



## **PixelFormatConverter lib**

**С++ программная библиотека конвертирования  
изображений в различные форматы пикселей**

Версия библиотеки: **1.0**

Дата релиза библиотеки: **03.01.2020**

Версия документа: **1.0**

[www.zaplatnikov.com](http://www.zaplatnikov.com)

# ОГЛАВЛЕНИЕ

1. ВЕРСИИ ДОКУМЕНТА .....	4
2. ВЕРСИИ ПРОГРАММНОЙ БИБЛИОТЕКИ .....	4
3. ОБЗОР .....	4
4. ПОДДЕРЖИВАЕМЫЕ ФОРМАТЫ ПИКСЕЛОВ .....	4
5. КОНВЕРТАЦИЯ МЕЖДУ ФОРМАТАМИ .....	5
5.1. RGBR (RGB24) в BGRB (BGR24) .....	5
5.2. RGBR (RGB24) в Y800 .....	5
5.3. RGBR (RGB24) в UYVY .....	5
5.4. RGBR (RGB24) в YUY2 .....	6
5.5. RGBR (RGB24) в YUV1 .....	6
5.6. RGBR (RGB24) в NV12 .....	6
5.7. BGRB (BGR24) в RGBR (RGB24) .....	6
5.8. BGRB (BGR24) в Y800 .....	7
5.9. BGRB (BGR24) в UYVY .....	7
5.10. BGRB (BGR24) в YUY2 .....	7
5.11. BGRB (BGR24) в YUV1 .....	7
5.12. BGRB (BGR24) в NV12 .....	8
5.13. Y800 в RGBR (RGB24) .....	8
5.14. Y800 в BGRB (BGR24) .....	8
5.15. Y800 в UYVY .....	8
5.16. Y800 в YUY2 .....	9
5.17. Y800 в YUV1 .....	9
5.18. Y800 в NV12 .....	9
5.19. UYVY в RGBR (RGB24) .....	10
5.20. UYVY в BGRB (BGR24) .....	10
5.21. UYVY в Y800 .....	10
5.22. UYVY в YUY2 .....	10
5.23. UYVY в YUV1 .....	11
5.24. UYVY в NV12 .....	11
5.25. YUY2 в RGBR (RGB24) .....	11
5.26. YUY2 в BGRB (BGR24) .....	12
5.27. YUY2 в Y800 .....	12
5.28. YUY2 в UYVY .....	12
5.29. YUY2 в YUV1 .....	12
5.30. YUY2 в NV12 .....	13
5.31. YUV1 в RGBR (RGB24) .....	13
5.32. YUV1 в BGRB (BGR24) .....	13
5.33. YUV1 в Y800 .....	13
5.34. YUV1 в UYVY .....	14
5.35. YUV1 в YUY2 .....	14
5.36. YUV1 в NV12 .....	14

5.37. NV12 в RGBA (RGBA24) .....	15
5.38. NV12 в BGRA (BGRA24) .....	15
5.39. NV12 в Y800 .....	16
5.40. NV12 в UYVY .....	16
5.41. NV12 в YUY2 .....	17
5.42. NV12 в YUV1 .....	17
6. СТРУКТУРЫ ДАННЫХ .....	17
7. ОПИСАНИЕ КЛАССА PixelFormatConverter .....	19
7.1. Объявление класса PixelFormatConverter .....	19
7.2. Метод Convert(...) .....	20
7.3. Метод GetVersion(...) .....	20
7.4. Метод isFourccCodeValid(...) .....	20
7.5. Метод GetSupportedFourccCodes() .....	20
8. ПРИМЕР ИСПОЛЬЗОВАНИЯ .....	21

## 1. ВЕРСИИ ДОКУМЕНТА

Таблица 1 – Версии документа.

Версия	Дата релиза	Что изменено
1.0	03.01.2020	Первая версия документа.

## 2. ВЕРСИИ ПРОГРАММНОЙ БИБЛИОТЕКИ

Таблица 2 – Версии программной библиотеки.

Версия	Дата релиза	Что изменено
1.0	30.12.2019	Первая версия программной библиотеки. Реализована конвертация между форматами пикселей RGBR (RGB24), BGRB (BGR24), UYVY, Y800, YUY2, YUV1 и NV12.

## 3. ОБЗОР


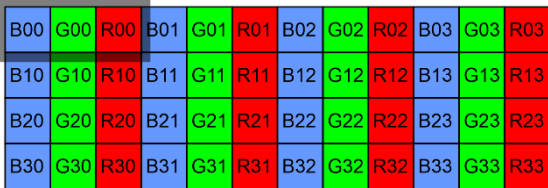
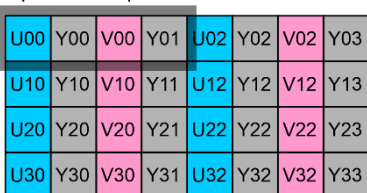
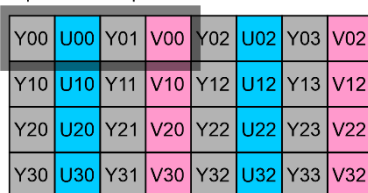
**PixelFormatConverter lib** – это C++ программная библиотека, предназначенная для конвертации изображений в различные форматы пикселей (далее – библиотека). Библиотека имеет простой интерфейс. Библиотека распространяется исходными кодами и совместима с любыми операционными системами, поддерживающими компилятор языка C++ (стандарт C++11). Библиотека включает следующие файлы исходного кода:

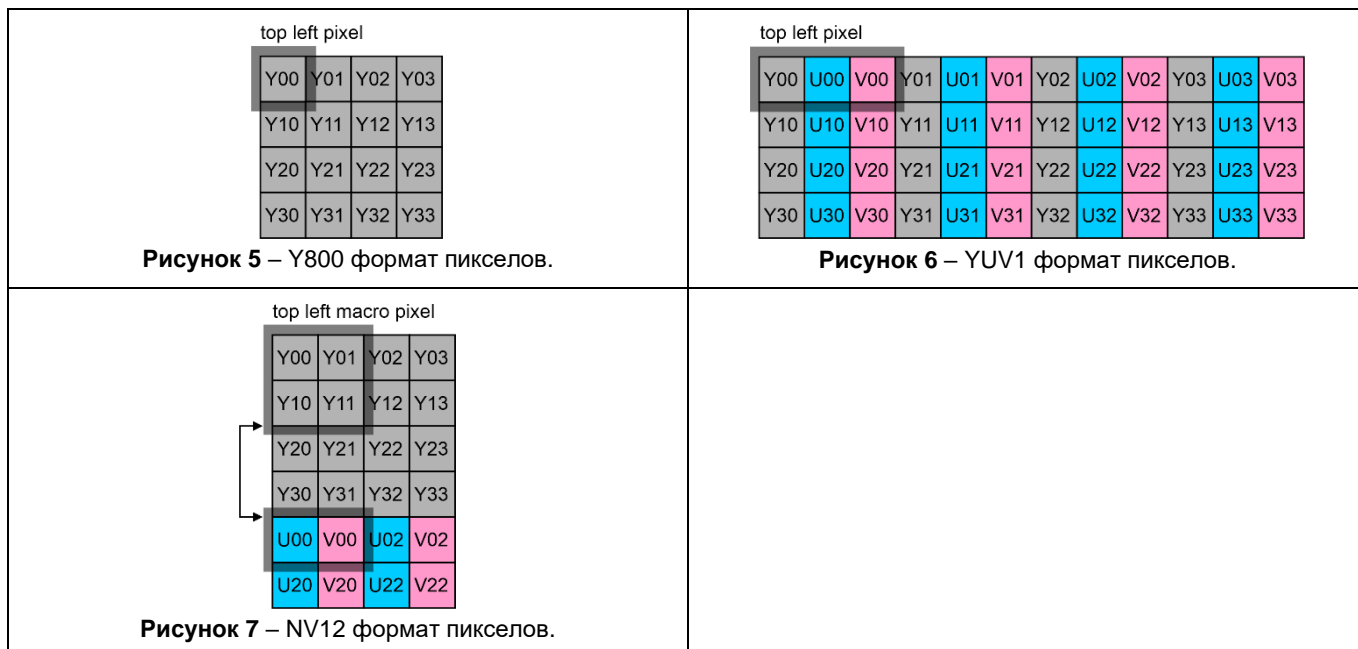
- **VideoDataStructures.h** – заголовочный файл, описывающий структуры данных для изображений и кадров видео;
- **PixelFormatConverter.h** – заголовочный файл, содержащий описание единственного программного класса **PixelFormatConverter**;
- **PixelFormatConverter.cpp** – файл исходного кода, содержащий реализацию методов программного класса **PixelFormatConverter**.

