



## **PixelFormatConverter lib**

**C++ software library for conversion images  
in various pixel formats**

Software version: **1.0**

Release date: **03.01.2020**

Document version: **1.0**

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

# CONTENTS

1. DOCUMENT VERSIONS .....	4
2. SOFTWARE VERSIONS .....	4
3. OVERVIEW.....	4
4. SUPPORTED PIXEL FORMATS .....	4
5. CONVERSION BETWEEN FORMATS .....	5
5.1. RGBR (RGB24) to BGRB (BGR24) .....	5
5.2. RGBR (RGB24) to Y800.....	5
5.3. RGBR (RGB24) to UYVY .....	5
5.4. RGBR (RGB24) to YUY2.....	6
5.5. RGBR (RGB24) to YUV1 .....	6
5.6. RGBR (RGB24) to NV12 .....	6
5.7. BGRB (BGR24) to RGBR (RGB24) .....	6
5.8. BGRB (BGR24) to Y800 .....	7
5.9. BGRB (BGR24) to UYVY.....	7
5.10. BGRB (BGR24) to YUY2 .....	7
5.11. BGRB (BGR24) to YUV1 .....	7
5.12. BGRB (BGR24) to NV12 .....	8
5.13. Y800 to RGBR (RGB24).....	8
5.14. Y800 to BGRB (BGR24) .....	8
5.15. Y800 to UYVY .....	8
5.16. Y800 to YUY2.....	9
5.17. Y800 to YUV1.....	9
5.18. Y800 to NV12 .....	9
5.19. UYVY to RGBR (RGB24) .....	10
5.20. UYVY to BGRB (BGR24).....	10
5.21. UYVY to Y800 .....	10
5.22. UYVY to YUY2 .....	10
5.23. UYVY to YUV1 .....	11
5.24. UYVY to NV12.....	11
5.25. YUY2 to RGBR (RGB24).....	11
5.26. YUY2 to BGRB (BGR24) .....	12
5.27. YUY2 to Y800.....	12
5.28. YUY2 to UYVY .....	12
5.29. YUY2 to YUV1.....	12
5.30. YUY2 to NV12 .....	13
5.31. YUV1 to RGBR (RGB24).....	13
5.32. YUV1 to BGRB (BGR24) .....	13
5.33. YUV1 to Y800.....	13
5.34. YUV1 to UYVY .....	14
5.35. YUV1 to YUY2.....	14
5.36. YUV1 to NV12 .....	14

5.37. NV12 to RGBR (RGB24) .....	15
5.38. NV12 to BGRB (BGR24) .....	15
5.39. NV12 to Y800 .....	16
5.40. NV12 to UYVY .....	16
5.41. NV12 to YUY2 .....	17
5.42. NV12 to YUV1 .....	17
6. DATA STRUCTURES .....	17
7. DESCRIPTION OF PixelFormatConverter CLASS .....	19
7.1. PixelFormatConverter class declaration .....	19
7.2. Convert(...) method .....	19
7.3. GetVersion(...) method .....	20
7.4. isFourccCodeValid(...) method .....	20
7.5. GetSupportedFourccCodes() method .....	20
8. EXAMPLE .....	21

# 1. DOCUMENT VERSIONS

Table 1 – Document versions.

Version	Release date	What is new
1.0	03.01.2020	First version of document.

# 2. SOFTWARE VERSIONS

Table 2 – Software versions.

Version	Release date	What is new
1.0	30.12.2019	First version of the software library. Implemented conversion between pixel formats: RGBR (RGB24), BGRB (BGR24), UYVY, Y800, YUY2, YUV1 and NV12.

# 3. OVERVIEW


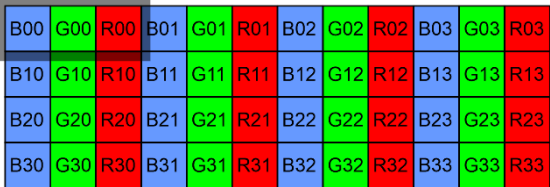


**PixelFormatConverter lib** is C++ software library, designed to convert images to various pixel formats (library). The library has a simple interface. The library is distributed as source codes and is compatible with any operating systems that support the C++ language compiler (C++11 standard). The library includes the following source code files:

- **VideoDataStructures.h** is header file describing data structures for images and video frames;
- **PixelFormatConverter.h** is header file containing a description of the only PixelFormatConverter program class;
- **PixelFormatConverter.cpp** – source code file containing the implementation of the methods of the program class PixelFormatConverter.

# 4. SUPPORTED PIXEL FORMATS

The library supports the following pixel formats: RGBR (RGB24), BGRB (BGR24), UYVY, Y800 (градации серого), YUY2, YUV1 and NV12. The numerical values of the pixel formats (the value of the FOURCC code) are determined by the **ValidFourccCodes** enum declared in the VideoDataStructures.h file. The library supports conversion between the specified formats. Table 3 shows illustrations of the location of pixel bytes in various formats for a 4x4 pixel image.

