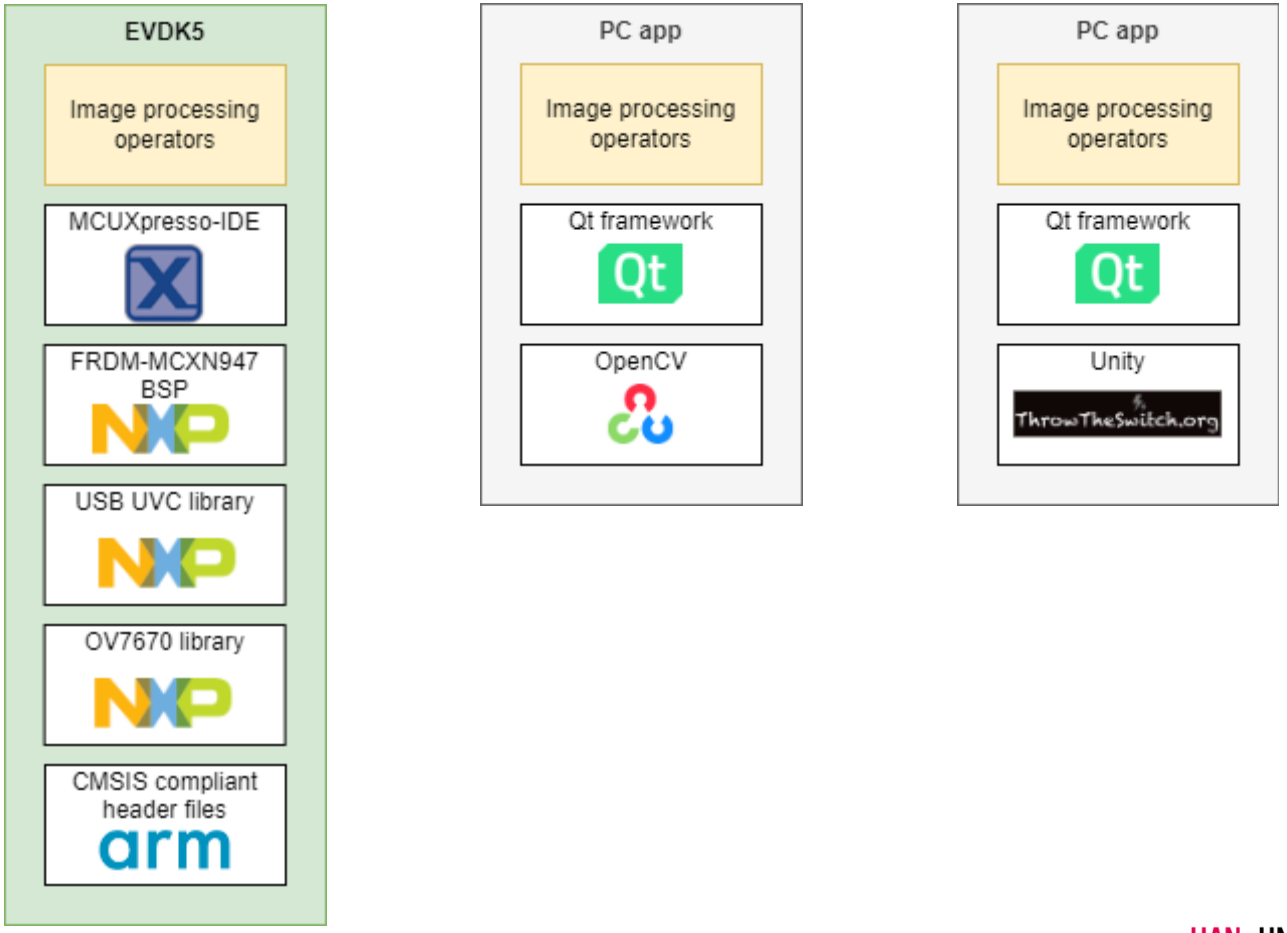
Embedded Vision Design

EVD1 Development basics

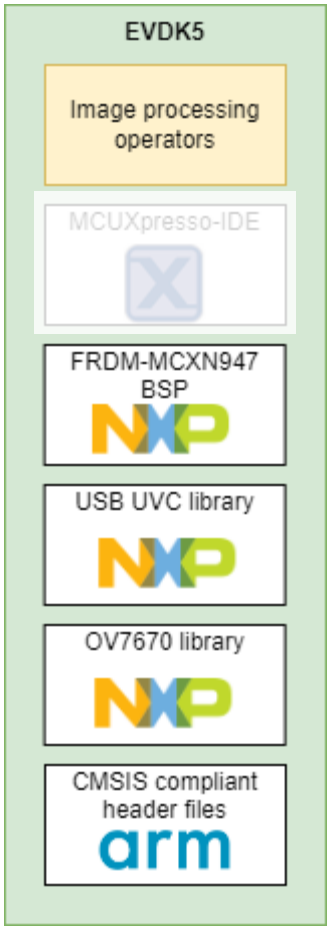
By Hugo Arends

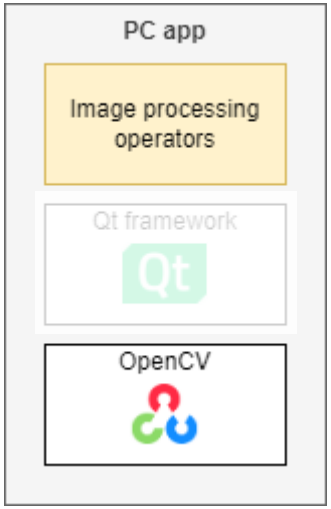
HAN_UNIVERSITY
OF APPLIED SCIENCES

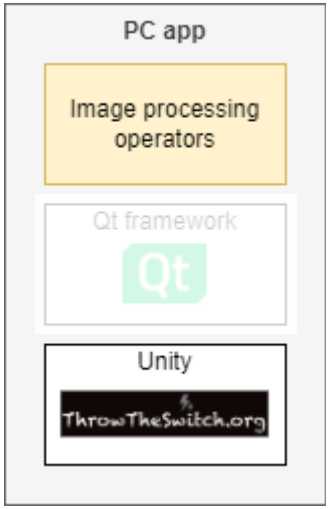
EVD1 – Tools and software



EVD1 – Tools and software







Tools vs software packages

EVD1 – File overview

- evdk5
  - evdk_images
  - evdk_operators
  - evdk_sheets
  - evdk_workspace_mcuxpresso
  - evdk_workspace_qt

