%VC\lib\amd64;

SET INCLUDE=%VSINSTALLDIR%VC\include;%VSINSTALLDIR%VC\atlmfc\include;%UniversalCRTSdkDir%Include\%UCRTVersion%\ucrt;%UniversalCRTSdkDir%Include\%UCRTVersion%\um;%UniversalCRTSdkDir%Include\%UCRTVersion%\shared;%UniversalCRTSdkDir%Include\%UCRTVersion%\winrt;C:\Program Files (x86)\Windows Kits\NETFXSDK\4.6\Include\um;

Open **MSYS2 Shell** from the command prompt above (use the correct drive and location of your MSYS2 installation). Note that the command shell above will close and it may take a while for the MSYS2 shell to launch.

C:\msys64\msys2\_shell.cmd

In your MSYS2 shell navigate to your cloned FFmpeg folder. E.g.

cd /c/ffmpeg

Invoke the following make commands

mkdir -p Output/Windows10/x64

cd Output/Windows10/x64

../../../configure \

--toolchain=msvc \

--disable-programs \

--disable-d3d11va \

--disable-dxva2 \

--arch=x86\_64 \

--enable-shared \

--enable-cross-compile \

--target-os=win32 \

--extra-cflags="-MD -DWINAPI\_FAMILY=WINAPI\_FAMILY\_APP -D\_WIN32\_WINNT=0x0A00" \

--extra-ldflags="-APPCONTAINER WindowsApp.lib" \

--prefix=../../../Build/Windows10/x64

make

make install

Generated libraries can be found in Build/Windows10/x64 folder specified in --prefix option above

If you encounter an error linking because link.exe complains that MSPDB140.dll has the wrong version installed. Run the following command from an administrative command-prompt and rebuild:

copy "C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC\bin\mspdbsrv.exe" "C:\Program Files (x86)\Microsoft Visual Studio 14.0\Common7\IDE"

## **Windows 10 ARM**

Launch **VS2015 x86 ARM Cross Tools Command Prompt**. E.g.

C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Visual Studio 2015\Visual Studio Tools\Windows Desktop Command Prompts\VS2015 x86 ARM Cross Tools Command Prompt

Set the following environment variables in the launched command prompt above

SET LIB=%VSINSTALLDIR%VC\lib\store\ARM;%VSINSTALLDIR%VC\atlmfc\lib\ARM;%UniversalCRTSdkDir%lib\%UCRTVersion%\ucrt\arm;;%UniversalCRTSdkDir%lib\%UCRTVersion%\um\arm;C:\Program Files (x86)\Windows Kits\NETFXSDK\4.6\lib\um\arm;;C:\Program Files (x86)\Windows Kits\NETFXSDK\4.6\Lib\um\arm

SET LIBPATH=%VSINSTALLDIR%VC\atlmfc\lib\ARM;%VSINSTALLDIR%VC\lib\ARM;

SET INCLUDE=%VSINSTALLDIR%VC\include;%VSINSTALLDIR%VC\atlmfc\include;%UniversalCRTSdkDir%Include\%UCRTVersion%\ucrt;%UniversalCRTSdkDir%Include\%UCRTVersion%\um;%UniversalCRTSdkDir%Include\%UCRTVersion%\shared;%UniversalCRTSdkDir%Include\%UCRTVersion%\winrt;C:\Program Files (x86)\Windows Kits\NETFXSDK\4.6\Include\um;

Open **MSYS2 Shell** from the command prompt above (use the correct drive and location of your MSYS2 installation). Note that the command shell above will close and it may take a while for the MSYS2 shell to launch.

C:\msys64\msys2\_shell.cmd

In your MSYS2 shell navigate to your cloned FFmpeg folder. E.g.

cd /c/ffmpeg

Invoke the following make commands

mkdir -p Output/Windows10/ARM

cd Output/Windows10/ARM

../../../configure \

--toolchain=msvc \

--disable-programs \

--disable-d3d11va \

--disable-dxva2 \

--arch=arm \

--as=armasm \

--cpu=armv7 \

--enable-thumb \

--enable-shared \

--enable-cross-compile \

--target-os=win32 \

--extra-cflags="-MD -DWINAPI\_FAMILY=WINAPI\_FAMILY\_APP -D\_WIN32\_WINNT=0x0A00 -D\_\_ARM\_PCS\_VFP" \

--extra-ldflags="-APPCONTAINER WindowsApp.lib" \

--prefix=../../../Build/Windows10/ARM

make

make install

Generated libraries can be found in Build/Windows10/ARM folder specified in --prefix option above

If you encounter an error linking because link.exe complains that MSPDB140.dll has the wrong version installed. Run the following command from an **administrative command-prompt**and rebuild:

copy "C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC\bin\mspdbsrv.exe" "C:\Program Files (x86)\Microsoft Visual Studio 14.0\Common7\IDE"

# **Compiling for Windows 8.1**

## **Windows Store 8.1 x86 (Windows 8.1 Win32 in Visual Studio)**

Launch **Developer Command Prompt for VS2013**

Start Menu > Visual Studio 2013 (Start Menu Folder) > Visual Studio Tools > Developer Command Prompt for VS2013

Set the following environment variables in the launched command prompt above. These environment variables overwrite the default paths with correct target specific ones.

