(PRACTICAL)
COMPUTATIONAL PHYSICS

Physics 551
Lecture 15

NOTATION

Extra Reading

Optional Exercise

Recommended

- This lecture slides for this course will attempt to use a uniform notation throughout. A normal paragraph looks like this.

☞ *Italicized paragraphs with pen bullets will indicate definitions, with the defined word or phrase shown in* ***SMALL-CAPS****.*

✏ Pencil bullets will indicate the introduction of **new notation**.

☞ Pointing hand bullets indicate important points that might otherwise be overlooked.

ANNOUNCEMENTS

ANNOUNCEMENTS

- To **clone** this week's **LabVIEW demonstration materials** please open the ***Git Shell*** utility and invoke

```
$ git clone https://github.com/hughdickinson/CompPhysL15Labview.git
  C:/Users/ComputationalPhysics/Documents/labview/lecture15
```

- To **clone** this week's **C++ demonstration materials** please open the ***Git Shell*** utility and invoke

```
$ git clone https://github.com/hughdickinson/CompPhysL15CPP.git
  C:/Users/ComputationalPhysics/Documents/cPlusPlus/lecture15
```

☞You can also find this command on the Blackboard Learn website.

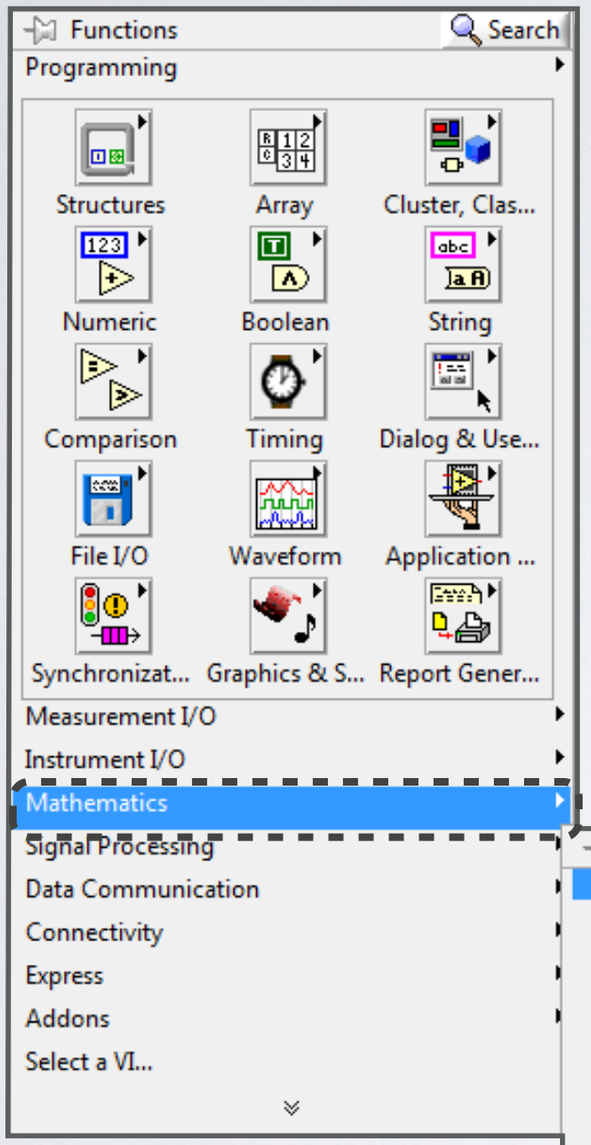
LABVIEW

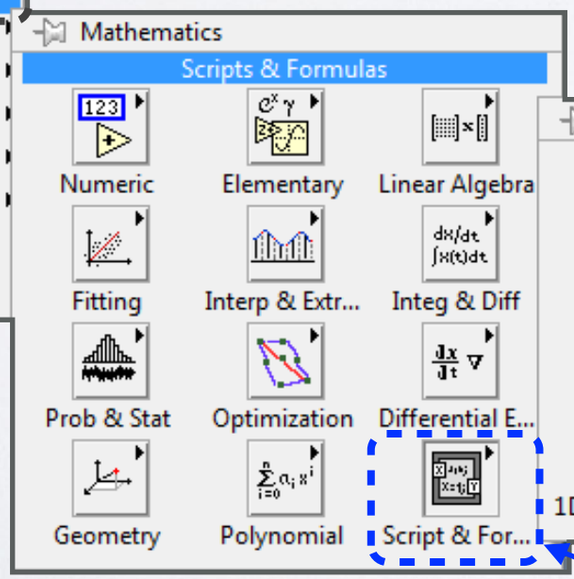
SIMPLIFYING THE BLOCK DIAGRAM

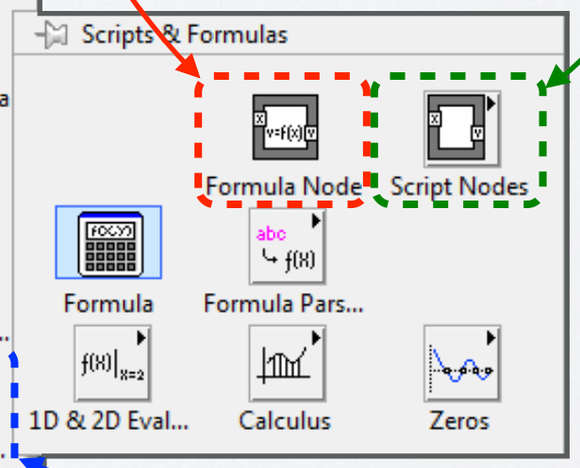
FORMULA NODES

- Implementing **moderately complex** data manipulation by connecting functions from the ***Numeric Palette*** rapidly **complicates** the block diagram.
  - ☞ *LabVIEW provides a* ***Formula Node****, which allows complicated* ***data-manipulation*** *operations to be defined using a* ***C++*** *or* ***C-like syntax****.*
- Formula nodes accept an **arbitrary** number of named **input** terminals and may output data through an **arbitrary** number of **output** terminals.

FORMULA NODES







Formula Nodes

Enable data manipulation using a C-style syntax

Script Nodes

*Enable invocation of MATLAB scripts **if** MATLAB is available on the machine running LabVIEW*

Scripts & Formulas

Palette

FORMULA NODE DETAILS

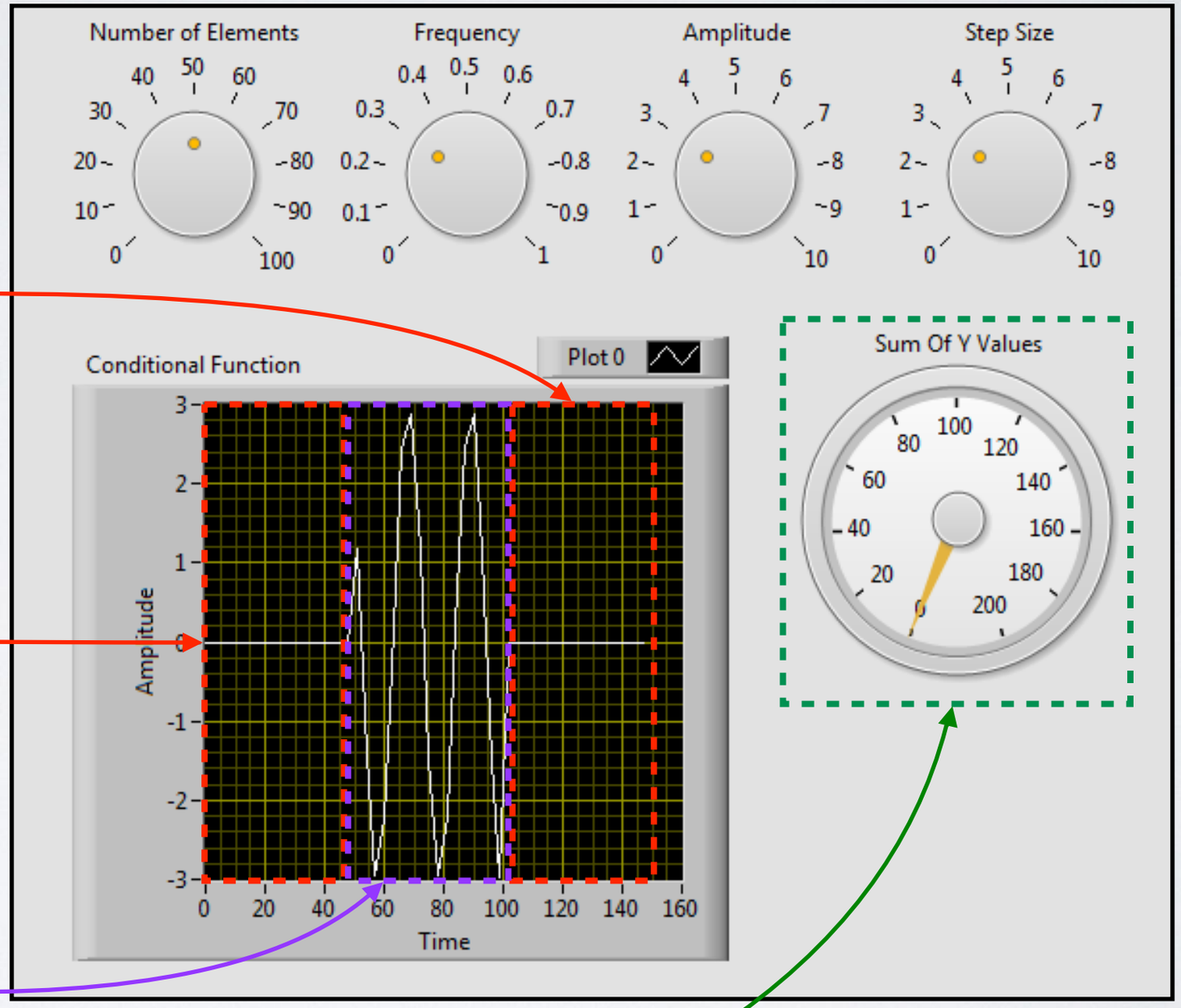
- ☞Formula nodes cannot ***generate* array-type** data. Any array-type outputs must have an **identically named and dimensioned** array-type input.
- ☞Formula nodes can substantially simplify block diagrams that implement **conditional branching** using ***Case Structures***.
- ☞Several **built-in mathematical functions** can be used within ***Formula Nodes***. See the help for a full list of available functions.
- ☞The `sizeOfDim(...)` function can be used to obtain the extent of array-type input data.

FORMULA NODE EXAMPLE

Final third
Set to zero

First third
Set to zero

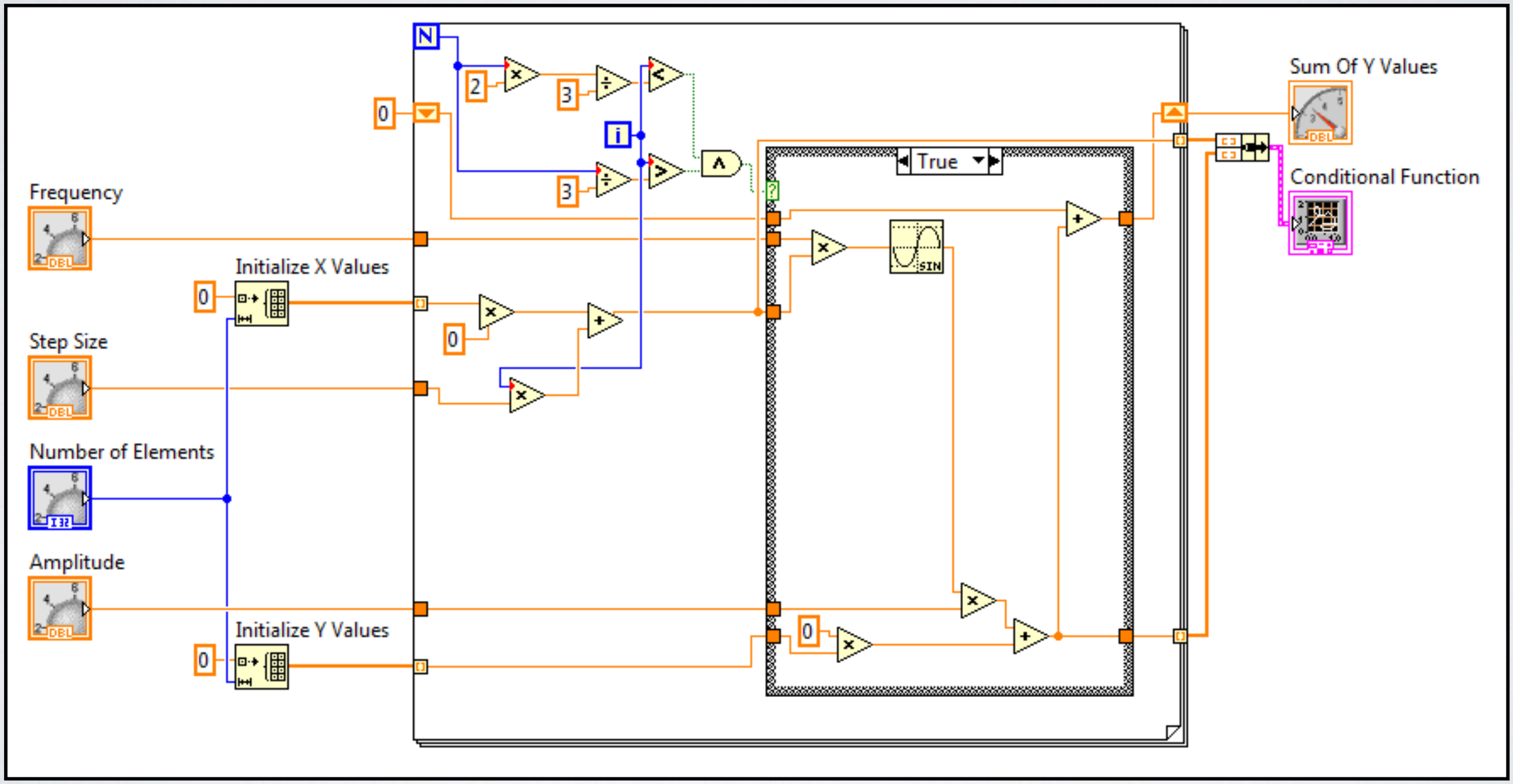
Second third
Set to `A*sin(f*x)`



Compute sum of all samples.

FORMULA NODE EXAMPLE

THE HARD WAY!

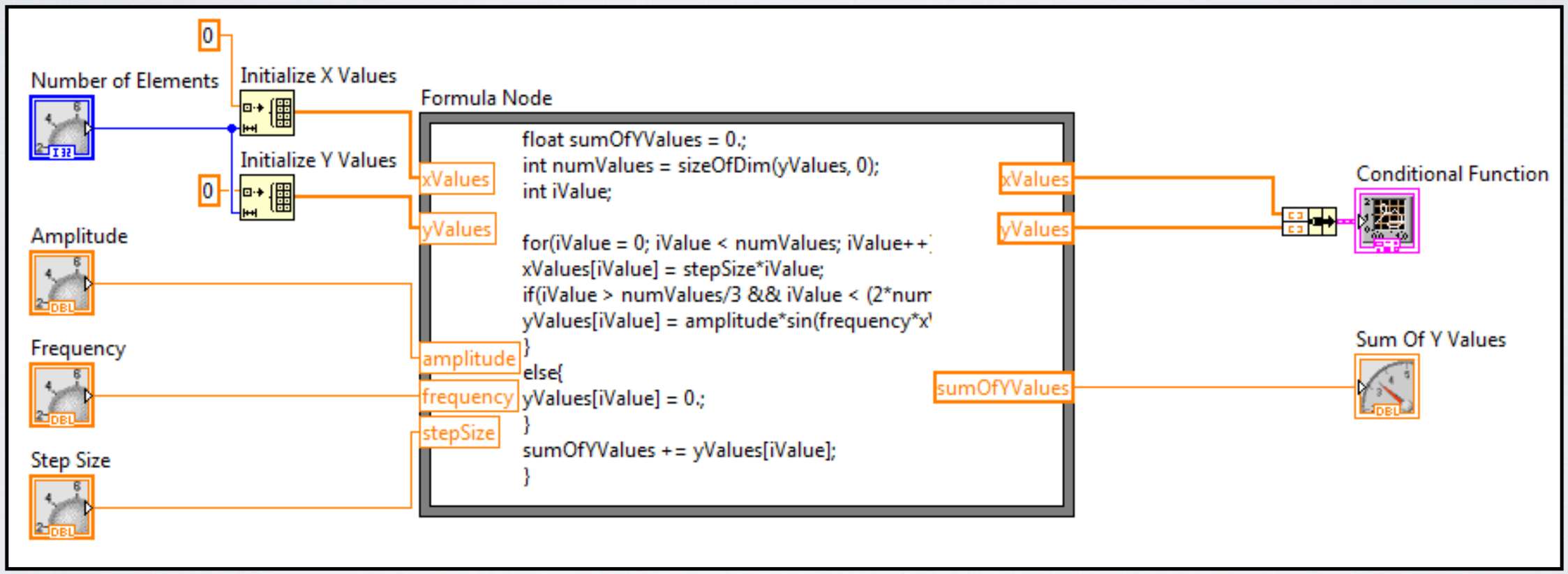


DEMONSTRATION

Using **Formula Nodes**

FORMULA NODE EXAMPLE

THE EASY WAY!



INVOKING EXTERNAL UTILITIES

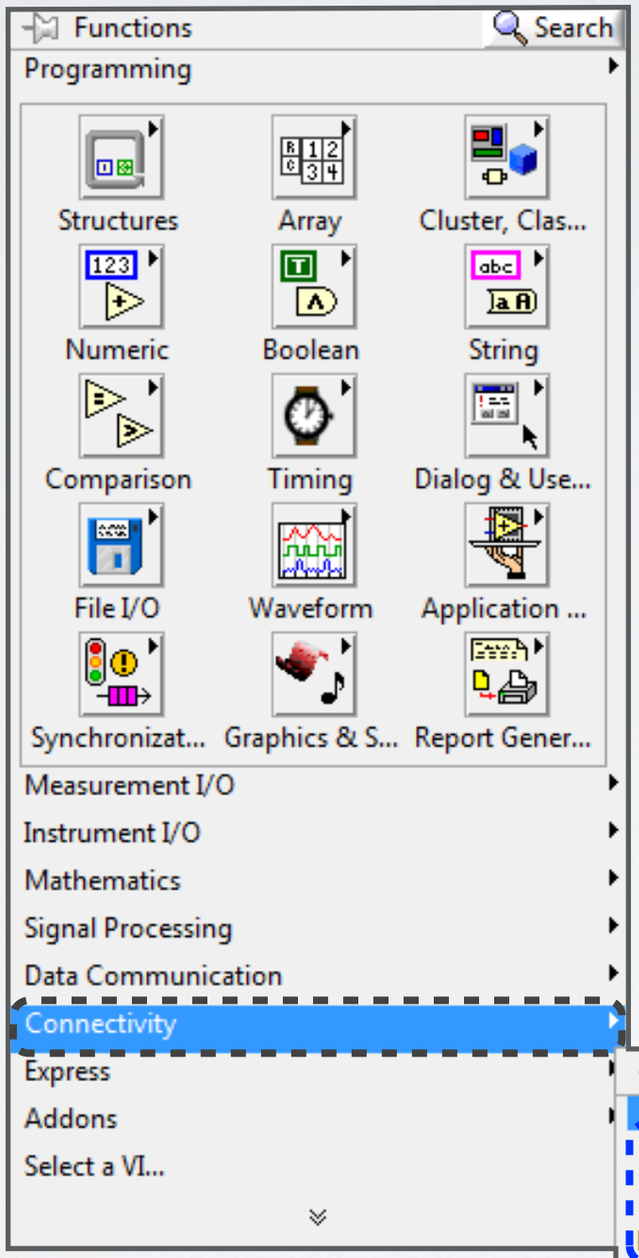
AUGMENTING LABVIEW WITH EXTERNAL UTILITY

- LabVIEW provides an **extensive** framework of **built-in** data acquisition and analysis **functionality**.
- Occasionally, highly **specific analyses** may benefit from **additional functionality** provided by external utilities e.g. *Python*.

