freescale semiconductor	AMCU Software	
VISION	Revision <0.0>	Page 1 of 11
	AMCU_SW	

Project Design for Apex Kernel Implementations in Visual Studio Solutions

ABSTRACT:
The document is a guide on how to design projects in order to use APEX Kernel implementations directly in Visual Studio test solutions
KEYWORDS:
APEX Kernels, Visual Studio, OpenCV, Windows Demos
APPROVED:

AUTHOR	SIGN-OFF SIGNATURE #1	SIGN-OFF SIGNATURE #2
Anca Dima		

Revision History

VERSION	DATE	AUTHOR	CHANGE DESCRIPTION
0.0	24-February-14	Anca Dima	First draft
0.1	24-April-15	Anca Dima	Adapted sdk directory to new name

Table of Contents

Project Design for Apex Kernel Implementations in Visual Studio Solutions 1

1 Introduction 5

- **1.1 Audience Description** 5
- **1.2 References** 5
- **1.3 Definitions, Acronyms, and Abbreviations** 5
- **1.4 Document Location** 5

2 Preliminaries 6

- **2.1 Preliminary Steps:** 6

3 The Example Project “apex_add”: 7

- **3.1 Project Organization** 7
- **3.2 General organization of the project folder, concerning a Demo-Application:** 8
- **3.3 Steps to build your own solution <myown_kernel.sln>** 8
- A. The usual way 8
- B. The lazy way 10
- **3.4 Special Settings one should not forget when creating your own <MyOwnKernel> project:** 10

1 Introduction

This is a guide how to design projects in order to use APEX Kernel implementations directly in Visual Studio test solutions. It is meant to ease the daily developers' work while keeping a unitary look and feel of the developed kernels.

1.1 Audience Description

This document is intended to the team of APEX/ACF Kernel developers.

1.2 References

Id	*Title*	*Location*
[1]	*OpenCV library*	http://opencv.org/
[2]	*TBB Library (Intel Thread Building Blocks)*	*https://www.threadingbuildingblocks.org/*

Table 1 References Table

1.3 Definitions, Acronyms, and Abbreviations

Term/Acronym	*Description*
VS	*Visual Studio*
SLN	*Solution file*
SDK	*System Development Kit*

1.4 Document Location

s32v234_sdk/docs/ProjectDesign_for_ApexKernelImplem_in_VS.doc

2 Preliminaries

Due to problems with Property Sheets in the Visual Studio Express 2012, we had to move on to Visual Studio Express 2013.

Therefore OpenCV 2.4.10 was compiled with Visual Studio Express 2013 and libs/dll/pdf files checked into the project. Due to a reported bug in the old Makefiles of our opencv_sources directory, only a new download of the library could be compiled. Thus our proprietary patches are not yet included into the compiled dll's.

2.1 Preliminary Steps:

1. Install Visual Studio Express 2013 (needs the installation of the Internet Explorer 11 which crashes upon opening PDF Files -> problem is not yet solved)
2. Setup following Environment Variables (only once at OpenCV and TBB installation) as in below example:
   CAUTION: Windows is fuzzy about nested environment variables. Therefore it is better to close the command window after each setx command and open a new one
   The A_ before OPENCV_LIB and TBB_LIB are, because alphabetic order matters somehow for windows when expanding...

   > setx -m _SDK_ROOT<your_SDK_ROOT> (e.g. C:\cygwin\home\<username>\s32v234_sdk)
   > setx -m OPENCV_DIR %_SDK_ROOT%\ocv\win32-x86\vc12
   > setx -m A_TBB_LIB %_SDK_ROOT% \tbb_lib\win32 # (or the installation directories of your OpenCV respectively TBB library)
   > setx -m TBB_LIB %A_TBB_LIB%
   > setx -m OPENCV_DIR %A_OPENCV_DIR%

3. In the Environment variables dialog set the path variable to:
   PATH = %A_OPENCV_DIR%\bin;%A_TBB_LIB%\bin\ia32\vc12;%PATH%;
   This is needed for loading the OpenCV and TBB DLL's directly from the installation directory
4. after saving, open a Cmd window and check, that all the pathes are expanded with the command
   > set PATH
   if not, mimic editing the variables which are not yet expanding, by opening the edit dialog for them and closing it with ok (without any editing). Thereafter close the Dialog for Environment Variables with ok and check the PATH variable again with "set PATH"

3 The Example Project “apex_add”:

3.1 Project Organization

Checkout the git/master branch and open the example project %_SDK_ROOT%\demos\apex_add\build-deskwin32\mvc\add_project.sln

This solution consists of following projects:

APU_Lib	contains the emulation functions for the APEX kernel
ACF_Lib	contains the emulation functions for the APEX kernels
common	contains helper classes, such as time measuring and several macro definitions
arithmetic_kernels_acf	contains the files with arithmetic operations’ ACF kernel wrappers which can be used directly for the Target compiler
arithmetic_kernels _apu	contains the files with arithmetic operations’ APEX kernel implementation which can be used directly for the Target compiler
process_add	contains the definition of a process class which instantiates, initializes and executes the add_graph
add_project	contains a main function with a small test environment

The strategy of the .sln (i.e. the Visual Studio solution file) is: for each project, which is not an executable, but acts like a library, a VS-PropertySheet is defined, where the necessary defines and its include and library paths are defined for it. Each project which depends on another project adds for its dependencies the corresponding property sheets and therefore must not add in its own project settings the include paths anymore. This leads to a bigger flexibility of relocating source/header files during the development.

