S32V234 ACF-Application Creation Demo

Dr. Anca Dima

April 2015

Prerequisites

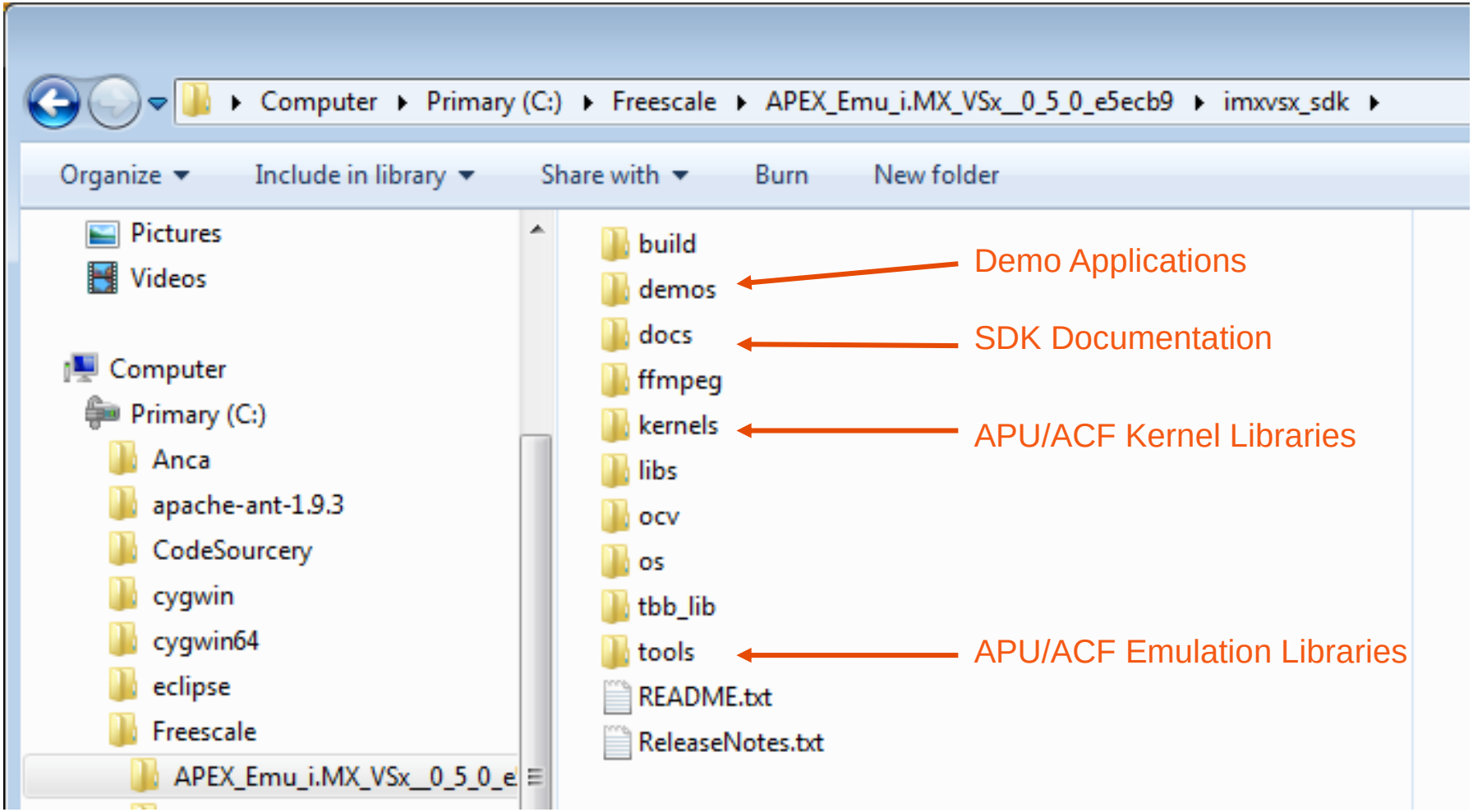
- Knowledge of the ACF and APU Programming Environment
- Knowledge of the APEX emulation library
- Knowledge of Visual Studio IDE

Overview

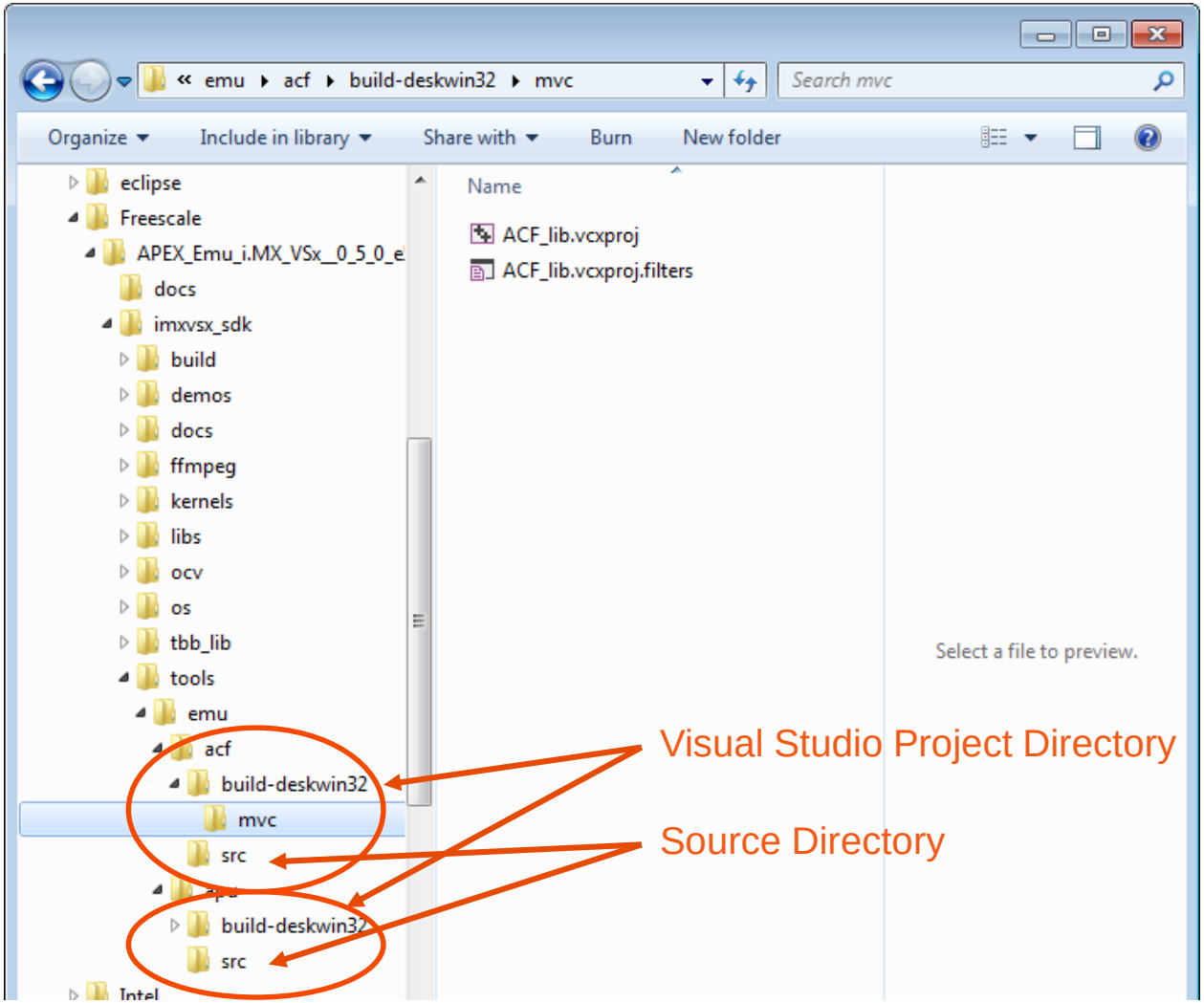
- Vision_SDK directory structure
- Current proposed structure for a Visual Studio project
- Setup of a new, custom Visual Studio project
- Filling the project with custom code
- Adding a new kernel library
- Adding the OpenCV library
- Cycle Counting

