DesIRE

Courses TMMA

Tempus

Colleagues

- Sofie Beerens
- Johan Van Bauwel
- Bart Tanghe
- Wim Dams
- Lars Struyf
- Peter Arras
- Dirk Van Merode







Courses

- C for Embedded Systems
- DSP
- Embedded Communication
- Embedded Operating Systems
- Embedded Software
- Multicore Programming
- MCAD
- ECAD

C for embedded systems

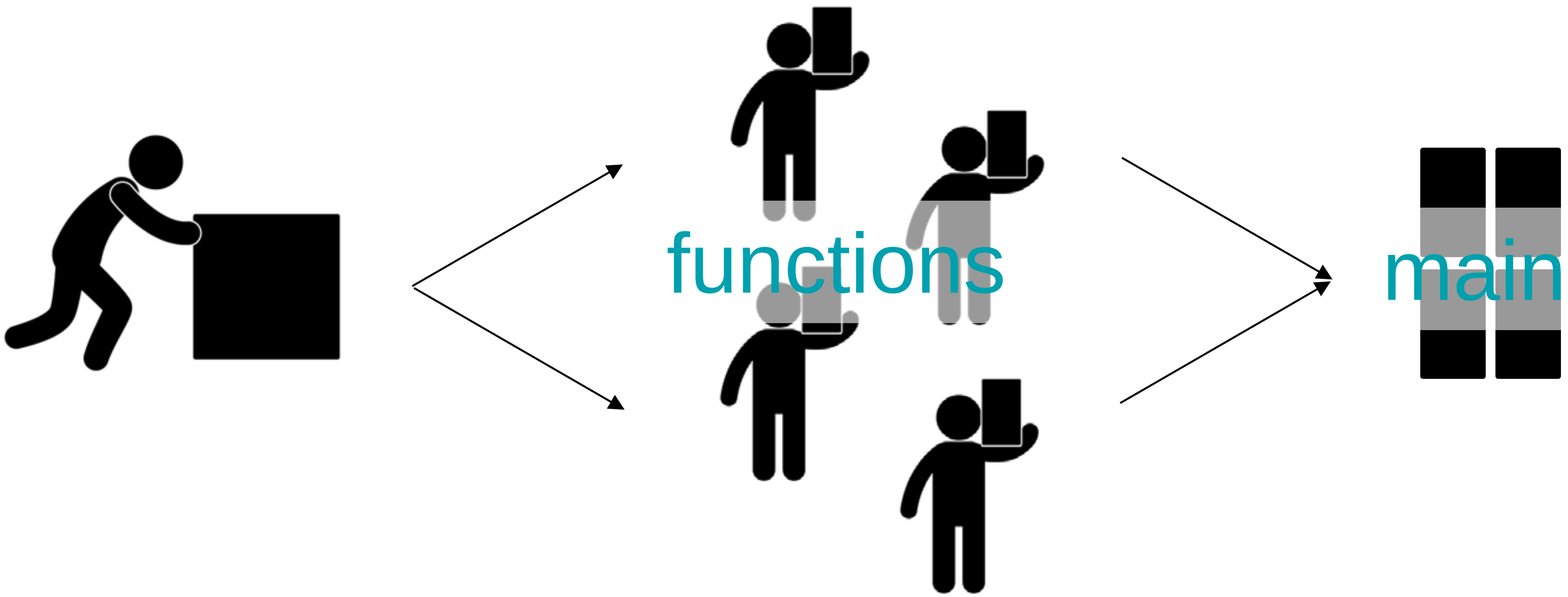
Sofie Beerens

Prerequisites

- Beginners course: first semester of first year
- A basic knowledge of common mathematical methods
- No programming knowledge is required

Objectives

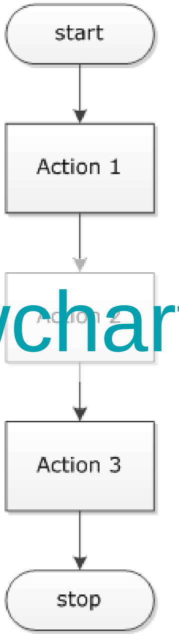
- Introduce basic programming principles:
  - o division of a problem into smaller sub problems



Objectives

- Introduce basic programming principles:
  - o convert (sub) problems into algorithms before coding

Think before coding



flowcharts

Objectives

- Understand and use C syntax:
  - Predict the outcome of programs written in C syntax
  - Create a well structured program in C code containing functions
  - Choice of appropriate datatypes
  - Use file handling in C
  - Perform bit operations

Why C?

- C is a flexible and well-structured language
- designed to:
  - provide low-level access to memory
  - provide language constructs that map efficiently to machine instruction
- available on a very wide range of platforms, from embedded microcontrollers to supercomputers.

Course material

- Textbook: C for Embedded Systems
  - For each chapter/subject:
    - Objectives
    - Theoretical explanation
    - Code examples
    - Pitfalls, do’s and don’ts
    - Exercises

lectures

practicum / individual work

- Visual Studio Express 2013 for Desktop

Content

- Basic description of programming languages

- Dynamic data structures like lists

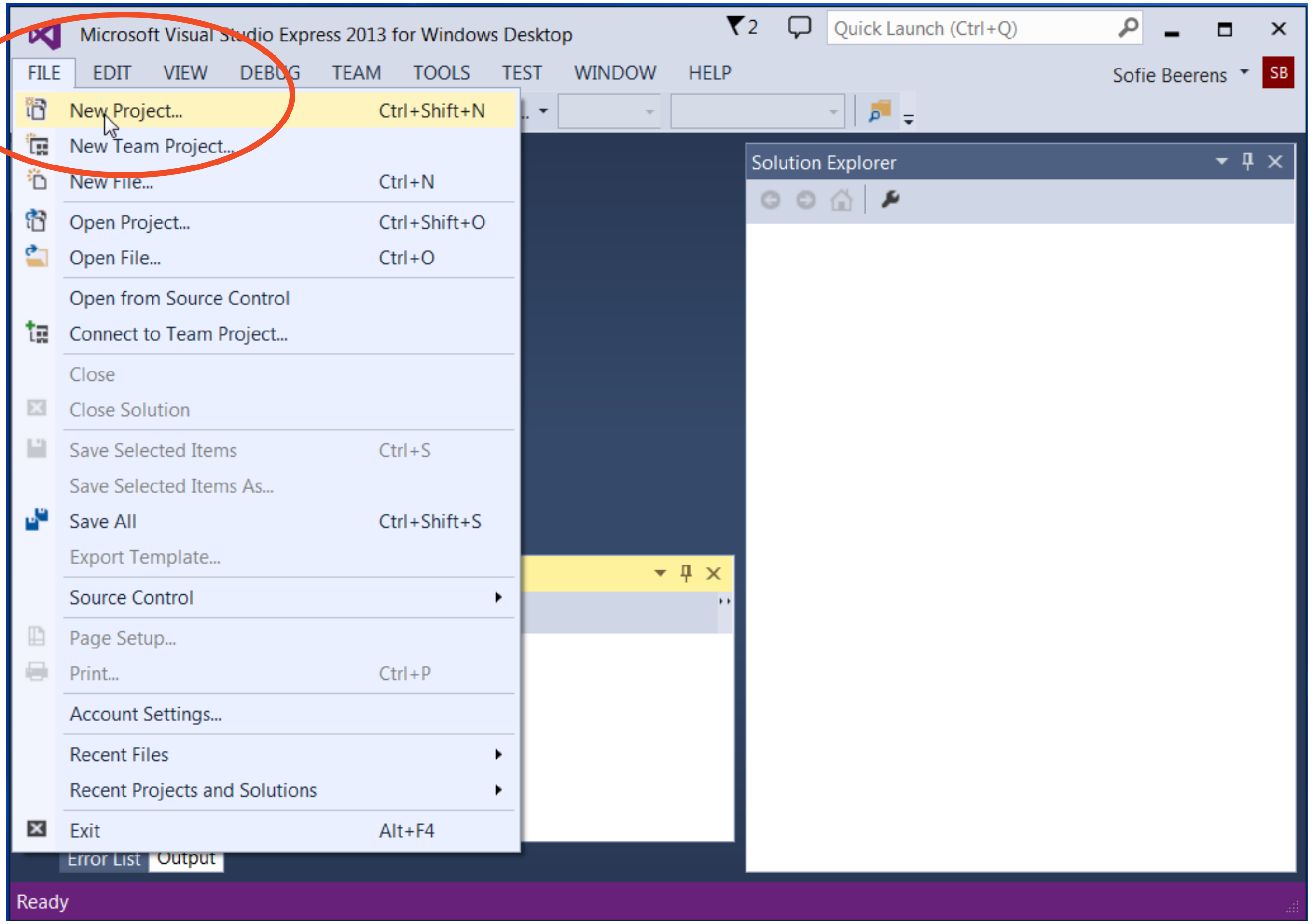
Content

Content

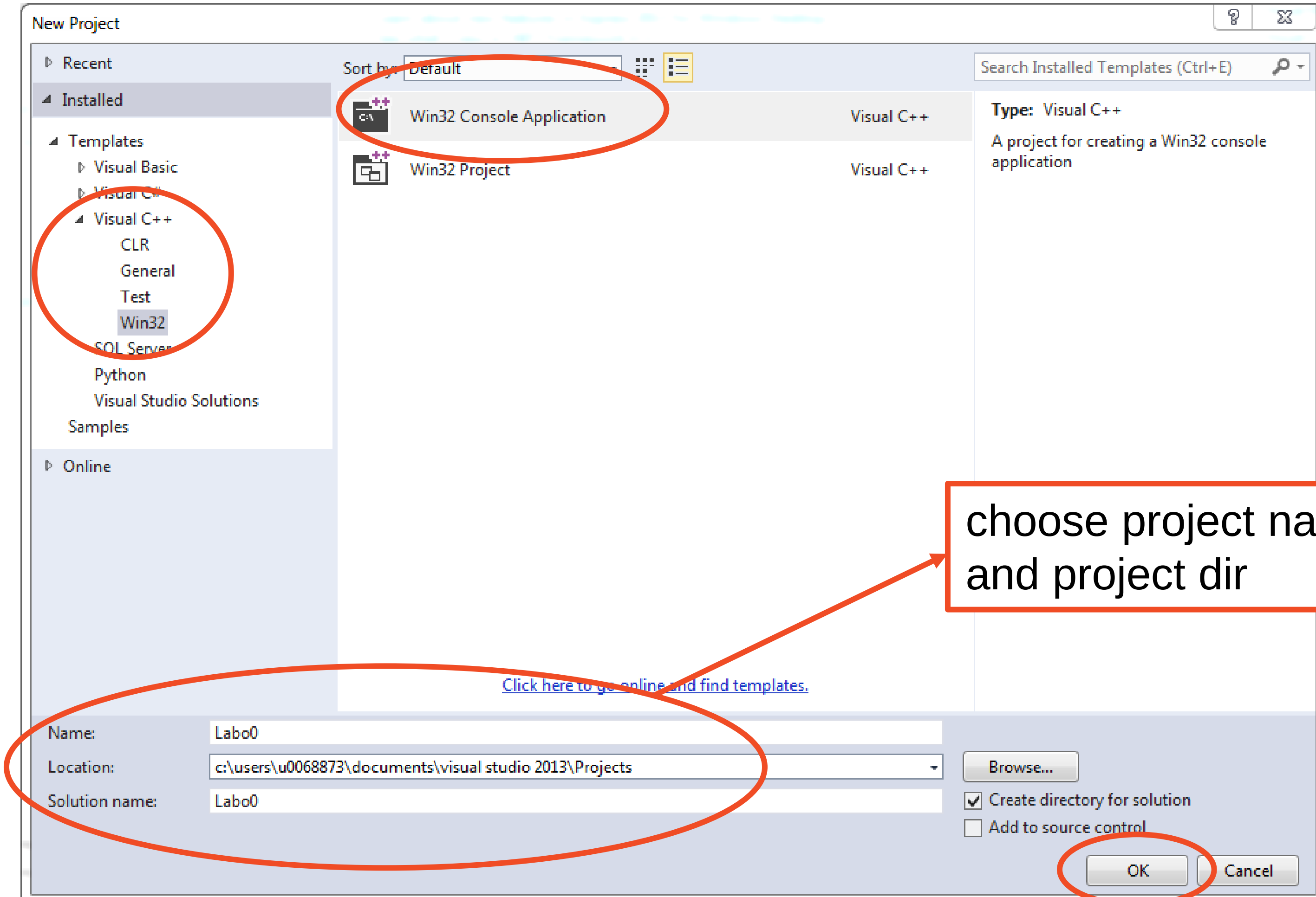
Hello world

- Create a new visual studio project
- Write C code
- Compile the code
- Run the executable
- Verify the output