SET LIB=%VSINSTALLDIR%VC\lib\store;%VSINSTALLDIR%VC\atlmfc\lib;%WindowsSdkDir%lib\winv6.3\um\x86;;

SET LIBPATH=%WindowsSdkDir%References\CommonConfiguration\Neutral;;%VSINSTALLDIR%VC\atlmfc\lib;%VSINSTALLDIR%VC\lib;

SET INCLUDE=%VSINSTALLDIR%VC\include;%VSINSTALLDIR%VC\atlmfc\include;%WindowsSdkDir%Include\um;%WindowsSdkDir%Include\shared;%WindowsSdkDir%Include\winrt;;

Open **MSYS2 Shell** from the command prompt above (use the correct drive and location of your MSYS2 installation). Note that the command shell above will close and it may take a while for the MSYS2 shell to launch.

C:\msys64\msys2\_shell.cmd

In your MSYS2 shell navigate to your cloned FFmpeg folder. E.g.

cd /c/ffmpeg

Invoke the following make commands

mkdir -p Output/Windows8.1/x86

cd Output/Windows8.1/x86

../../../configure \

--toolchain=msvc \

--disable-programs \

--disable-dxva2 \

--arch=x86 \

--enable-shared \

--enable-cross-compile \

--target-os=win32 \

--extra-cflags="-MD -DWINAPI\_FAMILY=WINAPI\_FAMILY\_PC\_APP -D\_WIN32\_WINNT=0x0603" \

--extra-ldflags="-APPCONTAINER" \

--prefix=../../../Build/Windows8.1/x86

make

make install

Generated libraries can be found in Build/Windows8.1/x86 folder specified in --prefix option above

## **Windows Store 8.1 x64 (Windows 8.1 x64 in Visual Studio)**

Launch **VS2013 x64 Cross Tools Command Prompt**

Start Menu > Visual Studio 2013 (Start Menu Folder) > Visual Studio Tools > VS2013 x64 Cross Tools Command Prompt

Set the following environment variables in the launched command prompt above. These environment variables overwrite the default paths with correct target specific ones.

SET LIB=%VSINSTALLDIR%VC\lib\store\amd64;%VSINSTALLDIR%VC\atlmfc\lib\amd64;%WindowsSdkDir%lib\winv6.3\um\x64;;

SET LIBPATH=%WindowsSdkDir%References\CommonConfiguration\Neutral;;%VSINSTALLDIR%VC\atlmfc\lib\amd64;%VSINSTALLDIR%VC\lib\amd64;

SET INCLUDE=%VSINSTALLDIR%VC\include;%VSINSTALLDIR%VC\atlmfc\include;%WindowsSdkDir%Include\um;%WindowsSdkDir%Include\shared;%WindowsSdkDir%Include\winrt;;

Open **MSYS2 Shell** from the command prompt above (use the correct drive and location of your MSYS2 installation). Note that the command shell above will close and it may take a while for the MSYS2 shell to launch.

C:\msys64\msys2\_shell.cmd

In your MSYS2 shell navigate to your cloned FFmpeg folder. E.g.

cd /c/ffmpeg

Invoke the following make commands

mkdir -p Output/Windows8.1/x64

cd Output/Windows8.1/x64

../../../configure \

--toolchain=msvc \

--disable-programs \

--disable-dxva2 \

--arch=x86\_64 \

--enable-shared \

--enable-cross-compile \

--target-os=win32 \

--extra-cflags="-MD -DWINAPI\_FAMILY=WINAPI\_FAMILY\_PC\_APP -D\_WIN32\_WINNT=0x0603" \

--extra-ldflags="-APPCONTAINER" \

--prefix=../../../Build/Windows8.1/x64

make

make install

Generated libraries can be found in Build/Windows8.1/x64 folder specified in --prefix option above

## **Windows Store 8.1 ARM (Windows 8.1 ARM in Visual Studio)**

Launch **VS2013 ARM Cross Tools Command Prompt**

Start Menu > Visual Studio 2013 (Start Menu Folder) > Visual Studio Tools > VS2013 ARM Cross Tools Command Prompt

Set the following environment variables in the launched command prompt above. These environment variables overwrite the default paths with correct target specific ones.