Vision_SDK directory structure

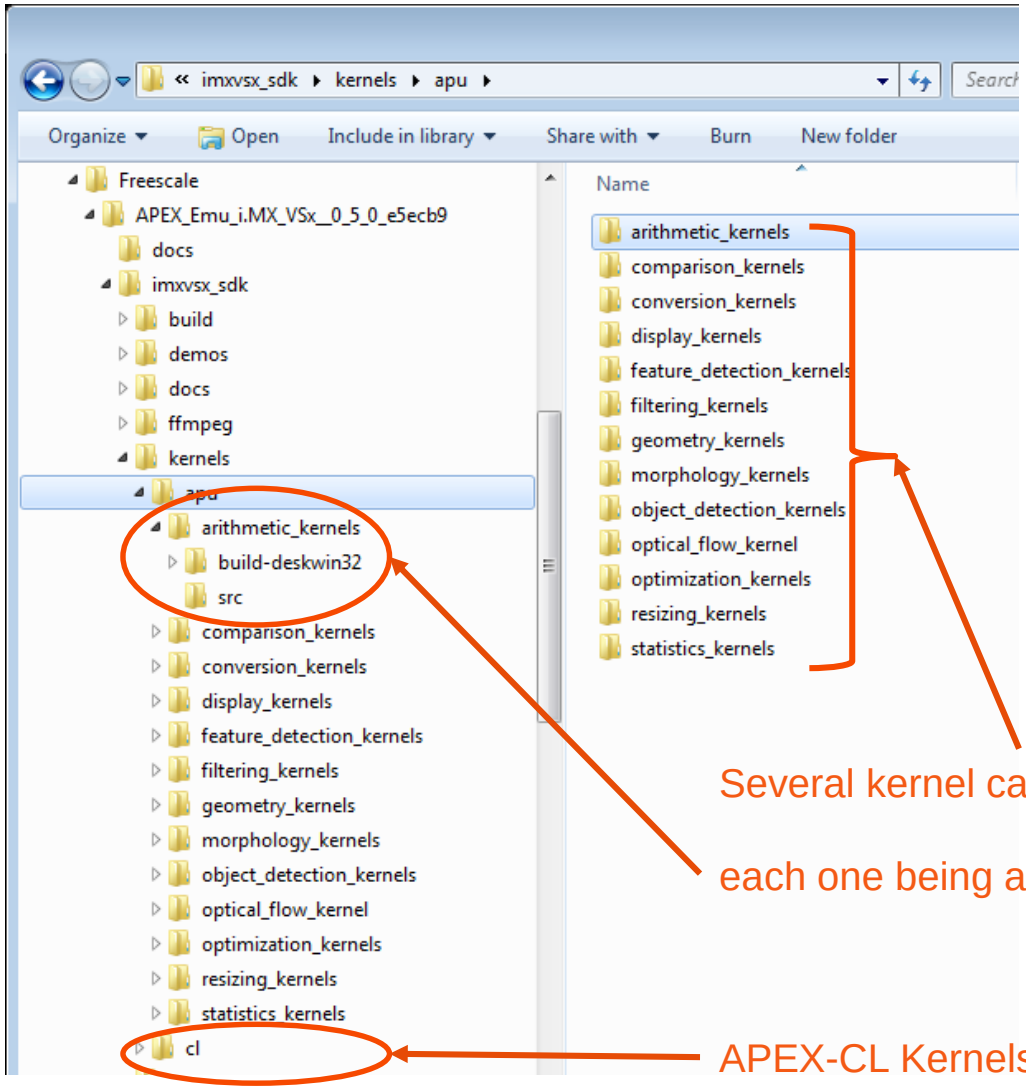
The Vision SDK directory structure



APU/ACF Libraries



The Kernels Libraries

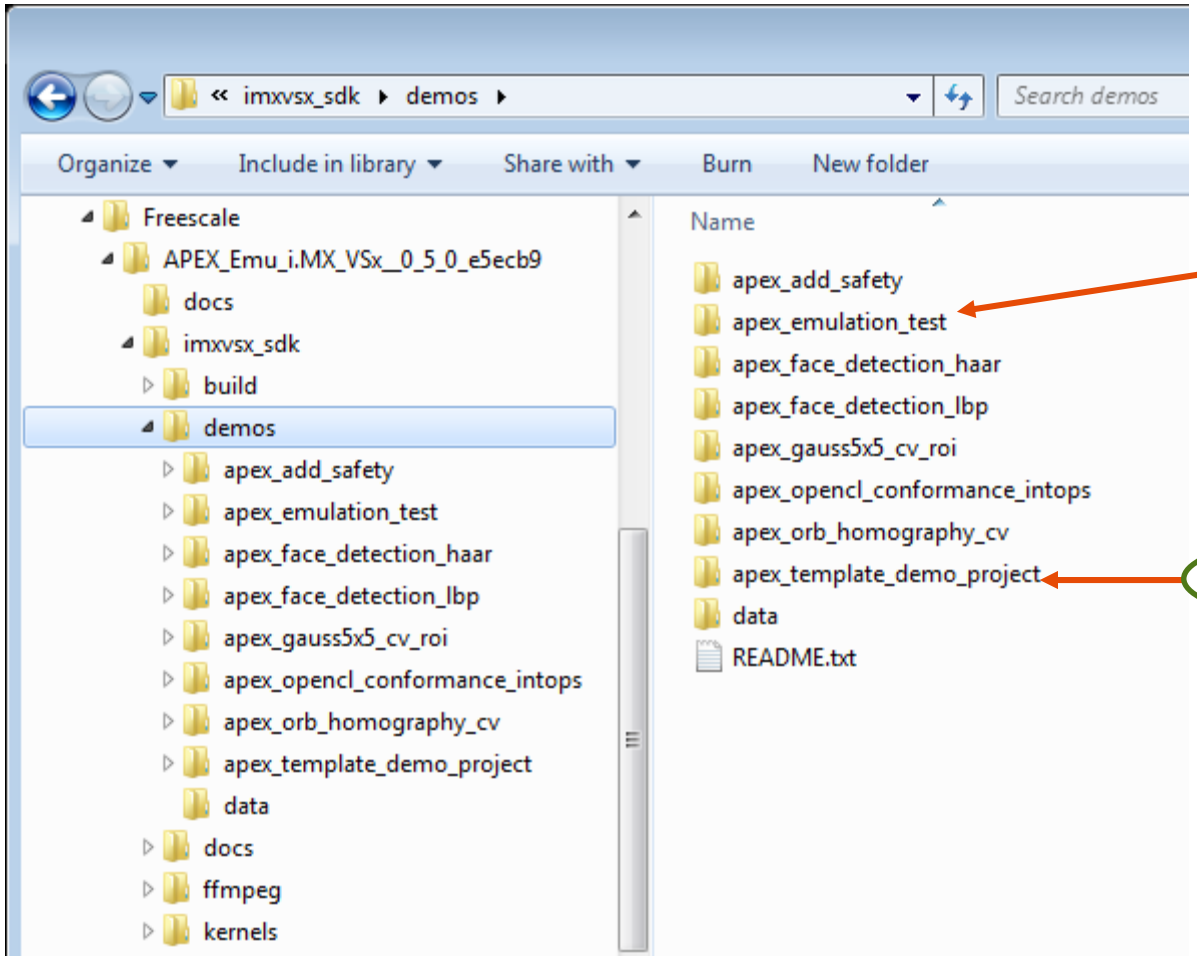


Several kernel categories,

each one being a separate library

APEX-CL Kernels

The Demos Directory



Several Kernels‘ Demo

Simplest Starting Demo

The apex_template_demo_project

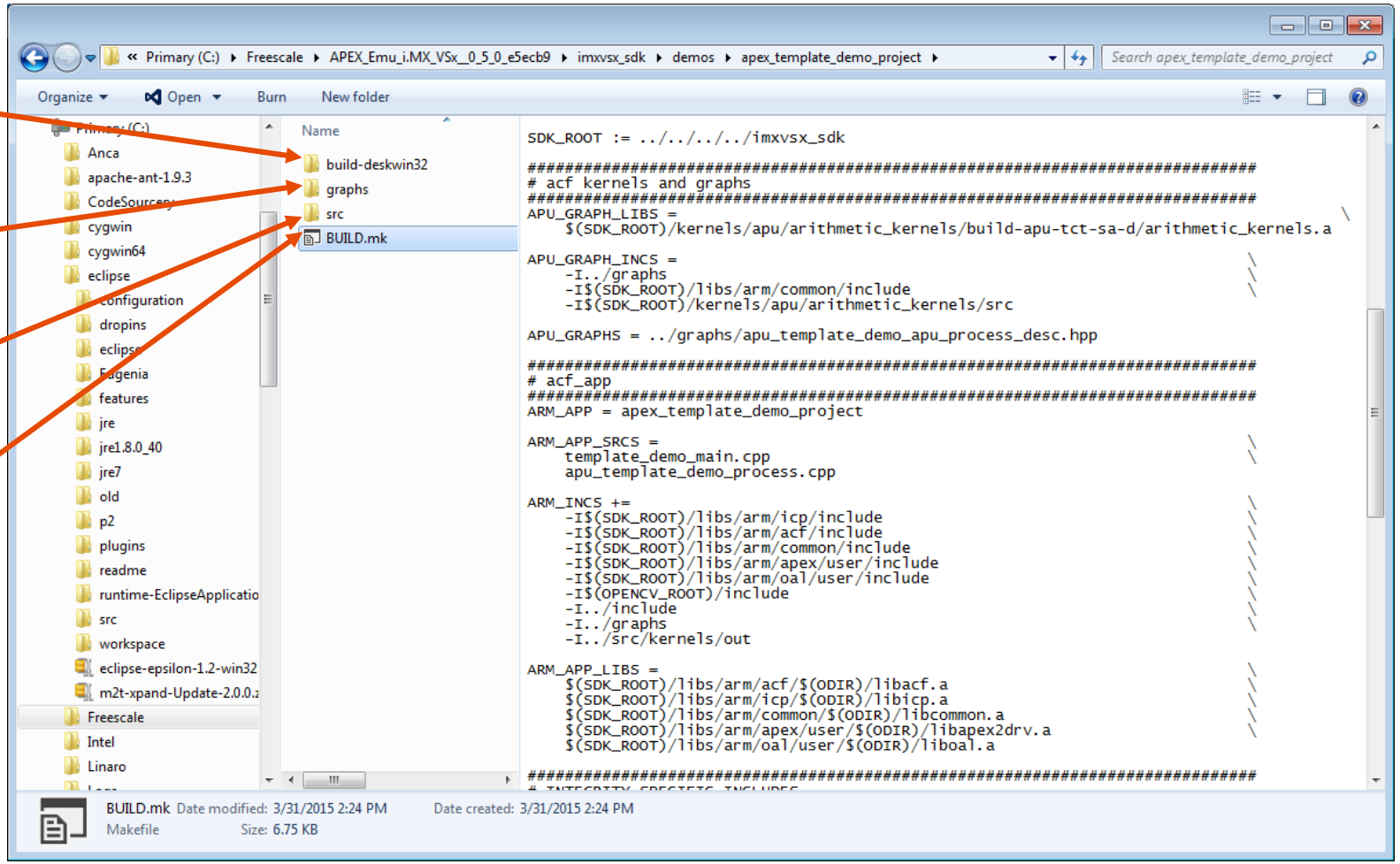
The apex_template_demo_project

Visual Studio project

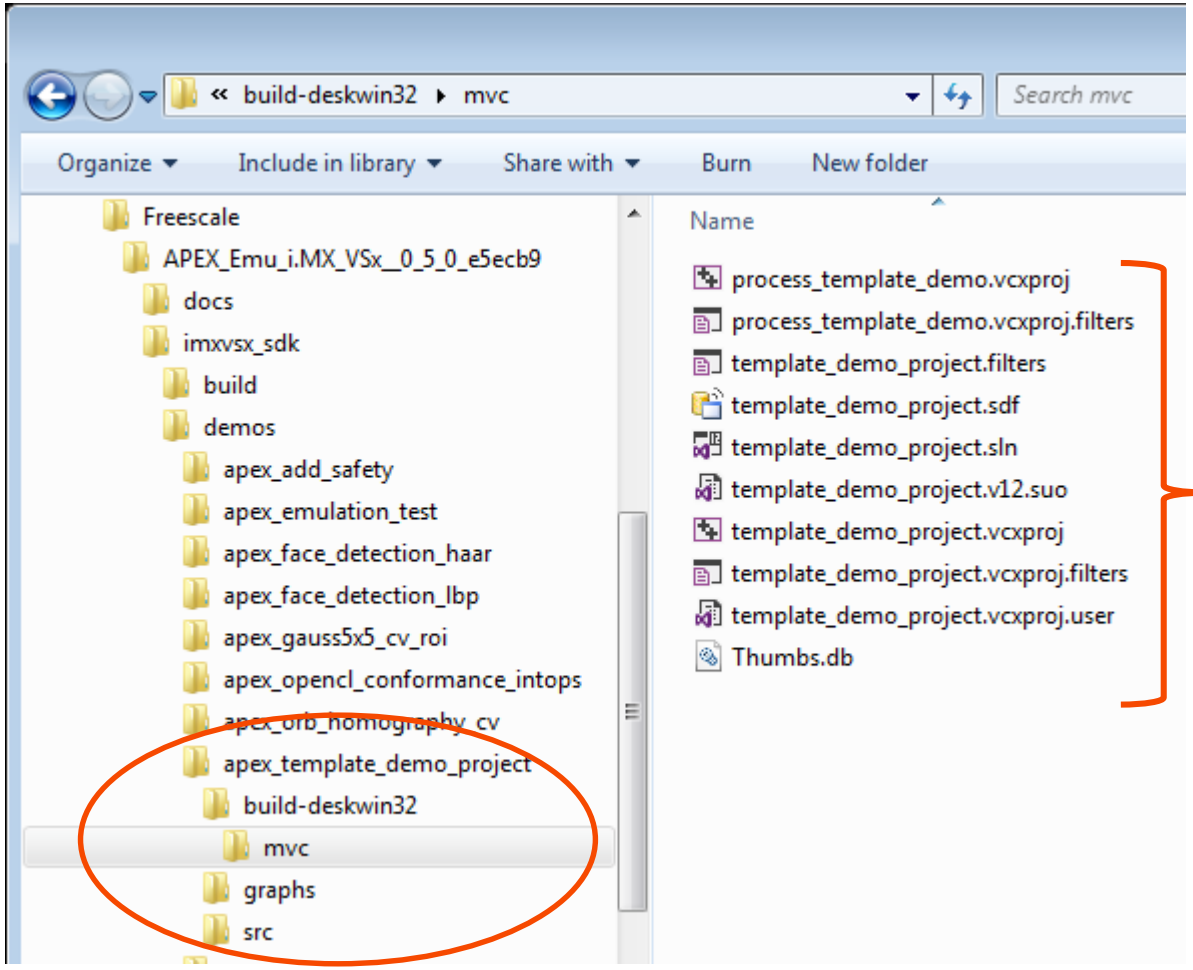
Own graphs & processes

Main file & other sources

Linux build file

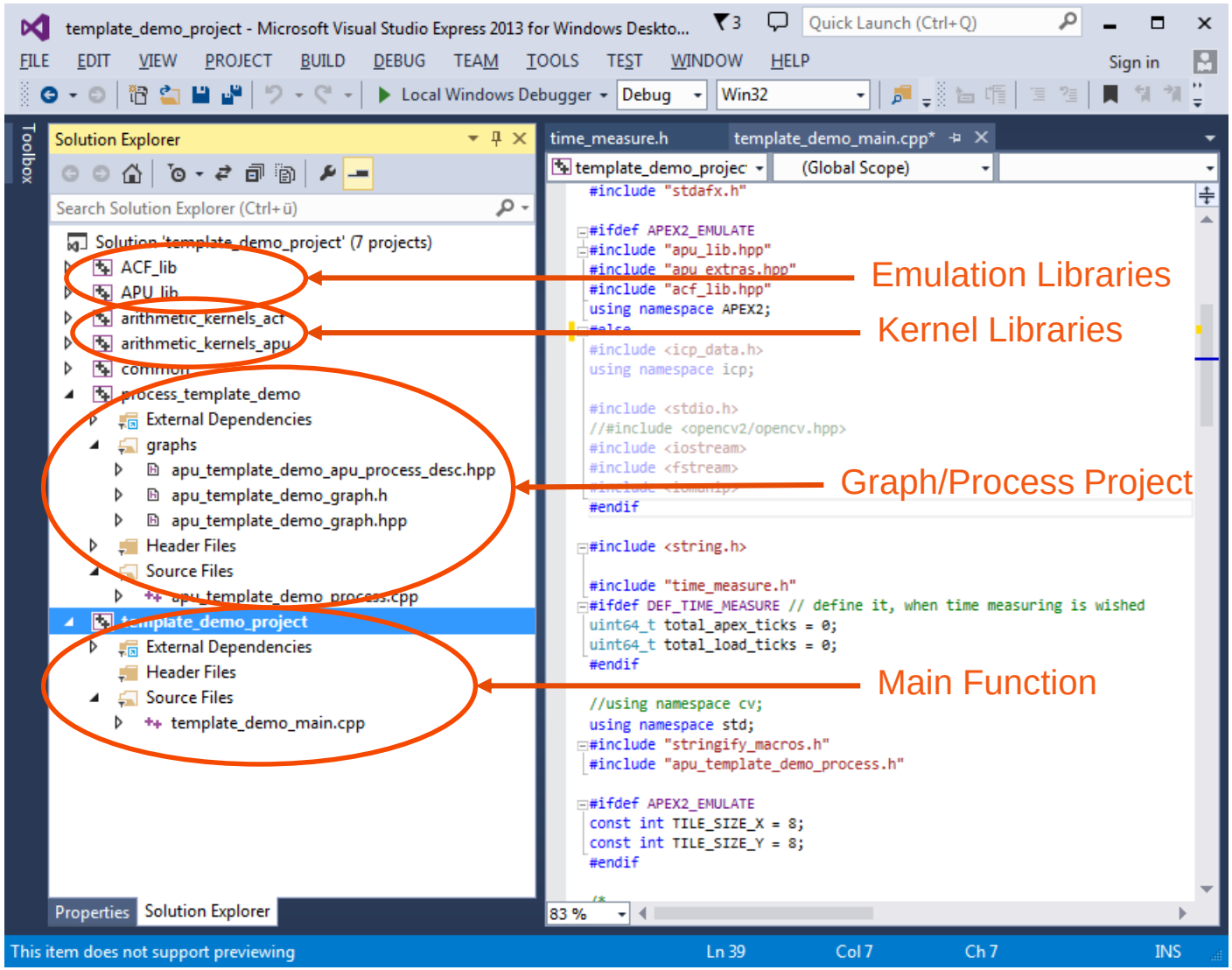


The apex_template_demo_project project directory