Table 3 – Illustrations of 4x4 pixel image data bytes in various formats.

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

top left pixel



Figure 5 – Y800 pixel format.

top left pixel



Figure 6 – YUV1 pixel format.

top left macro pixel



Figure 7 – NV12 pixel format.

## 5. CONVERSION BETWEEN FORMATS

### 5.1. RGBR (RGB24) to BGRB (BGR24)

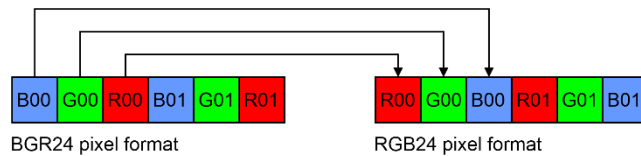


Figure 8 – Conversion RGBR (RGB24) to BGRB (BGR24). Only bytes swapping.

### 5.2. RGBR (RGB24) to Y800

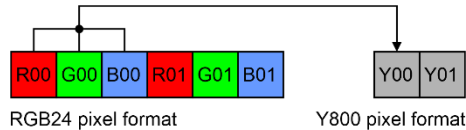


Figure 9 – Conversion RGBR (RGB24) to Y800.

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

(1)

### 5.3. RGBR (RGB24) to UYVY

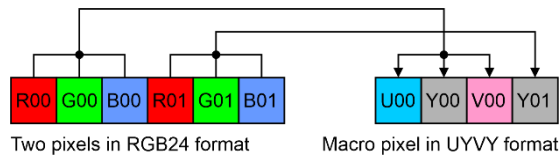


Figure 10 – Conversion RGBR (RGB24) to UYVY.

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

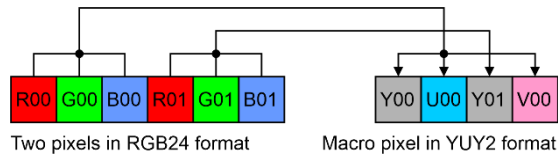
$$Y01 = 0.299 * R01 + 0.587 * G01 + 0.114 * B01$$

$$U00 = 0.492 * (B00 - Y00) + 128, \text{ if } U00 > 255 \text{ then } U00 = 255, \text{ if } U00 < 0 \text{ then } U00 = 0$$

$$V00 = 0.877 * (R00 - Y00) + 128, \text{ if } V00 > 255 \text{ then } V00 = 255, \text{ if } V00 < 0 \text{ then } V00 = 0$$

(2)

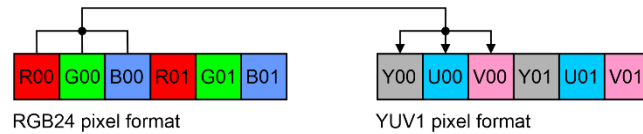
## 5.4. RGBR (RGB24) to YUY2



**Figure 11** – Conversion RGBR (RGB24) to YUY2.

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

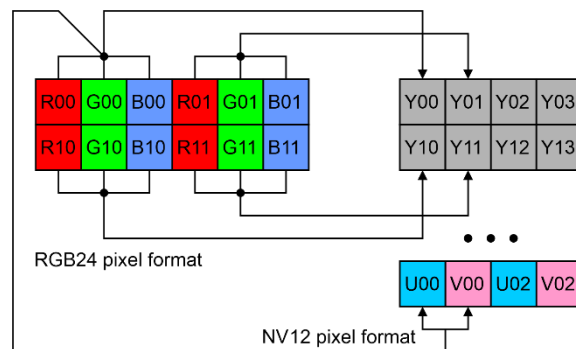
## 5.5. RGBR (RGB24) to YUV1



**Figure 12** – Conversion RGBR (RGB24) to YUV1.

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

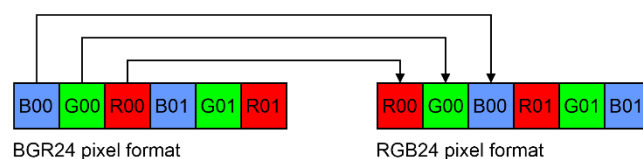
## 5.6. RGBR (RGB24) to NV12



**Figure 13** – Conversion RGBR (RGB24) to NV12.

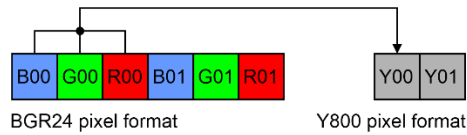
$$\begin{aligned}
 Y00 &= 0.299 * R00 + 0.587 * G00 + 0.114 * B00 \\
 Y01 &= 0.299 * R01 + 0.587 * G01 + 0.114 * B01 \\
 Y10 &= 0.299 * R10 + 0.587 * G10 + 0.114 * B10 \\
 Y11 &= 0.299 * R11 + 0.587 * G11 + 0.114 * B11 \\
 U00 &= 0.492 * (B00 - Y00) + 128, \text{if } U00 > 255 \text{ then } U00 = 255,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

