



# Frame C++ class

---

v5.0.6

## Table of contents

---

- [Overview](#)
- [Versions](#)
- [Supported pixel formats](#)
- [Frame class description](#)
  - [Frame class declaration](#)
  - [Default constructor](#)
  - [Constructor with parameters](#)
  - [Copy-constructor](#)
  - [getVersion method](#)
  - [Copy operator =](#)
  - [cloneTo method](#)
  - [Compare operator ==](#)
  - [Compare operator !=](#)
  - [release method](#)
  - [serialize method](#)
  - [deserialize method](#)
  - [Frame class public members](#)
- [Build and connect to your project](#)

## Overview

---

Frame class is basic class for other projects. Main file **Frame.h** contains declaration of **Frame** class and **Fourcc** enum which describes pixel formats supported by **Frame** class.

## Versions

---

**Table 1** - Library versions.

Version	Release date	What's new
1.0.0	08.09.2020	First Frame class version in VideoDataStructures repository.
2.0.0	19.11.2020	Transformation matrix added.
3.0.0	20.12.2020	Added NV12 pixel format.
4.0.0	20.01.2023	<ul style="list-style-type: none"> <li>- Interface changed.</li> <li>- Added new pixel formats.</li> <li>- Added shared_ptr for frame data.</li> <li>- Added new methods for copy and compare data.</li> <li>- Added new method for serieliazation and deserialization frame data.</li> <li>- Tests changed.</li> </ul>
5.0.0	19.03.2023	<ul style="list-style-type: none"> <li>- shared_ptr replaced by normal pointer.</li> <li>- Documentation updated.</li> </ul>
5.0.1	29.05.2023	- Pixel format description (NV12 and NV21) mistake fixed.
5.0.2	22.06.2023	<ul style="list-style-type: none"> <li>- Added LICENSE.</li> <li>- Repository made public.</li> </ul>
5.0.3	22.06.2023	- Added build guide.
5.0.4	06.07.2023	- Documentation updated.
5.0.5	12.11.2023	- Fixed errors serialization/deserialization functions.
5.0.6	14.12.2023	- Memory leakage from "=" operator fixed.

## Supported pixel formats

**Frame.h** file contains Fourcc enum which defines supported pixel formats. **Fourcc** enum declaration:

```

namespace cr
{
    namespace video
    {

        /// Macro to make FOURCC code.
        #define MAKE_FOURCC_CODE(a,b,c,d) (((d)<<24)|((c)<<16)|((b)<<8)|(a)))

        /**
         * @brief FOURCC codes enum.
         */
        enum class Fourcc
        {
            /// RGB 24bit pixel format.
            /// https://docs.kernel.org/userspace-api/media/v4l/pixfmt-rgb.html#v4l2-pix-fmt-rgb24
            RGB24 = MAKE_FOURCC_CODE('R', 'G', 'B', '3'),
            /// BGR 24bit pixel format.