The Visual Studio Solution & Projects

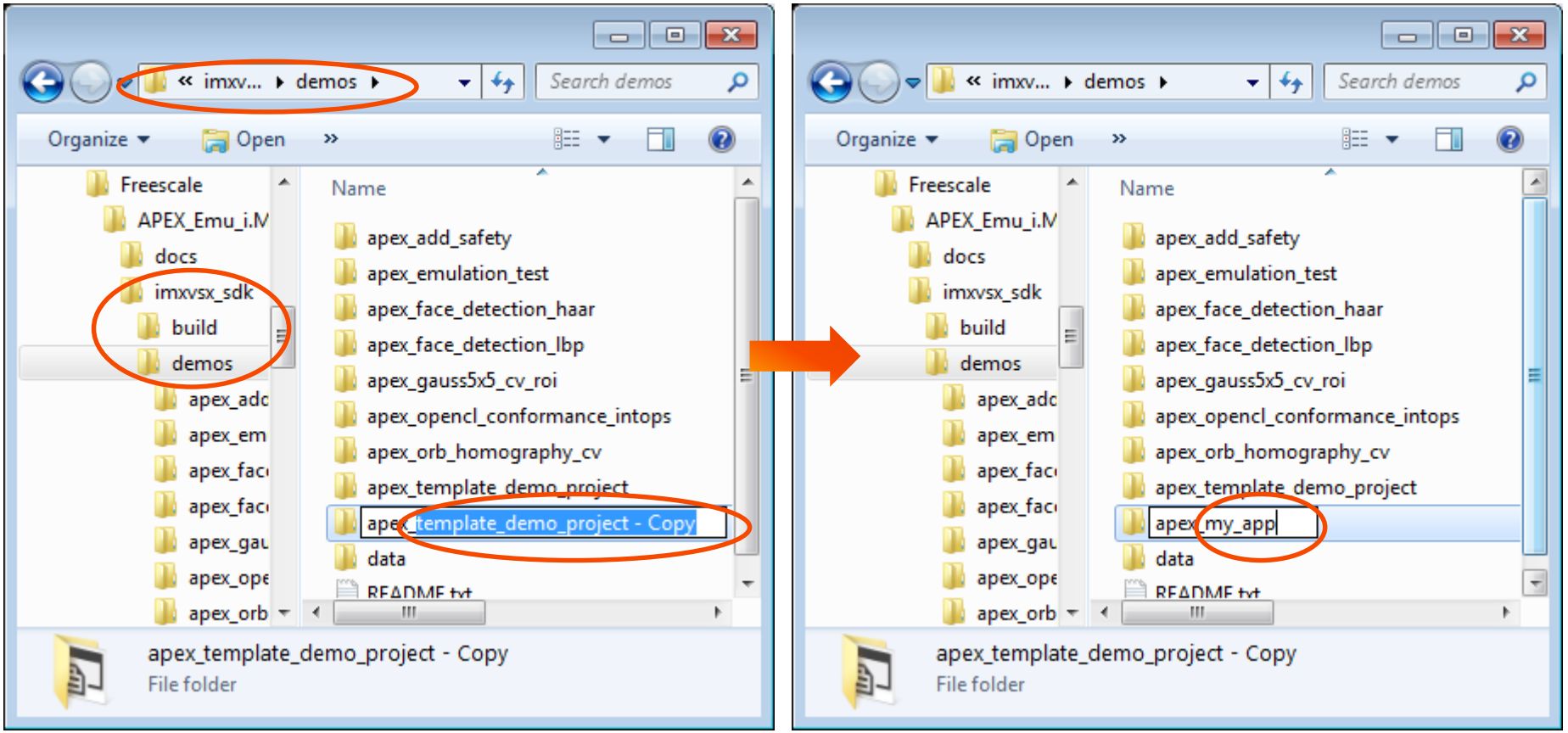
The Visual Studio Solution



Setup of a New, Custom Visual Studio Project

Setup of a New, Custom Visual Studio Project

- Copy the apex_template_demo_project directory and rename it to your own name
  (**CAUTION**: the prefix „apex_“ has to be kept)

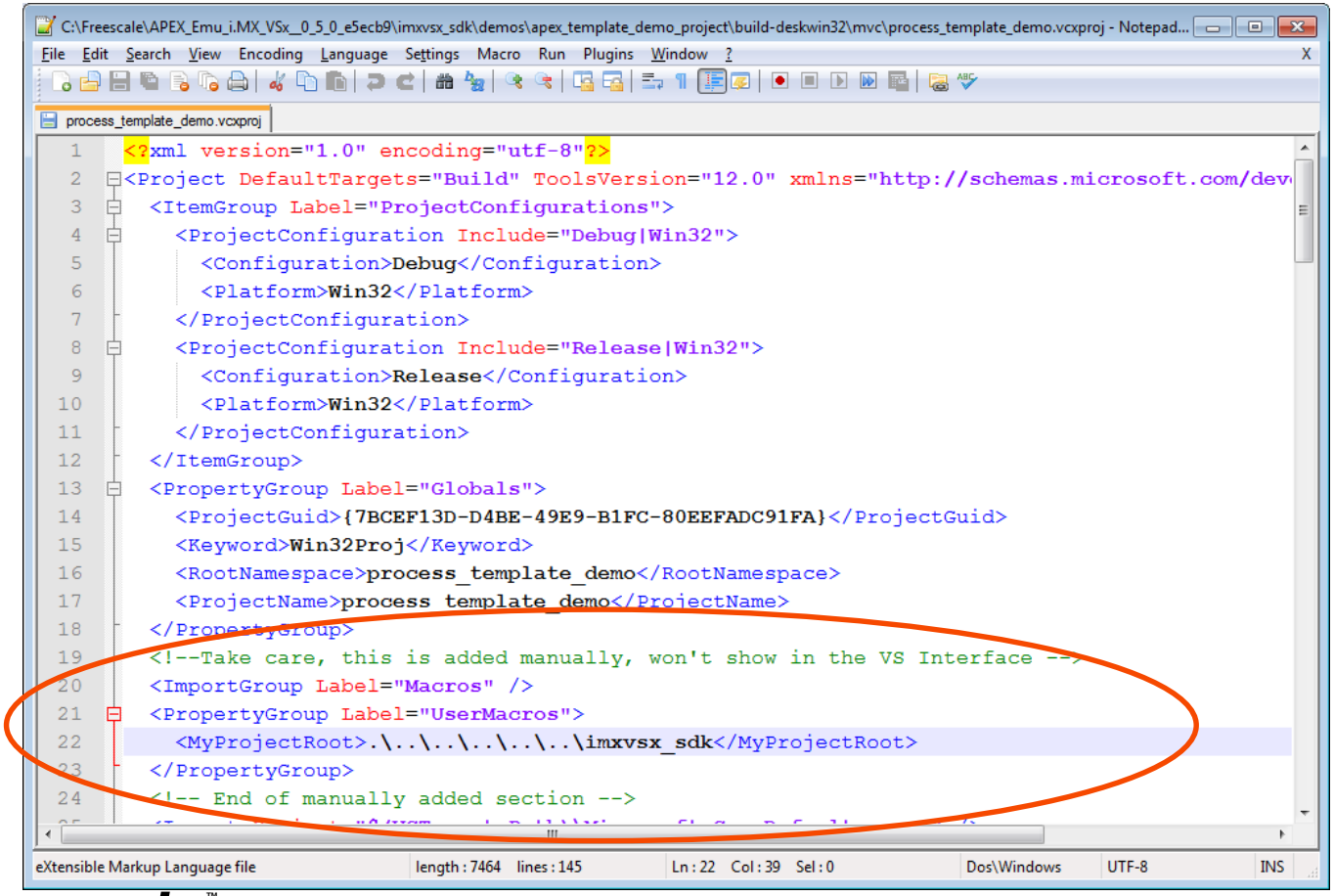


Motivation for the Copy Operation

Each project file contains a manually added section for a macro variable called „MyProjectRoot“ which points to the s32v234_sdk directory.

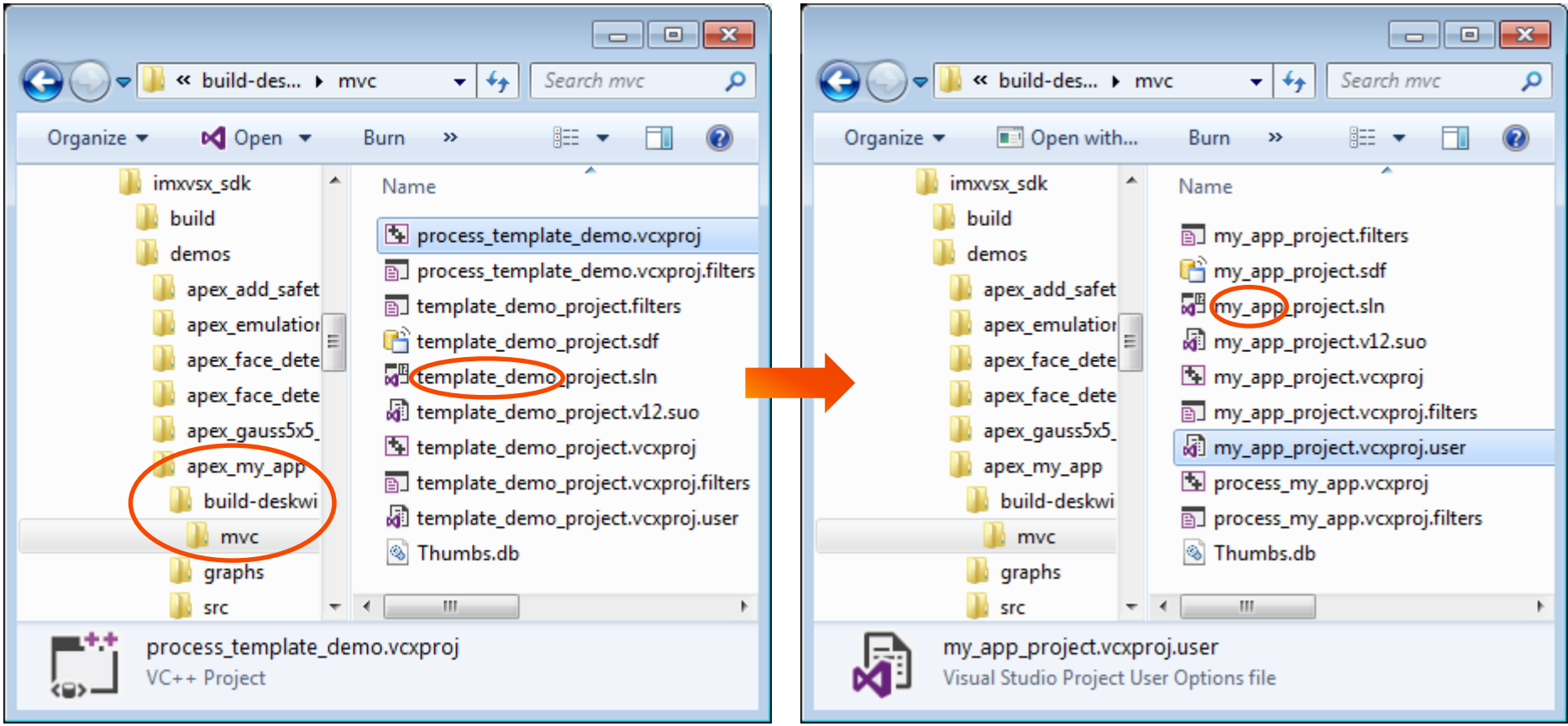
This is needed for the references to the other dependencies.

