



VCodecV4L2 C++ library

v2.1.0

Table of contents

- [Overview](#)
- [Versions](#)
- [VCodecV4L2 class description](#)
 - [VCodecV4L2 class declaration](#)
 - [transcode method](#)
 - [setParam method](#)
 - [getParam method](#)
 - [executeCommand method](#)
- [Data structures](#)
 - [VCodecCommand enum](#)
 - [VCodecParam enum](#)
- [Build and connect to your project](#)
- [Simple example](#)

Overview

VCodecV4L2 C++ library provides hardware video **encoding** for H264 and JPEG codecs for Raspberry Pi platform. **VCodecV4L2** class inherits interface and data structures from open source **VCodec** library and also includes **Logger** open source library. **VCodecV4L2** uses V4L2 API. Version 2.1.0 supports only **Raspberry Pi 4** and uses hardware encoders. The library provides simple programming interface to be implemented in different C++ projects for Raspberry Pi. The library was written with C++17 standard. The libraries are supplied as source code only. The library is a CMake project. Encoding time on **Raspberry Pi 4B**, msec:

codec / resolution	1920x1080	1280x720	640x512
H264	12 msec	5.4 msec	2.5 msec
JPEG	25 msec	13 msec	5.5 msec

Encoding time on **Raspberry Zero W2**, msec:

codec / resolution	1920x1080	1280x720	640x512
H264	19.5 msec	8.7 msec	2.8 msec
JPEG	33.4 msec	15.7 msec	5.1 msec

Versions

Table 1 - Library versions.

Version	Release date	What's new
1.0.0	29.06.2023	First version.
2.0.0	02.08.2023	- Interface changes to VCodec . - Added JPEG encoding support. - Test application updated.
2.0.1	21.08.2023	- Bug fixed in transcode on copy constructor of dst frame.
2.1.0	28.08.2023	- YUYV input pixel format changed to NV12 for H264 and JPEG codecs. - Added example application.

Library files

The VCodecV4L2 library is a CMake project. Library files:

```
CMakeLists.txt ----- Main CMake file of the library.
3rdparty ----- Folder with third-party libraries.
  CMakeLists.txt ----- CMake file which includes third-party libraries.
  Logger ----- Source code of Logger library.
  VCodec ----- Source code of VCodec library.
example ----- Folder with simple example of VCodecV4L2 usage.
  CMakeLists.txt ----- CMake file for example application.
  main.cpp ----- Source code file of example application.
test ----- Folder with codec test application.
  CMakeLists.txt ----- CMake file for codec test application.
  main.cpp ----- Source code file of codec test application.
src ----- Folder with source code of the library.
  CMakeLists.txt ----- CMake file of the library.
  VCodecV4L2.cpp ----- Source code file of the library.
  VCodecV4L2.h ----- Header file which includes VCodecV4L2 class
  declaration.
  VCodecV4L2Version.h ----- Header file which includes version of the library.
  VCodecV4L2Version.h.in ----- CMake service file to generate version file.
```

VCodecV4L2 class description

VCodecV4L2 class declaration

VCodecV4L2 class declared in VCodecV4L2.h file. Class declaration:

```
class VCodecV4L2 : public VCodec
{
public:
    /**
     * @brief Get library version.
     * @return String of current library version "Major.Minor.Patch".
     */
    static std::string getVersion();
    /**
     * @brief Class constructor.
     */
    VCodecV4L2();
    /**
     * @brief Class destructor.
     */
    ~VCodecV4L2();
    /**
     * @brief Encode video frame.
     * @param src Source RAW frame in NV12 format.
     * @param dst Result compressed frame (H264 or JPEG).
     * @return TRUE if frame was encoded or FLASE if not.
     */
    bool transcode(Frame& src, Frame& dst);
    /**
     * @brief Set parameter value.
     * @param id Parameter ID.
     * @param value Parameter value to set.
     * @return TRUE if parameter was set or FALSE if not.
     */
    bool setParam(VCodecParam id, float value);
    /**
     * @brief Get parameter value.
     * @param id Parameter ID.
     * @return Parameter value or -1 if parameter not supported.
     */
    float getParam(VCodecParam id);
    /**
     * @brief Execute command.
     * @param id Command ID.
     * @return TRUE if the command accepted or FALSE if not.
     */
    bool executeCommand(VCodecCommand id);
};
```

getVersion method

`getVersion()` method returns string of current version of **VCodecV4L2** class. Method declaration:

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

Method can be used without **VCodecV4L2** class instance:

```
cout << "VCodecV4L2 class version: " << VCodecV4L2::getVersion() << endl;
```