☞ *LabVIEW provides the **System Exec VI**, which allows an external utility to be invoked from the block diagram.*

☞ The System Exec VI provides a lot of **flexibility** to control how the utility is invoked and how its output is handled.

THE SYSTEM EXEC VI

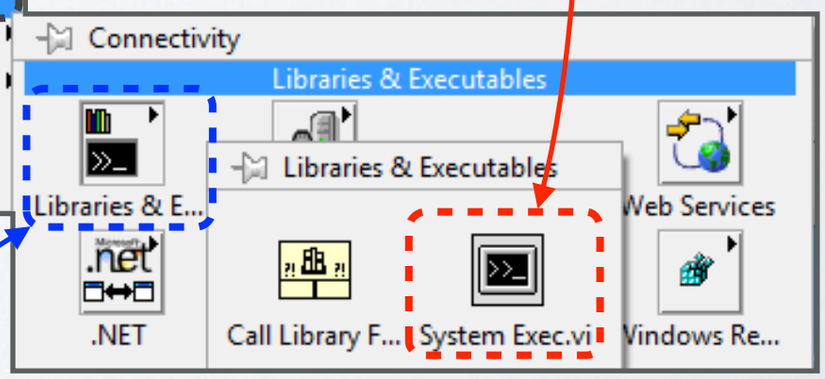


System Exec VI

Enables invocation of any ***executable*** *utility that could be run from the system* ***command prompt*** *or* ***shell****. Command line* ***arguments*** *can also be specified and any* ***output*** *can be* ***captured****.*

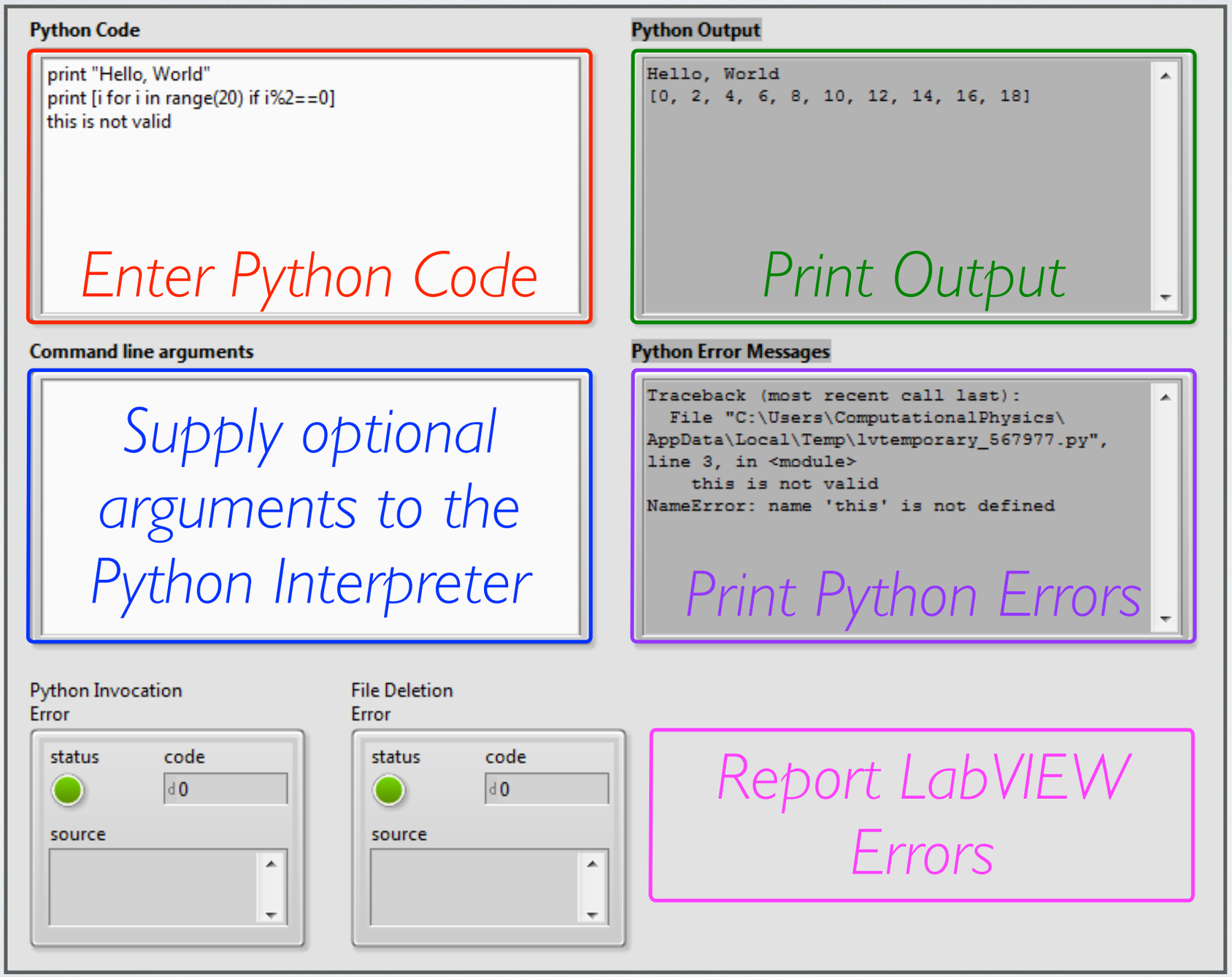
Libraries & Executables

Palette



SYSTEM EXEC EXAMPLE

PYTHON INTERPRETER



DEMONSTRATION

Using the **System Exec** VI

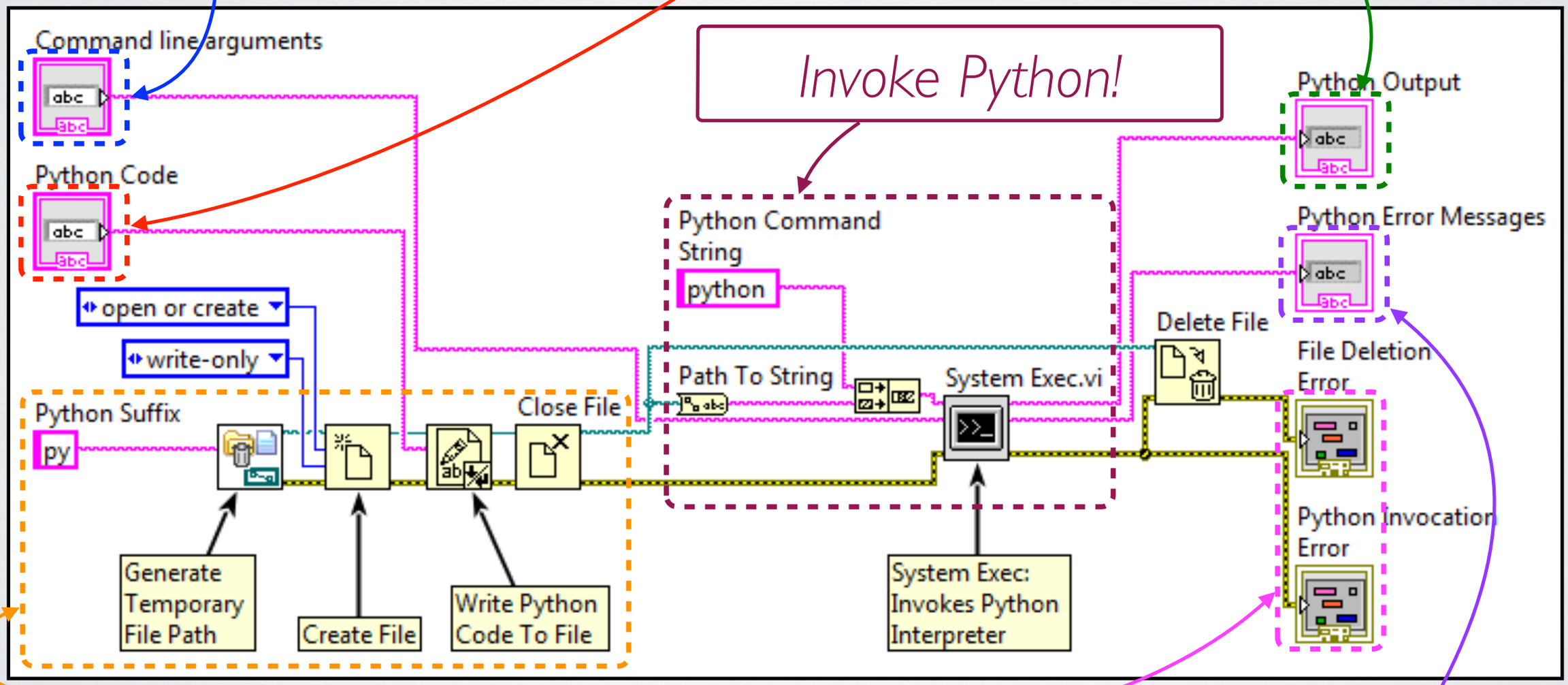
SYSTEM EXEC EXAMPLE

PYTHON INTERPRETER

Python Interpreter Arguments

Enter Python Code

Print Output



Generate Python Script

Report LabVIEW Errors

Print Python Errors

WORKING WITH C++ SHARED LIBRARIES

REUSING LEGACY ANALYSIS CODE

- LabVIEW is able to **import functions** provided by **shared libraries** and **generate** corresponding **VIs** that can be added to the Block Diagram.
  - ☞ LabVIEW automatically **maps** between function **parameter** and **return** types and LabVIEW data types that can be connected.
  - ☞ An **intuitive *wizard*** interface is provided to **simplify** the importation process, making it **straightforward** to reuse **legacy analysis routines**.
  - ☞ You may have to **build a shared library in Windows**.

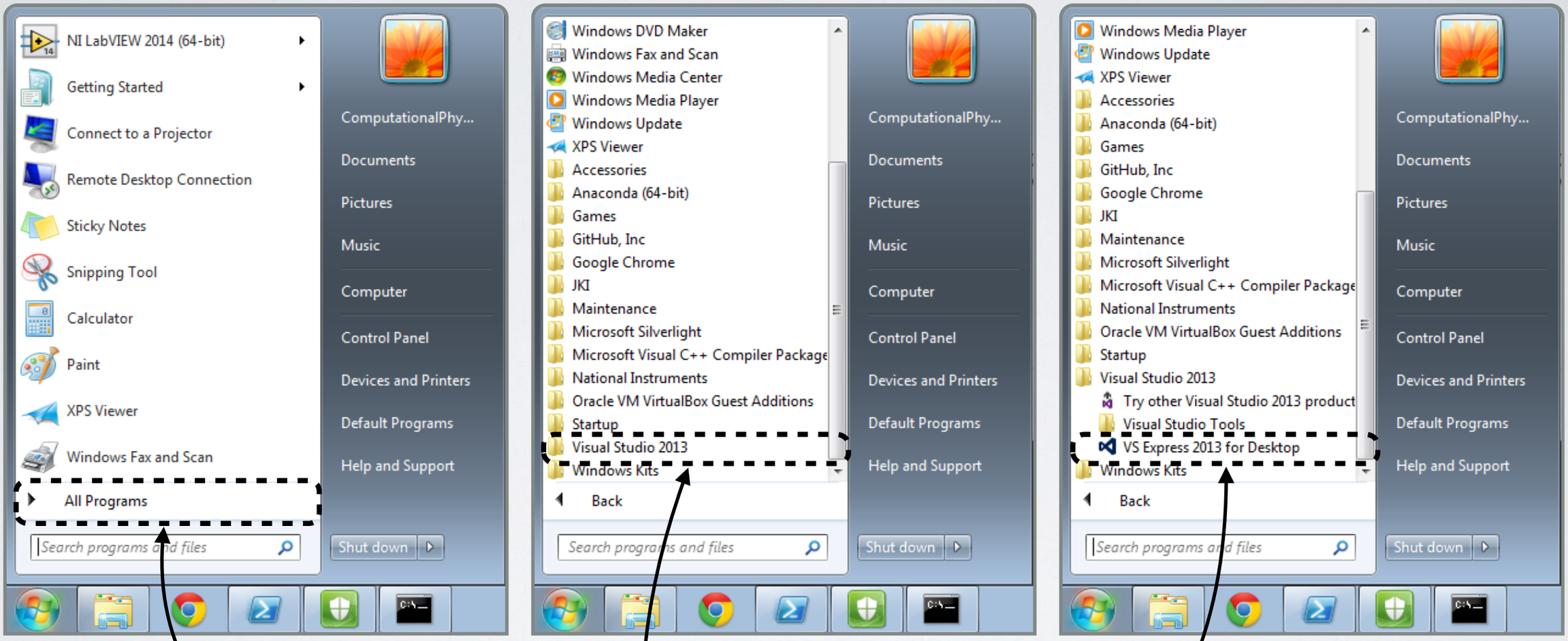
DEMONSTRATION

Importing **shared library functions** as **LabVIEW VIs**

STEP 1:

