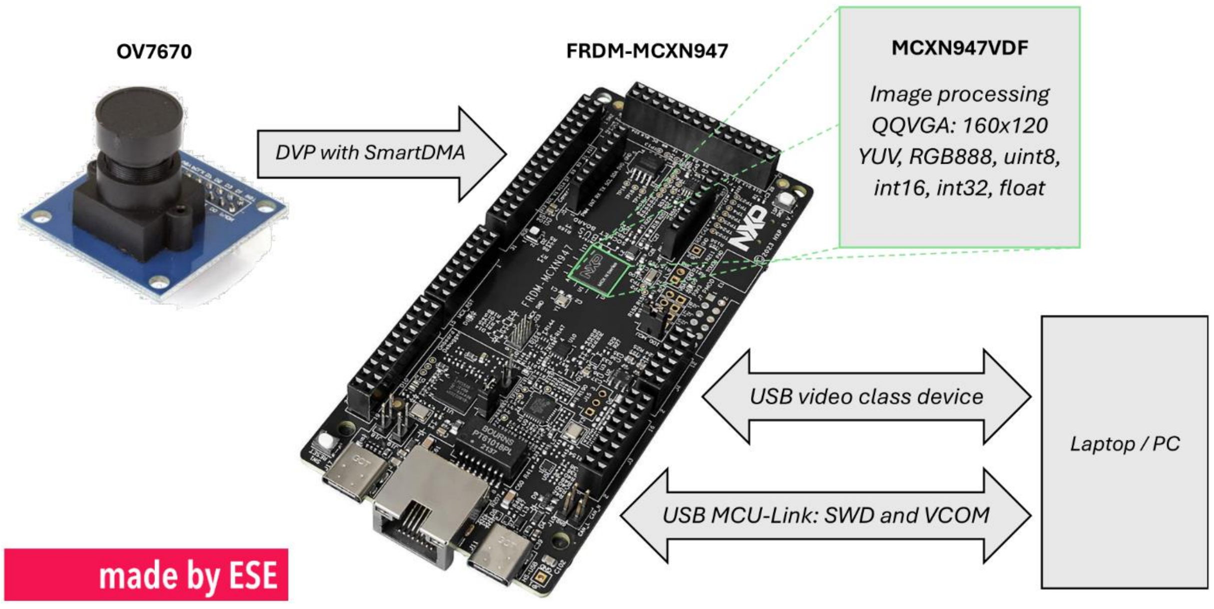
EVD1 Study guide

Image processing for microcontrollers



By Hugo Arends

Embedded Vision and Machine Learning

HAN Embedded Systems Engineering

Contents

Revisions

Version	When	Who	What
1.0	Nov. 2024	Hugo Arends	First version of the document.
1.1	Nov. 2024	Hugo Arends	Reworked week 1 – chapter 4

Introduction

This document is the study guide for the second period of the EVD1 course. The EVD1 course is part of the Embedded Vision and Machine Learning module taught as part of the HAN Embedded Systems Engineering program. The main goal of this module is to learn how to implement and use image processing operators on a microcontroller. The course will therefor discuss how to implement image processing operators in de C programming language with special attention for performance considerations.

This document comes with the following other resources:

- Example images
- Source files with template functions for image processing operators
- Powerpoint presentations
- MCUXpresso workspace with one example project
- Qt Creator workspace with several example projects

These resources are made available online.

The assignments for each week are described in this document. The theory is discussed in class by means of powerpoint presentations. In the first week we will install the software development environment, configure the hardware, test the installation and get familiar with writing and testing code. In the second week performance enhancement of code execution is discussed. The third week discusses image enhancement by means of filtering. The fourth week discusses algorithms for automatic thresholding for separating objects from the background. The fifth week reduces binary objects as much as possible to their basic shapes. And finally, the sixth week discusses techniques to extract object features. This document starts, however, with a description of the final assignment.

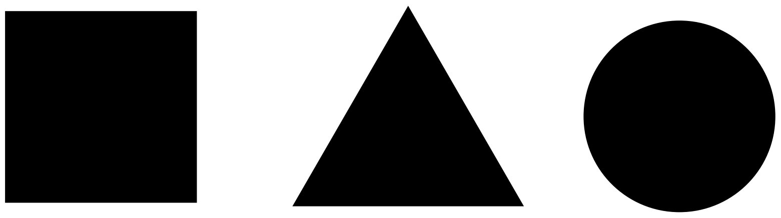
This course is organized as a workshop. This means that each class will start with a presentation of the topics for that particular week. There will also be time to work on the exercises as described in this document and ask questions.

Have fun!
Hugo

Final assignment

In the second period, various image processing operators are implemented including a unique operator by each student. During an individual oral assessment, the student's competence in designing, implementing and testing image processing operators for embedded devices is assessed.

During the assessment, the embedded device correctly classifies the following three shapes. All other shapes are classified as 'unknown'.



The visualization of the classification is up to the student. Examples are colour overlays in the camera image, coloured bounding boxes, text on the image, LEDs on the development kit, or a combination of these examples.

To accomplish this goal, the student implements a set of mandatory image processing operators. These operators, and other operators which will be made available, can be used to fulfil the task.

The solution has an expected framerate of 30 fps. A faster framerate is considered better.

Although there are many possible solutions, the object recognition task is the same for all students. To increase the authenticity of the examination, all students also design, implement and test a 'unique' image processing operator.

1 Professional products

The assessment consists of the following professional products:

- source code
- demonstration of the object recognition
- powerpoint presentation of a unique operator
- demonstration of a unique operator
- logbook or comment in source code

The mandatory operators to be implemented are discussed during the lessons. *All these operators must function correctly in order to participate in the assessment.* In other words, they must all give a pass in the unit tests. The unique operator must also function correctly. For this, students devise their own (unit) test(s) and demonstrate the correct operation as part of the assessment.

During the classes in the first weeks the lecturer decides which students is assigned which unique operator.

2 Handin

Prior to the oral assessment, the following files must be handed in via https://handin.han.nl/ in a single ZIP archive with the assessor(s):

- All files in the folder **/evdk5/evdk_operators/*.***
- The file **evdk5/evdk_workspace_mcuxpresso/frdmmcxn947_evdk5_0/source/main.c**
- A presentation of the unique operator

3 Oral assessment

By participating in the assessment, you declare that all the work submitted was designed, realised and tested by yourself. This means that you are allowed to *collaborate*, but not *copy verbatim*. If code checks by the examiner reveal that large pieces of code have been copied verbatim, the assessor will report this to the board of examiners.

A planning of the oral assessments is drawn up during class in the final weeks of the period.