Top level folder structure

EVD1 – File overview

- evdk5
  - evdk_images
  - evdk_operators
    - coding_and_compression.c
    - coding_and_compression.h
    - graphics_algorithms.c
    - graphics_algorithms.h
    - …
  - evdk_sheets
  - evdk_workspace_mcuxpresso
  - evdk_workspace_qt

Image processing source files, a file per class

EVD1 – File overview

- evdk5
  - evdk_images
  - evdk_operators
  - evdk_sheets
  - evdk_workspace_mcuxpresso
    - frdmmcxn947_evdk5_0
      - .settings
      - board
      - CMSIS
      - …
      - usb
      - utilities
      - video
      - .cproject
      - .project
      - frdmmcxn947_evdk5_0.mex
  - evdk_workspace_qt

MCUXpresso-IDE resource files and drivers

EVD1 – File overview

- evdk5
  - evdk_images
  - **evdk_operators**
  - evdk_sheets
  - evdk_workspace_mcuxpresso
    - frdmmcxn947_evdk5_0
      - .settings
      - board
      - CMSIS
      - …
      - usb
      - utilities
      - video
      - .cproject
      - .project
      - frdmmcxn947_evdk5_0.mex
  - evdk_workspace_qt

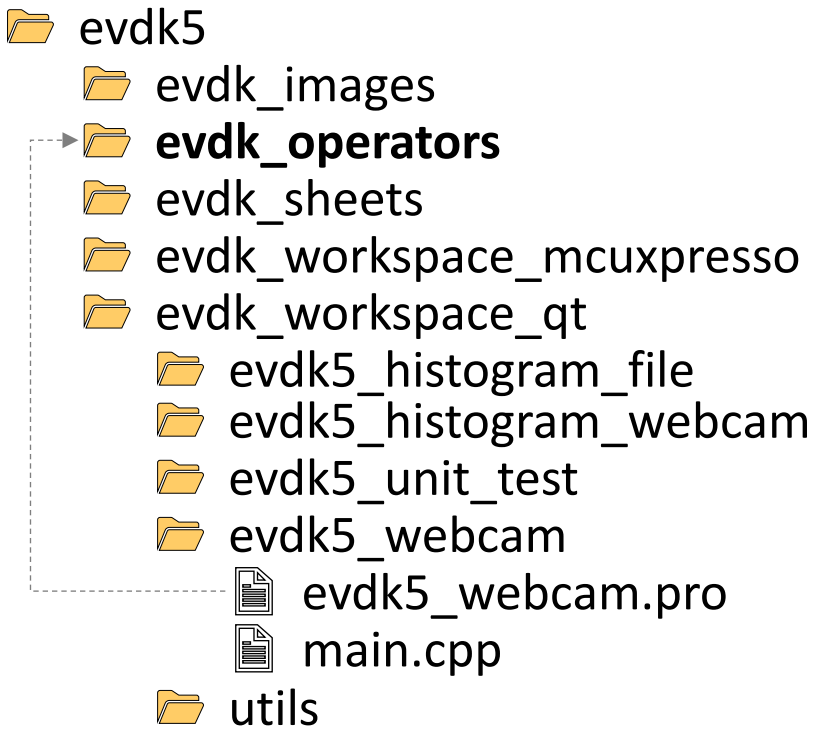
MCUXpresso-IDE resource files and drivers

EVD1 – File overview

- evdk5
  - evdk_images
  - evdk_operators
  - evdk_sheets
  - evdk_workspace_mcuxpresso
  - evdk_workspace_qt
    - evdk5_histogram_file
    - evdk5_histogram_webcam
    - evdk5_unit_test
    - evdk5_webcam
      - evdk5_webcam.pro
      - main.cpp
    - utils

Qt example projects

EVD1 – File overview



Qt example projects

EVD1 – File overview

- evdk5
  - evdk_images
  - evdk_operators
  - evdk_sheets
  - evdk_workspace_mcuxpresso
  - evdk_workspace_qt
    - evdk5_histogram_file
    - evdk5_histogram_webcam
    - evdk5_unit_test
      - Unity
      - evdk5_test.pro
      - main.c
      - test_coding_and_compression.c
      - test_coding_and_compression.h
      - …
    - evdk5_webcam
    - utils

Qt unit test project

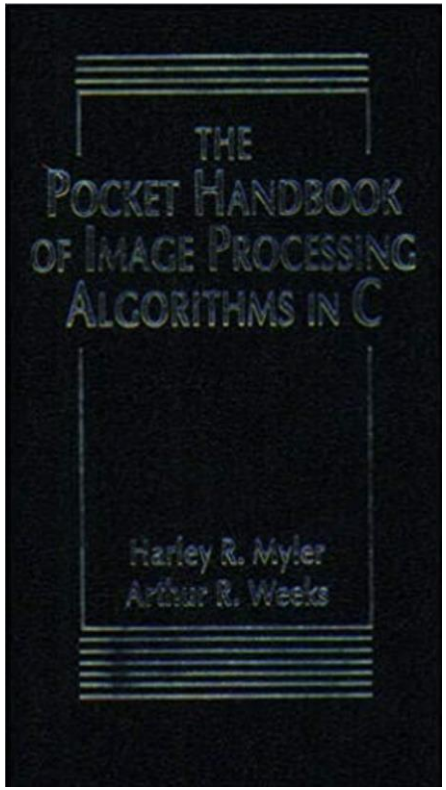
EVD1 – File overview

- evdk5
  - evdk_images
  - **evdk_operators**
  - evdk_sheets
  - evdk_workspace_mcuxpresso
  - evdk_workspace_qt
    - evdk5_histogram_file
    - evdk5_histogram_webcam
    - evdk5_unit_test
      - Unity
      - evdk5_test.pro
      - main.c
      - test_coding_and_compression.c
      - test_coding_and_compression.h
      - …
    - evdk5_webcam
    - utils

Qt unit test project

Image basics – eImageType

```
/// Defines the type of images
typedef enum
{
    IMGTYPE_UINT8  =  1, ///< Pixels of type ::uint8_pixel_t.
    IMGTYPE_INT16  =  2, ///< Pixels of type ::int16_pixel_t.
    IMGTYPE_INT32  =  4, ///< Pixels of type ::int32_pixel_t.
    IMGTYPE_FLOAT  =  8, ///< Pixels of type ::float_pixel_t.
    IMGTYPE_UYVY   = 16, ///< Pixels of type ::uyvy_pixel_t.
    IMGTYPE_BGR888 = 32, ///< Pixels of type ::bgr888_pixel_t.
}eImageType;
```



Myler, H. R., & Weeks, A. R. (2009). *The pocket handbook of image processing algorithms in C*. Prentice Hall Press.

