# Compiling FFmpeg QSV on Intel Platform

**White Paper**

*November 2022*

Document Number: 728030-2.0

# Contents

## Figures

## Tables

# Revision History

| Date | Revision | Description |
|---|---|---|
| November 2022 | 2.0 | Added Windows* Platform Libvpl support. |
| April 2022 | 1.0 | Initial release. |

§

# 1.0 Introduction

Ffmpeg is a free opensource software, which is widely used in multimedia processing, like video editing, video encoding, decoding and transcoding. For video processing function, like compression and decompression, each silicon vendor has its own implementation. With intel we provide QSV hardware in intel GPU as hardware acceleration.

QSV usually is not enabled by default in FFmpeg release - different procedures are needed for Linux* and Windows* OS.

In this guide, we provide steps to enable QSV using Intel Media SDK for hardware acceleration in both the OS. Additionally, we provide methods to enable OpenCL, QSV and so on, and to build FFmpeg debug version. Developers can use this guide to set up FFmpeg for hardware encoder, decoder, and filters on Intel GPU.

## 1.1 Terminology

**Table 1. Terminology**

| Term | Description |
|---|---|
| FFmpeg | An audio/video conversion tool |
| QSV | Quick Sync Video |
| MSYS2 | A Software distro and building platform for Windows* |
| Intel® Media SDK | API to access hardware-accelerated |
| VAAPI | Video Acceleration API |
| Libva | An implementation of VAAPI |
| Gmmlib | Intel Graphics Memory Management Library |
| OneVPL | OneAPI Video Processing Library |

## 1.2 Reference Documents

Log in to the Resource and Documentation Center (rdc.intel.com) to search for and download the document numbers listed in the following table. Contact your Intel field representative for access.

***Note:*** Third-party links are provided as a reference only. Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

**Table 2. Reference Documents**

| Document | Document No./Location |
|---|---|
| Linux* Installation Guide | https://gist.github.com/Brainiarc7/4f831867f8e55d35cbcb527e15f9f116 |
| MSYS2 | https://www.msys2.org/ |
| FFmpeg Compilation Guide | https://trac.ffmpeg.org/wiki/CompilationGuide |
| Intel GPU runtime | https://github.com/intel/compute-runtime/releases/tag/22.12.22749 |
| FFmpeg HW accel Intro | https://trac.ffmpeg.org/wiki/HWAccelIntro |
| Media SDK | https://github.com/Intel-Media-SDK |
| oneVPL | https://github.com/oneapi-src/oneVPL |

§

# 2.0 Build FFmpeg QSV on Windows*

This chapter describes necessary steps required for compiling FFmpeg and QSV from scratch, with Windows* via MSYS2.

Firstly, uninstall MSYS2 to prevent conflicts with already installed packages.

## 2.1 MSYS2 Setup

Download and install latest MSYS2 https://www.msys2.org/ - follow the installation instructions from web page.

Open C:\msys64\mingw64.exe, update the package database and base packages

```
$pacman -Syu
```

Run "MSYS2 MSYS" from Start menu. Update the rest of the base packages with

```
$pacman -Su
```

Install below required packages to start compiling:

```
$pacman -S --needed base-devel mingw-w64-x86_64-toolchain
$pacman -S python patch tar automake libtool autoconf git nasm
$pacman -S mingw-w64-x86_64-yasm mingw-w64-cross
$pacman -S mingw-w64-x86_64-cmake
$pacman -S mercurial
```

## 2.2 Build Libmfx

Install Intel media sdk dispatcher. it will be installed in c:/msys64/mingw64.

```
$git clone https://github.com/lu-zero/mfx_dispatch.git
$autoreconf -i
$./configure --prefix=/mingw64
$make -j8 && make install
```

autoreconf -i caused error:

Makefile.am:54: error: 'libintel_gfx_api-x64.a' is not a standard libtool library name

Makefile.am:54: did you mean 'libintel_gfx_api-x64.la'?

Makefile.am:51: error: 'libintel_gfx_api-x86.a' is not a standard libtool library name

Makefile.am:51: did you mean 'libintel_gfx_api-x86.la'?