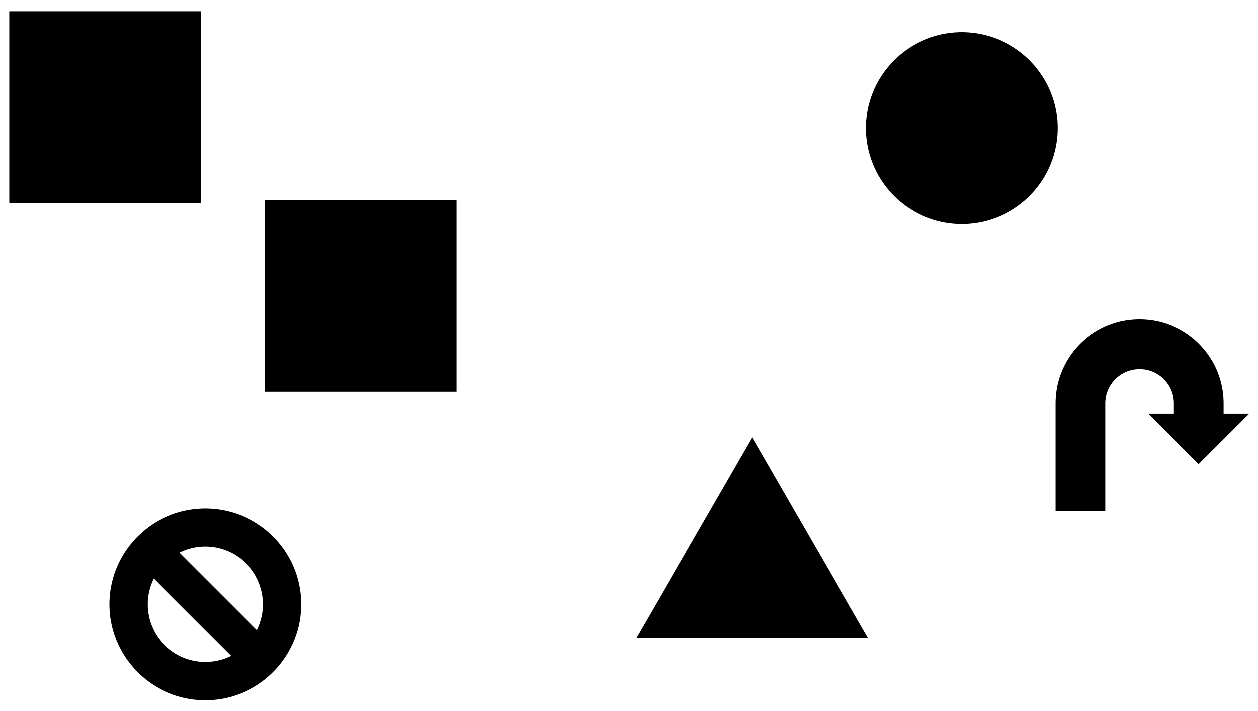
The oral assessment takes 20 minutes and is planned as follows:

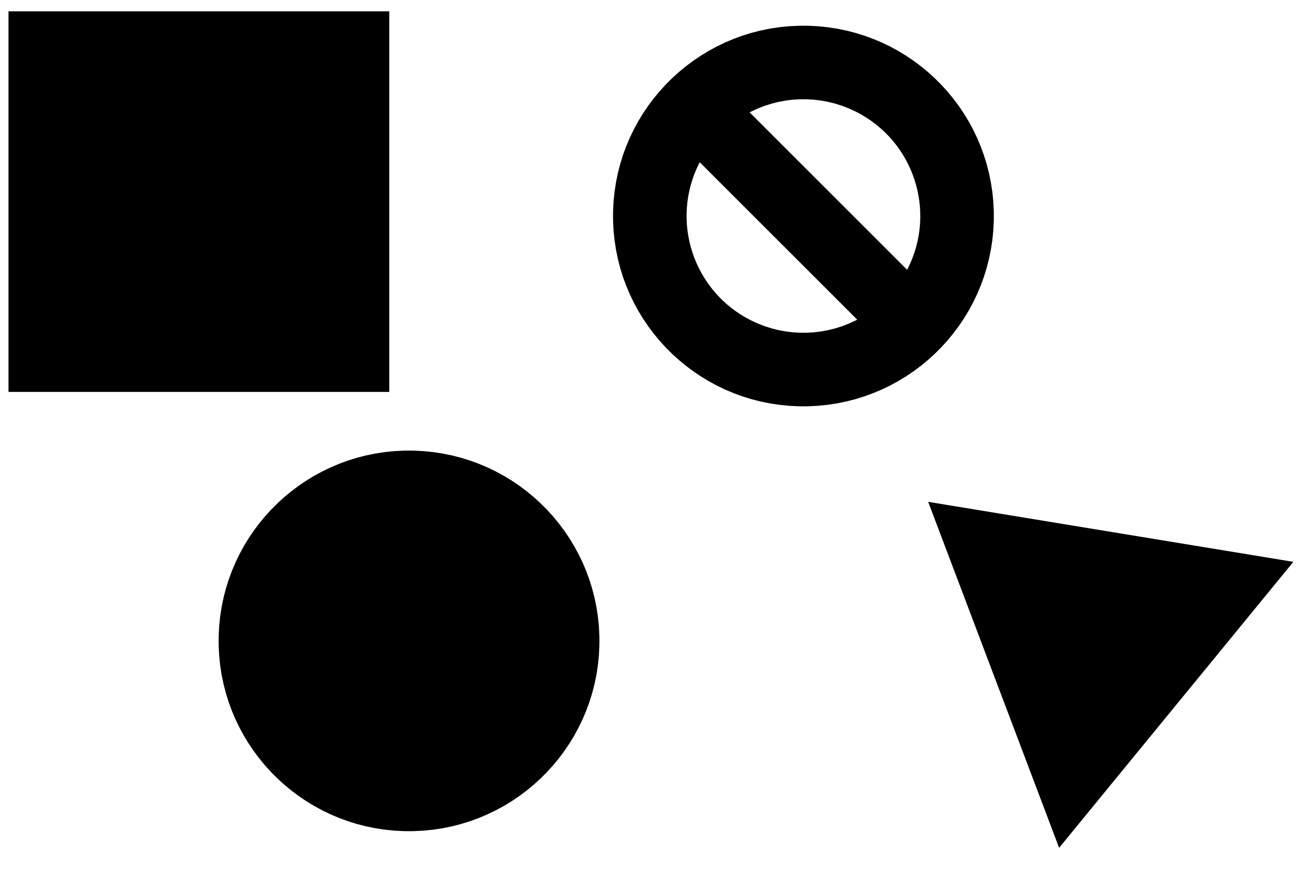
Time (min.)	Activity
~ 5	**Demonstration of the assignment** *The EVDK board is used to demonstrate that the three shapes – circle, square, and triangle. The next page of this document includes the shapes that will be used during the assessment. The source code, the used operators, the classification method, and overall performance are explained.*
~ 5	**Questions on a random operator** *The student is asked questions about the functional operation and technical implementation of a random operator based on the source code. The student must be able to provide an explanation and answer questions adequately. The handwritten logbook created by the student may be consulted.*
~ 10	**Presentation & demonstration unique operator** *The functional operation (to-the-point description and example) and the technical implementation (e.g. flowchart, mathematical equations, code snippets), of the unique operator is explained in a short presentation (approximately 5 sheets). The final slide must cite all used sources. Next, the operator is demonstrated. The student may use the EVDK board and/or the Qt development environment for this demonstration.*

As time is limited, students must ensure the hardware and software is ready for use when entering the exam room. If this is not the case, the examiner will have insufficient time to assess the knowledge and skills, which will result in a fail.

A printed copy of the sheet on the next page is used during the assessment. This page can be printed for testing.

EVD1 Assessment





Week 1

The goal for this week is to install the software development environment, configure the hardware, test the installation and get familiar with writing and testing code.

1 SDE setup

This section describes how to setup the Software Development Environment. Several of these tools are already installed, but are mentioned here for reference.

1.1 Qt Creator

Use the Qt Maintenance Tool (or go to https://www.qt.io/download) to install the latest version of Qt Creator and the MinGW compiler.

The latest successfully tested version is **Qt 6.7.1** and **MinGW 64 bit**.

No additional libraries are required.

1.2 OpenCV

Install OpenCV for Qt as described in the manual for EVD1 period 1.

The latest successfully tested version is **OpenCV 4.8.0**.

1.3 MCUXpresso-IDE

Download and install MCUXpresso-IDE.

1.4 Git client

Download and install a git client (such as Git for Windows).