!!! For newly created projects (not copied), this section has to be added to each *.vcxproj file as below!!!



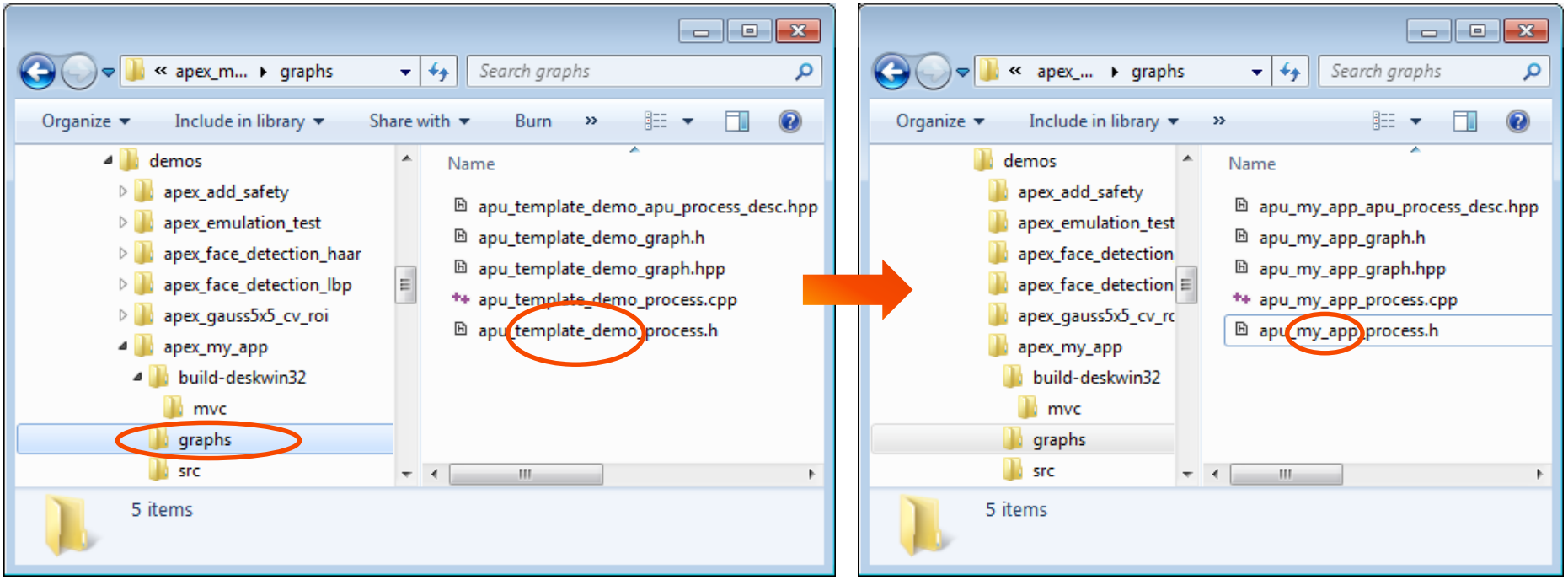
Rename all Projects

- Replace substrings „template_demo“ with „my_app“ in all project file names (*.sln, *.vcxproj*, *.filters)



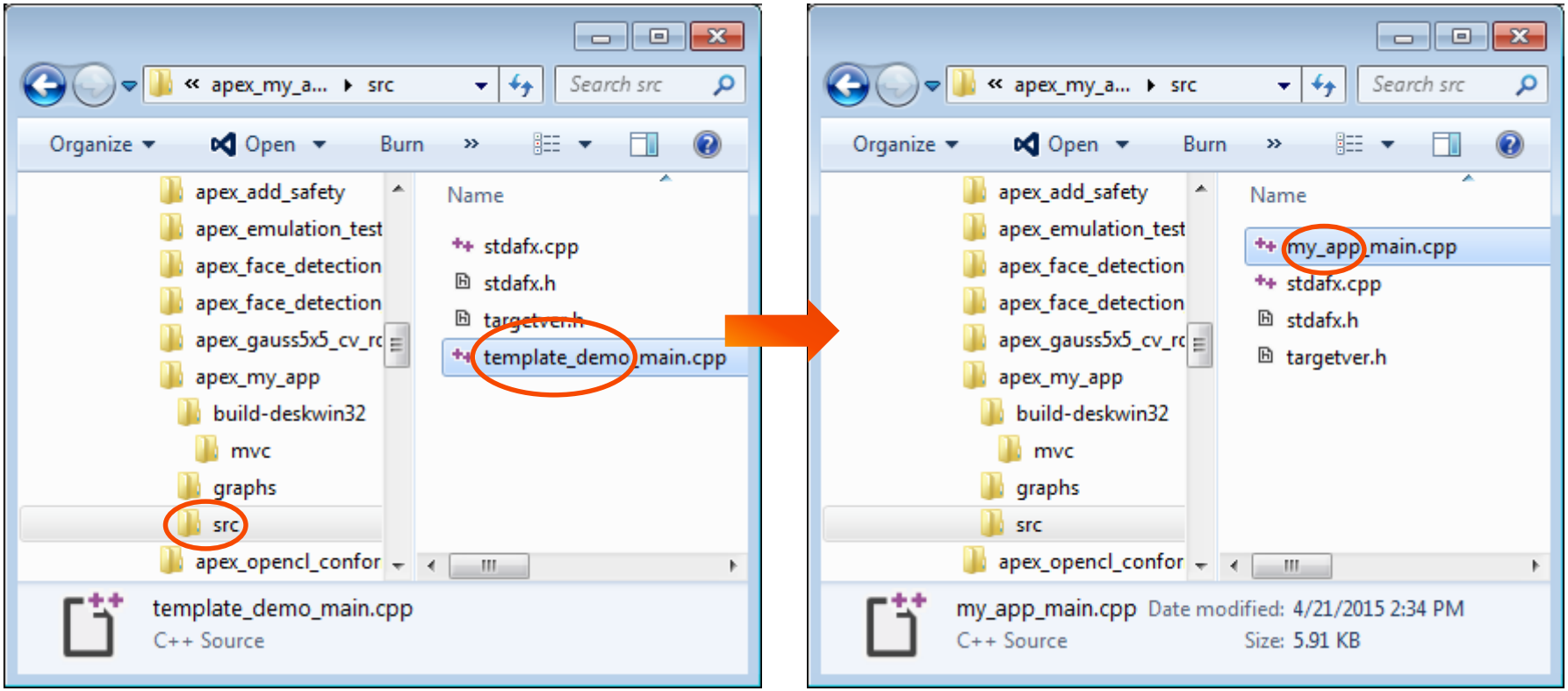
Rename all Graphs

- Replace substrings „template_demo“ with „my_app“ in all graph file names and the main file



Rename the Main File

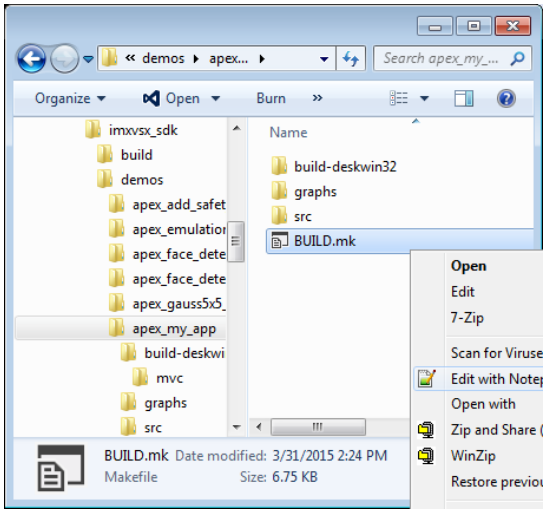
- Replace substrings „template_demo“ with „my_app“ in the main file