Console output:

```
VCodecV4L2 class version: 2.1.0
```

transcode method

`transcode(...)` method intended to encode and decode video frame ([Frame](#) class). Video codec encodes/decodes video frames frame-by-frame. Method declaration:

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

Parameter	Value
src	Source video frame (see Frame class description). To encode video src frame must have raw pixel format such as NV12 for H264 or BGR for JPEG . To decode video data src frame must have compressed pixel format (field fourcc of Frame class): JPEG, H264 .
dst	Result video frame (see Frame class description). To encode video data dst frame must have compressed pixel format (field fourcc of Frame class): JPEG, H264 . In contrary, src frame must be NV12, BGR

Returns: TRUE if frame was encoded/decoded or FALSE if not.

setParam method

`setParam(...)` method designed to set new video codec parameters value. Method declaration:

```
setParam(VCodecParam id, float value);
```

Parameter	Description
id	Video codec parameter ID according to VCodecParam enum.
value	Video codec parameter value.

Returns: TRUE is the parameter was set or FALSE if not.

getParam method

getParam(...) method designed to obtain video codec parameter value. Method declaration:

```
float getParam(VCodecParam id);
```

Parameter	Description
id	Video codec parameter ID according to VCodecParam enum.

Returns: parameter value or -1 if the parameter doesn't exist in particular video codec class.

executeCommand method

executeCommand(...) method designed to execute video codec command. Version 2.0.0 doesn't support commands. Method will return FALSE. Method declaration:

```
bool executeCommand(VCodecCommand id);
```

Parameter	Description
id	Video codec command ID according to VCodecCommand enum.

Returns: method returns FALSE in any case.

Data structures

VCodec.h file of [VCodec](#) library defines IDs for parameters (**VCodecParam** enum) and IDs for commands (**VCodecCommand** enum).

VCodecCommand enum

Enum declaration:

```
enum class VCodecCommand
{
    /// Reset.
    RESET = 1,
    /// Generate key frame. For H264 and H265 codecs.
    MAKE_KEY_FRAME
};
```

Table 2 - Video codec commands description. Some commands maybe unsupported by particular video codec class.

Command	Description
RESET	Not supported by VCodecV4L2.
MAKE_KEY_FRAME	Implemented but not tested.

VCodecParam enum

Enum declaration:

```
enum class VCodecParam
{
    /// [read/write] Log level:
    /// 0-Disable, 1-Console, 2-File, 3-Console and file.
    LOG_LEVEL = 1,
    /// [read/write] Bitrate, kbps. For H264 and H265 codecs.
    BITRATE_KBPS,
    /// [read/write] Quality 0-100%. For JPEG codecs.
    QUALITY,
    /// [read/write] FPS. For H264 and H265 codecs.
    FPS,
    /// [read/write] GOP size. For H264 and H265 codecs.
    GOP,
    /// [read/write] H264 profile: 0 - Baseline, 1 - Main, 2 - High.
    H264_PROFILE,
    /// [read/write] Codec type. Depends on implementation.
    TYPE,
    /// Custom 1. Depends on implementation.
    CUSTOM_1,
    /// Custom 2. Depends on implementation.
    CUSTOM_2,
    /// Custom 3. Depends on implementation.
    CUSTOM_3
};
```

Table 3 - Video codec params description. Some params maybe unsupported by particular video codec class.

Parameter	Access	Description
LOG_LEVEL	read / write	Logging mode. Default values: 0 - Disable, 1 - Only file, 2 - Only terminal, 3 - File and terminal.
BITRATE_KBPS	read / write	Bitrate, kbps. For H264 encoding only. According to this value, FPS and GOP size video codec calculate parameter for H264 encoding.
QUALITY	read / write	Quality 0(low quality)-100%(maximum quality). Only for JPEG encoding.

Parameter	Access	Description
FPS	read / write	FPS. For H264 encoding only. According to this value, FPS and GOP size video codec calculate parameter for H264 encoding.
GOP	read / write	GOP size (Period of key frames) for H264 encoding. Value: 1 - each output frame is key frame, 20 - each 20th frame is key frame etc.
H264_PROFILE	read / write	H264 profile for H264 encoding: 0 - Baseline, 1 - Main, 2 - High.
TYPE	read / write	Not supported.
CUSTOM_1	read / write	Not supported.
CUSTOM_2	read / write	Not supported.
CUSTOM_3	read / write	Not supported.