```

```

    /// https://docs.kernel.org/userspace-api/media/v4l/pixfmt-rgb.html#v4l2-pix-fmt-
bgr24
    BGR24 = MAKE_FOURCC_CODE('B', 'G', 'R', '3'),
    /// YUYV 16bits per pixel format.
    /// https://docs.kernel.org/userspace-api/media/v4l/pixfmt-packed-yuv.html#v4l2-pix-
fmt-yuyv
    YUYV = MAKE_FOURCC_CODE('Y', 'U', 'Y', 'V'),
    /// UYVY 16bits per pixel format.
    /// https://docs.kernel.org/userspace-api/media/v4l/pixfmt-packed-yuv.html#v4l2-pix-
fmt-vyuy
    UYVY = MAKE_FOURCC_CODE('U', 'Y', 'V', 'Y'),
    /// Grayscale 8bit.
    /// https://docs.kernel.org/userspace-api/media/v4l/pixfmt-yuv-luma.html#v4l2-pix-
fmt-grey
    GRAY = MAKE_FOURCC_CODE('G', 'R', 'A', 'Y'),
    /// YUV 24bit per pixel format.
    /// https://docs.kernel.org/userspace-api/media/v4l/pixfmt-packed-yuv.html#v4l2-pix-
fmt-yuv24
    YUV24 = MAKE_FOURCC_CODE('Y', 'U', 'V', '3'),
    /// NV12 pixel format.
    /// https://docs.kernel.org/userspace-api/media/v4l/pixfmt-yuv-planar.html#v4l2-pix-
fmt-nv12
    NV12 = MAKE_FOURCC_CODE('N', 'V', '1', '2'),
    /// NV21 pixel format.
    /// https://docs.kernel.org/userspace-api/media/v4l/pixfmt-yuv-planar.html#v4l2-pix-
fmt-nv21
    NV21 = MAKE_FOURCC_CODE('N', 'V', '2', '1'),
    /// YU12 (YUV420) - Planar pixel format.
    /// https://docs.kernel.org/userspace-api/media/v4l/pixfmt-yuv-planar.html#v4l2-pix-
fmt-yuv420
    YU12 = MAKE_FOURCC_CODE('Y', 'U', '1', '2'),
    /// YV12 (YVU420) - Planar pixel format.
    /// https://docs.kernel.org/userspace-api/media/v4l/pixfmt-yuv-planar.html#v4l2-pix-
fmt-yuv420
    YV12 = MAKE_FOURCC_CODE('Y', 'V', '1', '2'),
    /// JPEG compressed format.
    /// https://docs.kernel.org/userspace-api/media/v4l/pixfmt-compressed.html#v4l2-pix-
fmt-jpeg
    JPEG = MAKE_FOURCC_CODE('J', 'P', 'E', 'G'),
    /// H264 compressed format.
    /// https://docs.kernel.org/userspace-api/media/v4l/pixfmt-compressed.html#v4l2-pix-
fmt-h264
    H264 = MAKE_FOURCC_CODE('H', '2', '6', '4'),
    /// HEVC compressed format.
    /// https://docs.kernel.org/userspace-api/media/v4l/pixfmt-compressed.html#v4l2-pix-
fmt-hevc
    HEVC = MAKE_FOURCC_CODE('H', 'E', 'V', 'C')
};
}
}

```

**Table 2** - Bytes layout of supported pixel formats. Example of 4x4 pixels image.

pixel

R <sub>00</sub>	G <sub>00</sub>	B <sub>00</sub>	R <sub>01</sub>	G <sub>01</sub>	B <sub>01</sub>	R <sub>02</sub>	G <sub>02</sub>	B <sub>02</sub>	R <sub>03</sub>	G <sub>03</sub>	B <sub>03</sub>
R <sub>10</sub>	G <sub>10</sub>	B <sub>10</sub>	R <sub>11</sub>	G <sub>11</sub>	B <sub>11</sub>	R <sub>12</sub>	G <sub>12</sub>	B <sub>12</sub>	R <sub>13</sub>	G <sub>13</sub>	B <sub>13</sub>
R <sub>20</sub>	G <sub>20</sub>	B <sub>20</sub>	R <sub>21</sub>	G <sub>21</sub>	B <sub>21</sub>	R <sub>22</sub>	G <sub>22</sub>	B <sub>22</sub>	R <sub>23</sub>	G <sub>23</sub>	B <sub>23</sub>
R <sub>30</sub>	G <sub>30</sub>	B <sub>30</sub>	R <sub>31</sub>	G <sub>31</sub>	B <sub>31</sub>	R <sub>32</sub>	G <sub>32</sub>	B <sub>32</sub>	R <sub>33</sub>	G <sub>33</sub>	B <sub>33</sub>

RGB24

pixel

B <sub>00</sub>	G <sub>00</sub>	R <sub>00</sub>	B <sub>01</sub>	G <sub>01</sub>	R <sub>01</sub>	B <sub>02</sub>	G <sub>02</sub>	R <sub>02</sub>	B <sub>03</sub>	G <sub>03</sub>	R <sub>03</sub>
B <sub>10</sub>	G <sub>10</sub>	R <sub>10</sub>	B <sub>11</sub>	G <sub>11</sub>	R <sub>11</sub>	B <sub>12</sub>	G <sub>12</sub>	R <sub>12</sub>	B <sub>13</sub>	G <sub>13</sub>	R <sub>13</sub>
B <sub>20</sub>	G <sub>20</sub>	R <sub>20</sub>	B <sub>21</sub>	G <sub>21</sub>	R <sub>21</sub>	B <sub>22</sub>	G <sub>22</sub>	R <sub>22</sub>	B <sub>23</sub>	G <sub>23</sub>	R <sub>23</sub>
B <sub>30</sub>	G <sub>30</sub>	R <sub>30</sub>	B <sub>31</sub>	G <sub>31</sub>	R <sub>31</sub>	B <sub>32</sub>	G <sub>32</sub>	R <sub>32</sub>	B <sub>33</sub>	G <sub>33</sub>	R <sub>33</sub>

