Аппаратный энкодер Imagination E5810MP3

АО "МЦСТ"

11 декабря 2024 г.

1 Введение

Энкодер E5810MP3 поддерживает кодеки H.264 и HEVC. Аппаратура генерирует готовый битовый поток H.264/HEVC, который по необходимости упаковывается FFmpeg/GStreamer в контейнер. Поддерживается кодирование видео с цветовой субдискретизацией 4:2:2 и глубиной цвета 10 бит, а также масштабирование и обрезка входного изображения, даунсемплинг цветовой субдискретизации.

1.1 Примеры кодирования H.264/HEVC

Кодирование HEVC с упаковкой в контейнер:

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc target-bitrate=1000000 ! h265parse ! qtmux ! \
filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-b:v 1000000 output.mp4
```

Кодирование тестового видеопотока из videotestsrc:

GStreamer

```
gst-launch-1.0 videotestsrc num-buffers=60 ! 'video/x-raw,width=1920,height=1080,\
format=NV12,framerate=30/1' ! omxh265enc target-bitrate=20000000 ! h265parse ! qtmux ! \
filesink location=output.mp4
```

Перекодирование видео:

GStreamer

```
gst-launch-1.0 filesrc location=input.mp4 ! qtdemux ! h264parse ! omxh264dec ! \
omxh265enc target-bitrate=1000000 ! h265parse ! qtmux ! filesink location=output.mp4
```

FFmpeg

```
ffmpeg -i input.mp4 -c:v hevcenc_omx -b:v 1000000 output.mp4
```

2 Поддерживаемые входные форматы

GStreamer	FFmpeg	Краткое описание
i420	yuv420p	Planar 420 Y U V 8-bit
y42b	yuv422p	Planar 422 Y U V 8-bit
y444	yuv444p	Planar 444 Y U V 8-bit
nv12	nv12	Planar 420 byte-interleaved UV 8-bit
nv21	nv21	Planar 420 byte-interleaved VU 8-bit
nv16	nv16	Planar 422 byte-interleaved UV 8-bit
nv61	—	Planar 422 byte-interleaved VU 8-bit
nv24	nv24	Planar 444 byte-interleaved UV 8-bit
—	nv42	Planar 444 byte-interleaved VU 8-bit
argb	argb	Single plane 444 (Alpha, R, G, B) 8-bit packed
abgr	abgr	Single plane 444 (Alpha, B, G, R) 8-bit packed
bgra	bgra	Single plane 444 (B, G, R, Alpha) 8-bit packed
rgba	rgba	Single plane 444 (R, G, B, Alpha) 8-bit packed
uyvy	uyvy422	Single plane 422 (Cb, Y, Cr, Y) 8-bit packed
yvyu	yvyu422	Single plane 422 (Y, Cr, Y, Cb) 8-bit packed
yuy2	yuyv422	Single plane 422 (Y, Cb, Y, Cr) 8-bit packed
nv12-10le32	—	Planar 420 byte-interleaved UV 10-bit packed
nv16-10le32	—	Planar 422 byte-interleaved UV 10-bit packed
p010-10le	p010	Planar 420 byte-interleaved UV 16-bit (10 MSBits used)
y444-16le	yuv444p16le	Planar 444 Y U V 16-bit (10 MSBits used)

Пример кодирования видео входного формата yuv444p16le (y444-16le):

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=y444-16le framerate=30/1 ! omxh265enc control-rate=constant \
target-bitrate=8000000 predefined-gop=lowdelay ! 'video/x-h265,profile=(string)main10,\
level=(string)6.2,tier=(string)high' ! h265parse ! qtmux ! filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt yuv444p16le -s:v 1920x1080 -r 30 -i input.yuv \
-c:v hevcenc_omx -profile main10 -level 6.2 -tier high -control-rate constant \
-b:v 8000000 -predefined-gop lowdelay -enc-resolution 422 -enc-bitdepth 10 output.mp4
```

Минимальное входное разрешение 128x64. Максимальное разрешение (до применения кропа и даунскейлинга) и фреймрейт для H.264 ограничены уровнем 5.1 (не более 4096 по каждому измерению и не более 9 437 184 сэмплов яркостной компоненты), для HEVC — уровнем 6.2 (не более 8192 по каждому измерению и не более 35 651 584 сэмплов яркостной компоненты).

Для HEVC в случае, если разрешение превысит 4096x2304 (максимальный размер тайла), кадр будет разделён на несколько тайлов автоматически, если не был разделён вручную опциями num-tile-cols и num-tile-rows на достаточное количество тайлов (при использовании даунскейлинга учитывается разрешение после даунскейлинга).

3 Профили и уровни кодирования

3.1 Профили и уровни кодирования H.264

profile [high] — Профиль кодирования.

- **baseline**
  - I и P slices
  - P кадры могут использовать 2 референсных кадра
  - Все режимы Intra-кодирования:
    - Все 4 16x16 режимов intra-кодирования яркостной компоненты
    - Все 4 8x8 режимов intra-кодирования цветовой компоненты
    - Все 9 4x4 режимов intra-кодирования яркостной компоненты
  - разбиение макроблоков на блоки размера 16x16, 8x8, 16x8, 8x16
  - inter-предсказание с точностью 1/2 и 1/4 пикселя
  - Преобразованное/реконструированное изображение используется в функциях оценки
  - Деблок-фильтрация
- **main**
  - B slices
  - Обобщённое предсказание, т.е. Вперёд/Назад
  - CABAC
  - режим Temporal Direct
  - режим Spatial Direct
- **high**
  - Преобразование 8x8 в дополнение к 4x4
  - Предсказание Intra 8x8
  - Возможность отдельно устанавливать смещения QP для компонент Cb и Cr
- **high10**
  - Кодирование с глубиной цвета 10 бит
- **high422**
  - Кодирование с цветовой субдискретизацией 4:2:2

level [5.2] — Уровень кодирования.

Уровень	Макс. скорость декодирования (макроблоков/с)	Макс. размер кадра (макроблоков)	Макс. битрейт (профили baseline, extended, main) (кбит/с)
1.0	1 485	99	64
1b	1 485	99	128
1.1	3 000	396	192
1.2	6 000	396	384
1.3	11 880	396	768
2.0	11 880	396	2 000
2.1	19 800	792	4 000
2.2	20 250	1 620	4 000
3.0	40 500	1 620	10 000
3.1	108 000	3 600	14 000
3.2	216 000	5 120	20 000
4.0	245 760	8 192	20 000
4.1	245 760	8 192	50 000
4.2	522 240	8 704	50 000
5.0	589 824	22 080	135 000
5.1	983 040	36 864	240 000
5.2	2 073 600	36 864	240 000

Максимальный битрейт для профиля high в 1.25 раза больше такового для профилей baseline, extended и main, в 3 раза больше для профиля high10, в 4 раза для high422.

Кодирование с выставлением профиля и уровня кодирования:

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh264enc control-rate=constant target-bitrate=1000000 \
predefined-gop=random-access ! 'video/x-h264,profile=(string)high,level=(string)5.2' ! \
h264parse ! qtmux ! filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v h264enc_omx \
-profile high -level 5.2 -control-rate constant -b:v 1000000 \
-predefined-gop random-access output.mp4
```

3.2 Профили и уровни кодирования HEVC

profile [main] — Профиль кодирования.

- **main** — профиль для кодирования видео с глубиной цвета 8 бит на канал. Цветовая субдискретизация — 4:2:0
- **main10** — профиль для кодирования видео с глубиной цвета 10 бит на канал
- **rext** — профиль для кодирования видео с цветовой субдискретизацией 4:2:2

level [6.2] — Уровень кодирования.

Уровень	Макс. размер яркостной компоненты (пикс.)	Макс. частота дискретизации яркостной компоненты (пикс./с)	Макс. скорость потока основоного слоя (кбит/с)	Макс. скорость потока высокого слоя (кбит/с)	Мин. степень сжатия
1.0	36 864	552 960	128	—	2
2.0	122 880	3 686 400	1500	—	2
2.1	245 760	7 372 800	3000	—	2
3.0	552 960	16 588 800	6000	—	2
3.1	983 040	33 177 600	10 000	—	2
4.0	2 228 224	66 846 720	12 000	30 000	4
4.1	2 228 224	133 693 440	20 000	50 000	4
5.0	8 912 896	267 386 880	25 000	100 000	6
5.1	8 912 896	534 773 760	40 000	160 000	8
5.2	8 912 896	1 069 547 520	60 000	240 000	8
6.0	35 651 584	1 069 547 520	60 000	240 000	8
6.1	35 651 584	2 139 095 040	120 000	480 000	8
6.2	35 651 584	4 278 190 080	240 000	800 000	6

tier [high] — Слой кодирования.

- **main** — основной слой
- **high** — высокий слой

Кодирование с заданием профиля, уровня и слоя кодирования:

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=constant target-bitrate=1000000 \
predefined-gop=random-access ! 'video/x-h265,profile=(string)main,level=(string)6.2,\
tier=(string)high' ! h265parse ! qtmux ! filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-profile main -level 6.2 -tier high -control-rate constant -b:v 1000000 \
-predefined-gop random-access output.mp4
```