Image basics – eImageType

```
/// Defines the type of images
typedef enum
{
    IMGTYPE_UINT8  =  1, ///< Pixels of type ::uint8_pixel_t.
    IMGTYPE_INT16  =  2, ///< Pixels of type ::int16_pixel_t.
    IMGTYPE_INT32  =  4, ///< Pixels of type ::int32_pixel_t.
    IMGTYPE_FLOAT  =  8, ///< Pixels of type ::float_pixel_t.
    IMGTYPE_UYVY   = 16, ///< Pixels of type ::uyvy_pixel_t.
    IMGTYPE_BGR888 = 32, ///< Pixels of type ::bgr888_pixel_t.
}eImageType;
```

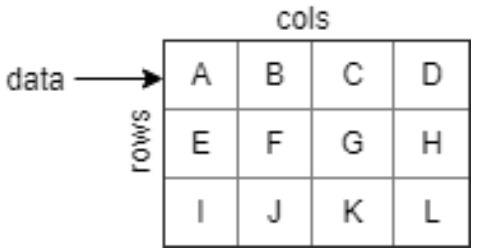
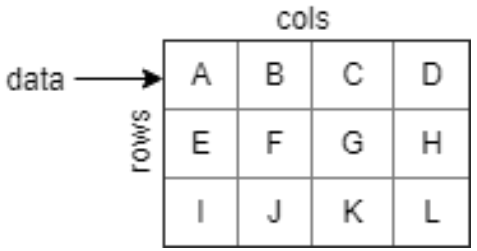


Image basics – eImageType

```
/// Defines the type of images
typedef enum
{
    IMGTYPE_UINT8  =  1, ///< Pixels of type ::uint8_pixel_t.
    IMGTYPE_INT16  =  2, ///< Pixels of type ::int16_pixel_t.
    IMGTYPE_INT32  =  4, ///< Pixels of type ::int32_pixel_t.
    IMGTYPE_FLOAT  =  8, ///< Pixels of type ::float_pixel_t.
    IMGTYPE_UYVY   = 16, ///< Pixels of type ::uyvy_pixel_t.
    IMGTYPE_BGR888 = 32, ///< Pixels of type ::bgr888_pixel_t.
}eImageType;
```

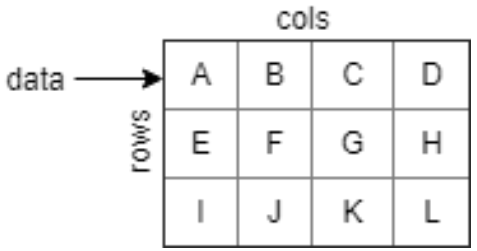
```
/// \brief Type definition of a uint8 pixel
///
/// 8 bits per pixel
typedef uint8_t uint8_pixel_t;
```



Memory allocation: $12 \times 1\ byte = 12\ bytes$

Image basics – eImageType

```
/// Defines the type of images
typedef enum
{
    IMGTYPE_UINT8  =  1, ///< Pixels of type ::uint8_pixel_t.
    IMGTYPE_INT16  =  2, ///< Pixels of type ::int16_pixel_t.
    IMGTYPE_INT32  =  4, ///< Pixels of type ::int32_pixel_t.
    IMGTYPE_FLOAT  =  8, ///< Pixels of type ::float_pixel_t.
    IMGTYPE_UYVY   = 16, ///< Pixels of type ::uyvy_pixel_t.
    IMGTYPE_BGR888 = 32, ///< Pixels of type ::bgr888_pixel_t.
}eImageType;
```

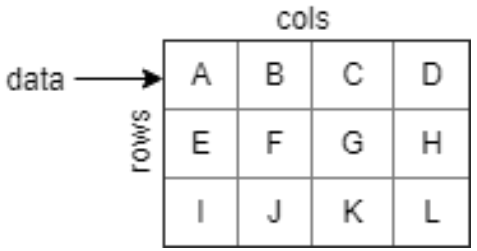



```
/// \brief Type definition of an int16 pixel
///
/// 16 bits per pixel
typedef int16_t int16_pixel_t;
```

Memory allocation: $12 \times 2\ bytes = 24\ bytes$

Image basics – eImageType

```
/// Defines the type of images
typedef enum
{
    IMGTYPE_UINT8  =  1, ///< Pixels of type ::uint8_pixel_t.
    IMGTYPE_INT16  =  2, ///< Pixels of type ::int16_pixel_t.
    IMGTYPE_INT32  =  4, ///< Pixels of type ::int32_pixel_t.
    IMGTYPE_FLOAT  =  8, ///< Pixels of type ::float_pixel_t.
    IMGTYPE_UYVY   = 16, ///< Pixels of type ::uyvy_pixel_t.
    IMGTYPE_BGR888 = 32, ///< Pixels of type ::bgr888_pixel_t.
}eImageType;
```




```
/// \brief Type definition of an int32 pixel
///
/// 32 bits per pixel
typedef int32_t int32_pixel_t;
```

Memory allocation: $12 \times 4\ bytes = 48\ bytes$

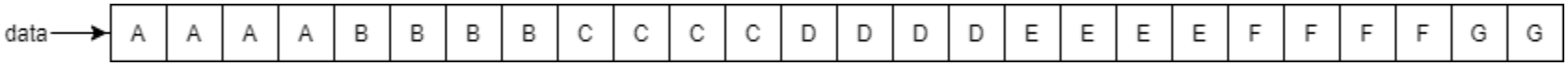
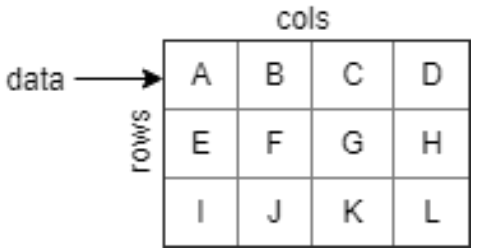


