

## **VC-5 Test Materials: Install and Build Instructions**

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Restrictions</b>	<b>1</b>
<b>3</b>	<b>Software Prerequisites</b>	<b>1</b>
3.1	MacOS . . . . .	2
3.2	Windows . . . . .	2
3.3	Linux . . . . .	2
<b>4</b>	<b>Software Distribution</b>	<b>2</b>
<b>5</b>	<b>Amazon Credentials</b>	<b>3</b>
<b>6</b>	<b>Test Media</b>	<b>3</b>
<b>7</b>	<b>Documentation</b>	<b>4</b>
<b>8</b>	<b>Modifying the Source Code</b>	<b>4</b>
8.1	Key Parameters . . . . .	4
8.2	Implicitly Defined Parts . . . . .	4
<b>9</b>	<b>Building the Encoder and Decoder</b>	<b>4</b>
9.1	MacOS . . . . .	4
9.2	Linux . . . . .	5
9.3	Windows . . . . .	5
9.4	CMake . . . . .	5
9.5	Xcode . . . . .	6
<b>10</b>	<b>Testing the Software</b>	<b>6</b>
<b>11</b>	<b>Utility Programs</b>	<b>6</b>
11.1	Converter . . . . .	7
11.2	Comparer . . . . .	7
<b>12</b>	<b>Known Problems</b>	<b>7</b>
12.1	Too Many Open Files . . . . .	7
<b>13</b>	<b>Contact Information</b>	<b>7</b>

---

## 1 Introduction

The VC-5 test materials include a sample encoder, reference decoder, and utility programs and scripts to assist in testing the VC-5 codec. This document provides detailed information on installing, building, and running the software, including tools and libraries required to run the software and the method for downloading test media.

## 2 Restrictions

This release is provided for the sole purpose of evaluating the source code prior to publication of the VC-5 standard ST 2073-1:2013 by SMPTE and should not be used after the VC-5 test materials are available from SMPTE.

All source code and other materials included in this release is provided as is and is subject to the following:

(c) 2013-17 Society of Motion Picture & Television Engineers LLC and Woodman Labs, Inc. All rights reserved—use subject to compliance with end user license agreement.

## 3 Software Prerequisites

The following software tools are required to build and run the test materials:

### C compiler and linker

Required to build the sample encoder, reference decoder, and utility programs.

### Tcl

The codec test script is written in Tcl. Some Tcl distributions do not include the cmdline package by default. This package is required by the codec test script and is included in [Tcllib](#).

### Python 2.7

Various scripts such as the program for downloading media from Amazon S3 are written in Python.

### Git

Used to obtain the software and updates from the SMPTE Bitbucket repository.

### Make

Used to build the sample encoder, reference decoder, and utility programs.

### AWS Command Line Interface

Used to configure credentials that grant access to the VC-5 media bucket on Amazon S3.

Optional software:

### CMake

Used to create build scripts for other build tools besides make (the default tool).

### Doxygen

Used to generate the software documentation.

Software documentation created using Doxygen is provided in the software distribution, so Doxygen is only required to create new documentation if the distributed software is substantially modified.

### 3.1 MacOS

Xcode includes the compiler, linker, make, and git and can be downloaded for free from the Apple developer web site.

It is strongly recommended that Python, Doxygen, CMake and other command-line tools be installed using **Homebrew** whenever the software is available through Homebrew. Tcl can be installed using the free community distribution provided by ActiveState: **ActiveTcl**.

The Python packages by this software distribution are listed in the `requirements.txt` file included in this distribution. Use the following command to install all Python packages required by this software distribution:

```
pip install -r requirements.txt
```

### 3.2 Windows

Visual Studio Community is available for free from Microsoft.

Tcl and Python can be installed using the community editions provided by ActiveState: **ActiveTcl** and **ActivePython**. These are full-featured distributions, so it should not be necessary to install additional Python or Tcl packages.

CMake can be installed from the web site: <https://cmake.org/download/>

### 3.3 Linux

Build tools can be installed using the preferred package manager for the specific Linux distribution.

## 4 Software Distribution

The source code distribution includes the following files and directory structure:

**common/**

Source code that is common to both the sample encoder and reference decoder.

**encoder/**

Source code for the sample encoder.

**decoder/**

Source code for the reference decoder.

**tables/**

Codebook used by the sample encoder and reference decoder.

**external/**

Software developed by third parties that is used by the test materials.

**converter/**

Source code for a C language program for converting between image file formats.

**comparer/**

Source code for a C language program that can be used to compare the decoded image with the image that was input to the encoder. The comparer program is not currently used by the test scripts.

**scripts/**

Scripts written in Tcl for testing the sample encoder and reference decoder.

**media/**

Sample images and reference bitstreams that are compliant with the VC-5 suite of standards.

---

**Makefile**

Make file for building the software, including targets to download media and run the comprehensive test suite.