Solution: Change the File extension to 'la' of line 51 and line 54 in Makefile.am

libintel_gfx_api-x86.a ->： libintel_gfx_api-x86.la libintel_gfx_api-x64.a -> libintel_gfx_api-x64.la

## 2.3 Build Libvpl

OneVPL is the successor of MediaSDK (libmfx), it is recommended to use on gen12+ above graphics like platform TigerLake, Alderlake and above. FFmpeg supports OneVPL with libvpl.

You can choose any of libmfx or libvpl as dispatcher frontend.

In our test, libvpl will outperform libmfx performance on Tiger Lake platform.

```
$git clone https://github.com/oneapi-src/oneVPL.git
cd <vpl-repo-clone-location>
$mkdir _build
$cd _build
cmake --build . --config Release --target install
```

The generated file will be in a folder _vplinstall, which includes libraries, lib and dll files.

## 2.4 Build libx264

This library provides H.264 video encoder, and it requires FFmpeg to be configured with --enable-gpl --enable-libx264.

```
$git clone https://code.videolan.org/videolan/x264.git
$mkdir build
$cd build
$../configure --prefix="/mingw64" --bindir="/mingw64/bin" --
enable-static
$make install
```

## 2.5 Build OpenCL

FFmpeg includes groups of OpenCL video filters, to enable compilation of these filters. You need to configure FFmpeg with --enable-opencl on Windows* platform, and also install opencl header files and opencl-icd.

```
$pacman -S mingw-w64-x86_64-opencl-headers mingw-w64-x86_64-
opencl-icd
```

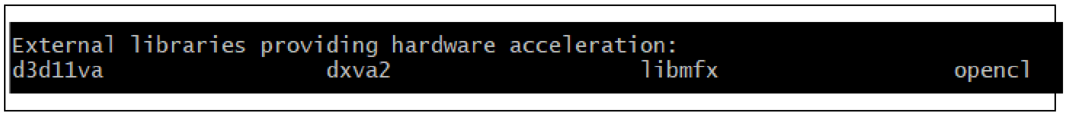
## 2.6 Build FFmpeg

On Windows* platform "--enable-libmfx" is used to enable QSV with MediaSDK.

```
$git clone https://github.com/FFmpeg/FFmpeg.git
$cd ffmpeg
$mkdir build
$cd build
$PKG_CONFIG_PATH="/mingw64/lib/pkgconfig" ../configure --enable-
static --disable-shared --enable-dxva2 --enable-d3d11va --
disable-ffplay --disable-sdl2 --enable-libx264 --enable-gpl --
enable-libmfx --enable-opencl  --enable-logging --disable-doc --
arch=x86_64 --target-os=mingw64 --disable-mmx --disable-stripping
--prefix=../build/
$make -j8
$make install
```

Validate and ensure the acceleration libraries are correct.

**Figure 1. Hardware Acceleration Libraries libmfx**



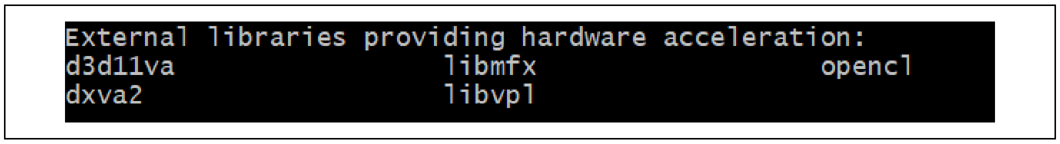
```
External libraries providing hardware acceleration:
d3d11va                   dxva2                     libmfx                    opencl
```

You can also choose "--enable-libvpl" option to enable QSV with oneVPL.  Libvpl pkgconfig path need to be configured manually.

```
$git clone https://github.com/FFmpeg/FFmpeg.git
$cd ffmpeg
$mkdir build
$cd build
$PKG_CONFIG_PATH="/mingw64/lib/pkgconfig:/C/msys64/home/ojuan/_vp
linstall/lib/pkgconfig:$PKG_CONFIG_PATH" ../configure --enable-
static --disable-shared --enable-dxva2 --enable-d3d11va --
disable-ffplay --disable-sdl2 --enable-libx264 --enable-gpl --
enable-libvpl --enable-opencl  --enable-logging --disable-doc --
arch=x86_64 --target-os=mingw64 --disable-mmx --disable-stripping
--prefix=../build/
$make -j8
$make install
```

Validate and ensure the acceleration libraries are correct.