4 Группы кадров (GOP)

Перед инициализацией энкодера необходимо выбрать структуру группы кадров выходного потока. Поддерживается 3 способа задания структуры группы кадров:

- Задание группы кадров с использованием конфигурационного файла
- Использование готовой схемы группы кадров
- Задание группы кадров отдельными параметрами

4.1 Задание группы кадров с использованием конфигурационного файла

```
#======== Coding Structure =============
IntraPeriod                   : 32          # Period of I-Frame ( -1 = only first)
DecodingRefreshType           : 1           # Random Accesss 0:none, 1:CDR, 2:IDR
GOPSize                       : 8           # GOP Size (number of B slice = GOPSize-1)
MinReqSrcBuffer               : 8
#        Type POC QPoffset QPfactor tcOffsetDiv2 betaOffsetDiv2  temporal_id #ref_pics_active #ref_pics reference pictures predict Level
Frame1:  B    8   1        0.442    0            0               0           2                 4         -8 -10 -12 -16     0       Level=0
Frame2:  B    4   2        0.3536   0            0               0           2                 3         -4 -6  4           0       Level=1
Frame3:  B    2   3        0.3536   0            0               0           2                 4         -2 -4  2 6         0       Level=2
Frame4:  B    1   4        0.68     0            0               0           2                 4         -1  1  3 7         0       Level=3
Frame5:  B    3   4        0.68     0            0               0           2                 4         -1 -3  1 5         0       Level=3
Frame6:  B    6   3        0.3536   0            0               0           2                 4         -2 -4 -6 2         0       Level=2
Frame7:  B    5   4        0.68     0            0               0           2                 4         -1 -5  1 3         0       Level=3
Frame8:  B    7   4        0.68     0            0               0           2                 4         -1 -3 -7 1         0       Level=3
```

Листинг 1: Группа кадров Random Access

```
#======== Coding Structure =============
IntraPeriod                   : -1          # Period of I-Frame ( -1 = only first)
DecodingRefreshType           : 0           # Random Accesss 0:none, 1:CDR, 2:IDR
GOPSize                       : 4           # GOP Size (number of B slice = GOPSize-1)
MinReqSrcBuffer               : 1
#        Type POC QPoffset QPfactor tcOffsetDiv2 betaOffsetDiv2  temporal_id #ref_pics_active #ref_pics reference pictures predict Level
Frame1:  B    1   3        0.4624   0            0               0           2                 4         -1 -5 -9 -13       0       Level=0
Frame2:  B    2   2        0.4624   0            0               0           2                 4         -1 -2 -6 -10       0       Level=0
Frame3:  B    3   3        0.4624   0            0               0           2                 4         -1 -3 -7 -11       0       Level=0
Frame4:  B    4   1        0.578    0            0               0           2                 4         -1 -4 -8 -12       0       Level=0
```

Листинг 2: Группа кадров Lowdelay

```
#======== Coding Structure =============
IntraPeriod                   : -1          # Period of I-Frame ( -1 = only first)
DecodingRefreshType           : 0           # Random Accesss 0:none, 1:CDR, 2:IDR
GOPSize                       : 4           # GOP Size (number of B slice = GOPSize-1)
MinReqSrcBuffer               : 1
#        Type POC QPoffset QPfactor tcOffsetDiv2 betaOffsetDiv2  temporal_id #ref_pics_active #ref_pics reference pictures predict Level
Frame1:  P    1   3        0.4624   0            0               0           2                 4         -1 -5 -9 -13       0       Level=0
Frame2:  P    2   2        0.4624   0            0               0           2                 4         -1 -2 -6 -10       0       Level=0
Frame3:  P    3   3        0.4624   0            0               0           2                 4         -1 -3 -7 -11       0       Level=0
Frame4:  P    4   1        0.578    0            0               0           2                 4         -1 -4 -8 -12       0       Level=0
```

Листинг 3: Группа кадров Lowdelay P

```
#======== Coding Structure =============
IntraPeriod                   : 1           # Period of I-Frame ( -1 = only first)
DecodingRefreshType           : 0           # Random Accesss 0:none, 1:CRA, 2:IDR
GOPSize                       : 1           # GOP Size (number of B slice = GOPSize-1)
MinReqSrcBuffer               : 1
#        Type POC QPoffset QPfactor tcOffsetDiv2 betaOffsetDiv2  temporal_id #ref_pics_active #ref_pics reference pictures
```

Листинг 4: Группа кадров All Intra

Параметр	Значение	Описание параметра
IntraPeriod	32	Период I-кадра (-1 - только первый кадр I)
DecodingRefreshType	0	0:нет, 1:CRA, 2:IDR
GOPSize	8	Число кадров в таблице GOP
MinReqSrcBuffer	8	Число буферов для кодирования

Таблица 1: Параметры структуры группы кадров Random Access

Данный метод задания GOP соответствует методу, описанному в HM software manual. Поддерживается и HEVC, и H.264. Для определения параметров кодирования используется текстовый файл. Энкодер поддерживает синтаксис конфигурационного файла версии HM-16.6. Конфигурационный файл можно передать, используя опцию **encodeconfigfile** <filename>, из него будет прочитана структура группы кадров. Структура конфигурационного фаа немного модифицирована:

- **MinReqSrcBuffer [GOPSize]** — Как много буферов кадров необходимо для хранения всех переупорядочиваемых кадров плюс текущего кодируемого кадра. В группах кадров, в которых последний кадр в порядке отображения является первым кодируемым, данное значение обычно равно GOPSize.
- **Level [0]** — Дополнительный параметр в таблице GOP, в конец каждой записи добавляется **Level=x**. Используется RDO и алгоритмом управления битрейтом. Меньшие значения, начиная с нуля, присваиваются кадрам, кодируемым большим количеством бит, которые будут использоваться как референсные кадры для остальных.

Так как опция Level добавляется в конец каждой записи таблицы структуры группы кадров, если значение "predict"нулевое, следующих значений в записи (deltaRPS, #ref_idcs и reference-idcs) быть не должно.

Поведение энкодера при использовании конфигурационного файла несколько отличается от поведения референсного энкодера HM:

- Параметры tcOffsetDiv2, betaOffsetDiv2 и temporal_id игнорируются и не поддерживаются.
- Параметр ref_pics_active, устанавливающий количество активных референсных кадров в списке, игнорируется, так как энкодер поддерживает только 2 референсных кадра.
- Параметр predict и все относящиеся к нему параметры игнорируются, так как энкодер не поддерживает предсказания между множествами референсных кадров.
- DecodingRefreshType = 3 не поддерживается.
- В таблице может быть указано более двух кадров в поле reference pictures. В этом случае как референсные используются 2 кадра, ближайшие к кодируемому по порядку отображения POC.

encodeconfigfile = config-file-path [-1] — Путь до используемого конфигурационного файла.

direct-spatial = 0,1 [1] — Использовать режим кодирования B-кадров spatial direct для H.264 (Иначе будет использован режим temporal direct). При кодировании B-кадров в H.264 поддерживаются режимы temporal direct и spatial direct. Однако, когда один из референсных кадров для текущего B-кадра также является B-кадром, аппаратура поддерживает только режим spatial direct, и параметр должен быть установлен в 1.