## 5.7. BGRB (BGR24) to RGBR (RGB24)



**Figure 14** – Conversion BGRB (BGR24) to RGBR (RGB24). Only bytes swapping.

## 5.8. BGRB (BGR24) to Y800

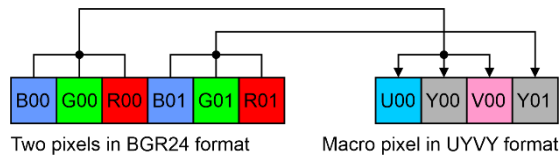


**Figure 15** – Conversion BGRB (BGR24) to Y800.

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

(6)

## 5.9. BGRB (BGR24) to UYVY

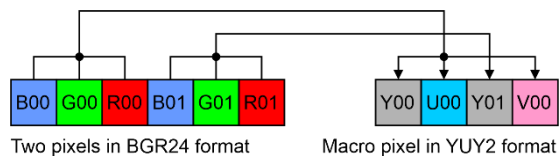


**Figure 16** – Conversion BGRB (BGR24) to UYVY.

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

(7)

## 5.10. BGRB (BGR24) to YUY2

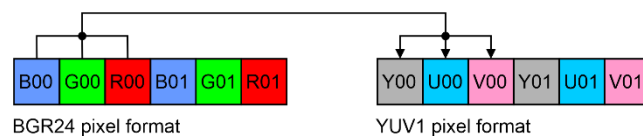


**Figure 17** – Conversion BGRB (BGR24) to YUY2.

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

(8)

## 5.11. BGRB (BGR24) to YUV1

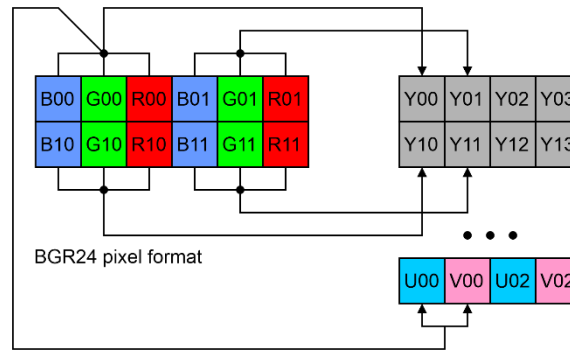


**Figure 18** – Conversion BGRB (BGR24) to YUV1.

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

(9)

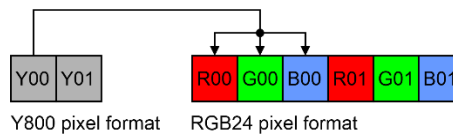
## 5.12. BGRB (BGR24) to NV12



**Figure 19** – Conversion BGRB (BGR24) to NV12.

$$\begin{aligned}
 Y00 &= 0.299 * R00 + 0.587 * G00 + 0.114 * B00 \\
 Y01 &= 0.299 * R01 + 0.587 * G01 + 0.114 * B01 \\
 Y10 &= 0.299 * R10 + 0.587 * G10 + 0.114 * B10 \\
 Y11 &= 0.299 * R11 + 0.587 * G11 + 0.114 * B11 \\
 U00 &= 0.492 * (B00 - Y00) + 128, \text{if } U00 > 255 \text{ then } U00 = 255, \text{if } U00 < 0 \text{ then } U00 = 0 \\
 V00 &= 0.877 * (R00 - Y00) + 128, \text{if } V00 > 255 \text{ then } V00 = 255, \text{if } V00 < 0 \text{ then } V00 = 0
 \end{aligned}
 \tag{10}$$

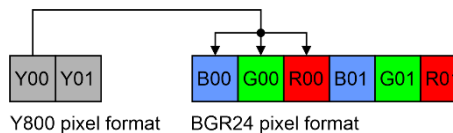
## 5.13. Y800 to RGBR (RGB24)



**Figure 20** – Conversion Y800 to RGBR (RGB24).

$$\begin{aligned}
 R00 &= Y00 \\
 G00 &= Y00 \\
 B00 &= Y00
 \end{aligned}
 \tag{11}$$

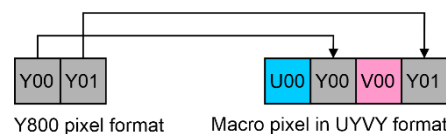
## 5.14. Y800 to BGRB (BGR24)



**Figure 21** – Conversion Y800 to BGRB (BGR24).

$$\begin{aligned}
 B00 &= Y00 \\
 G00 &= Y00 \\
 R00 &= Y00
 \end{aligned}
 \tag{12}$$

## 5.15. Y800 to UYVY



**Figure 22** – Conversion Y800 to UYVY.

$$Y00 = Y00
 \tag{13}$$



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

5.16. Y800 to YUY2

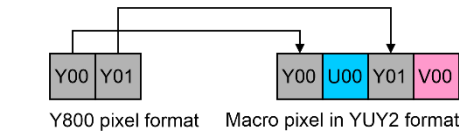


Figure 23 – Conversion Y800 to YUY2.