**Figure 2. Hardware Acceleration Libraries libvpl**



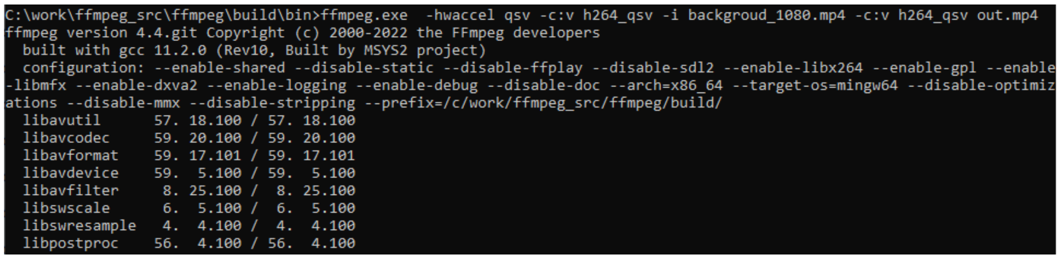
```
External libraries providing hardware acceleration:
d3d11va                   libmfx                    opencl
dxva2                     libvpl
```

In the MSYS2 MingW64 Windows*, the binary was generated at /c/ffmpeg/build/bin folder, use the following command to validate if QSV was built correctly.

```
$./ffmpeg.exe -hwaccel qsv -c:v h264_qsv -i backgroud_1080.mp4 -
c:v h264_qsv out.mp4
```

If you run ffmpeg.exe directly on the Windows* command, it will fail. It is because the generated ffmpeg.exe has dependency with library under C:\msys64\mingw64\bin. You can choose to add it to PATH environment or copy all the dll to /ffmpeg/build/bin. You can validate with above command again.

**Figure 3. FFmpeg Run with Windows* Command**



```
C:\work\ffmpeg_src\ffmpeg\build\bin>ffmpeg.exe  -hwaccel qsv -c:v h264_qsv -i backgroud_1080.mp4 -c:v h264_qsv out.mp4
ffmpeg version 4.4.git Copyright (c) 2000-2022 the FFmpeg developers
  built with gcc 11.2.0 (Rev10, Built by MSYS2 project)
  configuration: --enable-shared --disable-static --disable-ffplay --disable-sdl2 --enable-libx264 --enable-gpl --enable
-libmfx --enable-dxva2 --enable-logging --enable-debug --disable-doc --arch=x86_64 --target-os=mingw64 --disable-optimiz
ations --disable-mmx --disable-stripping --prefix=/c/work/ffmpeg_src/ffmpeg/build/
  libavutil      57. 18.100 / 57. 18.100
  libavcodec     59. 20.100 / 59. 20.100
  libavformat    59. 17.101 / 59. 17.101
  libavdevice    59.  5.100 / 59.  5.100
  libavfilter     8. 25.100 /  8. 25.100
  libswscale      6.  5.100 /  6.  5.100
  libswresample   4.  4.100 /  4.  4.100
  libpostproc    56.  4.100 / 56.  4.100
```

## 2.6.1 Debug FFMPEG

To enable debug build, please configure FFmpeg again with --disable-optimizations --enable-debug=3

You can debug FFmpeg with MSYS2 Window as below:

```
$ gdb --args ./ffmpeg.exe -h
Reading symbols from ./ffmpeg.exe...
(gdb) b main
Breakpoint 1 at 0x140035865: file
C:/work/ffmpeg_src/ffmpeg/fftools/ffmpeg.c, line 4846.
(gdb) r
Starting program: C:\ffmpeg\build\bin\ffmpeg.exe -h
[New Thread 8524.0x4048]
[New Thread 8524.0x3974]
[New Thread 8524.0x42e8]
Thread 1 hit Breakpoint 1, main (argc=2, argv=0x18a49d55f70) at
C:/work/ffmpeg_src/ffmpeg/fftools/ffmpeg.c:4846
4846        init_dynload();
(gdb) bt
#0  main (argc=2, argv=0x18a49d55f70)
    at C:/ffmpeg/fftools/ffmpeg.c:4846
```