**readme.pdf , readme.html**

Instructions for building the source code provided with the test materials.

**release.pdf , release.html**

Release notes listing the changes in each version of the test materials.

In this document, the variable `ROOT` is the location of the installed software with the directory structure as describe above.

The external sub-directory redistributes the third-party code for `getopt`, which is used for parsing the command-line arguments, and Standard C headers files for Boolean and integer data types that are omitted on some platforms. The third-party software is only needed for building the codec on Windows since `getopt` and all Standard C header files are available on macOS and Linux.

## 5 Amazon Credentials

The test media files are stored on Amazon S3 and downloaded using a script (see Section 6).

Credentials are required to access the media files in the VC-5 bucket on Amazon S3. A guest user account has been created on the Amazon account that hosts the VC-5 media bucket. The guest user has read-only access sufficient to allow the media to be downloaded from S3.

The following steps are required to obtain access to the VC-5 media bucket:

1. Install the AWS Command Line Interface: <http://docs.aws.amazon.com/cli/latest/userguide/installing.html>
2. Email the VC-5 Amazon account [administrator](#) to obtain the guest credentials. The administrator will reply with a CSV file that contains the credentials.
3. Run the following command:

```
aws configure --profile vc5codec-guest
AWS Access Key ID [None]: <Access key ID>
AWS Secret Access Key [None]: <Secret access key>
Default region name [None]:
Default output format [None]:
```

Replace `<Access key ID>` and `<Secret access key>` with the corresponding fields from the CSV file provided by the VC-5 Amazon account administrator.

This command will create a directory tree in the user home directory granting guest access to the VC-5 media bucket. It is important to install the credentials for the `vc5codec-guest` profile because the script for downloading media files uses this profile by name to access Amazon S3.

For further information about credentials for Amazon Web Services see the web page on [credentials](#).

## 6 Test Media

The test images and bitstreams used for conformance testing are not included in the software distribution, but must be downloaded separately from an Amazon S3 bucket. A script is provided in the software distribution to handle all of the details for downloading test media. The script checks whether a file exists on the local hard drive already and if the checksums of the local and remote files are identical, then the file is not downloaded again.

To run the script for downloading media, the user must obtain Amazon credentials from the VC-5 Amazon account administrator (see Section 5).

To install the test media, invoke the following command from the root directory of the software distribution:

```
make media
```

## 7 Documentation

Doxygen can be used to create documentation for the encoder and decoder by running the command

```
make docs
```

in the root directory of the software installation.

The documentation is in HTML format and is located in the `$(ROOT)/encoder/docs/html` and `$(ROOT)/decoder/docs/html` sub-directories. The root of the document tree is `index.html` in the respective sub-directories.

## 8 Modifying the Source Code

### 8.1 Key Parameters

Key compile-time parameters are in the file `$(ROOT)/common/include/config.h`, including:

- Limits that control the maximum sizes of data structures
- VC-5 parts that are enabled at compile-time by default

### 8.2 Implicitly Defined Parts

Some parts of the VC-5 standard are enabled implicitly if other parts are enabled. For example, VC-5 Part 3 Image Formats is enabled if VC-5 Part 6 Sections is enabled. This behavior can be modified by changing the table `enabled_parts_list` in the routine `CheckEnabledParts()` in the file `$(ROOT)/common/src/utilities.c` in the codec software distribution.

## 9 Building the Encoder and Decoder

The encoder and decoder have been built and tested on macOS 10.12.4 (16E195) using make files, CMake version 3.7.2, and Xcode version 8.3.2 (8E2002).

### 9.1 MacOS

The Xcode projects are not currently included in the software distribution. The CMake build script can be used to create Xcode projects for building the encoder and decoder. See Section 9.5.

The encoder and decoder can be built executing either of the following commands:

```
make TOOL=make
```

or

```
make TOOL=cmake
```

in the root directory of the installed software.

It may be necessary to run `cmake` explicitly to generate build files the first time that `cmake` is used. For example, to run `cmake` for preparing to build the encoder:

```
cd $(ROOT)/encoder/build/cmake  
cmake .
```

The encoder build files are placed in the make or cmake subdirectories of  $\$(ROOT)/encoder/build$ , depending on which tool was used to build the executable. The decoder build files are in the make or cmake subdirectories of  $\$(ROOT)/decoder/build$ , depending on which tool was used to build the executable. The CMake tool only builds the release configuration of the encoder or decoder, the make tool builds both debug and release configurations.

The default tool is make, so the encoder and decoder can be built using the simpler command:

```
make
```

with no command-line arguments required.

Build products can be removed by executing the following command in the root directory:

```
make clean
```

or

```
make clean-all
```

to remove all build products.

The command

```
make TOOL=cmake clean
```

will remove the build products created by running the make files created by CMake.

The command

```
make TOOL=cmake clean-all
```

will remove the build products created by running the make files created by CMake and all of the build files created by running CMake.