## 4. ПОДДЕРЖИВАЕМЫЕ ФОРМАТЫ ПИКСЕЛОВ

Библиотека поддерживает следующие форматы пикселей: RGBR (RGB24), BGRB (BGR24), UYVY, Y800 (градации серого), YUY2, YUV1 и NV12. Числовые значения форматов пикселей (значение кода FOURCC) определено перечислением **ValidFourccCodes**, объявленном в файле **VideoDataStructures.h**. Библиотека поддерживает конвертацию между указанными форматами. В таблице 3 приведены иллюстрации расположения байт пикселей в различных форматах для изображения размером **4x4** пиксела.

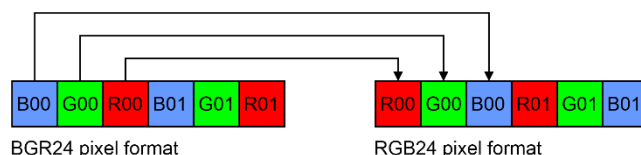
Таблица 3 – Иллюстрации расположения байт данных изображений размером 4x4 пиксела в различных форматах.

<p>top left pixel</p>  <p><b>Рисунок 1</b> – RGBR (RGB24) формат пикселей.</p>	<p>top left pixel</p>  <p><b>Рисунок 2</b> – BGRB (BGR24) формат пикселей.</p>
<p>top left macro pixel</p>  <p><b>Рисунок 3</b> – UYVY формат пикселей.</p>	<p>top left macro pixel</p>  <p><b>Рисунок 4</b> – YUY2 формат пикселей.</p>



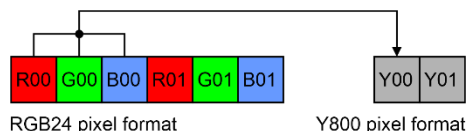
## 5. КОНВЕРТАЦИЯ МЕЖДУ ФОРМАТАМИ

### 5.1. RGBR (RGB24) в BGRB (BGR24)



**Рисунок 8** – Конвертация RGBR (RGB24) в BGRB (BGR24). Только замена байт местами.

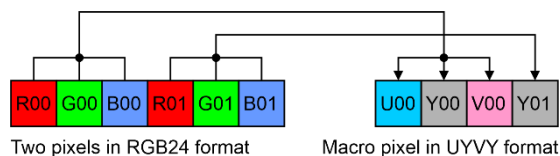
### 5.2. RGBR (RGB24) в Y800



**Рисунок 9** – Конвертация RGBR (RGB24) в Y800.

$Y00 = 0.299 * R00 + 0.587 * G00 + 0.114 * B00$	(1)
---	-----

### 5.3. RGBR (RGB24) в UYVY



**Рисунок 10** – Конвертация RGBR (RGB24) в UYVY.

$Y00 = 0.299 * R00 + 0.587 * G00 + 0.114 * B00$ $Y01 = 0.299 * R01 + 0.587 * G01 + 0.114 * B01$ $U00 = 0.492 * (B00 - Y00) + 128, \text{if } U00 > 255 \text{ then } U00 = 255, \text{if } U00 < 0 \text{ then } U00 = 0$ $V00 = 0.877 * (R00 - Y00) + 128, \text{if } V00 > 255 \text{ then } V00 = 255, \text{if } V00 < 0 \text{ then } V00 = 0$	(2)
---	-----

## 5.4. RGBR (RGB24) в YUY2

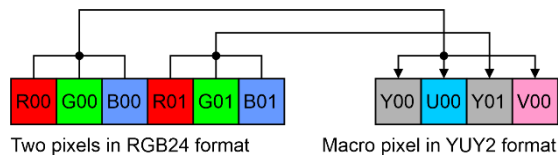


Рисунок 11 – Конвертация RGBR (RGB24) в YUY2.

$$\begin{aligned}
 Y00 &= 0.299 * R00 + 0.587 * G00 + 0.114 * B00 \\
 Y01 &= 0.299 * R01 + 0.587 * G01 + 0.114 * B01 \\
 U00 &= 0.492 * (B00 - Y00) + 128, \text{if } U00 > 255 \text{ then } U00 = 255, \text{if } U00 < 0 \text{ then } U00 = 0 \\
 V00 &= 0.877 * (R00 - Y00) + 128, \text{if } V00 > 255 \text{ then } V00 = 255, \text{if } V00 < 0 \text{ then } V00 = 0
 \end{aligned}
 \tag{3}$$

## 5.5. RGBR (RGB24) в YUV1

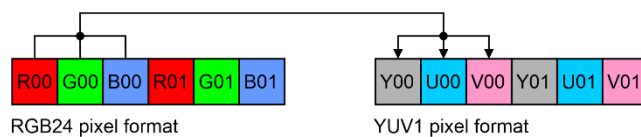


Рисунок 12 – Конвертация RGBR (RGB24) в YUV1.

$$\begin{aligned}
 Y00 &= 0.299 * R00 + 0.587 * G00 + 0.114 * B00 \\
 U00 &= 0.492 * (B00 - Y00) + 128, \text{if } U00 > 255 \text{ then } U00 = 255, \text{if } U00 < 0 \text{ then } U00 = 0 \\
 V00 &= 0.877 * (R00 - Y00) + 128, \text{if } V00 > 255 \text{ then } V00 = 255, \text{if } V00 < 0 \text{ then } V00 = 0
 \end{aligned}
 \tag{4}$$

## 5.6. RGBR (RGB24) в NV12

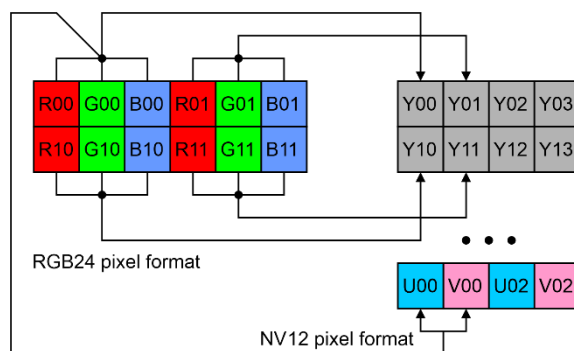


Рисунок 13 – Конвертация RGBR (RGB24) в NV12.

$$\begin{aligned}
 Y00 &= 0.299 * R00 + 0.587 * G00 + 0.114 * B00 \\
 Y01 &= 0.299 * R01 + 0.587 * G01 + 0.114 * B01 \\
 Y10 &= 0.299 * R10 + 0.587 * G10 + 0.114 * B10 \\
 Y11 &= 0.299 * R11 + 0.587 * G11 + 0.114 * B11 \\
 U00 &= 0.492 * (B00 - Y00) + 128, \text{if } U00 > 255 \text{ then } U00 = 255, \text{if } U00 < 0 \text{ then } U00 = 0 \\
 V00 &= 0.877 * (R00 - Y00) + 128, \text{if } V00 > 255 \text{ then } V00 = 255, \text{if } V00 < 0 \text{ then } V00 = 0
 \end{aligned}
 \tag{5}$$

## 5.7. BGRB (BGR24) в RGBR (RGB24)

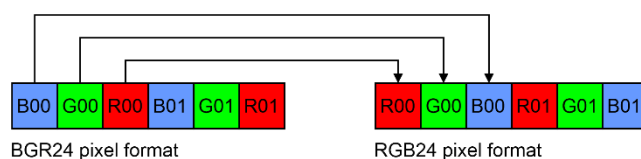


Рисунок 14 – Конвертация BGRB (BGR24) в RGBR (RGB24). Только замена байт местами.