Image basics – eImageType

```
/// Defines the type of images
typedef enum
{
    IMGTYPE_UINT8  =  1, ///< Pixels of type ::uint8_pixel_t.
    IMGTYPE_INT16  =  2, ///< Pixels of type ::int16_pixel_t.
    IMGTYPE_INT32  =  4, ///< Pixels of type ::int32_pixel_t.
    IMGTYPE_FLOAT  =  8, ///< Pixels of type ::float_pixel_t.
    IMGTYPE_UYVY   = 16, ///< Pixels of type ::uyvy_pixel_t.
    IMGTYPE_BGR888 = 32, ///< Pixels of type ::bgr888_pixel_t.
}eImageType;
```




```
/// \brief Type definition of a float pixel
///
/// 32 bits per pixel
typedef float float_pixel_t;
```

Memory allocation: $12 \times 4\ bytes = 48\ bytes$

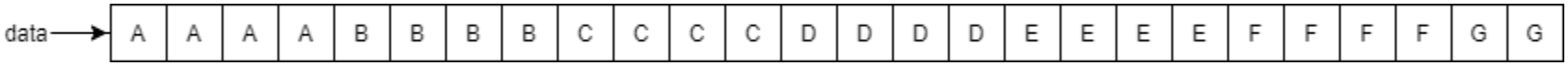


Image basics – Creating images

```
// Create an image
image_t *src = newUint8Image(4, 3);

// Use src in an image processing pipeline
// ...

// Cleanup
deleteUint8Image(src);
```

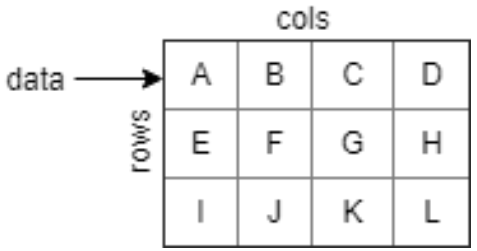


Image basics – Creating images

```
// Create an image
image_t *src = newFloatImage(4, 3);

// Use src in an image processing pipeline
// ...

// Cleanup
deleteFloatImage(src);
```

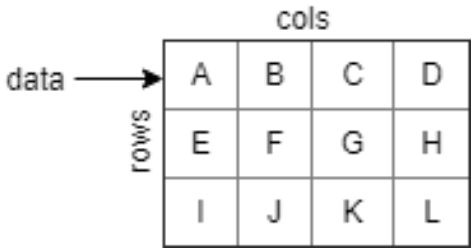
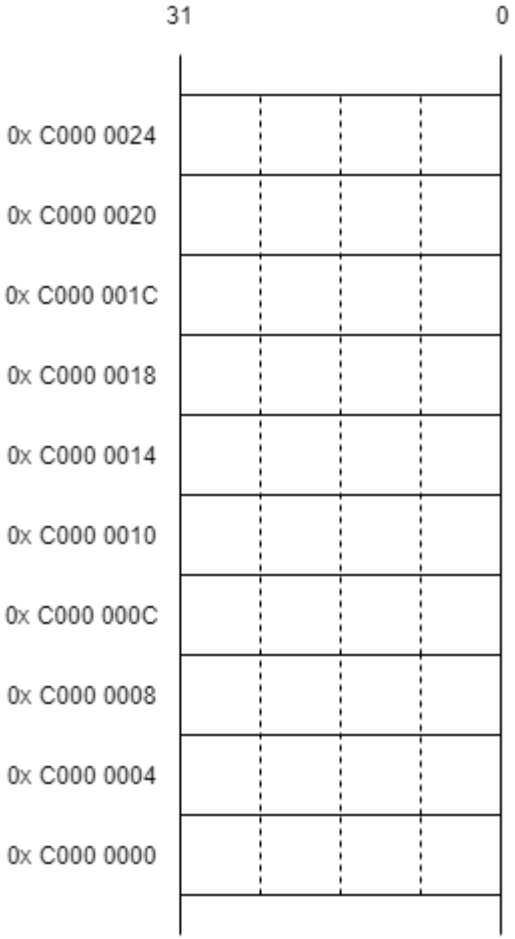


Image basics – Pointers




```
int a = 0;
```

- ‘a’ is a variable, but what is a variable?
  name of a storage area
- What does the type of a variable tell?
  size and layout in memory
- How can we get the memory address of a variable?
  by using the reference operator: &

