AN13725

Run openCV on Cortex-M7 MCU

Rev. 1 — 15 June 2023 Application note

Document Information

Information	Content
Keywords	OpenCV, MCU
Abstract	This document describes how to build an openCV library with GCC/ARMClang and run it on our RT-series MCU platform, such as, i.MX RT1170 EVKB board.

1 Introduction

OpenCV (Open Source Computer Vision Library: http://opencv.org) is an open source library that includes hundreds of computer vision algorithms.

OpenCV has a modular structure, which means that the package includes several shared or static libraries. The following modules are available: Core functionality, Image Processing, Video Analysis, Camera Calibration, 3D reconstruction (`calib3d`), 2D Features Framework (`features2D`), Object Detection (`objdetect`), High-level GUI (`highgui`), and Video I/O (`videoio`).

Although we saw so many OpenCV examples running on PC or other MPU platform, we hardly find one for MCU platform. This document describes how to build an openCV library with GCC/ARMClang and run it on our RT-Series MCU platform, such as, i.MX RT1170 EVKB board.

2 Environment setup

Before heading the configuration, download and install some additional libraries and resources required for this application note. Make sure that Windows system is installed with MCUXpresso IDE (latest version) and a serial terminal emulator (TeraTerm, Putty, or others). They allow us to evaluate the generated OpenCV library. Also a Git installed, which help us to get the source code of openCV from github. This application note assumes that the user has basic knowledge of Linux commands.

Following are the steps of the setup on Windows 10:

1. Download and install CMake for windows from https://github.com/opencv/opencv.git
   Note: *Make sure that the version of the CMake must be 3.10.2!*
   Choose a folder to store the code and run below line from a git-console:

```
git clone --depth=1 https://github.com/opencv/opencv.git
```

2. Download and Install CMake for windows from:
   https://cmake.org/files/v3.10/cmake-3.10.2-win64-x64.zip
   Unzip the file to your path, assuming that the path you place the CMake is:

```
your_path/cmake-3.10.2-win64-x64
```

3. We need **Make** to build the code, so download and install the MinGW which has **Make** inside from:
   https://sourceforge.net/projects/mingw/files/latest/download
   Install the MinGW to your PC, follow the tips, and assume that the path is:

```
your_path/MinGW
```

4. To choose the tool-chain to compile the code, we provide two selections: **ARMClang** from Keil and **arm-none-eabi-gcc** from the installed MCUXpresso. Make sure that you have either IDE installed in your PC. We take the **arm-none-eabi-gcc** as an example to show the flow. But it is easy to do the same thing with ARMClang, which is also mentioned later.
5. From https://gitee.com/crist_xu/opencv_mcu, find below helper files:

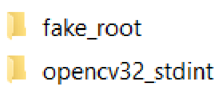


Figure 1. File location

   Inside the **fake_root**, there is some header and toolchain files. They are used to configure the toolchain for OpenCV. Inside the **opencv32_stdint**, there is a header. It redefines the int-type, which is necessary.

3 Configure the OpenCV module

Make sure that you have done all the preparations before you start here.

1. From where you store the OpenCV, make a new folder name build, for example. Then enter the folder and open a PowerShell window here. Type:

```
your_path/cmake-3.10.2-win64-x64\bin\cmake-gui.exe ..
```

   Open the CMake.