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

5.17. Y800 to YUV1

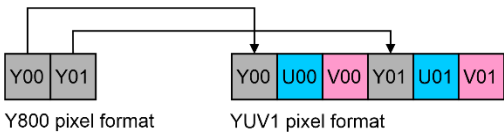


Figure 24 – Conversion Y800 to YUV1.

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

5.18. Y800 to NV12

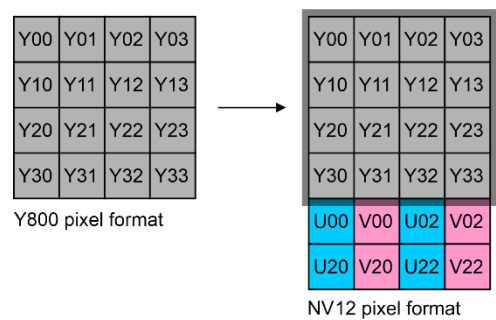
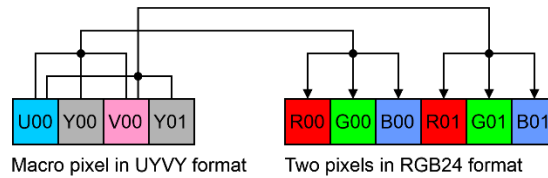


Figure 25 – Conversion Y800 to NV12.

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

### 5.19. UYVY to RGBR (RGB24)

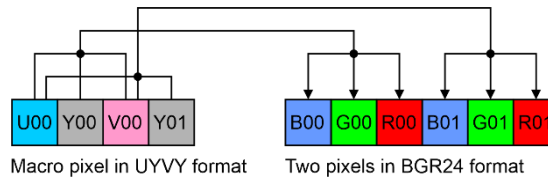


**Figure 26** – Conversion UYVY to RGBR (RGB24).

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

(17)

### 5.20. UYVY to BGRB (BGR24)

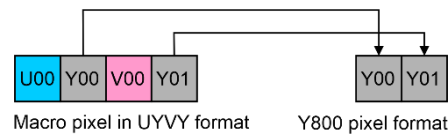


**Figure 27** – Conversion UYVY to BGRB (BGR24).

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

(18)

### 5.21. UYVY to Y800

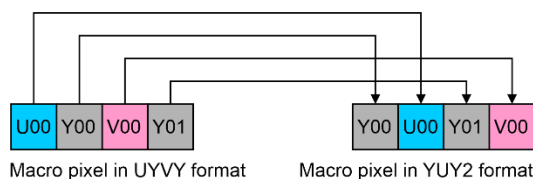


**Figure 28** – Conversion UYVY to Y800.

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

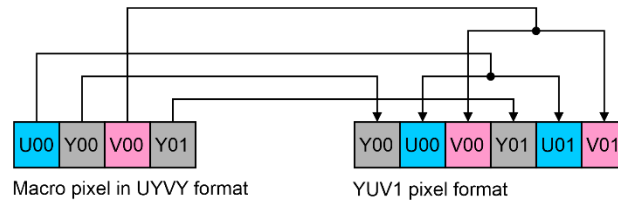
(19)

### 5.22. UYVY to YUY2



**Figure 29** – Conversion UYVY to YUY2. Only bytes swapping.

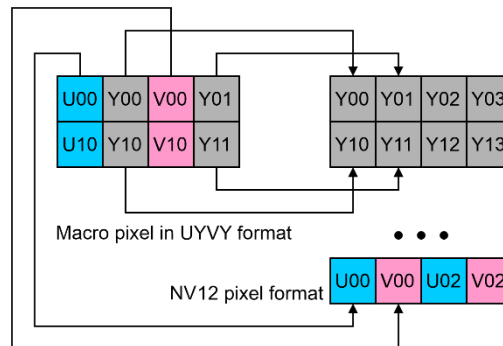
### 5.23. UYVY to YUV1



**Figure 30** – Conversion UYVY to YUV1.

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

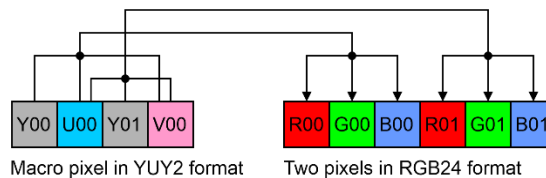
### 5.24. UYVY to NV12



**Figure 31** – Conversion UYVY to NV12.

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

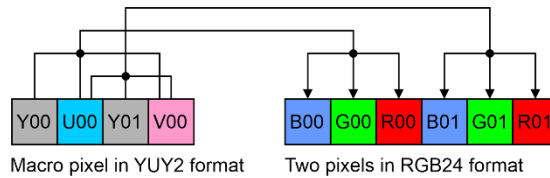
### 5.25. YUY2 to RGBR (RGB24)



**Figure 32** – Conversion YUY2 to 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 to BGRB (BGR24)

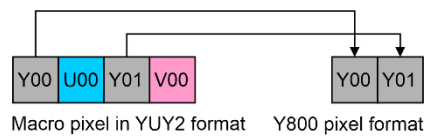


**Figure 33** – Conversion YUY2 to BGRB (BGR24).

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

(23)