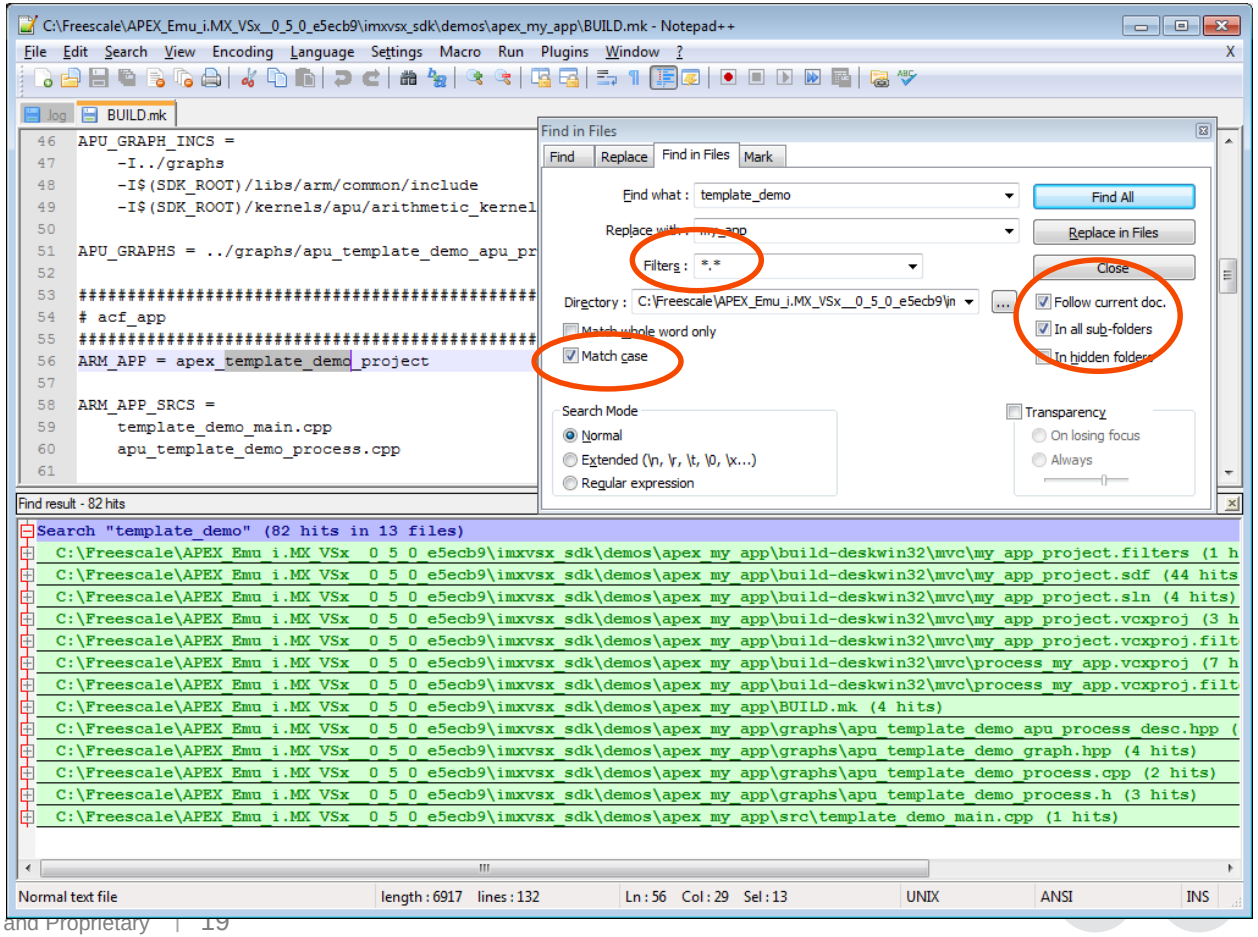


Filling the Project with Custom Code

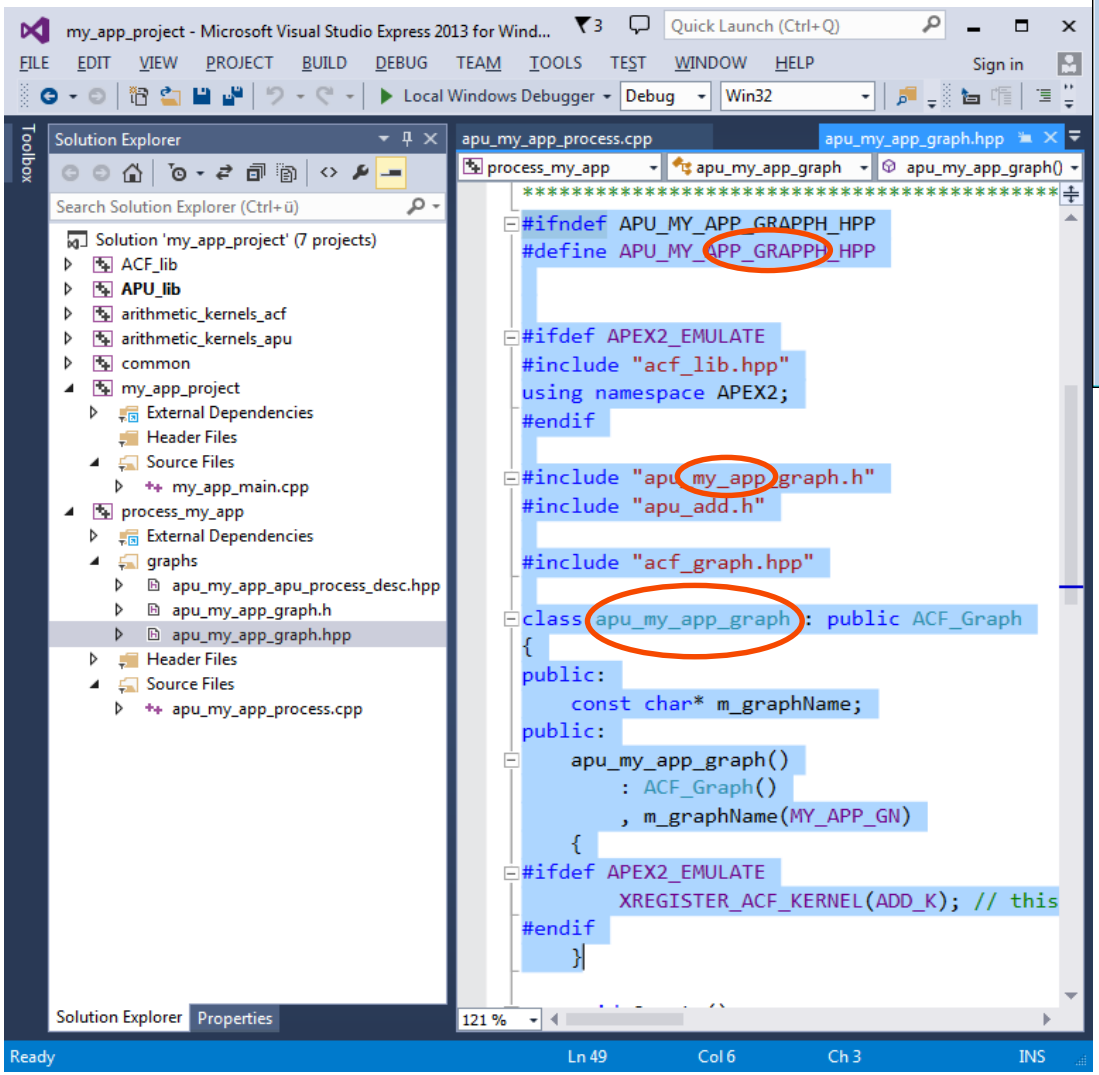
Filling the Project with Custom Code

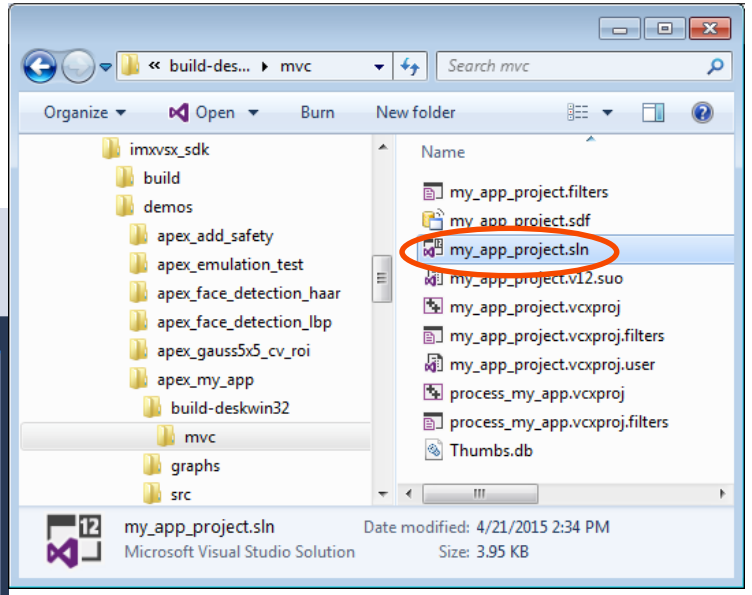
- Load the Build.mk file into a Notepad++ and replace case sensitive following strings in all files in the current directory and in all subdirectories:
  - „template_demo“ with „my_app“ (for the variables and includes)
  - „TEMPLATE_DEMO“ with „MY_APP“ (for the header defines)
  - „Template_Demo“ with „My_App“ (for the class names)





Load the „my_app“ Solution





Items replaced by Notepad++:

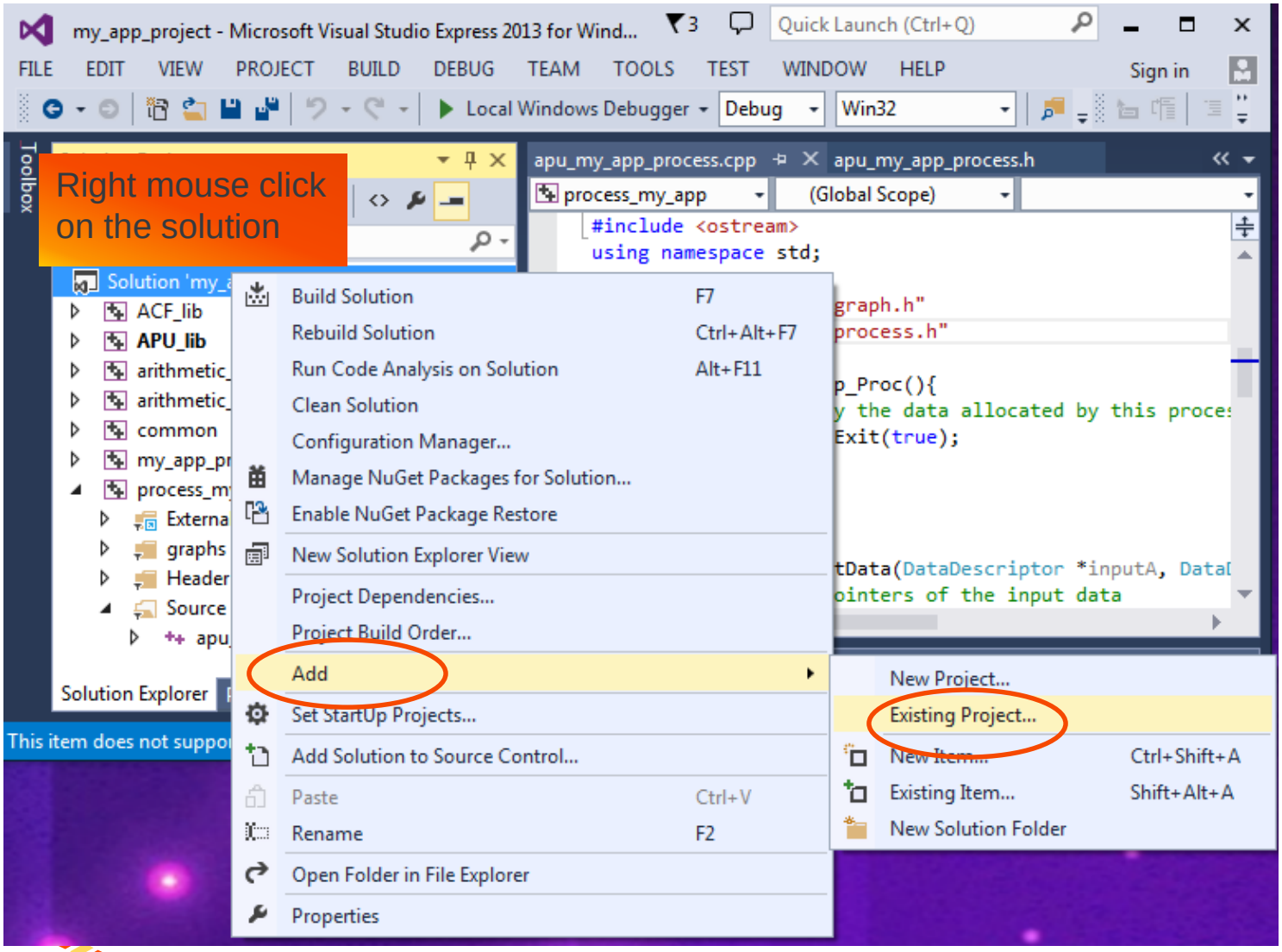
- project dependencies
- class names ➔ 3 classes
- includes
- class instantiations

➔ New Project containing all dependencies,

obtained by a few file renames and three string replacements.