START VISUAL STUDIO

START VISUAL STUDIO



1) All Programs

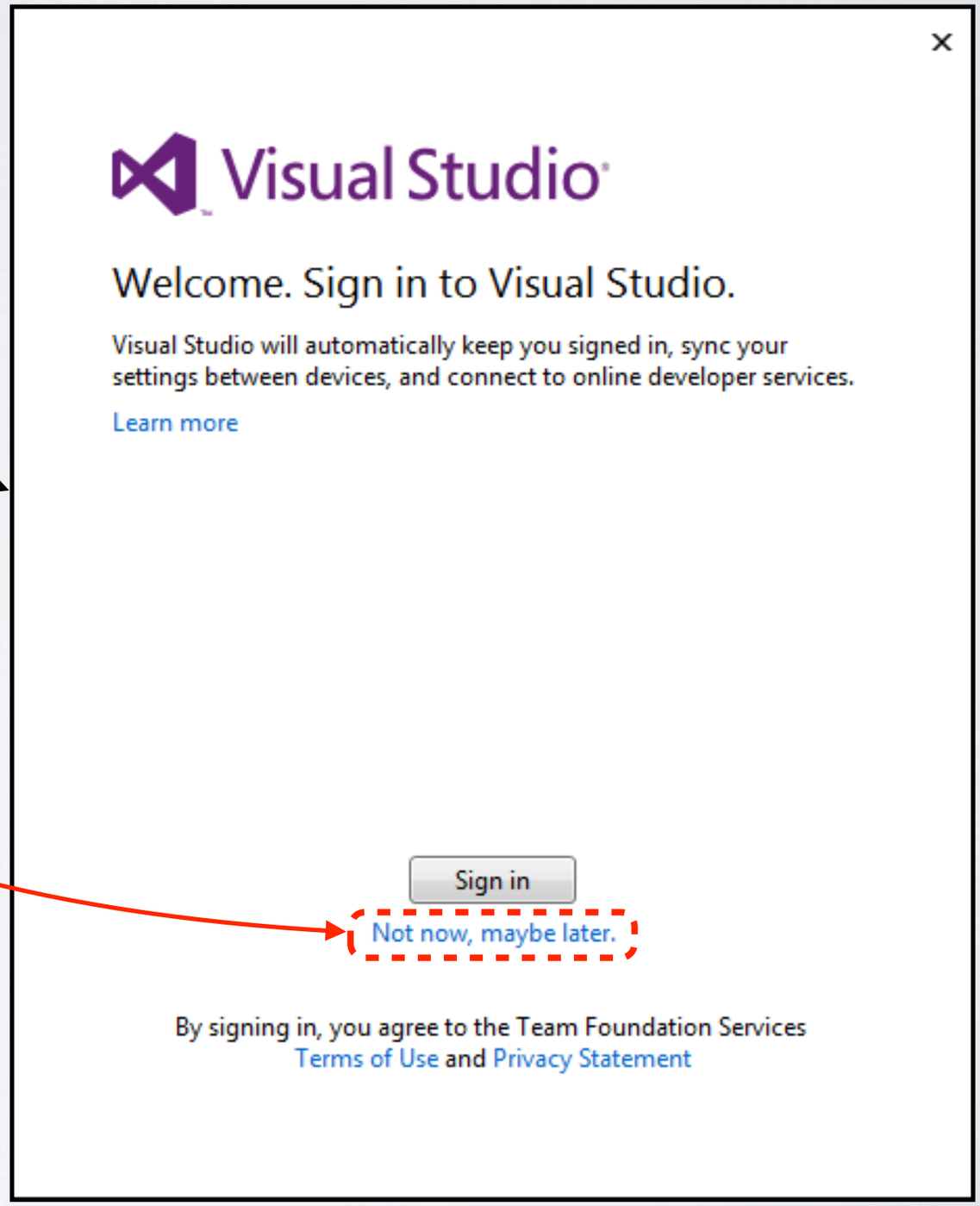
2) Visual Studio 2013

3) VS Express 2013 for Desktop

START VISUAL STUDIO

First Time Startup

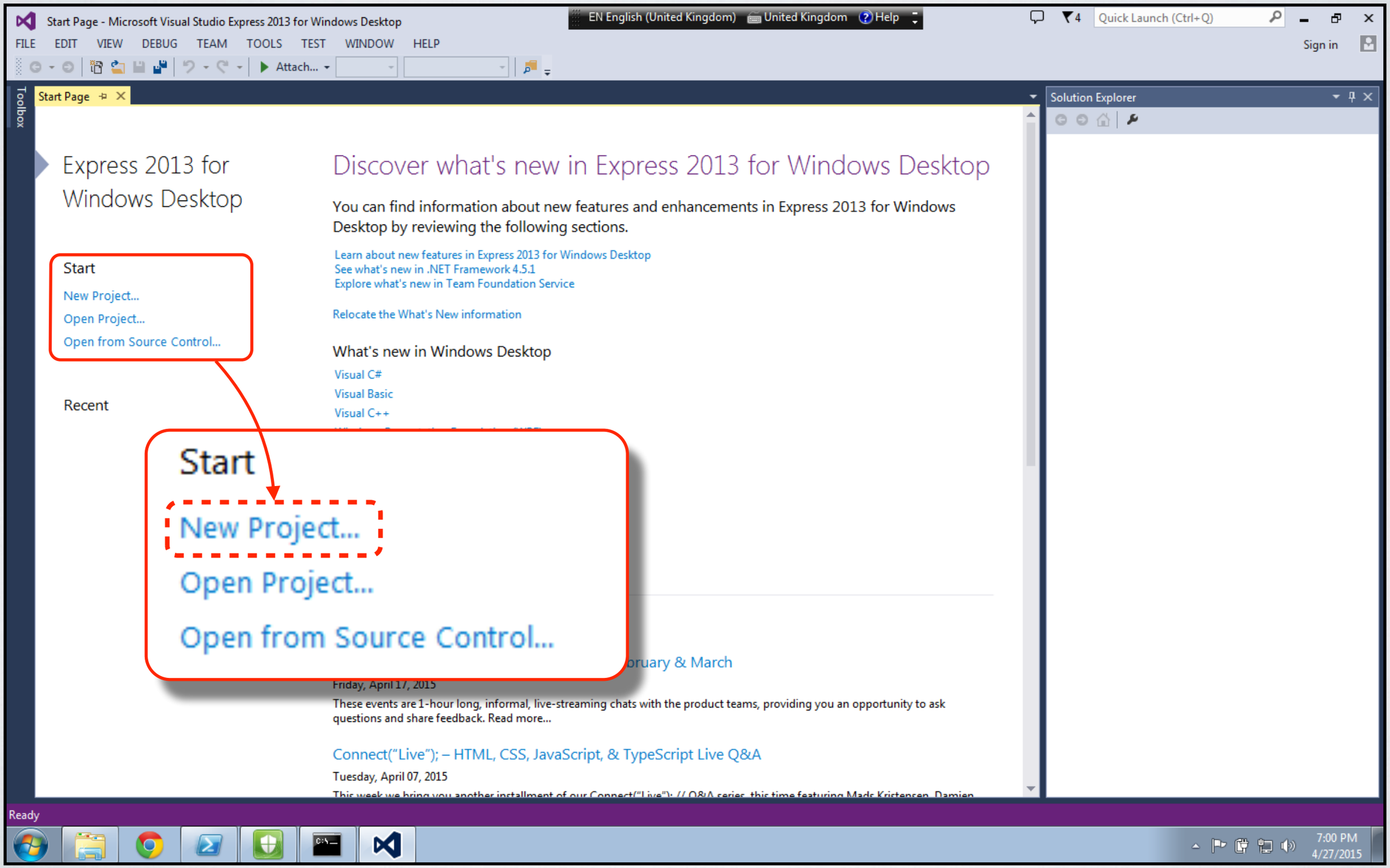
Skip Sign In



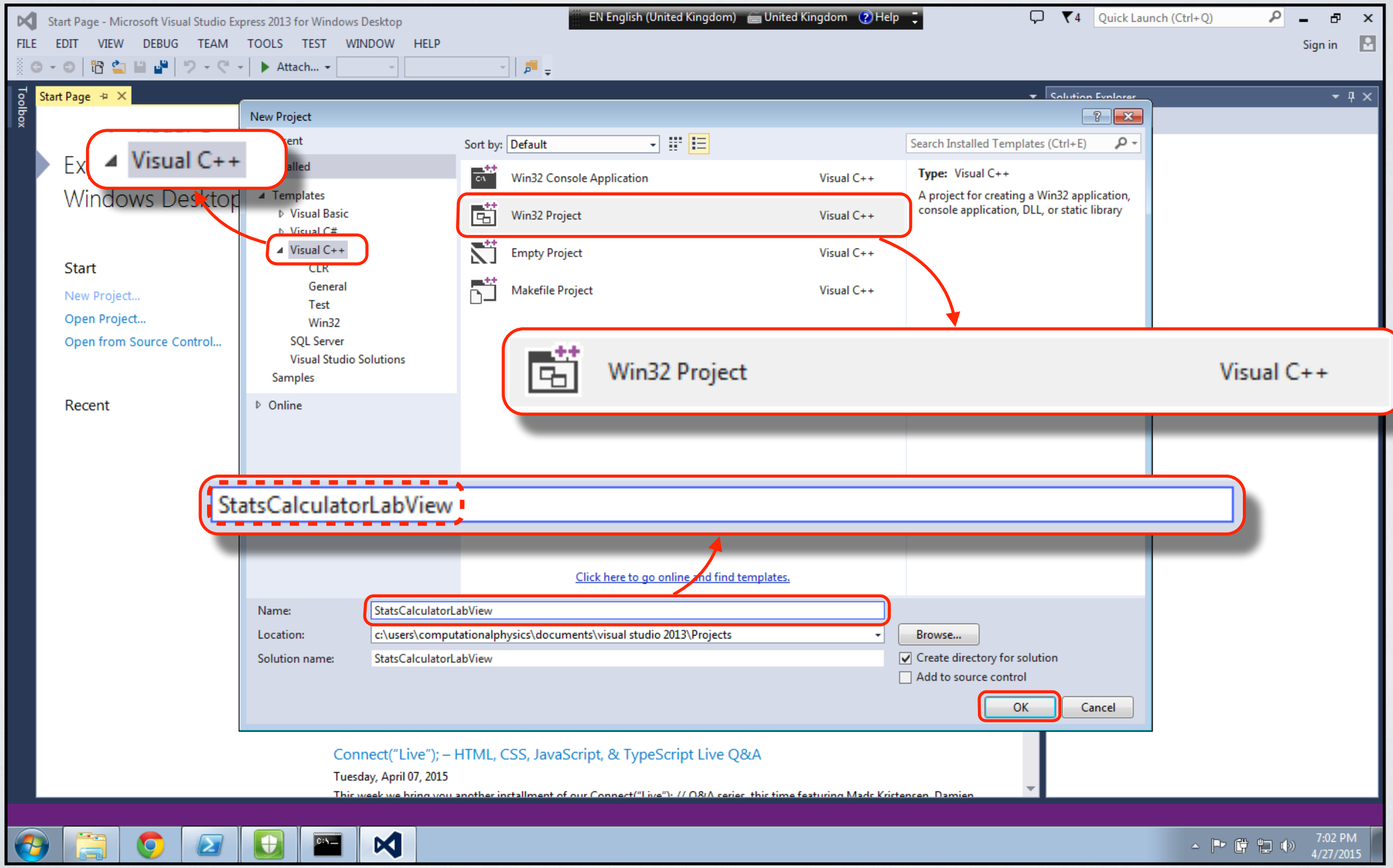
STEP 2:

CREATE A NEW PROJECT

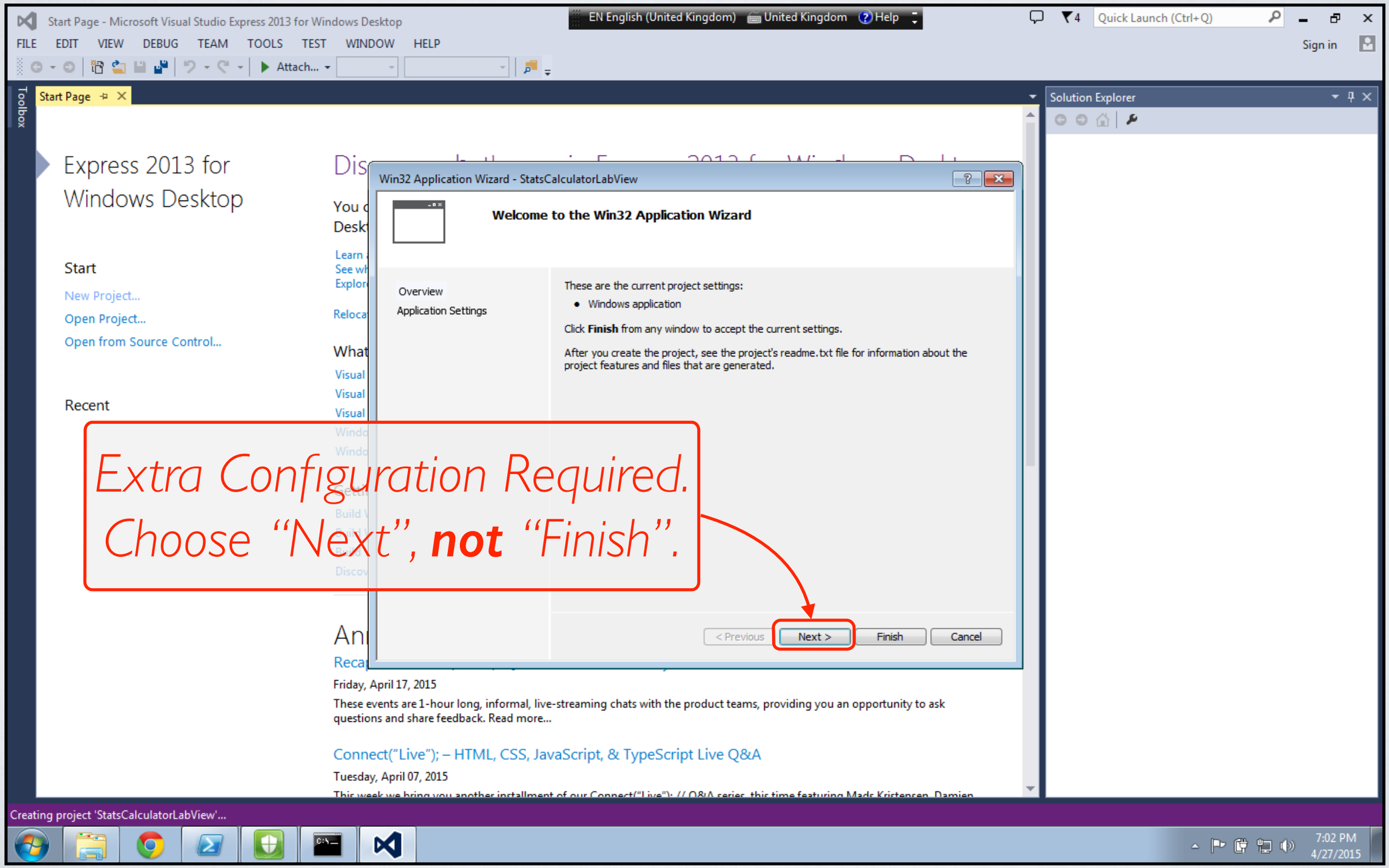
CREATE NEW PROJECT



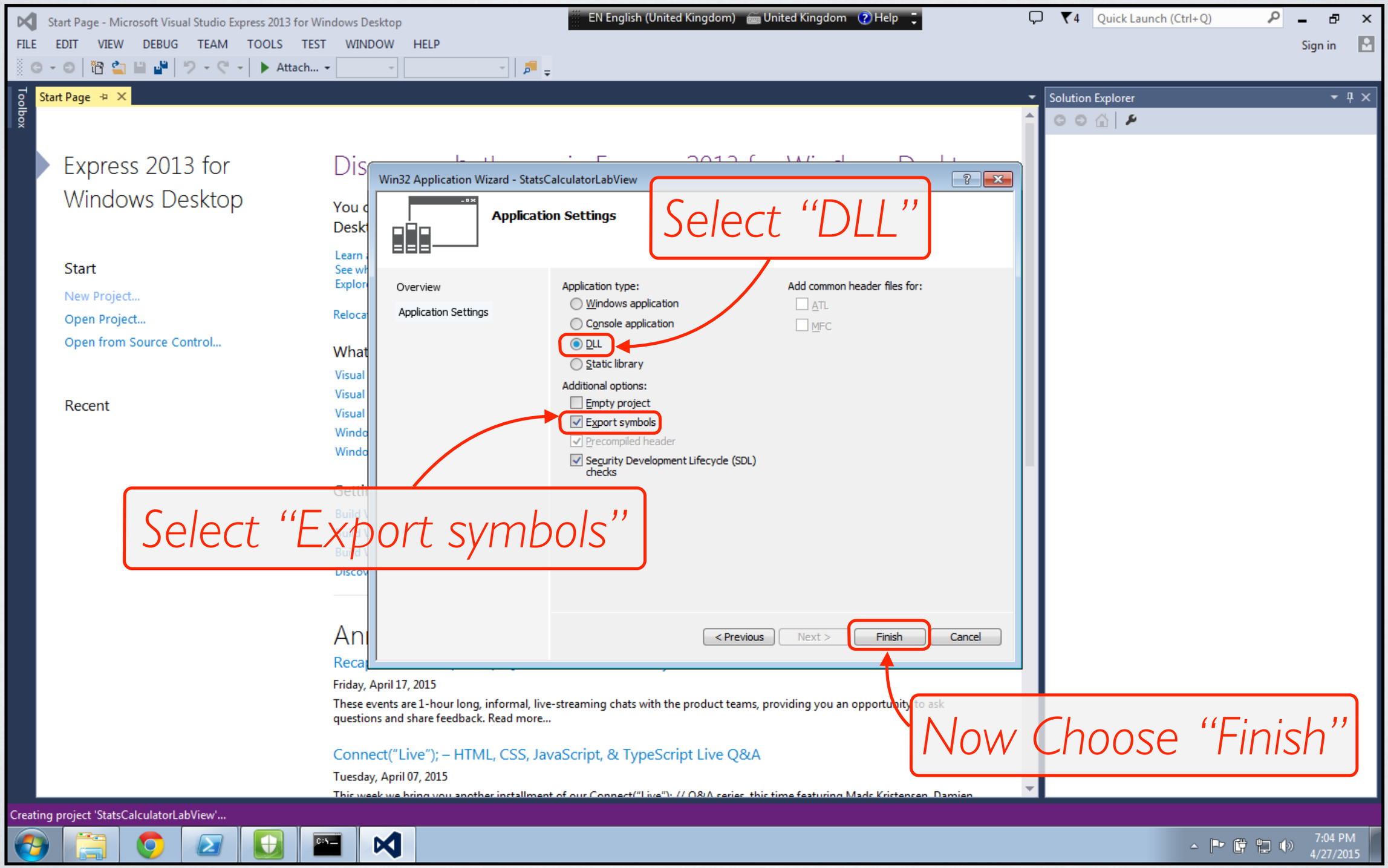
CREATE NEW PROJECT



CREATE NEW PROJECT



CREATE NEW PROJECT

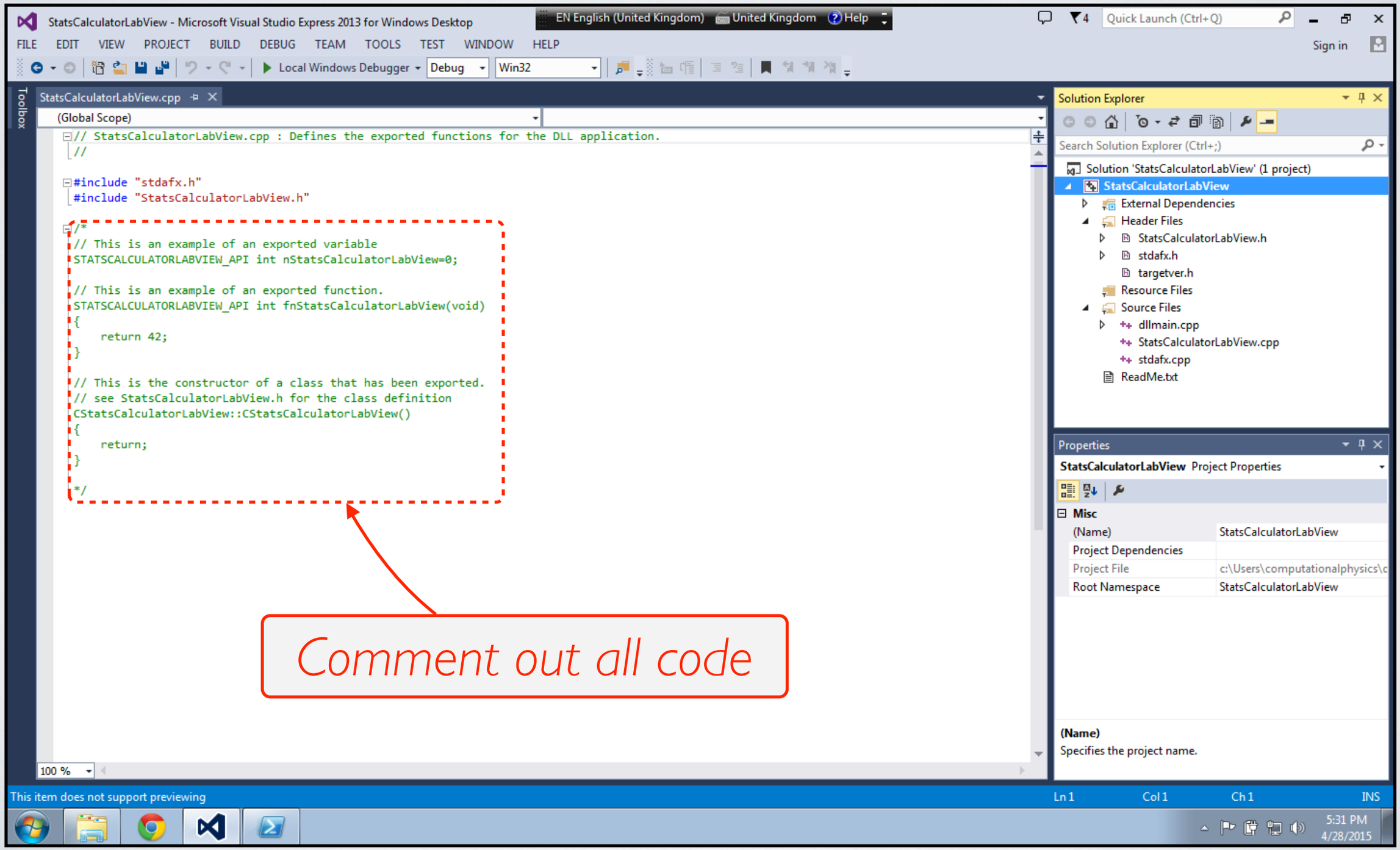


STEP 3:

MODIFY BOILERPLATE CODE

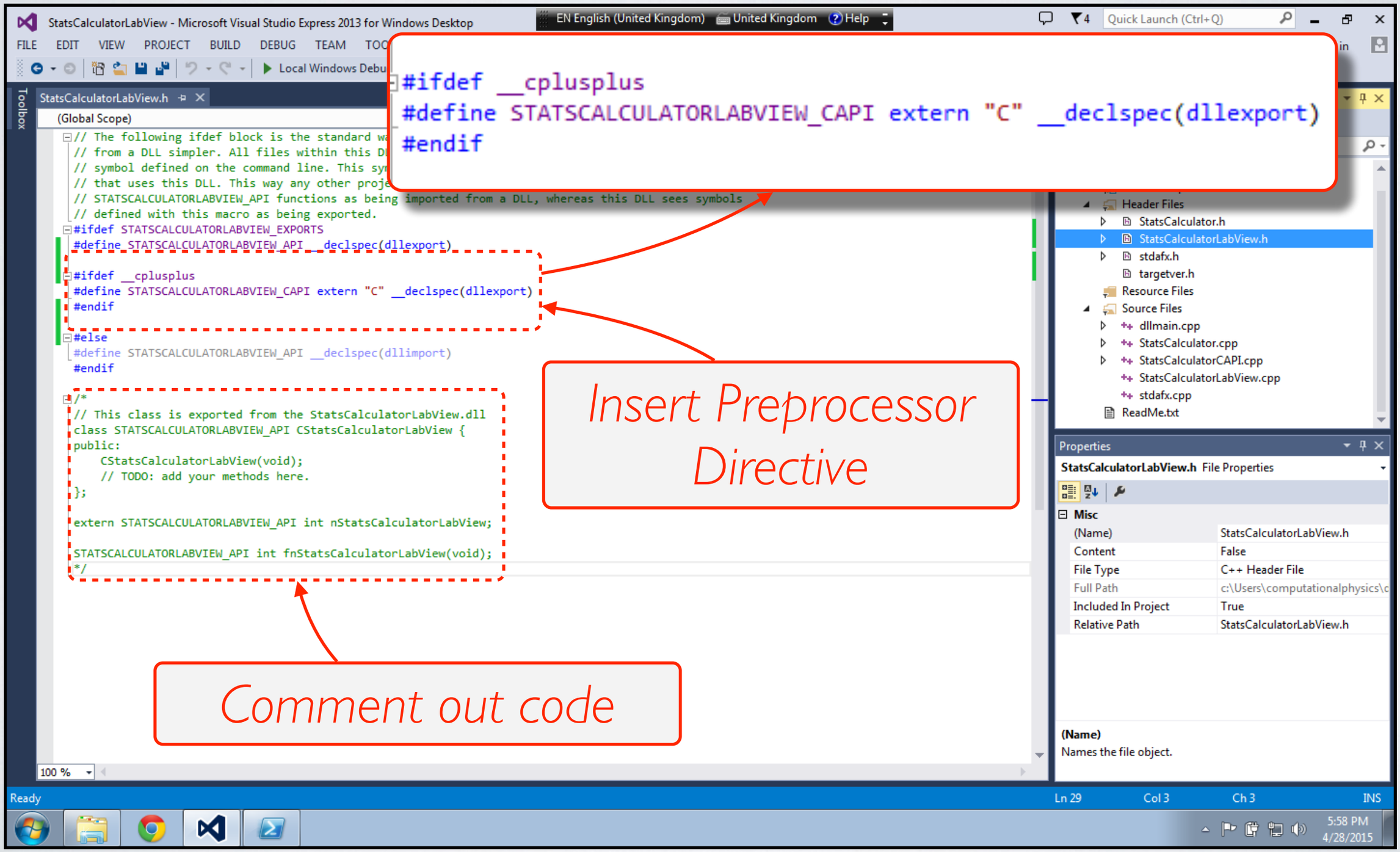
MODIFY BOILERPLATE CODE

(*StatsCalculatorLabView.cpp*)