Create new project



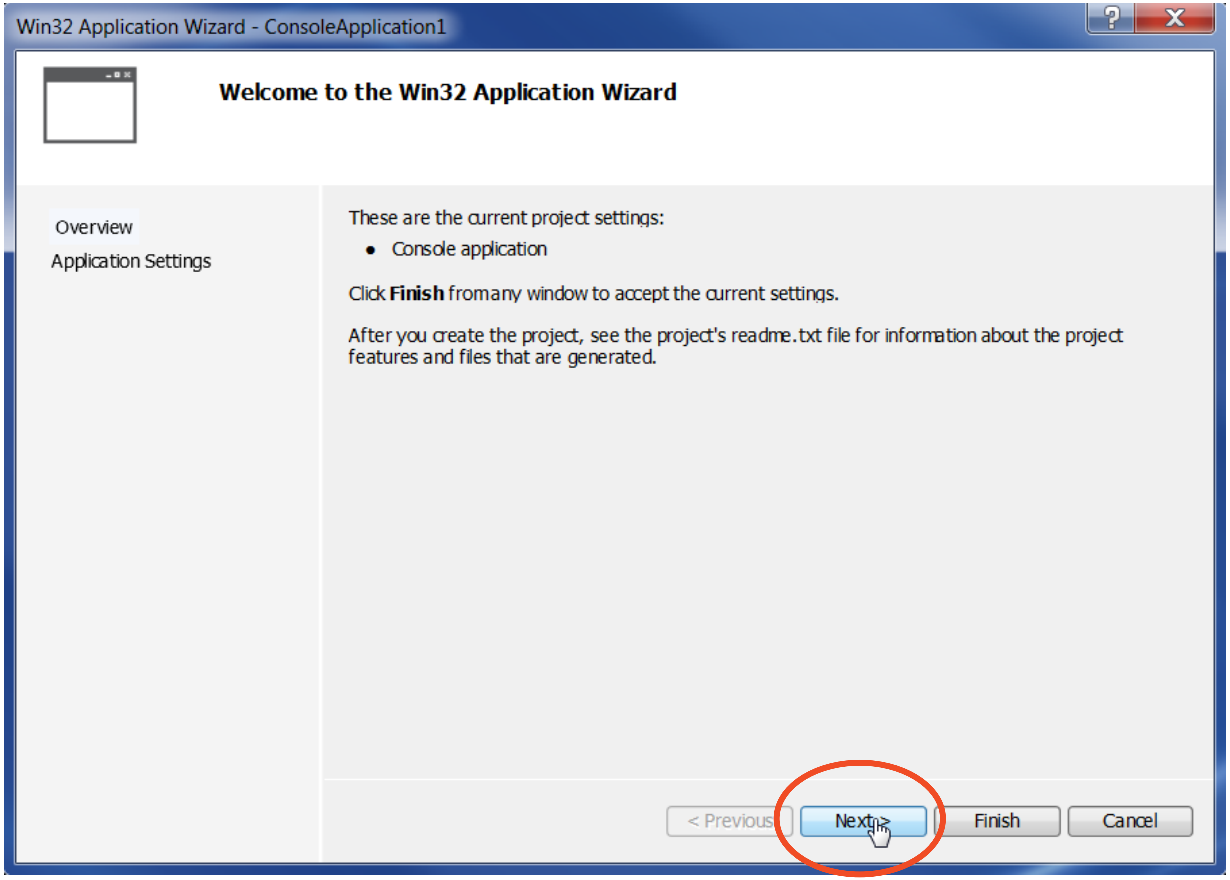
New Project Window



choose project name and project dir

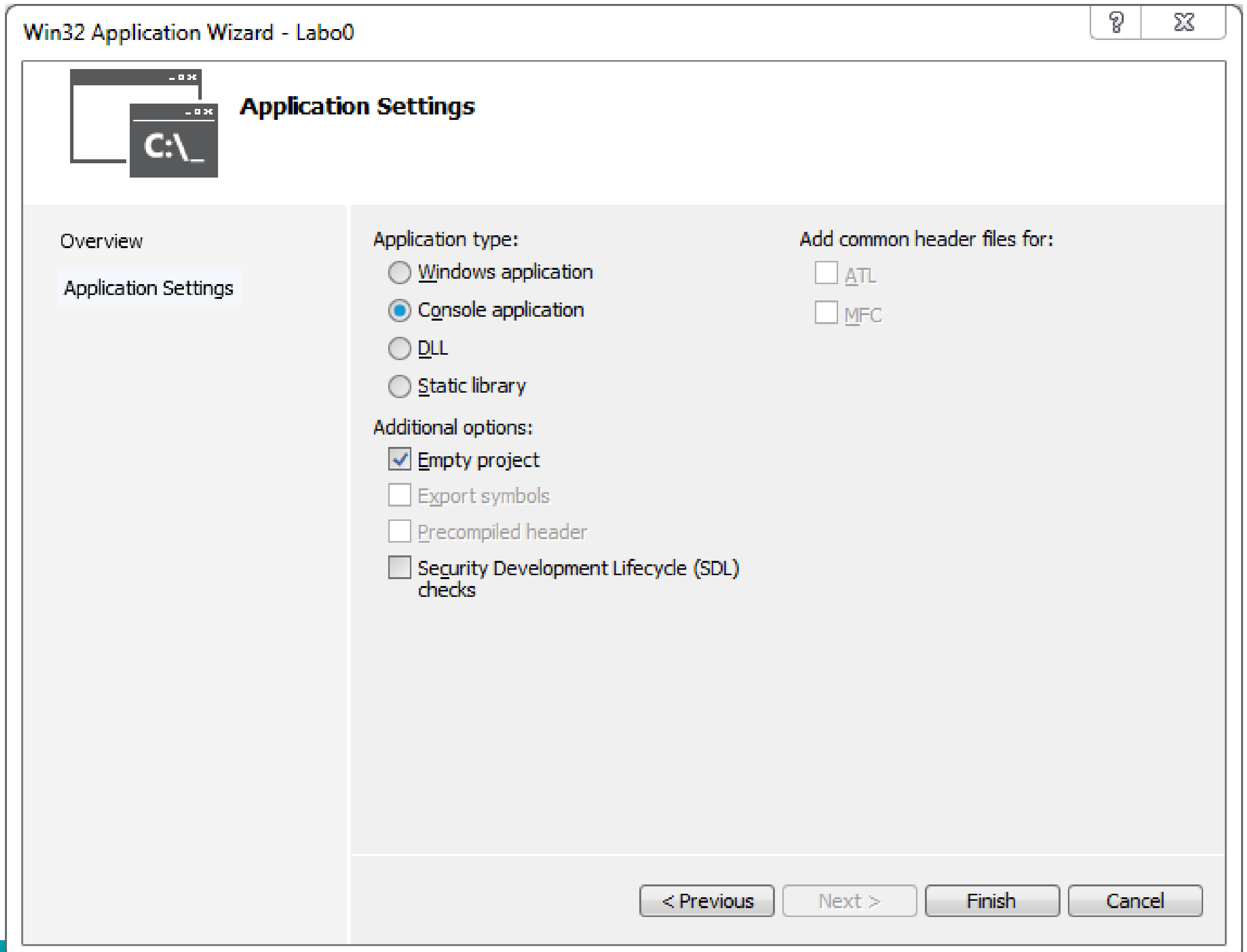
Application Wizard

- Press ‘next’



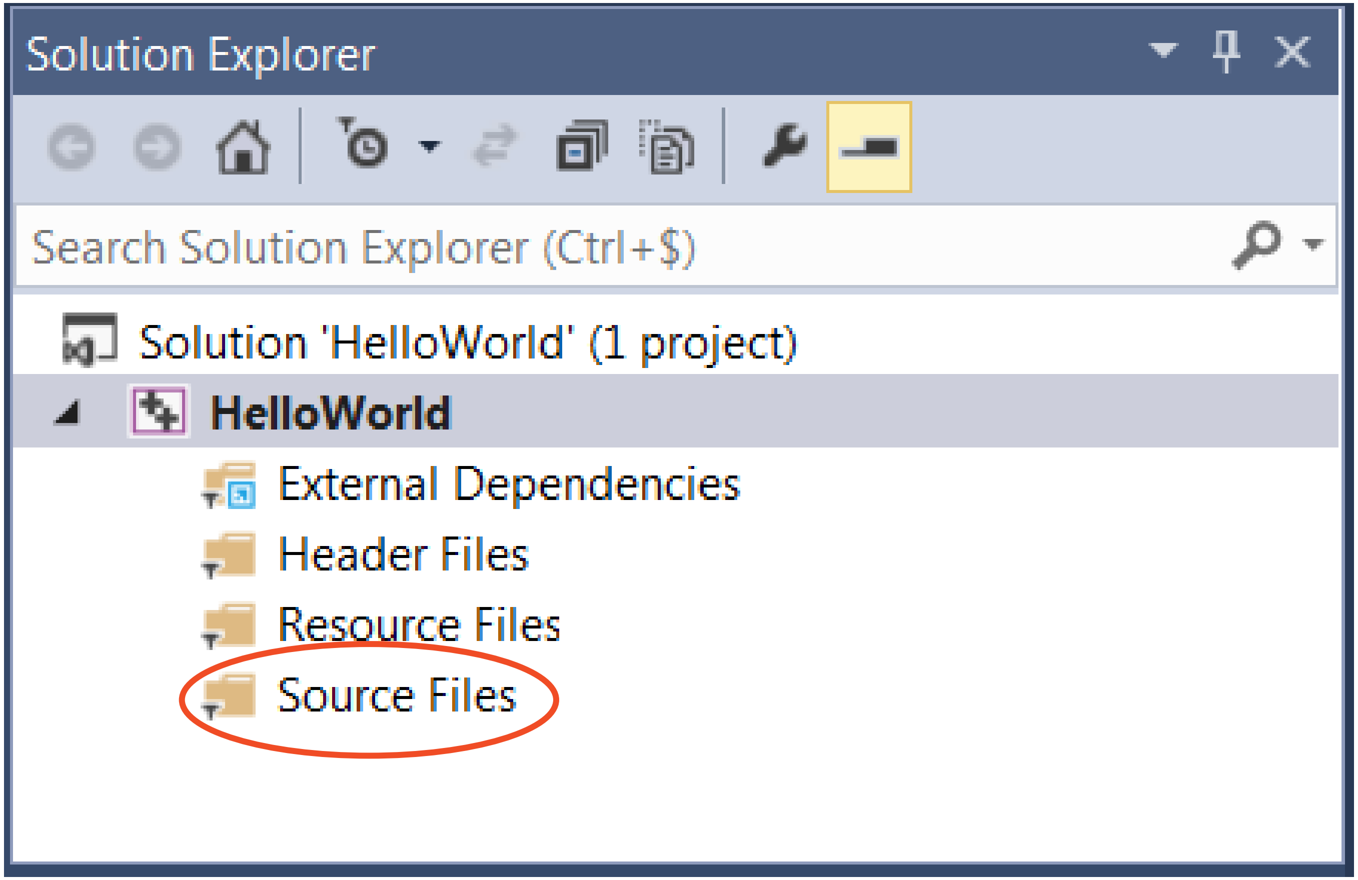
Tempus

Application Wizard



- Select 'console application'
- Deselect 'Precompiled header'
- Deselect Select SDL
- Select 'Empty project'
- Press 'Finish'

Solution explorer



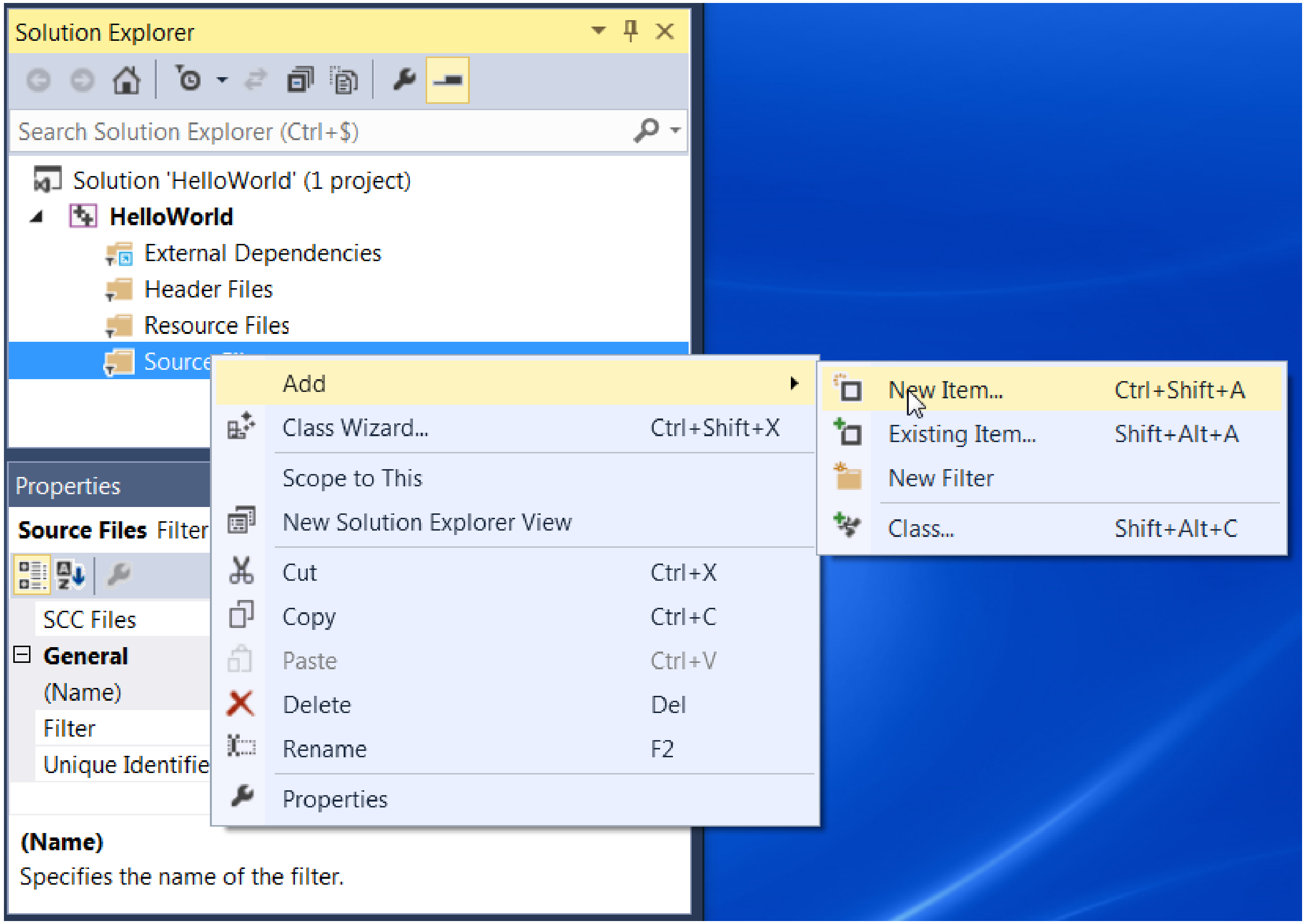
Create new source file

- Right click on “Source Files”, followed by “Add -> New Item”
- Select “Code” and enter a file name