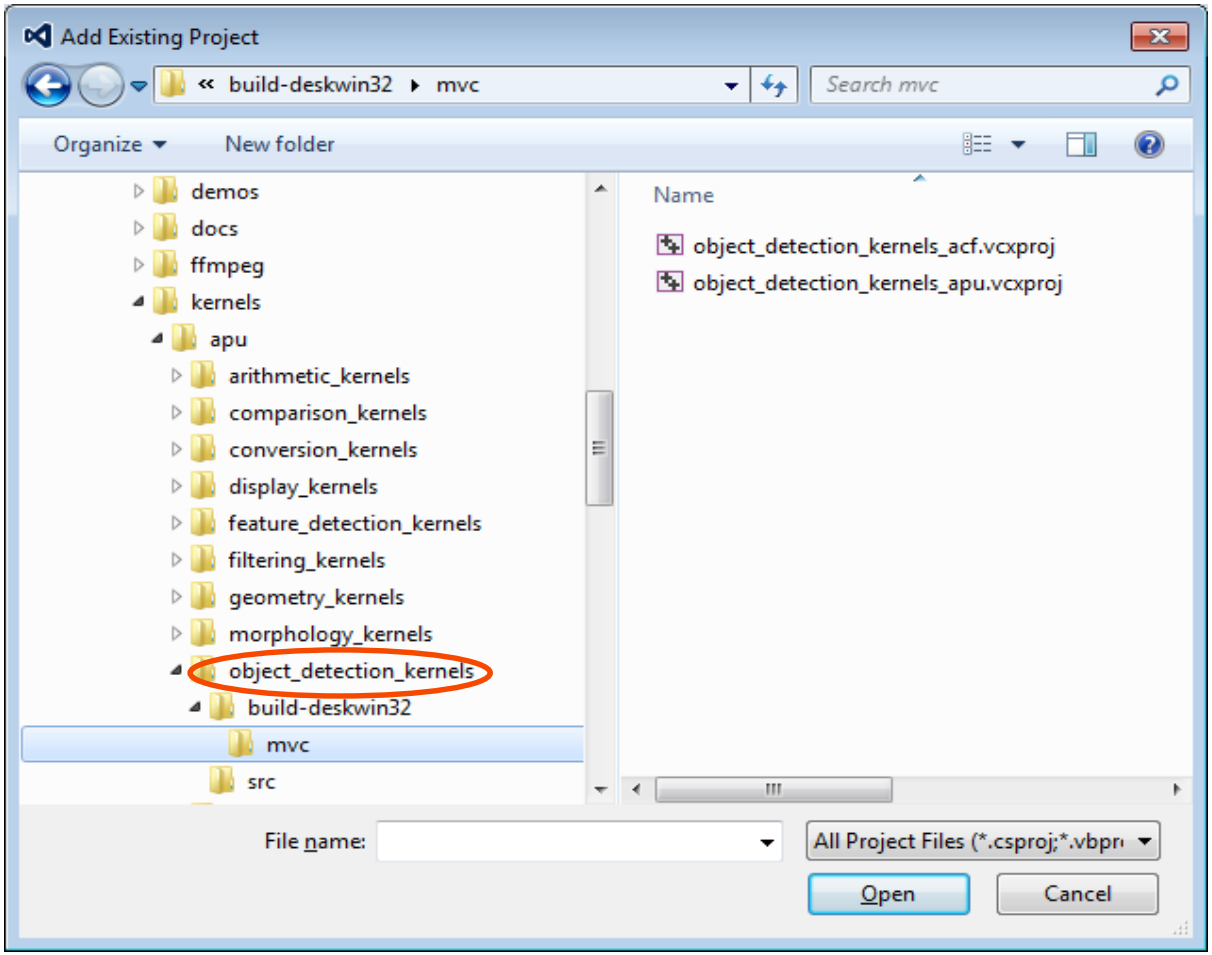
Adding a New Kernel Library

Adding a new Kernel Library



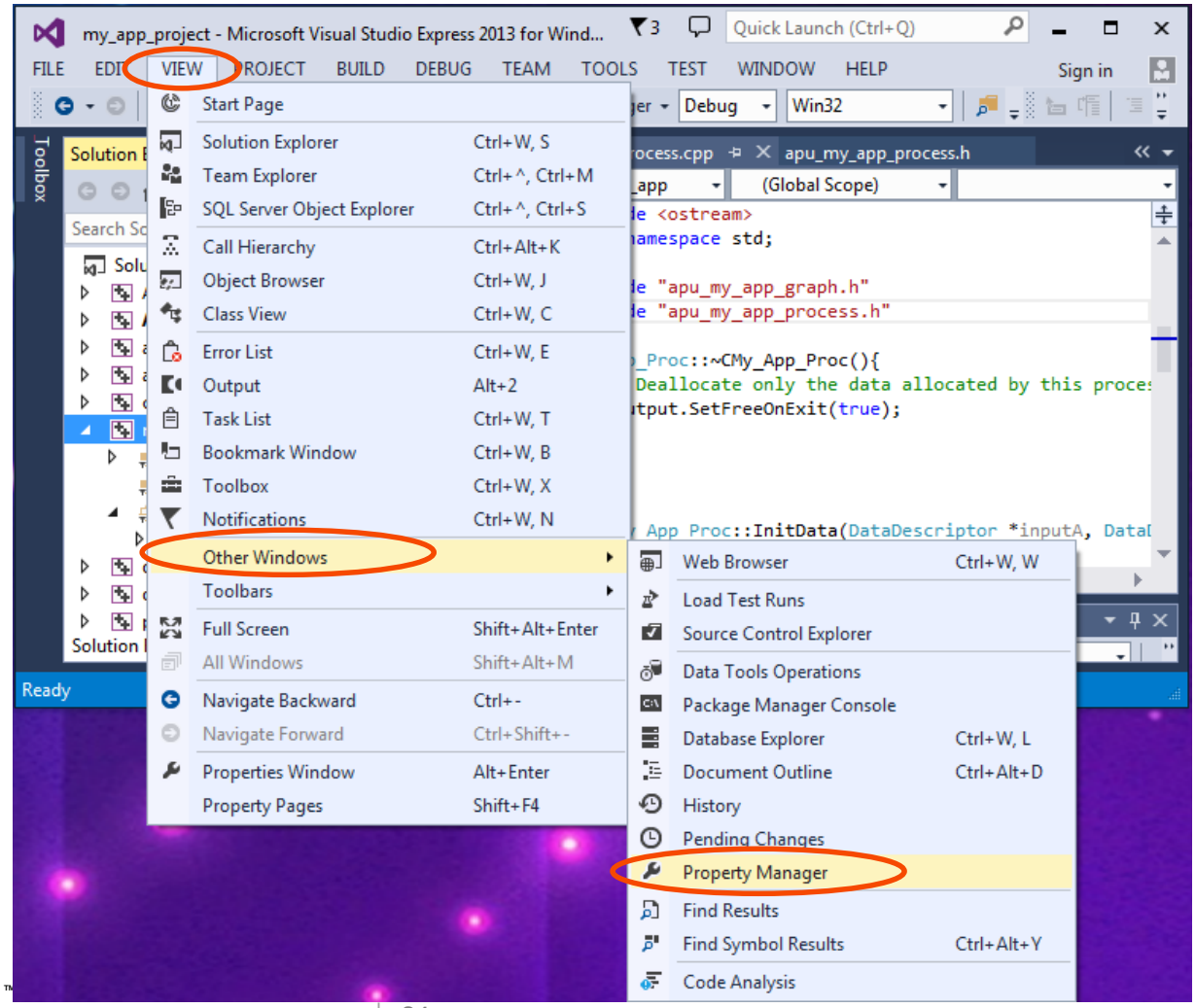
Choose the Desired Kernel Library

- Load both the *_acf.vcxproj and the *_apu.vcxproj into the solution

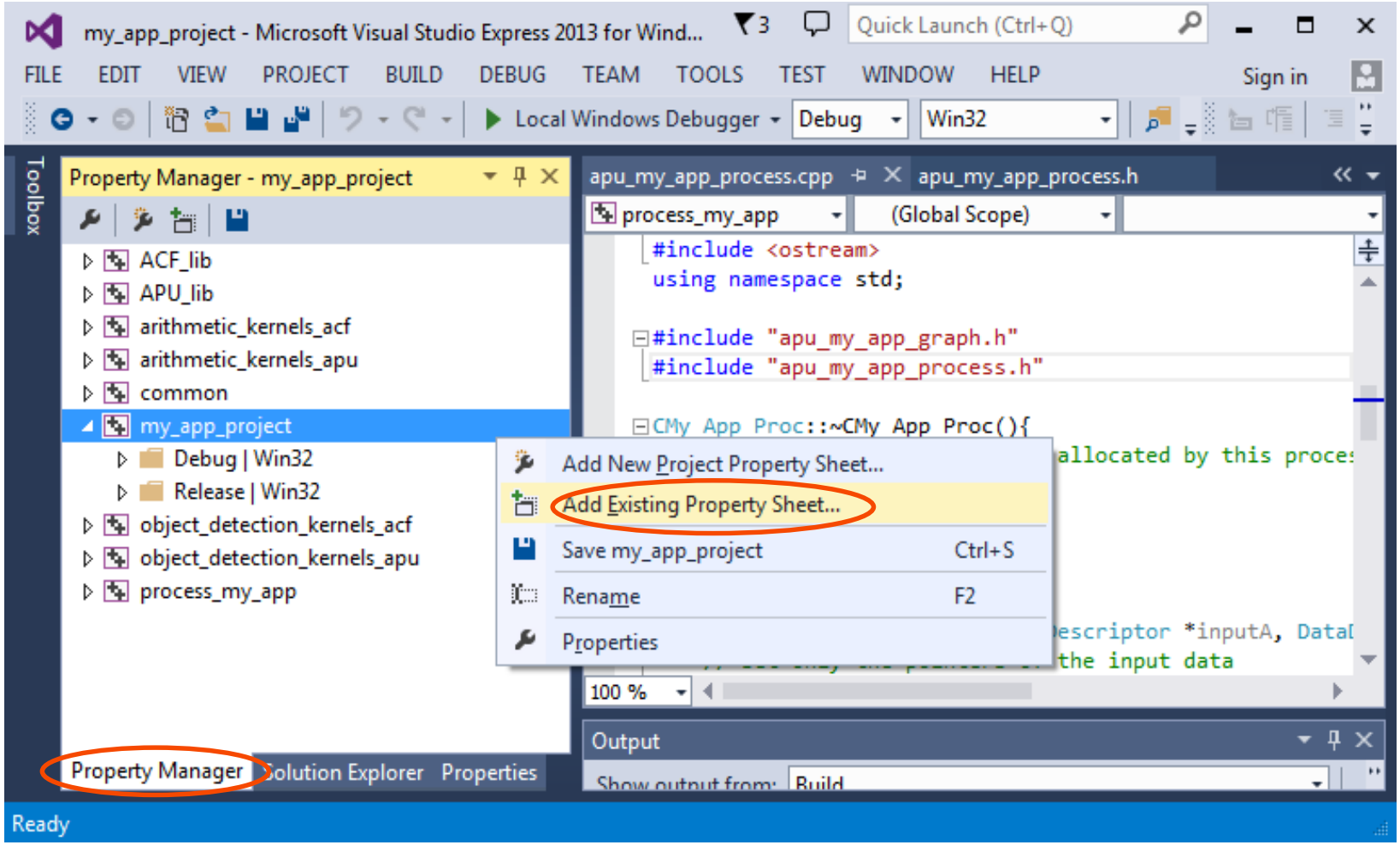


Add to the Main Project the Kernel‘s Property Sheet

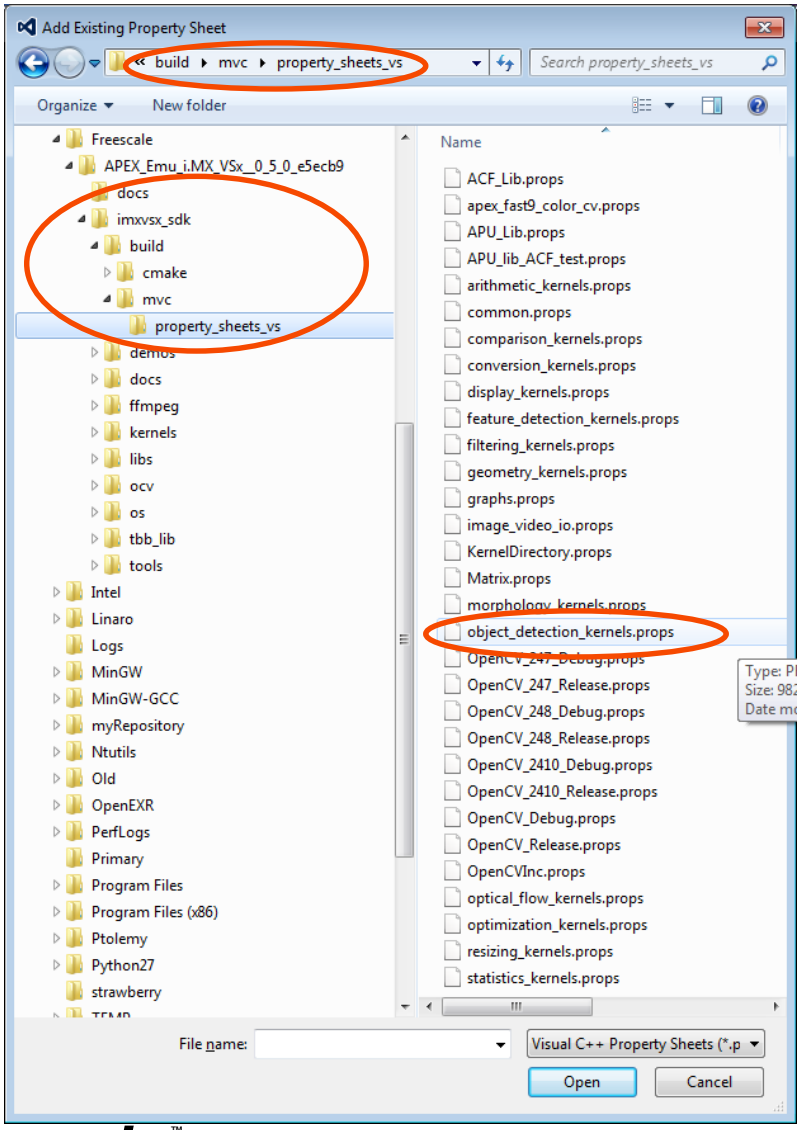
- Show the Property Manager



Add to the Main Project the Kernel‘s Property Sheet



Add to the Main Project the Kernel‘s Property Sheet

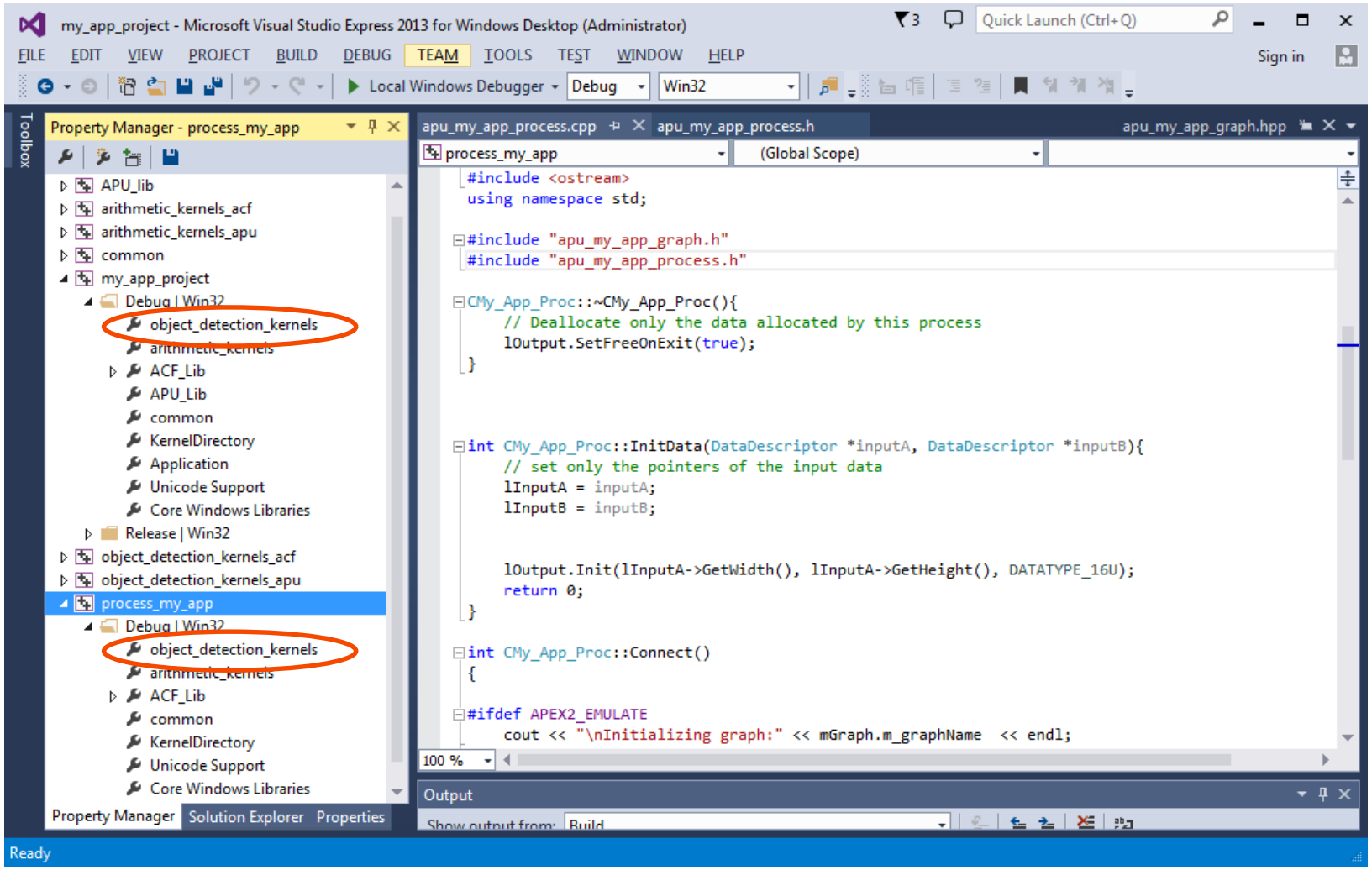