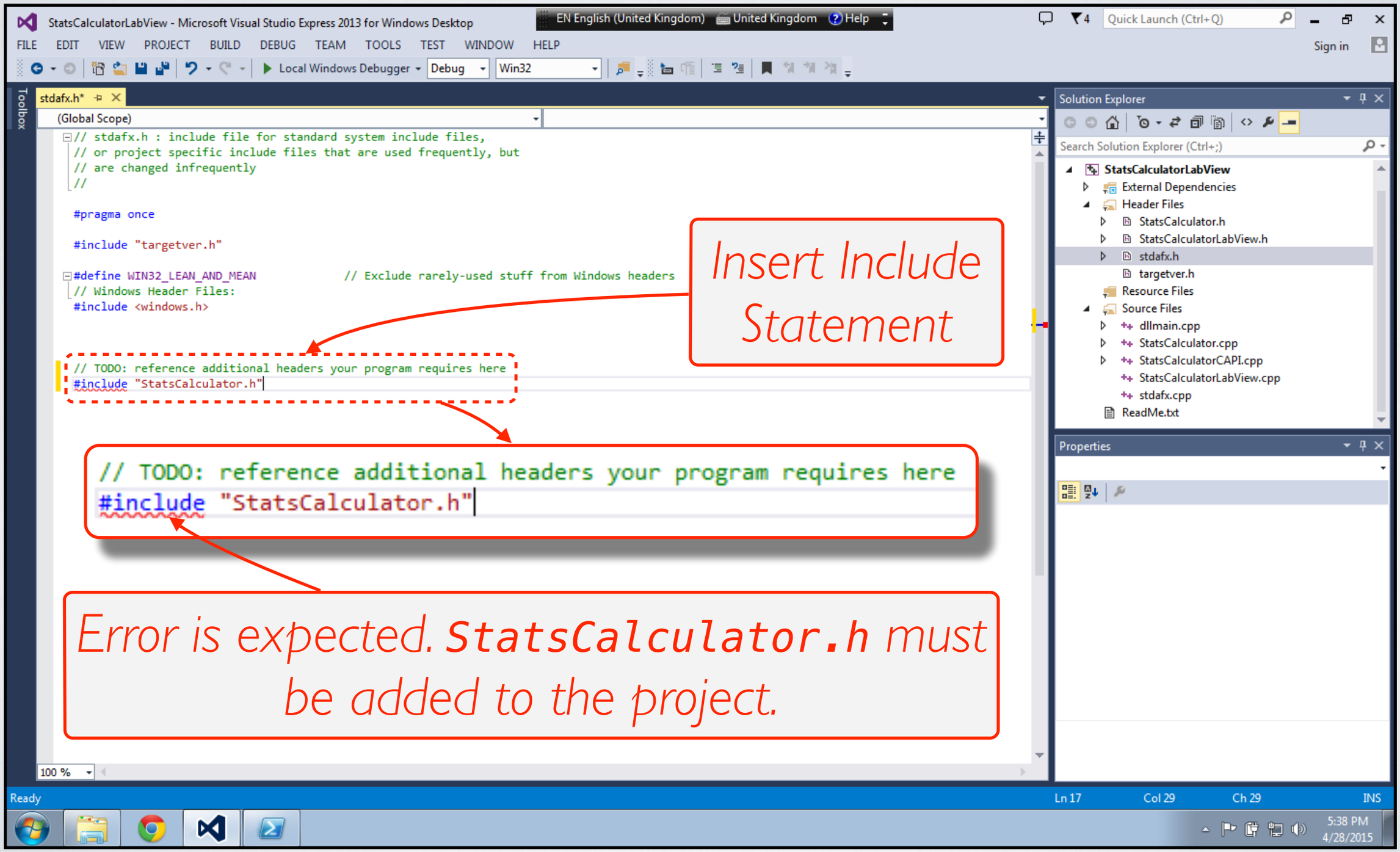
MODIFY BOILERPLATE CODE

(*StatsCalculatorLabView.h*)



MODIFY BOILERPLATE CODE

(*stdafx.h*)

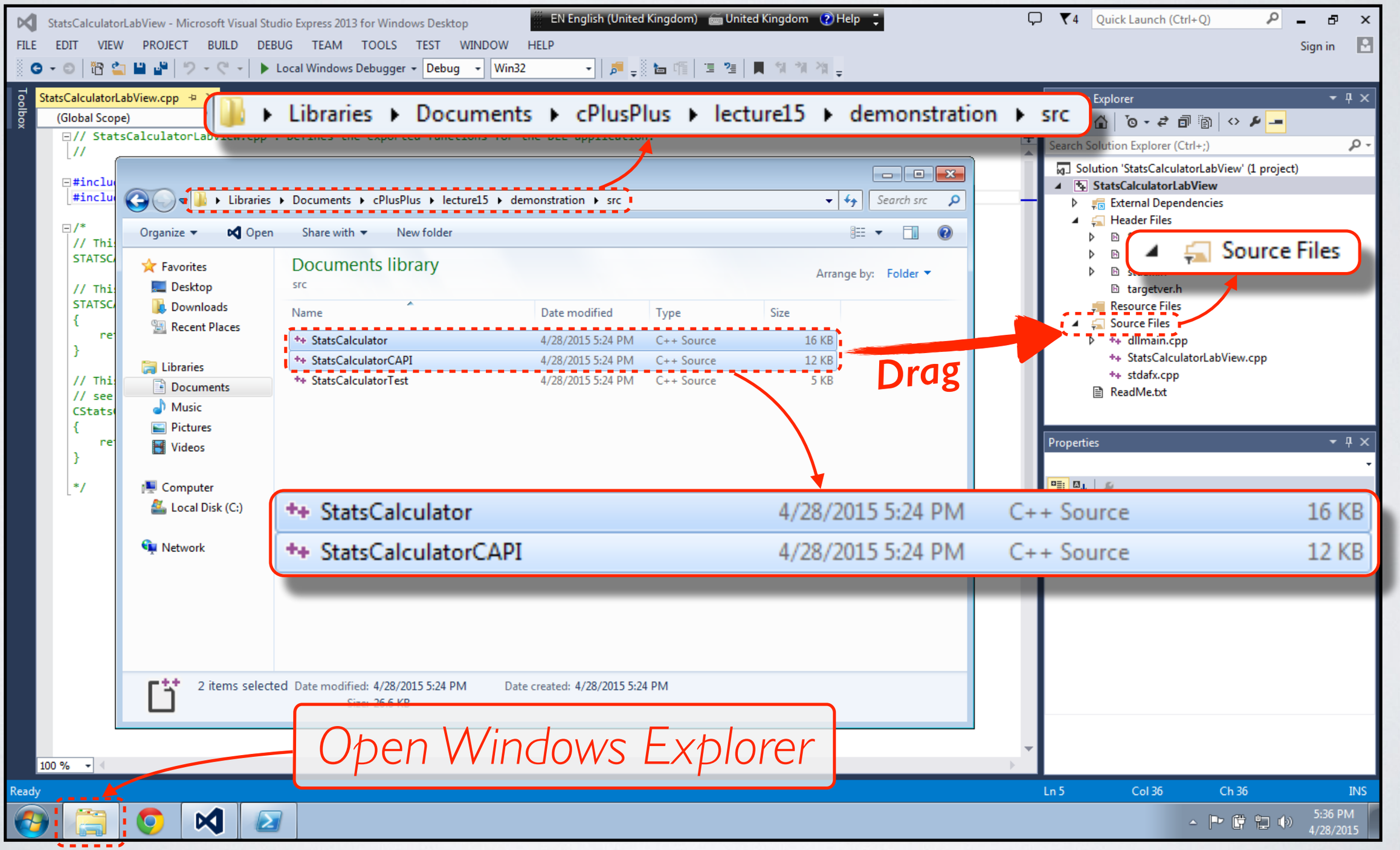


STEP 4:

ADD NEW CODE

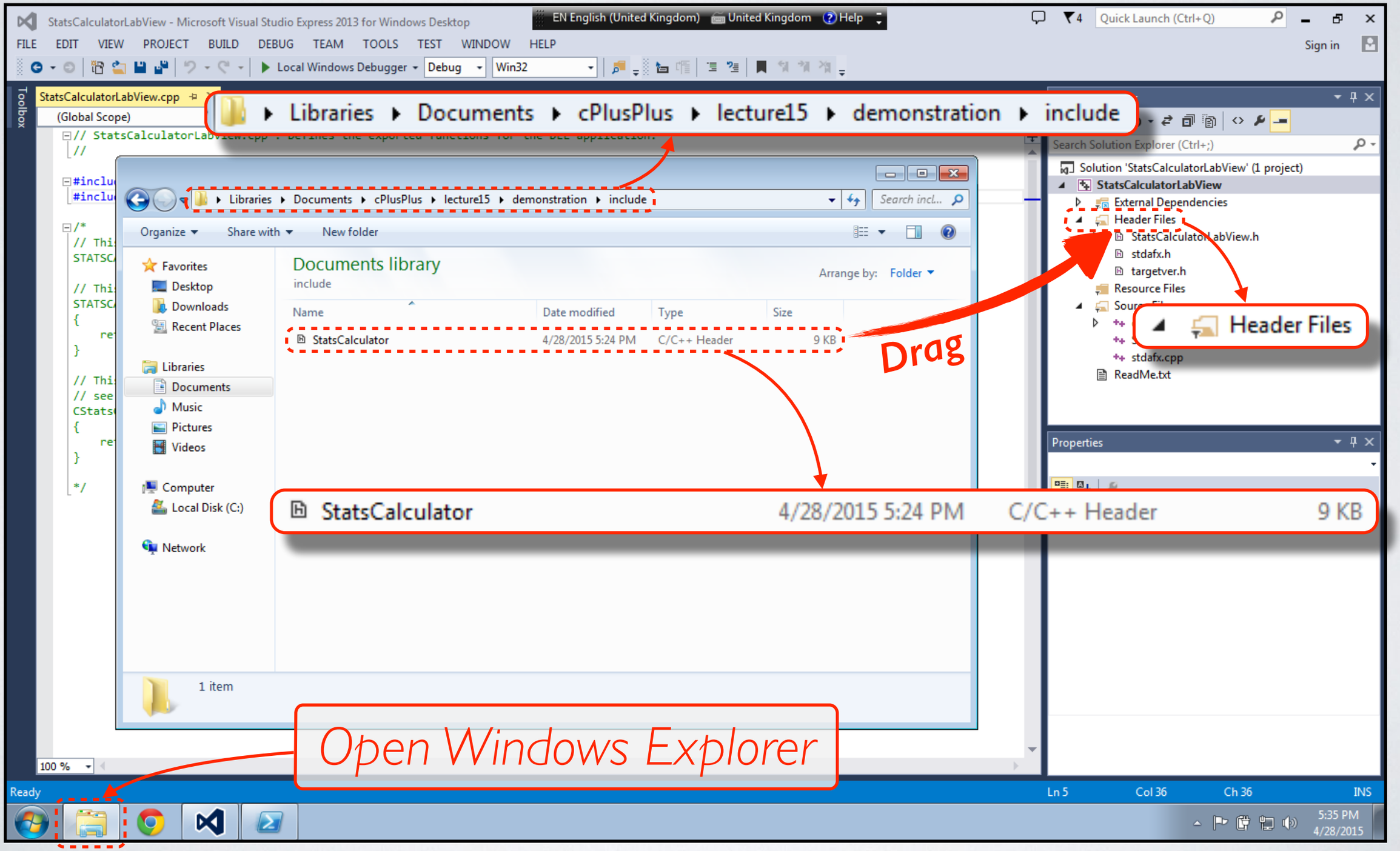
ADD NEW CODE

(*StatsCalculator.cpp*, *StatsCalculatorCAPI.cpp*)



ADD NEW CODE

(*StatsCalculator.h*)

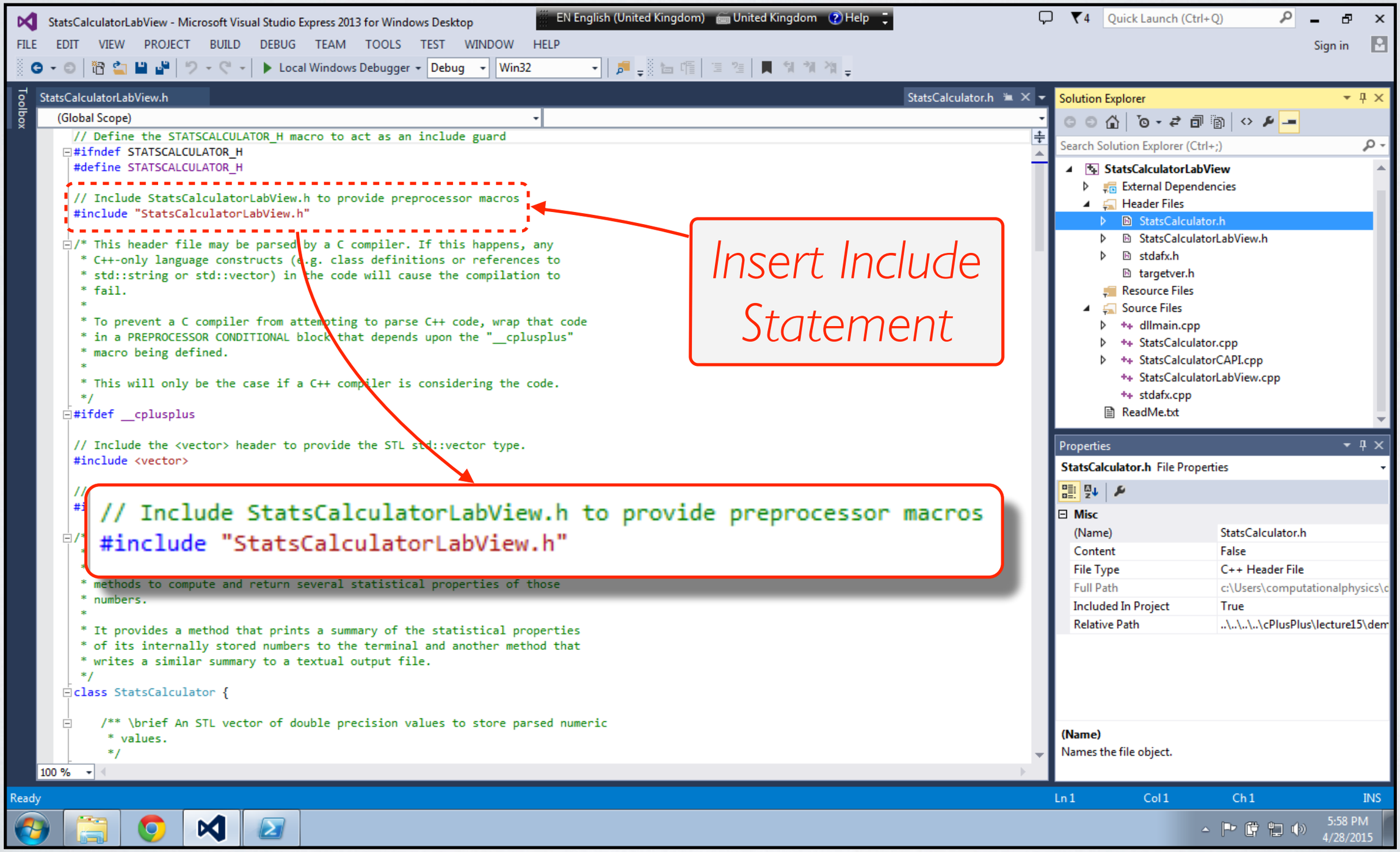


STEP 4:

MODIFY NEW CODE

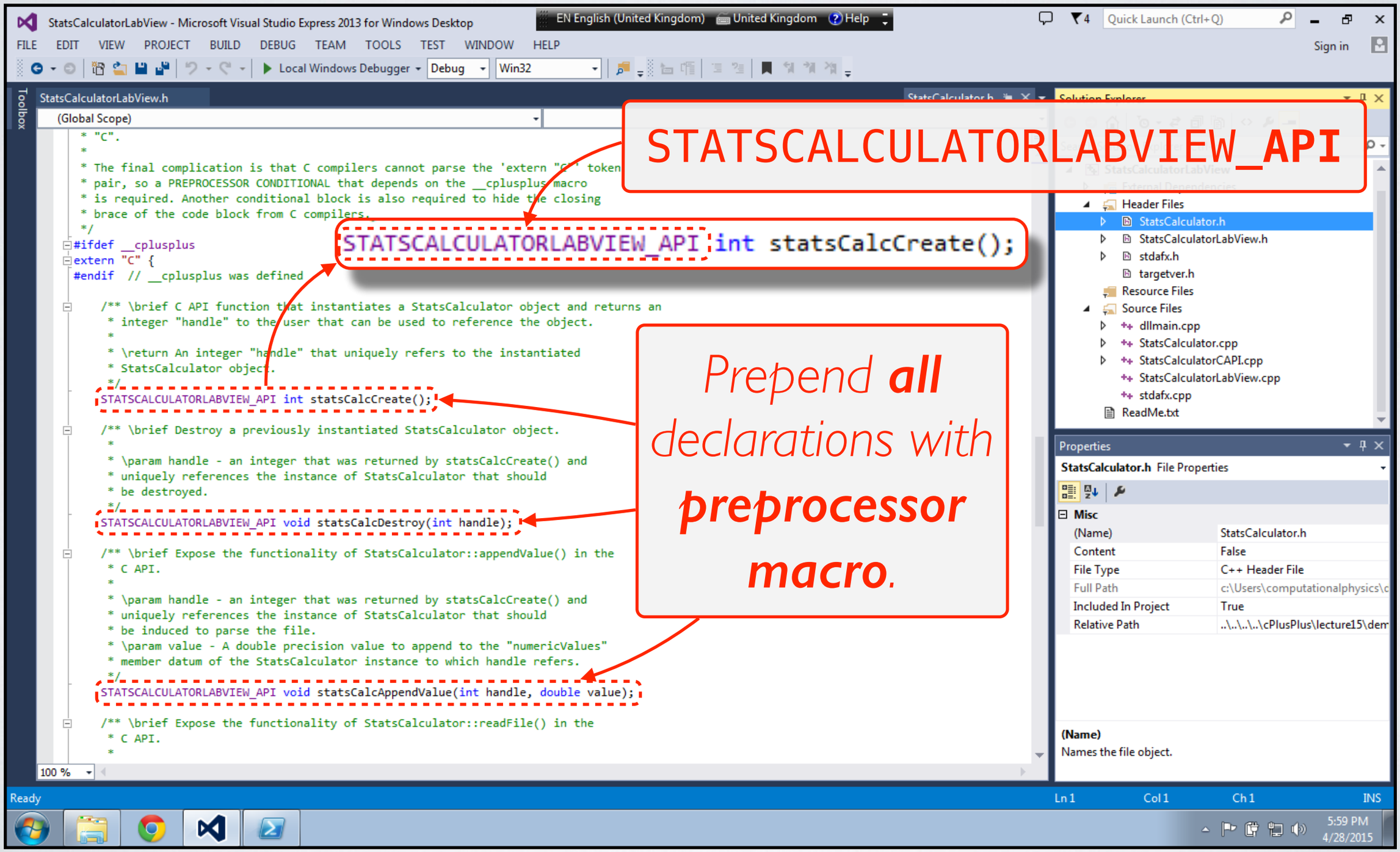
MODIFY NEW CODE

(*StatsCalculator.h*)