SET LIB=%VSINSTALLDIR%VC\lib\store\ARM;%VSINSTALLDIR%VC\atlmfc\lib\ARM;%WindowsSdkDir%lib\winv6.3\um\arm;;

SET LIBPATH=%WindowsSdkDir%References\CommonConfiguration\Neutral;;%VSINSTALLDIR%VC\atlmfc\lib\ARM;%VSINSTALLDIR%VC\lib\ARM;

SET INCLUDE=%VSINSTALLDIR%VC\include;%VSINSTALLDIR%VC\atlmfc\include;%WindowsSdkDir%Include\um;%WindowsSdkDir%Include\shared;%WindowsSdkDir%Include\winrt;;

Open **MSYS2 Shell** from the command prompt above (use the correct drive and location of your MSYS2 installation). Note that the command shell above will close and it may take a while for the MSYS2 shell to launch.

C:\msys64\msys2\_shell.cmd

In your MSYS2 shell navigate to your cloned FFmpeg folder. E.g.

cd /c/ffmpeg

Invoke the following make commands

mkdir -p Output/Windows8.1/ARM

cd Output/Windows8.1/ARM

../../../configure \

--toolchain=msvc \

--disable-programs \

--disable-d3d11va \

--disable-dxva2 \

--arch=arm \

--as=armasm \

--cpu=armv7 \

--enable-thumb \

--enable-shared \

--enable-cross-compile \

--target-os=win32 \

--extra-cflags="-MD -DWINAPI\_FAMILY=WINAPI\_FAMILY\_PC\_APP -D\_WIN32\_WINNT=0x0603 -D\_\_ARM\_PCS\_VFP" \

--extra-ldflags="-APPCONTAINER -MACHINE:ARM" \

--prefix=../../../Build/Windows8.1/ARM

make

make install

Generated libraries can be found in Build/Windows8.1/ARM folder specified in --prefix option above

## **Windows Phone 8.1 x86 (Windows Phone 8.1 Win32 Emulator in Visual Studio)**

Launch **Developer Command Prompt for VS2013**

Start Menu > Visual Studio 2013 (Start Menu Folder) > Visual Studio Tools > Developer Command Prompt for VS2013

Set the following environment variables in the launched command prompt above. These environment variables overwrite the default paths with correct target specific ones.

SET LIB=%VSINSTALLDIR%VC\lib\store;%VSINSTALLDIR%VC\atlmfc\lib;%WindowsSdkDir%..\..\Windows Phone Kits\8.1\lib\x86;;

SET LIBPATH=%VSINSTALLDIR%VC\atlmfc\lib;%VSINSTALLDIR%VC\lib

SET INCLUDE=%VSINSTALLDIR%VC\INCLUDE;%VSINSTALLDIR%VC\ATLMFC\INCLUDE;%WindowsSdkDir%..\..\Windows Phone Kits\8.1\Include;%WindowsSdkDir%..\..\Windows Phone Kits\8.1\Include\abi;%WindowsSdkDir%..\..\Windows Phone Kits\8.1\Include\mincore;%WindowsSdkDir%..\..\Windows Phone Kits\8.1\Include\minwin;%WindowsSdkDir%..\..\Windows Phone Kits\8.1\Include\wrl;

Open **MSYS2 Shell** from the command prompt above (use the correct drive and location of your MSYS2 installation). Note that the command shell above will close and it may take a while for the MSYS2 shell to launch.

C:\msys64\msys2\_shell.cmd

In your MSYS2 shell navigate to your cloned FFmpeg folder. E.g.

cd /c/ffmpeg

Invoke the following make commands

mkdir -p Output/WindowsPhone8.1/x86

cd Output/WindowsPhone8.1/x86

../../../configure \

--toolchain=msvc \

--disable-programs \

--disable-d3d11va \

--disable-dxva2 \

--arch=x86 \

--enable-shared \

--enable-cross-compile \

--target-os=win32 \

--extra-cflags="-MD -DWINAPI\_FAMILY=WINAPI\_FAMILY\_PHONE\_APP -D\_WIN32\_WINNT=0x0603" \

--extra-ldflags="-APPCONTAINER -subsystem:console -opt:ref WindowsPhoneCore.lib RuntimeObject.lib PhoneAppModelHost.lib -NODEFAULTLIB:kernel32.lib -NODEFAULTLIB:ole32.lib" \

--prefix=../../../Build/WindowsPhone8.1/x86

make

make install

Generated libraries can be found in Build/WindowsPhone8.1/x86 folder specified in --prefix option above

## **Windows Phone 8.1 ARM (Windows Phone 8.1 ARM Device in Visual Studio)**

Launch **VS2013 ARM Cross Tools Command Prompt**

Start Menu > Visual Studio 2013 (Start Menu Folder) > Visual Studio Tools > VS2013 ARM Cross Tools Command Prompt

Set the following environment variables in the launched command prompt above. These environment variables overwrite the default paths with correct target specific ones.