For an overview of common git commands, see:

https://docs.gitlab.com/ee/gitlab-basics/start-using-git.html

1.5 Course files

Clone the course files from the git repo with the following command:

```
git clone https://gitlab.com/hugoarends/evdk5-2425.git
```

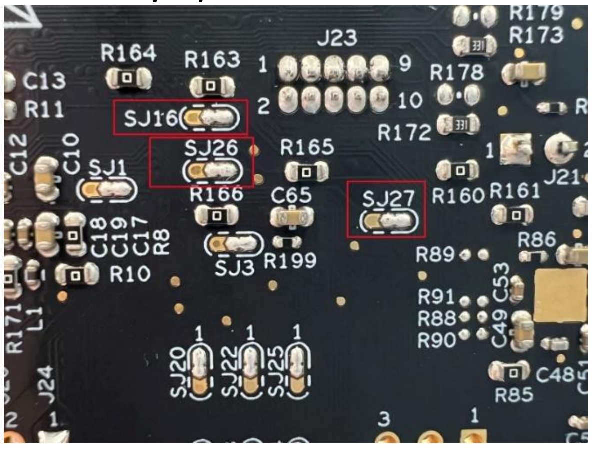
2 FRDM-MCXN947 changes

The FRDM-MCXN947 board does not work out-of-the-box with the provided project. Several jumpers need to be resoldered and it is recommended to change the MCU-Link firmware.

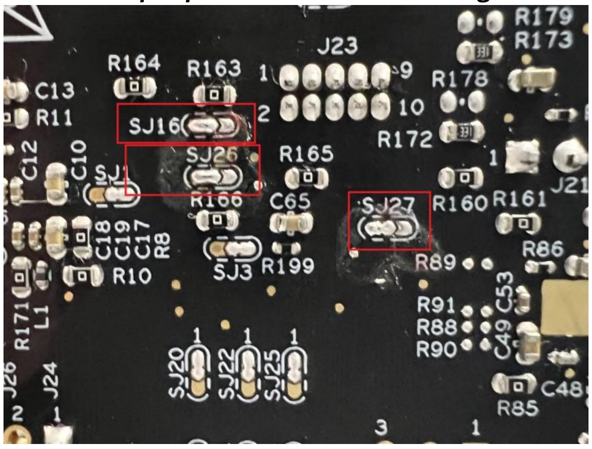
2.1 Resoldering jumpers

Three jumpers need to be resoldered, otherwise the OV7670 camera module cannot be connected to the J9 connector. The following images highlight these three changes:

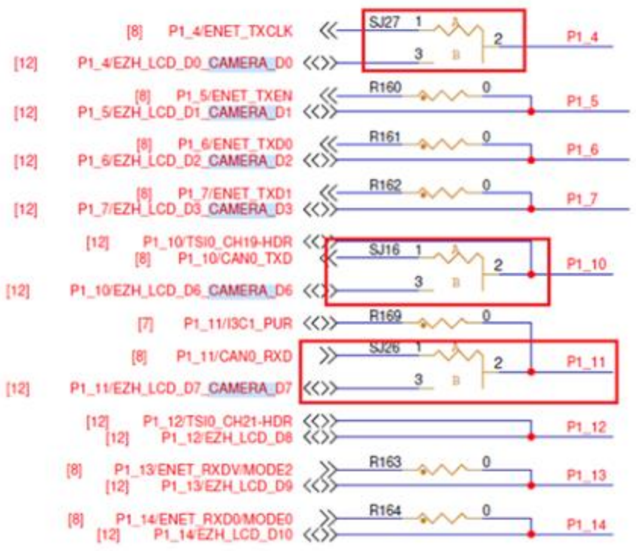
Jumper position out-of-the-box

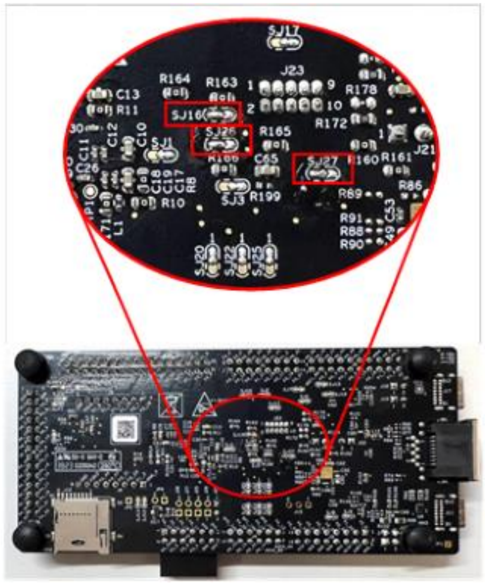


Jumper position after resoldering



The following images show the location of the jumpers at the bottom of the FRDM-MCXN947 board and how the changes are reflected in the schematic diagram.





These hardware changes are also described online.

2.2 MCU-Link firmware update

The FRDM-MCXN947 board comes with an MCU-Link debug probe. Out of the box, the CMSIS-DAP firmware is programmed. However, J-Link firmware is preferred. Changing the MCU-Link to J-Link firmware is described here. You can also ask your lecturer.

3 Hello Webcam!

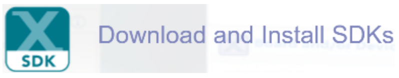
In this section the SDE setup and hardware changes will be verified by building the source files in MCUXpresso-IDE, programming the FRDM-MCXN947 board and see if your laptop displays images in the camera app.

3.1 Building the source files

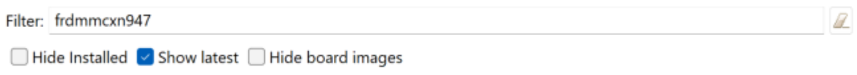
Build the source files:

1. Start *MCUXpresso-IDE*.

2. On the *Welcome* tab select *Download and Install SDKs*.

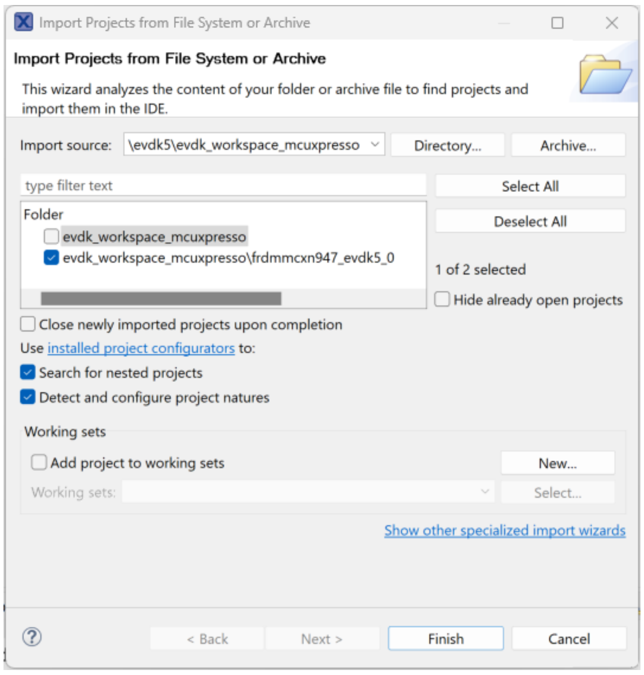


3. Select the FRDM-MCXN947 board.
   TIP. Use the filter option on the right top and type frdmmcxn947


Filter: frdmmcxn947

☐ Hide Installed ☑ Show latest ☐ Hide board images

4. Click *Install* and finish the installation.
5. Close the *Install MCUXpresso SDKs* tab.
6. Close the *Welcome* tab.
7. Click *File > Open Projects from File System…*
8. Select the folder *./evdk5/evdk_workspace_mcuxpresso*
9. Only select the project *evdk_workspace_mcuxpresso\frdmmcxn947_evdk5_0*


Import Projects from File System or Archive

Import Projects from File System or Archive

This wizard analyzes the content of your folder or archive file to find projects and import them in the IDE.