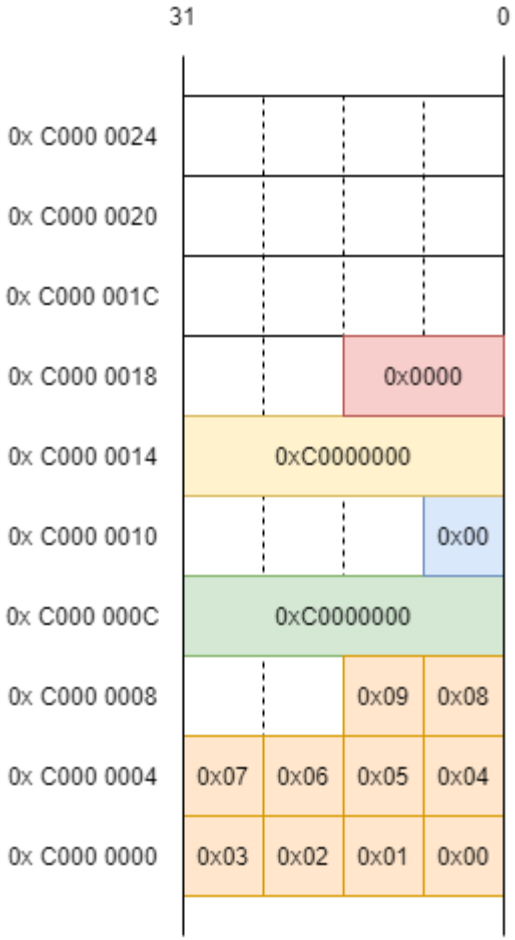
```
int *p1 = &a;
```

- Why is this incorrect?

```
char *p2 = &a;
```

The base-type of the pointer is different from the base-type of the variable

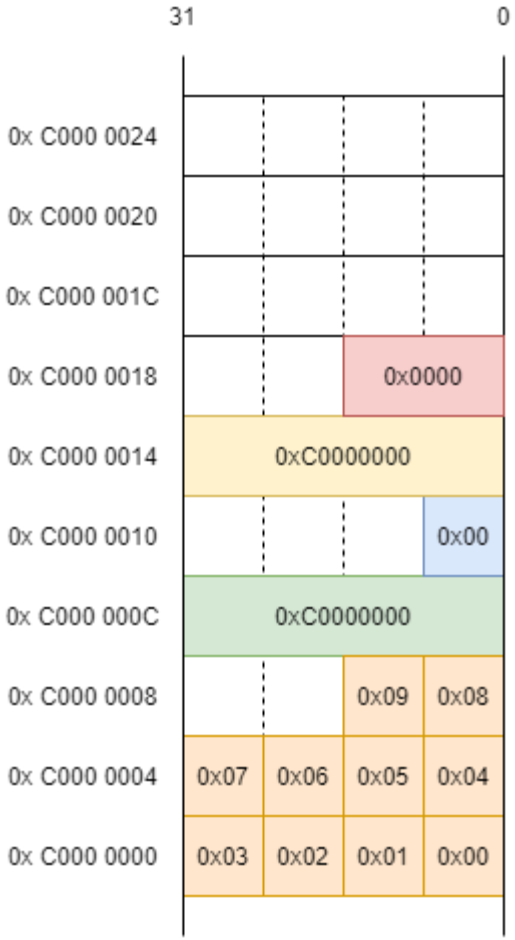
Image basics – Pointers




```
uint8_t data[10] = {0,1,2,3,4,5,6,7,8,9};
uint8_t *p = data; // alternative: &data[0]
uint8_t a = 0;
uint16_t *q = (uint16_t *)data;
uint16_t b = 0;
```

```
// Reading one element from the data array
a = data[3];   // a = 0x03
a = *(data+3); // a = 0x03
a = *(p+3);    // a = 0x03

// Reading two elements from the data array
b = *(q+3);    // b = 0x0706
```

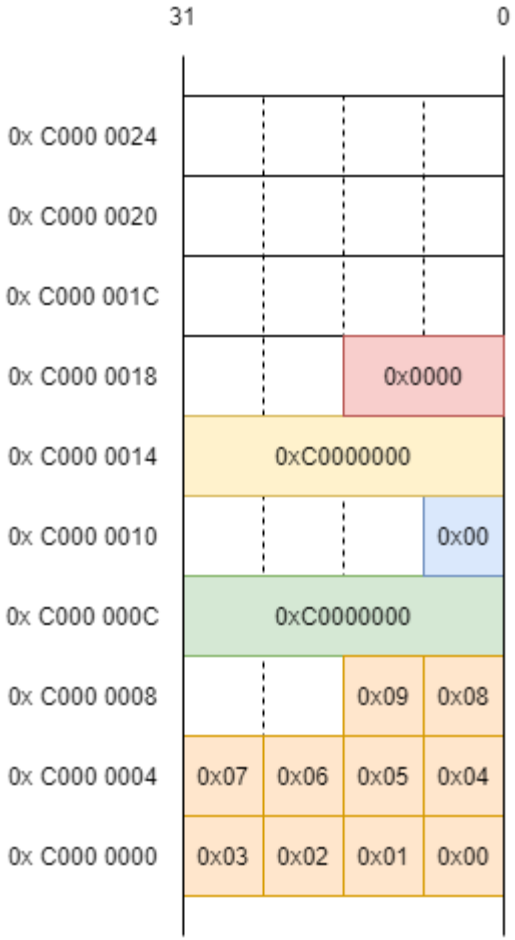
Image basics – Pointers




```
uint8_t data[10] = {0,1,2,3,4,5,6,7,8,9};
uint8_t *p = data; // alternative: &data[0]
uint8_t a = 0;
uint16_t *q = (uint16_t *)data;
uint16_t b = 0;
```

```
// Writing one element to the data array
p = data + 1; // p = 0xC000 0001
p++;          // p = 0xC000 0002
*p = 0;       // data = {0,1,0,3,4,5,6,7,8,9}

// Writing two elements to the data array
q = (uint16_t *)data + 1; // q = 0xC000 0002
q++;                      // q = 0xC000 0004
*q = 0;                   // data = {0,1,2,3,0,0,6,7,8,9}
```

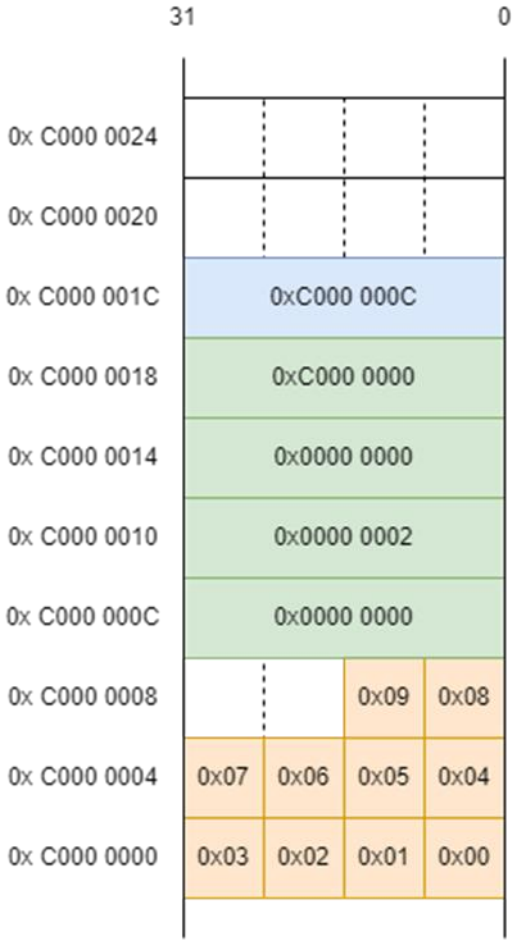
Image basics – Pointers




```
uint8_t data[10] = {0,1,2,3,4,5,6,7,8,9};
uint8_t *p = data; // alternative: &data[0]
uint8_t a = 0;
uint16_t *q = (uint16_t *)data;
uint16_t b = 0;
```

```
// Be careful with typecasting!
p = data + 1;                 // p = 0xC000 0001
q = (uint16_t *)data + 1;     // q = 0xC000 0002
q = (uint16_t *)(data + 1); // q = 0xC000 0001
```

Image basics – Pointers

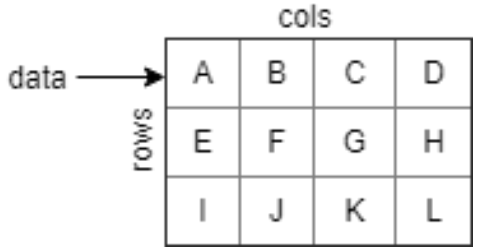



```
uint8_t data[10] = {0,1,2,3,4,5,6,7,8,9};
image_t image = {0,2,IMGTYPE_UINT8,data};
image_t *src = &image;
```

```
// Image manipulation
image.cols = 5; // image = {5,2,0,0xC000 0000}
src->cols = 5;  // image = {5,2,0,0xC000 0000}

*((uint8_t *)src->data + 3) = 0; // data={0,1,2,0,4,5,6,7,8,9}

*((uint16_t *)src->data + 3) = 0;// data={0,1,2,3,4,5,0,0,8,9}
```