## 5.8. BGRB (BGR24) в Y800

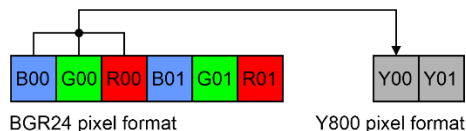


Рисунок 15 – Конвертация BGRB (BGR24) в Y800.

$$Y00 = 0.299 * R00 + 0.587 * G00 + 0.114 * B00$$

(6)

## 5.9. BGRB (BGR24) в UYVY

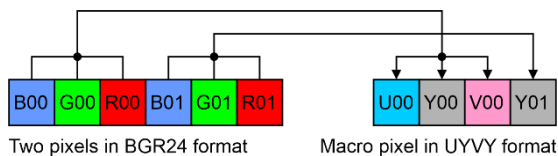


Рисунок 16 – Конвертация BGRB (BGR24) в UYVY.

$$\begin{aligned}
 Y00 &= 0.299 * R00 + 0.587 * G00 + 0.114 * B00 \\
 Y01 &= 0.299 * R01 + 0.587 * G01 + 0.114 * B01 \\
 U00 &= 0.492 * (B00 - Y00) + 128, \text{if } U00 > 255 \text{ then } U00 = 255, \text{if } U00 < 0 \text{ then } U00 = 0 \\
 V00 &= 0.877 * (R00 - Y00) + 128, \text{if } V00 > 255 \text{ then } V00 = 255, \text{if } V00 < 0 \text{ then } V00 = 0
 \end{aligned}$$

(7)

## 5.10. BGRB (BGR24) в YUY2

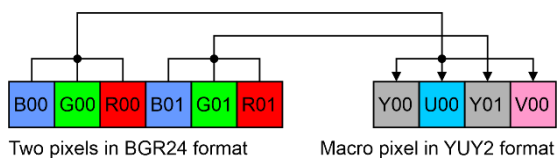


Рисунок 17 – Конвертация BGRB (BGR24) в YUY2.

$$\begin{aligned}
 Y00 &= 0.299 * R00 + 0.587 * G00 + 0.114 * B00 \\
 Y01 &= 0.299 * R01 + 0.587 * G01 + 0.114 * B01 \\
 U00 &= 0.492 * (B00 - Y00) + 128, \text{if } U00 > 255 \text{ then } U00 = 255, \text{if } U00 < 0 \text{ then } U00 = 0 \\
 V00 &= 0.877 * (R00 - Y00) + 128, \text{if } V00 > 255 \text{ then } V00 = 255, \text{if } V00 < 0 \text{ then } V00 = 0
 \end{aligned}$$

(8)

## 5.11. BGRB (BGR24) в YUV1

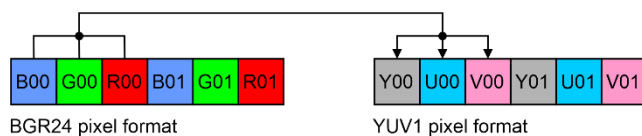


Рисунок 18 – Конвертация BGRB (BGR24) в YUV1.

$$\begin{aligned}
 Y00 &= 0.299 * R00 + 0.587 * G00 + 0.114 * B00 \\
 U00 &= 0.492 * (B00 - Y00) + 128, \text{if } U00 > 255 \text{ then } U00 = 255, \text{if } U00 < 0 \text{ then } U00 = 0 \\
 V00 &= 0.877 * (R00 - Y00) + 128, \text{if } V00 > 255 \text{ then } V00 = 255, \text{if } V00 < 0 \text{ then } V00 = 0
 \end{aligned}$$

(9)

## 5.12. BGRB (BGR24) в NV12

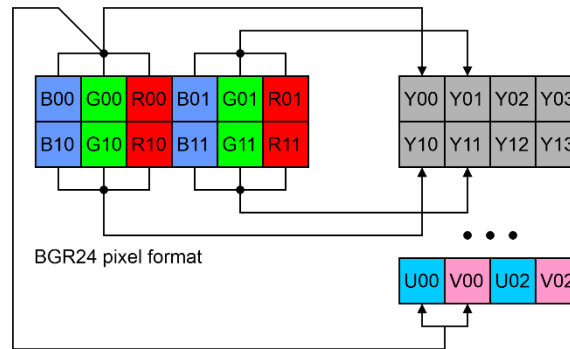


Рисунок 19 – Конвертация BGRB (BGR24) в NV12.

$$\begin{aligned}
 Y00 &= 0.299 * R00 + 0.587 * G00 + 0.114 * B00 \\
 Y01 &= 0.299 * R01 + 0.587 * G01 + 0.114 * B01 \\
 Y10 &= 0.299 * R10 + 0.587 * G10 + 0.114 * B10 \\
 Y11 &= 0.299 * R11 + 0.587 * G11 + 0.114 * B11 \\
 U00 &= 0.492 * (B00 - Y00) + 128, \text{if } U00 > 255 \text{ then } U00 = 255, \text{if } U00 < 0 \text{ then } U00 = 0 \\
 V00 &= 0.877 * (R00 - Y00) + 128, \text{if } V00 > 255 \text{ then } V00 = 255, \text{if } V00 < 0 \text{ then } V00 = 0
 \end{aligned}$$

(10)

## 5.13. Y800 в RGBR (RGB24)

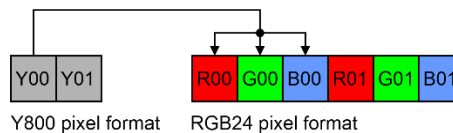


Рисунок 20 – Конвертация Y800 в RGBR (RGB24).

$$\begin{aligned}
 R00 &= Y00 \\
 G00 &= Y00 \\
 B00 &= Y00
 \end{aligned}$$

(11)

## 5.14. Y800 в BGRB (BGR24)

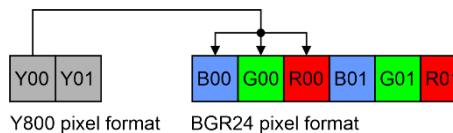


Рисунок 21 – Конвертация Y800 в BGRB (BGR24).

$$\begin{aligned}
 B00 &= Y00 \\
 G00 &= Y00 \\
 R00 &= Y00
 \end{aligned}$$

(12)

## 5.15. Y800 в UYVY

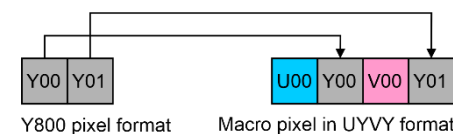


Рисунок 22 – Конвертация Y800 в UYVY.

$$Y00 = Y00$$

(13)



$Y01 = Y01$ $U00 = 0$ $V00 = 0$	
---------------------------------	--

5.16. Y800 в YUY2

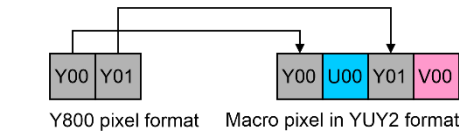


Рисунок 23 – Конвертация Y800 в YUY2.