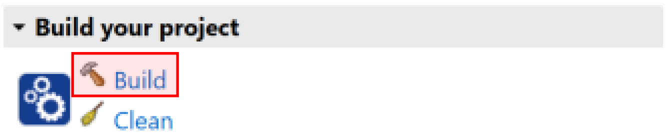
Import source: \evdk5\evdk_workspace_mcuxpresso

Folder
☐ evdk_workspace_mcuxpresso
☑ evdk_workspace_mcuxpresso\frdmmcxn947_evdk5_0

1 of 2 selected
☐ Hide already open projects
☐ Close newly imported projects upon completion
Use installed project configurators to:
☑ Search for nested projects
☑ Detect and configure project natures

Working sets
☐ Add project to working sets
Working sets:

10. Click *Finish*.
11. Click the *Build* button in the Quickstart panel.

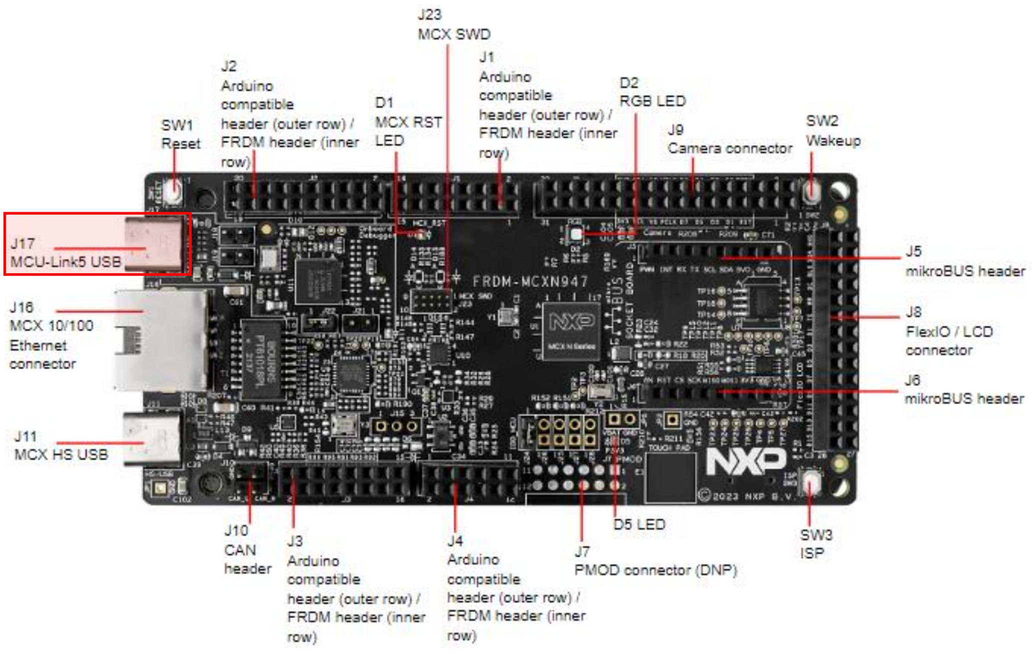


TIP. While the project is building, the console pane will display several warnings. These warnings were added to the project on purpose and will help to identify what image processing operators are to be implemented. When the source code is handed in, no warnings are allowed!

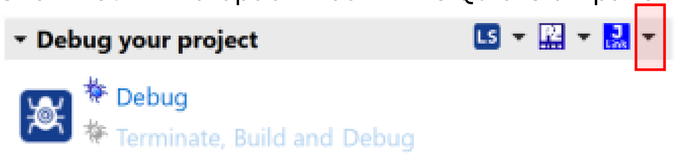
3.2 Programming the FRDM-MCXN947

Program the microcontroller as follows:

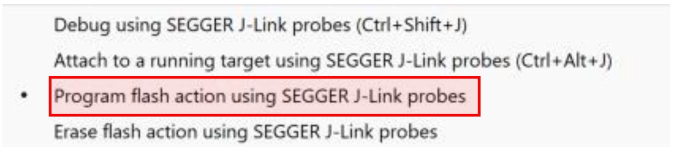
1. Connect the FRDM-MCXN947 development board to your laptop using connector *J17 MCU-Link USB*.


J23
MCX SWD
J2
Arduino compatible header (outer row) / FRDM header (inner row)
J1
Arduino compatible header (outer row) / FRDM header (inner row)
D2
RGB LED
D1
MCX RST LED
SW1
Reset
J9
Camera connector
SW2
Wakeup
J17
MCU-Link5 USB
J5
mikroBUS header
J16
MCX 10/100 Ethernet connector
J8
FlexIO / LCD connector
J6
mikroBUS header
J11
MCX HS USB
D5 LED
J10
CAN header
J3
Arduino compatible header (outer row) / FRDM header (inner row)
J4
Arduino compatible header (outer row) / FRDM header (inner row)
J7
PMOD connector (DNP)
SW3
ISP

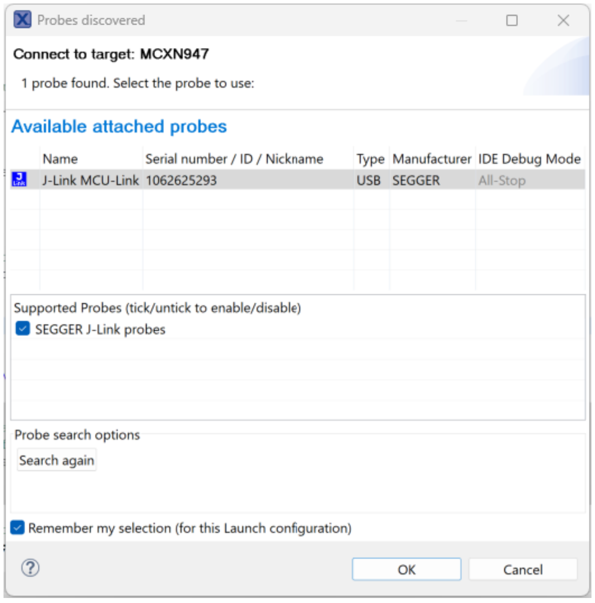
2. Click the J-Link dropdown box in the Quickstart panel.



3. Click *Program flash action using SEGGER J-Link probes*.


Debug using SEGGER J-Link probes (Ctrl+Shift+J)
Attach to a running target using SEGGER J-Link probes (Ctrl+Alt+J)
Program flash action using SEGGER J-Link probes
Erase flash action using SEGGER J-Link probes

4. Make sure a J-Link probe is found and click OK.


Probes discovered

Connect to target: MCXN947

1 probe found. Select the probe to use:

Available attached probes

Name	Serial number / ID / Nickname	Type	Manufacturer	IDE Debug Mode
J-Link MCU-Link	1062625293	USB	SEGGER	All-Stop

Supported Probes (tick/untick to enable/disable)

☑ SEGGER J-Link probes

Probe search options

Search again

☑ Remember my selection (for this Launch configuration)