You can run below FFmpeg command to check if d3d11va works:

```
$./ffmpeg.exe -hwaccel d3d11va -init_hw_device d3d11va -
hwaccel_flags allow_profile_mismatch -v verbose  -i
backgroud_1080.mp4 -pix_fmt yuv420p -f rawvideo -vsync
passthrough -vframes 500 -y output.yuv
```

§

# 3.0 Build FFmpeg QSV on Linux*

This chapter describes the steps required for compiling FFmpeg and QSV on Linux*.

Build platform is validated with Ubuntu* 18.04LTS and 20.04 LTS.

## 3.1 Environment Preparation

Ensure the platform is up to date

```
$ sudo apt update && sudo apt -y upgrade && sudo apt -y dist-
upgrade
```

Install the dependencies

```
$sudo apt-get -y install autoconf automake build-essential
libass-dev libtool pkg-config texinfo zlib1g-dev libva-dev cmake
mercurial libdrm-dev libvorbis-dev libogg-dev git libx11-dev
libperl-dev libpciaccess-dev libpciaccess0 xorg-dev intel-gpu-
tools opencl-headers libwayland-dev xutils-dev ocl-icd-* libssl-
dev
```

Setup build environment:

```
$mkdir -p ~/vaapi
$mkdir -p ~/ffmpeg_build
$mkdir -p ~/ffmpeg_sources
$mkdir -p ~/bin
```

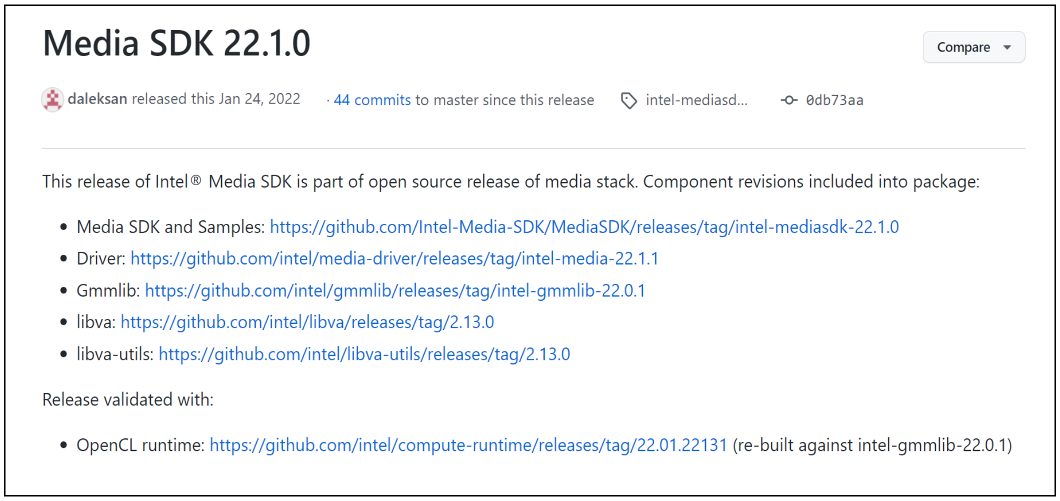
## 3.2 Build Media SDK and Related Libraries

There are dependencies between Media SDK, media driver, GMM lib and libva. Check the dependencies, taking MediaSDK release 22.1.0 as an example:

https://github.com/Intel-Media-SDK/MediaSDK/releases/tag/intel-mediasdk-22.1.0

The libraries list all the related packages and their version. In this guide, we will use below mentioned version to build.