## 5.27. YUY2 to Y800

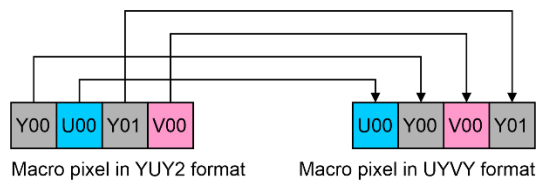


**Figure 34** – Conversion YUY2 to Y800.

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

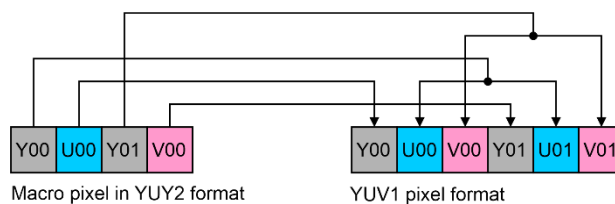
(24)

## 5.28. YUY2 to UYVY



**Figure 35** – Conversion YUY2 to UYVY. Only bytes swapping.

## 5.29. YUY2 to YUV1

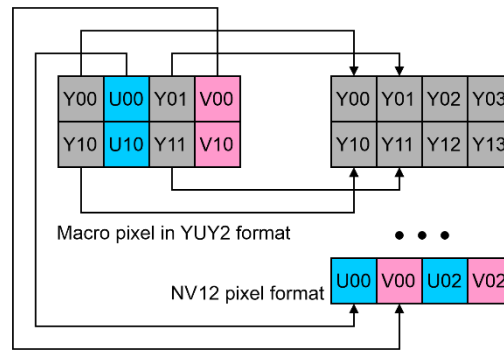


**Figure 36** – Conversion YUY2 to YUV1.

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

(25)

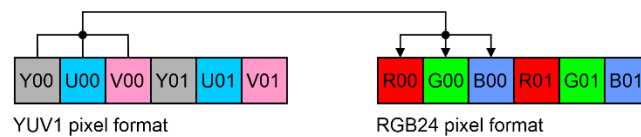
### 5.30. YUY2 to NV12



**Figure 37** – Conversion YUY2 to NV12.

$  \begin{aligned}  Y00 &= Y00 \\  Y01 &= Y01 \\  Y10 &= Y10 \\  Y11 &= Y11 \\  U00 &= U00 \\  V00 &= V00  \end{aligned}  $	(26)
---	------

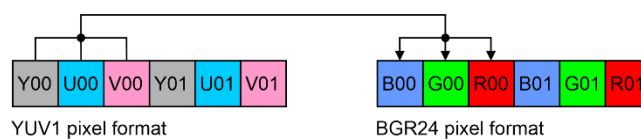
### 5.31. YUV1 to RGRB (RGB24)



**Figure 38** – Conversion YUV1 to RGRB (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)
---	------

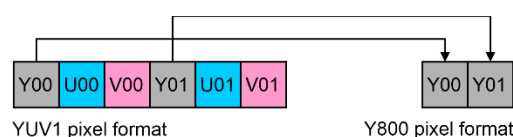
### 5.32. YUV1 to BGRB (BGR24)



**Figure 39** – Conversion YUV1 to RGRB (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 to Y800

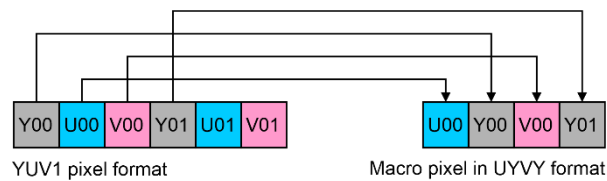


**Figure 40** – Conversion YUV1 to Y800.

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

(29)

### 5.34. YUV1 to UYVY



**Figure 41** – Conversion YUV1 to UYVY.

$$Y00 = Y00$$

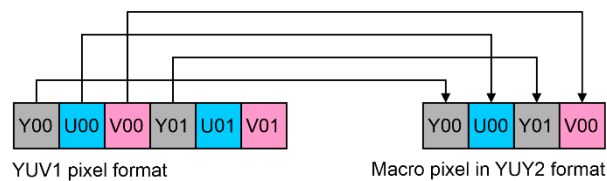
$$U00 = U00$$

$$V00 = V00$$

$$Y01 = Y01$$

(30)

### 5.35. YUV1 to YUY2



**Figure 42** – Conversion YUV1 to YUY2.

$$Y00 = Y00$$

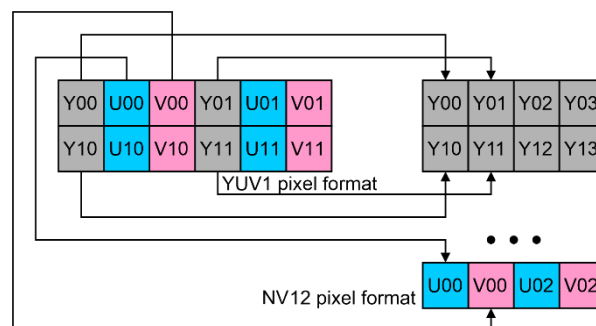
$$U00 = U00$$

$$V00 = V00$$

$$Y01 = Y01$$

(31)