OK Cancel

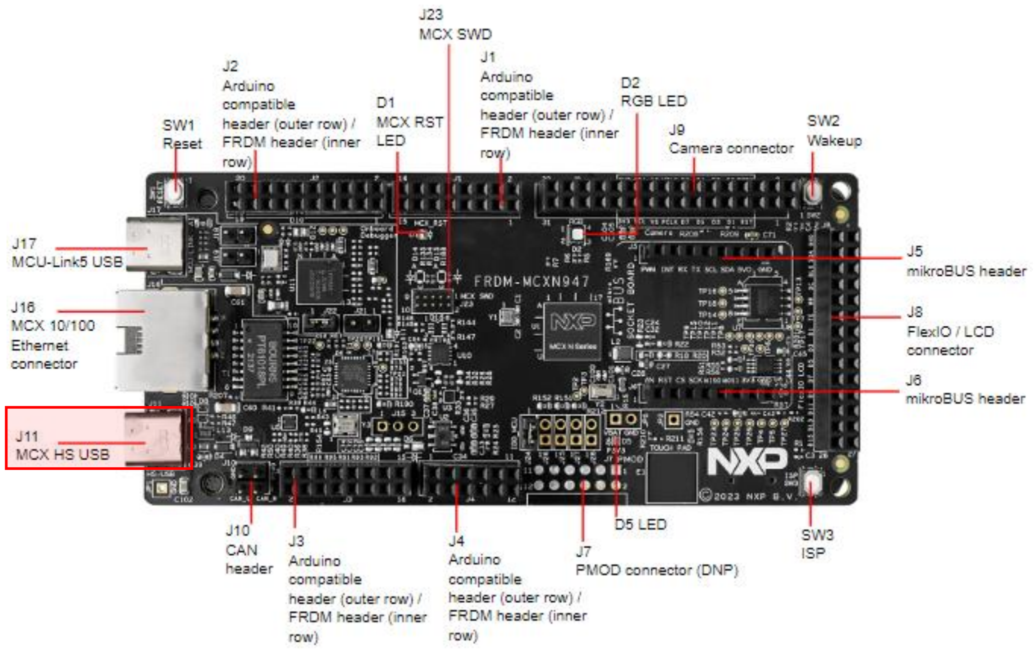
TIP. Clicking OK will start the programming flash action, but the first time, it also creates a file called frdmmcxn947_evdk5_0 JLink Debug.launch in your project. If you would like to make changes to this launch configuration, double click this file.

TIP. Chances are that the J-Link firmware is not up-to-date. The SEGGER tooling will automatically detect this and ask you to upgrade. Always click yes. Furthermore, J-Link can be used for free, however you will get a daily reminder of the fact that you are using the free version.

3.3 Display images in a camera app

Verify all of the steps above by displaying live images from the FRDM-MCXN947 board.

1. Connect the FRDM-MCXN947 development board to your laptop using connector *J11 MCX HS USB*.


J23
MCX SWD
J2
Arduino compatible header (outer row) / FRDM header (inner row)
J1
Arduino compatible header (outer row) / FRDM header (inner row)
D1
MCX RST LED
D2
RGB LED
SW1
Reset
J9
Camera connector
SW2
Wakeup
J17
MCU-Link5 USB
J5
mikroBUS header
J16
MCX 10/100 Ethernet connector
J8
FlexIO / LCD connector
J6
mikroBUS header
J11
MCX HS USB
J10
CAN header
J3
Arduino compatible header (outer row) / FRDM header (inner row)
J4
Arduino compatible header (outer row) / FRDM header (inner row)
D5 LED
J7
PMOD connector (DNP)
SW3
ISP

2. Open the camera app. On Windows: Start > Camera.
3. If your laptop's internal webcam shows images, switch to the EVDK5 by clicking

4 Change and run the MCUXpresso-IDE example project

The example selected in main() by default is exampleWebcamBgr888(). Change it to exampleWebcamBgr888TestPattern(), build and run the application.

When opening the camera app on your laptop, the image doesn't make sense. It is either completely black, or it might show coloured noise. Change the image processing pipeline by adding the following code:

```
// \todo Copy-and-paste week 1 code here

// Set all pixels to color
bgr888_pixel_t *bgr888_pixel = (bgr888_pixel_t *)bgr888->data;
int32_t len = bgr888->rows * bgr888->cols;

while(len > 0)
{
    *bgr888_pixel = color;

    bgr888_pixel++;
    len--;
}
```

Rebuild and run the application and verify using the camera app on your laptop that eight different colours are displayed, changing colour every second.

After an application reset, the example also prints a messages with the PRINTF() function to the serial interface. Open a terminal application (115200,8,n,1) and verify that the messages are printed.

TIP. There is a terminal application integrated in MCUXpresso-IDE: Window > Show View > Other... . In the filter input type 'Terminal' to select the terminal view.

5 Run Qt Webcam project

Connect the FRDM-MCXN947 board to your laptop and make sure that exampleThreshold() is running with the double for-loop as added in the previous section. Also make sure that the camera app is not running.

1. Open the project *./evdk5/evdk_workspace_qt/evdk5_webcam/evdk5_webcam.pro* in Qt creator.
2. Build and run the application.
3. If you get the following build error: *opencv2/opencv.hpp: No such file or directory*
   Set the *opencv_path* variable in the file *evdk5_webcam.pro*.
   Note. This should be done in all the example .pro files.
4. If you get the following error: *Process exited with code: -1073741515 (or similar)*
   Set the path variable to include the OpenCV binaries folder: *Projects > Environment > Path: C:\opencv-x.y.z\build\bin*
   Note. This should be done in all example projects when configuring a project.

When the app runs successfully, it shows an image similar to what could be seen in the camera app. Now, however, it is an OpenCV image viewer, so we can zoom in on the results.

5. Verify that when zooming in on pixel level, the individual pixels values are shown, for example(255, 0,0), (0,0,0) or (255, 255,255).
6. With the image view selected, press 's' on your keyboard. This will save an image to your disk, which might be convenient for future testing. The saved image is located at

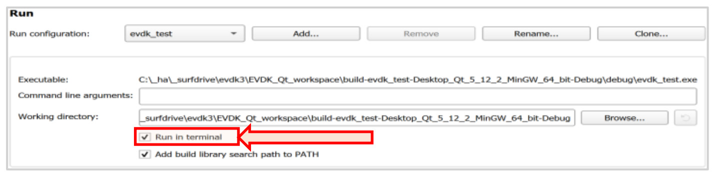
./evdk5/evdk_workspace_qt/build-evdk5_webcam-Desktop_Qt_x_y_z_MinGW_64_bit-Debug/image.bmp

6 Run Qt and Unity unit test project

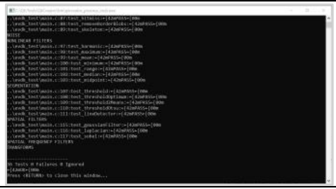
The Unity test framework is used within Qt Creator for unit testing the image processing operators. The project contains static images and several test cases per operator. There is no need to connect the EVDK5 hardware. The unit tests are executed as follows.

1. Open the project *./evdk5/evdk_workspace_qt/evdk5_unit_test/evdk5_ unit_test.pro* in Qt creator.
2. Build and run the application.
3. The output will be visualized in an internal or in an external terminal window as follows:

External terminal window (default setting):



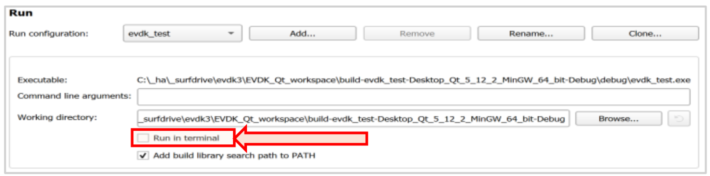
Run
Run configuration: evdk_test
Executable: C:_ha_surfdrive\evdk3\EVDK_Qt_workspace\build-evdk_test-Desktop_Qt_5_12_2_MinGW_64_bit-Debug\debug\evdk_test.exe
Command line arguments:
Working directory: _surfdrive\evdk3\EVDK_Qt_workspace\build-evdk_test-Desktop_Qt_5_12_2_MinGW_64_bit-Debug
✓ Run in terminal
✓ Add build library search path to PATH