BGR24

pixel

Y <sub>00</sub>	U <sub>00</sub>	V <sub>00</sub>	Y <sub>01</sub>	U <sub>01</sub>	V <sub>01</sub>	Y <sub>02</sub>	U <sub>02</sub>	V <sub>02</sub>	Y <sub>03</sub>	U <sub>03</sub>	V <sub>03</sub>
Y <sub>10</sub>	U <sub>10</sub>	V <sub>10</sub>	Y <sub>11</sub>	U <sub>11</sub>	V <sub>11</sub>	Y <sub>12</sub>	U <sub>12</sub>	V <sub>12</sub>	Y <sub>13</sub>	U <sub>13</sub>	V <sub>13</sub>
Y <sub>20</sub>	U <sub>20</sub>	V <sub>20</sub>	Y <sub>21</sub>	U <sub>21</sub>	V <sub>21</sub>	Y <sub>22</sub>	U <sub>22</sub>	V <sub>22</sub>	Y <sub>23</sub>	U <sub>23</sub>	V <sub>23</sub>
Y <sub>30</sub>	U <sub>30</sub>	V <sub>30</sub>	Y <sub>31</sub>	U <sub>31</sub>	V <sub>31</sub>	Y <sub>32</sub>	U <sub>32</sub>	V <sub>32</sub>	Y <sub>33</sub>	U <sub>33</sub>	V <sub>33</sub>

YUV24

pixel

Y <sub>00</sub>	Y <sub>01</sub>	Y <sub>02</sub>	Y <sub>03</sub>
Y <sub>10</sub>	Y <sub>11</sub>	Y <sub>12</sub>	Y <sub>13</sub>
Y <sub>20</sub>	Y <sub>21</sub>	Y <sub>22</sub>	Y <sub>23</sub>
Y <sub>30</sub>	Y <sub>31</sub>	Y <sub>32</sub>	Y <sub>33</sub>

GRAY

macro pixel

Y <sub>00</sub>	U <sub>00</sub>	Y <sub>01</sub>	V <sub>00</sub>	Y <sub>02</sub>	U <sub>02</sub>	Y <sub>03</sub>	V <sub>02</sub>
Y <sub>10</sub>	U <sub>10</sub>	Y <sub>11</sub>	V <sub>10</sub>	Y <sub>12</sub>	U <sub>12</sub>	Y <sub>13</sub>	V <sub>12</sub>
Y <sub>20</sub>	U <sub>20</sub>	Y <sub>21</sub>	V <sub>20</sub>	Y <sub>22</sub>	U <sub>22</sub>	Y <sub>23</sub>	V <sub>22</sub>
Y <sub>30</sub>	U <sub>30</sub>	Y <sub>31</sub>	V <sub>30</sub>	Y <sub>32</sub>	U <sub>32</sub>	Y <sub>33</sub>	V <sub>32</sub>

YUYV

macro pixel

U <sub>00</sub>	Y <sub>00</sub>	V <sub>00</sub>	Y <sub>01</sub>	U <sub>02</sub>	Y <sub>02</sub>	Y <sub>03</sub>	V <sub>02</sub>
U <sub>10</sub>	Y <sub>10</sub>	V <sub>10</sub>	Y <sub>11</sub>	U <sub>12</sub>	Y <sub>12</sub>	Y <sub>13</sub>	V <sub>12</sub>
U <sub>20</sub>	Y <sub>20</sub>	V <sub>20</sub>	Y <sub>21</sub>	U <sub>22</sub>	Y <sub>22</sub>	Y <sub>23</sub>	V <sub>22</sub>
U <sub>30</sub>	Y <sub>30</sub>	V <sub>30</sub>	Y <sub>31</sub>	U <sub>32</sub>	Y <sub>32</sub>	Y <sub>33</sub>	V <sub>32</sub>