### 5.36. YUV1 to NV12



**Рисунок 43** – Conversion YUV1 to NV12.

$$Y00 = Y00$$

$$Y01 = Y01$$

$$Y10 = Y10$$

$$Y11 = Y11$$

$$U00 = U00$$

$$V00 = V00$$

(32)

### 5.37. NV12 to RGBR (RGB24)

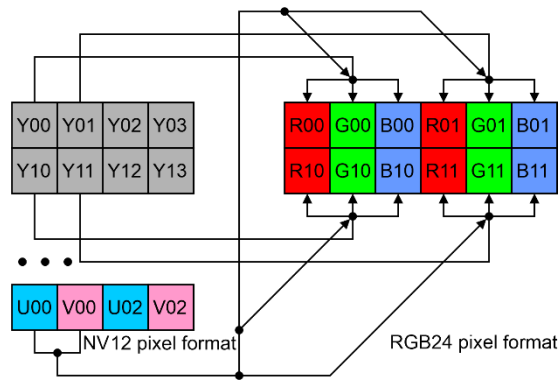


Figure 44 – Conversion NV12 to RGBR (RGB24).

$$\begin{aligned}
 R00 &= Y00 + 1.140 * (V00 - 128), \text{if } R00 > 255 \text{ then } R00 = 255, \text{if } R00 < 0 \text{ then } R00 = 0 \\
 G00 &= Y00 - 0.395 * (U00 - 128) - 0.581 * (V00 - 128), \text{if } G00 > 255 \text{ then } G00 = 255, \\
 &\quad \text{if } G00 < 0 \text{ then } G00 = 0 \\
 B00 &= Y00 + 2.032 * (U00 - 128), \text{if } B00 > 255 \text{ then } B00 = 255, \text{if } B00 < 0 \text{ then } B00 = 0 \\
 R01 &= Y01 + 1.140 * (V00 - 128), \text{if } R01 > 255 \text{ then } R01 = 255, \text{if } R01 < 0 \text{ then } R01 = 0 \\
 G01 &= Y01 - 0.395 * (U00 - 128) - 0.581 * (V00 - 128), \text{if } G01 > 255 \text{ then } G01 = 255, \\
 &\quad \text{if } G01 < 0 \text{ then } G01 = 0 \\
 B01 &= Y01 + 2.032 * (U00 - 128), \text{if } B01 > 255 \text{ then } B01 = 255, \text{if } B01 < 0 \text{ then } B01 = 0 \\
 R10 &= Y10 + 1.140 * (V00 - 128), \text{if } R10 > 255 \text{ then } R10 = 255, \text{if } R10 < 0 \text{ then } R10 = 0 \\
 G10 &= Y10 - 0.395 * (U00 - 128) - 0.581 * (V00 - 128), \text{if } G10 > 255 \text{ then } G10 = 255, \\
 &\quad \text{if } G10 < 0 \text{ then } G10 = 0 \\
 B10 &= Y10 + 2.032 * (U00 - 128), \text{if } B10 > 255 \text{ then } B10 = 255, \text{if } B10 < 0 \text{ then } B10 = 0 \\
 R11 &= Y11 + 1.140 * (V00 - 128), \text{if } R11 > 255 \text{ then } R11 = 255, \text{if } R11 < 0 \text{ then } R11 = 0 \\
 G11 &= Y11 - 0.395 * (U00 - 128) - 0.581 * (V00 - 128), \text{if } G11 > 255 \text{ then } G11 = 255, \\
 &\quad \text{if } G11 < 0 \text{ then } G11 = 0 \\
 B11 &= Y11 + 2.032 * (U00 - 128), \text{if } B11 > 255 \text{ then } B11 = 255, \text{if } B11 < 0 \text{ then } B11 = 0
 \end{aligned}
 \tag{33}$$

### 5.38. NV12 to BGRB (BGR24)

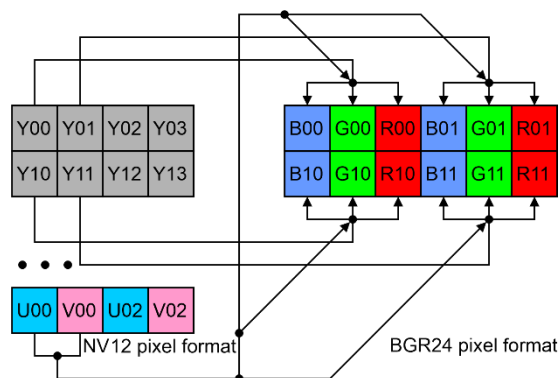


Figure 45 – Conversion NV12 to BGRB (BGR24).