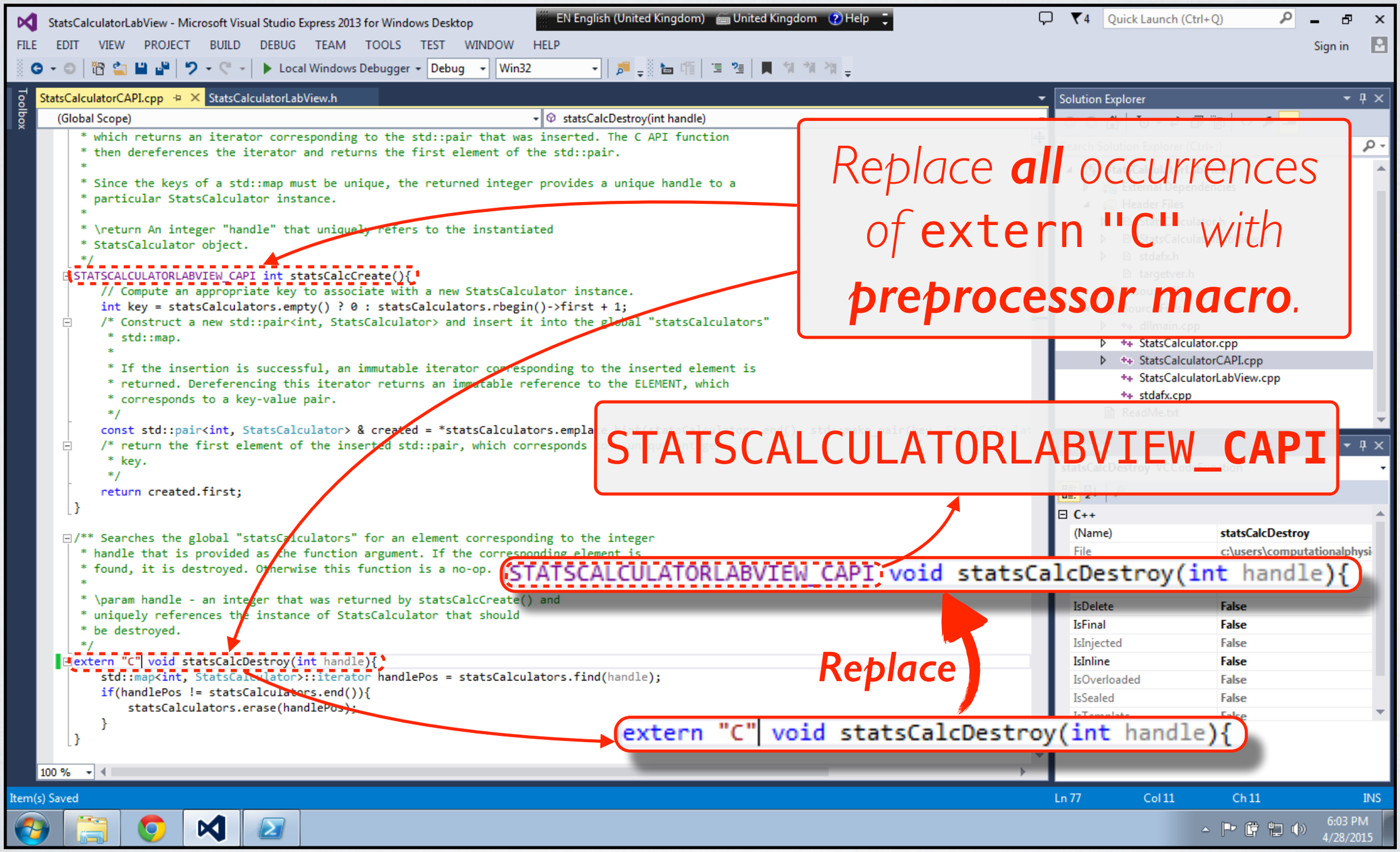
MODIFY NEW CODE

(*StatsCalculator.h*)



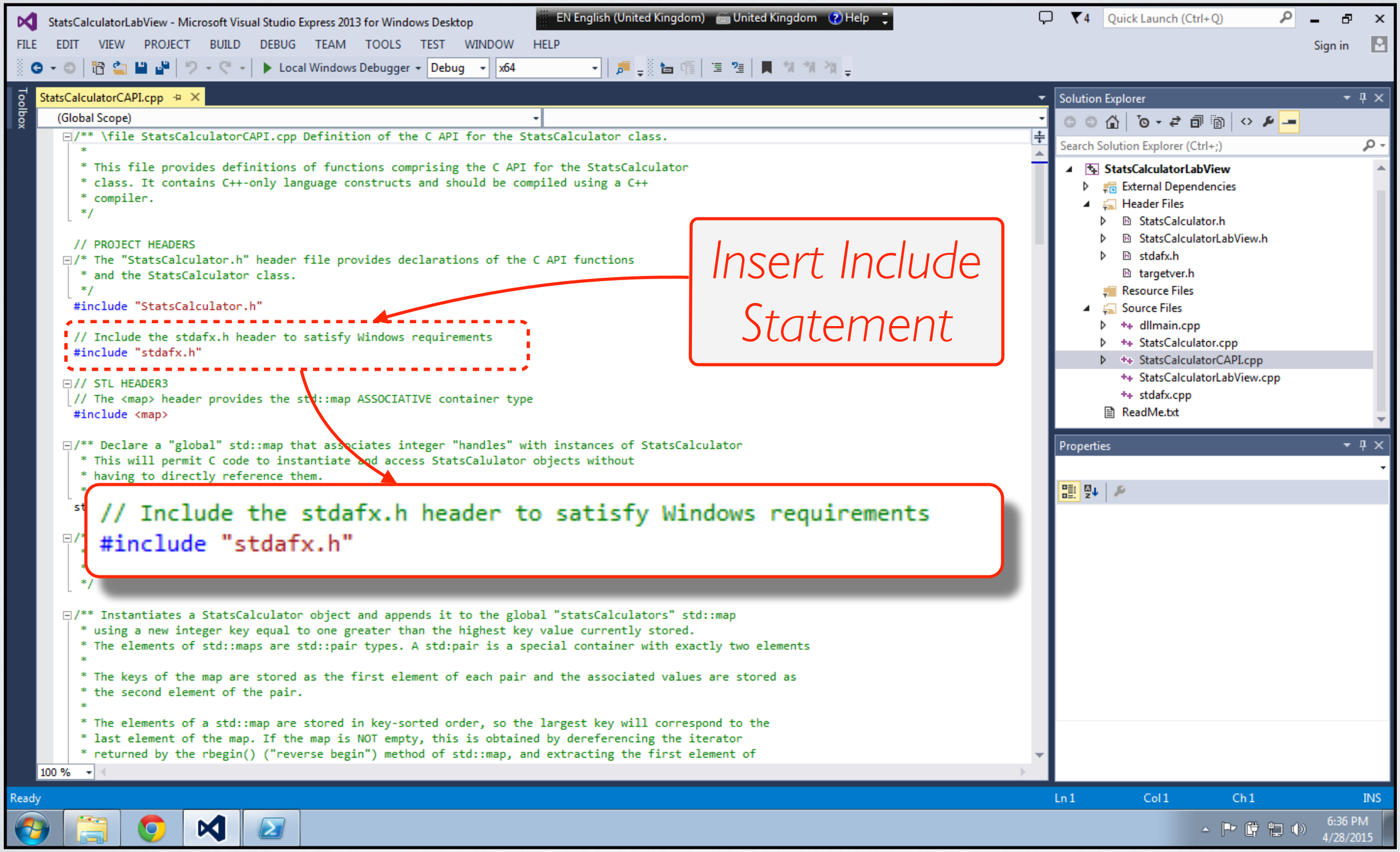
MODIFY NEW CODE

(*StatsCalculatorCAPI.cpp*)



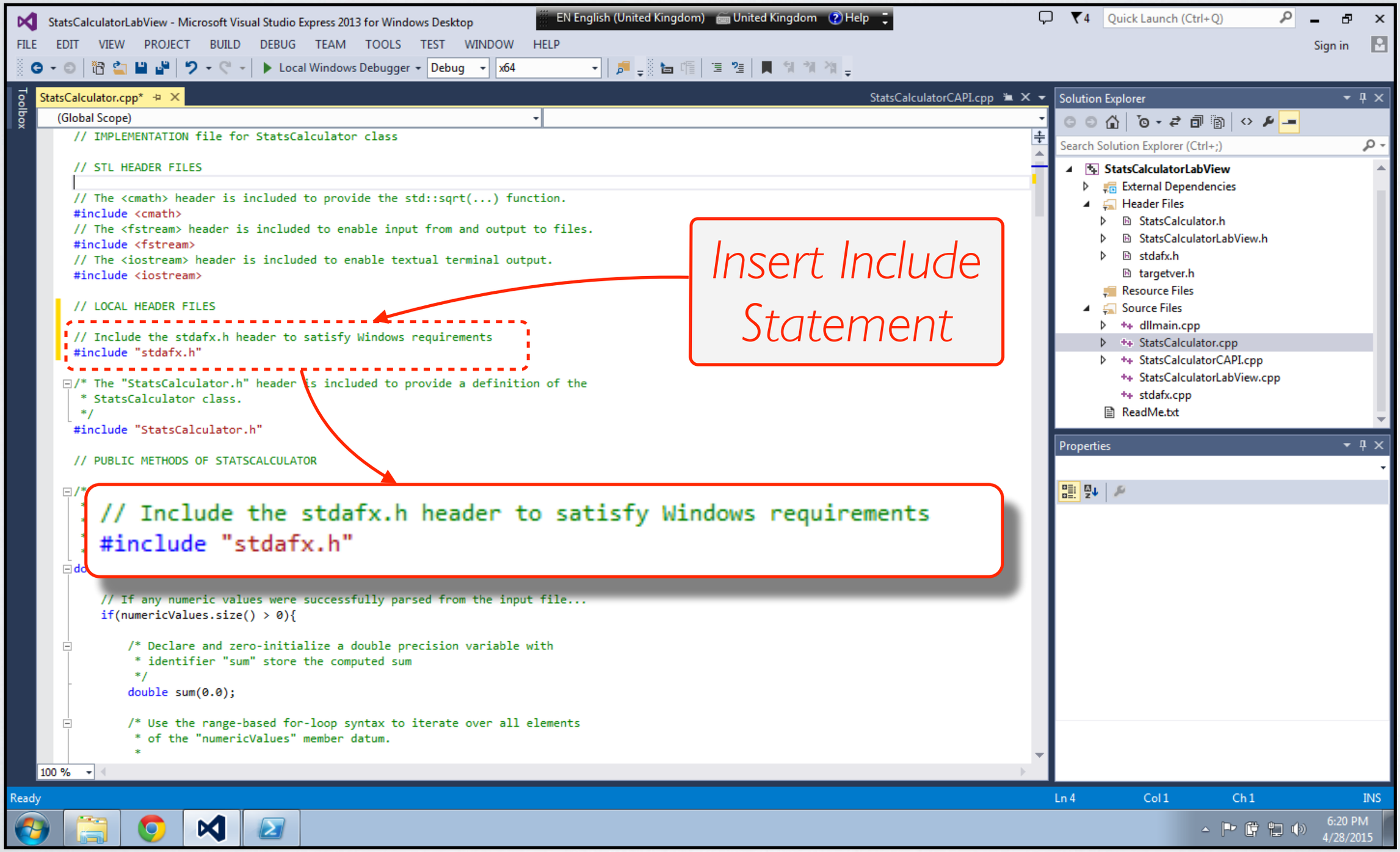
MODIFY NEW CODE

(*StatsCalculatorCAPI.cpp*)



MODIFY NEW CODE

(*StatsCalculator.cpp*)

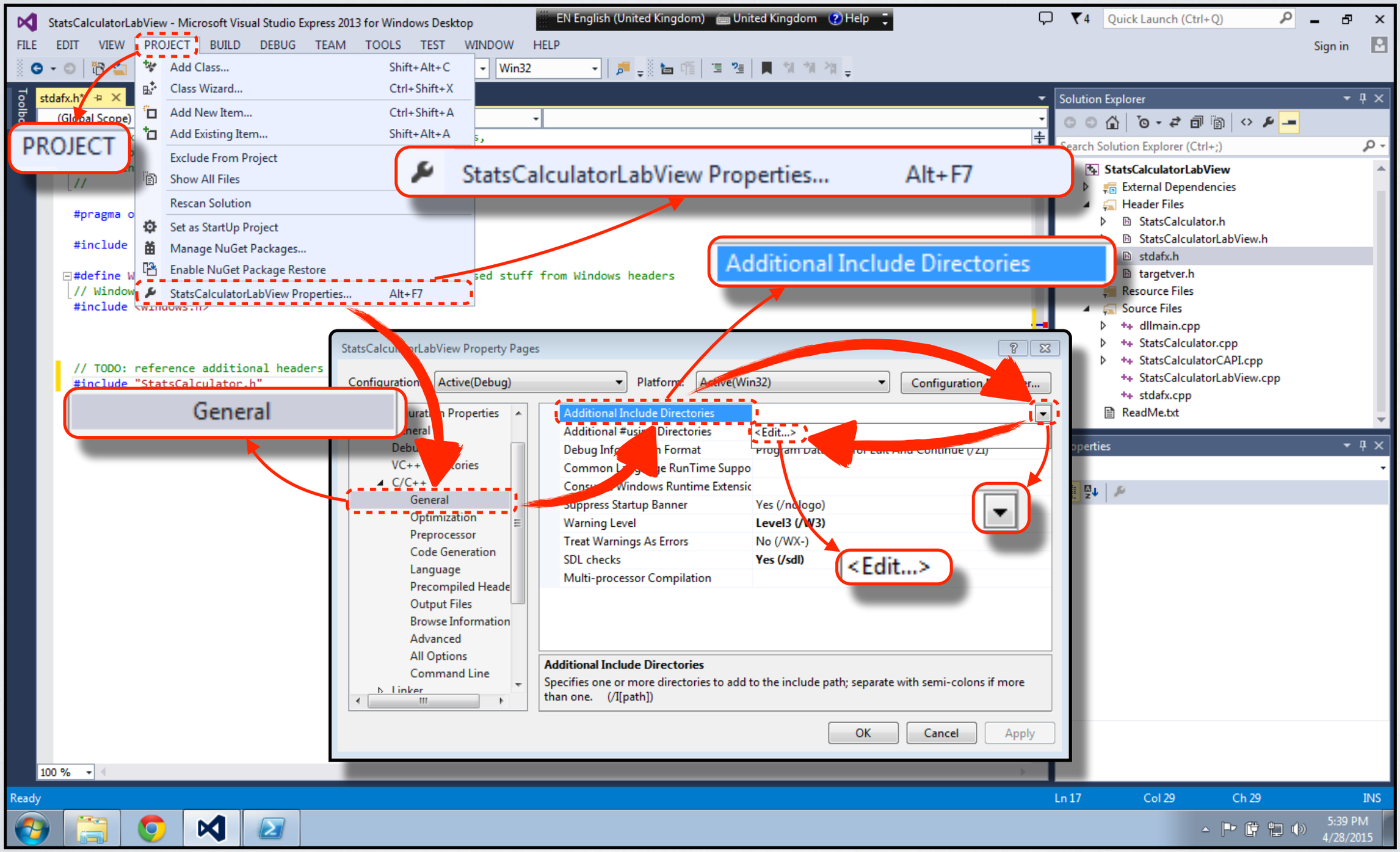


STEP 4:

MODIFY PROJECT SETTINGS

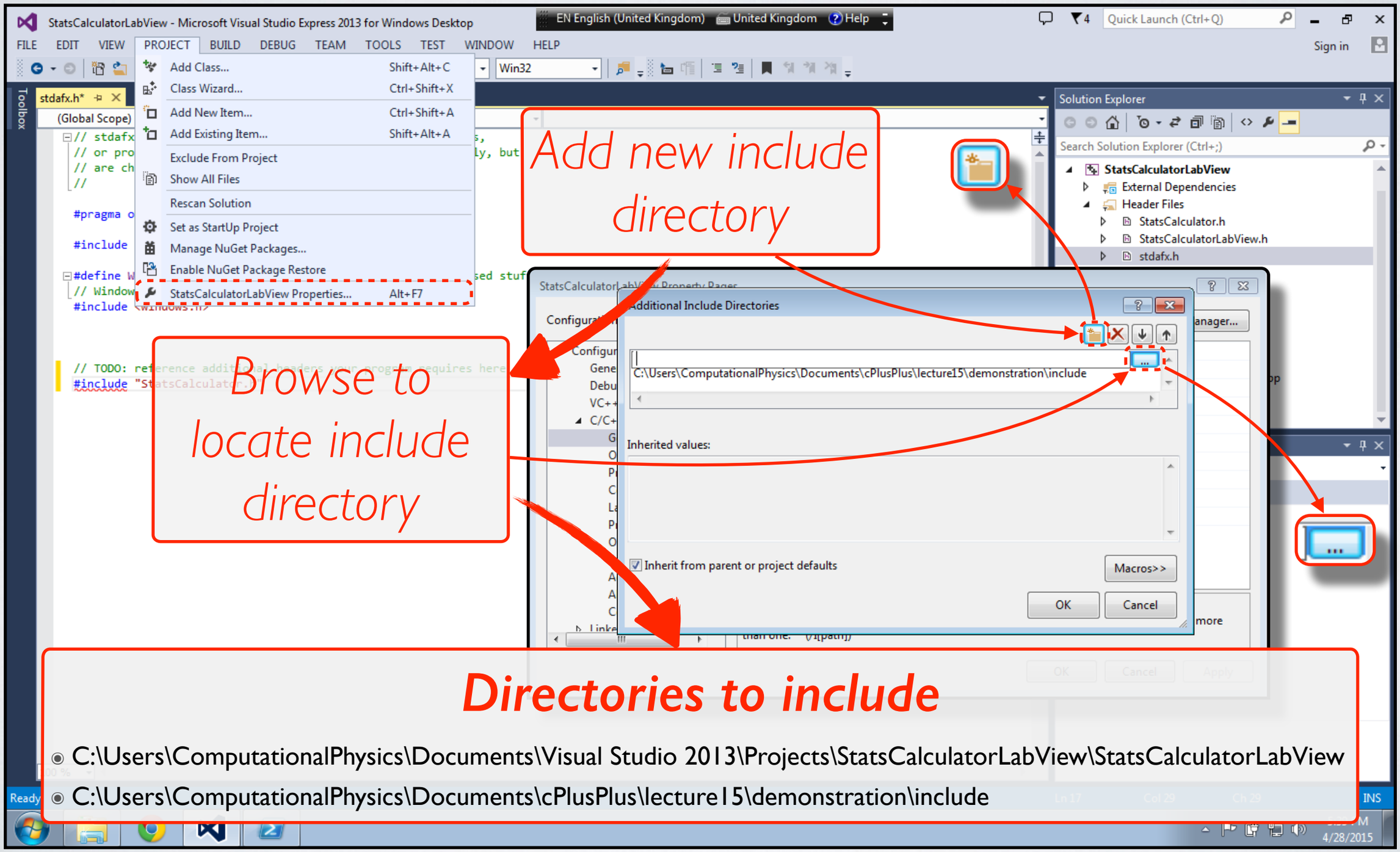
MODIFY PROJECT SETTINGS

ADD INCLUDE DIRECTORIES



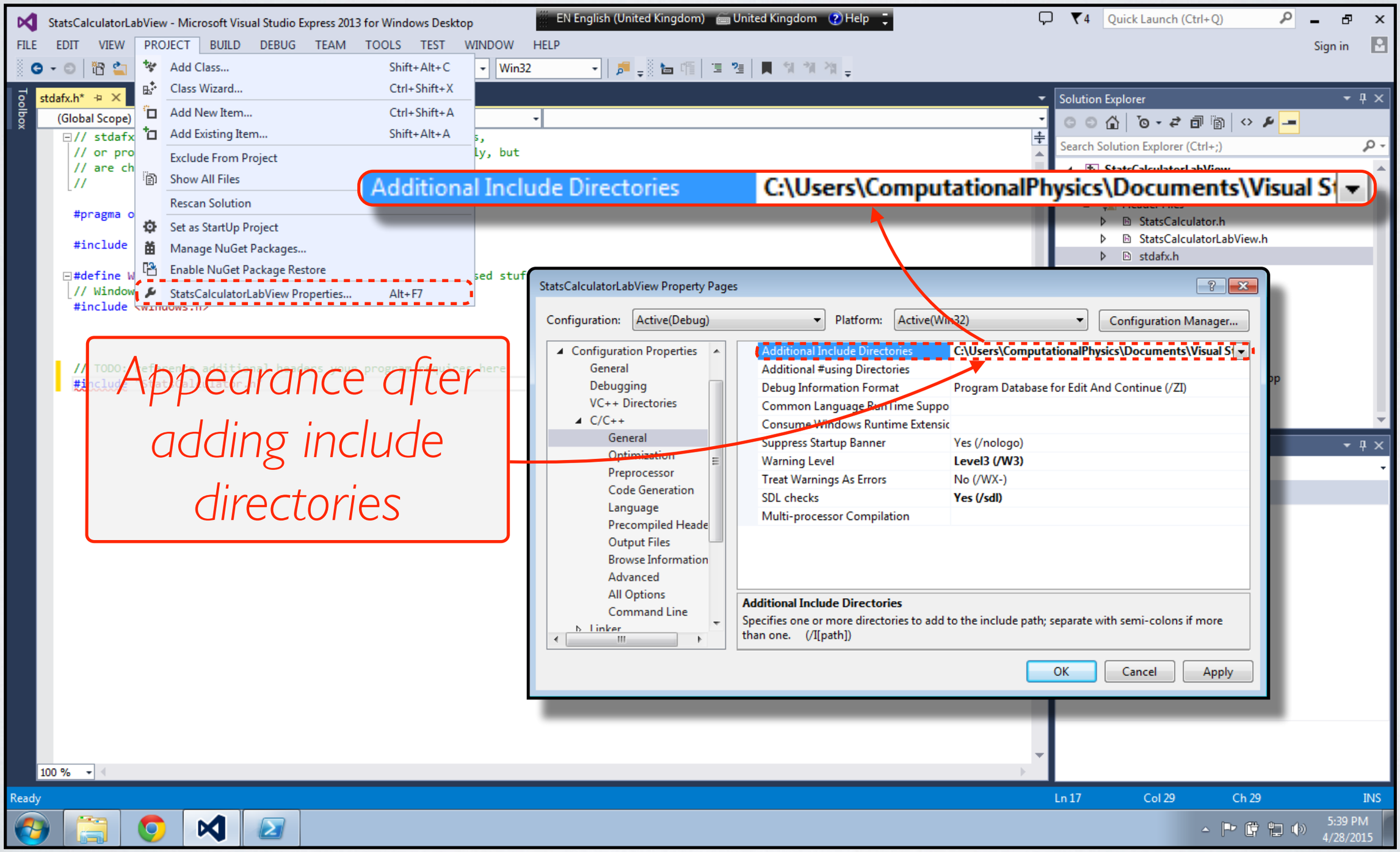
MODIFY PROJECT SETTINGS

ADD INCLUDE DIRECTORIES



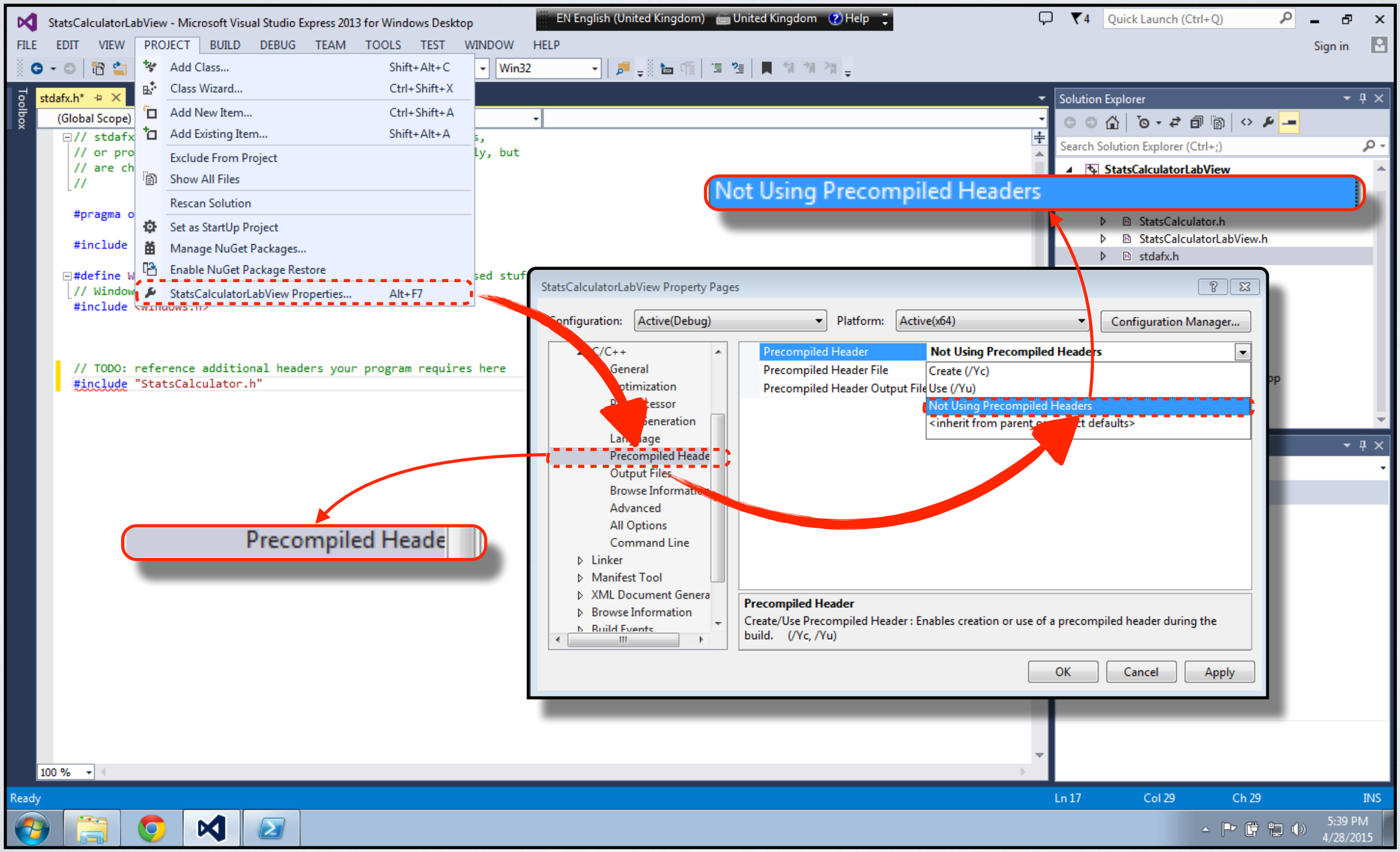
MODIFY PROJECT SETTINGS

ADD INCLUDE DIRECTORIES



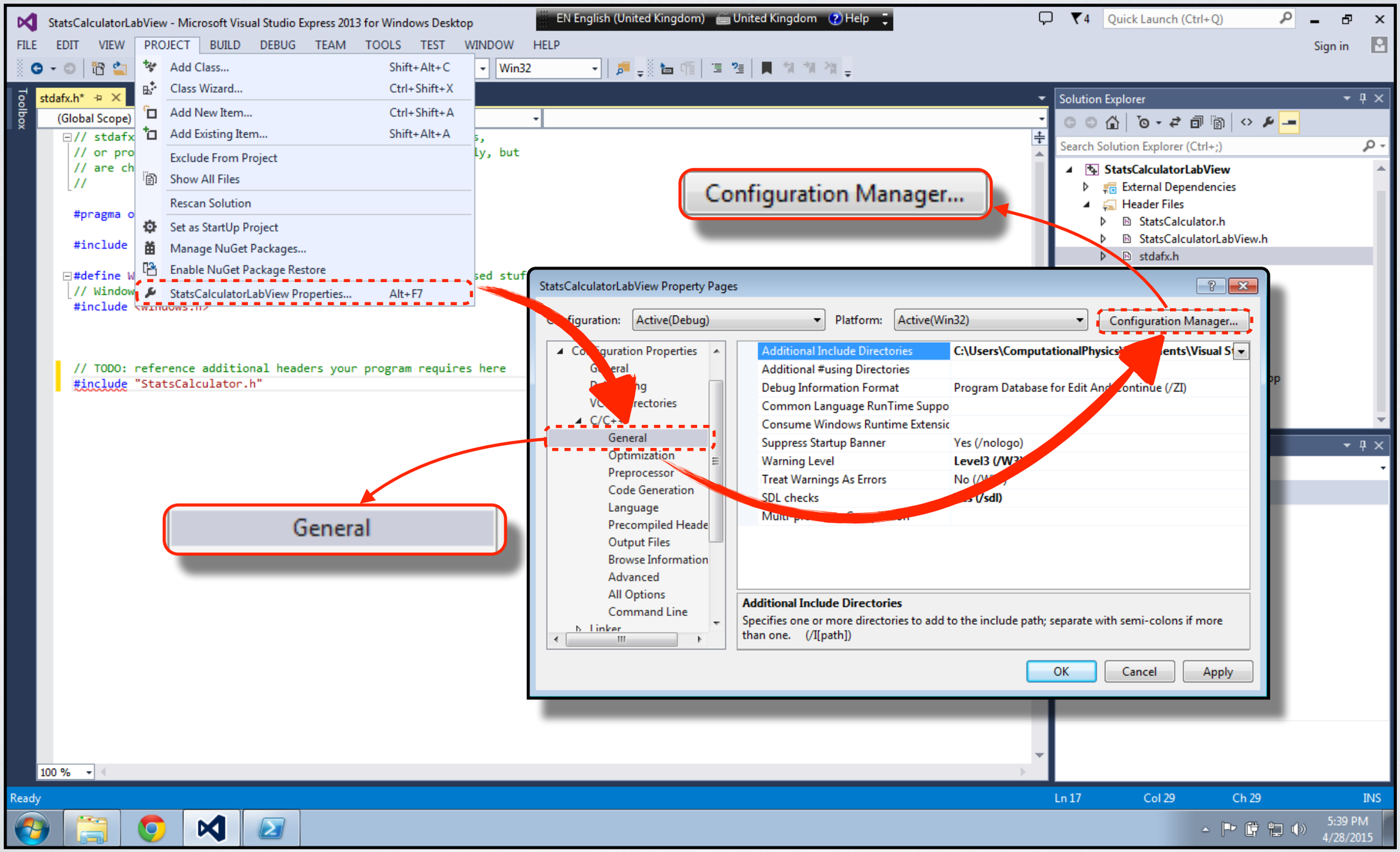
MODIFY PROJECT SETTINGS

DISABLE PRECOMPILED HEADER USAGE



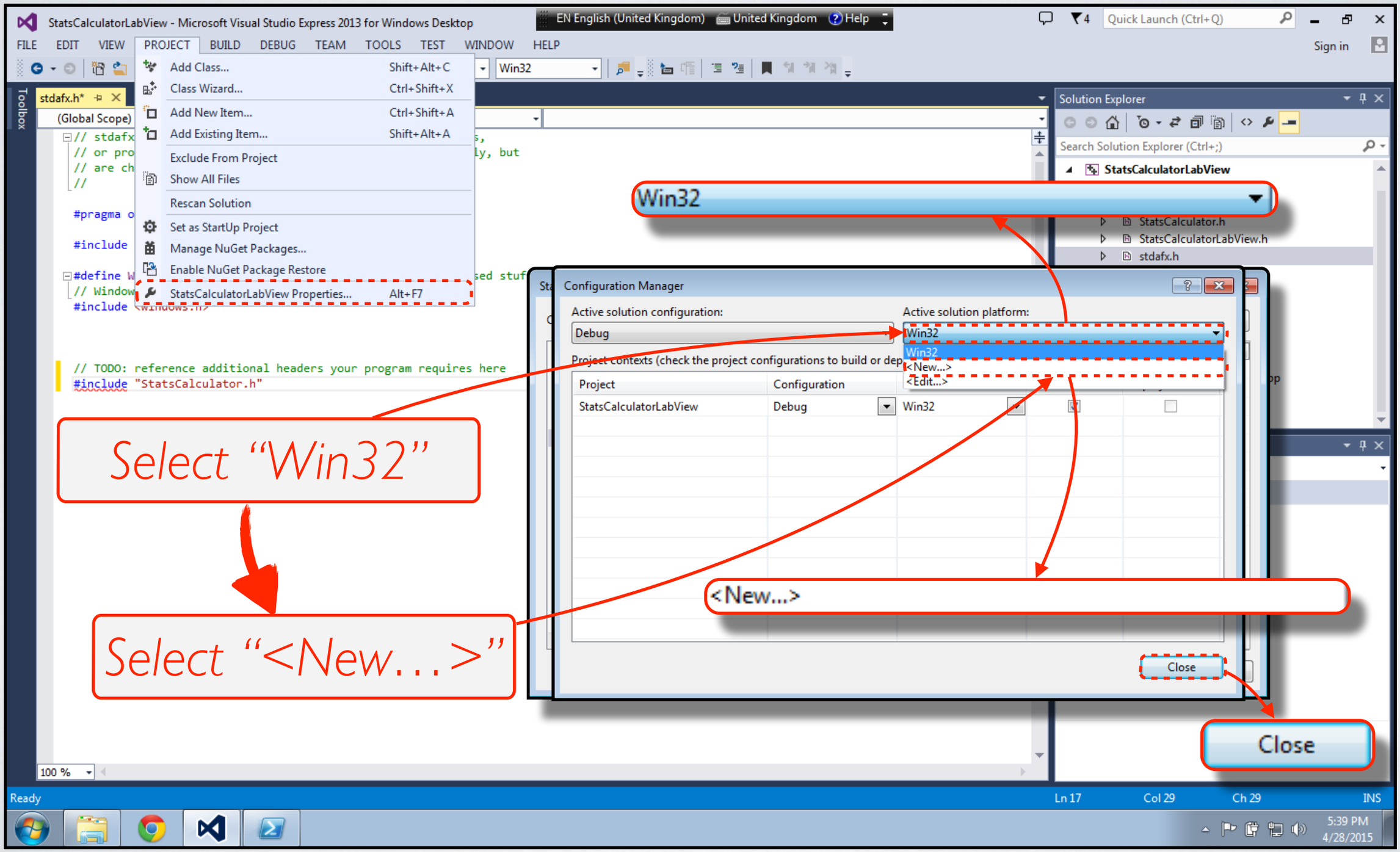
MODIFY PROJECT SETTINGS

ENABLE 64-BIT CODE GENERATION