Кодирование с группой кадров Random Access из конфигурационного файла:

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=constant target-bitrate=1000000 \
qp-luma=30 encodeconfigfile=ra_main.cfg rc-gop-type=random-access ! h265parse ! \
qtmux ! filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate constant -b:v 1000000 -qp-luma 30 -encodeconfigfile ra_main.cfg \
-rc-gop-type random-access output.mp4
```

#Frame	Type	POC	QPoffset	QPfactor	tcOffsetDiv2	betaOffsetDiv2	temporal_id	#ref_pics_active	#ref_pics	reference pictures	predict	deltaRPS	#ref$_i$dcs	refidcs	Level
Frame1	B	8	1	0.442	0	0	0	2	2	-8 -10	0	X	X	X	0
Frame2	B	4	2	0.3536	0	0	0	2	3	-4 -6 4	0	X	X	X	1
Frame3	B	2	3	0.3536	0	0	0	2	4	-2 -4 2 6	0	X	X	X	2
Frame4	B	1	4	0.68	0	0	0	2	4	-1 1 3 7	0	X	X	X	3
Frame5	B	3	4	0.68	0	0	0	2	4	-1 -3 1 5	0	X	X	X	3
Frame6	B	6	3	0.3536	0	0	0	2	2	-2 2	0	X	X	X	2
Frame7	B	5	4	0.68	0	0	0	2	3	-1 1 3	0	X	X	X	3
Frame8	B	7	4	0.68	0	0	0	2	2	-1 1	0	X	X	X	3

Таблица 2: Таблица структуры группы кадров Random Access

Кодирование с группой кадров Lowdelay P из конфигурационного файла:

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=constant target-bitrate=1000000 \
qp-luma=30 encodeconfigfile=lowdelay_p.cfg rc-gop-type=lowdelay ! h265parse ! qtmux ! \
filesink location=output.mp4
```

FFmpeg

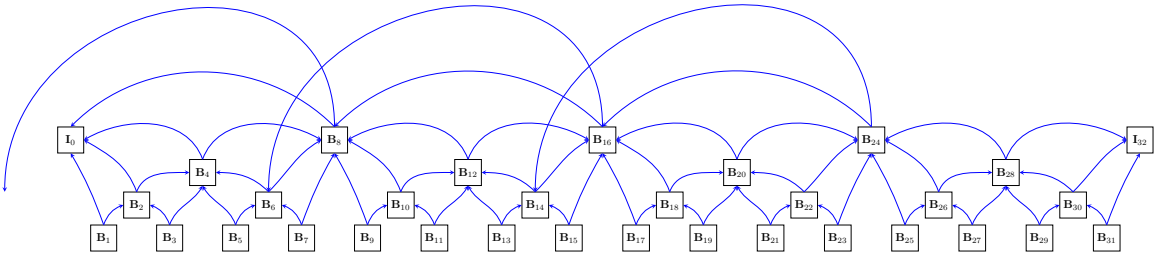
```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate constant -b:v 1000000 -qp-luma 30 -encodeconfigfile lowdelay_p.cfg \
-rc-gop-type lowdelay output.mp4
```

4.2 Использование готовой схемы группы кадров

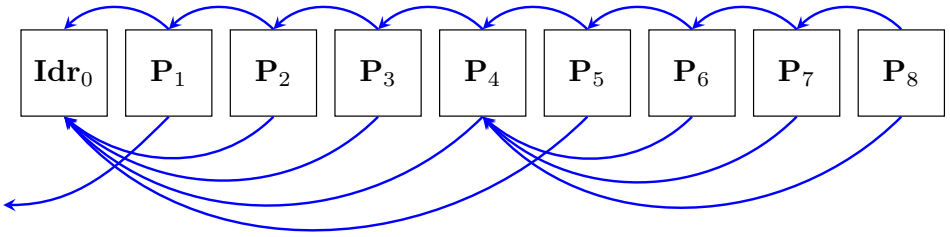
Для задания структуры группы кадров (GOP) можно использовать заранее определённые схемы.

predefined-gop [-1] — использовать предопределённую структуру группы кадров.

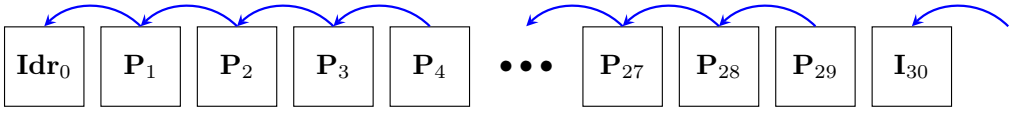
- **random-access** — Описана в предыдущей таблице. Данная группа кадров обеспечивает хорошее сжатие за счёт своей иерархической структуры с B-кадрами, но вводит некоторую задержку из-за необходимости переупорядочивания 7 кадров.



- **lowdelay-p** — В данной группе кадров содержатся только P-кадры, использующие предыдущие кадры в качестве референсных. Эта схема обеспечивает минимальную задержку (меньшую, чем для **random-access**) ценой ухудшения сжатия.



- **lowdelay** — структура аналогична **lowdelay-p**, но вместо P-кадров используются B-кадры.
- **single-p** — Более простая группа кадров, чем **lowdelay**. Все P-кадры используют только предыдущий кадр как референсный. Данная группа кадров обеспечивает худшее качество, но менее требовательна к пропускной способности памяти из-за необходимости выполнять поиск движения только в одном кадре. Гарантируется, что в среднем каждый закодированный P-кадр будет иметь одинаковый размер.



- **all-intra** — кодировать все кадры как IDR

Кодирование с предопределённой группой кадров random-access:

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=constant target-bitrate=1000000 \
predefined-gop=random-access ! h265parse ! qtmux ! filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate constant -b:v 1000000 -predefined-gop random-access output.mp4
```

4.3 Задание группы кадров отдельными параметрами

Простую структуру группы кадров можно задать, используя сокращённый набор параметров. Она будет состоять из начального I-кадра, последовательности из **bframes** B-кадров и одного P-кадра. Последовательность из B-кадров и P-кадра повторяется до тех пор, пока не будет достигнуто количество кадров, равное **intracnt**. Все B-кадры используют смежные P-кадры в качестве референсных (или P- и I-кадры в случае первого и последнего циклов), а P-кадры ссылаются на предыдущий P или I-кадр.

hierarchical = -1..1 [-1] — Использовать иерархическую структуру для B-кадров (только для **bframes** = 3 или 7). Нельзя закодировать более 16 hierarchical bframes.

intracnt = -1, 1..1000 [-1] — число кадров между двумя Intra-кадрами, включая Intra-кадр.

bframes = -1..8 [-1] — Число B-кадров между последовательными P-кадрами, либо между I и P-кадрами (для начала и конца группы кадров). Должно выполняться **intracnt** $= n \cdot$ (**bframes** $+ 1$), $n \in \mathbb{N}$, либо **bframes** $= 0$.

dec-refresh-type = -1..3 [-1] — Определяет тип I-кадра.

Значение	Тип I-кадра
0	none
1	CRA (Открытая группа кадров)
2	IDR (Закрытая группа кадров)
3	Recovery Point SEI

Кодирование с открытой группой кадров, состоящей из I-кадра и повторяющейся структуры из 7 иерархических B-кадров и одного P-кадра:

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=constant target-bitrate=1000000 \
intracnt=40 bframes=7 hierarchical=1 dec-refresh-type=1 ! h265parse ! qtmux ! \
filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate constant -b:v 1000000 -intracnt 40 -bframes 7 -hierarchical 1 \
-dec-refresh-type 1 output.mp4
```

Кодирование с закрытой группой кадров, состоящей из I-кадра и 15 последовательных P-кадров:

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=constant target-bitrate=1000000 \
intracnt=16 bframes=0 hierarchical=0 dec-refresh-type=2 ! h265parse ! qtmux ! \
filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate constant -b:v 1000000 -intracnt 16 -bframes 0 -hierarchical 0 \
-dec-refresh-type 2 output.mp4
```

5 Управление битрейтом

Цель управления битрейтом — обеспечить соблюдение наложенных на битрейт ограничений при минимальной потере качества. Существует несколько режимов управления битрейтом для различных вариантов использования.

Алгоритм управления битрейтом поддерживает работу с двумя стандартными группами кадров (GOP), а также имеет отдельный алгоритм для остальных групп кадров (опция **rc-gop-type**). Алгоритмы для стандартных групп кадров обеспечивают лучшие результаты кодирования для этих GOP, но не работают для GOP, имеющих другую структуру.

5.1 Режимы управления битрейтом

control-rate [variable] — Параметр задаёт режим управления битрейтом.

- **disable** — Отключает управление битрейтом. Используется постоянный параметр квантования QP.
- **variable** — Переменный битрейт. В режиме переменного битрейта алгоритм управления битрейтом стремится достичь наилучшего качества видео в рамках предустановленного целевого битрейта. В данном режиме энкодер усредняет битрейт на большом временном интервале, игнорируя мгновенные изменения битрейта. При этом мгновенный битрейт примерно соответствует сложности видеопотока: сложные сцены кодируются с лучшим качеством, простые кодируются меньшим количеством бит.

  Режим переменного битрейта используется для (пере)кодирования видео и последующего его хранения, когда производительность устройств и скорость сети не являются ограничивающим фактором. В этом режиме качество видео стремится к постоянному, что улучшает его визуальное восприятие.
- **constant** — Постоянный битрейт. Данный режим нацелен на выдачу энкодером видеопотока с постоянным битрейтом, не варьирующимся в зависимости от сложность входного видеопотока. Используется тогда, когда присутствуют сетевые ограничения, либо ограничения скорости хранилища.

Подразумевается, что в этом режиме декодер и энкодер имеют настраиваемый объём буферизации. Режим постоянного битрейта гарантирует, что битрейт выходного видео постоянен. При достаточной буферизации видео может быть передано и декодировано с тем же самым постоянным битрейтом. Когда видеопоследовательность становится слишком простой, энкодер использует биты заполнения (stuffing, параметр **disable-stuffing**), чтобы поддерживать битрейт на желаемом уровне.

- **vcm** — Режим видеоконференции. Режим видеоконференции предназначен для передачи видео по сети с ограниченной пропускной способностью, например для видеоконференций. При использовании данного режима целевой битрейт кодируемого видео трактуется как максимальный. В этом режиме алгоритм управления битрейтом гарантирует, что битрейт видео не превысит целевой, обеспечивая при этом такой выходной видеопоток, который может быть декодирован и отображён на удалённом хосте с минимально возможной задержкой.

  Из-за требований минимальной задержки данный режим не поддерживает кодирование с зависимостями текущего кадра от кадров, которые будут показаны в будущем. (**-predefined-gop random-access**) Из-за жёстких требований к выходному видеопотоку рекомендуется выставлять период Intra-кадров достаточно большим, так как I-кадры из-за своего размера заметно нарушают битрейт видеопоследовательности. Желательно, чтобы данный режим использовался с открытой группой кадров IPP.

  Так как качество видео сильно зависит от качества опорных кадров, QP опорных кадров не контролируется в режиме видеоконференции, Intra-кадры кодируются с постоянным значением QP, равным **qp-luma**.

  В режиме видеоконференции битрейт измеряется в рамках определённого окна, что делает возможным более строгий контроль за пиковым битрейтом, чем в других режимах. Окно битрейта — это количество предыдущих закодированных кадров, которые участвуют в подсчёте среднего битрейта. Размер окна битрейта можно изменить, используя параметр **cbr-buffer-tenths**. Если необходимо ещё лучше контролировать битрейт получаемого видеопотока, то используется режим **cfs**.

- **svbr** — Режим переменного битрейта с ограниченным пиковым битрейтом. Битрейт ограничен алгоритмом HRD, описанным в стандарте H.264 Annex C.

  Так как в этом режиме не используются биты заполнения, алгоритм управления битрейтом не может гарантировать постоянный результирующий битрейт видеопотока. Вместо этого он гарантирует, что видео может быть декодировано декодером с определённой скоростью передачи **buf-bitrate**. Эта скорость может быть больше, чем целевой битрейт кодируемого видео.

  Режим переменного битрейта не накладывает никаких ограничений на пиковый битрейт, в результате чего видеопоток страдает от флуктуаций полосы пропускания. Режим видеотрансляции с переменным битрейтом такие ограничения накладывает.

  Данный режим полезен для устройств с ограниченным размером буферов, либо с ограничениями полосы пропускания.

- **cfs** — Режим фиксированного размера кадров. В некоторых сценариях использования, например для беспроводных дисплеев, необходимо минимизоровать задержку между кодированием и декодированием видео до такой степени, что уже нельзя использовать буферизацию для сглаживания битрейта. Это требует от энкодера гораздо более жесткого контроля за битрейтом. Режим фиксированного размера кадра устанавливает максимальный размер кадра для каждого кодируемого кадра. В этом отношении он ведет себя аналогично режиму видеоконференции с окном битрейта в один кадр.

  При использовании данного режима целевой битрейт используется для расчета целевого размера кадра. Алгоритм управления битрейтом работает так, чтобы гарантировать, что ни один кадр не будет закодирован с размером, превышающим "битрейт"/"частоту кадров"с настраиваемым допуском превышения. Допустимое превышение размера кадра над идеальным значением указывается опцией **rc-cfs-max-margin-perc** как целочисленный процент. Типичное допустимое превышение составляет 10%.

  По умолчанию только Inter-кадры имеют фиксированный размер (изменяется параметром **cfs-on-i-frames**). QP Intra-кадров регулируется параметром **qp-luma**.

  В данном режиме не поддерживаются группы кадров с переупорядочиванием (random-access).

  - **cfs-on-i-frames = 0, 1 [1]** — Фиксированный размер всех кадров, включая Intra. Приводит к наличию заметных артефактов на каждом Intra-кадре.

qp-luma = -13..51 [-13] — При выключенном управлении битрейтом — фиксированное значение QP всех Intra-кадров. Для вычисления QP промежуточных кадров в структуре GOP, стоящих на позиции POC в порядке отображения, будут использованы соответствующие смещения QPoffset. При включенном управлении битрейтом — QP первого кадра, если значение больше вычисленного оптимального значения ($QP_{opt} \geqslant 4$). Для режимов видеоконференции — QP всех Intra-кадров.

Минимальное значение **qp-luma** для десятибитного выходного потока (**enc-bitdepth = 10**) — −12, для восьмибитного — 0. Значение −13 — устанавливать энкодером автоматически.

maxqp = -13/0..51 [-13] — Максимальное значение QP, которое может быть использовано энкодером. Не работает при отключенном управлении битрейтом.

minqp = -13/0..51 [-13] — Минимальное значение QP, которое может быть использовано энкодером. Не работает при отключенном управлении битрейтом.

Кодирование с заданием границ возможных значений параметра квантования:

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=variable target-bitrate=1000000 \
predefined-gop=random-access enable-deblocking=1 minqp=30 maxqp=35 ! h265parse ! \
qtmux ! filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate variable -b:v 1000000 -predefined-gop random-access -enable-deblocking 1 \
-minqp 30 -maxqp 35 output.mp4
```

В случае, если алгоритм управления битрейтом не сможет удовлетворить заданному битрейту при установленных ограничениях, будет применён пропуск кадров.

disable-frame-skipping = 0,1 [0] — Отключить пропуск кадров. В случае использования группы кадров random-access пропуск кадров отключается автоматически.

disable-stuffing = 0,1 [0] — Не использовать заполнение в режиме постоянного битрейта, если результирующий битрейт меньше целевого. Всегда 1 для svbr.

scene-detect-disable = 0,1 [0] — Отключить обнаружение изменений сцены.

buf-bitrate = -1..2147483647 [-1] — Целевой битрейт буферизации. Только для режима переменного битрейта с ограниченным пиковым битрейтом.

cbr-buffer-tenths = 0..2147483647 [10] — Размер буфера гипотетического референсного декодера в десятых долях секунды. Не связан с фактическим размером буферов, выделенных хостом для хранения закодированного потока. Параметр используется только в режимах постоянного битрейта и переменного битрейта с ограниченным пиковым битрейтом. Должен быть установлен в пределах 0.5 — 2 битрейтов с эмпирическим оптимальным значением, равным фактическому значению битрейта (соответствует одной секунде, т.е. cbr-buffer-tenths = 10). Бóльшие значения позволяют более гибко управлять битрейтом, изменяя размеры кадров, но приводят к бóльшим флуктуациям битрейта.

rc-cfs-max-margin-perc = 0..2147483647 [9] — Процент, на который может быть превышен максимальный размер кадра в режиме видеоконференции с постоянным размером кадра.

rc-gop-type [-1] — Использовать алгоритм управления битрейтом, адаптированный для определённого типа групп кадров. Опция используется при задании структуры группы кадров через конфигурационный файл (параметр **encodeconfigfile**).

- **lowdelay**
- **random-access**
- **other**

enable-hrd-param = 0,1 [0] — Включает SEI-сообщения buffering_period и picture_timing. Только для режима постоянного битрейта и режима видеотрансляции с переменным битрейтом. Для активации должна быть отключена опция **disable-vui-params**.