$Y00 = Y00$ $Y01 = Y01$ $U00 = 0$ $V00 = 0$	(14)
---	------

5.17. Y800 в YUV1

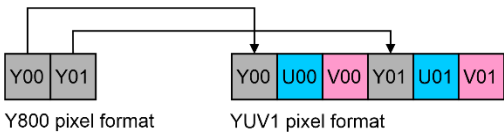


Рисунок 24 – Конвертация Y800 в YUV1.

$Y00 = Y00$ $Y01 = Y01$ $U00 = 0$ $V00 = 0$ $U01 = 0$ $V01 = 0$	(15)
---	------

5.18. Y800 в NV12

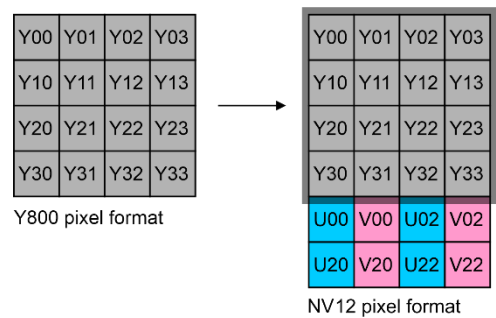


Рисунок 25 – Конвертация Y800 в NV12.

$Y \text{ data of NV12 format} = Y \text{ data of Y800 format}$ $U = 0$ $V = 0$	(16)
---	------

## 5.19. UYVY в RGBR (RGB24)

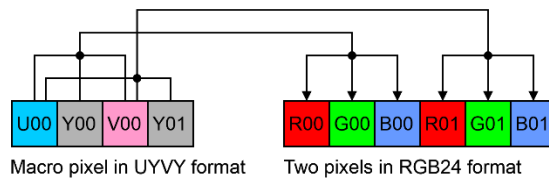


Рисунок 26 – Конвертация UYVY в RGBR (RGB24).

$$\begin{aligned}
 R00 &= Y00 + 1.140 * (V00 - 128), \text{if } R00 > 255 \text{ then } R00 = 255, \text{if } R00 < 0 \text{ then } R00 = 0 \\
 G00 &= Y00 - 0.395 * (U00 - 128) - 0.581 * (V00 - 128), \text{if } G00 > 255 \text{ then } G00 = 255, \\
 &\quad \text{if } G00 < 0 \text{ then } G00 = 0 \\
 B00 &= Y00 + 2.032 * (U00 - 128), \text{if } B00 > 255 \text{ then } B00 = 255, \text{if } B00 < 0 \text{ then } B00 = 0 \\
 R01 &= Y01 + 1.140 * (V00 - 128), \text{if } R01 > 255 \text{ then } R01 = 255, \text{if } R01 < 0 \text{ then } R01 = 0 \\
 G01 &= Y01 - 0.395 * (U00 - 128) - 0.581 * (V00 - 128), \text{if } G01 > 255 \text{ then } G01 = 255, \\
 &\quad \text{if } G01 < 0 \text{ then } G01 = 0 \\
 B01 &= Y01 + 2.032 * (U00 - 128), \text{if } B01 > 255 \text{ then } B01 = 255, \text{if } B01 < 0 \text{ then } B01 = 0
 \end{aligned}$$

(17)

## 5.20. UYVY в BGRB (BGR24)

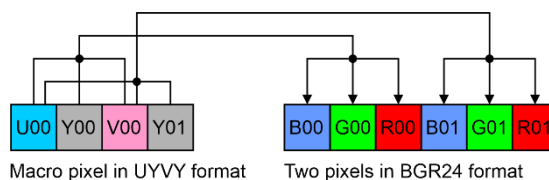


Рисунок 27 – Конвертация UYVY в BGRB (BGR24).

$$\begin{aligned}
 B00 &= Y00 + 2.032 * (U00 - 128), \text{if } B00 > 255 \text{ then } B00 = 255, \text{if } B00 < 0 \text{ then } B00 = 0 \\
 G00 &= Y00 - 0.395 * (U00 - 128) - 0.581 * (V00 - 128), \text{if } G00 > 255 \text{ then } G00 = 255, \\
 &\quad \text{if } G00 < 0 \text{ then } G00 = 0 \\
 R00 &= Y00 + 1.140 * (V00 - 128), \text{if } R00 > 255 \text{ then } R00 = 255, \text{if } R00 < 0 \text{ then } R00 = 0 \\
 B01 &= Y01 + 2.032 * (U00 - 128), \text{if } B01 > 255 \text{ then } B01 = 255, \text{if } B01 < 0 \text{ then } B01 = 0 \\
 G01 &= Y01 - 0.395 * (U00 - 128) - 0.581 * (V00 - 128), \text{if } G01 > 255 \text{ then } G01 = 255, \\
 &\quad \text{if } G01 < 0 \text{ then } G01 = 0 \\
 R01 &= Y01 + 1.140 * (V00 - 128), \text{if } R01 > 255 \text{ then } R01 = 255, \text{if } R01 < 0 \text{ then } R01 = 0
 \end{aligned}$$

(18)

## 5.21. UYVY в Y800

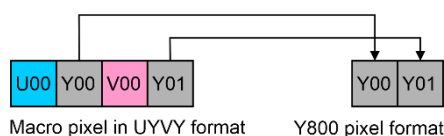


Рисунок 28 – Конвертация UYVY в Y800.

$$Y \text{ data of Y800 format} = Y \text{ data of UYVY format}$$

(19)

## 5.22. UYVY в YUY2

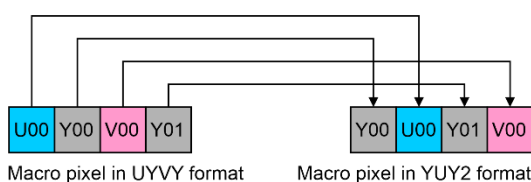


Рисунок 29 – Конвертация UYVY в YUY2. Только замена байт местами.

### 5.23. UYVY в YUV1

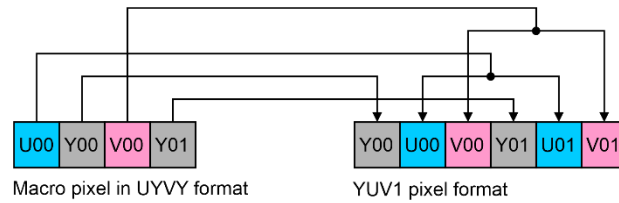


Рисунок 30 – Конвертация UYVY в YUV1.

$  \begin{aligned}  Y00 &= Y00 \\  U00 &= U00 \\  V00 &= V00 \\  Y01 &= Y01 \\  U01 &= U00 \\  V01 &= V00  \end{aligned}  $	(20)
---	------

### 5.24. UYVY в NV12

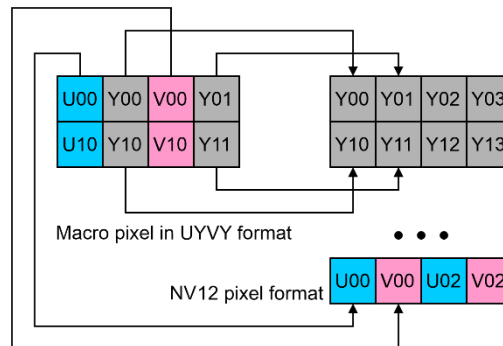


Рисунок 31 – Конвертация UYVY в NV12.