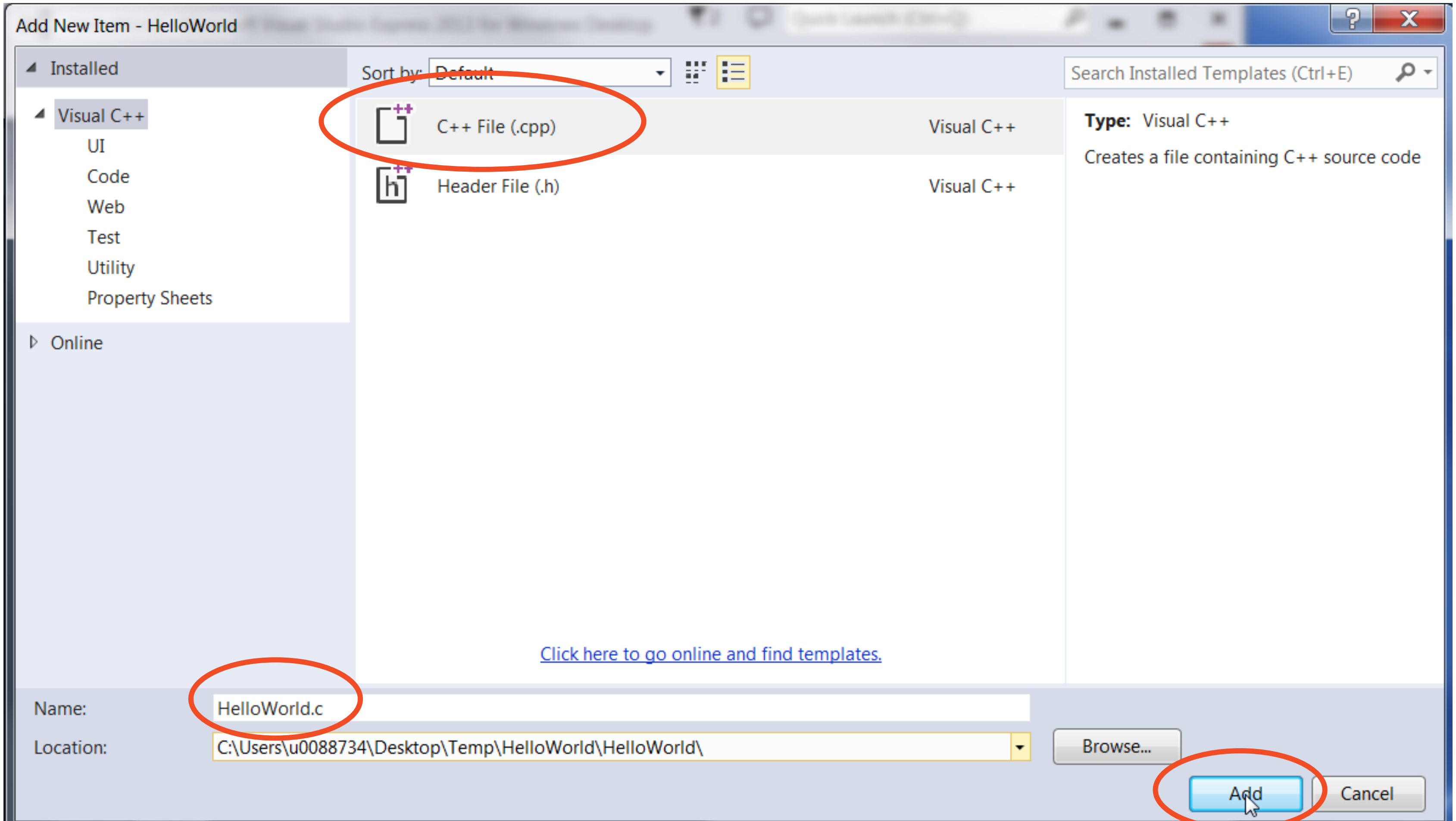
!! Default extension is “.cpp” (for a C++ file).

Make sure you save your file as a “*.c” file

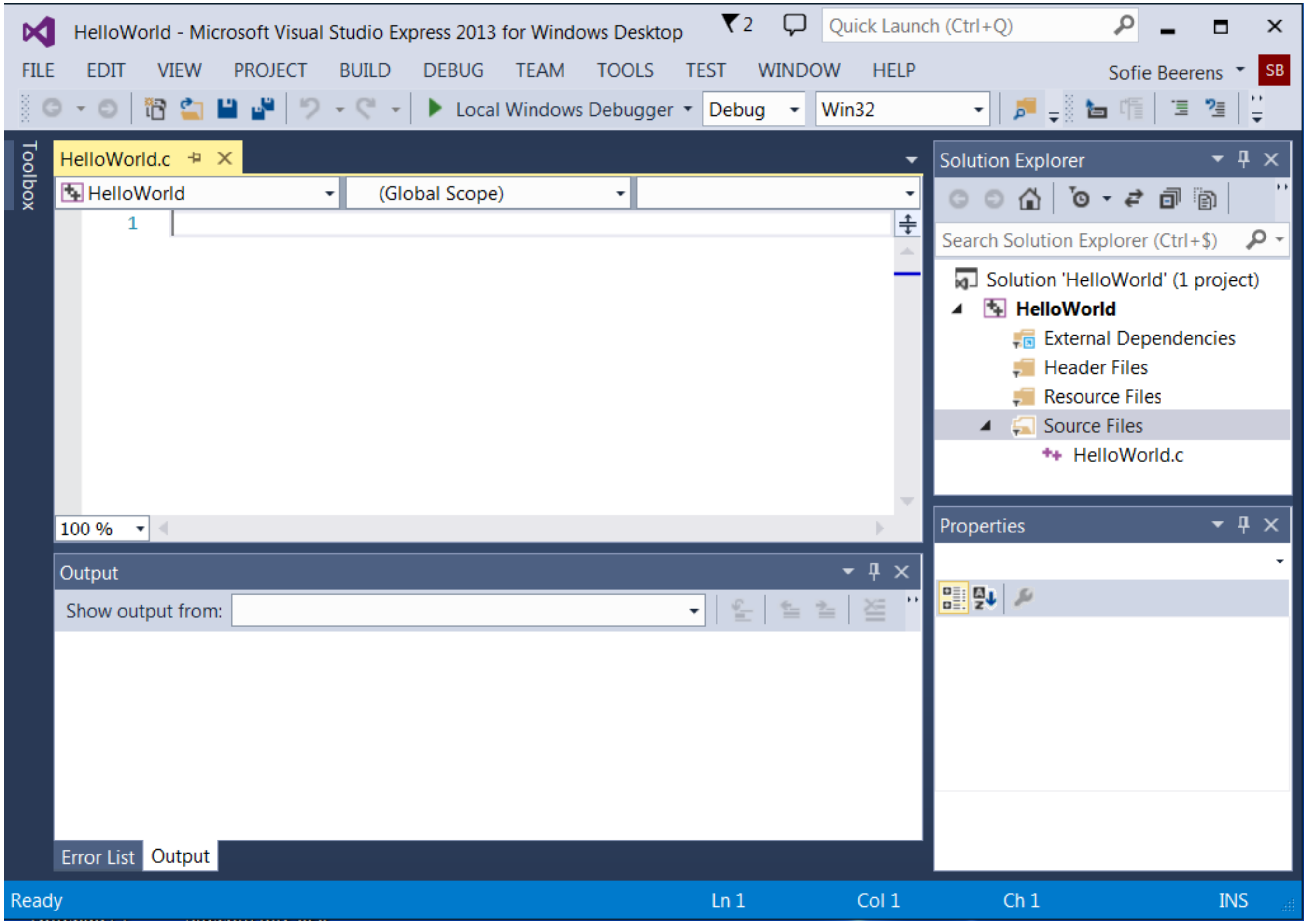
Create new source file



Create new source file



Writing Hello World program



Hello World!

```
/*
        HelloWorld.c
        Our first C program
*/
```

comments

```
#include <stdio.h>
```

preprocessor directive

```
int main(void)
{
        printf(“Hello world\n”);

        return 0;
}
```

no input parameters

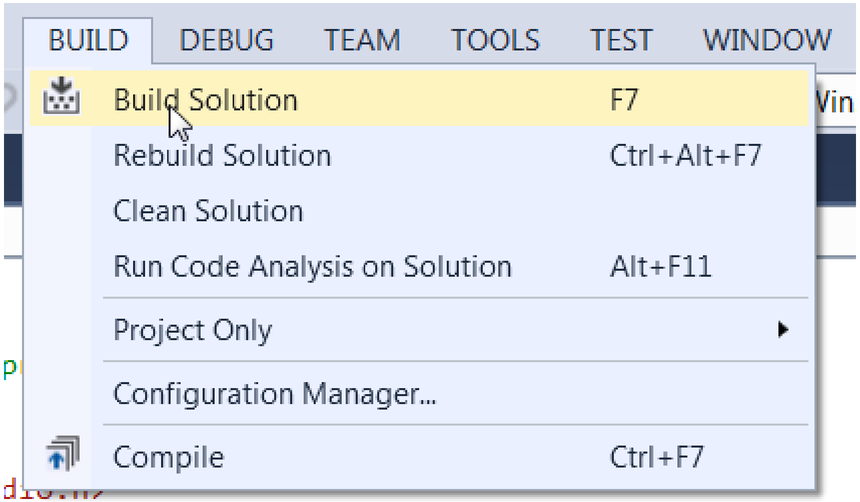
main has an integer return value

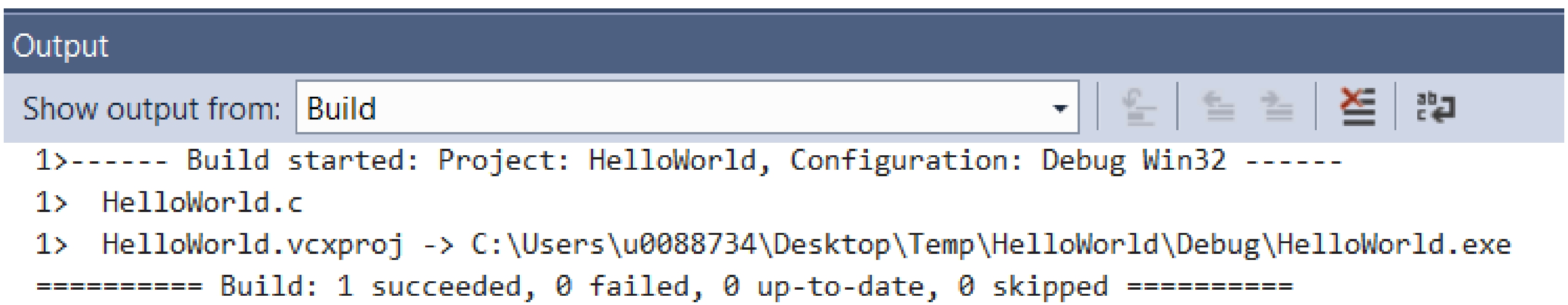
main

statement

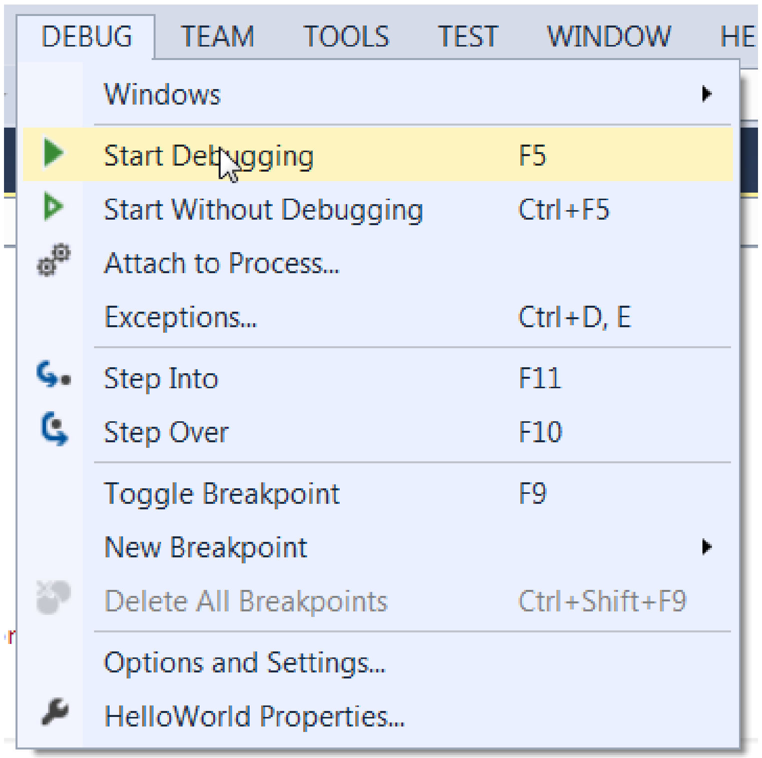
statement

Hello World: create exe file





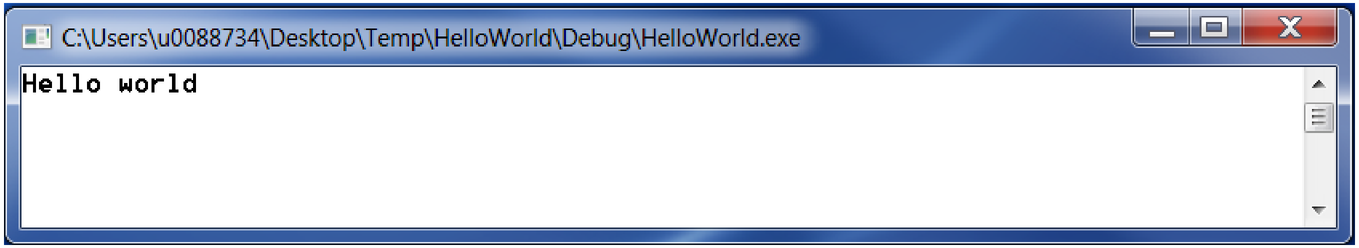
Hello World: run program



- cmd window opens shortly and closes again after execution

=> add breakpoint

Hello World: run program



Other options to keep console

- Use “Start Without Debugging”

- Add a line “getchar();” before “return 0;”

DesIRE

Digital Signal Processing

Johan Van Bauwel

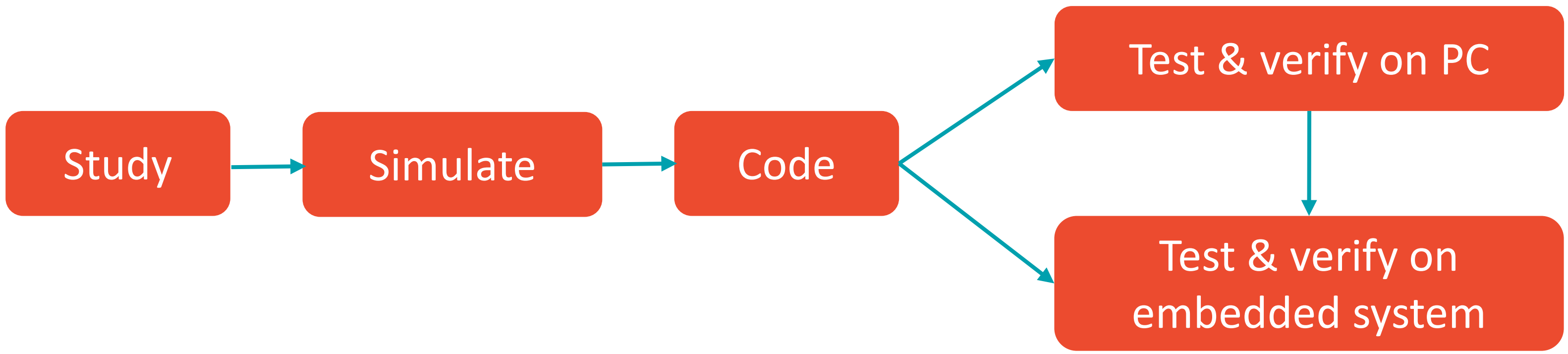
Tempus

Learning objectives

- Students are able to design DSP algorithms using C and/or a higher level language
- Students are able to test the algorithms in a sensible manner
- Students can calculate the impulse response of a LTI-system
- Students can calculate a convolution sum and a DFT
- Students can analyze frequency content of digital signals using the DFT/FFT
- Students are able to create a transfer function and are able to draw a pole-zero plot using the Z-transform
- Students possess knowledge of the various topics treated in this course
- **To make students enthousiast about DSP!**

