Qualcomm Technologies, Inc.

# Qualcomm® High Efficiency Video Coding (HEVC) Video Encoder

## User Manual

80-PC029-2 Rev. C

December 12, 2018



Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

# Revision history

| Revision | Date | Description |
|---|---|---|
| A | April 2017 | Initial release |
| B | July 2017 | Added Advanced RISC Machines (ARM) support |
| C | December 2018 | Added prediction statistics based quality propagation (see Table 5-1) |

# Contents

# Tables

# 1 Introduction

## 1.1 Purpose

The Qualcomm High Efficiency Video Coding (HEVC) Video Encoder is a video encoder based on H.265/HEVC, a video encoding method to provide high quality video at a low bit rate. The Qualcomm HEVC Video Encoder can be used in broad applications that require real time encoding environment with multithreading or offline encoding for enhanced visual quality.

This document provides the following:

- Chapter 2 – Quick guide for running the software
- Chapter 3 – Detailed parameter descriptions for advanced users
- Chapter 4 – Support for using the Qualcomm HEVC Video Encoder with FFmpeg
- Chapter 5 – Full parameter list

H265/HEVC is a new compression standard, designed to be the successor to the H264 standard.

The primary benefit of H265 is that it creates data compression ratios almost twice that obtained by H264 for the same encoding quality. These ratios are obtained by refining the principles used for H264 compression, such as expansion of the pattern comparison areas. This increases the encoding complexity, but affects the decoding very little.

The new compression standard means large savings in terms of storage for saving 4K, and even 8K which the H265/HEVC supports.

## 1.2 Conventions

Function declarations, function names, type declarations, attributes, and code samples appear in a different font, for example, `#include`.

Code variables appear in angle brackets, for example, `<number>`.

Commands to be entered appear in a different font, for example, **`copy a:*.* b:`**.

Button and key names appear in bold font, for example, click **Save** or press **Enter**.

## 1.3 Technical assistance

For assistance or clarification regarding information in this document, submit a case to Qualcomm Technologies, Inc. (QTI) at hevc-encoder-support@qualcomm.com.

# 2 Parameter Setting Examples

Various parameters are controlled by the user to provide a flexible encoding environment. For simplicity, these parameters are set with default values so the user can try the software without complex parameter settings. The following examples show how to use the software with minimal parameter settings.

## 2.1 Input and output specification

The minimum parameter setting requires the input and output file name with the input format specification as follows:

```
QcHevcEncode -i <input YUV file name> -b <output bitstream file name> -w
<width> -h <height> -InputDepth <bit-depth> -f <number of frames>
```

The input file must be a raw YUV file with 4:2:0 planar chroma format.

The input bit-depth is specified through "-InputDepth" option. The Windows and Linux encoder binary supports both 8 and 10 input bit-depth, and the Advanced RISC Machines (ARM) encoder binary is always 8.

The Windows and Linux encoder binaries support 8-bit and 10-bit internal bit-depth and output bit-depth. The ARM encoder binary supports 8-bit internal bit-depth and output bit-depth.

For detailed descriptions with more parameter options, see Section 3.1.1.

## 2.2 Rate control

The rate control can be used by specifying the target bit rate to control the output bit stream size. There are two different rate control methods supported, single-pass and two-pass.

For single-pass rate control, the following example is encoding with the target bit rate set to 1 Mbps.

```
QcHevcEncode -RC 1 -BR 1000 -i <input> -b <output> -w <width> -h <height> -
r <frame rate> -f <number>
```

For two-pass rate control, encoding must be performed twice (see Section 3.4). In the following example, the first line is for the first pass encoding, and the second line is for the second pass encoding with the target bit rate set to 1 Mbps.

```
QcHevcEncode -RC 2 -RcPass 1 -i <input> -b <output> -w <width> -h <height>
-r <frame rate> -InputDepth <bit-depth> -f <number>

QcHevcEncode -RC 2 -RcPass 2 -BR 1000 -i <input> -b <output> -w <width> -h
<height> -r <frame rate> -InputDepth <bit-depth> -f <number>
```

NOTE: The current evaluation binary only supports single-pass rate control on ARM, and two-pass rate control on Windows and Linux.