$  \begin{aligned}  Y00 &= Y00 \\  Y01 &= Y01 \\  Y10 &= Y10 \\  Y11 &= Y11 \\  U00 &= U00 \\  V00 &= V00  \end{aligned}  $	(21)
---	------

### 5.25. YUY2 в RGBR (RGB24)

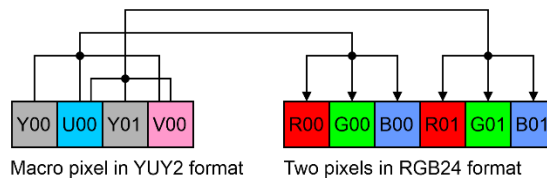


Рисунок 32 – Конвертация YUY2 в RGBR (RGB24).

$  \begin{aligned}  R00 &= Y00 + 1.140 * (V00 - 128), \text{if } R00 > 255 \text{ then } R00 = 255, \text{if } R00 < 0 \text{ then } R00 = 0 \\  G00 &= Y00 - 0.395 * (U00 - 128) - 0.581 * (V00 - 128), \text{if } G00 > 255 \text{ then } G00 = 255, \\  &\quad \text{if } G00 < 0 \text{ then } G00 = 0 \\  B00 &= Y00 + 2.032 * (U00 - 128), \text{if } B00 > 255 \text{ then } B00 = 255, \text{if } B00 < 0 \text{ then } B00 = 0 \\  R01 &= Y01 + 1.140 * (V00 - 128), \text{if } R01 > 255 \text{ then } R01 = 255, \text{if } R01 < 0 \text{ then } R01 = 0 \\  G01 &= Y01 - 0.395 * (U00 - 128) - 0.581 * (V00 - 128), \text{if } G01 > 255 \text{ then } G01 = 255, \\  &\quad \text{if } G01 < 0 \text{ then } G01 = 0 \\  B01 &= Y01 + 2.032 * (U00 - 128), \text{if } B01 > 255 \text{ then } B01 = 255, \text{if } B01 < 0 \text{ then } B01 = 0  \end{aligned}  $	(22)
---	------

## 5.26. YUY2 в BGRB (BGR24)

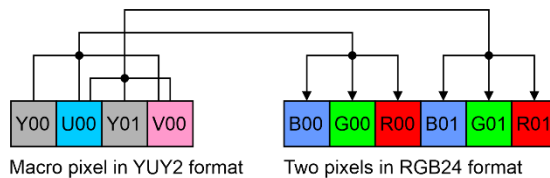


Рисунок 33 – Конвертация YUY2 в BGRB (BGR24).

$$\begin{aligned}
 B00 &= Y00 + 2.032 * (U00 - 128), \text{if } B00 > 255 \text{ then } B00 = 255, \text{if } B00 < 0 \text{ then } B00 = 0 \\
 G00 &= Y00 - 0.395 * (U00 - 128) - 0.581 * (V00 - 128), \text{if } G00 > 255 \text{ then } G00 = 255, \\
 &\quad \text{if } G00 < 0 \text{ then } G00 = 0 \\
 R00 &= Y00 + 1.140 * (V00 - 128), \text{if } R00 > 255 \text{ then } R00 = 255, \text{if } R00 < 0 \text{ then } R00 = 0 \\
 B01 &= Y01 + 2.032 * (U00 - 128), \text{if } B01 > 255 \text{ then } B01 = 255, \text{if } B01 < 0 \text{ then } B01 = 0 \\
 G01 &= Y01 - 0.395 * (U00 - 128) - 0.581 * (V00 - 128), \text{if } G01 > 255 \text{ then } G01 = 255, \\
 &\quad \text{if } G01 < 0 \text{ then } G01 = 0 \\
 R01 &= Y01 + 1.140 * (V00 - 128), \text{if } R01 > 255 \text{ then } R01 = 255, \text{if } R01 < 0 \text{ then } R01 = 0
 \end{aligned}$$

(23)

## 5.27. YUY2 в Y800

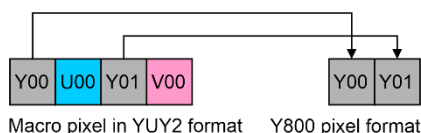


Рисунок 34 – Конвертация YUY2 в Y800.

$$Y \text{ data of Y800 format} = Y \text{ data of YUY2 format}$$

(24)

## 5.28. YUY2 в UYVY

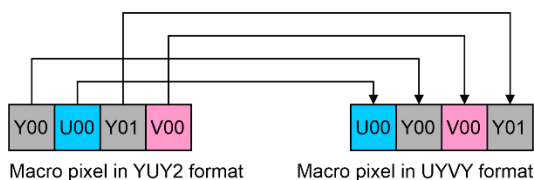


Рисунок 35 – Конвертация YUY2 в UYVY. Только замена байт местами.

## 5.29. YUY2 в YUV1

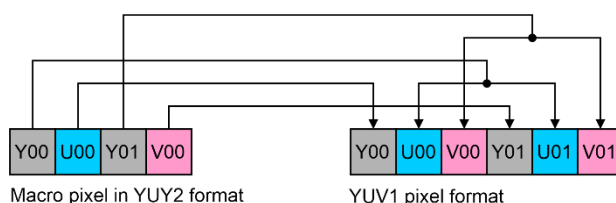


Рисунок 36 – Конвертация YUY2 в YUV1.

$$\begin{aligned}
 Y00 &= Y00 \\
 U00 &= U00 \\
 V00 &= V00 \\
 Y01 &= Y01 \\
 U01 &= U00 \\
 V01 &= V00
 \end{aligned}$$

(25)

### 5.30. YUY2 в NV12

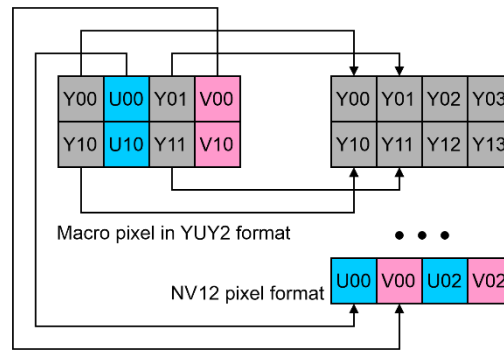


Рисунок 37 – Конвертация YUY2 в NV12.

$$\begin{aligned} Y00 &= Y00 \\ Y01 &= Y01 \\ Y10 &= Y10 \\ Y11 &= Y11 \\ U00 &= U00 \\ V00 &= V00 \end{aligned}$$

(26)

### 5.31. YUV1 в RGBR (RGB24)

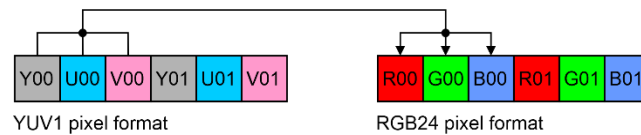


Рисунок 38 – Конвертация YUV1 в RGBR (RGB24).

$$\begin{aligned} R00 &= Y00 + 1.140 * (V00 - 128), \text{if } R00 > 255 \text{ then } R00 = 255, \text{if } R00 < 0 \text{ then } R00 = 0 \\ G00 &= Y00 - 0.395 * (U00 - 128) - 0.581 * (V00 - 128), \text{if } G00 > 255 \text{ then } G00 = 255, \\ &\quad \text{if } G00 < 0 \text{ then } G00 = 0 \\ B00 &= Y00 + 2.032 * (U00 - 128), \text{if } B00 > 255 \text{ then } B00 = 255, \text{if } B00 < 0 \text{ then } B00 = 0 \end{aligned}$$

(27)

### 5.32. YUV1 в BGRB (BGR24)

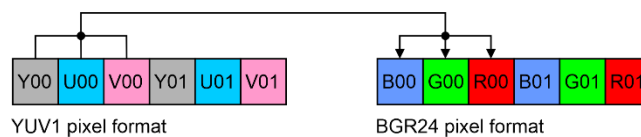


Рисунок 39 – Конвертация YUV1 в RGBR (RGB24).