UYVY

macro pixel

Y <sub>00</sub>	Y <sub>01</sub>	Y <sub>02</sub>	Y <sub>03</sub>
Y <sub>10</sub>	Y <sub>11</sub>	Y <sub>12</sub>	Y <sub>13</sub>
Y <sub>20</sub>	Y <sub>21</sub>	Y <sub>22</sub>	Y <sub>23</sub>
Y <sub>30</sub>	Y <sub>31</sub>	Y <sub>32</sub>	Y <sub>33</sub>
U <sub>00</sub>	V <sub>00</sub>	U <sub>02</sub>	V <sub>02</sub>
U <sub>20</sub>	V <sub>20</sub>	U <sub>22</sub>	V <sub>22</sub>

NV12

macro pixel

Y <sub>00</sub>	Y <sub>01</sub>	Y <sub>02</sub>	Y <sub>03</sub>
Y <sub>10</sub>	Y <sub>11</sub>	Y <sub>12</sub>	Y <sub>13</sub>
Y <sub>20</sub>	Y <sub>21</sub>	Y <sub>22</sub>	Y <sub>23</sub>
Y <sub>30</sub>	Y <sub>31</sub>	Y <sub>32</sub>	Y <sub>33</sub>
V <sub>00</sub>	U <sub>00</sub>	V <sub>02</sub>	U <sub>02</sub>
V <sub>20</sub>	U <sub>20</sub>	V <sub>22</sub>	U <sub>22</sub>

NV21

macro pixel

Y <sub>00</sub>	Y <sub>01</sub>	Y <sub>02</sub>	Y <sub>03</sub>
Y <sub>10</sub>	Y <sub>11</sub>	Y <sub>12</sub>	Y <sub>13</sub>
Y <sub>20</sub>	Y <sub>21</sub>	Y <sub>22</sub>	Y <sub>23</sub>
Y <sub>30</sub>	Y <sub>31</sub>	Y <sub>32</sub>	Y <sub>33</sub>
U <sub>00</sub>	U <sub>02</sub>	U <sub>20</sub>	U <sub>22</sub>
V <sub>00</sub>	V <sub>02</sub>	V <sub>20</sub>	V <sub>22</sub>

YU12

macro pixel

Y <sub>00</sub>	Y <sub>01</sub>	Y <sub>02</sub>	Y <sub>03</sub>
Y <sub>10</sub>	Y <sub>11</sub>	Y <sub>12</sub>	Y <sub>13</sub>
Y <sub>20</sub>	Y <sub>21</sub>	Y <sub>22</sub>	Y <sub>23</sub>
Y <sub>30</sub>	Y <sub>31</sub>	Y <sub>32</sub>	Y <sub>33</sub>
V <sub>00</sub>	V <sub>02</sub>	V <sub>20</sub>	V <sub>22</sub>
U <sub>00</sub>	U <sub>02</sub>	U <sub>20</sub>	U <sub>22</sub>

YV12

# Frame class description

---

## Frame class declaration

---

**Frame.h** file contains **Frame** class declaration. Frame class declaration:

```
namespace cr
{
    namespace video
    {
        /**
         * @brief video frame class.
         */
        class Frame
        {
        public:

            /**
             * @brief Get string of current class version.
             * @return String of current class version.
             */
            static std::string getVersion();

            /**
             * @brief Default class constructor.
             */
            Frame();

            /**
             * @brief Class constructor with parameters.
             * @param width Frame width (pixels).
             * @param height Frame height (pixels).
             * @param fourcc FOURCC code of data format.
             * @param size Frame data size (bytes).
             * @param data Pointer to data buffer.
             */
            Frame(uint32_t width,
                  uint32_t height,
                  Fourcc fourcc,
                  uint32_t size = 0,
                  uint8_t* data = nullptr);

            /**
             * @brief Copy class constructor.
             * @param src Source class object.
             */
            Frame(Frame& src);

            /**
             * @brief Class destructor.
             */
            ~Frame();

            /**
```

```

    * @brief Operator "=". Operator makes full copy of data.
    * @param src Source frame object.
    */
    Frame& operator= (const Frame& src);

/**
    * @brief Operator "!=". Operator to compare two frame objects.
    * @param src Source frame object.
    * @return TRUE if the frames are not identical or FALSE.
    */
    bool operator!= (Frame& src);

/**
    * @brief Operator "==". Operator to compare two frame objects.
    * @param src Source frame object.
    * @return TRUE if the frames are identical or FALSE.
    */
    bool operator== (Frame& src);

/**
    * @brief Clone data. Method copies frame and copy just pointer to data.
    * @param dst Output frame.
    */
    void cloneTo(Frame& dst);

/**
    * @brief Free frame memory.
    */
    void release();

/**
    * @brief Serialize frame data. The method will encode data with params.
    * @param data Pointer to data buffer.
    *          Buffer size mus be >= frame data size + 26.
    * @param size Size of serialized data.
    */
    void serialize(uint8_t* data, int& size);

/**
    * @brief Deserialize data data to frame object.
    * @param data Pointer to serialized data.
    * @param size Size of serialized data.
    * @return TRUE if the data deserialized or FALSE.
    */
    bool deserialize(uint8_t* data, int size);

    /// Frame width (pixels).
    uint32_t width{0};
    /// Frame height (pixels).
    uint32_t height{0};
    /// FOURCC code of data format.
    Fourcc fourcc{Fourcc::YUV24};
    /// Frame data size (bytes).
    uint32_t size{0};
    /// ID of frame.