## 2.3 Encoding speed control

For fast encoding, multithreading or parallel processing is used by default (see Section 3.3.2). The user can also select the preset mode to improve encoding speed at the cost of quality degradation, as described in Section 3.3.1.

Also, with multithreading support, a sequence can be divided into multiple segments and coded independently, which is called *multiple segments coding*. See Section 3.1.2 for details and an example.

# 3 Control Parameter Usage

## 3.1 Input and output related parameters

### 3.1.1 Basic parameters

Table 3-1 lists the basic input and output related parameters. For usage examples, see Section 2.1.

**Table 3-1 Basic input and output related parameters**

| Parameter | Description |
|---|---|
| -i <input file name> | The input file must be a raw YUV file with 4:2:0 planar chroma format. The length of the file name should be less than 256 characters. |
| -b <output bitstream file name> | This is the output file name, which is the encoded bit stream. The length of the file name should be less than 256 characters. |
| -w <width><br>-h <height> | Specify the resolution of the input video, which is also the same with the output video resolution.<br><br>NOTE: They should be multiples of 2. |
| -r <input frame rate> | Specifies how many frames to be displayed per second and is used to calculate the encoded bit stream bit rate. It can be either an integer or a floating point number. |
| -InputDepth <bit-depth> | Specifies the bit depth of the input YUV video. It is set to 10 by default. To encode 8-bit video, this parameter must be set to 8.<br><br>NOTE: In this version, only 8 and 10 bits are supported. The output bit depth is fixed to 10 regardless of the input bit depth. |
| -f <number of frames> | Specifies the number of frames to be encoded. |
| -fs <number of frames to be skipped> | When this is set, the specified number of frames are skipped from the input YUV file, so the encoding begins from the designated frame in the sequence. |
| -o <output YUV file name> | When this is specified, the reconstructed video is stored as a file with the specified file name. The bit depth of the output video is 10-bit and the format is a raw YUV file with 4:2:0 planar chroma format. |

### 3.1.2 Multiple segments coding

A video clip or sequence can be divided into multiple segments and each segment can be coded independently. This feature is only applicable to two pass encoding. To use this feature, the following parameters must be specified:

```
-fs <L> -f <M> -SegmentTotal <N> -SegmentIdx <n>
```

*where,*

- L – Denotes the number of frames to be skipped for the current segment
- M – Represents the number of frames in the current segment
- N – Represents the total number of segments
- n – Specifies the index for the segment to be coded (Note: The segment index starts from 0.)

When multiple segments coding is used with the rate control, the same parameters must be used in the first and the second pass, i.e., the number of total segments and the number of frames in each segment must be the same in the first and the second pass.

For example, if a sequence with 300 frames is divided into three segments with 120, 100, and 80 frames in each segment, the multiple segments coding can be used as follows:

```
-fs 0 -f 120 -SegmentTotal 3 -SegmentIdx 0
-fs 120 -f 100 -SegmentTotal 3 -SegmentIdx 1
-fs 220 -f 80 -SegmentTotal 3 -SegmentIdx 2
```

## 3.2 High-level settings

Table 3-2 lists the parameters related to the high-level encoding control.

**Table 3-2 High-level encoding control parameters**

| Parameter | Description |
|---|---|
| -c <GOP structure> | Changes the group of pictures (GOP) structure.<br>0: All Intra<br>1: Low Delay P<br>2: Low Delay B<br>3: Random Access |
| -IDR <IDR picture period> | Sets the period of instantaneous decoder refresh (IDR) picture. If set to -1, no such frame is encoded except for the first encoded frame. |
| -I <intra picture period> | Sets the period of intra picture. If set to -1, no such frame is encoded except for the first encoded frame. If the intra picture period is not the multiples of the GOP size, it is changed to a multiple of GOP size before encoding.<br>**Note:** In this version, GOP size is fixed to 8 (RA) or 4 (LP/LB).<br><br>The intra picture period should not be larger than the IDR picture period. If the intra picture period is larger than the IDR picture period, it is changed to the IDR picture period. Also, when the IDR picture period is larger than 0, the intra picture period should not be infinite by setting it to -1. In this case, the intra picture period is changed to the IDR picture period. |

| Parameter | Description |
|---|---|
| -AUD <value> | If the value is set to 1, the access unit delimiter (AUD) is added into the bit stream, which indicates the slice type values that may be present in the coded picture. The default value is 0. |
| -RepeatHeader <value> | If the value is set to 1, the header information (such as video parameter set, sequence parameter set, and picture parameter set) are coded into the bit stream before every IDR picture. The value cannot be to set to 1 when the IDR picture period is infinite. The default value is 0. |