- Kernel property sheets contain the include path for the respective library
- Property sheets for the kernels and other libraries are in folder:

  s32v234_sdk/build/mvc/property_sheets_vs

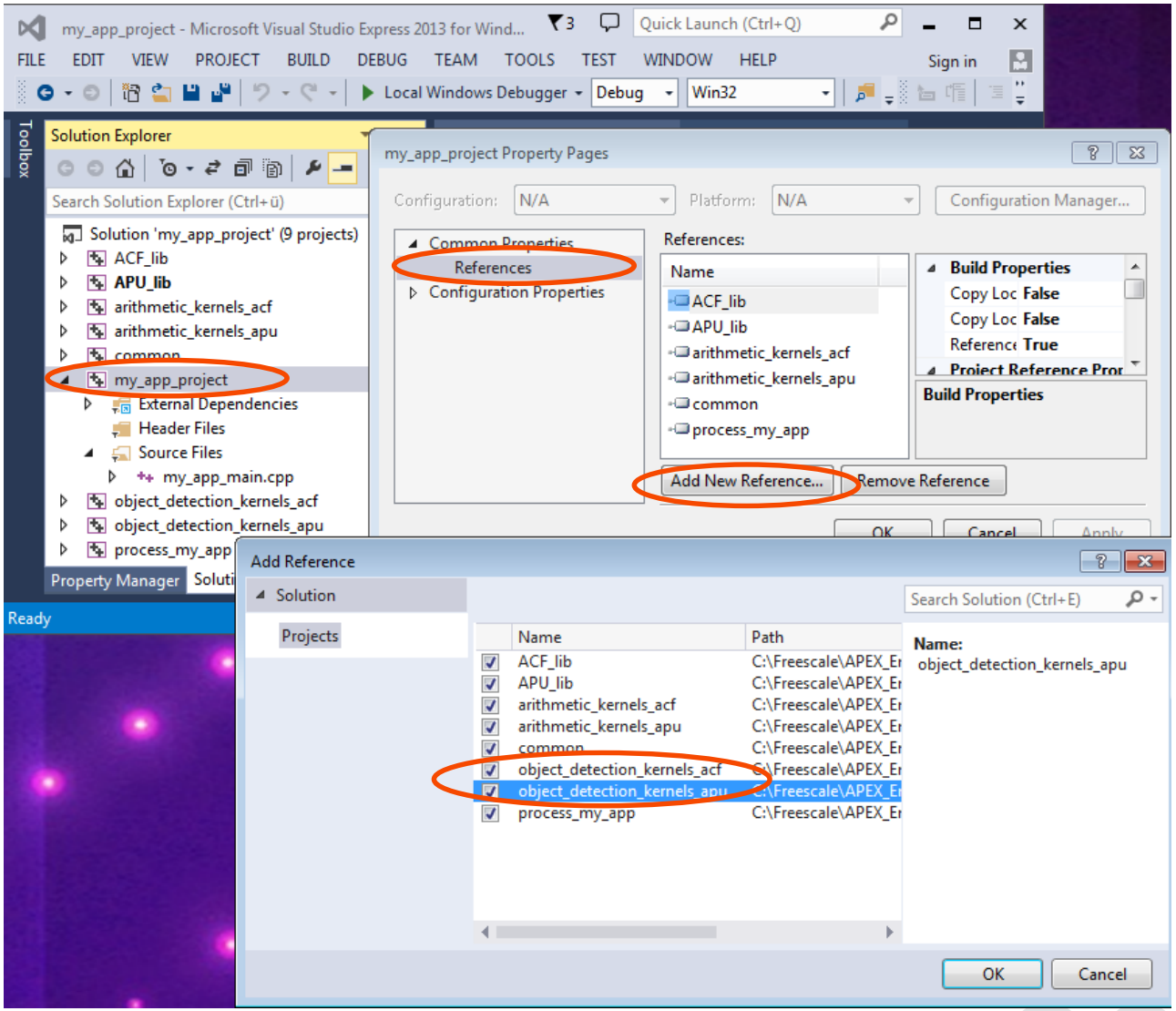
- Repeat the same two steps for all process projects, in which graphs use a kernel from the chosen kernel library

The Property Manager will look like this



Add to the Main Project the Kernel Library‘s Dependency

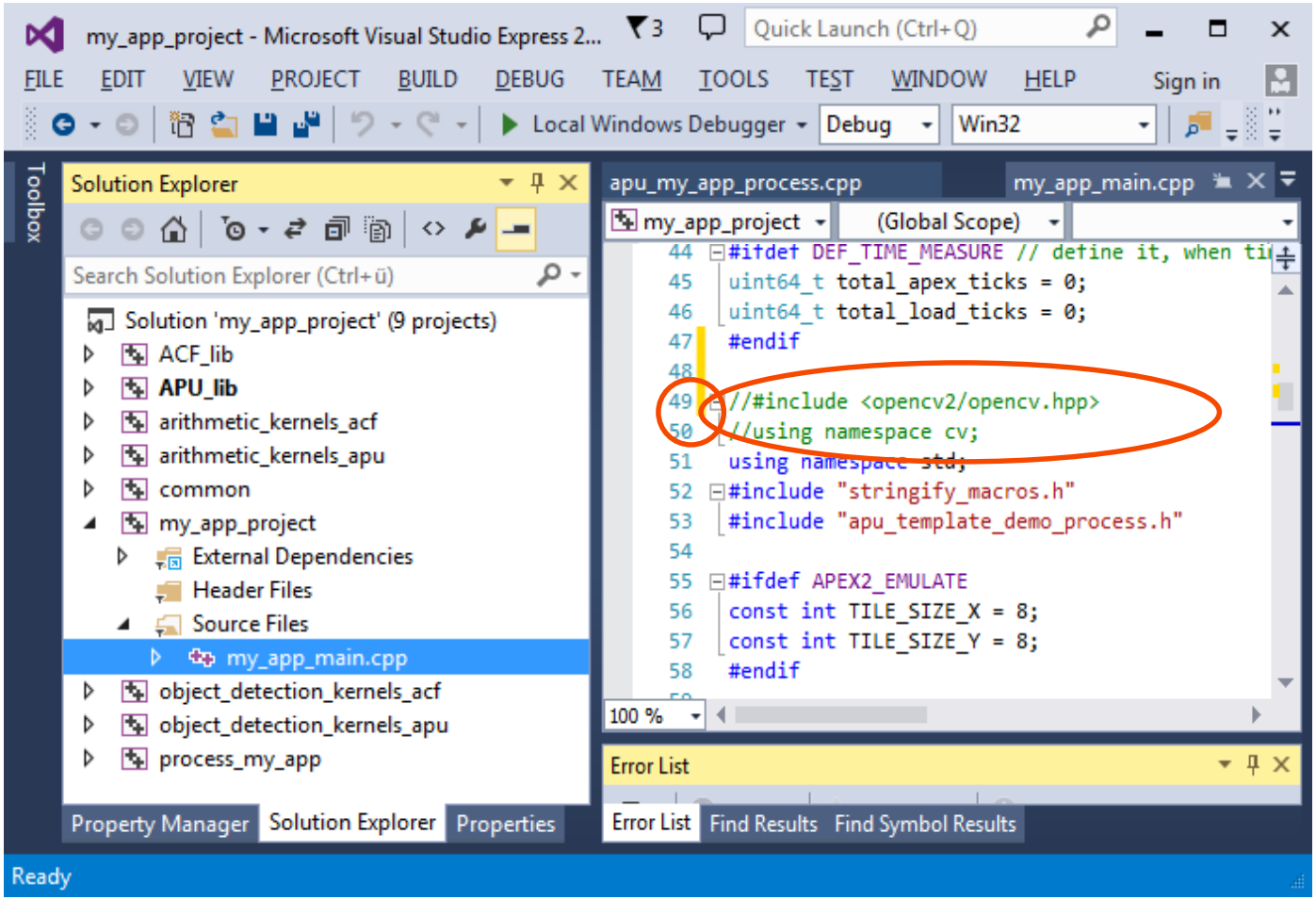
Right mouse click on the project and choose the „Properties“ menu from the dropdown