$$\begin{aligned}
 R00 &= Y00 + 1.140 * (V00 - 128), \text{if } R00 > 255 \text{ then } R00 = 255, \text{if } R00 < 0 \text{ then } R00 = 0 \\
 G00 &= Y00 - 0.395 * (U00 - 128) - 0.581 * (V00 - 128), \text{if } G00 > 255 \text{ then } G00 = 255, \\
 &\quad \text{if } G00 < 0 \text{ then } G00 = 0 \\
 B00 &= Y00 + 2.032 * (U00 - 128), \text{if } B00 > 255 \text{ then } B00 = 255, \text{if } B00 < 0 \text{ then } B00 = 0 \\
 R01 &= Y01 + 1.140 * (V00 - 128), \text{if } R01 > 255 \text{ then } R01 = 255, \text{if } R01 < 0 \text{ then } R01 = 0 \\
 G01 &= Y01 - 0.395 * (U00 - 128) - 0.581 * (V00 - 128), \text{if } G01 > 255 \text{ then } G01 = 255, \\
 &\quad \text{if } G01 < 0 \text{ then } G01 = 0 \\
 B01 &= Y01 + 2.032 * (U00 - 128), \text{if } B01 > 255 \text{ then } B01 = 255, \text{if } B01 < 0 \text{ then } B01 = 0 \\
 R10 &= Y10 + 1.140 * (V00 - 128), \text{if } R10 > 255 \text{ then } R10 = 255, \text{if } R10 < 0 \text{ then } R10 = 0
 \end{aligned}
 \tag{34}$$

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

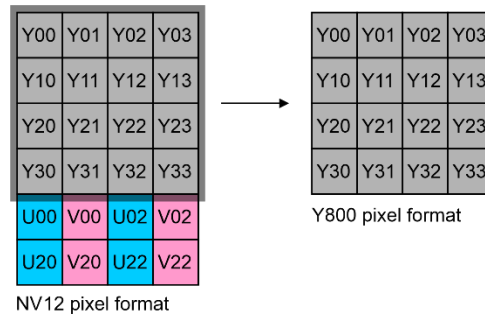
$$B10 = Y10 + 2.032 * (U00 - 128), \text{if } B10 > 255 \text{ then } B10 = 255, \text{if } B10 < 0 \text{ then } B10 = 0$$

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

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

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

### 5.39. NV12 to Y800

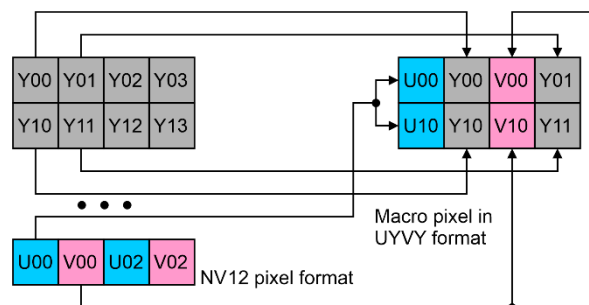


**Figure 46** – Conversion NV12 to Y800.

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

(35)

### 5.40. NV12 to UYVY



**Figure 47** – Conversion NV12 to UYVY.

$$Y00 = Y00$$

$$Y01 = Y01$$

$$Y10 = Y10$$

$$Y11 = Y11$$

$$U00 = U00$$

$$U10 = U00$$

$$V00 = V00$$

$$V10 = V00$$

(36)



## 5.41. NV12 to YUY2

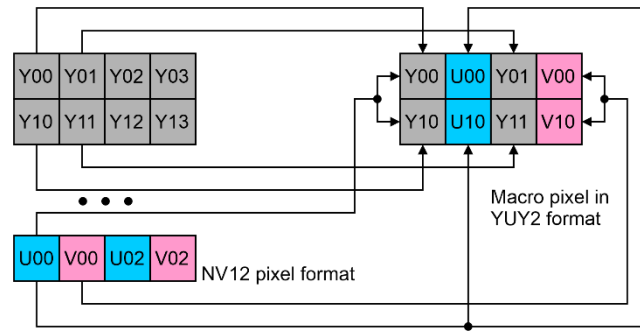


Figure 48 – Conversion NV12 to YUY2.

$  \begin{aligned}  Y00 &= Y00 \\  Y01 &= Y01 \\  Y10 &= Y10 \\  Y11 &= Y11 \\  U00 &= U00 \\  U10 &= U00 \\  V00 &= V00 \\  V10 &= V00  \end{aligned}  $	(37)
---	------

## 5.42. NV12 to YUV1

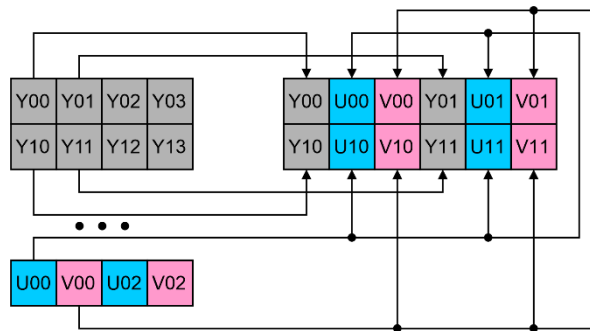


Figure 49 – Conversion NV12 to YUV1.