Teaching methodology

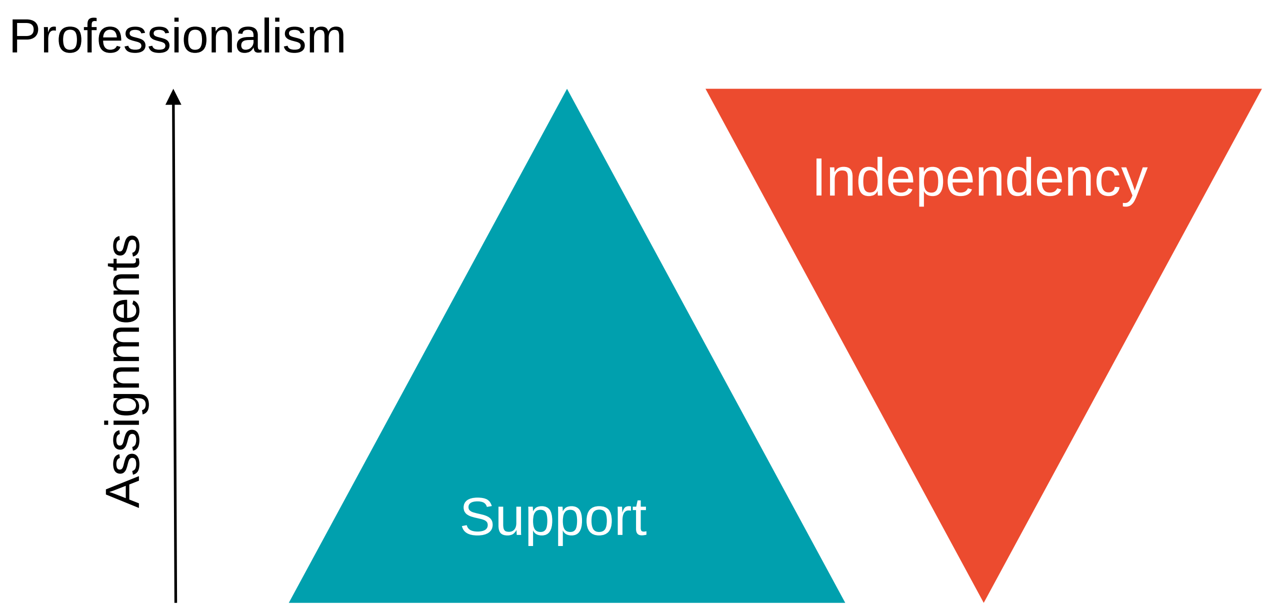
- Prerequisites: C programming, simulation software skills, Linux basics
- Theory: lectures + exercises
- Labs: lab assignments



- Theory: 50%, Labs: 50%

Teaching methodology

- Lab assignments: decreasing support, increasing independency



Teaching methodology

- Lectures: 12 weeks, 2hrs/week
  => 24 contact hours
- Labs: 12 weeks, 3hrs/week
  => 36 contact hours
- Students will have to prepare the lab assignments, study the theory, ...
- Evaluation: theory: closed-book exam
  labs: practical test

Learning tools

- Course text
- Lab assignments, library API & UDOO reference manual
- Various on-line resources
- Supplementary reading material:
  several outstanding books (cfr. next slide)

Tempus

Learning tools

- Oppenheim & Schafer: “Discrete Time Signal Processing“
- Lyons: “Understanding Digital Signal Processing”
- Smith: “The Scientist & Engineer’s Guide to Digital Signal Processing”
- Gonzalez & Woods: “Digital Image Processing”
- Proakis: “Digital Signal Processing”
- Orfanidis: “Introduction to Signal Processing”
- Analog Devices (Walt Kester): “Data Conversion Handbook”
- Ifeachor & Jervis: “Digital Signal Processing: A Practical Approach”
- Tan: “Digital Signal Processing: Fundamentals and Applications”

Course text

- Signals and systems
- Sampling
- Convolution
- DFT
- FFT
- FIR filters
- IIR filters
- Filter Design
- Filter Structures
- Z-transform
- DSP software & hardware
- Multirate DSP

Lab assignments

- Signals and systems, sampling
- Convolution (1D and 2D)
- DFT (incl. windowing and zeropadding)
- FFT
- Digital filters: FIR and IIR
- Filter structures (DF, DF2, cascade)
- The Goertzel algorithm (standard and optimized)
- Basic image processing (color inversion, RGB to grayscale)
- Median image filtering
- Edge detection (using Laplacian, Sobel, Prewitt, ...)
- Histogram equalization
- Audio filters on .WAV files

Demo

Tempus

Embedded OS

Bart Tanghe

Tempus

Scratch on Raspberry Pi

- Presentation & demo

DesIRE

Embedded SW

Wim Dams

Tempus

Embedded Software Course

Bare metal C on ARM Cortex M4

Tempus

Objectives of the Embedded Software Course

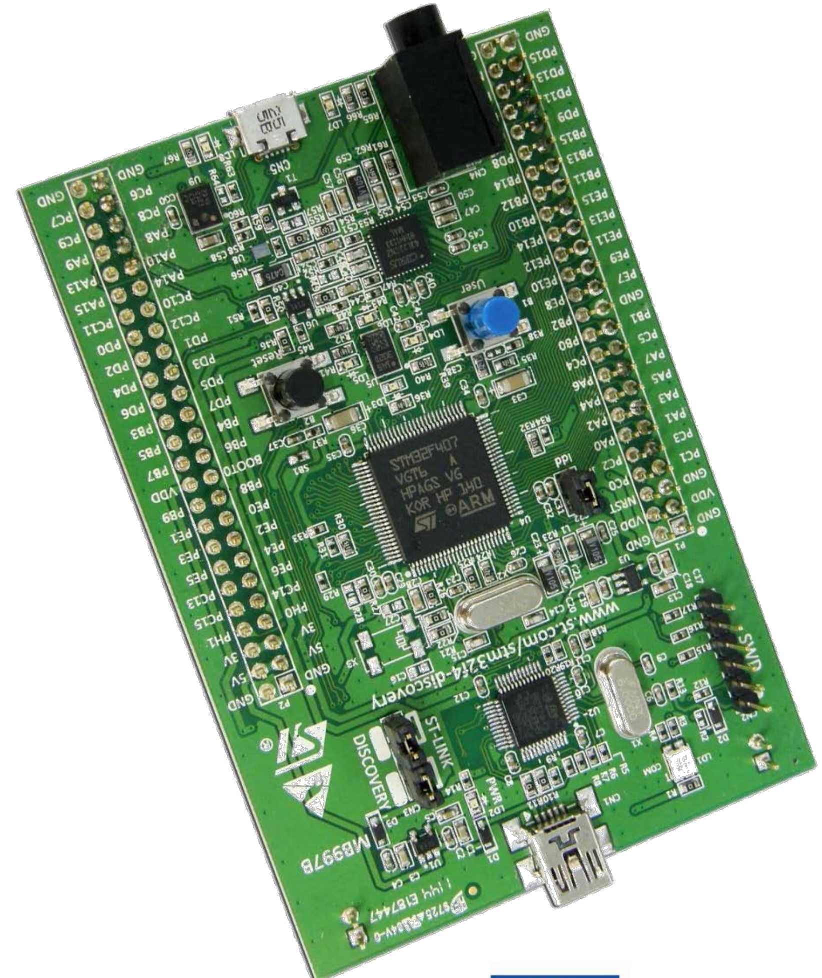
- Develop an understanding of the technologies behind an embedded system
  - Software components: RTOS, HAL Drivers, Libs
  - Hardware Modules: USB, Ethernet,
  - Interaction between software and hardware
  - Build system, compiler settings, performance

ARM Cortex M core

- ARM Cortex-M processors have been licensed to over **175** ARM partners (vendors) and benefits from the widest third-party tools, RTOS and middleware support of any architecture. Which makes it the best choice for embedded applications

The evaluation board: STM32F4DISCOVERY

- Coded in “Bare metal” C
- Embedded ST-LINK/V2 (USB->JTAG)
- LEDs, PushButton
- USB (Host, Device, OTG)
- Motion Sensor
- MEMS audio sensor



The controller: STM32F407VGT6

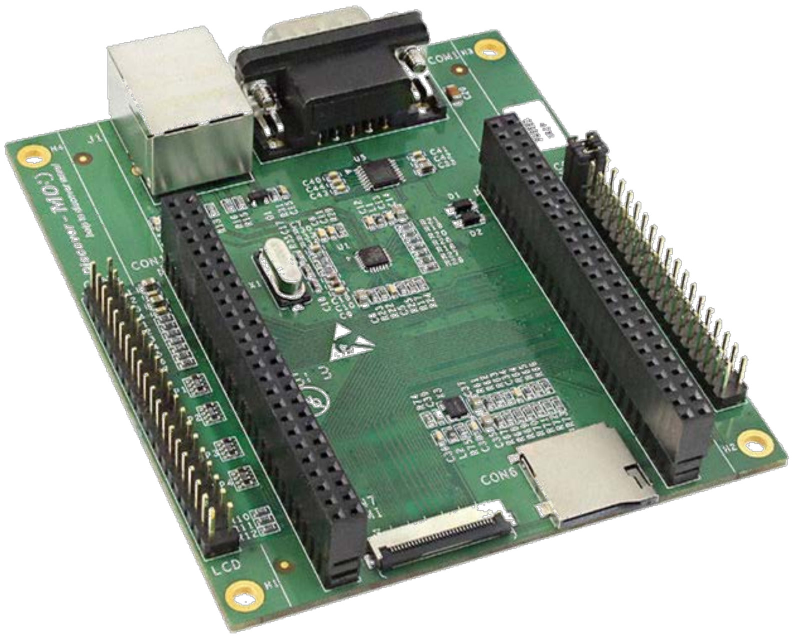
- ARM™ Cortex-M4 core (168MHz/210DMIPS)
- Single Cycle DSP MAC & FPU
- USB, Ethernet MAC, DMA, 6xUSART, 2xCAN, 3xI²C, 3xI²S, 3xSPI, SDIO/MMC
- 1 Mbytes Flash
- 192 Kbytes SRAM

STMicroelectronics

DesIRE

Optional: Base Board (STM32F4DIS-BB)

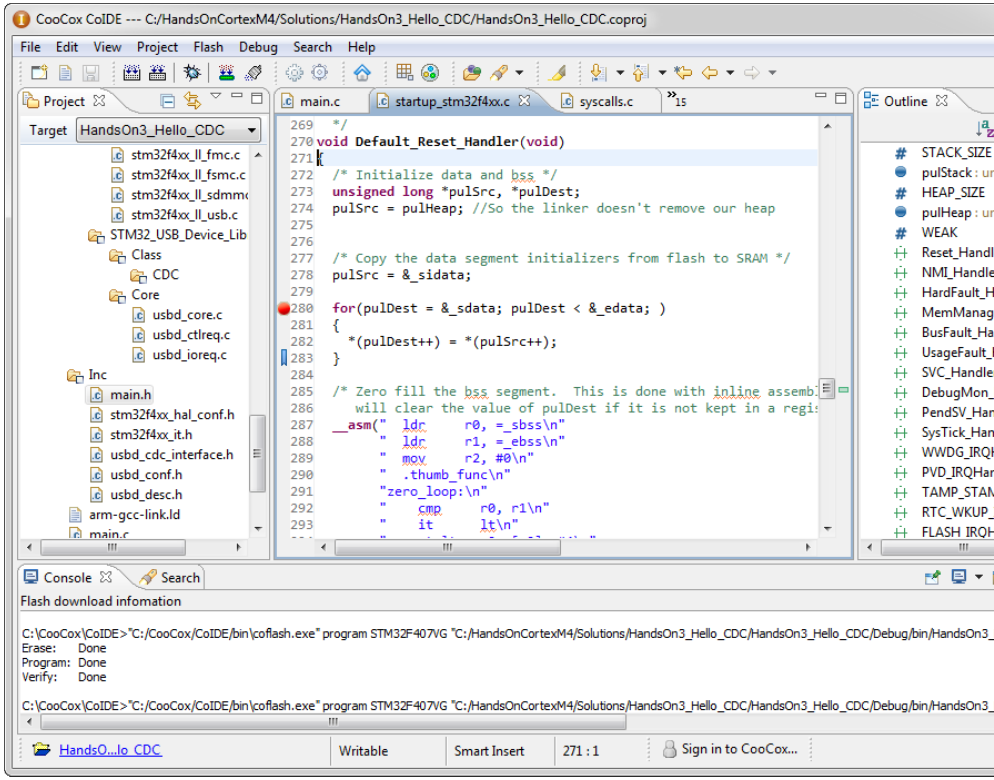
- Interfaces for:
  - o Serial Port (RS232 levels)
  - o Ethernet (Phy)
  - o MicroSD
  - o TFT LCD (sold separately)
  - o Camera (sold separately)