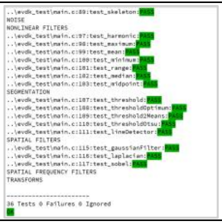
```
..\evdk_test\main.c:87:test_[illegible]:←[42mPASS←[00m
..\evdk_test\main.c:88:test_removeBorderBlobs:←[42mPASS←[00m
..\evdk_test\main.c:89:test_skeleton:←[42mPASS←[00m
NOISE
NONLINEAR FILTERS
..\evdk_test\main.c:97:test_harmonic:←[42mPASS←[00m
..\evdk_test\main.c:98:test_maximum:←[42mPASS←[00m
..\evdk_test\main.c:99:test_mean:←[42mPASS←[00m
..\evdk_test\main.c:100:test_minimum:←[42mPASS←[00m
..\evdk_test\main.c:101:test_range:←[42mPASS←[00m
..\evdk_test\main.c:102:test_median:←[42mPASS←[00m
..\evdk_test\main.c:103:test_midpoint:←[42mPASS←[00m
SEGMENTATION
..\evdk_test\main.c:107:test_threshold:←[42mPASS←[00m
..\evdk_test\main.c:108:test_thresholdOptimum:←[42mPASS←[00m
..\evdk_test\main.c:109:test_threshold2Means:←[42mPASS←[00m
..\evdk_test\main.c:110:test_thresholdOtsu:←[42mPASS←[00m
..\evdk_test\main.c:111:test_lineDetector:←[42mPASS←[00m
SPATIAL FILTERS
..\evdk_test\main.c:115:test_gaussianFilter:←[42mPASS←[00m
..\evdk_test\main.c:116:test_laplacian:←[42mPASS←[00m
..\evdk_test\main.c:117:test_sobel:←[42mPASS←[00m
SPATIAL FREQUENCY FILTERS
TRANSFORMS

-----------------------
36 Tests 0 Failures 0 Ignored
←[42mOK←[00m
Press <RETURN> to close this window...
```

Internal terminal window:



Run
Run configuration: evdk_test
Executable: C:_ha_surfdrive\evdk3\EVDK_Qt_workspace\build-evdk_test-Desktop_Qt_5_12_2_MinGW_64_bit-Debug\debug\evdk_test.exe
Command line arguments:
Working directory: _surfdrive\evdk3\EVDK_Qt_workspace\build-evdk_test-Desktop_Qt_5_12_2_MinGW_64_bit-Debug
☐ Run in terminal
✓ Add build library search path to PATH



```
..\evdk_test\main.c:89:test_skeleton:PASS
NOISE
NONLINEAR FILTERS
..\evdk_test\main.c:97:test_harmonic:PASS
..\evdk_test\main.c:98:test_maximum:PASS
..\evdk_test\main.c:99:test_mean:PASS
..\evdk_test\main.c:100:test_minimum:PASS
..\evdk_test\main.c:101:test_range:PASS
..\evdk_test\main.c:102:test_median:PASS
..\evdk_test\main.c:103:test_midpoint:PASS
SEGMENTATION
..\evdk_test\main.c:107:test_threshold:PASS
..\evdk_test\main.c:108:test_thresholdOptimum:PASS
..\evdk_test\main.c:109:test_threshold2Means:PASS
..\evdk_test\main.c:110:test_thresholdOtsu:PASS
..\evdk_test\main.c:111:test_lineDetector:PASS
SPATIAL FILTERS
..\evdk_test\main.c:115:test_gaussianFilter:PASS
..\evdk_test\main.c:116:test_laplacian:PASS
..\evdk_test\main.c:117:test_sobel:PASS
SPATIAL FREQUENCY FILTERS
TRANSFORMS

-----------------------
36 Tests 0 Failures 0 Ignored
OK
```

The unit test output will show several failures, because these operators have not yet been implemented. The implementation is part of your assignments and these unit tests can be used to test your implementation.

TIP. The unit tests can also be used as a TODO list, because part of the EVD1 assignments is the implement of all missing image processing operators.

7 Image fundamentals – convertUyvyToUint8()

Implement and test the function *convertUyvyToUint8()*.

1. Open the project *./evdk5/evdk_workspace_qt/evdk5_ unit_test/evdk5_ unit_test.pro* in Qt creator.
2. Open the file Sources > ../../evdk_operators/image_fundamentals.c
3. Find the function *convertUyvyToUint8()*.
4. Remove the warning and start the implementation.
5. Make sure the unit test gives a PASS instead of a FAIL.

TIP: In the file test_image_fundamentals.c, you will find the test function test_convertUyvyToUint8(). In this function, the source image data, destination image data and expected image data can be printed! All it takes is changing the following 0 to 1:

```
#if 0 // Change 0 to 1 to enable printing
```

```
    // Print testcase info
    printf("\n--------------------------------------------\n");
    printf("%s\n", name);

    // Print image data
    prettyprint(&src, "src");
    prettyprint(&exp, "exp");
    prettyprint(&dst, "dst");

#endif
```

Also test the implementation on the hardware.

1. Open the MCUXpresso-IDE project.
2. In main, select *exampleWebcamUint8()* and make sure no other example is selected.
3. Build the project and upload to the FRDM-MCXN947.
4. Test the result with a camera app (Windows Camera app of Qt webcam project). A live grayscale image must be visible.

8 Histogram operations – contrast()

Implement and test the function *contrast()*.

1. The function is located in the file ./evdk5/evdk_operators/histogram_operations.c
2. Make sure the unit test gives a PASS instead of a FAIL.

If you would like to inspect the histogram of an image, an example Qt Creator project is available.

1. Use a webcam or the EVDK5 with the *exampleWebcamBgr888()* running.
2. Open the project *./evdk5/evdk_workspace_qt/evdk5_histogram_webcam/evdk5_histogram_webcam.pro*
3. Run the application. Five images will popup:
    a. The original bgr888_pixel_t camera image
    b. The same image, but converted to uint8_pixel_t
    c. The histogram for this image
    d. The image after an operation, such as scaling
    e. The histogram for this image
4. Change the operation to *contrast()* and test your implementation with the following contrast correction arguments:
    a. 1.0
    b. 2.0
    c. 0.5
    d. 10.0

Week 2

The goal for this week is to enhance the performance of code execution. You will learn how to improve execution time, especially for 32-bit Cortex-M33 microcontrollers. To get the best performance, you will also learn how to implement a function in assembly programming language.

1 Image fundamentals – scaleFast()

Implement and test the function *scaleFast()*. It always stretches from 0 up and until 255.

1. Open the file Sources > ../../evdk_operators/image_funcdamentals.c
2. Find the function *scaleFast()*.
3. Remove the warning and start the implementation.
4. Combine several of the performance optimization techniques to make the operator execute **within 2.5ms**. This must be tested in MCUXpresso-IDE by calling the function *scaleFast()* in the function *exampleTemplate()*. For example:

```
        // --------------------------------------------------------------
        // Image processing pipeline
        // --------------------------------------------------------------
        // Convert uyvy_pixel_t camera image to uint8_pixel_t image
        convertToUint8(cam, src);

        // Copy timestamp
        ms1 = ms;

        scaleFast(src, dst);

        // Copy timestamp
        ms2 = ms;

        // Convert uint8_pixel_t image to bgr888_pixel_t image for USB
        convertToBgr888(dst, usb);
```

5. Note (in your log book, or in comment in your code) the performance gained for each technique.

The record is ~1770 us, with the FRDM-MCXN947 board connected to the computer and streaming images to a camera app.

6. Also make sure that the *scaleFast()* operator passes the unit test. Open the project *./evdk5/evdk_workspace_qt/evdk5_ unit_test/evdk5_ unit_test.pro* in Qt creator.
7. Make sure the unit test gives a PASS instead of a FAIL.

2 Image fundamentals – clearUint8Image_cm33()

Implement and test the function *clearUint8Image_cm33()*. The function clears the image data by setting all pixels to the value 0. This is the same functionality as the function *clearUint8Image()*.