$  \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. DATA STRUCTURES

As the input and output data for the program library, the **Frame** image class declared in the **VideoDataStructures.h** file is used. The declaration of the **Frame** class is given below.

```

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();

};

```

Table 4 – Description of fields of class **Frame**.

Class field	Description
data	Pointer to image data.
width	Image width in pixels.
height	Image height in pixels.
size	Size of image data in bytes. Corresponds to the size of the image data buffer.
fourcc	<p>FOURCC code of image pixels format. The following pixel formats are implemented in the software library: <b>RGBR (RGB24)</b>, <b>BGRB (BGR24)</b>, <b>Y800</b>, <b>UYVY</b>, <b>YUY2</b>, <b>YUV1</b> and <b>NV12</b>. You can get the numerical value of the code using the macro <b>MAKE_FOURCC_CODE</b> declared in file <b>VideoDataStructures.h</b>. Also, the numerical values of the valid FOURCC codes are defined in the enum <b>ValidFourccCodes</b> declared in file <b>VideoDataStructures.h</b>:</p> <pre> enum class ValidFourccCodes {      RGB24 = MAKE_FOURCC_CODE('R', 'G', 'B', 'R'),     BGR24 = 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'),     JPEG = MAKE_FOURCC_CODE('J', 'P', 'E', 'G'),     JPG2 = MAKE_FOURCC_CODE('J', 'P', 'G', '2'), </pre>

	<pre> H264 = MAKE_FOURCC_CODE('H', '2', '6', '4'), H265 = MAKE_FOURCC_CODE('H', '2', '6', '5') }; </pre>
sourceID	ID of the image or video source. User Defined.
frameID	ID of the image or frame of the video. It can be used for numbering frames and is user-defined.

Table 5 – Description of methods of the **Frame** class.

Class methos	Description
Frame()	Default class constructor.
~Frame()	Class destructor.
Frame(const Frame& src)	Copy class constructor.
Frame(uint32_t width, uint32_t height, uint32_t fourcc)	Class constructor with parameters: <b>width</b> – image width in pixels; <b>height</b> – image height in pixels; <b>fourcc</b> – FOURCC code of pixels format. A constructor with parameters allocates memory for image data and fills it with zero values.
Frame& operator= (const Frame& src)	Copy operator.
void Release()	The method of freeing memory. Releases the memory allocated for the image data array and initializes all fields with default values.

## 7. DESCRIPTION OF PixelFormatConverter CLASS

### 7.1. PixelFormatConverter class declaration

The **PixelFormatConverter** class is designed to convert image pixel formats. The class is declared in the file **PixelFormatConvrtter.h**. Class methods do not perform any background tasks. Calculations begin with a call to the **Convert(...)** method of the class and end with returning control to the calling thread. Instances of the **Frame** class are used as input and output data. Below is the class declaration.

```

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. Convert(...) method

The **Conver(...)** method is designed to convert the image pixel format. The method converts the original format of the pixels specified in the input image (fourcc field of the Frame class) to the format specified in the object of the output image class (fourcc field of the Frame class). Method declaration:

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

*Parameters:*

src	A link to an object of the Frame class of the source image. <b>Note:</b> The height of the source image must be a multiple of 2.
dst	A link to an object of the Frame class of the result image.

*Return value:*

The method returns TRUE in case success or returns FALSE in following cases:

1. if the height of the original image is not a multiple of 2;
2. if FOURCC code of source image (fourcc field of the Frame class) does not match the allowed list (the ValidFourccCodes enum in the VideoDataStructures.h file);
3. if FOURCC code for the result image (fourcc field of an object dst) does not match the allowed list (enum ValidFourccCodes in the file VideoDataStructures.h);
4. if the source image (src) is not initialized: the image size is zero, there is no allocated image data array, or the image data array size does not match the specified pixel format.

### 7.3. GetVersion(...) method

The GetVersion(...) method is designed to obtain the numerical value of the current version of the PixelFormatConverter software library. Method declaration:

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

*Parameters:*

major	Link to the returned major value of the library version.
minor	Link to the returned minor value of the library version.

### 7.4. isFourccCodeValid(...) method

The isFourccCodeValid(...) method is used to check if the FOURCC code matches the allowed list defined in the ValidSourccCodes enumeration (file VideoDataStructures.h). Method declaration:

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

*Parameters:*

fourcc	FOURCC code.
--------	--------------

*Return value:*

The method returns TRUE if FOURCC code is allowed or returns FALSE.

### 7.5. GetSupportedFourccCodes() method

The GetSupportedFourccCodes() method is intended to get a list of allowed FOURR codes. Method Declaration:

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

*Return value:*

List of valid FOURCC codes.

## 8. EXAMPLE

The following is an example of converting an image with the pixel format YUV1 to BGRB (BGR24). The example compares the correctness of the conversion with the conversion implemented in the OpenCV open library.

```
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;
    }
}
```