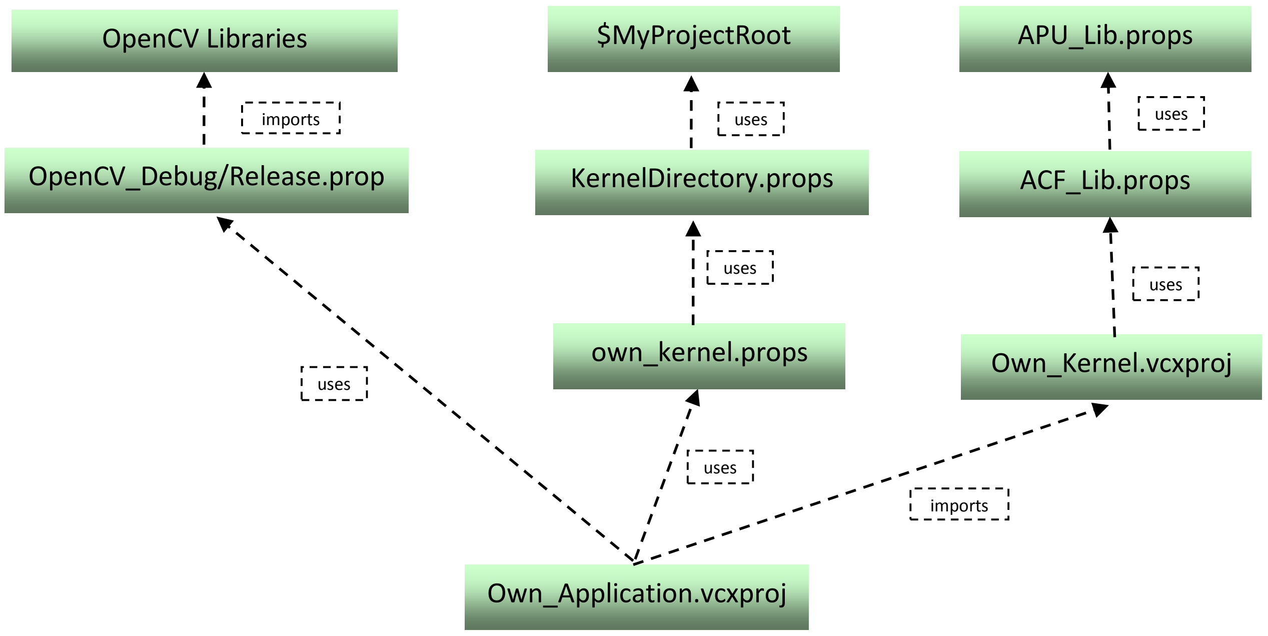
Look at the property sheets each project has, which are located under:
%_SDK_ROOT%\build\mvc\property_sheets_vs

To browse them, use Visual Studio-> View Menu -> Other Windows->Property Manager

There are following property sheets:

- **OpenCVInc.props**: defines the include pathes for the installation of the OpenCV library (relative pathes to the environment variable OPENCV_DIR)
- **OpenCV_Release.props** and **OpenCV_Debug.props** defines the OpenCV libs which have to be linked in and the library pathes.
- **APU_Lib.props**: defines APEX2_EMULATE, ACF_KERNEL_IMPLEMENTATION for the precompiler in order to be able to use apex kernels also in VS-projects. It defines also the include path of the APU library

- **ACF_Lib.props**: defines APEX2_EMULATE, ACF_KERNEL_METADATA, ACF_KERNEL_IMPLEMENTATION for the precompiler (in order to be able to use and it defines the include path of the ACF library
- **common.props**: defines the include path for the common library
- **arithmetic_kernels.props**: defines the include path for the arithmetic kernels used in the apu_add_process.cpp and apu_add_graph.hpp files



3.2 General organization of the project folder, concerning a Demo-Application:

- %_SDK_ROOT%\demos : Contains the more complex applications and test programs which use APEX kernel implementations
- %_SDK_ROOT%\tools\emu\apu: Contains the Apex emulation library
- %_SDK_ROOT%\tools\emu\acf: Contains the ACF emulation library
- %_SDK_ROOT%\kernels\apu: Contains all the kernel implementations for the APEX
- %_SDK_ROOT%\libs\: Contains libraries and drivers needed for target compilation