Tempus

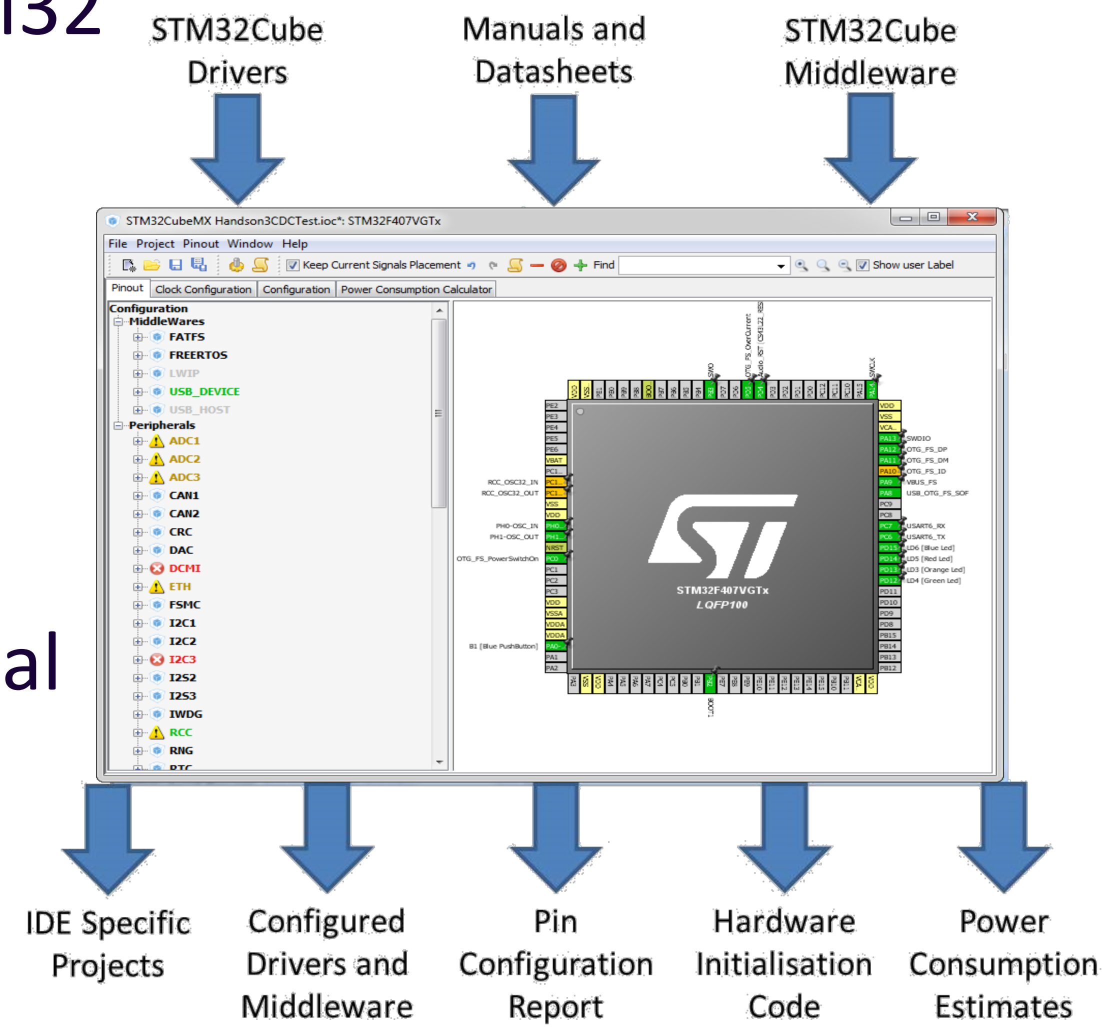
Software environment

- CooCox CoIDE
  - Free
  - Based on Eclipse but better GUI (less options)
  - GCC toolchain
  - Integrated debugger
  - No Simulator (not needed)



STM32CubeMX

- Firmware for STM32 microcontrollers
- Generates boilerplate code based on wizard
- CoIDE is not supported. Manual import is needed



Course Overview

- Labs (Hands On) 2,5h x 12 = 30h
  - Blinky (GPIO)
  - Hello World (UART, Semihosting, USB CDC)
  - Timer (Interrupt)
  - DMA
  - Embedded TCP/IP lwIP
  - RTOS (FreeRTOS)

Course Overview

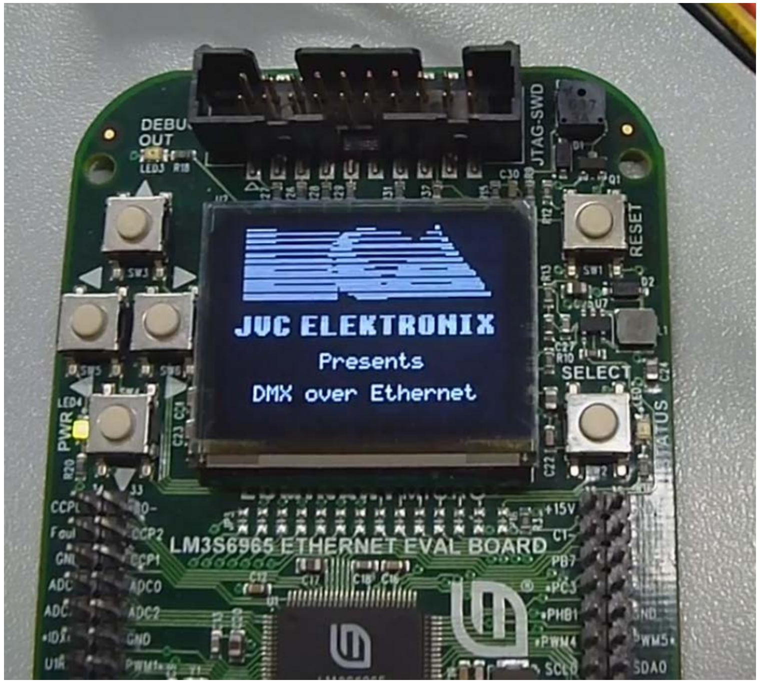
- Project during remaining labs (+homework)
  - Student chooses there own project (functional).
  - Student needs to implement a middleware library (e.g. lwIP or FatFS or FreeRTOS)
  - Teacher guards complexity, cost, …

Tempus

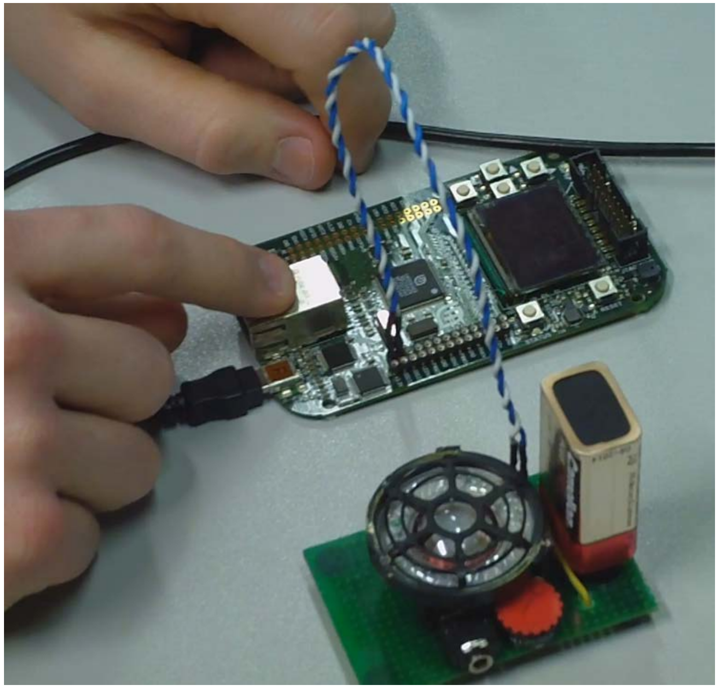
Course Overview

- Results of a Project (on a previous MCU)

Ethernet to DMX



Audio Player

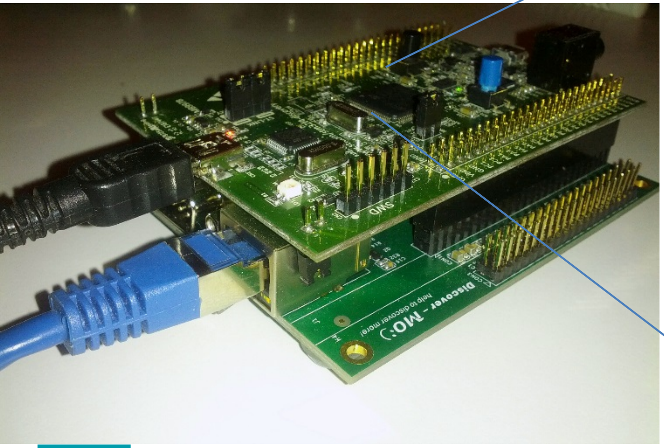


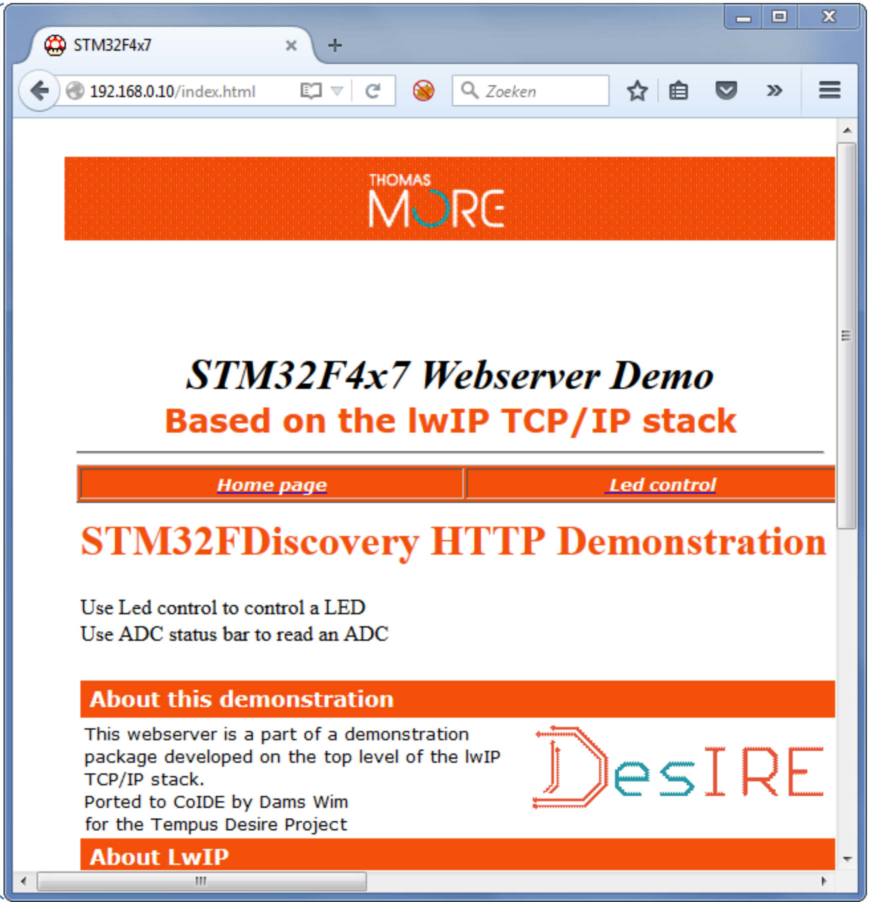
Course Overview

- Lectures 30h
  - Development environment
  - GPIO
  - Semihosting
  - Uart Communication
  - USB Communication
  - Interrupts (Systick, GPIO)
  - lwIP
  - DMA (Uart)
  - RTOS
  - Start-up/Boot code Linker scripts
  - Coding Standards (CERT C, MISRA C)
  - Coding Style
  - ARM Cortex-M Core

Demonstration

- Http webserver based on Hands On 5 (lwIP)
  - LED3
  - ADC





Multicore Programming

Lars Struyf

Tempus

Goals Of The Course

- Give students an understanding about multicore programming
- Give students some background on multicore hardware
- Learn students how to program multicore hardware with OpenCL

Goals Of The Course

- Theory: 12 weeks x 1h
- Labs: 8 weeks x 3h

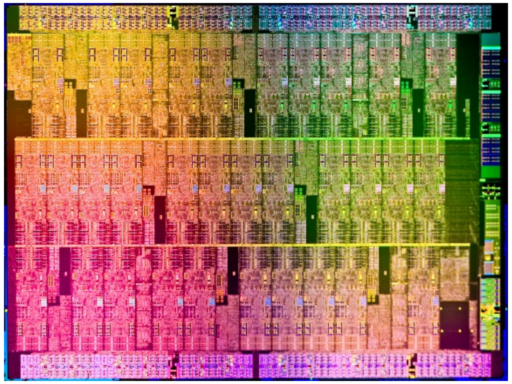
- Material
  - PowerPoint presentations
  - Lab assignments
- Complementary reading material
  - OpenCL 1.2 specification
  - Heterogeneous Computing with OpenCL v2
  - Internet

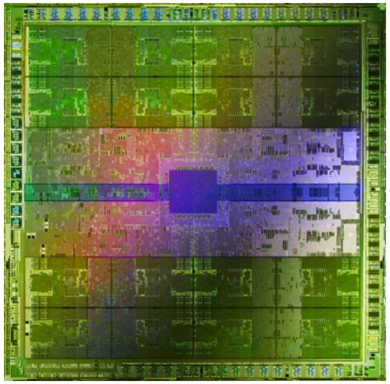
Course Overview

- Introduction
  - Parallel Computing
  - OpenCL
- GPU Architectures
- OpenCL
  - Buffers
  - Images
  - Memory
  - Threading HW
  - Optimizations
  - Nbody Optimizations
  - Extensions
  - Timing
  - Debugging
  - Multidevice

Parallel Computing

- Parallel computing
- Multicore hardware
- Determining parallelism in software





Tempus

OpenCL

- What is OpenCL?
- OpenCL platforms
- OpenCL devices
  - Contexts
  - Queues
- Data transfers
- OpenCL programs and kernels
- Threads
- Memory model
- Address space

Useful Information

- OpenCL Specification
- Books
  - Heterogeneous Computing with OpenCL
    - Second Edition revised for OpenCL 1.2
  - OpenCL Programming Guide
  - OpenCL In Action

Content

- Goals
- Course Overview
- **Labs**
  - Hardware
  - Software
  - Lab Setup
  - Lab Exercises
  - Lab Projects

Hardware

- NVidia
  - GPU
    - GeForce 8400GS and up
    - Quadro NVS 295 and up
    - Tesla
  - Tegra2 and up
- AMD/ATI
  - GPU
    - Radeon HD4000 series and up
    - FireStream 9250 and up
    - FirePro V3750 and up
  - CPU
    - AMD x86 with SSE 2 and up
- Intel
  - CPU
    - Intel x86 with SSE 4.1 and up