# 3.3 Encoding complexity control

## 3.3.1 Presets

```
-Preset <value>
```

Preset parameters allow users to control encoding speed by choosing among six different complexity levels.

Preset 0 is the most complex and Preset 5 is the simplest. For example, Preset 0 provides the best performance in terms of the encoded visual quality at the given bit rate, while Preset 5 achieves fast encoding speed at the cost of lower quality. The default value is 3.

## 3.3.2 Multithreading

Multithreading parameters allow running encoding using multiple processing cores. The user can use the following parameter to specify the number of threads to be used by the encoder:

```
-Threads <number of threads>
```

If the number of threads is set to 0, the number of maximum available cores will be used. If it is set to 1, no parallel processing will occur. The default value is 0.

# 3.4 Rate control

Rate control generates a bit stream with the size according to the target bit rate.

## 3.4.1 Single-pass encoding

Single-pass encoding encodes a given video at a specified target bit rate, through one-pass encoding.

**Table 3-3 Single pass parameter settings**

| Parameter | Description |
|---|---|
| -RC 1 -BR <target bitrate> | Target bit rate specified in Kbps. |

### 3.4.2 Two-pass encoding

Two-pass encoding maximizes coding performance by utilizing statistical information collected during the first-pass encoding and generates bitstream in the second-pass with better rate control performance than single-pass encoding.

In the first pass, encoding generates a set of data to be used during the second pass. Once these data are generated, they can be used multiple times to code the same input video at different target bit rates.

**Table 3-4 First and second pass parameter settings**

| Parameter | Description |
|---|---|
| ***First pass*** | |
| -RC 2 -RcPass 1 | Once the data files are generated, place them where the second pass encoding is performed. |
| -RcStatsFileGuid <index number> | Puts an infix index number in the rate control statistics file name generated during the first pass encoding.<br>The statistics file name has the form of:<br>`vbr_stat_<a>_s<b>`<br>*where*,<br>a – The index number specified by RcStatsFileGuid parameter.<br>b – The segment index used in the multiple segments coding. The index number specified by RcStatsFileGuid parameter can be used to indicate a different video sequence. |
| ***Second pass*** | |
| -RC 2 -RcPass 2 -BR <target bitrate> | Target bit rate specified in Kbps.<br>Excluding this parameter, the same parameter values should be used between the first and second pass encoding to achieve precise target bit rate matching.<br>The default buffer size is set to four times the target bit rate. |
| -RcStatsFileGuid <index number> | Same as first-pass. |
| -RcBufSize <VBV buffer size> | The size of the video buffering verifier (VBV) buffer (kbits) is specified when it enables VBV in two-pass encoding.<br>The default is 0 (where VBV is disabled). |
| -RcMaxRate <VBV maximum rate> | This is the maximum local bit rate (kbits/sec). It will be used only if the RcBufufSize parameter is not zero.<br>The default is 0 (where it is disabled). |

**NOTE:** In this version, for Windows and Linux encoder binary, random access structure with two-pass rate control is supported. For ARM encoder binary, low delay structure with single-pass rate control is supported.

## 3.5 Quality control

### 3.5.1 Quantization parameter control

The quantization parameter (QP) can be specified to control bit rate or the quality. If QP is low, the bit rate will increase and quality will improve, and vice versa. Different offsets to the base QP can be applied for different color components.

Table 3-5 lists the related parameter descriptions.