```

```

uint32_t frameId{0};
/// ID of video source.
uint32_t sourceId{0};
/// Pointer to frame data.
uint8_t* data{nullptr};
};
}
}

```

## Default constructor

Default Frame class constructor doesn't do anything. It doesn't allocate memory. Constructor declaration:

```
Frame();
```

## Constructor with parameters

Constructor with parameters allocates memory and initilises Frame attributes (size, pixelsl format etc.). By default allocated memory filled by 0. Constructor declaration:

```
Frame(uint32_t width, uint32_t height, Fourcc fourcc, uint32_t size = 0, uint8_t* data = nullptr);
```

Parameter	Description
width	Frame width. Must be > 0.
height	Frame height. Must be > 0.
fourcc	Pixel format according to <b>Fourcc</b> enum declared in Frame.h file.
size	Optional parameter. Size of external frame data. If user wants to \initialize frame data from another buffer it can be don by initializing parameter <b>size</b> and <b>data</b> .
data	Optional parameter. Pointer to external frame data to be copied.

Example of frame initialization:

```

// Frame filled by 0.
cr::video::Frame image1(640, 480, cr::video::Fourcc::RGB24);

// Copy.
cr::video::Frame image2 = cr::video::Frame(640, 480, cr::video::Fourcc::YUV24);

// Init by external data. Data will be copied.
cr::video::Frame image3(640, 480, cr::video::Fourcc::BGR24, 921600, externalDataBuffer);

// Init frame attributes manually.
Frame image4;
image4.data = externalDataBuffer;

```

```

image4.width = 640;
image4.height = 480;
image4.size = image4.width * image4.height * 3;
image4.fourcc = cr::video::Fourcc::BGR24;

// Dynamic allocation.
cr::video::Frame* image5 = new cr::video::Frame(640, 480, cr::video::Fourcc::YUV24);

```

## Copy-constructor

Copy constructor copy frame data from other Frame class instance. Constructor declaration:

```
Frame(Frame& src);
```

Example of frame initialization:

```

// Frame filled by 0.
cr::video::Frame image1(640, 480, cr::video::Fourcc::RGB24);

// Copy-constructor.
cr::video::Frame image2(image1);

// Copy-constructor.
cr::video::Frame image3 = cr::video::Frame(image1);

```

## getVersion method

**getVersion()** method return string of current version of **Frame** class. Method declaration:

```
static std::string getVersion();
```

Method can be used without **Frame** class instance. Example:

```
std::cout << "Frame class version: " << cr::video::Frame::getVersion() << std::endl;
```

## Copy operator =

Copy operator "=" intended to full copy of frame data. Operator copies frame data and frame attributes. Operator declaration:

```
Frame& operator= (const Frame& src);
```

Example:



```
// Frame filled by 0.
cr::video::Frame image1(640, 480, cr::video::Fourcc::RGB24);

// Make copy.
cr::video::Frame image2 = image1;
```

## cloneTo method

**cloneTo(...)** method designed to clone frame object without copy of data. Method copies frame attributes and initialize pointer to frame data without copy of data. Method declaration:

```
void cloneTo(Frame& dst);
```

Parameter	Description
dst	Frame object for initialization. Method initialize only frame attributes and copies pointer to frame data.