1. Open the project MCUXpresso-IDE project.
2. Make sure the *Release* target is selected.
3. Use the following code in the function *exampleTemplate()* to measure how long the execution time of the function *clearUint8Image() is.*

```
// Copy timestamp
ms1 = ms;

clearUint8Image(dst); // Measured execution time ... us

// Copy timestamp
ms2 = ms;
```

4. Replace this function call as follows:

```
// Copy timestamp
ms1 = ms;

//clearUint8Image(dst); // Measured execution time ... us
clearUint8Image_cm33(dst); // Measured execution time ... us

// Copy timestamp
ms2 = ms;
```

5. Open the file *source/clearuint8image_cm33.s*
6. Implement the function.

TIP. The ARM Cortex-M33 instruction set can be found online:

[*https://developer.arm.com/documentation/100235/0100/The-Cortex-M33-Instruction-Set/Cortex-M33-instructions?lang=en*](https://developer.arm.com/documentation/100235/0100/The-Cortex-M33-Instruction-Set/Cortex-M33-instructions?lang=en)

The pseudo code for this function is:

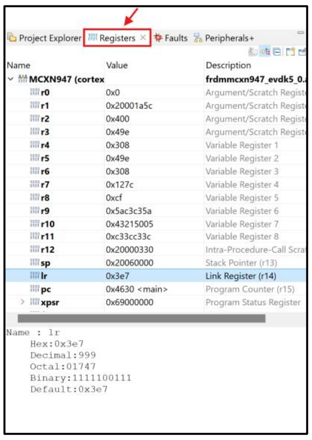
```
Push all used core registers to the stack.
Load four words from image pointer into R0 to R3.
Calculate the image size and store it in R1.
Move the image data pointer in R0.
Clear as many core registers as possible (search the instruction
set for a suitable instruction).
Do
    Store all these core registers to the image pointer (R0) and
    make sure that the pointer is also updated.
    Use the SUBS instruction to subtract the number of written
    pixels from the image size (R1).
    Use the BNE instruction to branch to the beginning of the
    loop.
Pop all used registers from the stack.
Return from the function.
```

As this function is specifically written for an ARM Cortex-M33 microcontroller, there is no unit test in the Qt unit test project.

1. Test the project by opening the Qt project *./evdk5/evdk_workspace_qt/evdk5_webcam/evdk5_webcam.pro* in Qt creator. Confirm that the pixels are all set to 0.

2. Verify its performance. It must be **< 100 µs**.

Important! If you would like to step through the assembly instructions, make sure the *Debug* target is selected! Furthermore, the contents of the core registers can be made visible in the debugger by selecting the *Registers* tab.



Name	Value	Description
MCXN947 (cortex		frdmmcxn947_evdk5_0.
r0	0x0	Argument/Scratch Regist
r1	0x20001a5c	Argument/Scratch Regist
r2	0x400	Argument/Scratch Regist
r3	0x49e	Argument/Scratch Regist
r4	0x308	Variable Register 1
r5	0x49e	Variable Register 2
r6	0x308	Variable Register 3
r7	0x127c	Variable Register 4
r8	0xcf	Variable Register 5
r9	0x5ac3c35a	Variable Register 6
r10	0x43215005	Variable Register 7
r11	0xc33cc33c	Variable Register 8
r12	0x20000330	Intra-Procedure-Call Scra
sp	0x20060000	Stack Pointer (r13)
lr	0x3e7	Link Register (r14)
pc	0x4630 <main>	Program Counter (r15)
xpsr	0x69000000	Program Status Register

```
Name : lr
    Hex:0x3e7
    Decimal:999
    Octal:01747
    Binary:1111100111
    Default:0x3e7
```

3 Image fundamentals – convertUyvyToUint8_cm33() EXTRA

This is an EXTRA assignment for those students who like a challenge. The function *convertUyvyToUint8()* is called every loop, so it makes sense to enhance its performance.

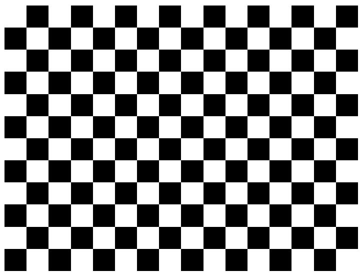
The function *convertUyvyToUint8()* takes approximately 970 µs to execute (Release target selected and webcam app running on PC). The function can be improved to make it run in approximately 430 µs.

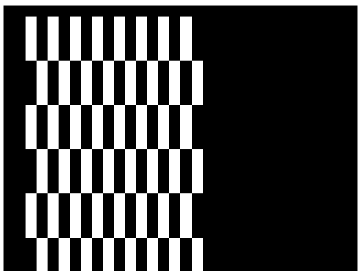
Implement and test the function *convertUyvyToUint8_cm33()*.

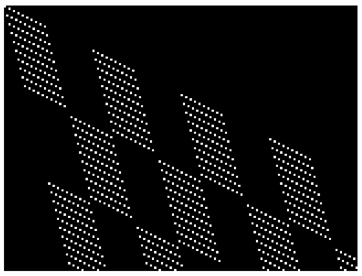
TIP. In the loop, consider using the LSR and BFI instructions.

4 Graphics algorithms – affineTransformation()

Examine the following images. These images are also provided in the evdk repo in the folder ./evdk5/evdk_images.


affine-src.png


affine-dst1.png


affine-dst2.png

1. Open these images in OpenCV by using the Qt example project *./evdk5/evdk_workspace_qt/evdk5_img_from_file/evdk5_img_from_file.pro* Zoom in for a detailed, pixel level analysis of the images.
2. Answer the following questions for both destination images:
   a. Has backward (inverse) or forward transformation been applied?
   b. Has scaling been applied? If yes, how much?
   c. Has rotation been applied? If yes, how much?

d. Has translation been applied? If yes, how much?
e. Has shearing been applied? If yes, how much?

3. Determine for both destination images the values for the affine transformation coefficients *a..f*.
4. Use these coefficients to produce the same images from the given *affine-src.png*. Make sure the dst image is cleared (*clearUint8Image()*) before calling the *affineTransformation()* function.

For affine_dst1.png: $\begin{vmatrix} 0.5 & 0 & 10 \\ 0 & 2 & 5 \\ 0 & 0 & 1 \end{vmatrix}$ and backward transformation

For affine_dst2.png: $\begin{vmatrix} 2 & 1 & 0 \\ 1 & 4 & 0 \\ 0 & 0 & 1 \end{vmatrix}$ and forward transformation

Week 3

The theme for this week is image enhancement by means of filtering. Most of the filters are already implemented and can be used as a reference implementation when enhancing their performance.

1 Unique operator selection

During this week each student is assigned a unique image processing operator. This is an operator that is not part of the EVDK framework. This operator is unique in that sense, that it is implemented by a single student. All other students are assigned a different operator.

The lecturer maintains a list of operators to choose from. This list is on a first-come-first serve basis. During this week, each student must have chosen an operator. Otherwise, the lecturer will assign one.

During this week, students have to come up with a function prototype and a description of the arguments and return value. This function prototype is discussed with the lecturer during class and is a go/no-go moment.

Use the following template for your function. Make sure to give the function a suitable name, change the arguments and/or the return value as needed.

```c
/*!
 * \brief Description of the function in one sentence
 *
 * A detailed description
 * of the function that takes
 * several lines, but each line
 * doesn’t contain more than
 * 80 characters.
 *
 * \param[in]  src A pointer to the source image, must be of type ...
 * \param[out] dst A pointer to the destination image, must be of type ...
 * \param[in]  val Value that is ...
 *
 * \return \li -1 Failure
 *         \li  0 Memory error
 *         \li  1 Success
 */
int32_t myUniqueOperator(image_t *src, image_t *dst, int32_t val)
{

}
```

Depending on the type of function, it must be located in one of the existing files.

2 Image fundamentals – convolveFast()

Implement and test the function *convolveFast()*.