**Table 3-5 QP control parameters**

| Parameter | Description |
|---|---|
| -q <QP> | Sets the base QP value. Its range should be from 0 to 51, inclusive. The default value is 32.<br><br>NOTE: This is ignored when the rate control is in use. |
| -CbQPOffset <value1> -CrQPOffset <value2> | Set QP offsets for the Cb and Cr component, respectively. A negative offset value will improve the corresponding component quality, and a positive offset value will decrease bit rate to code the corresponding component. The default values are 0. |

### 3.5.2 Loop filters

H.265/HEVC provides two loop filters, the deblock filter and the sample adaptive offset (SAO) filter, to improve visual quality. Each filter can be controlled independently using the parameters in Table 3-6.

**Table 3-6 Loop filter parameters**

| Parameter | Description |
|---|---|
| -DB <value> | When set to 1, the deblock filter is enabled. When set to 0, it is disabled.<br>The default value is 1. |
| -s <value> | When set to 1, the SAO filter is enabled. When set to 0, it is disabled.<br>The default value is 1. |

## 3.6 Miscellaneous parameters

Table 3-7 lists other miscellaneous parameters.

**Table 3-7 Miscellaneous parameters**

| Parameter | Description |
|---|---|
| -psnr <value> | When set to 1, peak signal to noise ratio (PSNR) is calculated for each frame.<br>The default value is 0. |
| -ssim <value> | When set to 1, structural similarity (SSIM) is calculated for each frame.<br>The default value is 0. |
| -verbose <value> | When set to 1, encoding information is displayed for each frame.<br>The default value is 0. |
| -csv <file name> | If this parameter is specified, encoding results information is written as a text file. If a file exists with the specified file name, the output is appended to the file. |

# 4 FFmpeg Support

This chapter provides guidance for using the Qualcomm HEVC Video Encoder with FFmpeg.

## 4.1 Installation

Refer to http://ffmpeg.org/ to download the FFmpeg source code, or enter the following command:

```
git clone https://git.ffmpeg.org/ffmpeg.git ffmpeg
```

Before compiling FFmpeg, modify the following:

### Source code modification

1. Copy libqhevc.c to the libavcodec folder.
2. Add the following line to libavcodec/allcodecs.c in the avcodec_register_all function:
   ```
   REGISTER_ENCODER(LIBQHEVCENC,       libqhevcenc);
   ```

### Configure file modification

1. Add the following to the External library support section:
   ```
   --enable-libqhevcenc     enable Qualcomm HEVC encoding [no]
   ```
2. Add the following to the EXTERNAL_LIBRARY_LIST section:
   ```
   libqhevcenc
   ```
3. Add the following in the # external libraries section:
   ```
   libqhevc_encoder_deps="libqhevcenc"
   ```

### Makefile file modification

Add the following to libavcodec/Makefile:

```
OBJS-$(CONFIG_LIBQHEVCENC_ENCODER)        += libqhevc.o
```

### Configure command parameter setting

If using Linux, add the directory where the library file is located to the LD_LIBRARY_PATH variable. For example:

```
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/<…>/HEVC_HQ_Encoder_SDK_10bits/build/linu
x64sharedlib
```

The following is the configure command parameter option:

```
./configure --enable-libqhevcenc --extra-cflags="-
I/<…>/HEVC_HQ_Encoder_SDK_10bits/api/" --extra-ldflags="-
L/<…>/HEVC_HQ_Encoder_SDK_10bits/build/linux64sharedlib" --extra-libs="-
lqhevcenc -ldl"
```

If using Windows in a MSys environment, use the following configure command parameter options:

```
./configure --arch=x86_64 --target-os=mingw32_w64 --cross-prefix=x86_64-
w64-mingw32- --pkg-config=pkg-config --enable-memalign-hack --enable-
libqhevcenc --extra-cflags="-I/<…>/HEVC_HQ_Encoder_SDK_10bits/api/" --
extra-ldflags="-
L/<…>/HEVC_HQ_Encoder_SDK_10bits/build/linuxcrossforwindows64dll" --extra-
libs=-lqhevcenc.dll
```

## 4.2 Parameter usage

The following example shows how to encode a raw YUV sequence with FFmpeg using the Qualcomm HEVC Video Encoder:

```
ffmpeg.exe -s <width x height> -r <frame rate> -pix_fmt yuv420p10 -i <input
YUV file name> -RC 2 -RcPass <1/2> -b:v <bitrate> -vsync passthrough -
vcodec libqhevcenc <output bitstream file name>
```

The input sequence related parameters should precede the input sequence file name parameter, and the encoding option parameters should be placed between the input sequence file name and the output bit stream file name.