Example:

```
// Frame filled by 0.
cr::video::Frame image1(640, 480, cr::video::Fourcc::RGB24);

// Clone.
cr::video::Frame image2;
image1.cloneTo(image2);
```

## Compare operator ==

Compare operator "==" compares data attributes and frame data of to Frame objects. Operator declaration:

```
bool operator== (Frame& src);
```

**Returns:** TRUE if all attributes and data of two Frame objects are identical (or when comparing an object to itself) or FALSE if not.

Example:

```
// Frame filled by 0.
cr::video::Frame image1(640, 480, cr::video::Fourcc::RGB24);
cr::video::Frame image2(640, 480, cr::video::Fourcc::RGB24);

// Compare.
if (image1 == image2)
    std::cout << "Identical" << std::endl;

// Change data.
```

```

image1.data[0] = rand() % 255;

// Compare.
if (image1 == image2)
    std::cout << "Identical" << std::endl;
else
    std::cout << "Not identical" << std::endl;

```

## Compare operator !=

Compare operator "!=" compares data attributes and frame data of to Frame objects. Operator declaration:

```

bool operator!= (Frame& src);

```

**Returns:** TRUE if all attributes and data of two Frame objects are identical (or when comparing an object to itself) or FALSE if not.

Example:

```

// Frame filled by 0.
cr::video::Frame image1(640, 480, cr::video::Fourcc::RGB24);
cr::video::Frame image2(640, 480, cr::video::Fourcc::RGB24);

// Compare.
if (image1 != image2)
    std::cout << "Not identical" << std::endl;
else
    std::cout << "Identical"

// Change data.
image1.data[0] = rand() % 255;

// Compare.
if (image1 != image2)
    std::cout << "Not identical" << std::endl;
else
    std::cout << "Identical" << std::endl;

```

## release method

**release()** method intended to release allocated memory and reset frame attributes. Method declaration:

```

void release();

```

Example:

```
// Create Frame object with memory allocation.
cr::video::Frame image1(640, 480, cr::video::Fourcc::RGB24);

// Release Frame object.
image1.release();
```

## serialize method

**serialize(...)** method intended for serialization of Frame object with data. Sometimes the user needs to serialize an object in order to transfer or write it somewhere. Method declaration:

```
void serialize(uint8_t* data, int& size);
```

Parameter	Description
data	Pointer to data buffer.
size	Size of serialized data.

Example:

```
// Init frames.
Frame srcFrame(640, 480, Fourcc::BGR24);

// Fill source frame.
for (uint32_t i = 0; i < srcFrame.size; ++i)
    srcFrame.data[i] = (uint8_t)(rand() % 255);

// Serialize data.
uint8_t* data = new uint8_t[1920 * 1080 * 4];
int size = 0;
srcFrame.serialize(data, size);
```

## deserialize method

**deserialize(...)** method intended for deserialization of Frame object. Method declaration:

```
bool deserialize(uint8_t* data, int size);
```

Parameter	Description
data	Pointer to data buffer.
size	Size of serialized data.

**Returns:** TRUE if the data is deserialized or FALSE if not.

Example:

```

// Init frames.
Frame srcFrame(640, 480, Fourcc::BGR24);
Frame dstFrame(1280, 720, Fourcc::YUV24);

// Fill source frame.
for (uint32_t i = 0; i < srcFrame.size; ++i)
    srcFrame.data[i] = (uint8_t)(rand() % 255);

// Serialize data.
uint8_t* data = new uint8_t[1920 * 1080 * 4];
int size = 0;
srcFrame.serialize(data, size);

// Deserialize data.
if (!dstFrame.deserialize(data, size))
{
    std::cout << "Data not deserialized" << std::endl;
    return false;
}

// Compare attributes.
if (srcFrame.size != dstFrame.size)
{
    std::cout << "[" << __LINE__ << "]" " << __FILE__ << " : ERROR" << std::endl;
    return false;
}
if (srcFrame.width != dstFrame.width)
{
    std::cout << "[" << __LINE__ << "]" " << __FILE__ << " : ERROR" << std::endl;
    return false;
}
if (srcFrame.height != dstFrame.height)
{
    std::cout << "[" << __LINE__ << "]" " << __FILE__ << " : ERROR" << std::endl;
    return false;
}
if (srcFrame.fourcc != dstFrame.fourcc)
{
    std::cout << "[" << __LINE__ << "]" " << __FILE__ << " : ERROR" << std::endl;
    return false;
}
if (srcFrame.sourceId != dstFrame.sourceId)
{
    cout << "[" << __LINE__ << "]" " << __FILE__ << " : ERROR" << std::endl;
    return false;
}
if (srcFrame.frameId != dstFrame.frameId)
{
    std::cout << "[" << __LINE__ << "]" " << __FILE__ << " : ERROR" << std::endl;
    return false;
}

// Compare frame data.
for (uint32_t i = 0; i < srcFrame.size; ++i)
{