**Figure 4. Media SDK and Dependency Components**



Media SDK 22.1.0

daleksan released this Jan 24, 2022 · 44 commits to master since this release intel-mediasd... 0db73aa

This release of Intel® Media SDK is part of open source release of media stack. Component revisions included into package:

- Media SDK and Samples: https://github.com/Intel-Media-SDK/MediaSDK/releases/tag/intel-mediasdk-22.1.0
- Driver: https://github.com/intel/media-driver/releases/tag/intel-media-22.1.1
- Gmmlib: https://github.com/intel/gmmlib/releases/tag/intel-gmmlib-22.0.1
- libva: https://github.com/intel/libva/releases/tag/2.13.0
- libva-utils: https://github.com/intel/libva-utils/releases/tag/2.13.0

Release validated with:

- OpenCL runtime: https://github.com/intel/compute-runtime/releases/tag/22.01.22131 (re-built against intel-gmmlib-22.0.1)

### 3.2.1 Build Libva

Libva is an implementation for VA-API (Video Acceleration API), VA-API is an open-source library and API specification, which provides access to graphics hardware acceleration capabilities for video processing. It consists of a main library and driver-specific acceleration backends for each supported hardware vendor.

```
$cd ~/vaapi
$wget
https://github.com/intel/libva/archive/refs/tags/2.13.0.tar.gz -O
libva.tar.gz
$tar zxf libva.tar.gz --one-top-level=libva --strip-components 1
$cd libva
$./autogen.sh --prefix=/usr --libdir=/usr/lib/x86_64-linux-gnu
$make -j$(nproc) VERBOSE=1
$sudo make -j$(nproc) install
$sudo ldconfig -vvvv
```

### 3.2.2 Build Gmmlib

Gmmlib is Intel Graphics Memory Management Library, it provides device specific buffer management graphics compute runtime for OpenCL and intel Media driver for VAAPI

```
$cd ~/vaapi/workspace
$wget https://github.com/intel/gmmlib/archive/refs/tags/intel-
gmmlib-22.0.1.tar.gz
$tar zxf intel-gmmlib-22.0.1.tar.gz --one-top-level=gmmlib --
strip-components 1
$mkdir build && cd build
```

```
$cmake -DCMAKE_BUILD_TYPE= Release ../gmmlib
$make -j$(nproc)
$sudo make -j$(nproc) install
```

### 3.2.3 Build Intel Media Driver

Gmmlib is Intel Graphics Memory Management Library, it provides device specific buffer management graphics compute runtime for OpenCL and intel Media driver for VAAPI

```
$cd ~/vaapi/workspace
$wget https://github.com/intel/media-
driver/archive/refs/tags/intel-media-22.1.1.tar.gz
tar zxf intel-media-22.1.1.tar.gz --one-top-level=media-driver --
strip-components 1
$cd media-driver
$mkdir -p ~/vaapi/workspace/build_media
$cd ~/vaapi/workspace/build_media
$cmake ../media-driver \
-DBS_DIR_GMMLIB=$PWD/../gmmlib/Source/GmmLib/ \
-DBS_DIR_COMMON=$PWD/../gmmlib/Source/Common/ \
-DBS_DIR_INC=$PWD/../gmmlib/Source/inc/ \
-DBS_DIR_MEDIA=$PWD/../media-driver \
-DCMAKE_INSTALL_PREFIX=/usr \
-DCMAKE_INSTALL_LIBDIR=/usr/lib/x86_64-linux-gnu \
-DINSTALL_DRIVER_SYSCONF=OFF \
-DLIBVA_DRIVERS_PATH=/usr/lib/x86_64-linux-gnu/dri
$make -j$(nproc)
$sudo make -j$(nproc) install
```

Now, export environment variables as shown below, or you can declare them to ~/.bashrc

```
LIBVA_DRIVERS_PATH=/usr/lib/x86_64-linux-gnu/dri
LIBVA_DRIVER_NAME=iHD
```

### 3.2.4 Build Libva-Utils

libva-utils is a collection of utilities and examples to exercise VA-API in accordance with the libva project. Such as Vainfo, it can be used to validate a platform's supported features.

```
$cd ~/vaapi
$wget https://github.com/intel/libva-
utils/archive/refs/tags/2.13.0.tar.gz -O libva-utils.tar.gz
$tar zxf libva-utils.tar.gz --one-top-level=libva-utils --strip-
components 1
$cd libva-utils
$./autogen.sh --prefix=/usr --libdir=/usr/lib/x86_64-linux-gnu
$make -j$(nproc)
$sudo make -j$(nproc) install
```

### 3.2.5 Install Neo Driver

We suggest using the latest Neo driver manually. Please refer to the following link for procedure.