$$\begin{aligned} B00 &= Y00 + 2.032 * (U00 - 128), \text{if } B00 > 255 \text{ then } B00 = 255, \text{if } B00 < 0 \text{ then } B00 = 0 \\ G00 &= Y00 - 0.395 * (U00 - 128) - 0.581 * (V00 - 128), \text{if } G00 > 255 \text{ then } G00 = 255, \\ &\quad \text{if } G00 < 0 \text{ then } G00 = 0 \\ R00 &= Y00 + 1.140 * (V00 - 128), \text{if } R00 > 255 \text{ then } R00 = 255, \text{if } R00 < 0 \text{ then } R00 = 0 \end{aligned}$$

(28)

### 5.33. YUV1 в Y800

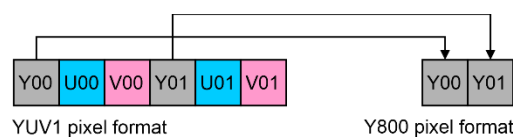
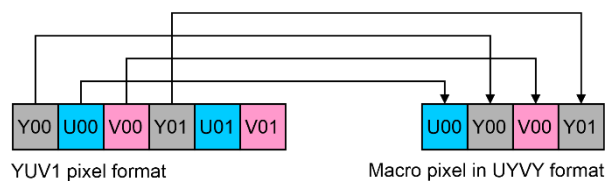


Рисунок 40 – Конвертация YUV1 в Y800.

*Y data of Y800 format = Y data of YUV1 format*

(29)

### 5.34. YUV1 в UYVY



**Рисунок 41** – Конвертация YUV1 в UYVY.

$$Y00 = Y00$$

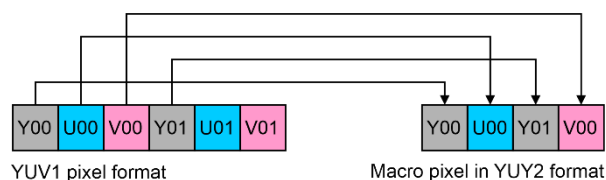
$$U00 = U00$$

$$V00 = V00$$

$$Y01 = Y01$$

(30)

### 5.35. YUV1 в YUY2



**Рисунок 42** – Конвертация YUV1 в YUY2.

$$Y00 = Y00$$

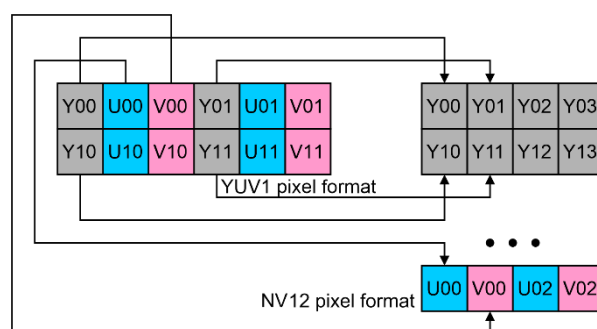
$$U00 = U00$$

$$V00 = V00$$

$$Y01 = Y01$$

(31)

### 5.36. YUV1 в NV12



**Рисунок 43** – Конвертация YUV1 в NV12.

$$Y00 = Y00$$

$$Y01 = Y01$$

$$Y10 = Y10$$

$$Y11 = Y11$$

$$U00 = U00$$

$$V00 = V00$$

(32)

### 5.37. NV12 в RGBR (RGB24)

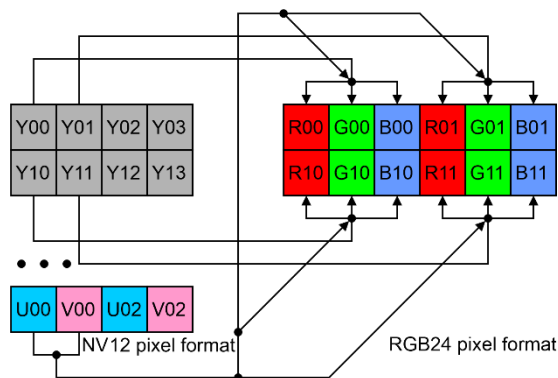


Рисунок 44 – Конвертация NV12 в RGBR (RGB24).

$$\begin{aligned}
 R00 &= Y00 + 1.140 * (V00 - 128), \text{if } R00 > 255 \text{ then } R00 = 255, \text{if } R00 < 0 \text{ then } R00 = 0 \\
 G00 &= Y00 - 0.395 * (U00 - 128) - 0.581 * (V00 - 128), \text{if } G00 > 255 \text{ then } G00 = 255, \\
 &\quad \text{if } G00 < 0 \text{ then } G00 = 0 \\
 B00 &= Y00 + 2.032 * (U00 - 128), \text{if } B00 > 255 \text{ then } B00 = 255, \text{if } B00 < 0 \text{ then } B00 = 0 \\
 R01 &= Y01 + 1.140 * (V00 - 128), \text{if } R01 > 255 \text{ then } R01 = 255, \text{if } R01 < 0 \text{ then } R01 = 0 \\
 G01 &= Y01 - 0.395 * (U00 - 128) - 0.581 * (V00 - 128), \text{if } G01 > 255 \text{ then } G01 = 255, \\
 &\quad \text{if } G01 < 0 \text{ then } G01 = 0 \\
 B01 &= Y01 + 2.032 * (U00 - 128), \text{if } B01 > 255 \text{ then } B01 = 255, \text{if } B01 < 0 \text{ then } B01 = 0 \\
 R10 &= Y10 + 1.140 * (V00 - 128), \text{if } R10 > 255 \text{ then } R10 = 255, \text{if } R10 < 0 \text{ then } R10 = 0 \\
 G10 &= Y10 - 0.395 * (U00 - 128) - 0.581 * (V00 - 128), \text{if } G10 > 255 \text{ then } G10 = 255, \\
 &\quad \text{if } G10 < 0 \text{ then } G10 = 0 \\
 B10 &= Y10 + 2.032 * (U00 - 128), \text{if } B10 > 255 \text{ then } B10 = 255, \text{if } B10 < 0 \text{ then } B10 = 0 \\
 R11 &= Y11 + 1.140 * (V00 - 128), \text{if } R11 > 255 \text{ then } R11 = 255, \text{if } R11 < 0 \text{ then } R11 = 0 \\
 G11 &= Y11 - 0.395 * (U00 - 128) - 0.581 * (V00 - 128), \text{if } G11 > 255 \text{ then } G11 = 255, \\
 &\quad \text{if } G11 < 0 \text{ then } G11 = 0 \\
 B11 &= Y11 + 2.032 * (U00 - 128), \text{if } B11 > 255 \text{ then } B11 = 255, \text{if } B11 < 0 \text{ then } B11 = 0
 \end{aligned}$$

(33)

### 5.38. NV12 в BGRB (BGR24)

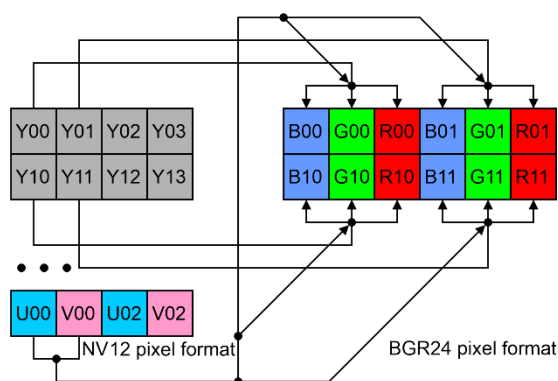


Рисунок 45 – Конвертация NV12 в BGRB (BGR24).