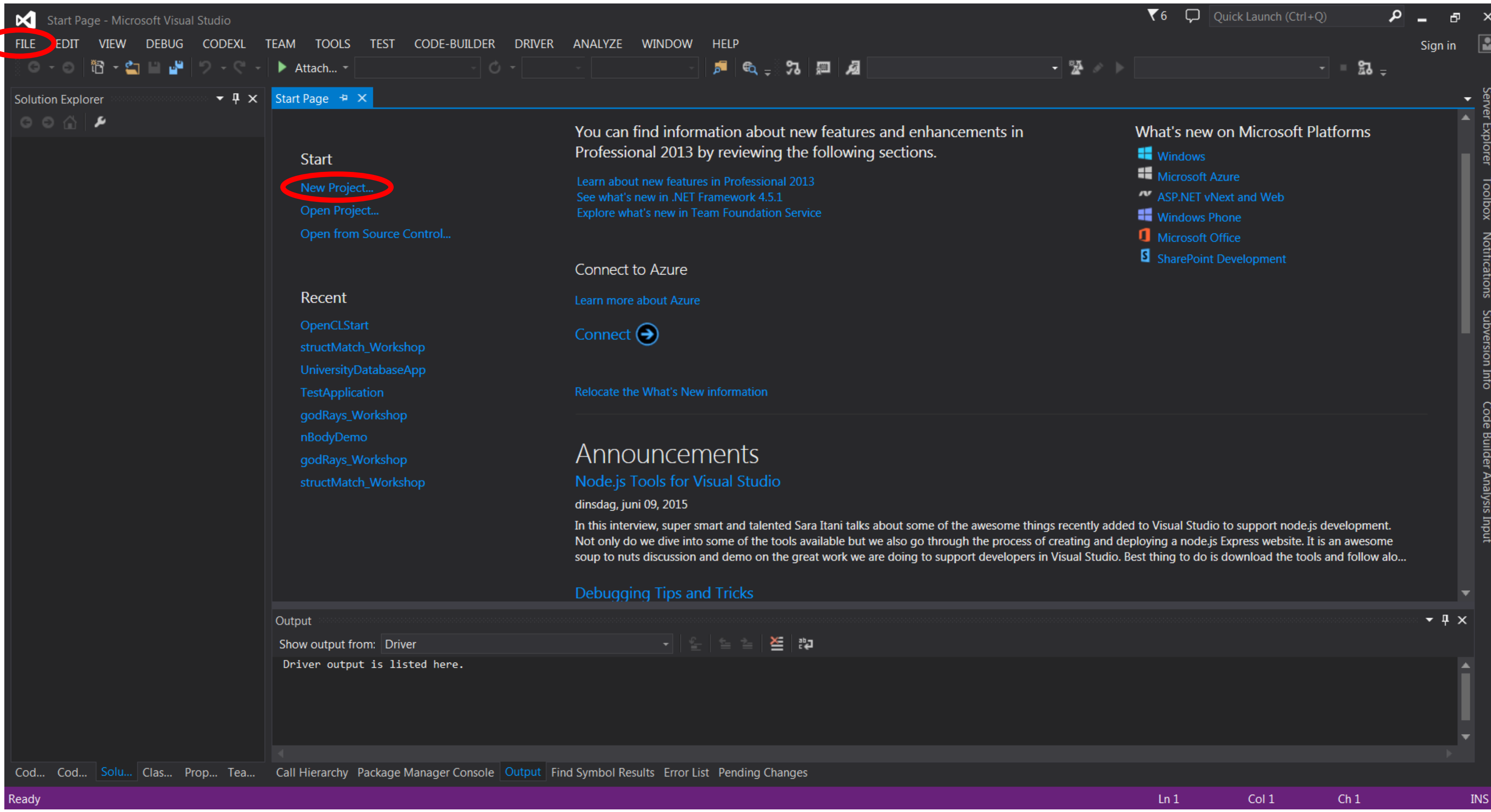
Hardware

- ARM
  - GPU
    - ARM Mali 600 and up
  - CPU
    - Cortex-A7, A9, A15, A17, A53, A57
- IBM
  - Processor
    - Cell/B.E.
- Altera
  - FPGA
    - Cyclone V
    - Stratix V
    - Arria V

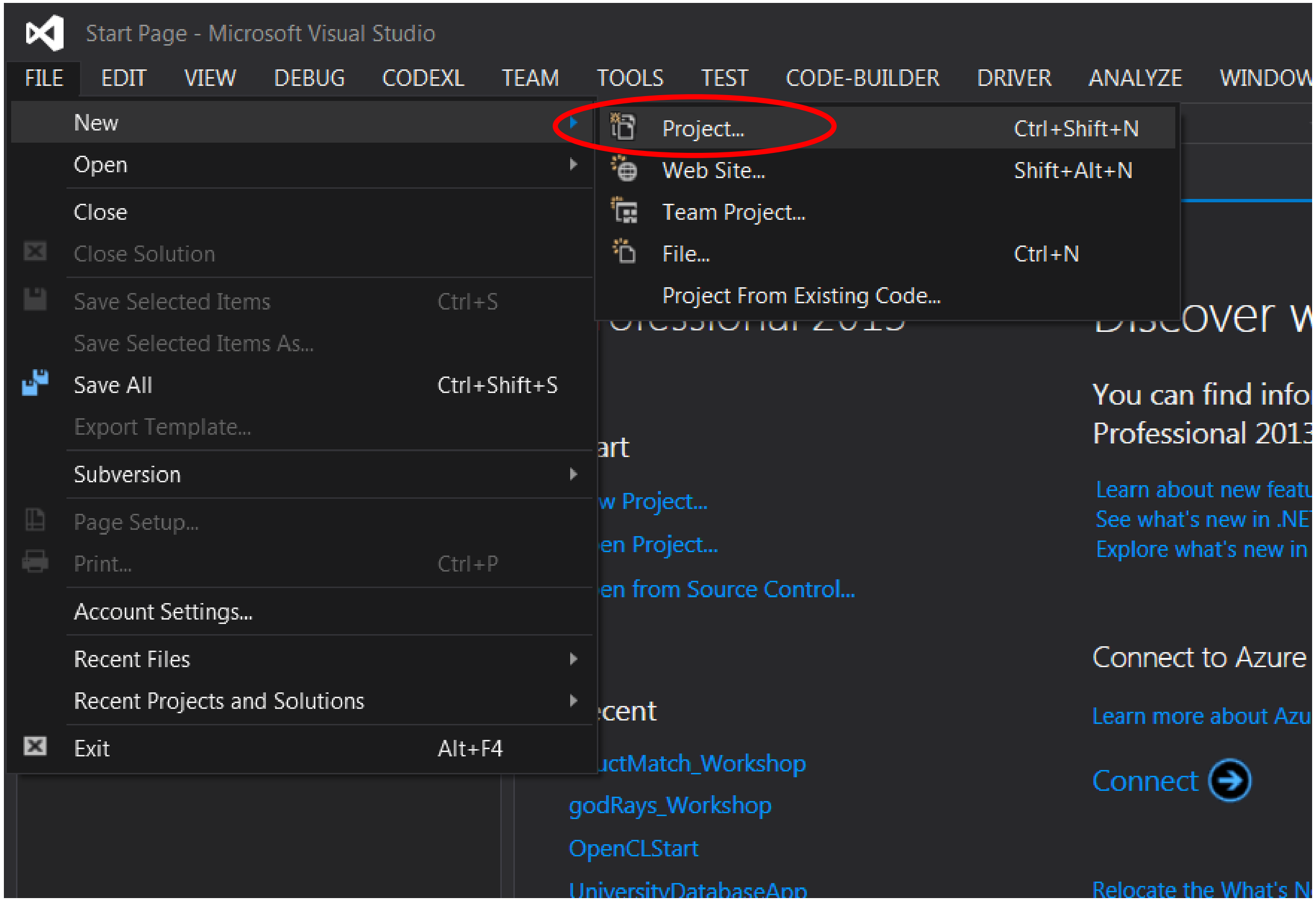
Software

- AMD
  - AMD Accelerated Parallel Programming (APP) SDK
  - CodeXL
  - http://developer.amd.com/tools-and-sdks/opencl-zone/
- NVidia
  - CUDA
  - https://developer.nvidia.com/get-started-parallel-computing
- Intel
  - Integrated Native Developer Experience
  - https://software.intel.com/en-us/intel-opencl