https://github.com/intel/compute-runtime/releases/tag/22.01.22131

Test that neo driver is installed correctly with clino.

```
$clinfo
```

### 3.2.6 Build Intel Media SDK

Intel media SDK is one API to access hardware-accelerated video encode, decode and pre & post processing on inte Graphics hardware platform. Please follow below steps to build the Media SDK. It will be installed to default location /opt/intel/mediasdk

```
$cd ~/vaapi
$wget https://github.com/Intel-Media-
SDK/MediaSDK/archive/refs/tags/intel-mediasdk-22.1.0.tar.gz
$tar zxf intel-mediasdk-21.1.3.tar.gz --one-top-level=MediaSDK --
strip-components 1
$cd MediaSDK
$mkdir -p build
$cd build
$cmake ../
$make -j$(nproc)
$sudo make -j$(nproc) install
```

When the above is done, please issue a reboot:

```
$sudo reboot
```

## 3.3 Build FFMPEG with x264 and QSV

### 3.3.1 Build Nasm

Nasm is an assembler for x86 optimization used by x264 and FFmpeg.

```
$cd ~/ffmpeg_sources
$wget http://www.nasm.us/pub/nasm/releasebuilds/2.14.02/nasm-
2.14.02.tar.gz
$tar xzvf nasm-2.14.02.tar.gz
$cd nasm-2.14.02
./configure --prefix="$HOME/ffmpeg_build" --bindir="$HOME/bin"
$make -j$(nproc) VERBOSE=1
$make -j$(nproc) install
$make -j$(nproc) distclean
```

### 3.3.2 Build libx264

This library provides H.264 video encoder, it requires FFmpeg to be configured with --enable-gpl --enable-libx264

```
$cd ~/ffmpeg_sources
$git clone http://git.videolan.org/git/x264.git
cd x264/
$PATH="$HOME/bin:$PATH" ./configure --prefix="$HOME/ffmpeg_build"
--enable-static --enable-pic --bit-depth=all
$PATH="$HOME/bin:$PATH" make -j$(nproc) VERBOSE=1
$make -j$(nproc) install VERBOSE=1
$make -j$(nproc) distclean
```

### 3.3.3 Build FFMPEG

In this step we build FFmpeg with enabled x264, OpenCL and intel Linux* acceleration libmfx and vaapi.

```
$cd ~/ffmpeg_sources
$git clone https://github.com/FFmpeg/FFmpeg
$cd FFmpeg
$PATH="$HOME/bin:$PATH"
PKG_CONFIG_PATH="$HOME/ffmpeg_build/lib/pkgconfig:/opt/intel/medi
asdk/lib/pkgconfig" ./configure \
  --pkg-config-flags="--static" \
  --enable-static --disable-shared \
  --prefix="$HOME/ffmpeg_build" \
  --bindir="$HOME/bin" \
  --extra-cflags="-I$HOME/ffmpeg_build/include" \
  --extra-ldflags="-L$HOME/ffmpeg_build/lib" \
  --extra-cflags="-I/opt/intel/mediasdk/include" \
  --extra-ldflags="-L/opt/intel/mediasdk/lib" \
  --extra-ldflags="-L/opt/intel/mediasdk/plugins" \
  --enable-libmfx \
  --enable-vaapi \
  --enable-opencl \
  --disable-debug \
  --enable-libdrm \
  --enable-gpl \
  --enable-runtime-cpudetect \
  --enable-libx264 \
  --enable-openssl \
  --enable-pic \
  --extra-libs="-lpthread -lm -lz -ldl" \
  --enable-nonfree
$PATH="$HOME/bin:$PATH" make -j$(nproc)
$make -j$(nproc) install
$make -j$(nproc) distclean
$hash -r
```

Please use the following command to validate if QSV was built correctly.

```
$./ffmpeg -hwaccel qsv -c:v h264_qsv -i backgroud_1080.mp4 -c:v
h264_qsv out.mp4
```

To enable debug option, add add the '--enable-debug=3' configuration flag, remove –'disable-debug' and omit the distclean step and you'll find the ffmpeg_g binary under the source subdirectory.

§