$$\begin{aligned}
 R00 &= Y00 + 1.140 * (V00 - 128), \text{if } R00 > 255 \text{ then } R00 = 255, \text{if } R00 < 0 \text{ then } R00 = 0 \\
 G00 &= Y00 - 0.395 * (U00 - 128) - 0.581 * (V00 - 128), \text{if } G00 > 255 \text{ then } G00 = 255, \\
 &\quad \text{if } G00 < 0 \text{ then } G00 = 0 \\
 B00 &= Y00 + 2.032 * (U00 - 128), \text{if } B00 > 255 \text{ then } B00 = 255, \text{if } B00 < 0 \text{ then } B00 = 0 \\
 R01 &= Y01 + 1.140 * (V00 - 128), \text{if } R01 > 255 \text{ then } R01 = 255, \text{if } R01 < 0 \text{ then } R01 = 0 \\
 G01 &= Y01 - 0.395 * (U00 - 128) - 0.581 * (V00 - 128), \text{if } G01 > 255 \text{ then } G01 = 255, \\
 &\quad \text{if } G01 < 0 \text{ then } G01 = 0 \\
 B01 &= Y01 + 2.032 * (U00 - 128), \text{if } B01 > 255 \text{ then } B01 = 255, \text{if } B01 < 0 \text{ then } B01 = 0 \\
 R10 &= Y10 + 1.140 * (V00 - 128), \text{if } R10 > 255 \text{ then } R10 = 255, \text{if } R10 < 0 \text{ then } R10 = 0
 \end{aligned}$$

(34)

$$G10 = Y10 - 0.395 * (U00 - 128) - 0.581 * (V00 - 128), \text{if } G10 > 255 \text{ then } G10 = 255, \\ \text{if } G10 < 0 \text{ then } G10 = 0$$

$$B10 = Y10 + 2.032 * (U00 - 128), \text{if } B10 > 255 \text{ then } B10 = 255, \text{if } B10 < 0 \text{ then } B10 = 0$$

$$R11 = Y11 + 1.140 * (V00 - 128), \text{if } R11 > 255 \text{ then } R11 = 255, \text{if } R11 < 0 \text{ then } R11 = 0$$

$$G11 = Y11 - 0.395 * (U00 - 128) - 0.581 * (V00 - 128), \text{if } G11 > 255 \text{ then } G11 = 255, \\ \text{if } G11 < 0 \text{ then } G11 = 0$$

$$B11 = Y11 + 2.032 * (U00 - 128), \text{if } B11 > 255 \text{ then } B11 = 255, \text{if } B11 < 0 \text{ then } B11 = 0$$

### 5.39. NV12 в Y800

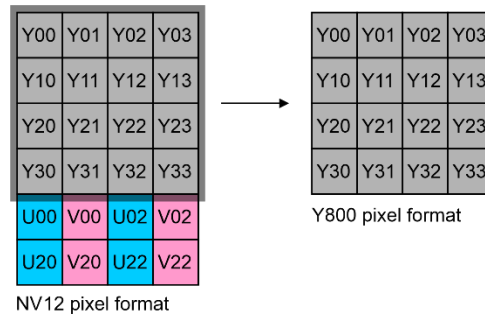


Рисунок 46 – Конвертация NV12 в Y800.

*Y data of Y800 format = Y data of NV12 format*

(35)

### 5.40. NV12 в UYVY

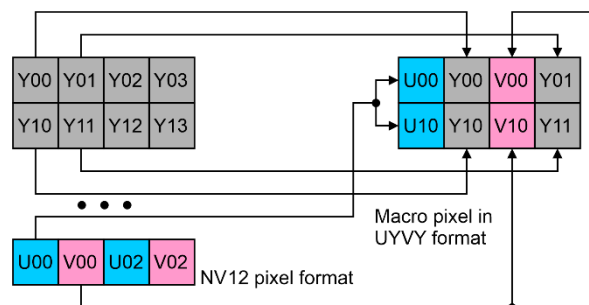


Рисунок 47 – Конвертация NV12 в UYVY.

$$Y00 = Y00$$

$$Y01 = Y01$$

$$Y10 = Y10$$

$$Y11 = Y11$$

$$U00 = U00$$

$$U10 = U00$$

$$V00 = V00$$

$$V10 = V00$$

(36)



## 5.41. NV12 в YUY2

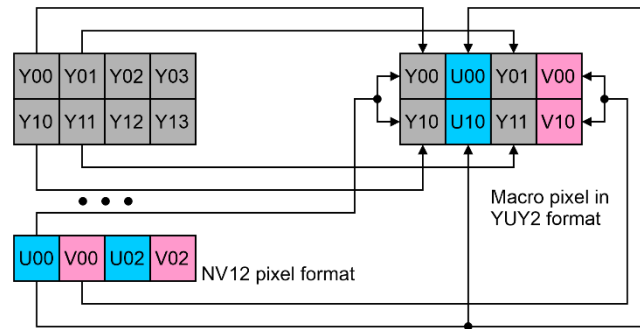


Рисунок 48 – Конвертация NV12 в YUY2.

$$\begin{aligned} Y00 &= Y00 \\ Y01 &= Y01 \\ Y10 &= Y10 \\ Y11 &= Y11 \\ U00 &= U00 \\ U10 &= U00 \\ V00 &= V00 \\ V10 &= V00 \end{aligned}$$

(37)

## 5.42. NV12 в YUV1

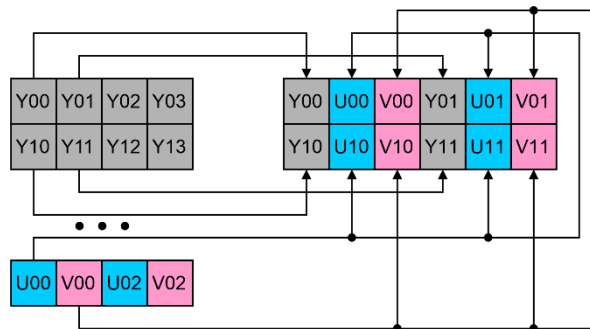


Рисунок 49 – Конвертация NV12 в YUV1.

$$\begin{aligned} Y00 &= Y00 \\ Y01 &= Y01 \\ Y10 &= Y10 \\ Y11 &= Y11 \\ U00 &= U00 \\ U01 &= U00 \\ U10 &= U00 \\ U11 &= U00 \\ V00 &= V00 \\ V01 &= V00 \\ V10 &= V00 \\ V11 &= V00 \end{aligned}$$

(38)

## 6. СТРУКТУРЫ ДАННЫХ

В качестве входных и выходных данных для программной библиотеки используется класс изображения **Frame**, объявленный в файле **VideoDataStructures.h**. Объявление класса **Frame** приведено ниже.

```

class Frame {
public:
    uint8_t* data;
    uint32_t width;
    uint32_t height;
    uint32_t size;
    uint32_t fourcc;
    uint32_t sourceID;
    uint32_t frameID;

    Frame() : data(nullptr), width(0), height(0), size(0), fourcc(0), sourceID(0), frameID(0) { };

    Frame(const Frame& src) : data(nullptr), width(0), height(0), size(0), fourcc(0), sourceID(0),
frameID(0);

    Frame(uint32_t width, uint32_t height, uint32_t fourcc) : data(nullptr), width(0), height(0),
size(0), fourcc(0), sourceID(0), frameID(0);

    Frame& operator= (const Frame& src);

    ~Frame();

    void Release();

};

```

Таблица 4 – Описание полей класса **Frame**.