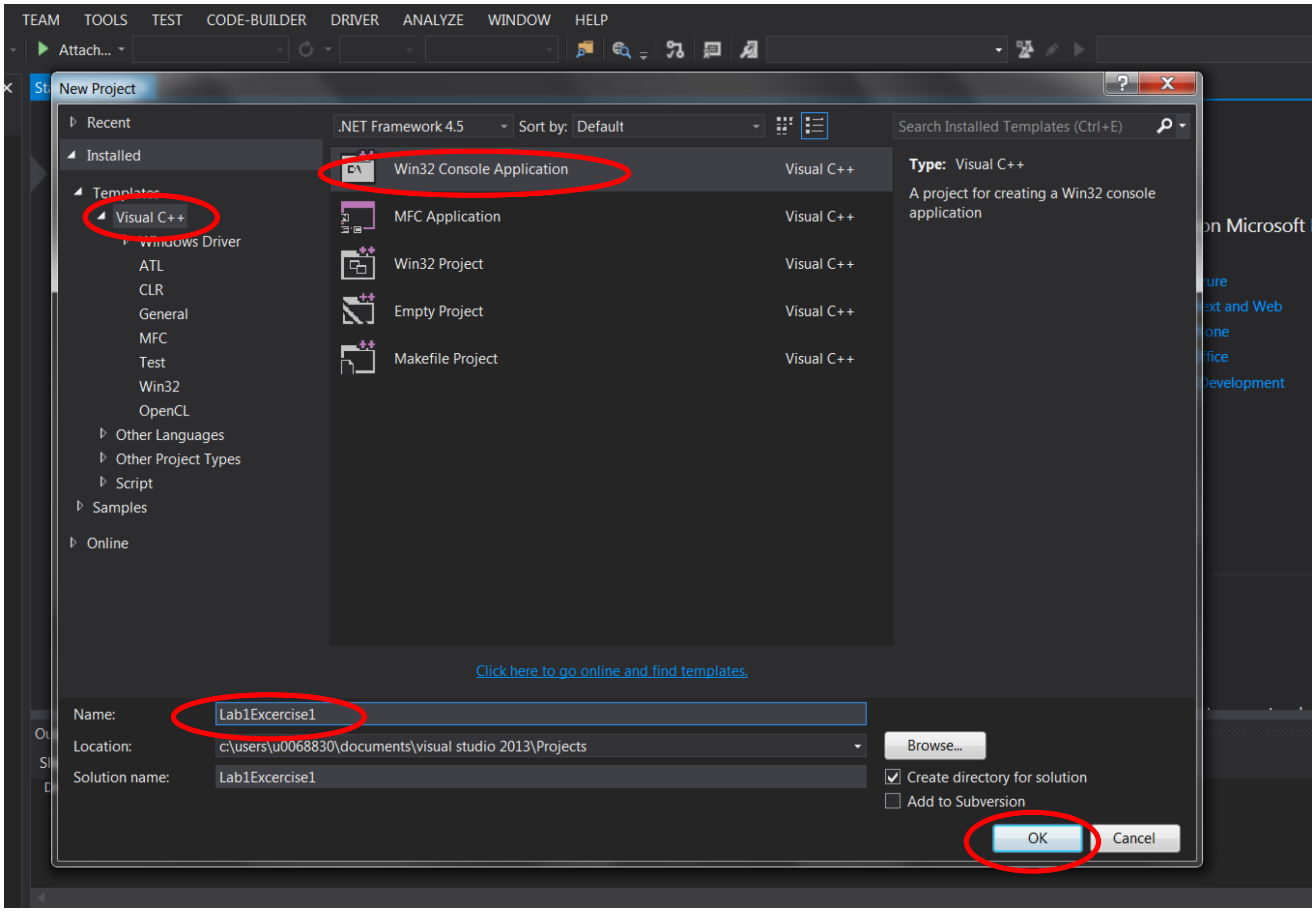
Start Up: Create Project



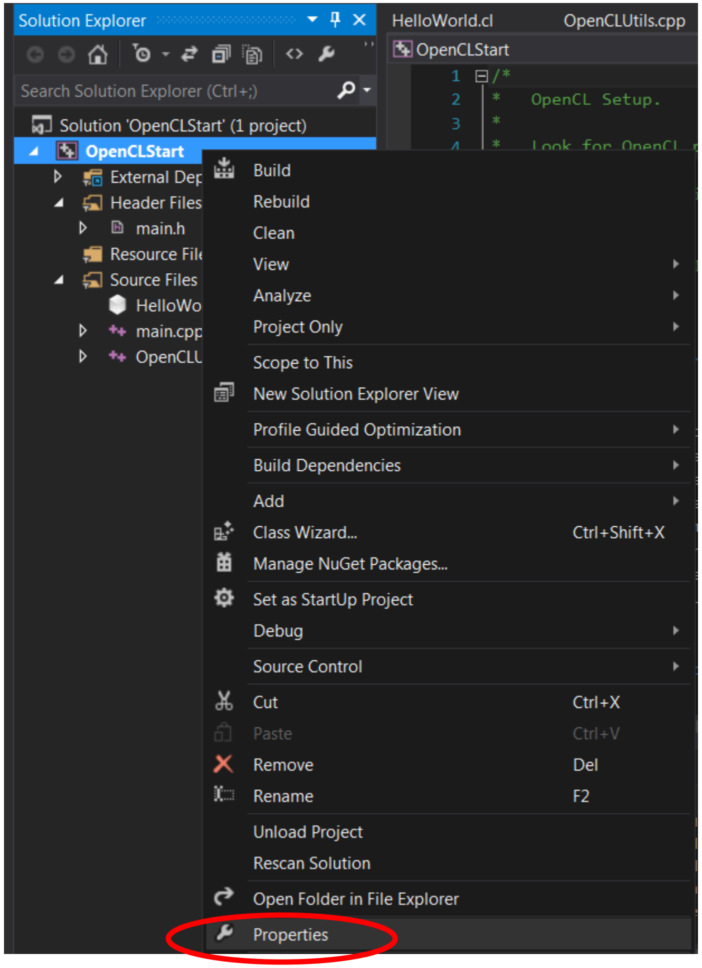
Start Up: Create Project



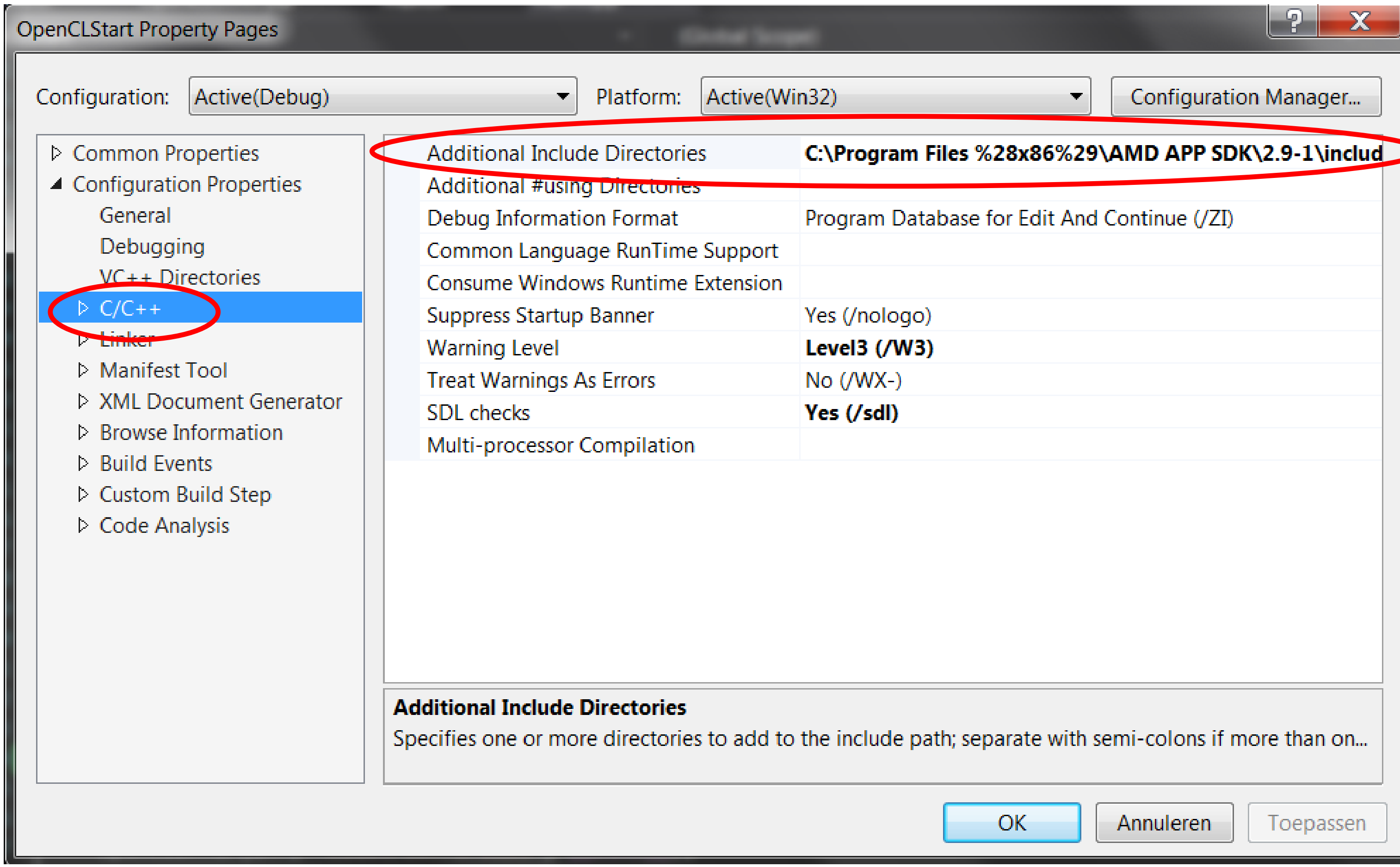
Start Up: Create Project



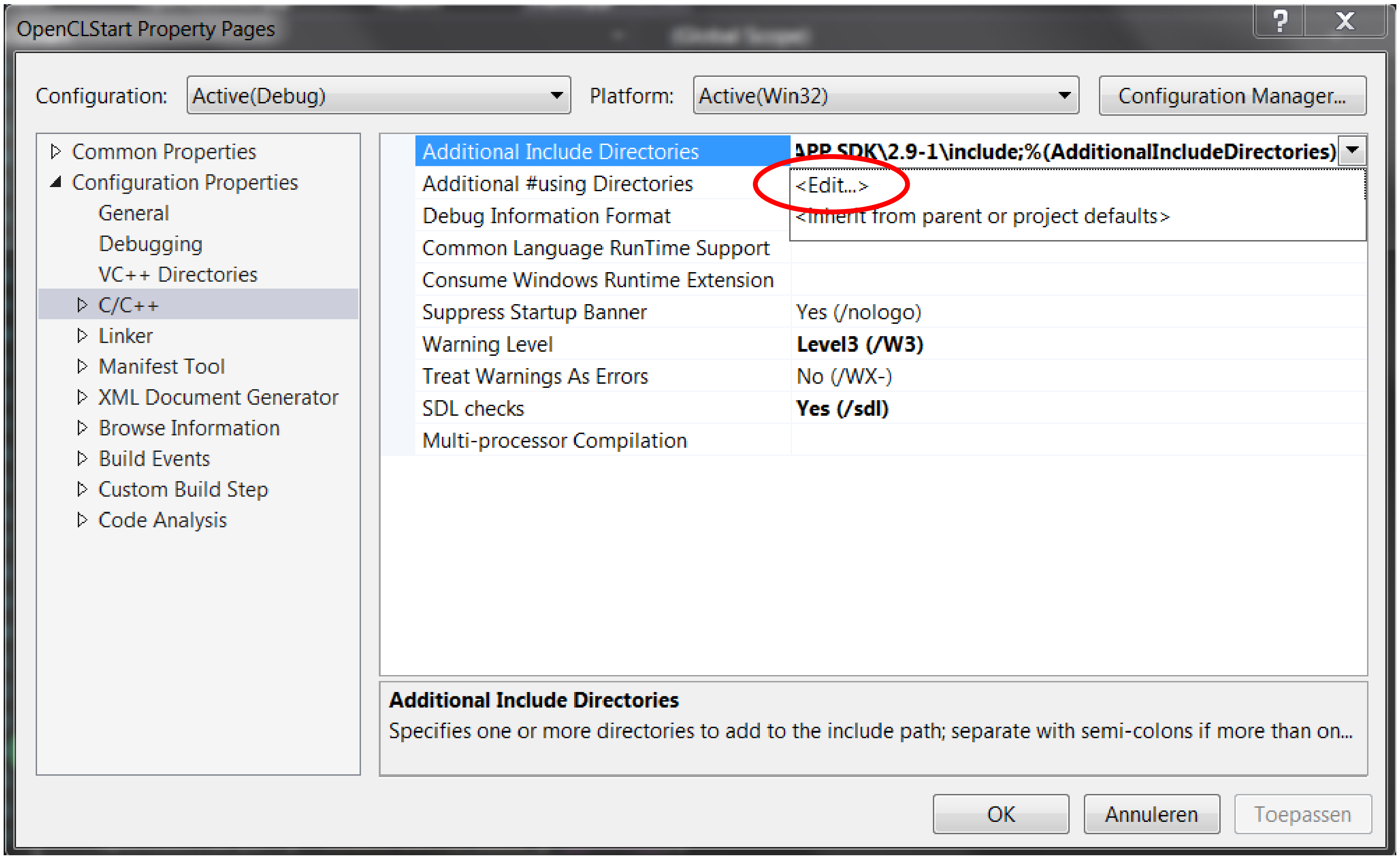
Start Up: Add Library Files



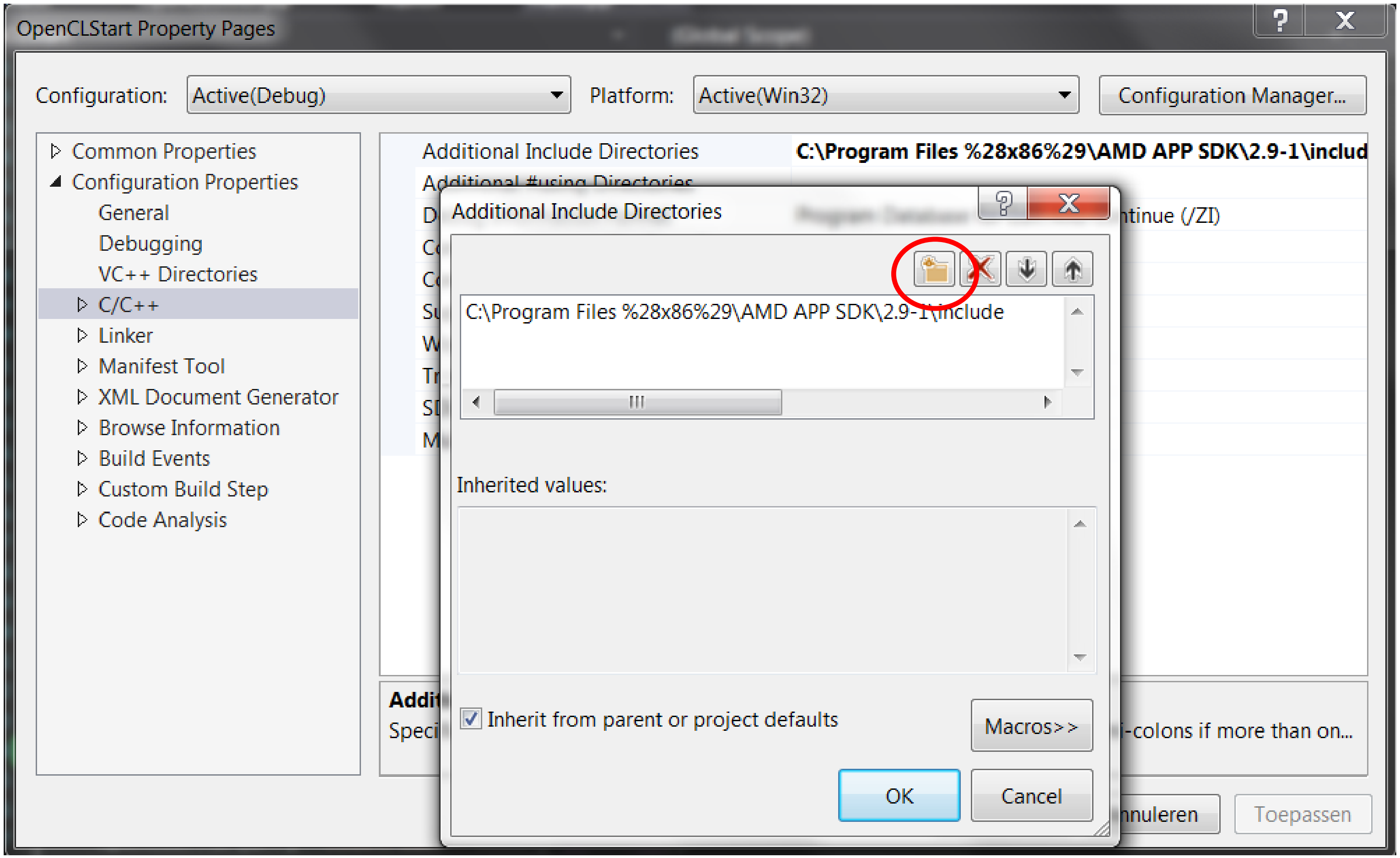
Start Up: Add Library Files



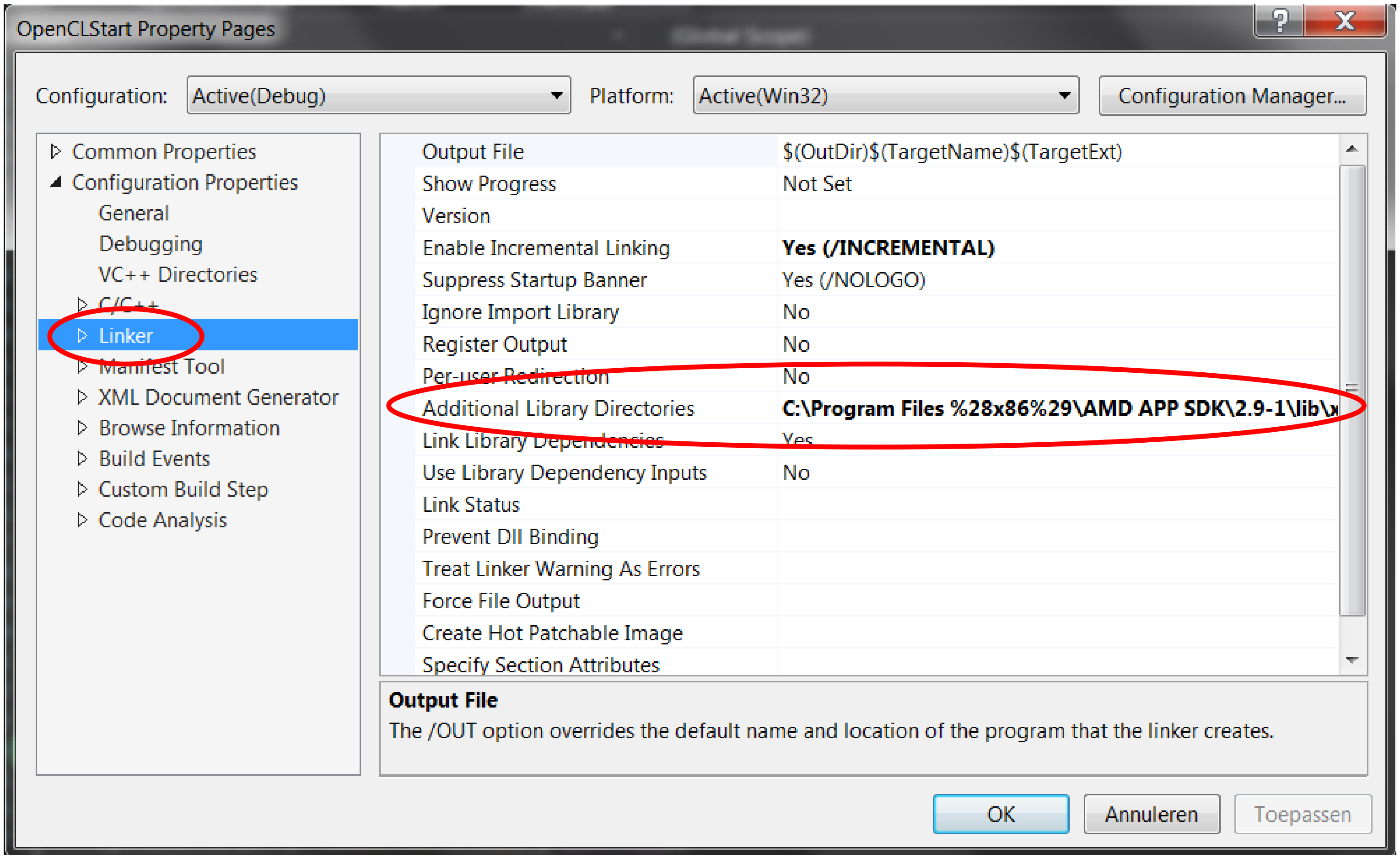
Start Up: Add Library Files



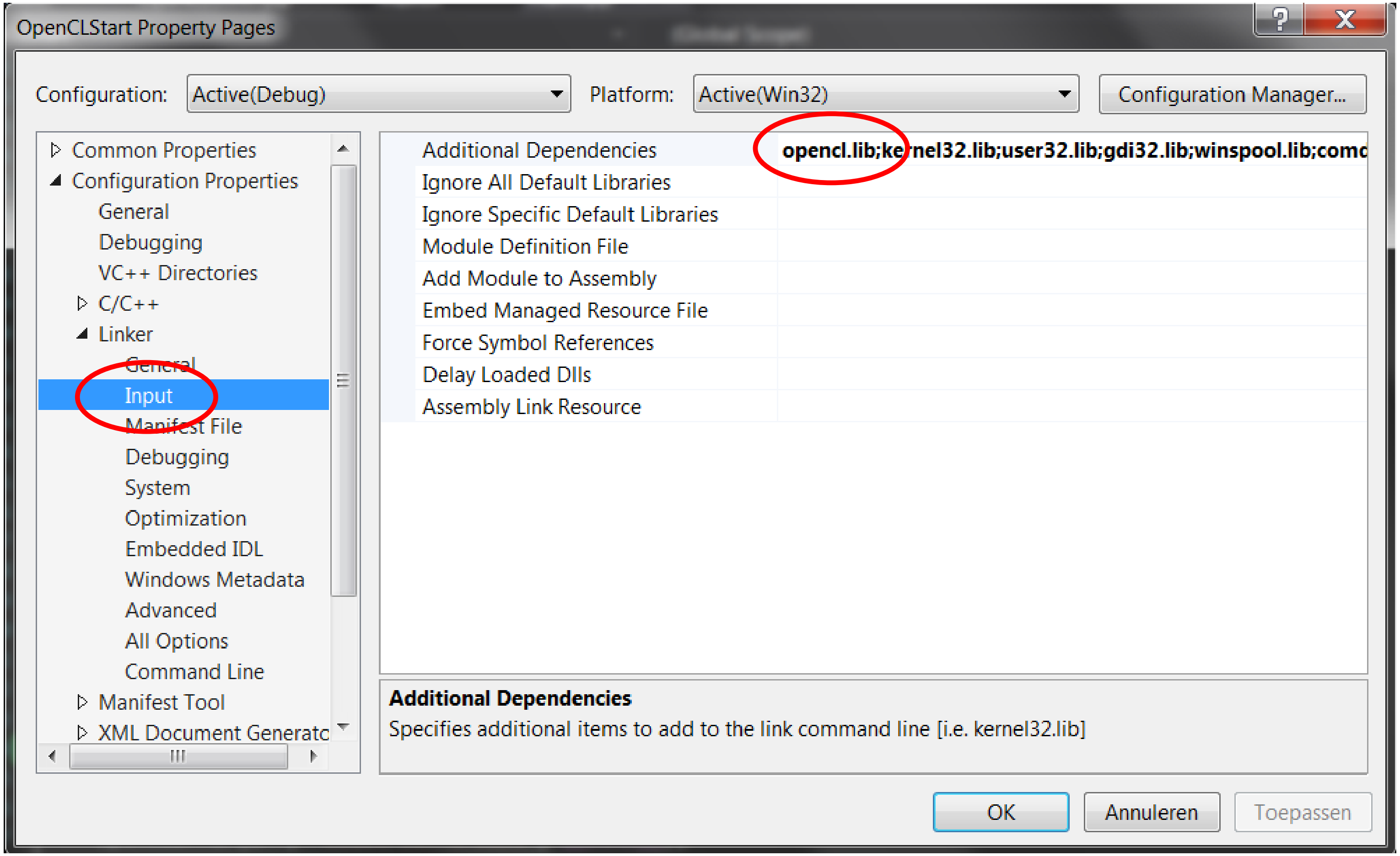
Start Up: Add Library Files



Start Up: Add Library Files

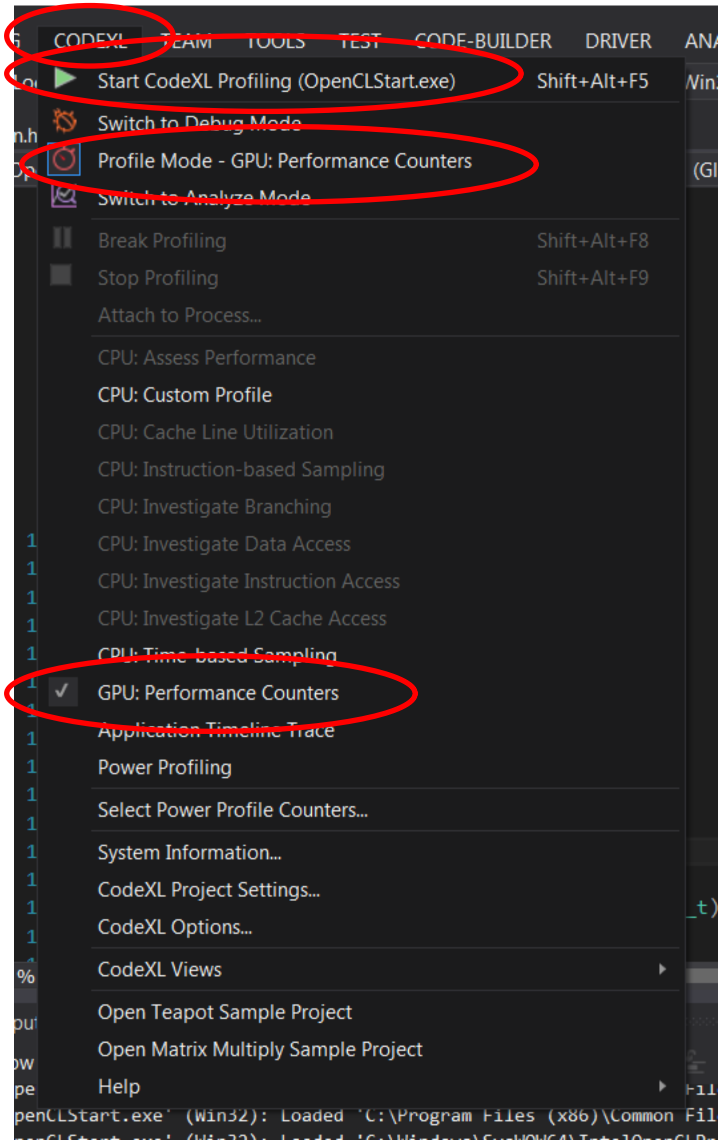


Start Up: Add Library Files



Code Profiling

- CodeXL integrates in Visual Studio and can be used from there



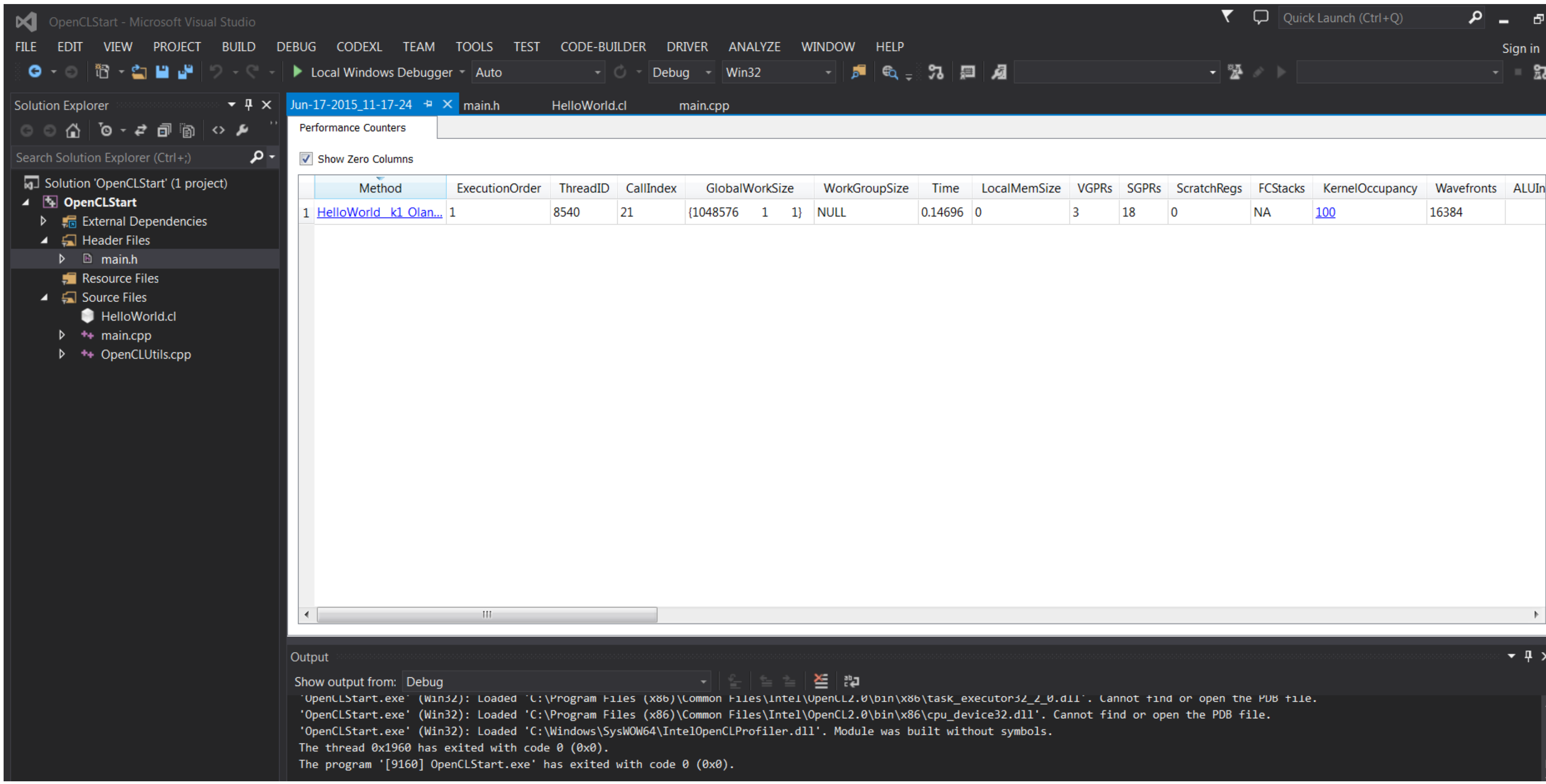
Code Profiling

- The console window shows some extra output

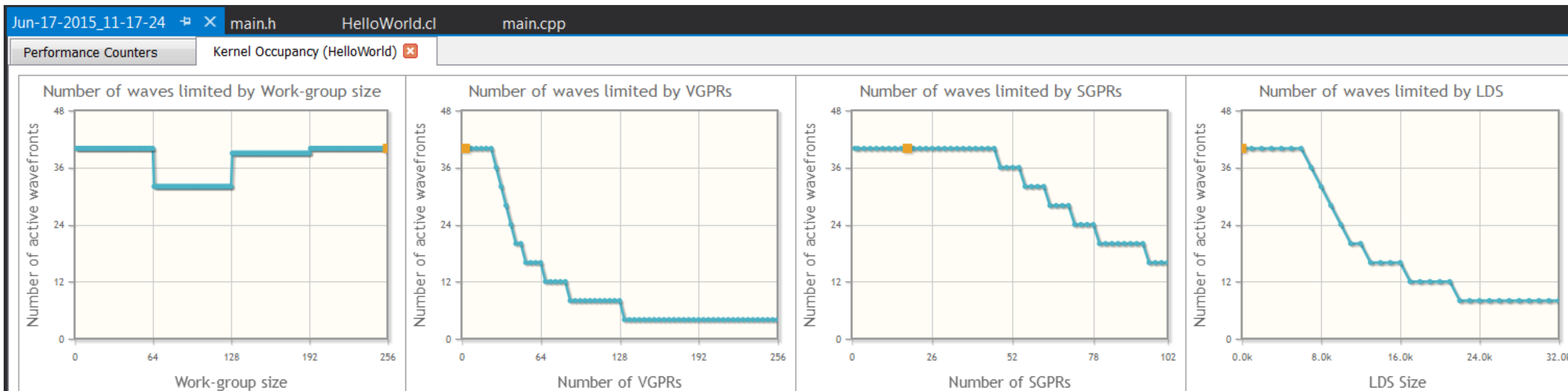
```
C:\Users\u0068830\Documents\De Nayer\Research\Desire\Multicore Programming\ProjectSolutions...
AMD CodeXL GPU Profiler V3.1.8364 is Enabled
AMD CodeXL GPU Profiler Kernel occupancy module is enabled
Which platform to use?
Platform 1: Intel(R) OpenCL
Platform 2: AMD Accelerated Parallel Processing
Platform 3: Experimental OpenCL 2.0 CPU Only Platform
2
Available devices on the chosen platform:
1: GPU
2: CPU
1
Oland
Available device information:
Number of compute units: 6
Max clock frequency: 900MHz
Global memory size: 2048MB
Local memory size: 32KB
Supported OpenCL version: OpenCL 1.2 AMD-APP (1642.5)