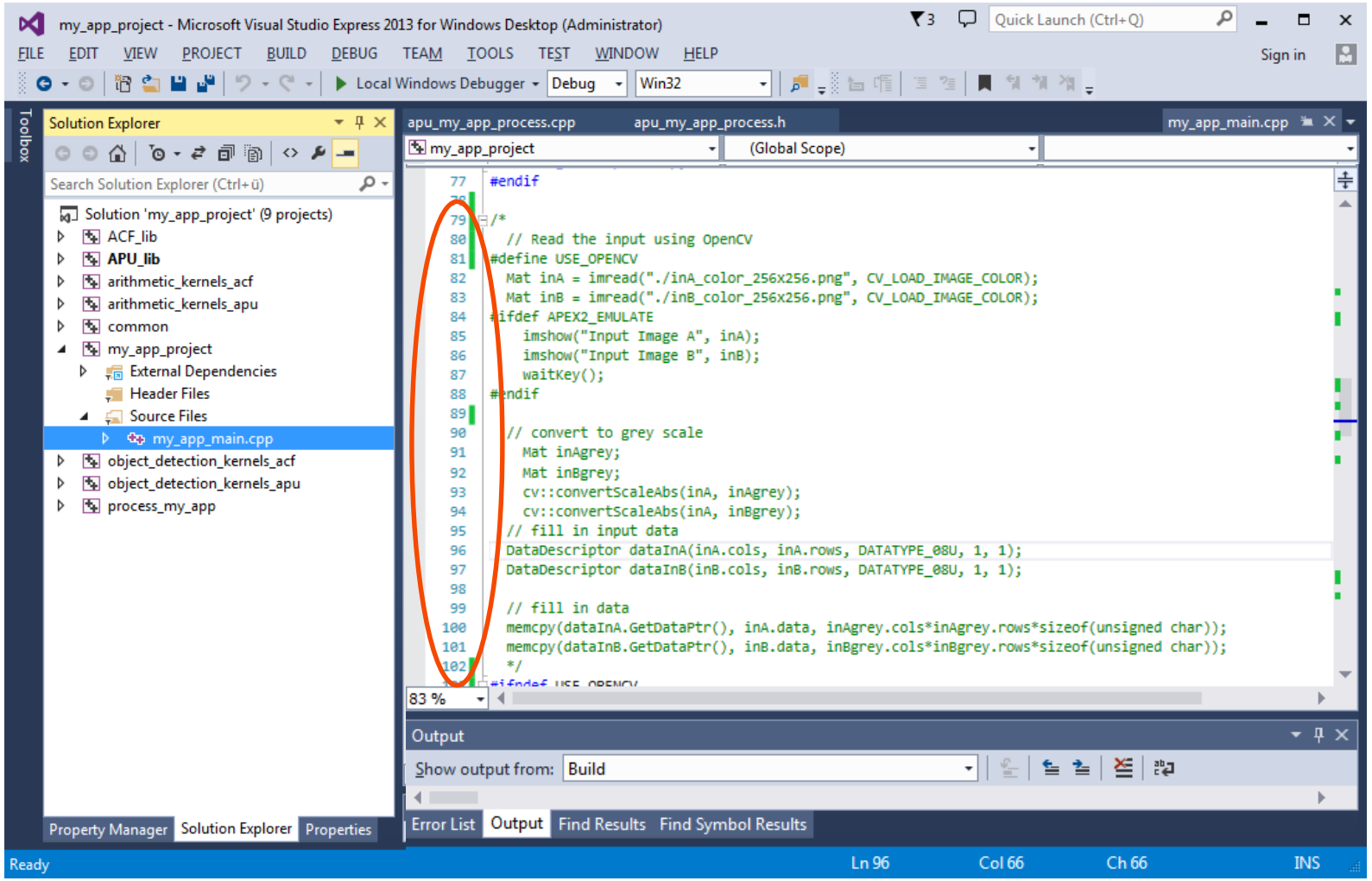
Adding the OpenCV Library

Adding the OpenCV Library

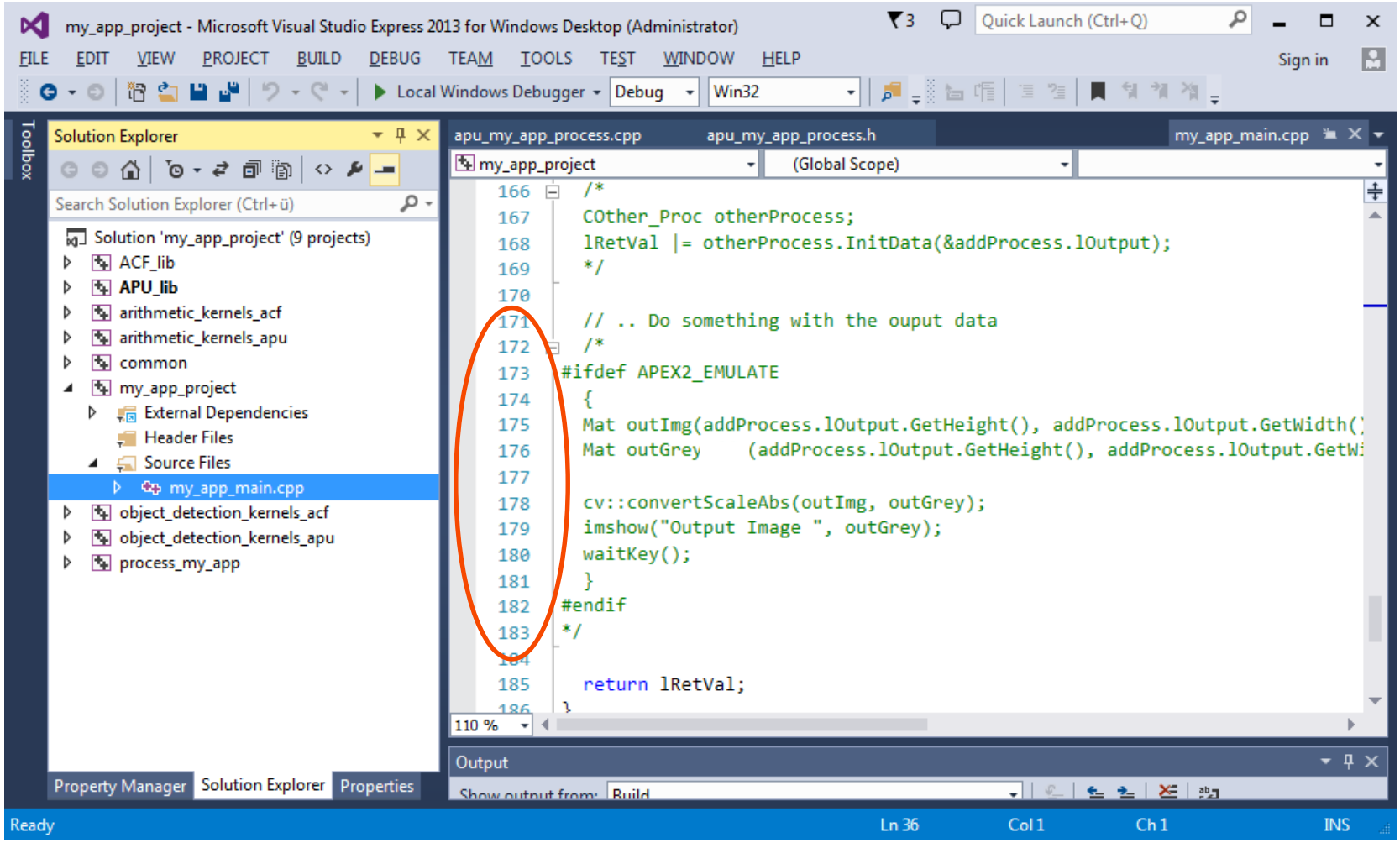
The main function contains already commented-out code for reading/writing images with OpenCV. Uncomment all OpenCV-related code from the main



Uncomment all OpenCV-related code from the main



Uncomment all OpenCV-related code from the main

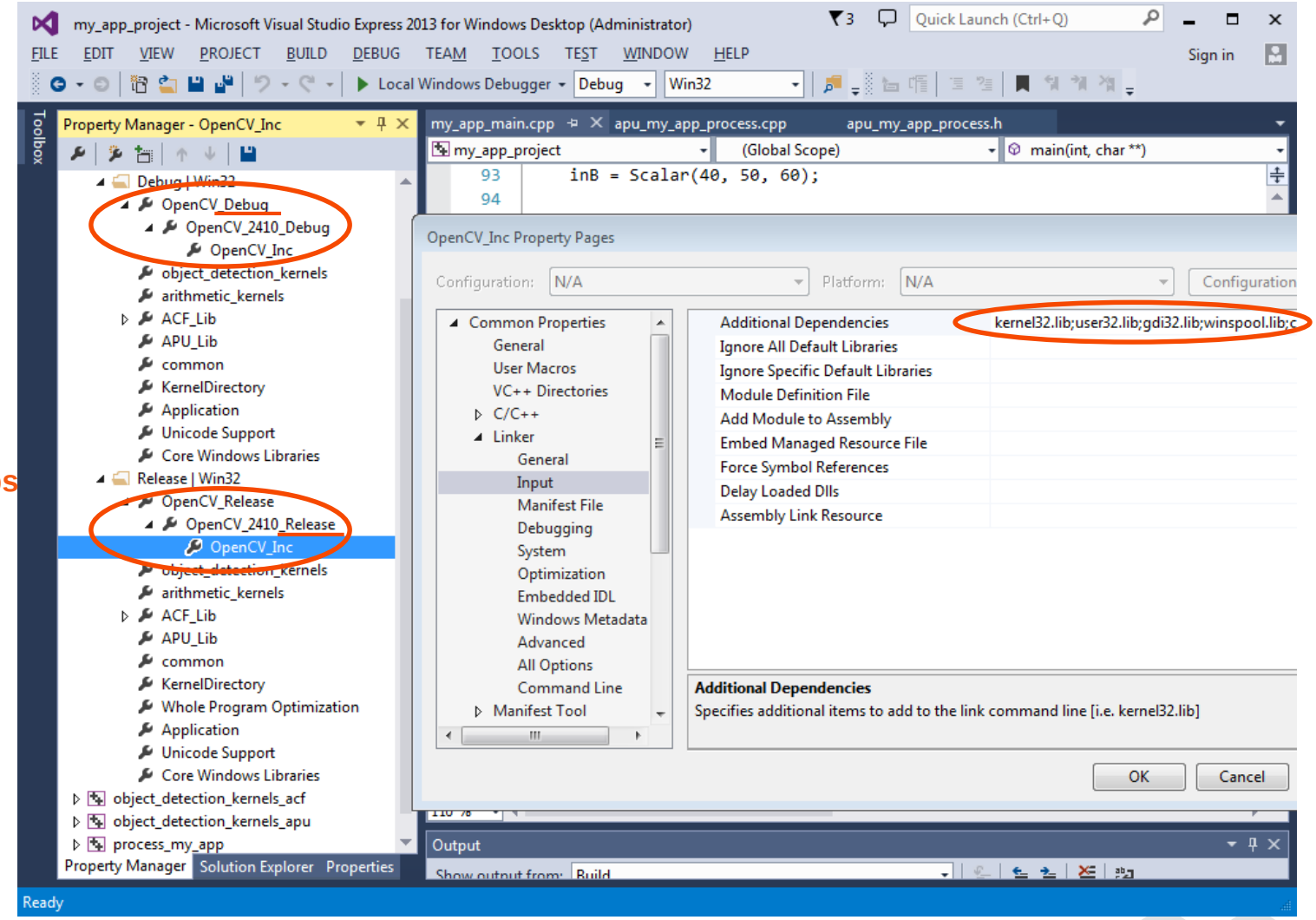


Add the OpenCV Property Sheets to the Main Project

The OpenCV properties contain also the dependencies to the OpenCV libraries.

➔ Therefore a **OpenCV_Debug.props** and a **OpenCV_Release.props**

property sheets exist



Last Steps

- Adapt the system variables to the OpenCV library path
  - setx -m _SDK_ROOT <your_SDK_ROOT>
    (e.g. C:\Freescale\APEX_Emu_i.MX_VSx__0_5_0_e5ecb9\s32v234_sdk)
  - setx -m OPENCV_DIR %_SDK_ROOT%\ocv\win32-x86\vc12
    (or the installation directoriy of your OpenCV library)
  - setx -m TBB_LIB %_SDK_ROOT% \tbb_lib\win32
    (or the installation directoriy of your TBB library)
- Compile
- Set the my_app_project as „StartUp Project“ (by right clicking on it)
- Run it

Cycle Counting

- Base APEX operations‘ cycle counts are defined in tools/emu/apu/src/apu_cycle_database.hpp and can be altered as wished
- The base class – CBase_ProcInit – used to create and initialize processes already contains the mechanisms to count the cycles needed for the execution of an ACF-Graph.
- CBase_ProcInit is residing in the directory: libs/arm/common/include/base_process_init.h
- To include the time-measuring facility into your project, please include the common project from libs/arm and its corresponding property sheet (build\mvc\property_sheets_vs\common.props).
- The main function of the apu_template_demo_project contains sample code lines on how to measure the cycles of each graph and the overall cycle time.
- For cycle counting to be active the DEF_TIME_MEASURE macro definition has to be defined
- ! Cycle counting slows the application significantly down !

Thank you!

www.Freescale.com