Build and connect to your project

Typical commands to build **VCodecV4L2** library:

```
git clone https://github.com/ConstantRobotics-Ltd/VCodecV4L2.git
cd VCodecV4L2
git submodule update --init --recursive
mkdir build
cd build
cmake ..
make
```

If you want connect **VCodecV4L2** 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 **VCodecV4L2** as submodule by commands:

```
cd <your repository folder>
git submodule add https://github.com/ConstantRobotics-Ltd/VCodecV4L2.git
3rdparty/VCodecV4L2
git submodule update --init --recursive
```

In your repository folder will be created folder **3rdparty/VCodecV4L2** which contains files of **VCodecV4L2** repository with subrepositories **Frame** and **VCodec**. New structure of your repository:

```
CMakeLists.txt
src
  CMakeList.txt
  yourLib.h
  yourLib.cpp
3rdparty
  VCodecV4L2
```

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(${PARENT}_SUBMODULE_VCODEC_V4L2 ON CACHE BOOL "" FORCE)
if (${PARENT}_SUBMODULE_VCODEC_V4L2)
  SET(${PARENT}_VCODEC_V4L2 ON CACHE BOOL "" FORCE)
  SET(${PARENT}_VCODEC_V4L2_TEST OFF CACHE BOOL "" FORCE)
  SET(${PARENT}_VCODEC_V4L2_EXAMPLE OFF CACHE BOOL "" FORCE)
endif()

#####
## INCLUDING SUBDIRECTORIES
## Adding subdirectories according to the 3rd-party configuration
#####
if (${PARENT}_SUBMODULE_VCODEC_V4L2)
  add_subdirectory(VCodecV4L2)
endif()
```

File **3rdparty/CMakeLists.txt** adds folder **VCodecV4L2** to your project and will exclude test application from compiling. Your repository new structure will be:


```
CMakeLists.txt
src
  CMakeList.txt
  yourLib.h
  yourLib.cpp
3rdparty
  CMakeLists.txt
  VCodecV4L2
```

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 VCodecV4L2 library in your **src/CMakeLists.txt** file:

```
target_link_libraries(${PROJECT_NAME} VCodecV4L2)
```

Done!

Simple example

Example application generates image colour pattern with moving rectangle and writes compressed data to binary file **"out.h264"**. Example shows how to create codec objects and how to encode video frames:

```
#include <iostream>
#include "VCodecV4L2.h"

/// Entry point.
int main(void)
{
    // Create codec.
    cr::video::VCodec* videoCodec = new cr::video::VCodecV4L2();

    // Set codec parameters.
    videoCodec->setParam(cr::video::VCodecParam::BITRATE_KBPS, 2500);
    videoCodec->setParam(cr::video::VCodecParam::GOP, 30);
    videoCodec->setParam(cr::video::VCodecParam::FPS, 30);
    videoCodec->setParam(cr::video::VCodecParam::H264_PROFILE, 0);

    // Create NV12 frame.
    const int width = 1280;
    const int height = 720;
    cr::video::Frame frameNv12(width, height, cr::video::Fourcc::NV12);
    // Fill NV12 frame by random values.
    for (uint32_t i = 0; i < frameNv12.size; ++i)
        frameNv12.data[i] = (uint8_t)i;

    // Create output H264 frame.
    cr::video::Frame frameH264(width, height, cr::video::Fourcc::H264);
```

```

// Create output frame.
FILE *outputFile = fopen("out.h264", "w+b");

// Params for moving object.
int objectwidth = 128;
int objectHeight = 128;
int directionX = 1;
int directionY = 1;
int objectX = width / 4;
int objectY = height / 2;

// Encode and record 200 frames.
for (uint8_t n = 0; n < 200; ++n)
{
    // Draw moving object.
    memset(frameNv12.data, 128, width * height);
    for (int y = objectY; y < objectY + objectHeight; ++y)
        for (int x = objectX; x < objectX + objectHeight; ++x)
            frameNv12.data[y * width + x] = 255;
    objectX += directionX;
    objectY += directionY;
    if (objectX >= width - objectwidth - 5 || objectX <= objectwidth + 5)
        directionX = -directionX;
    if (objectY >= height - objectHeight - 5 || objectY <= objectHeight + 5)
        directionY = -directionY;

    // Encode.
    if (!videoCodec->transcode(frameNv12, frameH264))
    {
        std::cout << "Can't encode frame" << std::endl;
        continue;
    }

    // write to file.
    fwrite(frameH264.data, frameH264.size, 1, outputFile);
}

// Close file.
fclose(outputFile);

return 1;
}

```