Image basics – Accessing pixels

Use convenience functions for accessing pixels




```
inline void setUint8Pixel(const image_t *img, const int32_t c, const int32_t r, const uint8_pixel_t value)
{
    *((uint8_pixel_t *)(img->data) + (r * img->cols + c)) = value;
}
```

Explicit type cast to pixel type

Calculating the offset

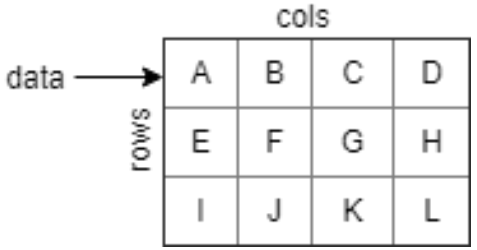
```
inline uint8_pixel_t getUint8Pixel(const image_t *img, const int32_t c, const int32_t r)
{
    return (*((uint8_pixel_t *)(img->data) + (r * img->cols + c)));
}
```

Explicit type cast to pixel type

Calculating the offset

Image basics – Accessing pixels

Use convenience functions for accessing pixels




```
inline void setFloatPixel(const image_t *img, const int32_t c, const int32_t r, const float_pixel_t value)
{
    *((float_pixel_t *)(img->data) + (r * img->cols + c)) = value;
}
```

Explicit type cast to pixel type

Calculating the offset

```
inline float_pixel_t getFloatPixel(const image_t *img, const int32_t c, const int32_t r)
{
    return (*((float_pixel_t *)(img->data) + (r * img->cols + c)));
}
```

Explicit type cast to pixel type

Calculating the offset

Image basics – Accessing pixels

```
// Create a new image
image_t *src = newUint8Image(4, 3);

// Clear the image
clearImage(src);

// Get the value of pixel B (1,0)
if(getUint8Pixel(src, 1, 0) == 0)
{
    // Set pixel G (2,1) to the value 100
    setUint8Pixel(src, 2, 1, 100);
}

// Cleanup
deleteUint8Image(src);
```

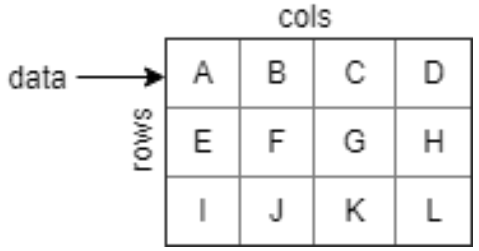
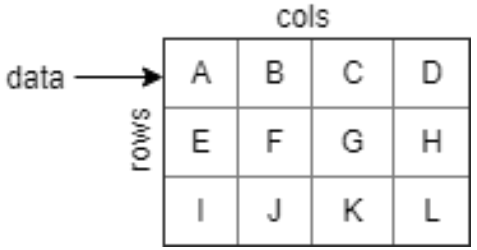


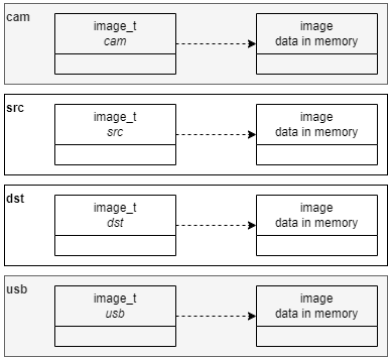
Image basics – Accessing pixels

```
// Create a new image
image_t *src = newFloatImage(4, 3);

// Clear the image
clearImage(src);

// Get the value of pixel B (1,0)
if(getFloatPixel(src, 1, 0) == 0)
{
    // Set pixel G (2,1) to the value 100
    setFloatPixel(src, 2, 1, 100);
}

// Cleanup
deleteFloatImage(src);
```



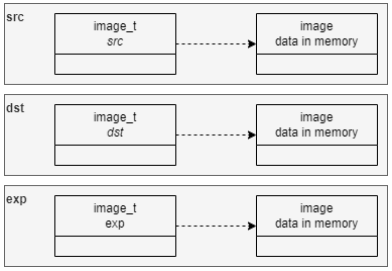
Anatomy of a project

MCUXpresso-IDE project



For running the image processing pipeline on the microcontroller

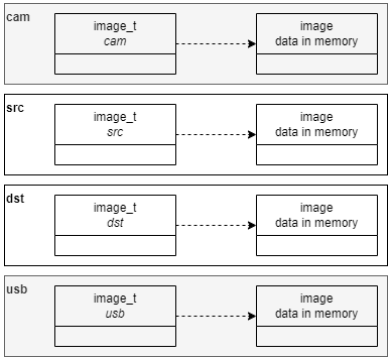
Qt & Unity unit test project



For unit testing the individual image processing operators

Anatomy of a project

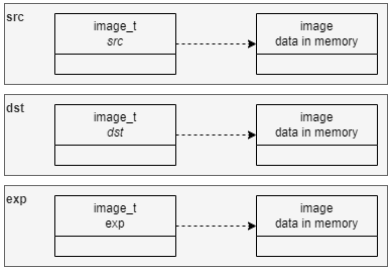
MCUXpresso-IDE project



For running the image processing pipeline on the microcontroller

No hardware?

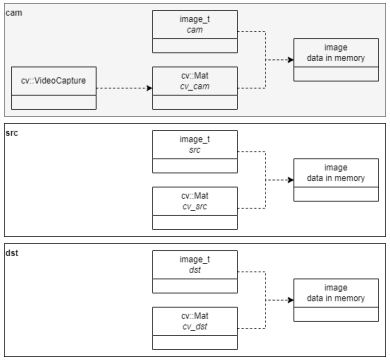
Qt & Unity unit test project



For unit testing the individual image processing operators

Anatomy of a project

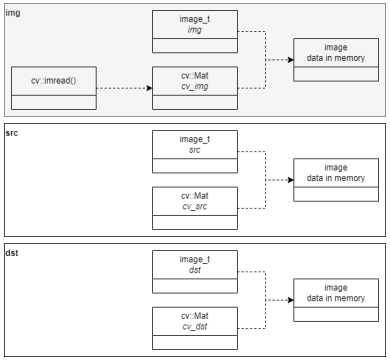
Qt webcam project



For running the image processing pipeline on your laptop with a webcam

No hardware?
No problem !