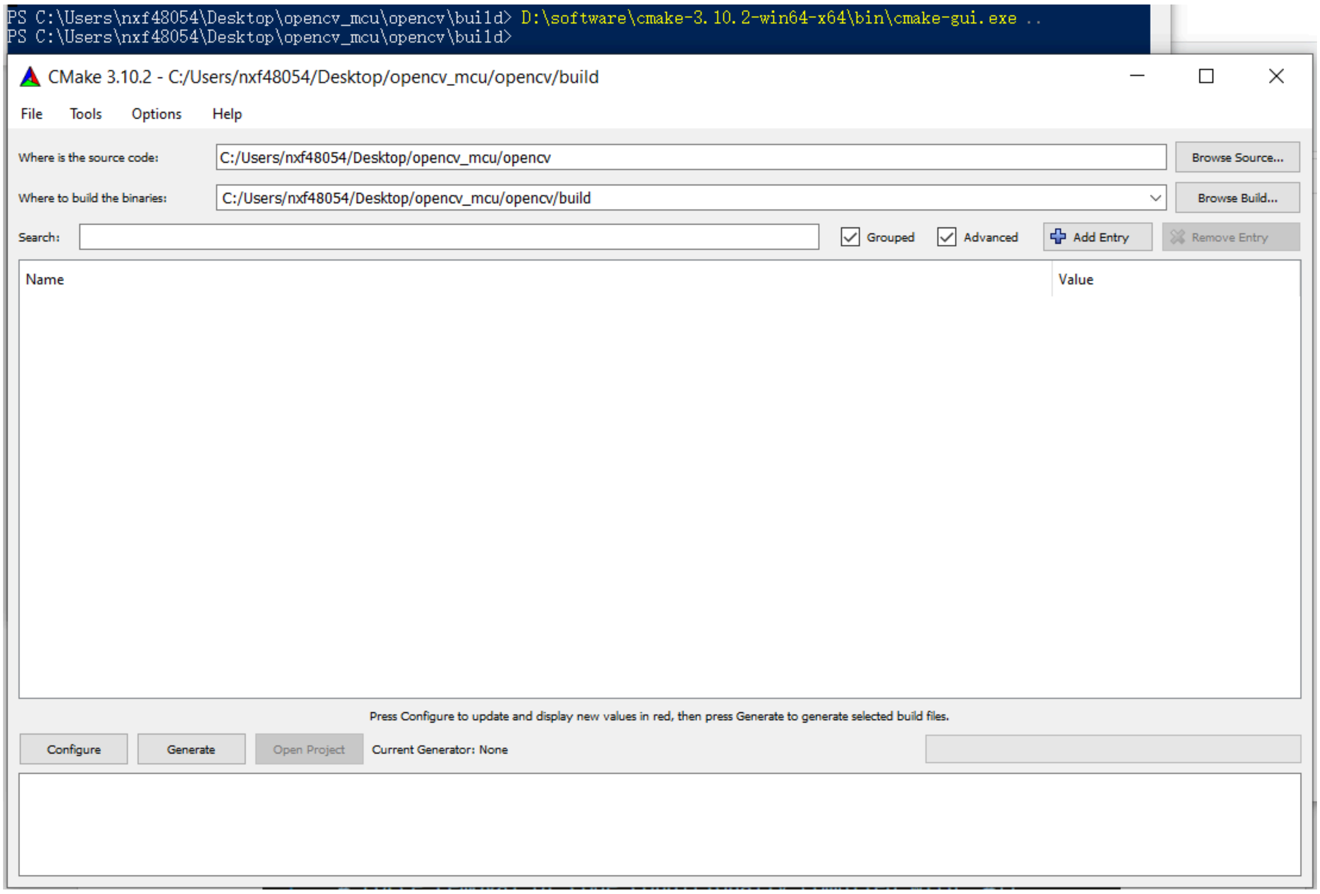


Figure 2. Open Cmake

   Start with the empty project, and click **Configure**.

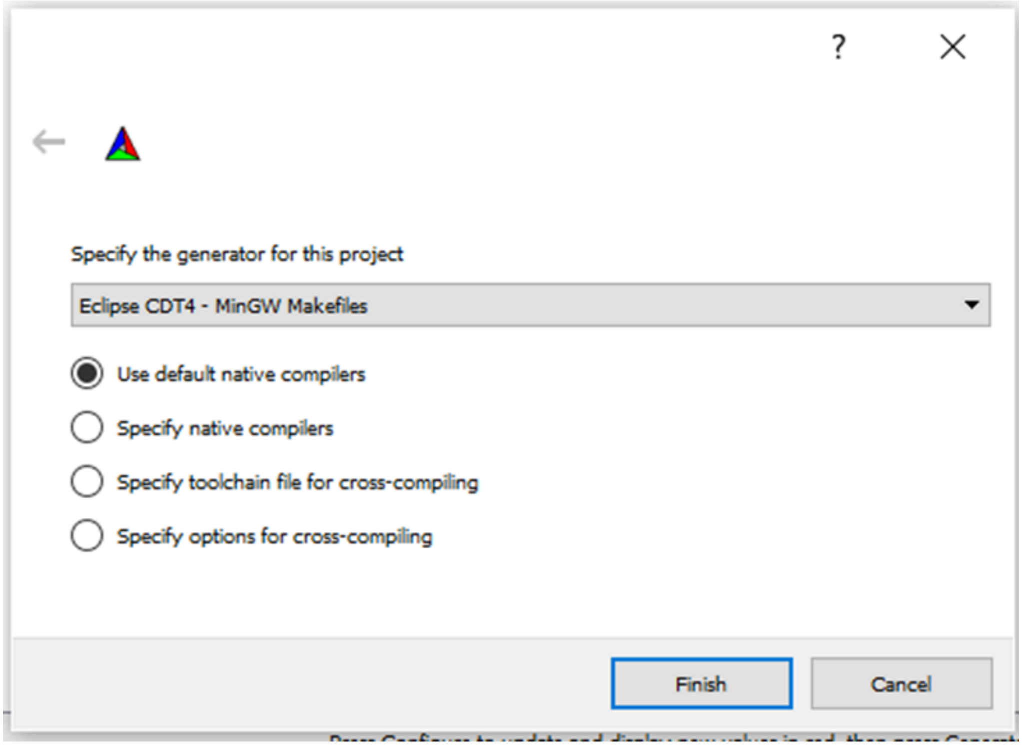


Figure 3. Select Eclipse CDT4 – MinGW Makefiles

Here we can specify the generator. To build the code with the MCUXpresso IDE, choose **Eclipse CDT4 – MinGW Makefiles**. An Eclipse project is generated, allowing you to build the code through the MCUXpresso IDE.

But at the first time, we recommend that you choose the below option with the specific toolchain, the **MinGW + specific toolchain**.