1. The function is located in the file ./evdk5/evdk_operators/image_fundamentals.c
2. Remove the warning and start the implementation.

3. Make sure the unit test gives a PASS instead of a FAIL.
4. Combine several of the performance optimization techniques to make the operator execute **within 10 ms**. Use the FRDM-MCXN947 board to measure the performance in ms precision.

3 Nonlinear filters – meanFast()

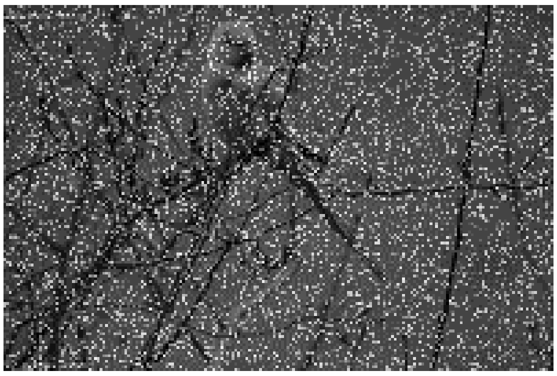
Although all nonlinear filters have the same basic implementation structure, for this assignment you will improve the performance of one of them. Implement and test the function *meanFast()*. Remember that the implementation of a nonlinear filter is very similar to a convolution.

1. The function is located in the file ./evdk5/evdk_operators/nonlinear_filters.c
2. Remove the warning and start the implementation.
3. Make sure the unit test gives a PASS instead of a FAIL.
4. Combine several of the performance optimization techniques to make the operator execute **within 10 ms**. Use the FRDM-MCXN947 board to measure the performance in ms precision.

4 Nonlinear filters – EXTRA

This is an EXTRA assignment that addresses the use of nonlinear filters. Answer the following questions.

1. Which nonlinear filter performs best on the adjacent image **filters.bmp** (see images folder)? Why?
2. Which of the nonlinear filters can be used for binary erosion? Provide an example. *Use the function threshold() to create a binary image.*
3. Which of the nonlinear filters can be used for binary dilation? Provide an example.
4. Which of the nonlinear filters can be used for binary detecting edges? Provide an example.


filters.bmp

1. The median filter, because the image mainly contains salt and pepper noise.
2. The minimum filter can be used for binary erosion.
3. The maximum filter can be used for binary erosion.
4. The range filter can be used for binary edge detection.

5 Spatial filters – sobelFast()

Implement and test the function *sobelFast()*. Use, amongst others, the convolveFast() function to improve performance.

1. The function is located in the file ./evdk5/evdk_operators/spatial_filters.c
2. Remove the warning and start the implementation.
3. Make sure the unit test gives a PASS instead of a FAIL.

4. Combine several of the performance optimization techniques to make the operator execute **within 20 ms**. Use the FRDM-MCXN947 board to measure the performance in ms precision.

Week 4

This week discusses algorithms for automatic thresholding for separating objects from the background. These algorithms are based on the graylevel histogram.

1 Segmentation – threshold2Means()

Implement and test the function *threshold2Means()*.

1. The function is located in the file ./evdk5/evdk_operators/segmentation.c
2. Remove the warning and start the implementation.
3. Make sure the unit test gives a PASS instead of a FAIL.

2 Segmentation – thresholdOtsu()

Implement and test the function *thresholdOtso()*.

1. The function is located in the file ./evdk5/evdk_operators/segmentation.c
2. Remove the warning and start the implementation.
3. Make sure the unit test gives a PASS instead of a FAIL.

Week 5

The goal for this week is to reduce binary objects as much as possible to their basic shapes. Morphological filters are used for that purpose.

1 Morphological filters – removeBorderBlobsTwoPass()

Implement and test the function *removeBorderBlobsTwoPass()*.

1. The function is located in the file ./evdk5/evdk_operators/morphological_filters.c
2. Remove the warning and start the implementation.

Make sure the unit test gives a PASS instead of a FAIL.

TIPS

- The argument *lutSize* is used to dynamically allocate memory in the removeBorderBlobsTwoPass() function for the equivalence lookup table.
    - Use the function malloc().
    - Check if the allocation succeeded by verifying if the returned pointer is not equal to NULL.
    - Use the function memset() to set the entire lookup table to zero.
    - Do not forget to use free() when the function finishes.
- The function returns 1 on successful execution and returns 0 in case of the following failures:
    - Memory allocation for the lookup table failed
    - The lookup table is too small. In other words, the image requires more unique labels than can be stored in the lookup table.

2 Morphological filters – fillHolesTwoPass()

Implement and test the function *fillHolesTwoPass()*.

1. The function is located in the file ./evdk5/evdk_operators/morphological_filters.c
2. Remove the warning and start the implementation.

Make sure the unit test gives a PASS instead of a FAIL.

TIPS. You should be able to reuse the majority of the code from the removeBorderBlobsTwoPass() function.

Week 6

This last week discusses techniques to extract object features. Some of these features are translation, rotation and scaling invariant. These features can be used in the final assignment for object classification.

1 Mensuration – labelTwoPass()

Implement and test the function *labelTwoPass()* according the two-pass labelling algorithm. To ensure a fast implementation, it is allowed to skip the border pixels.

1. The function is located in the file ./evdk5/evdk_operators/mensuration.c
2. Remove the warning and start the implementation.
3. Make sure the unit test gives a PASS instead of a FAIL.
4. Use the FRDM-MCXN947 board to measure the performance in ms precision.

EXTRA. Skipping the border pixels has two major disadvantages. The first is the decrease in resolution, which especially is a problem when working with an image pyramid and very small images. The second is that an object's connectivity might be broken. This is visualized in the following images. The left image handles the border pixels correctly and shows a single 8-connected BLOB. The right image, however, skips the borders and shows two BLOBs.

			1	1			
	1	1			1	1	
	1	1			1	1	
	1	1			1	1	

	1	1			2	2	
	1	1			2	2	
	1	1			2	2	

In this EXTRA assignment, change the implementation so that border pixels are not skipped anymore. Handle these border pixels separately from the inner pixels, to prevent an out-of-bounds check for every pixel.

2 Mensuration – perimeter()

Implement and test the function *perimeter()* according the alternative d algorithm (see week 6 slides).

1. The function is located in the file ./evdk5/evdk_operators/mensuration.c
2. Remove the warning and start the implementation.
3. Make sure the unit test gives a PASS instead of a FAIL.
4. Use the FRDM-MCXN947 board to measure the performance in ms precision.

3 Mensuration – circularity()

The *circularity()* function is already implemented. Use it in the FRDM-MCXN947 project and print the circularity of the objects as used in the assessment sheet. From these printed values, deduce the expected minimum and maximum values for each shape (square, triangle, circle and unknown).

4 Mensuration – huMoments()

The *huMoments()* function is already implemented. Use it to print the first four hu invariant moments of the objects as used in the assessment sheet. From these printed values, deduce the expected minimum and maximum values for each shape (square, triangle, circle and unknown).

5 Final assignment

The final assignment is described in detail in section Final assignment. The solution implements the following steps:

- Acquisition: convert the image from uyvy_pixel_t to uint8_pixel_t. Use an image pyramid to perform image processing operations on images as small as possible.
- Enhancement: pre-process the image (if needed) to enhance the objects.
- Segmentation: segment objects from background.
- Feature extraction: calculate object features, such as circularity or Hu invariant moments.
- Classification: classify the object based on the features.

The camera produces images at 30fps. This makes for a timing budget $\frac{1}{30} = 33\text{ms}$. Try to make the solution execute on the FRDM-MCXN947 board within this timing budget. Provide an answer to the following questions:

- How long does each operation take?
- How long does the total of all operations take?
- If the total of all operations takes longer than timing budget, what can be done to improve performance?