Кодирование с отключенным управлением битрейтом (disable) и группой кадров random-access:

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=disable qp-luma=12 \
predefined-gop=random-access enable-deblocking=1 ! h265parse ! qtmux ! \
filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate disable -qp-luma 12 -predefined-gop random-access -enable-deblocking 1 \
output.mp4
```

Кодирование в режиме переменного битрейта (variable) с группой кадров random-access:

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=variable target-bitrate=1000000 \
qp-luma=12 predefined-gop=random-access enable-deblocking=1 ! h265parse ! qtmux ! \
filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate variable -b:v 1000000 -predefined-gop random-access -enable-deblocking 1 \
output.mp4
```

Кодирование в режиме постоянного битрейта (constant) с группой кадров random-access и битами заполнения:

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=constant target-bitrate=1500000 \
disable-stuffing=0 predefined-gop=random-access enable-deblocking=1 ! h265parse ! \
qtmux ! filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate constant -b:v 1500000 -disable-stuffing 0 -predefined-gop random-access \
-enable-deblocking 1 output.mp4
```

Кодирование в режиме видеоконференции (vcm) с отключенным пропуском кадров:

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=vcm target-bitrate=1000000 \
qp-luma=26 cbr-buffer-tenths=10 disable-frame-skipping=1 intracnt=600 bframes=0 \
hierarchical=0 dec-refresh-type=2 enable-deblocking=1 ! h265parse ! qtmux ! \
filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate vcm -b:v 1000000 -qp-luma 26 -cbr-buffer-tenths 10 \
-disable-frame-skipping 1 -intracnt 600 -bframes 0 -hierarchical 0 -dec-refresh-type 2 \
-enable-deblocking 1 output.mp4
```

Кодирование в режиме переменного битрейта с ограниченным пиковым битрейтом (svbr) с целевым битрейтом буферизации **buf-bitrate = 2000000**:

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=svbr target-bitrate=1000000 \
buf-bitrate=2000000 intracnt=600 bframes=0 hierarchical=0 dec-refresh-type=2 \
enable-deblocking=1 ! h265parse ! qtmux ! filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate svbr -b:v 1000000 -buf-bitrate 2000000 -intracnt 600 -bframes 0 \
-hierarchical 0 -dec-refresh-type 2 -enable-deblocking 1 output.mp4
```

Кодирование с фиксированным размером Inter-кадров (cfs):

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=cfs target-bitrate=1000000 \
qp-luma=20 cfs-on-i-frames=0 rc-cfs-max-margin-perc=10 intracnt=600 bframes=0 \
hierarchical=0 dec-refresh-type=2 enable-deblocking=1 ! h265parse ! qtmux ! \
filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate cfs -b:v 1000000 -qp-luma 20 -cfs-on-i-frames 0 \
-rc-cfs-max-margin-perc 10 -intracnt 600 -bframes 0 -hierarchical 0 -dec-refresh-type 2 \
-enable-deblocking 1 output.mp4
```

Кодирование с фиксированным размером всех кадров (cfs):

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=cfs target-bitrate=1000000 \
qp-luma=20 cfs-on-i-frames=1 rc-cfs-max-margin-perc=10 intracnt=600 bframes=0 \
hierarchical=0 dec-refresh-type=2 enable-deblocking=1 ! h265parse ! qtmux ! \
filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate cfs -b:v 1000000 -cfs-on-i-frames 1 -rc-cfs-max-margin-perc 10 \
-intracnt 600 -bframes 0 -hierarchical 0 -dec-refresh-type 2 -enable-deblocking 1 \
output.mp4
```

5.2 Контекстно-адаптивное управление битрейтом

В процессе кодирования параметр квантования яркостной компоненты QP_Y может изменяться сообразно сложности отдельных блоков. Этот алгоритм называется контекстно-адаптивным управлением битрейтом

(CARC). Он позволяет использовать больше информации для кодирования блоков, на которых артефакты кодирования могут быть более заметны, ценой ухудшения качества блоков с лучшей детализацией (мелкие детали помогают скрыть артефакты).

Если размер кодируемого блока (CTU) составляет 64x64, сложность может быть посчитана отдельно для каждого из подблоков размера 32x32, и 4 различных значения ΔQP могут быть применены раздельно к четвертям блока 64x64. Контролируется параметром **qp-delta-depth**.

qp-delta-depth = 0,1 [1] — Определяет, может ли размер группы квантования HEVC быть меньше размера блока дерева кодирования.

Значение	Размер группы квантования
0	Соответсвует размеру блока дерева кодирования
1 (default)	32x32, если размер блока дерева кодирования 64x64

К каждому блоку яркостной компоненты размера 8x8 применяется преобразование Адамара размера 8x8, затем из подмножества полученных коэффициентов, которое определяется параметром **carc-cutoff**, вычисляется число тех коэффициентов, которые имеют абсолютное значение больше порогового **carc-threshold**, чтобы получить количество коэффициентов $coeff\_count$ для каждого блока — сложнось блока. (Для 10-битного видео алгоритм использует только 8 старших бит)

Значение $coeff\_count$ рассчитывается следующим образом:

```
if ((lookup_classification[y][x] <= carc_cutoff) && ((abs(coeff) >> 3) > carc_threshold))
    coeff_count++
```

Таблица lookup_classification имеет следующий вид:

1	5	4	2	3	4	3	5
6	15	13	8	11	12	10	14
4	13	10	7	9	10	8	12
2	8	7	5	6	7	6	8
4	11	9	6	7	9	7	11
5	12	10	7	9	9	8	11
3	10	8	6	7	8	6	9
5	14	12	8	10	11	9	13

Чтобы вычислить значение ΔQP, которое будет добавлено к изначальному QP, вычисляется разность между ctu_complexity и carc_baseline а полученное значение масштабируется. Ожидается, что carc_baseline — это средняя сложность блока 64x64.

```
complexity_delta = ctu_complexity - carc_baseline
if (complexity_delta < 0)
    scale = carc_neg_scale
    sign = -1
else
    scale = carc_pos_scale
    sign = 1
qp_delta = ( scale * abs(complexity_delta) ) >> ( 9 + carc_shift )
qp_delta = qp_delta * sign
ctu_qp_delta = clip( -carc_neg_range, carc_pos_range, qp_delta )
```

Затем полученное значение суммируется с QP для получения QPy блока CTU, если не выставлен флаг exact_qp для данного CTU:

```
if ( not exact_qp ) {
    if(encode_bitdepth is 10-bit) max_qp = 63
    else max_qp = 51
    ctu_qpy = clip( 0, max_qp, ctu_qpy + ctu_qp_delta )
}
```

Параметры, контролирующие работу CARC:

carc = 0,1 [0] — использовать контекстно-адаптивное управление битрейтом. Включает обновление параметра квантования QP на каждом блоке дерева кодирования (CTU).

carc-baseline = 0..65535 [0] — значение, при пересечении которого алгоритм управления QP переключается с отрицательных значений на положительные.

carc-threshold = 0..65535 [2] — пороговое значение коэффициентов DCT. Если коэффициенты превышают это значение, они будут учтены в сложности CARC.

carc-cutoff = 0..255 [15] — Подмножество коэффициентов в блоке 8x8, которые учитываются в расчёте метрики сложности CARC. Чем больше значение, тем больше коэффициентов учитывается. Параметр устанавливается, если есть необходимость отфильтровать высокочастотные коэффициенты.

carc-shift = 0..255 [2] — При вычислении результирующего ΔQP влияет на величину сдвига вправо.

carc-pos-range = 0..255 [10] — Максимальное значение, на которое CARC может уветичить QP.

carc-pos-scale = 0..255 [0] — Параметр масштабирования для CARC. Используется, когда сложность больше, чем carc-baseline. Чем больше, тем сильнее изменяется QP.

carc-neg-range = 0..255 [10] — Максимальное значение, на которое CARC может уменьшить QP.

carc-neg-scale = 0..255 [0] — Параметр масштабирования для CARC. Используется, когда сложность меньше, чем carc-baseline. Чем больше, тем сильнее изменяется QP.

cabac = 0, 1 (H.264) [1] — Использовать контекстно-адаптивное двоичное арифметическое кодирование (CABAC) вместо контекстно-адаптивного неравномерного кодирования (CAVLC). Для HEVC всегда 1.

ctu-size = 16,32,64 [64] — переопределёние размера блока дерева кодирования. Для H264 всегда равно 16. В стандарте HEVC кадр сначала разбивается на блоки дерева кодирования (CTUs) одинакового размера (за размер CTU принимается размер яркостной компоненты). Размер CTU содержится в заголовке SPS, т.е. все блоки CTU имеют одинаковый размер в пределах видеопоследовательности.

Кодирование с использованием контекстно-адаптивного управления битрейтом:

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=constant target-bitrate=1000000 \
predefined-gop=lowdelay carc=1 qp-delta-depth=1 carc-baseline=0 carc-threshold=2 \
carc-cutoff=15 carc-shift=2 carc-pos-range=10 carc-pos-scale=0 carc-neg-range=10 \
carc-neg-scale=0 ! h265parse ! qtmux ! filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate constant -b:v 1000000 -predefined-gop lowdelay -carc 1 \
-qp-delta-depth 1 -carc-baseline 0 -carc-threshold 2 -carc-cutoff 15 -carc-shift 2 \
-carc-pos-range 10 -carc-pos-scale 0 -carc-neg-range 10 -carc-neg-scale 0 output.mp4
```

6 Постпроцессинг

Первый этап пост-обработки — деблок-фильтрация — предназначен для снижения краевых эффектов (эффектов блочности). Каждый видеокадр при кодировании равномерно разбивается на квадратные блоки одинакового размера, называемые LCU (сокр. от англ. Largest Coding Unit). Каждый LCU может быть разбит при кодировании на четыре квадратных блока CU (сокр. от англ. Coding Unit), каждый из которых, в свою очередь, может быть разбит еще на четыре CU. В результате таких разбиений образуется квадродерево CTU (сокр. от англ. Coding Tree Unit). CU нижнего уровня в CTU, в свою очередь, разбиваются при кодировании на блоки предсказания PU (сокр. от англ. Prediction Unit).