Figure 4 selects **MinGW Makefiles** generator as an example. For other generators, the results are the same. Then check **Specify toolchain file for cross-compiling** and click **Next**.

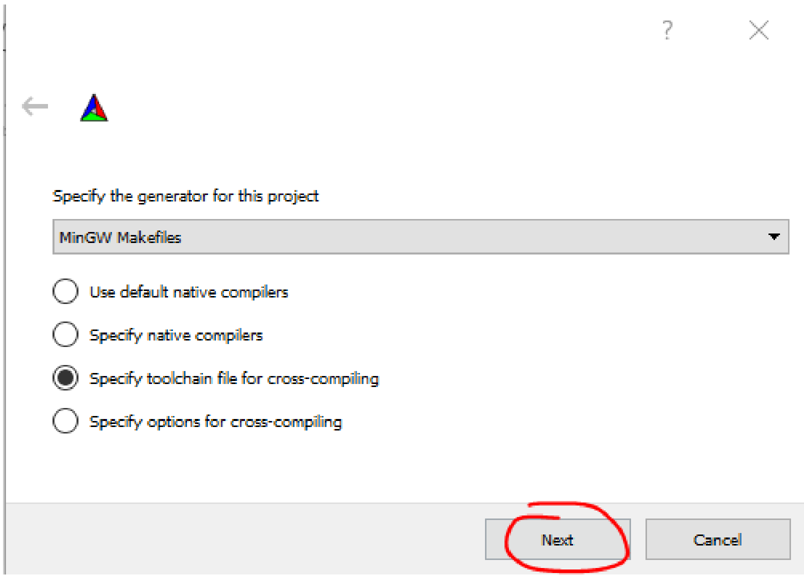


Figure 4. Select MinGW Makefiles

Click **…** and browse your PC to find the toolchain file where you unzip the attachment. Before that operation, some modifications are required. To use the ARMCLANG, edit **armgcc.cmake** under the **fake_root**.

```
# helper folder
SET(HELPER_FILE_PATH "")

# TOOLCHAIN_DIR AND NANO LIBRARY
SET(TOOLCHAIN_DIR "C:/Keil_v5/ARM")
```

Figure 5. Edit armgcc.cmake

The **HELPER_FILE_PATH** to your unzip location, and the **TOOLCHAIN_DIR** is where you install the Keil. And if GCC, edit the gcc.cmake under the fake_root.

```
# helper folder
SET(HELPER_FILE_PATH "")

# TOOLCHAIN_DIR AND NANO LIBRARY
SET(TOOLCHAIN_DIR "C:/NXP/MCUXpressoIDE_11.5.0_7232/ide/plugins/com.nxp.mcuxpresso.tools.win32_11.5.0.202107051138/tools")
```

Figure 6. Edit gcc.cmake

The **HELPER_FILE_PATH** has the same meaning, and the **TOOLCHAIN_DIR** is a path related to the MCUXpresso IDE.

After modifying the toolchain files, choose GCC as the default tool-chain and click **Continue**.

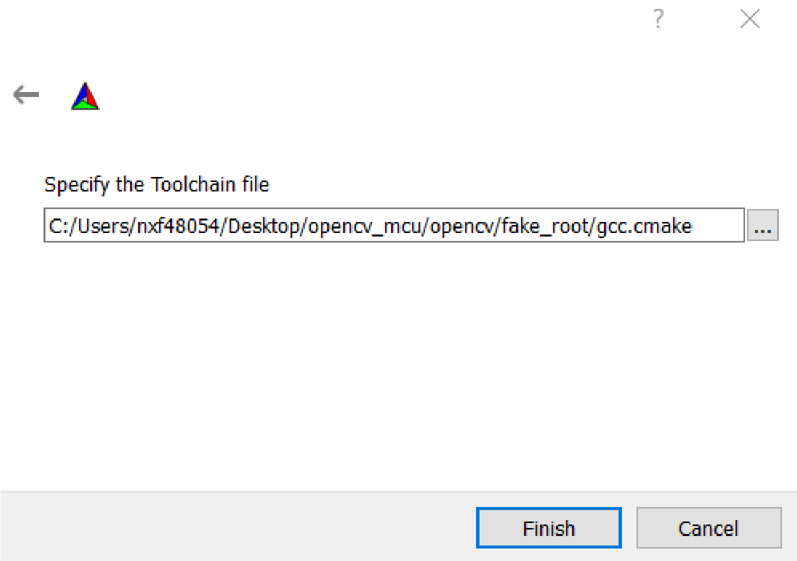


Figure 7. Select GCC

Click **Finish**.