Поле класса	Описание
data	Указатель на массив данных изображения.
width	Ширина изображения в пикселах.
height	Высота изображения в пикселах.
size	Размер данных изображения в байтах. Соответствует размеру буфера данных изображения.
fourcc	<p>FOURCC код формата пикселей изображения. В программной библиотеке реализованы следующие форматы пикселей: <b>RGBA (RGB24)</b>, <b>BGRB (BGR24)</b>, <b>Y800</b>, <b>UYVY</b>, <b>YUY2</b>, <b>YUV1</b> и <b>NV12</b>. Получить числовое значение кода можно с помощью макроса <b>MAKE_FOURCC_CODE</b>, объявленного в файле <b>VideoDataStructures.h</b>. Также числовые значения разрешенных FOURCC кодов определены в перечислении <b>ValidFourccCodes</b>, объявленного в файле <b>VideoDataStructures.h</b>:</p> <pre> enum class ValidFourccCodes {     RGBA = MAKE_FOURCC_CODE('R', 'G', 'B', 'A'),     BGRB = MAKE_FOURCC_CODE('B', 'G', 'R', 'B'),     UYVY = MAKE_FOURCC_CODE('U', 'Y', 'V', 'Y'),     Y800 = MAKE_FOURCC_CODE('Y', '8', '0', '0'),     YUY2 = MAKE_FOURCC_CODE('Y', 'U', 'Y', '2'),     YUV1 = MAKE_FOURCC_CODE('Y', 'U', 'V', '1'),     NV12 = MAKE_FOURCC_CODE('N', 'V', '1', '2'), </pre>

	<pre> JPEG = MAKE_FOURCC_CODE('J', 'P', 'E', 'G'), JPG2 = MAKE_FOURCC_CODE('J', 'P', 'G', '2'), H264 = MAKE_FOURCC_CODE('H', '2', '6', '4'), H265 = MAKE_FOURCC_CODE('H', '2', '6', '5') }; </pre>
sourceID	Идентификатор источника изображения или видео. Определяется пользователем.
frameID	Идентификатор изображения или кадра видео. Может использоваться для нумерации кадров и определяется пользователем.

Таблица 5 – Описание методов класса **Frame**.

Метод класса	Описание
Frame()	Конструктор класса по умолчанию.
~Frame()	Деструктор класса.
Frame(const Frame& src)	Конструктор копирования.
Frame(uint32_t width, uint32_t height, uint32_t fourcc)	Конструктор с параметрами: <b>width</b> – ширина изображения в пикселах; <b>height</b> – высота изображения в пикселах; <b>fourcc</b> – FOURCC код формата пикселей. Конструктор с параметрами выделяет память для данных изображения и заполняет ее нулевыми значениями.
Frame& operator= (const Frame& src)	Оператор копирования.
void Release()	Метод освобождения памяти. Освобождает память, выделенную для массива данных изображения, и инициализирует все поля значениями по умолчанию.

## 7. ОПИСАНИЕ КЛАССА PixelFormatConverter

### 7.1. Объявление класса PixelFormatConverter

Класс **PixelFormatConverter** предназначен для конвертации форматов пикселей изображений. Класс объявлен в файле **PixelFormatConvtrter.h**. Методы класса не выполняют каких-либо фоновых задач. Расчеты начинаются с вызовом метода **Convert(...)** класса и заканчиваются возвращением управления вызывающему потоку. В качестве входных и выходных данных используются экземпляры класса **Frame**. Ниже приведено объявление класса.

```

class PixelFormatConverter {
public:
    PixelFormatConverter();
    ~PixelFormatConverter();

    bool Convert(Frame& src, Frame& dst);

    void GetVersion(uint32_t& major, uint32_t& minor);

    bool isFourccCodeValid(uint32_t fourcc);

    std::vector<uint32_t> GetSupportedFourccCodes();
};

```

## 7.2. Метод Convert(...)

Метод Convert(...) предназначен для конвертации формата пикселей изображений. Метод конвертирует исходный формат пикселей, указанных во входном изображении (поле fourcc класса Frame) в формат, указанный в объекте класса выходного изображения (поле fourcc класса Frame). Объявление метода:

```
bool Convert(Frame& src, Frame& dst);
```

*Параметры:*

src	Ссылка на объект класса Frame исходного изображения. <b>Внимание:</b> Высота исходного изображения должна быть кратна 2.
dst	Ссылка на объект класса Frame результирующего изображения.

*Возвращаемое значение:*

Метод возвращает TRUE в случае успешной конвертации или возвращает FALSE в следующих случаях:

1. если высота исходного изображения не кратна 2;
2. если FOURCC код исходного изображения (поле fourcc класса Frame) не соответствует списку разрешенных (перечисление ValidFourccCodes в файле VideoDataStructures.h);
3. если FOURCC код результирующего изображения (поле fourcc объекта класса dst) не соответствует списку разрешенных (перечисление ValidFourccCodes в файле VideoDataStructures.h);
4. если исходное изображение src не проинициализировано: размеры изображения равны нулю, нет выделенного массива данных изображения или размер массива данных изображения не соответствует указанному формату пикселей кадров.

## 7.3. Метод GetVersion(...)

Метод GetVersion(...) предназначен для получения числового значения текущей версии программной библиотеки PixelFormatConverter. Объявление метода:

```
void GetVersion(uint32_t& major, uint32_t& minor);
```

*Параметры:*

major	Ссылка на возвращаемое мажорное значение версии библиотеки.
minor	Ссылка на возвращаемое минорное значение версии библиотеки.

## 7.4. Метод isFourccCodeValid(...)

Метод isFourccCodeValid(...) предназначен для проверки соответствия FOURCC кода разрешенному перечню, определенному в перечислении ValidSourccCodes (файл VideoDataStructures.h). Объявление метода:

```
bool isFourccCodeValid(uint32_t fourcc);
```

*Параметры:*

fourcc	Код FOURCC для проверки.
--------	--------------------------

*Возвращаемое значение:*

Метод возвращает TRUE, если переданный в параметрах FOURCC код разрешен. В противном случае метод возвращает FALSE.

## 7.5. Метод GetSupportedFourccCodes()

Метод GetSupportedFourccCodes() предназначен для получения списка разрешенных FOURCC кодов. Объявление метода:

```
std::vector<uint32_t> GetSupportedFourccCodes();
```

*Возвращаемое значение:*

Метод возвращает вектор разрешенных FOURCC кодов.

## 8. ПРИМЕР ИСПОЛЬЗОВАНИЯ

Ниже приведен пример конвертации изображения с форматом пикселей YUV1 в BGRB (BGR24). В примере производится сравнение правильности конвертации с конвертацией, реализованной в открытой библиотеке OpenCV.

```
const uint32_t width = 1280;
const uint32_t height = 1024;

// Create random OpenCV YUV image
cv::Mat yuvImage = cv::Mat(cv::Size(width, height), CV_8UC3);
for (size_t i = 0; i < (size_t)width * (size_t)height * 3; ++i)
    yuvImage.data[i] = (uint8_t)(rand() % 255);

// Create YUV frame
zs::Frame yuvFrame = zs::Frame(width, height, MAKE_FOURCC_CODE('Y', 'U', 'V', '1'));
memcpy(yuvFrame.data, yuvImage.data, yuvFrame.size);

// Convert image to RGB with OpenCV
cv::Mat bgrImage;
cv::cvtColor(yuvImage, bgrImage, cv::COLOR_YUV2BGR);

// Convert frame to RGB
zs::Frame bgrFrame;
bgrFrame.fourcc = MAKE_FOURCC_CODE('B', 'G', 'R', 'B');
zs::PixelFormatConverter converter;
if (!converter.Convert(yuvFrame, bgrFrame)) {
    Assert::Fail(L"Convert function returned FALSE");
    return;
}

// Compare data
uint8_t val0, val1;
for (size_t i = 0; i < (size_t)yuvFrame.size; ++i) {
    val0 = bgrFrame.data[i];
    val1 = bgrImage.data[i];
    if (abs((int)val0 - (int)val1) > 1) {
        Assert::Fail(L"Data not equal");
        return;
    }
}
```