## 9.2 Linux

Although Linux is not used routinely for building and testing the software, either of the methods described above (make or cmake) should work on Linux and other Unix systems.

## 9.3 Windows

The codec has not been developed and tested under Windows in some time, but it should be possible to build the software using Visual Studio solution and project files created by CMake (see Section 9.4).

## 9.4 CMake

CMake has been used to build the encoder and decoder using make (the default CMake generator).

It is expected that CMake will correctly generate build files on all platforms and for all build tools supported by CMake such as Xcode, Visual Studio, and Eclipse, but not all platforms and tools have been tested. Submit a bug report for any combination of platform or build tool that CMake should support, but the build fails (see Section 13).

In the future, CMake will be used to create build projects for various tools and platforms. Project files for specific platforms and tools (for example, Visual Studio project and solution files) will not be included in the software distribution.

The recommended practice is to create a sub-directory in the cmake directory for a specific build tool and create build files in that directory. For example, to create Xcode projects for building the encoder:

```
cd $(ROOT)/encoder/build/cmake
mkdir xcode
cd xcode
cmake -G Xcode ..
```

See the CMake [documentation](#) for information about determining the generators that are available on a specific platform.

## 9.5 Xcode

Xcode projects for the encoder and decoder can be created for macOS using CMake. For example, to create an Xcode project for the decoder:

```
cd $(ROOT)/decoder/build
mkdir xcode
cd xcode
cp ../cmake/CMakeLists.txt .
cmake -G Xcode .
```

This will create an Xcode project that can be used to build and debug the decoder.

## 10 Testing the Software

The encoder and decoder can be invoked from the command line with the arguments described in VC-5 Part 2, but it may be more convenient to use the `testcodec.tcl` test script that is included in the distribution.

As described in VC-5 Part 2, the codec test script relies on a specific directory structure and filename conventions.

All regression tests can be performed by executing the command

```
testcodec.tcl -a
```

in the directory `$(ROOT)/scripts`. By default, the codec test script uses the release configuration of the encoder and decoder built using the make tool.

For every image file in the `$(ROOT)/media` directory tree, the image file is encoded into a bitstream which is then decoded into an image file. The bitstream and decoded image are compared against the corresponding master files using a simple binary file comparison. Any difference in file size or content is reported as a failure.

The codec test script accepts several command-line arguments that use different build products or limit testing to a specific part of the VC-5 standards. For example, the command

```
testcodec.tcl -p 3 -b debug -v
```

runs only the tests for VC-5 part 3 using the debug configuration of the encoder and decoder built using the make tool and prints information about the tests to the terminal window.

The command

```
testcodec.tcl -p 5 -c cmake -l testcodec.log -v
```

runs only tests for VC-5 part 5 using the release configuration of the encoder and decoder built using the cmake build tool and outputs information to both the terminal window and a logfile.

For convenience, the comprehensive codec test script can be run from the root directory using make:

```
make test
```

## 11 Utility Programs

The converter and comparer utility programs, described in VC-5 Part 2, are included in the software distribution, but have not been used or tested in some time. It is expected that the utility programs will still work, as the code has not been changed in some time.

In the future, CMake scripts will be provided to allow the utility programs to be built on any platform and enable creation of projects for specific tools such as Xcode and Visual Studio, for example.



## 11.1 Converter

The converter program was used to create image files in byr4 pixel format from DPX files.

The converter can be built using make files or CMake. By default, CMake creates a project that uses the make tool, but CMake can be used to create other types of projects. For example, to create an Xcode project for the converter:

```
cd $(ROOT)/converter/build
mkdir xcode
cd xcode
cp ../cmake/CMakeLists.txt .
cmake -G Xcode .
```

## 11.2 Comparer

The comparer program was intended to be used to compare decoded images with the images input to the encoder, but this functionality has been superseded by the codec test script that performs a binary compare of encoded bitstreams and decoded images with master files that have been checked for correctness.

The converter program was used to create image files in byr4 pixel format from DPX files.

The converter can be built using make files or CMake. By default, CMake creates a project that uses the make tool, but CMake can be used to create other types of projects. For example, to create an Xcode project for the comparer:

```
cd $(ROOT)/comparer/build
mkdir xcode
cd xcode
cp ../cmake/CMakeLists.txt .
cmake -G Xcode .
```

## 12 Known Problems

### 12.1 Too Many Open Files

The encoder and decoder are passing all regression tests performed by running:

```
testcodec.tcl -a
```

but the user may see an error message about too many pipes. To solve this problem, increase the limit on the number of open files:

```
ulimit -n 2048
```

## 13 Contact Information

Please report bugs or suggestions for improvements by sending email to [bugs@vc5codec.org](mailto:bugs@vc5codec.org).

Include in the bug report step by step instructions for reproducing the problem. Describe the expected behavior and the behavior that was observed.

---