VCodecJetPack C++ library

v2.0.0

Table of contents

Overview

VCodecJetPack C++ library provides hardware video **encoding** for H264, HEVC codecs for **Jetson** platforms. **VCodecJetPack** class inherits interface and data structures from open source **VCodec** library and also includes **Logger** open source library. **VCodecJetPack** uses Jetson Multimedia API. The library provides simple programming interface to be implemented in different C++ projects. 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 **Jetson Orin NX**:

codec / resolution	2560x1440	1920x1080	1280x720	640x512
H264	14.4 msec	8.6 msec	4.2 msec	2 msec
HEVC	13.8 msec	8.4 msec	4.2 msec	2 msec

Versions

Table 1 - Library versions.

Version	Release date	What's new
1.0.0	09.12.2022	First version.
1.0.1	10.04.2023	- support for non-standard resolution
2.0.0	19.09.2023	- Interface changes to VCodec. - Test application updated. - Added example application.

Library files

The VCodecJetPack 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 VCodecJetPack
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.
    VCodecJetPack.cpp ------------ Source code file of the library.
    VCodecJetPack.h -------------- Header file which includes VCodecJetPack class
declaration.
    VCodecJetPackVersion.h ------- Header file which includes version of the
library.
    VCodecJetPackVersion.h.in ---- CMake service file to generate version file.
```

VCodecJetPack class description

VCodecJetPack class declaration

VCodecJetPack class declared in **VCodecJetPack.h** file. Class declaration:

```
class VCodecJetPack : public VCodec
{
public:
```

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

getVersion method

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

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

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

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

Console output:

```
VCodecJetPack class version: 2.0.0
```

transcode method

transcode(...) method intended to encode video frame (**Frame** class). Video codec encodes 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 **HEVC H264** encoding.
dst	Result video frame (see **Frame** class description). To encode video data **dst** frame must have compressed pixel format (field **fourcc** of **Frame** class): **HEVC**, **H264**.

Returns: TRUE if frame was encoded 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 of the parameters 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 VCodecJetPack.
MAKE_KEY_FRAME	Not supported by VCodecJetPack.

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. According to this value, FPS and GOP size video codec calculate parameter for encoding.
QUALITY	read / write	Not supported.
FPS	read / write	FPS. According to this value, FPS and GOP size video codec calculate parameter for encoding.
GOP	read / write	GOP size (Period of key frames). 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.

Parameter	Access	Description
CUSTOM_2	read / write	Not supported.
CUSTOM_3	read / write	Not supported.

Build and connect to your project

Typical commands to build **VCodecJetPack** library:

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

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

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

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

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

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(${PARENT}_SUBMODULE_VCODEC_JET_PACK                 ON  CACHE BOOL "" FORCE)
if (${PARENT}_SUBMODULE_VCODEC_JET_PACK )
    SET(${PARENT}_VCODEC_JET_PACK                       ON  CACHE BOOL "" FORCE)
    SET(${PARENT}_VCODEC_JET_PACK_TEST                  OFF CACHE BOOL "" FORCE)
    SET(${PARENT}_VCODEC_JET_PACK_EXAMPLE               OFF CACHE BOOL "" FORCE)
endif()

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

File **3rdparty/CMakeLists.txt** adds folder **VCodecJetPack** 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
    VCodecJetPack
```

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

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

Done!

Installation on Jetson platforms

In general, when installing **JetPack** on your Jetson platform via Nvidia **SDK Manager**, there should be a **jetson_multimedia_api** folder in the **/usr/src** directory. If it is missing, you need to copy the **jetson_multimedia_api** folder from the /_static/jetson_multimedia_api.zip to the **/usr/src** directory on your Jetson.

Simple example

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

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

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

    // Set codec parameters.
    videoCodec->setParam(cr::video::VCodecParam::BITRATE_KBPS, 7500);
    videoCodec->setParam(cr::video::VCodecParam::GOP, 30);
    videoCodec->setParam(cr::video::VCodecParam::FPS, 30);

    // 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 HEVC frame.
    cr::video::Frame frameHEVC(width, height, cr::video::Fourcc::HEVC);

    // Create output file.
    FILE *outputFile = fopen("out.hevc", "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 (uint32_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, frameHEVC))
        {
            std::cout << "Can't encode frame" << std::endl;
            continue;
        }

        // Write to file.
        fwrite(frameHEVC.data, frameHEVC.size, 1, outputFile);
    }

    // Close file.
    fclose(outputFile);

    return 1;
}
```