Двумерный разностный сигнал, полученный вычитанием результатов предсказания из кодируемого изображения, подвергается двумерному спектральному преобразованию. Размер матрицы спектрального преобразования определяется размером блока преобразования TU (сокр. от англ. Transform Unit). В HEVC возможны преобразования 4 размеров: 4×4, 8×8, 16×16 и 32×32. Квантование спектральных коэффициентов приводит к возникновению эффекта блочности декодированных изображений. По вполне понятным причинам эффект проявляется как на границах блоков PU, так и на границах блоков TU.

Процедура деблок-фильтрации изменяет значения пикселей, лежащих вблизи вертикальных и горизонтальных линий, образующих на видеокадре эквидистантную сетку с шагом 8 пикселей и являющихся границами TU или PU. Фильтрация производится сначала для вертикальных, а затем для горизонтальных линий.

enable-deblocking = 0, 1 [0] — Использовать деблокирующий фильтр.

Эффективность использования деблокирующего фильтра в HEVC не вызывает сомнений. Особенно эффективным представляется его использование при высоких значениях параметра квантования, когда краевые эффекты особенно велики. При уменьшении значений параметра квантования блочность декодированного изображения снижается, что, в свою очередь, приводит к уменьшению эффективности деблокирующего фильтра.

Второй этап постобработки декодированных изображений — SAO (сокр. от англ. Sample Adaptive Offset) — предназначен для частичной компенсации потерь, обусловленных квантованием спектральных коэффициентов. Для этого к значениям некоторых пикселей изображения после декодирования добавляются смещения, вычисляемые на этапе кодирования и передаваемые в закодированном потоке в виде таблиц для каждого LCU. Отбор пикселей, подвергаемых изменениям, производится по их интенсивности, что и обуславливает нелинейность преобразования. Данный этап постобработки энкодером E5810MP3 **не поддерживается**.

7 Разбиение кадра

7.1 Разбиение кадра на части (slices)

Кадр может быть разбит области, которые могут быть переданы по сети по отдельности. Каждый слайс состоит из целого числа блоков дерева кодирования. Разделение кадра на слайсы улучшает возможности параллельного декодирования видео и повышает устойчивость видеопотока к ошибкам, но уменьшает эффективность кодирования.

slices = 0..255 [0] — Количество частей (slices), на которое будет разбит кадр. Для H264 slice всегда занимает целое число строк, а сам кадр разбивается на части примерно одинакового размера. Кадр разбивается приоритетно по строкам, для более точного разбиения необходимо использовать параметр **slice-ctu-limit**.

slice-ctu-limit = 0..2147483647 [0] — Максимальное число блоков дерева кодирования в одном slice.

nalsize = -1..2147483647 [-1] — Максимальный размер slice (байт).

loop-filter-across-slice = 0,1 [0] — Использовать контурный фильтр для границ slices. Должен быть включен деблокирующий фильтр **enable-deblocking**.

Кодирование с разбиением кадра на 8 частей:

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=variable target-bitrate=1000000 \
predefined-gop=random-access slices=8 enable-deblocking=1 loop-filter-across-slice=1 ! \
h265parse ! qtmux ! filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate variable -b:v 1000000 -predefined-gop random-access -slices 8 \
-enable-deblocking 1 -loop-filter-across-slice 1 output.mp4
```

Кодирование с разбиением кадра на части посредством ограничения максимального количества блоков дерева кодирования в slice:

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=variable target-bitrate=1000000 \
predefined-gop=random-access slice-ctu-limit=12 enable-deblocking=1 \
loop-filter-across-slice=1 ! h265parse ! qtmux ! filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate variable -b:v 1000000 -predefined-gop random-access -slice-ctu-limit 12 \
-enable-deblocking 1 -loop-filter-across-slice 1 output.mp4
```

Кодирование с разбиением кадра на части посредством ограничения максимального размера slice в байтах:

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=variable target-bitrate=1000000 \
predefined-gop=random-access nalsize=1500 enable-deblocking=1 \
loop-filter-across-slice=1 ! h265parse ! qtmux ! filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate variable -b:v 1000000 -predefined-gop random-access -nalsize 1500 \
-enable-deblocking 1 -loop-filter-across-slice 1 output.mp4
```

7.2 Разбиение кадра на тайлы (Только для HEVC)

Представляет из себя разбиение кадра на прямоугольные области, которые могут кодироваться и декодироваться независимо друг от друга. В отличие от слайсов (slices), тайлы не являются отдельными синтаксическими единицами кодирования, за счет чего достигается дополнительное увеличение степени сжатия. С другой стороны, как и в случае со слайсами в AVC, накладные расходы, возникающие из-за сокращения вариантов внутрикадрового предсказания и обновления контекстов энтропийного кодера на границах, пренебрежительно малы.

num-tile-cols = 0..255 [0] — Число столбцов, образующих тайлы.

num-tile-rows = 0..255 [0] — Число строк, образующих тайлы.

user-tile-cols = 0,pos1,pos2,.. [0] — Переопределить начальные позиции тайлов (в CTUs) по горизонтали. Первое значение всегда нулевое. Количество значений должно соответствовать **num-tile-cols**.

user-tile-rows = 0,pos1,pos2,.. [0] — Переопределить начальные позиции тайлов (в CTUs) по вертикали. Первое значение всегда нулевое. Количество значений должно соответствовать **num-tile-rows**.

loop-filter-across-tile = 0,1 [0] — использовать контурный фильтр на границах tiles. Должен быть включен деблокирующий фильтр **enable-deblocking**.

Кодирование с разбиением кадра на 4 тайла одинакового размера:

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=variable target-bitrate=1000000 \
predefined-gop=random-access num-tile-cols=2 num-tile-rows=2 enable-deblocking=1 \
loop-filter-across-tile=1 ! h265parse ! qtmux ! filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate variable -b:v 1000000 -predefined-gop random-access -num-tile-cols 2 \
-num-tile-rows 2 -enable-deblocking 1 -loop-filter-across-tile 1 output.mp4
```

Кодирование с разбиением кадра на 4 тайла с заданными позициями начала тайлов:

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=variable target-bitrate=1000000 \
predefined-gop=random-access num-tile-cols=2 user-tile-cols=0,5 num-tile-rows=2 \
user-tile-rows=0,5 enable-deblocking=1 loop-filter-across-tile=1 ! h265parse ! qtmux ! \
filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate variable -b:v 1000000 -predefined-gop random-access -num-tile-cols 2 \
-user-tile-cols 0,5 -num-tile-rows 2 -user-tile-rows 0,5 -enable-deblocking 1 \
-loop-filter-across-tile 1 output.mp4
```

7.3 Кодирование с использованием нескольких конвейеров

Энкодер E5810MP3 имеет 3 конвейера, которые могут быть использованы либо для кодирования независимых видеопотоков, либо для ускорения кодирования одного видеопотока. При использовании нескольких конвейеров для кодирования одного видеопотока обязательным становится разбиение кадра на slices (столбцы тайлов для HEVC), равное числу конвейеров.

num-pipes-to-use = 1,2,3 [1] — Число конвейеров для кодирования текущего видеопотока. При **num-pipes-to-use = 2,3** и кодировании в H.264 параметр **loop-filter-across-slice** не поддерживается, отсутствие контурной фильтрации может привести к наличию заметных артефактов на границах slices.

Кодирование с использованием трёх конвейеров:

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=variable target-bitrate=1000000 \
predefined-gop=random-access num-pipes-to-use=3 slices=3 num-tile-cols=3 \
enable-deblocking=1 loop-filter-across-slice=1 loop-filter-across-tile=1 ! h265parse ! \
qtmux ! filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate variable -b:v 1000000 -predefined-gop random-access -num-pipes-to-use 3 \
-slices 3 -num-tile-cols 3 -enable-deblocking 1 -loop-filter-across-slice 1 \
-loop-filter-across-tile 1 output.mp4
```

8 Intra-обновление

Intra-обновление позволяет постепенно восстановить корректность воспроизводимого видеопотока при перемотке видео или потере кадра, если в видеопоследовательности только первый кадр является Intra, или Intra-кадры крайне редки. Группа кадров random-access не поддерживается.

irefresh-mode [none] — Выбор режима Intra-обновления.