SET LIB=%VSINSTALLDIR%VC\lib\store\ARM;%VSINSTALLDIR%VC\atlmfc\lib\ARM;%WindowsSdkDir%..\..\Windows Phone Kits\8.1\lib\arm;;

SET LIBPATH=%VSINSTALLDIR%VC\atlmfc\lib\ARM;%VSINSTALLDIR%VC\lib\ARM

SET INCLUDE=%VSINSTALLDIR%VC\include;%VSINSTALLDIR%VC\atlmfc\include;%WindowsSdkDir%..\..\Windows Phone Kits\8.1\Include;%WindowsSdkDir%..\..\Windows Phone Kits\8.1\Include\abi;%WindowsSdkDir%..\..\Windows Phone Kits\8.1\Include\mincore;%WindowsSdkDir%..\..\Windows Phone Kits\8.1\Include\minwin;%WindowsSdkDir%..\..\Windows Phone Kits\8.1\Include\wrl;

Open **MSYS2 Shell** from the command prompt above (use the correct drive and location of your MSYS2 installation). Note that the command shell above will close and it may take a while for the MSYS2 shell to launch.

C:\msys64\msys2\_shell.cmd

In your MSYS2 shell navigate to your cloned FFmpeg folder. E.g.

cd /c/ffmpeg

Invoke the following make commands

mkdir -p Output/WindowsPhone8.1/ARM

cd Output/WindowsPhone8.1/ARM

../../../configure \

--toolchain=msvc \

--disable-programs \

--disable-d3d11va \

--disable-dxva2 \

--arch=arm \

--as=armasm \

--cpu=armv7 \

--enable-thumb \

--enable-shared \

--enable-cross-compile \

--target-os=win32 \

--extra-cflags="-MD -DWINAPI\_FAMILY=WINAPI\_FAMILY\_PHONE\_APP -D\_WIN32\_WINNT=0x0603 -D\_\_ARM\_PCS\_VFP" \

--extra-ldflags="-APPCONTAINER -MACHINE:ARM -subsystem:console -opt:ref WindowsPhoneCore.lib RuntimeObject.lib PhoneAppModelHost.lib -NODEFAULTLIB:kernel32.lib -NODEFAULTLIB:ole32.lib" \

--prefix=../../../Build/WindowsPhone8.1/ARM

make

make install

Generated libraries can be found in Build/WindowsPhone8.1/ARM folder specified in --prefix option above

# **Troubleshooting**

Building ARM libraries using Visual Studio **armasm** ARM assembler requires proper setup of gas-preprocessor.pl and **Perl** in your **MSYS2** environment. Please follow the setup instruction closely.

# **Windows Store Certification, File I/O, and Other Details**

FFmpeg is an excellent alternative to Windows Media Foundation, which supports a small fraction of the codecs that FFmpeg has. Fortunately, FFmpeg does not use any prohibited Windows or CRT APIs, and apps linked to the DLLs built with this method pass the latest Windows App Certification Kit. However, there are important things to remember.

First, don't forget to make sure your app package includes all the necessary FFmpeg DLLs in the root folder. In your Visual C++ project, you should include links to the FFmpeg DLLs in the root project folder, making sure to set "Build Action" to "Content" and "Copy to Output Directory" to "Copy Always". It is not enough just to link to the import libraries. You will fail certification if you require your users to install the FFmpeg DLLs separately.

You will also need to supply a custom file I/O context to any AVFormatContext rather than relying on avio\_xxx functions, because the standard I/O functions utilize CRT I/O that is not supported in WinRT. (You won't fail certification, but your app will be unable to access the file specified in the AVFormatContext.filename member, unless potentially it is located in the app's local storage). The file I/O context needs to be initialized to point to your read, write, and seek functions, with the opaque member pointing to a struct containing an IRandomAccessStream^ that you obtain when opening a file using proper WinRT IO calls in the Windows::Storage namespace. (You need a struct to hold the IRandomAccessStream^, since you cannot cast a void\* to a ref handle). Your read, write, and seek functions should cast the void\* opaque argument to the struct\* you define, and then use the IRandomAccessStream^ for all I/O.

Since FFmpeg I/O calls are synchronous, but the IRandomAccessStream members are all asynchronous, you will need to utilize create\_task and wait() on all I/O calls in your IO context implementation functions. Note that this necessitates that any FFmpeg functions utilizing file I/O be executed on a WORKER thread, as WinRT will generate an exception if you use wait() in the UI thread. The best solution here is to create WinRT-friendly wrapper functions of the major FFmpeg functions you will utilize, by using the concurrency::create\_async function, for example:

IAsyncOperation<int>^ av\_read\_frame\_async(AVFormatContext\* s, AVPacket\* pkt)

{

return create\_async([s, pkt]()->int

{

return av\_read\_frame(s, pkt);

});

}