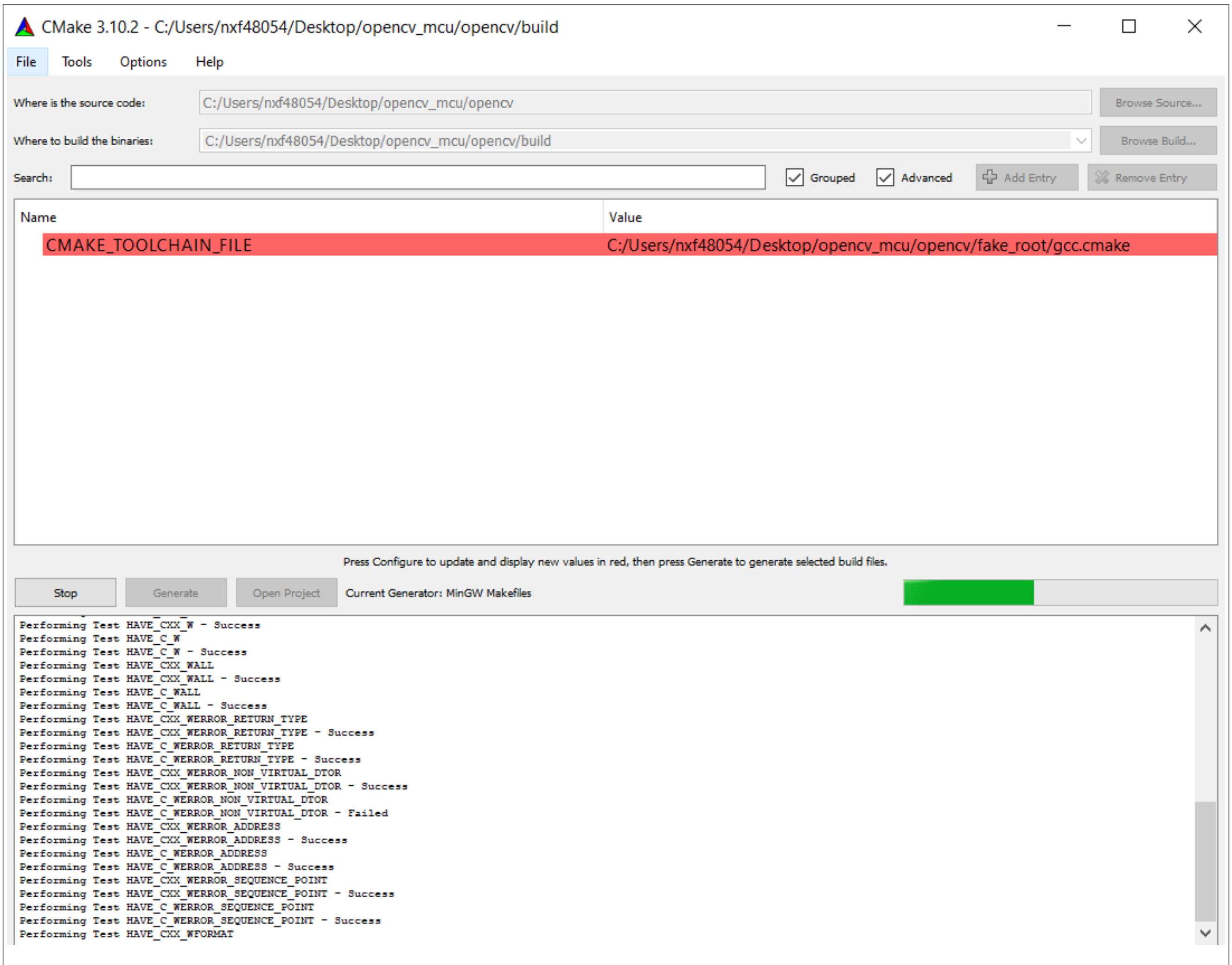


Figure 8. Click Finish

Wait for the configuration done. All the new-options are in Red.