If the input sequence has a 10-bit bit depth, the -pix_fmt parameter is set to yuv420p10. The frame rate is specified by the -r parameter.

The use of the Qualcomm HEVC Video Encoder is enabled by the -vcodec libqhevcenc parameter. The -vsync passthrough parameter is specified to avoid any skipping or duplication of frames. The target bit rate is specified by the -b:v parameter. The rate control related parameters are used similar to those described in Section 3.4.

The following is another example where multiple segment coding is used:

```
ffmpeg.exe -s <width x height> -r <frame rate> -pix_fmt yuv420p10 -i <input
YUV file name> -RC 2 -RcPass <1/2> -b:v <bitrate> -vsync passthrough -
vcodec libqhevcenc -SegmentTotal 3 -SegmentIdx 0 -ss 0 -t 3 <output
bitstream file name>

ffmpeg.exe -s <width x height> -r <frame rate> -pix_fmt yuv420p10 -i <input
YUV file name> -RC 2 -RcPass <1/2> -b:v <bitrate> -vsync passthrough -
vcodec libqhevcenc -SegmentTotal 3 -SegmentIdx 1 -ss 3 -t 3 <output
bitstream file name>
```

```
ffmpeg.exe -s <width x height> -r <frame rate> -pix_fmt yuv420p10 -i <input
YUV file name> -RC 2 -RcPass <1/2> -b:v <bitrate> -vsync passthrough -
vcodec libqhevcenc -SegmentTotal 3 -SegmentIdx 2 -ss 6 -t 4 <output
bitstream file name>
```

The segment related parameters, -SegmentTotal and -SegmentIdx, are used as described in Section 3.1.2 . However, instead of the number of frames to be skipped and coded, -ss and -t parameters are used, which specify the time to skip before encoding and the time for encoding duration in the unit of second.

In the example, three segments are coded, where the first segment is the first 3 seconds, the second segment corresponds to the next 3 seconds, and the third segment corresponds to the last 4 second part.

The encoding parameters listed in Chapter 5 are supported, though some parameter names are different. Table 4-1 lists the parameters where the name is different when used with FFmpeg.

**Table 4-1 FFmpeg parameter names**

| FFmpeg parameter name | Description |
|---|---|
| -qp | Defines the QP value. |
| -configuration | Defines the GOP structure selection. |
| -intra_period | Defines the period of intra picture. |
| -idr_period | Defines the period of IDR picture. |
| -sao_enabled | Enables or disables the SAO filter. |
| -deblock_enabled | Enables or disables the deblock filter. |
| -show_psnr | Enables or disables PSNR calculation |
| -show_ssim | Enables or disables SSIM computation. |
| -TotalFrames | Defines the number of frames to be coded. |