Device Available: Yes


Writing to file: C:/Users/u0068830/AppData/Roaming/AMD/CodeXL/OpenCLStart_ProfilerOutput/Jun-17-2015_11-17-24/sp_tmp.HelloWorld__k1_Oland1.il
Writing to file: C:/Users/u0068830/AppData/Roaming/AMD/CodeXL/OpenCLStart_ProfilerOutput/Jun-17-2015_11-17-24/sp_tmp.HelloWorld__k1_Oland1.isa
Writing to file: C:/Users/u0068830/AppData/Roaming/AMD/CodeXL/OpenCLStart_ProfilerOutput/Jun-17-2015_11-17-24/sp_tmp.HelloWorld__k1_Oland1.cl
```

Code Profiling



Code Profiling



Variable	Value	Device Limit
Device Info		
Device name	Oland	
Number of compute units	6	
Max number of waves per compute unit	40	
Max number of work-groups per compute unit	16	
Wavefront size	64	
Kernel Info		
Kernel name	HelloWorld	
Vector GPR usage per work-item	3	256
Scalar GPR usage per work-item	18	102
LDS usage per work-group	0	65536

Questions?

Contact

Ing. Dirk Van Merode MSc.
Project Coordinator DESIRE
Thomas More | Campus De Nayer
Technology & Design
J. P. De Nayerlaan 5
2860 Sint-Katelijne-Waver

Belgium
Tel. + 32 15 31 69 44
Gsm + 32 496 26 84 15
dirk.vanmerode@thomasmore.be
Skype dirkvanmerode
www.thomasmore.be

Dr. Ing. Peter Arras MSc.
International Relations Officer
KU Leuven | Campus De Nayer
Faculty of engineering technology
J. P. De Nayerlaan 5
2860 Sint-Katelijne-Waver

Belgium
Tel. + 32 15 31 69 44
Gsm + 32 486 52 81 96
peter.arras@kuleuven.be
Skype pfjlarras
www.iiw.kuleuven.be

Tempus