- **none** — Intra-обновление отключено.
- **cyclic** — Цикличное Intra-обновление по строкам (Обновлять **irefresh-ctu-rows** строк блоков дерева кодирования каждый кадр, увеличивая позицию на **irefresh-increment** строк каждый кадр, циклично сверху вниз).
- **column** — Цикличное Intra-обновление по столбцам (Обновлять **irefresh-ctu-rows** столбцов блоков дерева кодирования каждый кадр, увеличивая позицию на **irefresh-increment** столбцов каждый кадр, циклично слева направо).

irefresh-ctu-rows = 0..65535 [2] — Определяет число Intra-обновляемых строк или столбцов.

irefresh-increment = 0..65535 [1] — Определяет число строк или столбцов, на которое сдвигается Intra-обновляемая область с каждым последующим кадром.

irefresh-period = 0..65535 [0] — Определяет период Intra-обновления.

irefresh-qp-delta = -63..63 [2] — Значение ΔQP, на которое изменяется QP Intra-обновляемых макроблоков.

irefresh-size-increase = 0..65535 [30] — Допустимое увеличение размера кадров с Intra-обновлением, выраженное в процентах от размера остальных кадров.

Intra-обновление рекомендуется использовать с простыми группами кадров с минимальними расстояниями до референсных кадров.

Кодирование с группой кадров из 1 I-кадра и 999 P-кадров с Intra-обновлением каждого кадра по строкам:

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=variable target-bitrate=1000000 \
intracnt=1000 bframes=0 hierarchical=0 dec-refresh-type=2 enable-deblocking=1 \
irefresh-mode=cyclic irefresh-ctu-rows=2 irefresh-increment=1 irefresh-period=0 \
irefresh-qp-delta=2 irefresh-size-increase=30 ! h265parse ! qtmux ! \
filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate variable -b:v 1000000 -intracnt 1000 -bframes 0 -hierarchical 0 \
-dec-refresh-type 2 -enable-deblocking 1 -irefresh-mode cyclic -irefresh-ctu-rows 2 \
-irefresh-increment 1 -irefresh-period 0 -irefresh-qp-delta 2 \
-irefresh-size-increase 30 output.mp4
```

Кодирование с группой кадров из 1 I-кадра и 999 P-кадров с Intra-обновлением каждого кадра по столбцам:

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=variable target-bitrate=1000000 \
intracnt=1000 bframes=0 hierarchical=0 dec-refresh-type=2 enable-deblocking=1 \
irefresh-mode=column irefresh-ctu-rows=2 irefresh-increment=1 irefresh-period=0 \
irefresh-qp-delta=2 irefresh-size-increase=30 ! h265parse ! qtmux ! \
filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate variable -b:v 1000000 -intracnt 1000 -bframes 0 -hierarchical 0 \
-dec-refresh-type 2 -enable-deblocking 1 -irefresh-mode column -irefresh-ctu-rows 2 \
-irefresh-increment 1 -irefresh-period 0 -irefresh-qp-delta 2 \
-irefresh-size-increase 30 output.mp4
```

9 Препроцессинг и постпроцессинг

enc-bitdepth = 8, 10 [8] — Глубина цвета закодированного видео.

enc-resolution = 420, 422 [420] — Цветовая субдискретизация закодированного видео.

csc-mode [-1] — Преобразование цветовых пространств (Только для входных данных с субдискретизацией 4:4:4).

- **none**
- **709-to-601**
- **601-to-709**
- **RGB-to-601-analog**
- **RGB-to-601-digital**
- **RGB-to-601-digital-fs**
- **RGB-to-709**

- **YIQ-to-601**
- **YIQ-to-709**
- **BRG-to-601** — для формата XRGB
- **RBG-to-601** — для формата XBGR
- **BGR-to-601**
- **UYV-to-YUV**

Кодирование видео с входным форматом ARGB.

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=argb framerate=30/1 ! omxh265enc control-rate=variable target-bitrate=1000000 \
predefined-gop=random-access enable-deblocking=1 ! h265parse ! qtmux ! \
filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt argb -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate variable -b:v 1000000 -predefined-gop random-access -enable-deblocking 1 \
output.mp4
```

input-crop = crop_left,crop_width:crop_top,crop_height [-1,-1:-1,-1] — Обрезка видео. Чтобы обрезать левые 100px видео 640x480: crop=100,540:0,480. Обрезка по 80px с каждой стороны: crop=80,480:80,320.

Обрезка 1080p видео по 100 пикселей с каждой стороны до кодирования.

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=variable target-bitrate=1000000 \
predefined-gop=random-access enable-deblocking=1 input-crop=100,1720:100,880 ! \
h265parse ! qtmux ! filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate variable -b:v 1000000 -predefined-gop random-access -enable-deblocking 1 \
-input-crop 100,1720:100,880 output.mp4
```

downscale = downscaled_width x downscaled_height [-1x-1] — Даунскейлинг видео. Максимальный коэффициент даунскейлинга 1/8 для каждого измерения. Выполняется после обрезки входного изображения.

Масштабирование 1080p видео до 720p до кодирования.

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=variable target-bitrate=1000000 \
predefined-gop=random-access enable-deblocking=1 downscale=1280x720 ! h265parse ! \
qtmux ! filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate variable -b:v 1000000 -predefined-gop random-access -enable-deblocking 1 \
-downscale 1280x720 output.mp4
```

output-crop = crop_left,crop_width:crop_top,crop_height [-1,-1:-1,-1] — Обрезка видео при воспроизведении. Чтобы обрезать левые 100px видео 640x480: crop=100,540:0,480. Обрезка по 80px с каждой стороны: crop=80,480:80,320.

Обрезка 1080p видео по 100 пикселей с каждой стороны при декодировании.

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=variable target-bitrate=1000000 \
predefined-gop=random-access enable-deblocking=1 output-crop=100,1720:100,880 ! \
h265parse ! qtmux ! filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate variable -b:v 1000000 -predefined-gop random-access -enable-deblocking 1 \
-output-crop 100,1720:100,880 output.mp4
```

Обрезка 4k видео до нижней правой четверти (1920x1080), даунскейлинг получившегося изображения до 1280x720 и обрезка по 100 пикселей с каждой стороны при декодировании:

GStreamer

```
gst-launch-1.0 filesrc location=input_4k.yuv ! rawvideoparse width=3840 height=2160 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=variable target-bitrate=1000000 \
predefined-gop=random-access enable-deblocking=1 input-crop=1920,1920:1080,1080 \
downscale=1280x720 output-crop=100,1080:100,520 ! h265parse ! qtmux ! \
filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 3840x2160 -r 30 -i input_4k.yuv -c:v hevcenc_omx \
-control-rate variable -b:v 1000000 -predefined-gop random-access -enable-deblocking 1 \
-input-crop 1920,1920:1080,1080 -downscale 1280x720 -output-crop 100,1080:100,520 \
output.mp4
```

10 PCM-кодирование

use-pcm = 0,1 [1] — Использовать Intra PCM-кодирование (кодирование режимов внутрикадрового предсказания).

Следующие настройки применимы только для HEVC, для H.264 используются фиксированные значения.

pcm-depth = 0,8,10 [0] — Число бит, с которым кодируются сэмплы PCM. Либо совпадает с выходной глубиной цвета (H.264, HEVC), либо меньше (только для HEVC). Если 0, то выставляется равным **enc-bitdepth**.

pcm-max-cb-size = 8,16,32 [16] — Максимальный размер блока, для которого применяется PCM-кодирование. Значение должно быть больше или равно **pcm-min-cb-size**, но не должно быть больше размера блоков дерева кодирования. Для H.264 всегда равно 16.

pcm-min-cb-size = 8,16,32 [8] — Минимальный размер блока, для которого применяется PCM-кодирование. Значение не должно быть больше размера блоков дерева кодирования. Для H.264 всегда равно 16.

pcm-loop-filter-disable = 0,1 [1] — Если 0, то к Intra-PCM блокам применяется контурный фильтр. Для H.264 всегда равно 1.