```

```

    if (srcFrame.data[i] != dstFrame.data[i])
    {
        std::cout << "[" << __LINE__ << "]" " << __FILE__ << " : ERROR" << std::endl;
        return false;
    }
}

```

## Frame class public members

Frame class public members declaration:

```

/// Frame width (pixels).
uint32_t width{0};
/// Frame height (pixels).
uint32_t height{0};
/// FOURCC code of data format.
Fourcc fourcc{Fourcc::YUV24};
/// Frame data size (bytes).
uint32_t size{0};
/// ID of frame.
uint32_t frameId{0};
/// ID of video source.
uint32_t sourceId{0};
/// Pointer to frame data.
uint8_t* data{nullptr};

```

**Table 3** - Frame class public members.

Field	Description
width	Frame width.
height	Frame height.
fourcc	Fourcc code according to <b>Fourcc</b> enum declared in <b>Frame.h</b> file.
size	Size of frame data.
frameId	Frame ID. User defines this field.
sourceId	Source ID. User defines this field.
data	Pointer to frame data.

## Build and connect to your project

Typical commands to build **Frame** library:

```
git clone https://github.com/ConstantRobotics-Ltd/Frame.git
cd Frame
mkdir build
cd build
cmake ..
make
```

If you want connect Frame library to your CMake project as source code you can make follow. For example, if your repository has structure:

```
CMakeLists.txt
src
    CMakeList.txt
    yourLib.h
    yourLib.cpp
```

You can add repository **Frame** as submodule by command:

```
cd <your repository folder>
git submodule add https://github.com/ConstantRobotics-Ltd/Frame.git 3rdparty/Frame
```

In you repository folder will be created folder **3rdparty/Frame** which contains files of **Frame** repository. New structure of your repository:

```
CMakeLists.txt
src
    CMakeList.txt
    yourLib.h
    yourLib.cpp
3rdparty
    Frame
```

Create CMakeLists.txt file in **3rdparty** folder. CMakeLists.txt should contain:

```
cmake_minimum_required(VERSION 3.13)

#####
## 3RD-PARTY
## dependencies for the project
#####
project(3rdparty LANGUAGES CXX)

#####
## SETTINGS
## basic 3rd-party settings before use
#####
# To inherit the top-level architecture when the project is used as a submodule.
SET(PARENT ${PARENT}_YOUR_PROJECT_3RDPARTY)
# Disable self-overwriting of parameters inside included subdirectories.
SET(${PARENT}_SUBMODULE_CACHE_OVERWRITE OFF CACHE BOOL "" FORCE)

#####
## CONFIGURATION
```

```
## 3rd-party submodules configuration
#####
SET(${PARENT}_SUBMODULE_FRAME ON CACHE BOOL "" FORCE)
if (${PARENT}_SUBMODULE_FRAME)
    SET(${PARENT}_FRAME ON CACHE BOOL "" FORCE)
    SET(${PARENT}_FRAME_TEST OFF CACHE BOOL "" FORCE)
endif()

#####
## INCLUDING SUBDIRECTORIES
## Adding subdirectories according to the 3rd-party configuration
#####
if (${PARENT}_SUBMODULE_FRAME)
    add_subdirectory(Frame)
endif()
```

File **3rdparty/CMakeLists.txt** adds folder **Frame** to your project and excludes test application (Frame class tests) from compiling. Your repository new structure will be:

```
CMakeLists.txt
src
    CMakeList.txt
    yourLib.h
    yourLib.cpp
3rdparty
    CMakeLists.txt
    Frame
```

Next you need include folder 3rdparty in main **CMakeLists.txt** file of your repository. Add string at the end of your main **CMakeLists.txt**:

```
add_subdirectory(3rdparty)
```

Next you have to include Frame library in your **src/CMakeLists.txt** file:

```
target_link_libraries(${PROJECT_NAME} Frame)
```

Done!