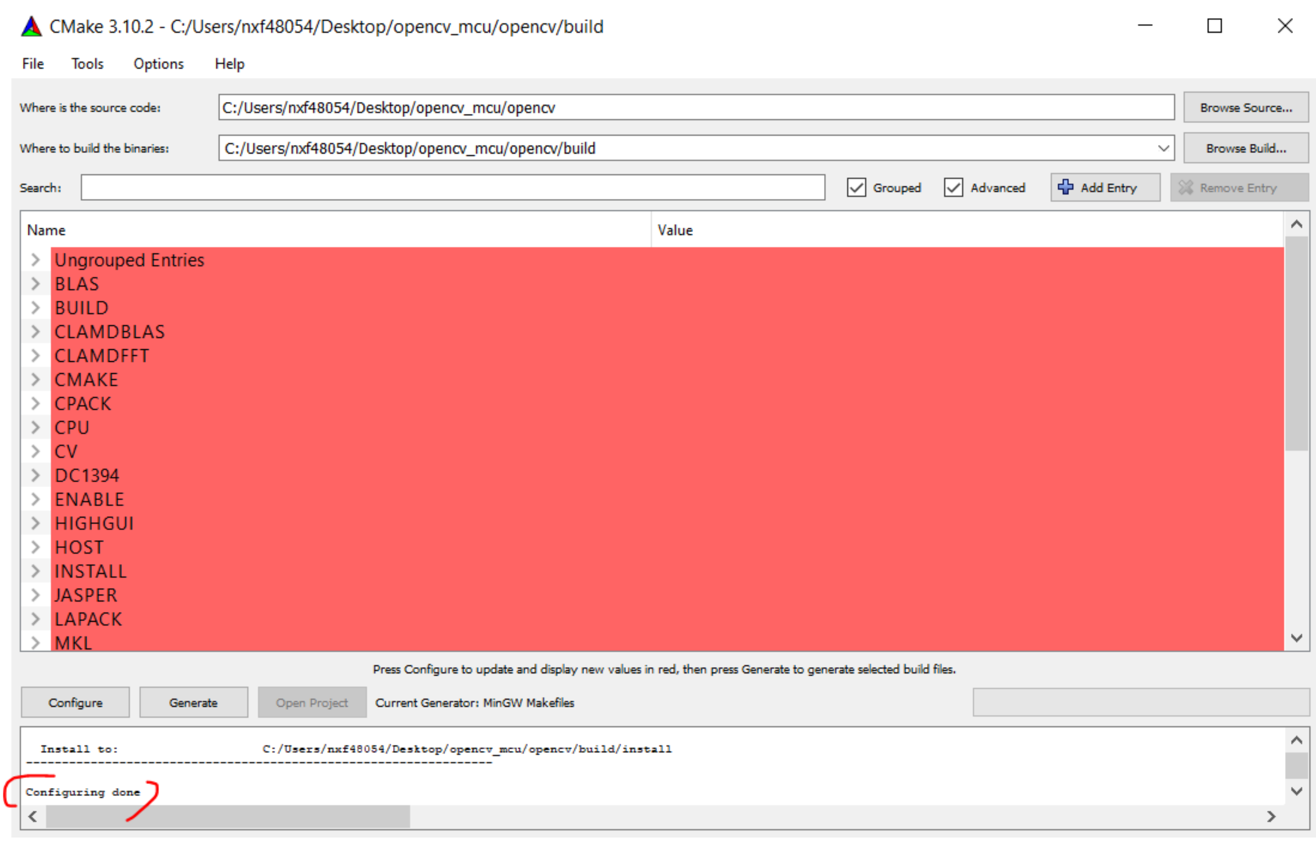


Figure 9. Configuration done

2. Specify the module. It is better to check **Grouped** and **Advanced**. Organize the module in order. It is easy to find what we need or necessary for an MCU platform.

Table 1. Specify modules

Groups	Options
Ungrouped Entries	Unchecked all
BLAS	Skipped
BUILD	Checked: `BUILD_JPEG`, `BUILD_OPENJPEG`, `BUILD_PNG`, `BUILD_opencv_core`, `BUILD_opencv_features2d`, `BUILD_opencv_highgui`, `BUILD_opencv_imgcodecs`, `BUILD_opencv_imgproc`, `BUILD_opencv_video`, `BUILD_opencv_world`
CLAMDBLAS	Skipped
CLAMDFT	Skipped
CMAKE	Skipped, but can change the `CMAKE_BUILD_TYPE` (Debug/Release) if you want
CPACK	Unchecked all
CPU	Skipped
CV	Unchecked all
DC1391	Skipped
ENABLE	Unchecked all except the `ENABLE_FAST_MATH`

Table 1. Specify modules*...continued*

Groups	Options
HIGHGUI	Unchecked
HOST	Skipped
INSTALL	Skipped
OPENCV	Checked: `OPENCV_DISABLE_FILESYSTEM_SUPPORT`, `OPENCV_DISABLE_THREAD_SUPPORT`, `OPENCV_ENABLE_MEMALIGN`
OPJ	Unchecked all
OpenBLAS	Skipped
PNG	Skipped
PYTHON2/PYTHON3	Skipped
VIDEOIO	Unchecked all
WITH	Checked: `WITH_OPENJPEG`, `WITH_PNG`
ZLIB	Skipped

Click **Configure** again and then **Generate**.