11 Параметры, влияющие на содержимое заголовков

disable-vui-params = 0,1 [1] — Не записывать в заголовок SPS VUI (video usability information) — высокоуровневую информацию о видеоконтенте.

allow-overscan = 0,1 [1] — Записать 1 в поле VUI overscan_appropriate_flag. Обрезанное декодированное изображение подходит для отображения с вылетами развёртки. Только для HEVC.

repeat-sequence-header = 0,1 [0] — Повторять sequence header каждый IDR-кадр (не CRA).

enable-sei-dph = 0,1 [0] — Включает SEI-сообщения Suffix Decoded Picture Hash. Только для HEVC.

select-ntsc = 0,1 [0] — Использовать фреймрейт NTSC. Если включено, фреймрейт 30 соответствует 29.97, 24 соответствует 23.976. Изменяет значение num_units_in_tick в VUI для H.264. Для HEVC изменяет значения vui_num_units_in_tick и vps_num_units_in_tick.

deblock-beta = -6..6 [0] — Значение поля $'beta\_offset\_div2'$ (Для AVC и HEVC).

deblock-tc = -6..6 [0] — Значение поля $'tc\_offset\_div2'$ (Для AVCи HEVC).

video-format = 0..5 [1] — Заполнить в VUI поле video_format.

Значение	Формат
0	COMPONENT
1 (default)	PAL
2	NTSC
3	SECAM
4	MAC
5	UNSPECIFIED

12 Ограничения на векторы движения

subpel-res = pixel, half, quarter [quarter] — Точность векторов движения (пиксель, пол-пикселя, четверть пикселя).

mv-limit-x = 0..2048 [2048](H.264), 0..4095 [4095](HEVC) — Область поиска векторов движения по оси x, в целых пикселях.

mv-limit-y = 0..511 [511](H.264), 0..4095 [4095](HEVC) — Область поиска векторов движения по оси y, в целых пикселях.

Ограничение области поиска векторов движения

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc qp-luma=12 control-rate=disable \
predefined-gop=random-access enable-deblocking=1 subpel-res=quarter mv-limit-x=64 \
mv-limit-y=64 ! h265parse ! qtmux ! filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate disable -qp-luma 12 -predefined-gop random-access -enable-deblocking 1 \
-subpel-res quarter -mv-limit-x 64 -mv-limit-y 64 output.mp4
```

13 Коррекция параметров квантования

qp-cb-offset = -12..12 [0] — Смещение QP цветовой плоскости Cb относительно световой.

qp-cr-offset = -12..12 [0] — Смещение QP цветовой плоскости Cr относительно световой.

qp-chroma-offset = -12..12 [0] — Смещение QP цветовых плоскостей относительно световой.

14 Специальные опции

disable-intra-res-skip = 0,1 [0] — Энкодер может отбросить оставшиеся после квантования Intra-блоков данные (residuals). 1 — всегда кодировать оставшиеся после квантования Intra-блоков данные.

inter-scale-update-by-qp = 0,1 [0] — Inter Scale using Qp update

lowlatency = 0,1 [0] — Выводить выходные буферы сразу после их заполнения, с минимальной задержкой.

15 Сценарии использования

Далее приведены примеры, оптимизированные под конкретные сценарии использования.

Wireless — режим управления битрейтом svbr, кодек H.264, профиль main.

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh264enc control-rate=svbr target-bitrate=1000000 \
buf-bitrate=1000000 predefined-gop=single-p enable-deblocking=1 ! h264parse ! \
qtmux ! filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v h264enc_omx \
-profile main -level 5.2 -control-rate svbr -b:v 1000000 -buf-bitrate 1000000 \
-predefined-gop single-p -enable-deblocking 1 output.mp4
```

RTC — режим управления битрейтом vcm.

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=vcm target-bitrate=1000000 \
qp-luma=20 cbr-buffer-tenths=10 hierarchical=0 bframes=0 intracnt=300 \
dec-refresh-type=2 enable-deblocking=1 ! h265parse ! qtmux ! \
filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate vcm -b:v 1000000 -qp-luma 20 -cbr-buffer-tenths 10 -hierarchical 0 \
-bframes 0 -intracnt 300 -dec-refresh-type 2 -enable-deblocking 1 output.mp4
```

Camera — режим управления битрейтом cfs.

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=cfs target-bitrate=1000000 \
buf-bitrate=1000000 qp-luma=20 cfs-on-i-frames=0 rc-cfs-max-margin-perc=10 \
predefined-gop=single-p enable-deblocking=1 ! h265parse ! qtmux ! \
filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate cfs -b:v 1000000 -qp-luma 20 -cfs-on-i-frames 0 \
-rc-cfs-max-margin-perc 10 -predefined-gop single-p -enable-deblocking 1 output.mp4
```

RTP — режим управления битрейтом vcm, nalsize 1500.

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=vcm target-bitrate=1000000 \
qp-luma=20 cbr-buffer-tenths=10 hierarchical=0 bframes=0 intracnt=300 \
dec-refresh-type=2 enable-deblocking=1 loop-filter-across-slice=1 nalsize=1500 ! \
h265parse ! qtmux ! filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate vcm -b:v 1000000 -qp-luma 20 -cbr-buffer-tenths 10 -hierarchical 0 \
-bframes 0 -intracnt 300 -dec-refresh-type 2 -enable-deblocking 1 \
-loop-filter-across-slice 1 -nalsize 1500 output.mp4
```

Lowlatency — lowlatency.

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=vcm target-bitrate=1000000 \
qp-luma=20 cbr-buffer-tenths=10 hierarchical=0 bframes=0 intracnt=300 \
dec-refresh-type=2 enable-deblocking=1 lowlatency=1 ! h265parse ! qtmux ! \
filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate vcm -b:v 1000000 -qp-luma 20 -cbr-buffer-tenths 10 -hierarchical 0 \
-bframes 0 -intracnt 300 -dec-refresh-type 2 -enable-deblocking 1 -lowlatency 1 \
output.mp4
```

Lowlatency RTP — lowlatency, nalsize 1500.

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=vcm target-bitrate=1000000 \
qp-luma=20 cbr-buffer-tenths=10 hierarchical=0 bframes=0 intracnt=300 \
dec-refresh-type=2 enable-deblocking=1 loop-filter-across-slice=1 nalsize=1500 \
lowlatency=1 ! h265parse ! qtmux ! filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate vcm -b:v 1000000 -qp-luma 20 -cbr-buffer-tenths 10 -hierarchical 0 \
-bframes 0 -intracnt 300 -dec-refresh-type 2 -enable-deblocking 1 \
-loop-filter-across-slice 1 -nalsize 1500 -lowlatency 1 output.mp4
```

SSIM — оптимизированные настройки для метрики ssim, disable-intra-res-skip, carc.

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=disable qp-luma=20 \
predefined-gop=random-access enable-deblocking=1 disable-intra-res-skip=1 carc=1 \
qp-delta-depth=1 carc-baseline=0 carc-threshold=2 carc-cutoff=15 carc-shift=2 \
carc-pos-range=10 carc-pos-scale=0 carc-neg-range=10 carc-neg-scale=0 ! h265parse ! \
qtmux ! filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate disable -qp-luma 20 -predefined-gop random-access -enable-deblocking 1 \
-disable-intra-res-skip 1 -carc 1 -qp-delta-depth 1 -carc-baseline 0 -carc-threshold 2 \
-carc-cutoff 15 -carc-shift 2 -carc-pos-range 10 -carc-pos-scale 0 -carc-neg-range 10 \
-carc-neg-scale 0 output.mp4
```

PSNR — оптимизированные настройки для метрики psnr, disable-intra-res-skip 0, carc 0.

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh265enc control-rate=disable qp-luma=20 \
predefined-gop=random-access enable-deblocking=1 disable-intra-res-skip=0 carc=0 ! \
h265parse ! qtmux ! filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v hevcenc_omx \
-control-rate disable -qp-luma 20 -predefined-gop random-access -enable-deblocking 1 \
-disable-intra-res-skip 0 -carc 0 output.mp4
```

H.264 Detail — повышенная детализация для H.264 видео, disable-intra-res-skip, inter-scale-update-by-qp.

GStreamer

```
gst-launch-1.0 filesrc location=input.yuv ! rawvideoparse width=1920 height=1080 \
format=nv12 framerate=30/1 ! omxh264enc control-rate=disable qp-luma=20 \
predefined-gop=random-access enable-deblocking=1 disable-intra-res-skip=1 \
inter-scale-update-by-qp=1 ! h264parse ! qtmux ! filesink location=output.mp4
```

FFmpeg

```
ffmpeg -f rawvideo -pix_fmt nv12 -s:v 1920x1080 -r 30 -i input.yuv -c:v h264enc_omx \
-control-rate disable -qp-luma 20 -predefined-gop random-access -enable-deblocking 1 \
-disable-intra-res-skip 1 -inter-scale-update-by-qp 1 output.mp4
```