VCodecImsdk C++ library

v4.0.0

Table of contents

Overview

VCodecImsdk C++ library provides hardware video **encoding/decoding** for H264, HEVC and JPEG codecs for **Intel HD Graphics**. **VCodecImsdk** class inherits interface and data structures from open source **VCodec** library. **VCodecImsdk** uses IntelMediaSDK API. The library provides simple programming interface to be implemented in different C++ projects. The library was written with C++11 standard. The libraries are supplied as source code only. The library is a CMake project.

Versions

Table 1 - Library versions.

Version	Release date	What's new
1.0.1	19.05.2022	First version.
2.0.0	27.06.2022	- new interface
3.0.0	27.01.2023	- Added JPEG encoding support. - Added test application .
4.0.0	15.09.2023	- Interface changes to VCodec. - Test application updated. - Added example application.

Library files

The VCodecImsdk 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.
    IMSDK_LINUX ------------------ Implimentation of IMSDK on Linux.
    IMSDK_WIN -------------------- Implimentation of IMSDK on Windows.
    VCodec ----------------------- Source code of VCodec library.
example -------------------------- Folder with simple example of VCodecImsdk 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.
    Backend ---------------------- Folder with bakend files.
    MediaSamples ----------------- Folder with Samples
    CMakeLists.txt --------------- CMake file of the library.
    VCodecImsdk.cpp -------------- Source code file of the library.
    VCodecImsdk.h ---------------- Header file which includes VCodecImsdk class
declaration.
    VCodecImsdkVersion.h --------- Header file which includes version of the library.
    VCodecImsdkVersion.h.in ------ CMake service file to generate version file.
```

VCodecImsdk class description

VCodecImsdk class declaration

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

```
class VCodecImsdk : public VCodec
{
public:
    /**
```

```
     * @brief Get library version.
     * @return String of current library version "Major.Minor.Patch".
     */
    static std::string getVersion();
    /**
     * @brief Class constructor.
     */
    VCodecImsdk();
    /**
     * @brief Class destructor.
     */
    ~VCodecImsdk();
    /**
     * @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/Decode video frame.
     * @param src Source RAW frame in NV12 format for Encoding
                  Source in HEVC / H264 / JPEG for Decoding
     * @param dst Result compressed frame (HEVC / H264 / JPEG) for Encoding
                  Result decoded frame in NV12 for Decoding.
     * @return TRUE if frame was encoded/decoded 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 **VCodecImsdk** class. Method declaration:

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

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

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

Console output:

```
VCodecImsdk class version: 4.0.0
```

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 HEVC H264 JPEG**. To decode video data **src** frame must have compressed pixel format (field **fourcc** of **Frame** class): **HEVC**, **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): **HEVC**, **JPEG**, **H264**. In contrary, src frame must be **NV12**

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 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 VCodecImsdk.
MAKE_KEY_FRAME	Not supported by VCodecImsdk.

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	Not supported.
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	Not supported.
TYPE	read / write	Not supported.

Parameter	Access	Description
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 **VCodecImsdk** library:

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

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

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

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

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

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(${PARENT}_SUBMODULE_VCODEC_IMSDK                    ON  CACHE BOOL "" FORCE)
if (${PARENT}_SUBMODULE_VCODEC_IMSDK)
    SET(${PARENT}_VCODEC_IMSDK                          ON  CACHE BOOL "" FORCE)
    SET(${PARENT}_VCODEC_IMSDK_TEST                     OFF CACHE BOOL "" FORCE)
    SET(${PARENT}_VCODEC_IMSDK_EXAMPLE                  OFF CACHE BOOL "" FORCE)
endif()

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

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

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

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

Done!

Installation on Linux

There are several steps to launching VCodecImsdk on Linux (tested on Ubuntu 22.04 LTS):

1. Install ubuntu with last updates:

   ```
   sudo apt-get update
   sudo apt-get upgrade
   sudo reboot
   ```

2. Install additional dependencies:

   ```
   sudo apt-get install wayland-protocols libx11-xcb-dev libxcb-present-dev libxcb-dri3-dev ffmpeg build-essential cmake git libopencv-dev vainfo v4l-utils net-tools openssh-client openssh-server ninja-build libmfx1 libmfx-tools libva-drm2 libva-x11-2 libva-wayland2 libva-glx2 libva-dev libmfx-dev autoconf libtool libdrm-dev xorg xorg-dev openbox libx11-dev libgl1-mesa-glx libgl1-mesa-dev
   ```

3. Install gmmlib:

   ```
   cd Downloads
   git clone https://github.com/intel/gmmlib.git
   cd gmmlib
   mkdir build
   cd build
   cmake -DCMAKE_BUILD_TYPE=Release ..
   make -j4
   sudo make install
   ```

4. Install intel media driver

```
cd Downloads
git clone https://github.com/intel/media-driver.git
cd media-driver
nano CMakeLists.txt
Replace default CMAKE_INSTALL_PREFIX to "/usr/lib/x86_64-linux-gnu" (insert line
"set(CMAKE_INSTALL_PREFIX "/usr/lib/x86_64-linux-gnu")" on 37 line )
Save changes: ctrl + S and then ctrl + X
cd ..
mkdir build_media
cd build_media
cmake -DCMAKE_BUILD_TYPE=Release ../media-driver
make -j4
sudo make install
```

5. Change driver according CPU version. **Additional information**

```
export LIBVA_DRIVER_NAME=iHD
sudo apt-get install intel-media-va-driver-non-free
```

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 "VCodecImsdk.h"

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

    // 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;
}
```