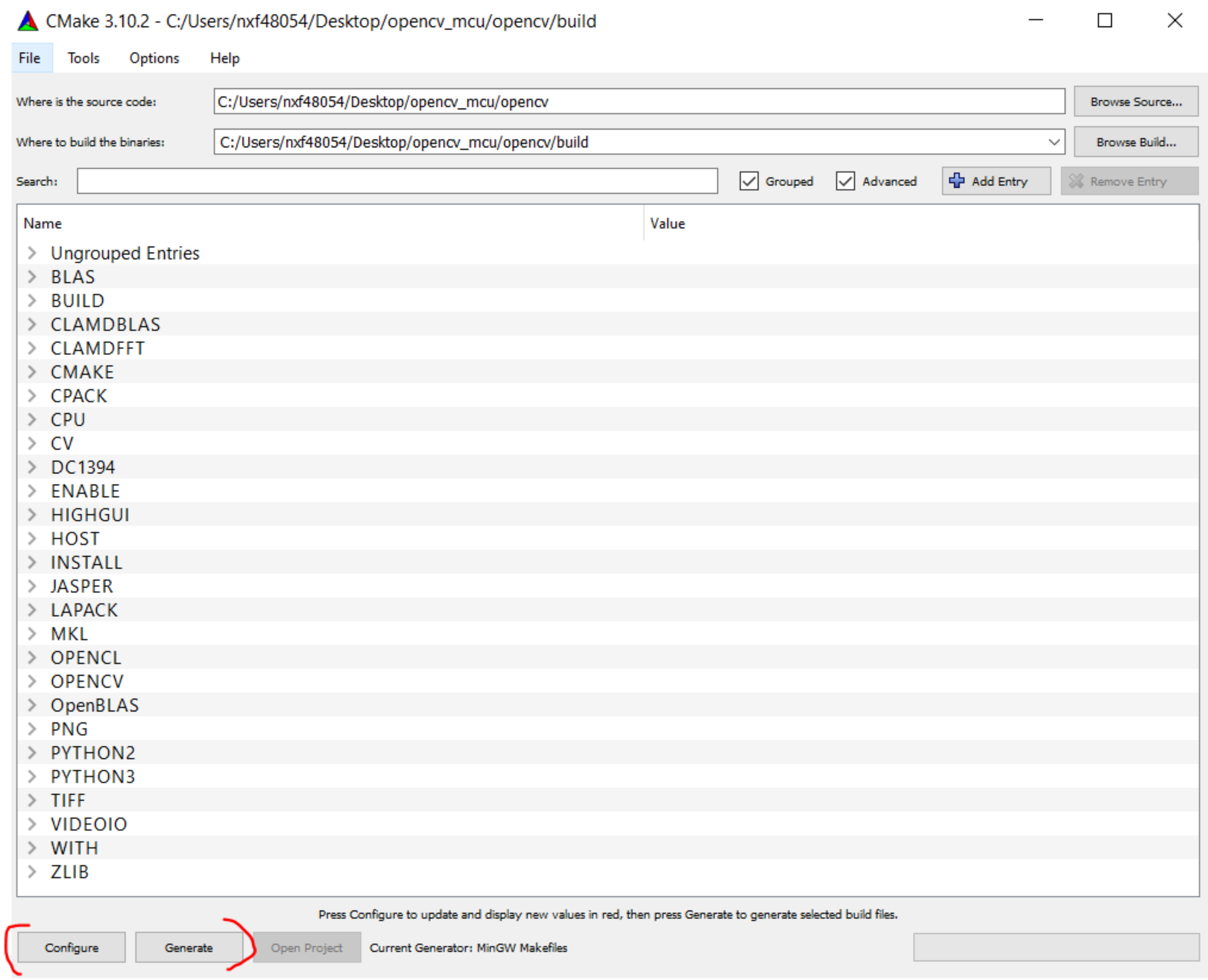


Figure 10. Select Generate

3. Compile the code. From the build folder, open the **PowerShell** window again, and run **Make**.

```
your_path/MinGW/bin/mingw32-make.exe -j7
```