# 5 Parameters

**Table 5-1 Windows and Linux encoder binary parameters**

| Parameter | Description | Min | Max | Default |
|---|---|---|---|---|
| i | Input YUV file name | N/A | N/A | test.yuv |
| o | Output YUV file name | N/A | N/A | - |
| b | Output bit stream file name | N/A | N/A | out.265 |
| RepeatHeader | Repeat VPS/SPS/PPS before IDR | 0 | 1 | 0 |
| AUD | Adding the access unit delimiter | 0 | 1 | 0 |
| w | Input YUV width | 64 | 7680 | 1920 |
| h | Input YUV height | 64 | 4320 | 1080 |
| q | QP | 1 | 51 | 32 |
| CbQPOffset | QP offset for Cb | -12 | 12 | 0 |
| CrQPOffset | QP offset for Cr | -12 | 12 | 0 |
| I | Period of intra picture | -1 | MAX_32 | -1 |
| IDR | Period of IDR picture | -1 | MAX_32 | 0 |
| f | Number of frames to be coded | 1 | MAX_UINT | 32 |
| fs | Number of frames to be skipped before encoding | 0 | MAX_UINT | 0 |
| c | GOP structure selection | 0 | 5 | 3 |
| s | Enable/disable SAO | 0 | 1 | 1 |
| r | Frame rate | 1 | 240 | 30 |
| DB | Enable/disable deblock filter | 0 | 1 | 1 |
| RC | Enable/disable rate control | 0 | 4 | 0 |
| ladqp | Prediction statistics based quality propagation | 0 | 1 | 0 |
| RcPass | Rate control pass specification | 0 | 2 | – |
| RcStatsFileGuid | Index number assigned to the rate control statistics file name | 0 | MAX_UINT | 0 |
| BR | Target bit rate in rate control | 0 | MAX_UINT | 40000 |
| RcBufSize | Size of the VBV buffer (kbits) | 0 | MAX_UINT | 0 |
| RcMaxRate | Maximum local bit rate (kbits/sec) | 0 | MAX_UINT | 0 |
| SegmentTotal | Total number of segments in the multiple segments coding | 1 | MAX_UINT | 1 |
| SegmentIdx | Index of the segment to be coded in the multiple segments coding | 0 | MAX_UINT | 0 |
| Threads | Number of threads used for parallel processing | 0 | 64 | 0 |
| Preset | Encoding speed control | 0 | 5 | 3 |
| psnr | Enable/disable PSNR calculation | 0 | 1 | 0 |

| Parameter | Description | Min | Max | Default |
|---|---|---|---|---|
| ssim | Enable/disable SSIM computation | 0 | 1 | 0 |
| verbose | Enable/disable displaying log for each frame | 0 | 1 | 0 |
| csv | File name to store encoding results information | n/a | N/A | – |
| InputDepth | Input bit depth | 8 | 10 | 10 |
| MAX_32 is the maximum value for a 32-bit signed integer variable, which is 0x7FFFFFFF.<br>MAX_UINT is the maximum value for a 32-bit unsigned integer variable, which is 0xFFFFFFFF. | | | | |

### Table 5-2 ARM encoder binary parameters

| Parameter | Description | Min | Max | Default |
|---|---|---|---|---|
| i | Input YUV file name | N/A | N/A | - |
| o | Output YUV file name | N/A | N/A | - |
| b | Output bit stream file name | N/A | N/A | out.265 |
| w | Input YUV width | 64 | 7680 | 832 |
| h | Input YUV height | 64 | 4320 | 480 |
| q | QP | 1 | 51 | 32 |
| I | Period of intra picture | -1 | MAX_32 | -1 |
| f | Number of frames to be coded | 1 | MAX_UINT | 1 |
| fs | Number of frames to be skipped before encoding | 0 | MAX_UINT | 0 |
| c | GOP structure selection | 0 | 3 | 1 |
| s | Enable/disable SAO | 0 | 1 | 1 |
| r | Frame rate | 1 | 240 | 30 |
| DB | Enable/disable deblock filter | 0 | 1 | 1 |
| RC | Enable/disable rate control | 0 | 1 | 0 |
| BR | Target bit rate in rate control | 0 | MAX_UINT | 800 |
| Threads | Number of threads used for parallel processing | 0 | 4 | 1 |
| Preset | Encoding speed control | 0 | 4 | 2 |
| psnr | Enable/disable PSNR calculation | 0 | 1 | 0 |
| verbose | Enable/disable displaying log for each frame | 0 | 1 | 0 |
| MAX_32 is the maximum value for a 32-bit signed integer variable, which is 0x7FFFFFFF.<br>MAX_UINT is the maximum value for a 32-bit unsigned integer variable, which is 0xFFFFFFFF. | | | | |

# A References

## A.1 Acronyms and terms

**Table A-1 Acronyms and terms**

| Acronym or term | Definition |
|---|---|
| ARM | Advanced RISC Machines |
| AUD | Access unit delimiter |
| GOP | Group of pictures |
| IDR | Instantaneous decoder refresh |
| PSNR | Peak signal to noise ratio |
| QP | Quantization parameter |
| SAO | Sample adaptive offset |
| SSIM | Structural similarity |
| VBV | Video buffering verifier |