3.3 Steps to build your own solution <myown_kernel.sln>

A. The usual way

1. Create with Visual Studio a new solution file <MyOwnAPU_SLN> in the directory: %_SDK_ROOT%\demos\<MyOwnAPU_SLN>

General structure of the projects should be inside this directory:

./src - Containing all .c/.cpp/.h/.hpp files

./build-deskwin32 - Containing the Makefile for the nightly build for windows

./build-deskwin32/mvc - Containing the .sln/.vcproj/... etc files

2. If necessary, create a new <MyOwnKernel> project in Visual Studio, located in %_SDK_ROOT%\kernels\apu\<SomeCategory>\<MyOwnKernel>

   General structure of the projects should be inside this directory:

   ./src - Containing all .cpp/.h files

   ./build-deskwin32 - Containing the Makefile for the nightly build for windows

   ./build-deskwin32/mvc - Containing the .sln/.vcproj/... etc files

3. Generate for this new project also a Property Sheet containing the include path of the files here and save it into directory:
   %_SDK_ROOT%\build\mvc\property_sheets_vs
4. Import the APU_Lib project from
   %_SDK_ROOT%\tools\emu\apu\build-deskwin32\mvc
5. If necessary import the ACF project from
   %_SDK_ROOT%\tools\emu\acf\build-deskwin32\mvc
6. For any of the used kernels in the solution, import the corresponding project from
   %_SDK_ROOT%\kernels\apu\<MyOwnKernel>\build-deskwin32\mvc
7. Open the Property Manager (Visual Studio-> View Menu -> Other Windows->Property Manager):
   For the project <MyOwnAPU_SLN> import/add following property sheets:
   a. ACF_Lib.props
   b. <MyOwnKernel>.props
   c. ... any other property sheets from further dependencies ...
   d. If you are using OpenCV, then import
      OpenCV_Release.props into the Release configuration of the Property Sheets
      and OpenCV_Debug.props into the Debug configuration of the Property Sheets
   e. For the project <MyOwnKernel> import/add ACF_Lib.props
8. Open the <MyOwnKernel>.vcxproj file in a text editor and add parallel to an <ImportProperty> following lines:

```
  <ImportGroup Label="Macros" />
  <PropertyGroup Label="UserMacros">
<MyProjectRoot> <the path from your project dir to s32v234_sdk dir>
\s32v234_sdk</MyProjectRoot>
</PropertyGroup>
```

B. The lazy way

CAUTION: THIS PROCEDURE IS DANGEROS, BECAUSE IT CAN DAMAGE PROJECT FILES. It requires a bit of excercise and some attention. It might also take longer to fix html faults than doing it from scratch

1. For each project from which you want to create a similar one:
2. Copy the whole project folder <oldName> and give it a new name, i.e. <NewName>
3. Rename all files <oldName>.* according to <NewName>.*
4. Open a Notepad++ on the new project file and make inside the folder a replace inside all files/subdirectories: <oldName> with <NewName>
5. Open Visual Studio->Tools->Create GUID -> Create a new GUID in Registry format and make in Notepad++ another replace from the old GUID of the project(to be found in the .vcxproj file) to the newly generated GUID-key.
6. Create an other GUID and replace with the GUID for the “Source Files” in .vcxproj.filters and yet two other GUIDs for the “Header Files” and the “Resource Files” (i.e. one for each filter section)
7. Go to the directory of the property sheets (i.e. %_SDK_ROOT%\build\mvc\property_sheets_vs)
8. Copy file <oldName.props> and rename it to <NewName>.props and replace inside the file all <oldName> with <NewName>
9. Copy the convolution_apu solution directory and rename it to <NewName>_apu.sln and do steps 4. and 5 inside that directory and replace the GUIDs of the referenced projects in the solution with the corresponding newly created ones

3.4 Special Settings one should not forget when creating your own <MyOwnKernel> project:

1. In Project <MyOwnKernel>, set the compiling option to /Tp (i.e. treat as C++ code):
   In Visual Studio->Solution Explorer-><MyOwnKernel> right click -> Properties -> C/C++ ->Advanced-> CompileAs: set option to "Compile As C++"
2. In <MyOwnKernel_impl.h> include following code snippet:

```
#ifdef APEX2_EMULATE
#include "apu_lip.hpp" // if using the APU emulation library
using namespace APEX2;
#endif
```

3. In <MyOwnKernel.c> include following code snippet

```
#ifdef APEX2_EMULATE
#include "acf_kernel.hpp" // if using the ACF emulation library
```

```
using namespace APEX2;
#endif

#ifndef APEX2_EMULATE
static  // declare as static only if the code is compiled by the Target compiler
#endif
KERNEL_INFO _kernel_info_apu_gauss_5x5
(..//kernel definition

);
```