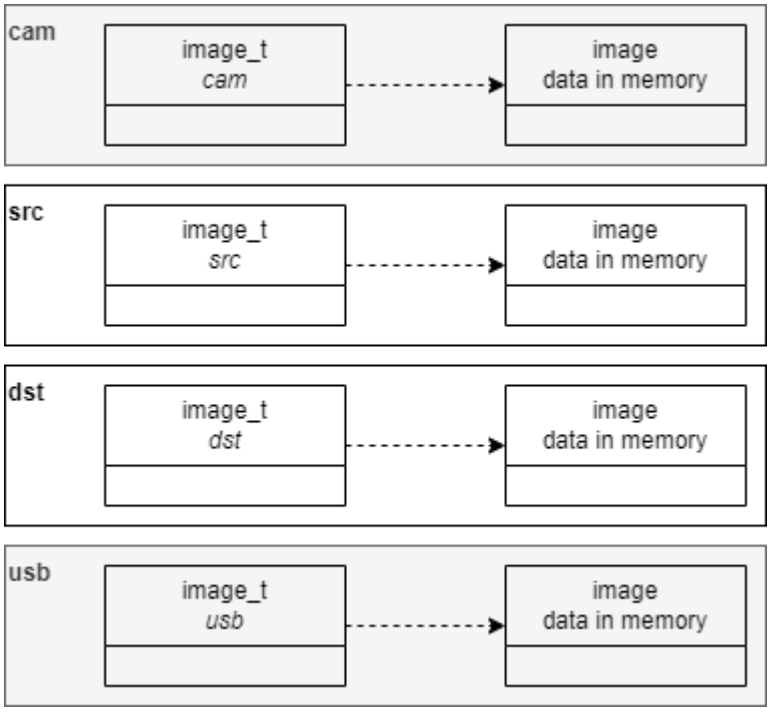
Qt file project



For running the image processing pipeline on your laptop with a file

Anatomy of an MCUXpresso-IDE project

Mandatory



Mandatory

Anatomy of an MCUXpresso-IDE project

```
image_t *cam = NULL;
image_t *usb = NULL;

int main(void)
{
    // ------------------------------------------------------------------------
    // Initialize the system
    // ------------------------------------------------------------------------
    systemInit();
```

Anatomy of an MCUXpresso-IDE project

```
    // ---------------------------------------------------------------------------
    // Select a single example
    // ---------------------------------------------------------------------------
//    exampleWebcamBgr888();
//    exampleWebcamUint8();
    exampleThreshold();

    // ---------------------------------------------------------------------------
    // Should never reach this
    // ---------------------------------------------------------------------------
    while(1U)
    {}
}
```

Anatomy of an MCUXpresso-IDE project

```
void systemInit(void)
{
    // ...

    // ---------------------------------------------------------------------------
    // Image memory allocation for static images required by camera and USB.
    // ---------------------------------------------------------------------------
    cam = newUyvyImage(EVDK5_WIDTH, EVDK5_HEIGHT);
    usb = newBgr888Image(EVDK5_WIDTH, EVDK5_HEIGHT);

    // ...
}
```

Anatomy of an MCUXpresso-IDE project

```
void exampleThreshold(void)
{
    // ---------------------------------------------------------------------
    // Local image memory allocation
    // ---------------------------------------------------------------------
    image_t *src = newUint8Image(EVDK5_WIDTH, EVDK5_HEIGHT);
    image_t *dst = newUint8Image(EVDK5_WIDTH, EVDK5_HEIGHT);

    while(1U)
    {
        // ---------------------------------------------------------------------
        // Wait for camera image complete
        // ---------------------------------------------------------------------
        while(smartdma_camera_image_complete == 0)
        {}

        smartdma_camera_image_complete = 0;
```

Anatomy of an MCUXpresso-IDE project

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

        threshold(src, dst, 0, 64);

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

Anatomy of an MCUXpresso-IDE project

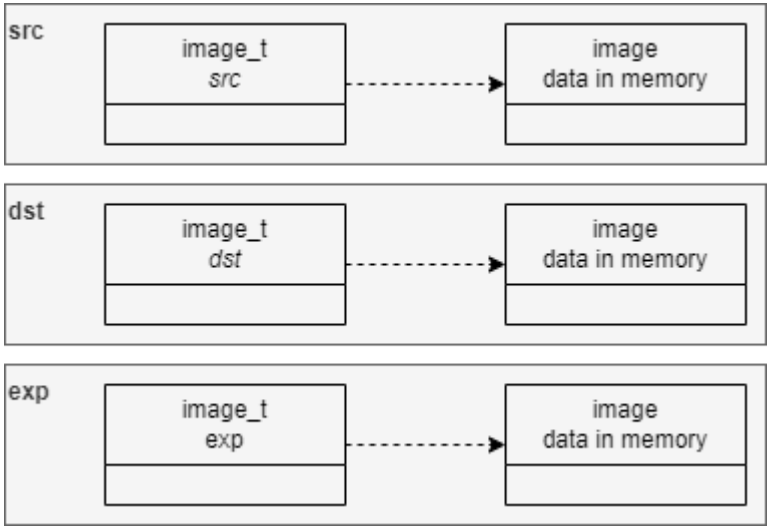
Demo

Anatomy of a Qt and Unity unit test project

Mandatory

Mandatory

Mandatory



Anatomy of a Qt and Unity unit test project

```
int main(void)
{
    UNITY_BEGIN();

    RUN_TEST(test_threshold);

    return UNITY_END();
}
```

Anatomy of a Qt and Unity unit test project

```
void test_threshold(void)
{
    // Prepare image data
    uint8_pixel_t src_data[4] = {0, 32,128,255}
    uint8_pixel_t dst_data[4] = {0,  0,  0,  0}
    uint8_pixel_t exp_data[4] = {1,  1,  0,  0}

    // Prepare images: src, dst and exp
    image_t src = {2, 2, IMGTYPE_UINT8, src_data};
    image_t dst = {2, 2, IMGTYPE_UINT8, dst_data};
    image_t exp = {2, 2, IMGTYPE_UINT8, exp_data};
```