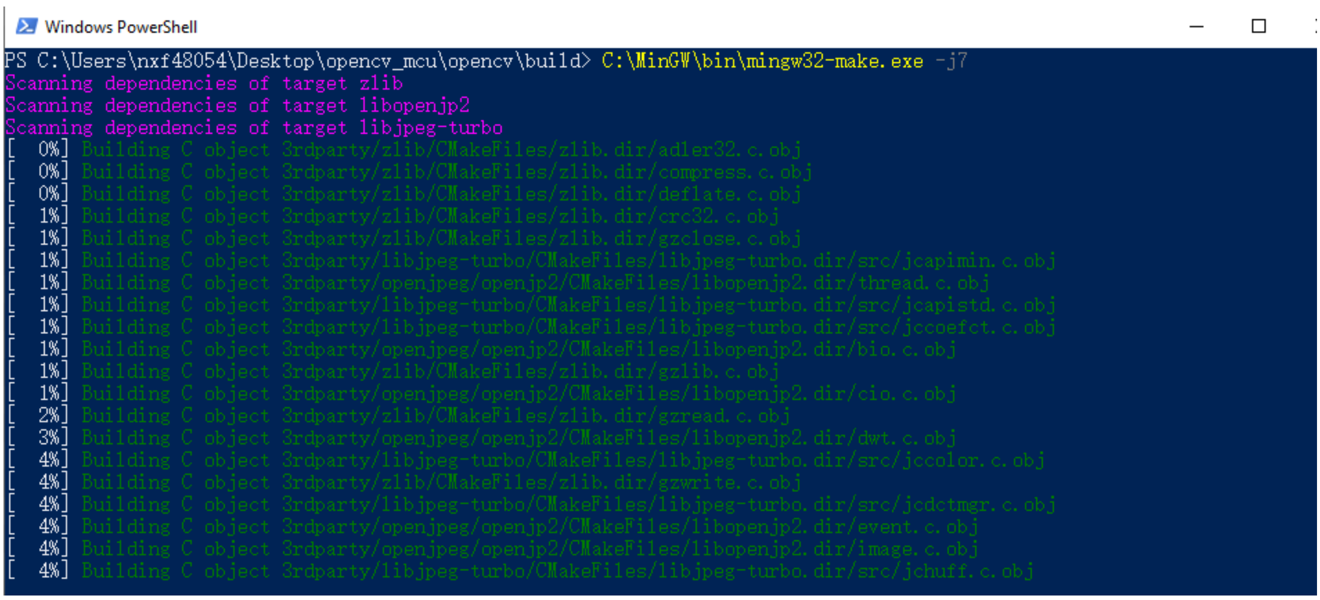


Figure 11. PowerShell window

4. Finally, we got five libraries, `libopencv_world.a` under build/lib and other four libraries for image encode/decode.

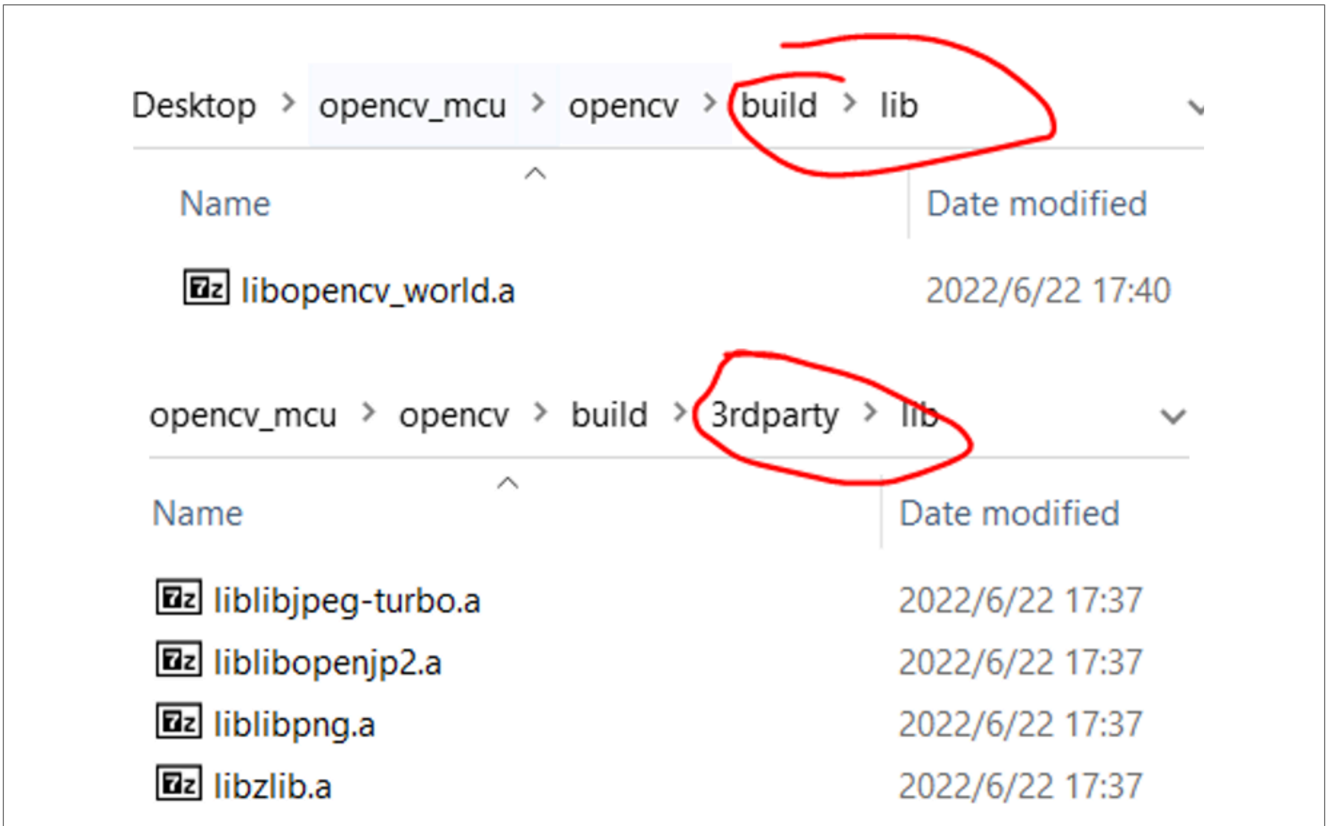


Figure 12. Five libraries

4 Deploy an OpenCV example on MIMXRT1170 EVK

Next step is to deploy the library on the board. We make an example to show that the library can work on our board. The case is to decode a `jpg` image and resize it. In MCUXpresso IDE, import the SDK example project `Hello World` into the workplace and take it as the template to build the case (this document does not focus on how to create the project). You can find the project from the attachment. Copy all the libraries to the `source/library` folder within the project and copy test image into `source/data` at the same time.

Choose one picture, such as, the famous Lena. It is a 500*500 rgb24 jpg picture. Let us decode and resize it to (320, 240).