Anatomy of a Qt and Unity unit test project

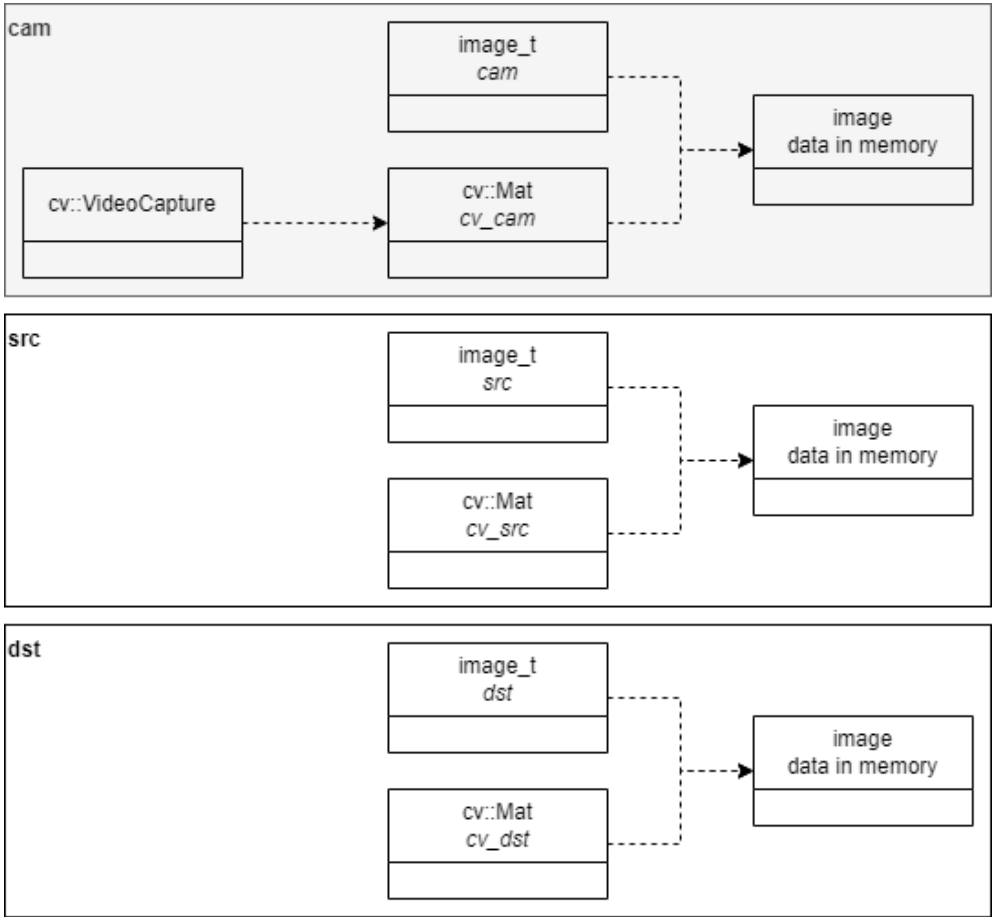
```
    // Execute test
    threshold(&src, &dst, 0, 64);

#if 0
     // Print image data for debugging
     prettyprint(&src, "src");
     prettyprint(&exp, "exp");
     prettyprint(&dst, "dst");
#endif

    // Verify the destination to expected result
    TEST_ASSERT_EQUAL(exp,dst);
}
```

Anatomy of a Qt and Unity unit test project

Demo

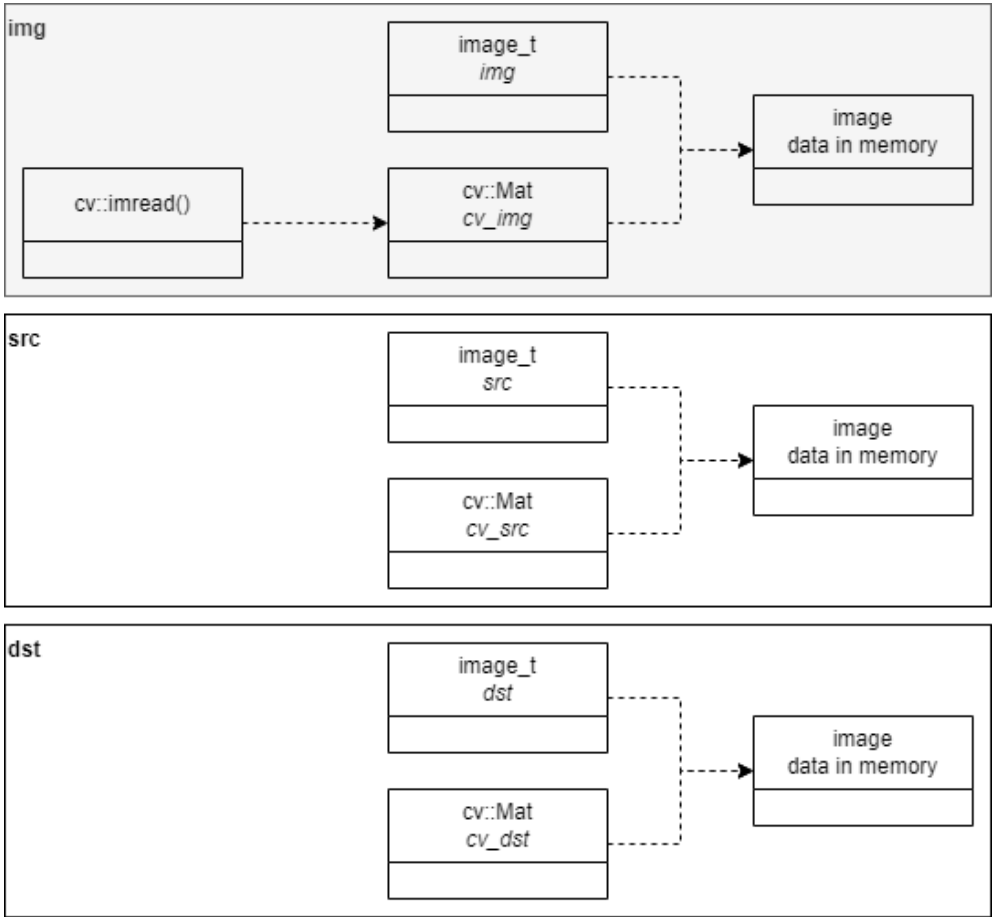
Anatomy of a Qt webcam project

Mandatory

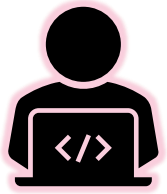


Anatomy of a Qt file project

Mandatory



EVD1 – Assignment



Study guide

Week 1

4 Change and run the MCUXpresso-IDE example project

5 Run Qt Webcam project

6 Run Qt and Unity unit test project