Change the *image_data.s* file as below. Declare two external symbols tell the code where we can find that:

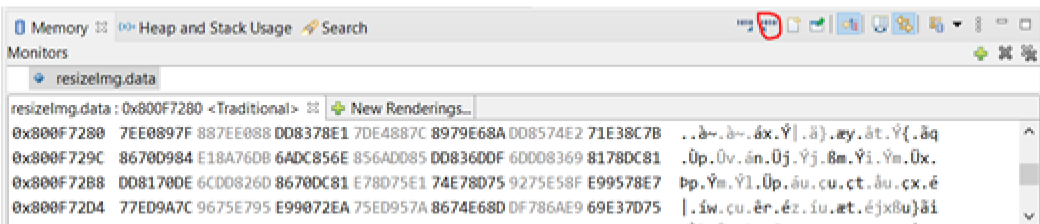
```
     .global img_start
     .global img_end

img_start:
   .incbin "data/lena.jpg"
img_end:
```

The main cv-code is here:

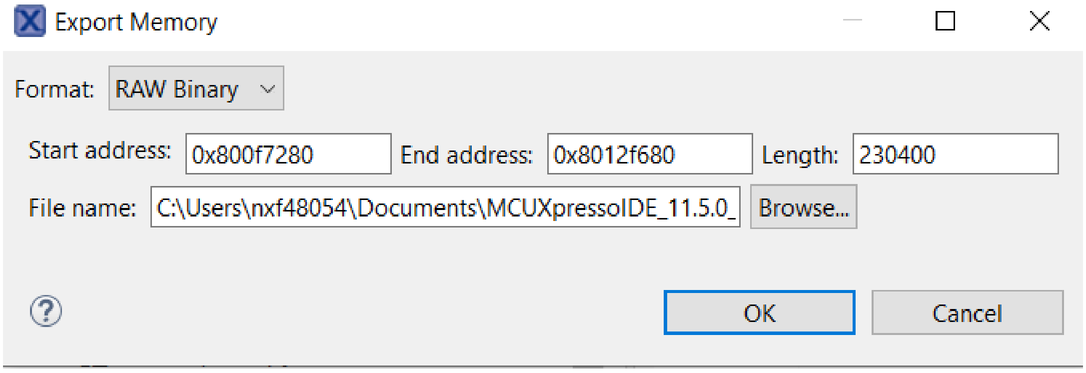
```
// decode the image
std::vectorr<char> data(img_start, img_start + IMG_LEN);
cv::Mat img_encode(data);
cv::Mat img = cv::imdecode(img_encode, cv::IMREAD_UNCHANGED);

// resize the decoded image
cv::Mat resizeImg;

cv::resize(img, resizeImg, cv::Size(320, 240), cv::INTER_LINEAR);
```

After building and downloading the project into the board, we can see `Decode & Resize Done!` from the serial-console. To check whether the result is correct, download the data to our PC:

1. Add a memory window and typing: `resizeImg.data`.



2. Click **Export**, input the Start address and End address, and the IDE calculates the **Length** then. Choose the Format as RAW Binary, and browse your PC to choose a place.

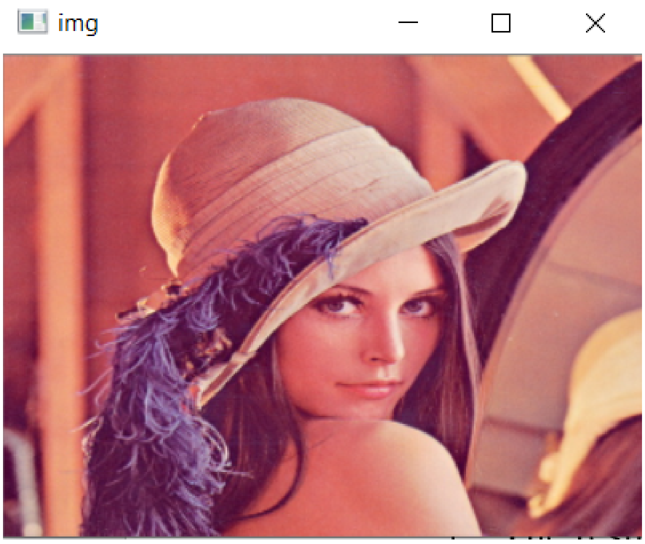


3. To download the data, click **OK**.
4. To check the data, make sure that there is a tool which can help review the image. If not, try below code snippet:

```
import numpy as np
import cv2 as cv
img_raw_data = np.fromfile("your_path/your_file_name.bin", dtype="uint8")
img = np.reshape(img_raw_data, (240, 320, 3)) # the new shape
cv.imshow("img", img)
cv.waitKey(0)
```

Note: *Assume that you have python in your PC and also have the **numpy** and the **opencv** library installed.*

5. The result is as below.



5 Reference

The file mentioned in this application note is shipped in the attachments.

6 Revision history

Table 2 summarizes the revisions to this document.

Table 2. Revision history

Revision number	Date	Substantive changes
1	15 June 2023	• Updated the link in Section 2 • Added Note About the Source Code in the Document
0	23 September 2022	Initial release

7 Legal information

7.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

7.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. - NXP B.V. is not an operating company and it does not distribute or sell products.

7.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents

Date of release: 15 June 2023
Document identifier: AN13725