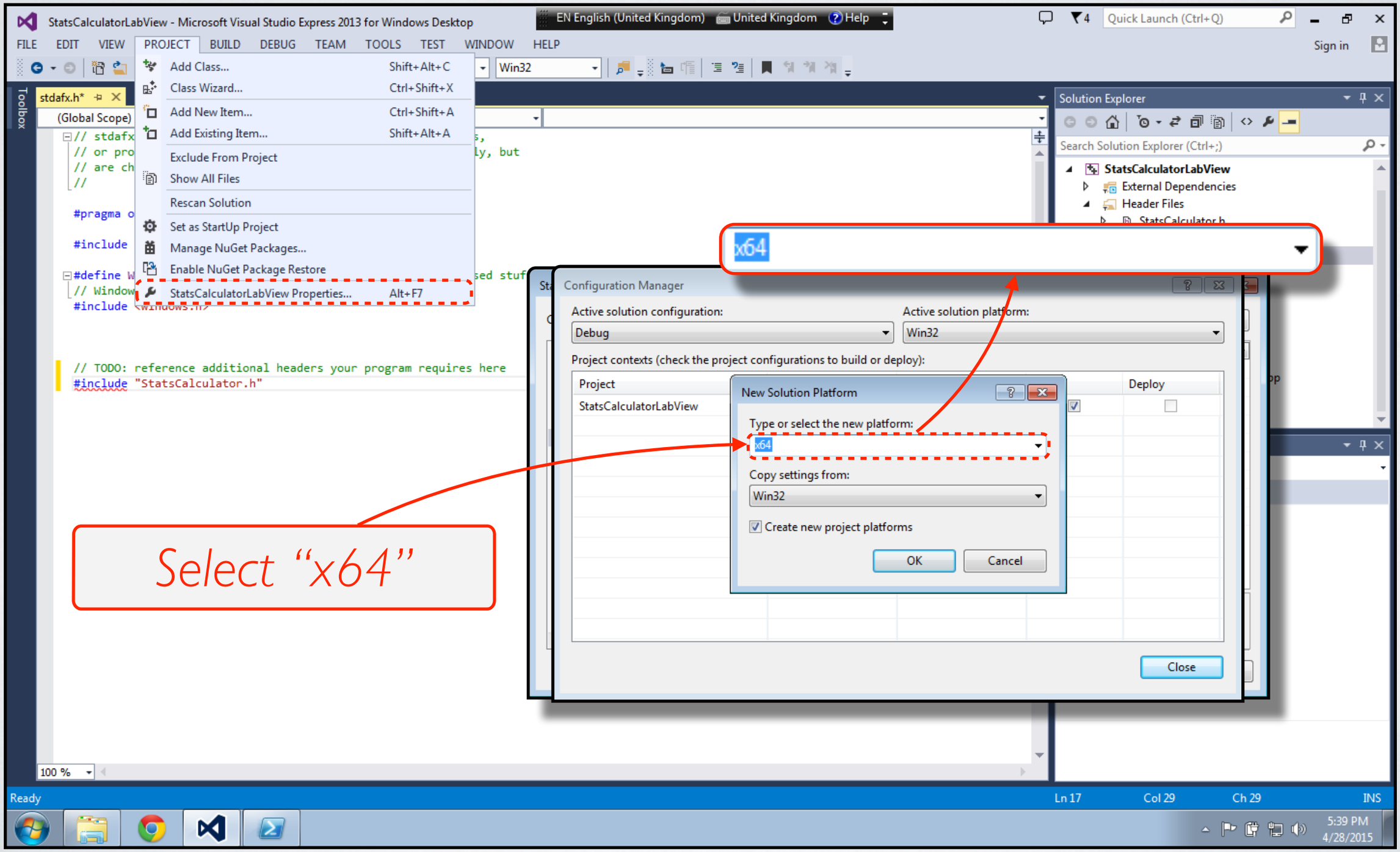
MODIFY PROJECT SETTINGS

ENABLE 64-BIT CODE GENERATION



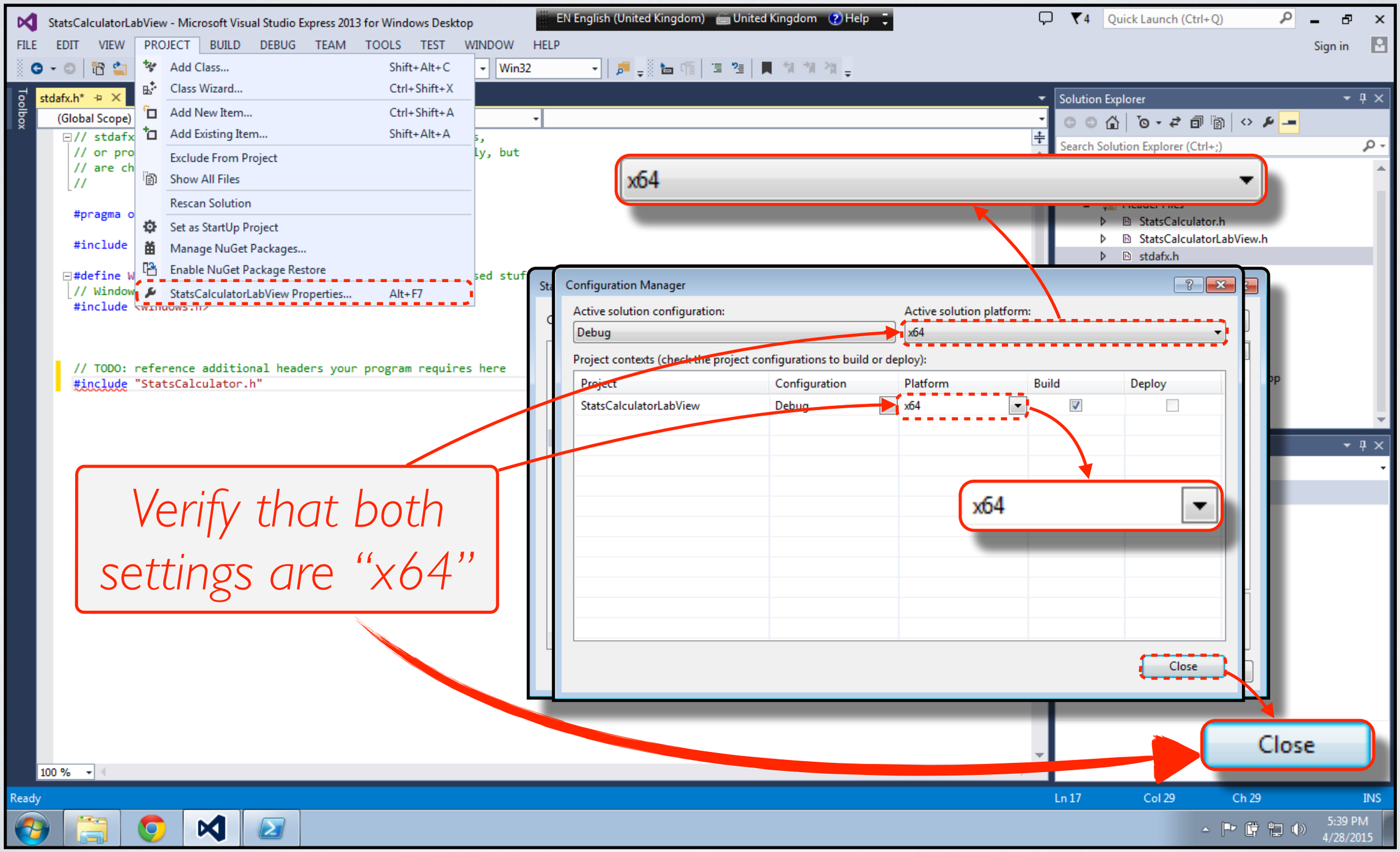
MODIFY PROJECT SETTINGS

ENABLE 64-BIT CODE GENERATION



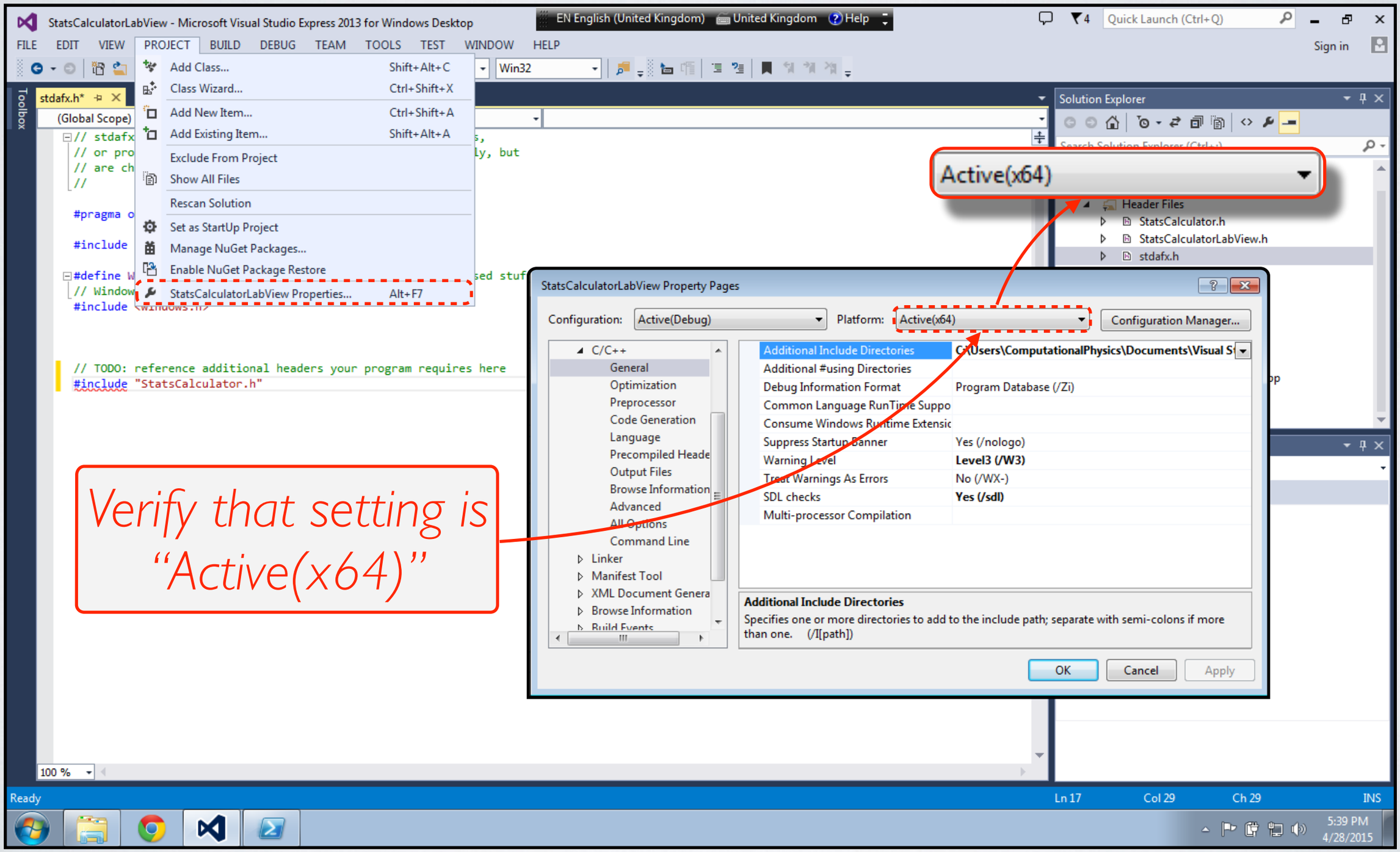
MODIFY PROJECT SETTINGS

ENABLE 64-BIT CODE GENERATION



MODIFY PROJECT SETTINGS

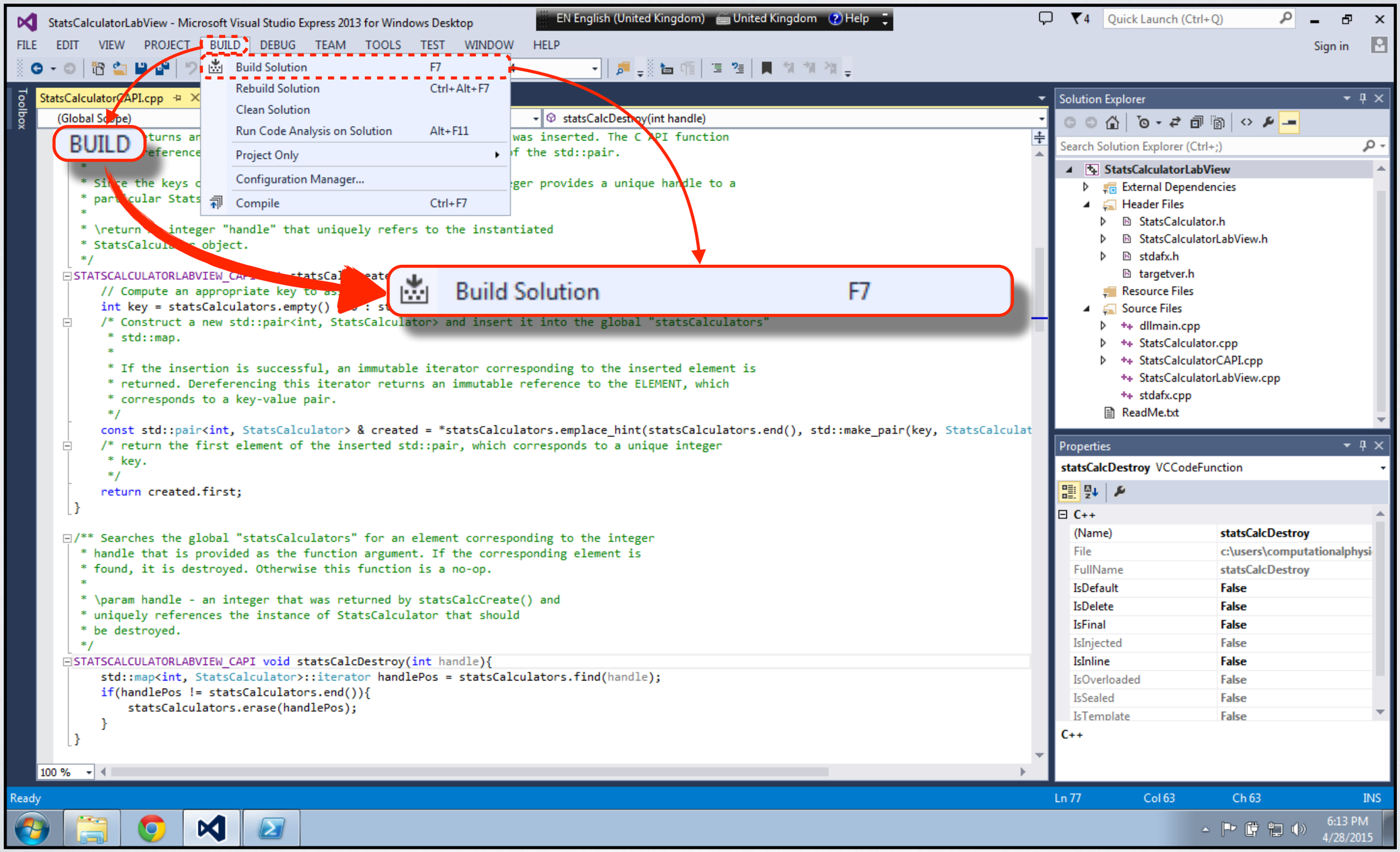
ENABLE 64-BIT CODE GENERATION



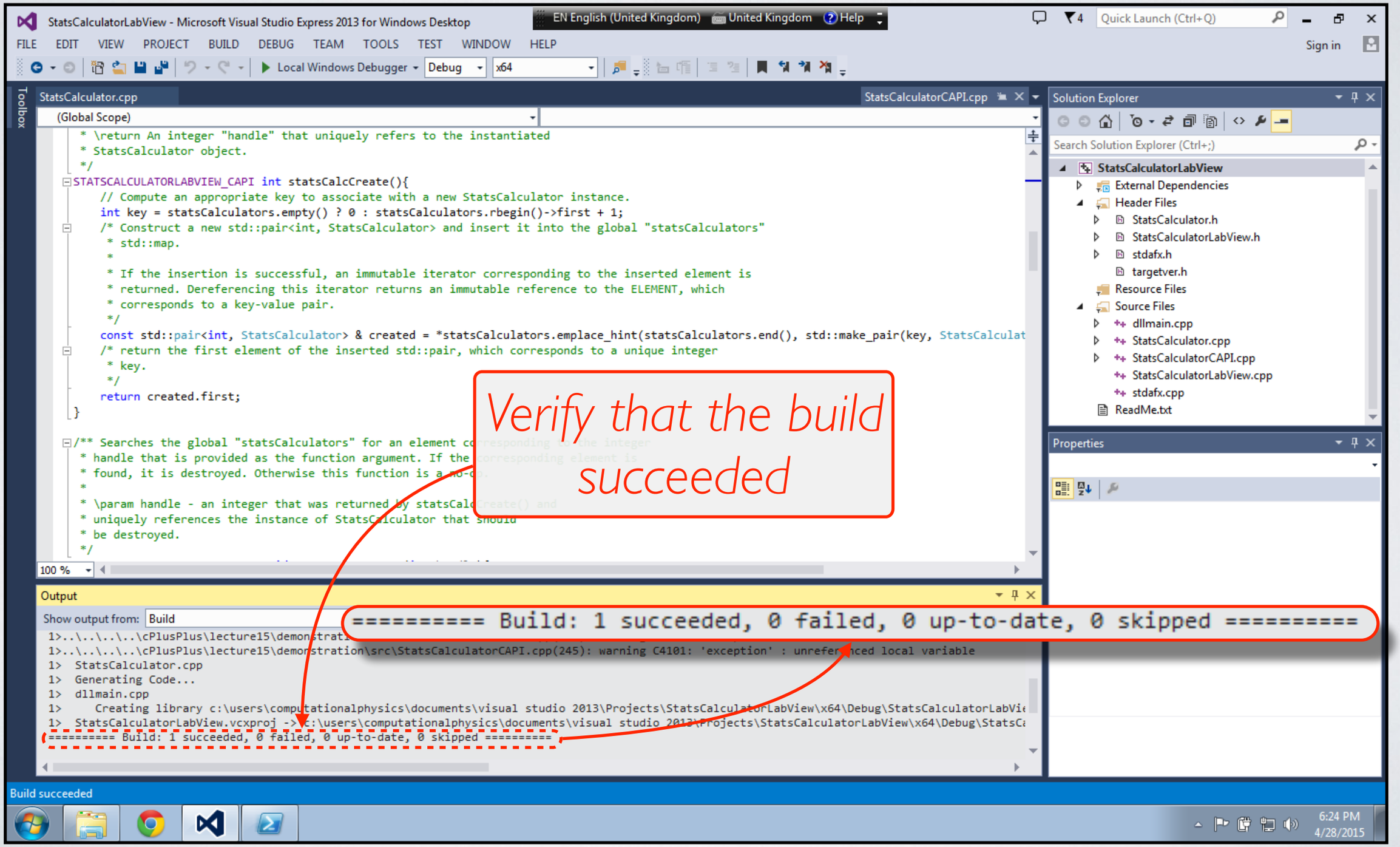
STEP 5:

BUILD PROJECT

BUILD PROJECT



BUILD PROJECT

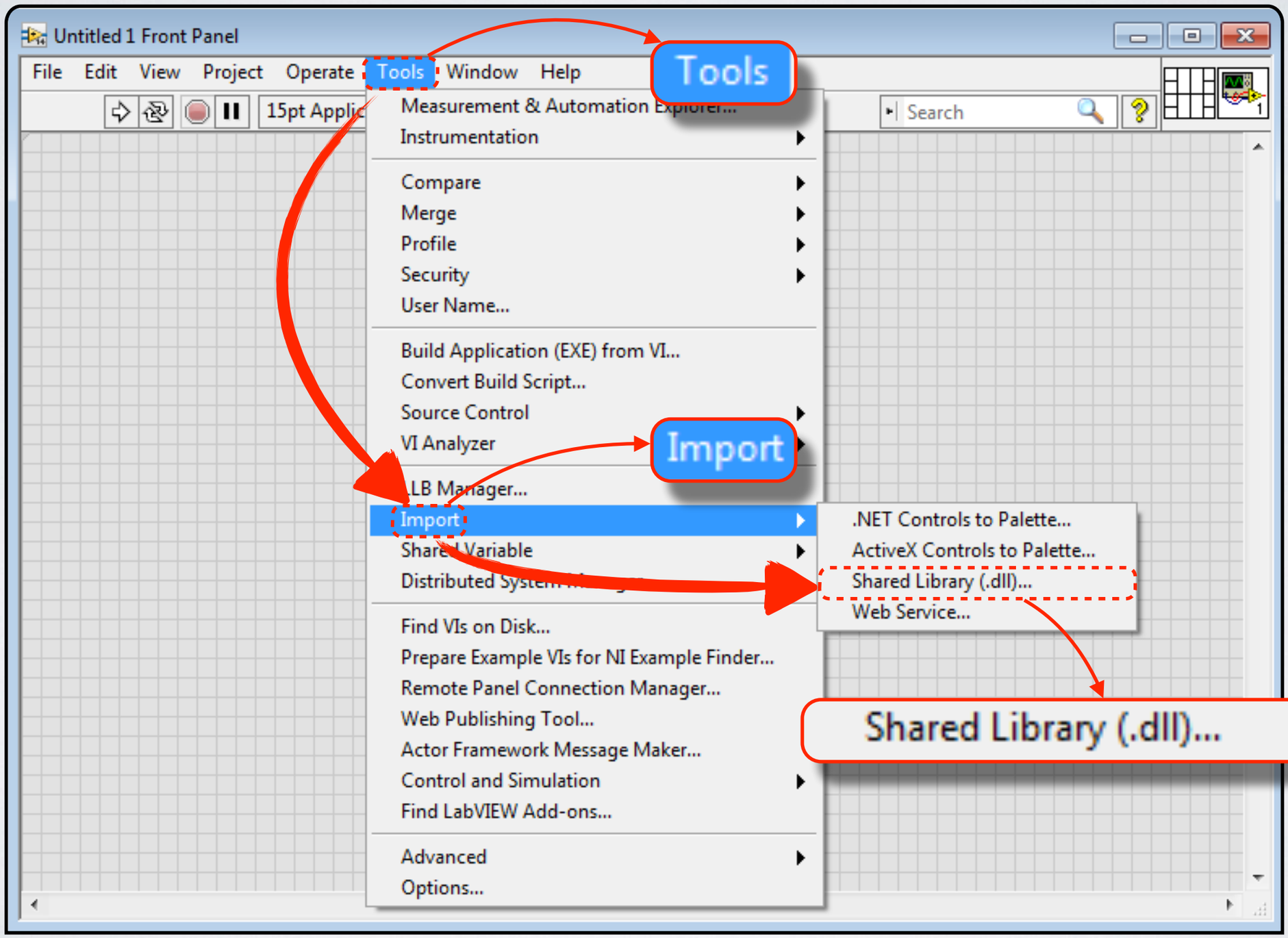


STEP 6:

IMPORT THE LIBRARY

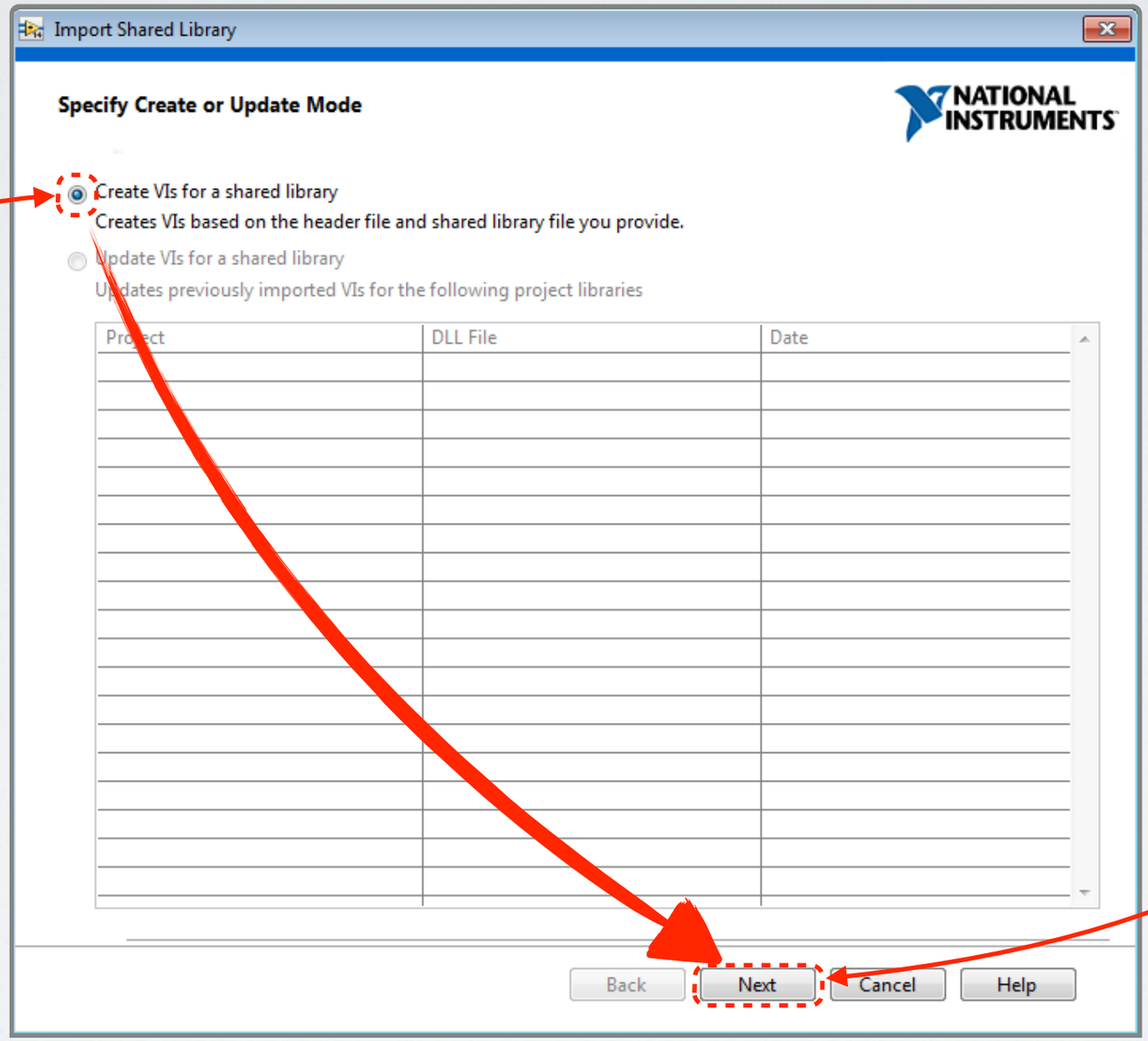
IMPORT SHARED LIBRARY

USING THE IMPORT WIZARD TOOL



IMPORT SHARED LIBRARY

SPECIFY CREATE OR UPDATE MODE

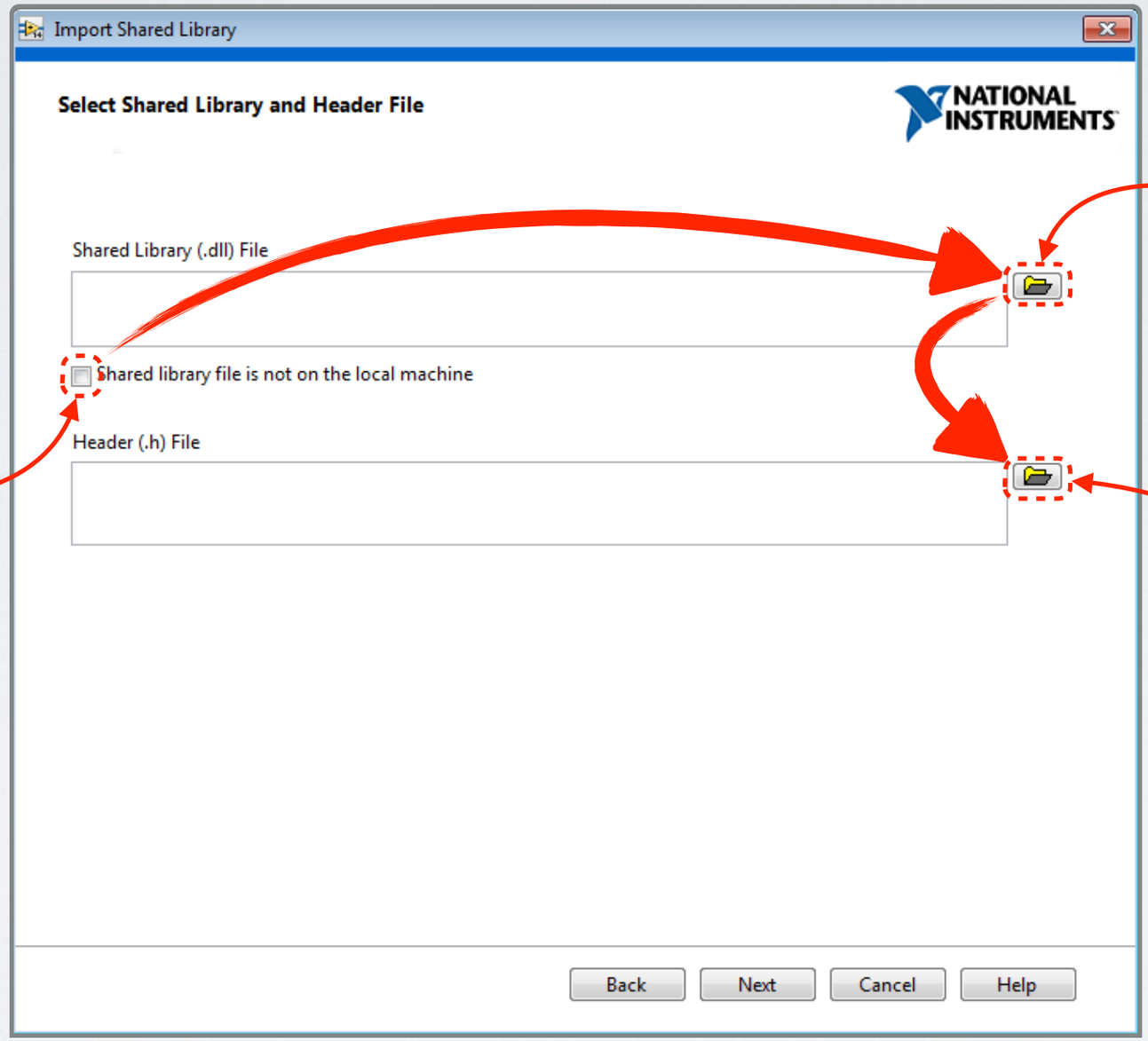


Select "Create VIs for a shared library"

Select "Next"

IMPORT SHARED LIBRARY

SELECT SHARED LIBRARY AND HEADER FILE



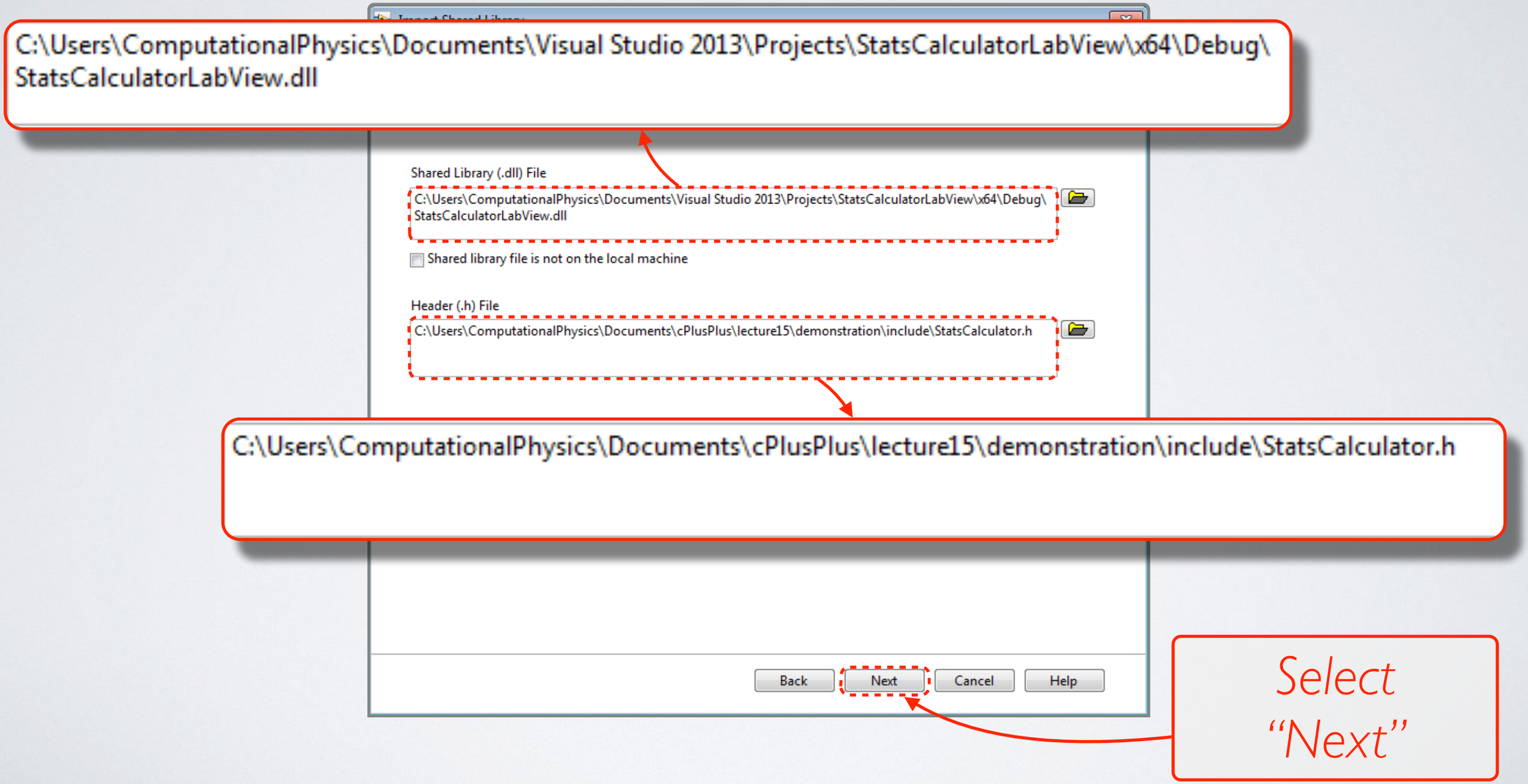
Verify that checkbox is ***not*** *selected.*

Browse for shared library file.

Browse for header file that defines the exported symbols.

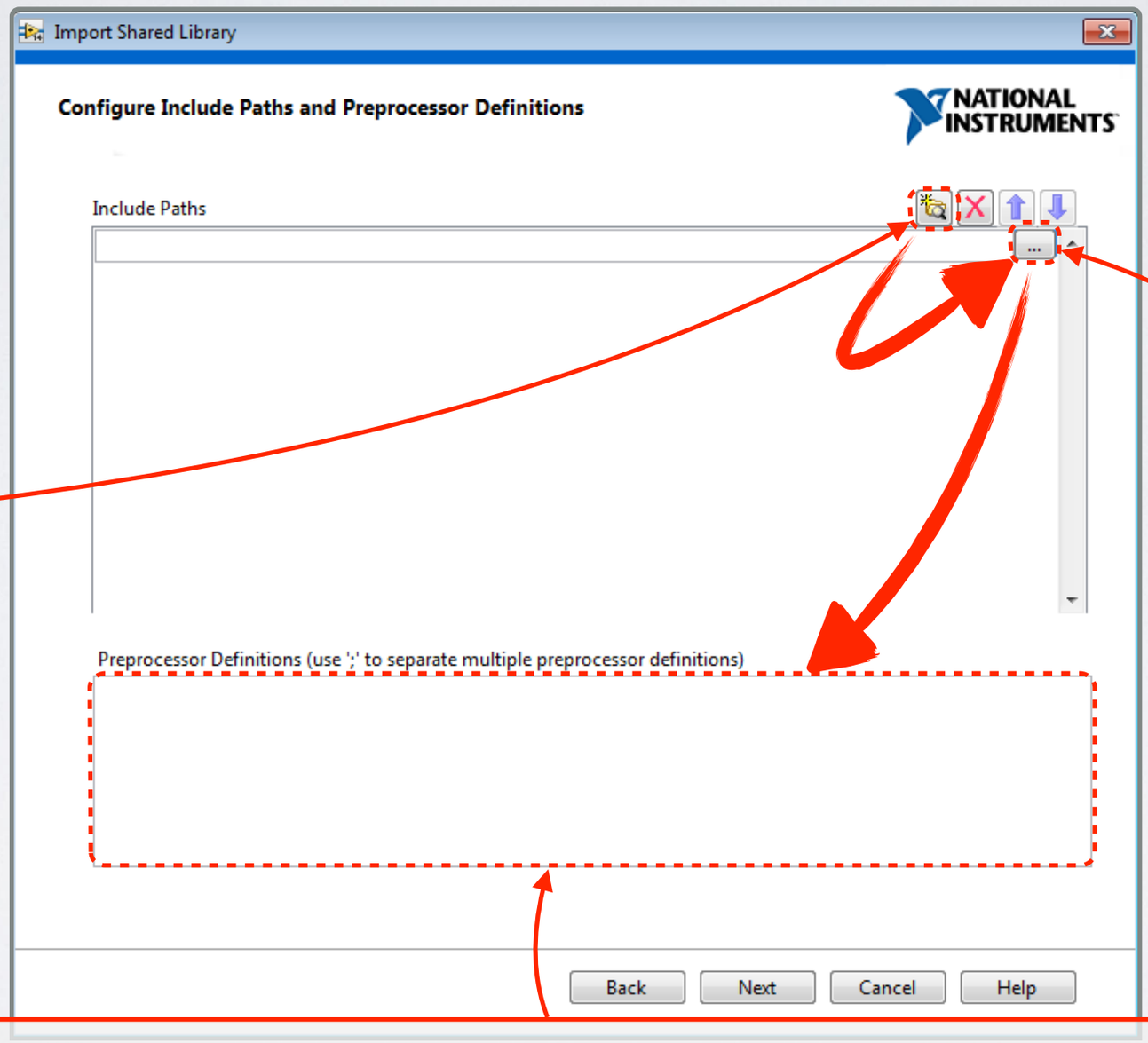
IMPORT SHARED LIBRARY

SELECT SHARED LIBRARY AND HEADER FILE



IMPORT SHARED LIBRARY

CONFIGURE INCLUDE PATHS AND PREPROCESSOR DEFINITIONS



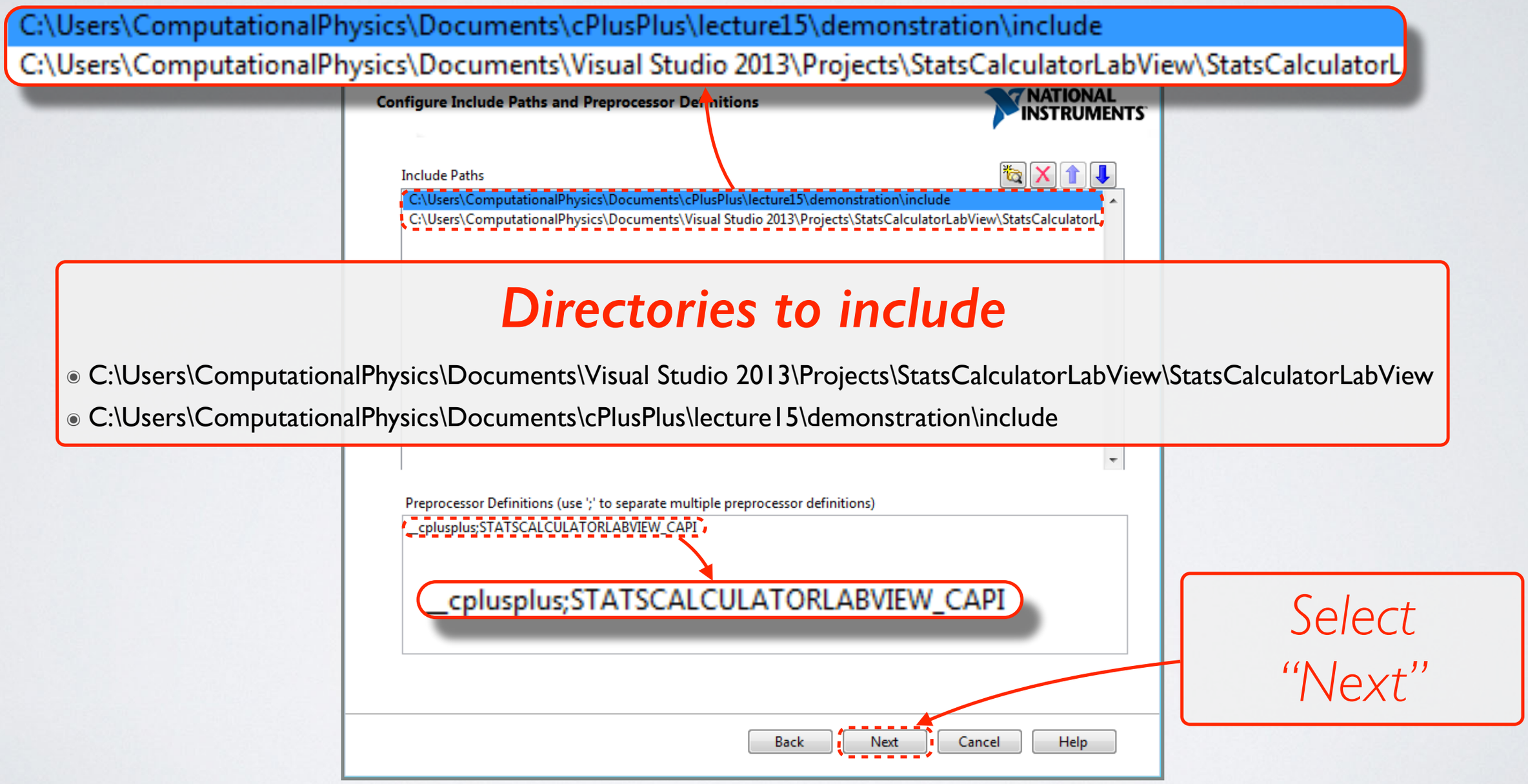
Add new Include Path.

Browse for Include Paths.

Add preprocessor definitions that make functions visible.

IMPORT SHARED LIBRARY

CONFIGURE INCLUDE PATHS AND PREPROCESSOR DEFINITIONS



Directories to include

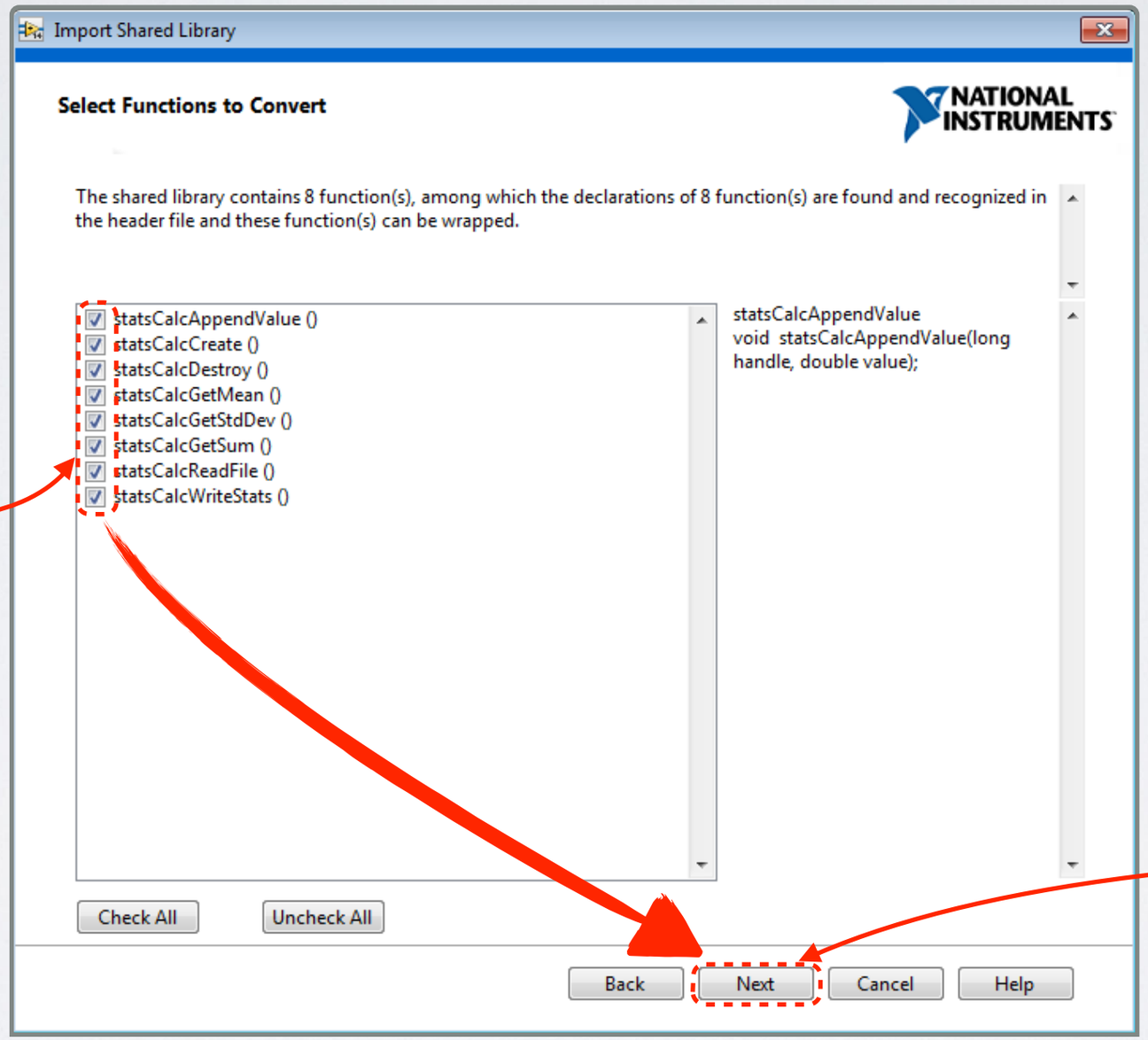
- C:\Users\ComputationalPhysics\Documents\Visual Studio 2013\Projects\StatsCalculatorLabView\StatsCalculatorLabView
- C:\Users\ComputationalPhysics\Documents\cPlusPlus\lecture15\demonstration\include

Select "Next"

IMPORT SHARED LIBRARY

SELECT FUNCTIONS TO CONVERT

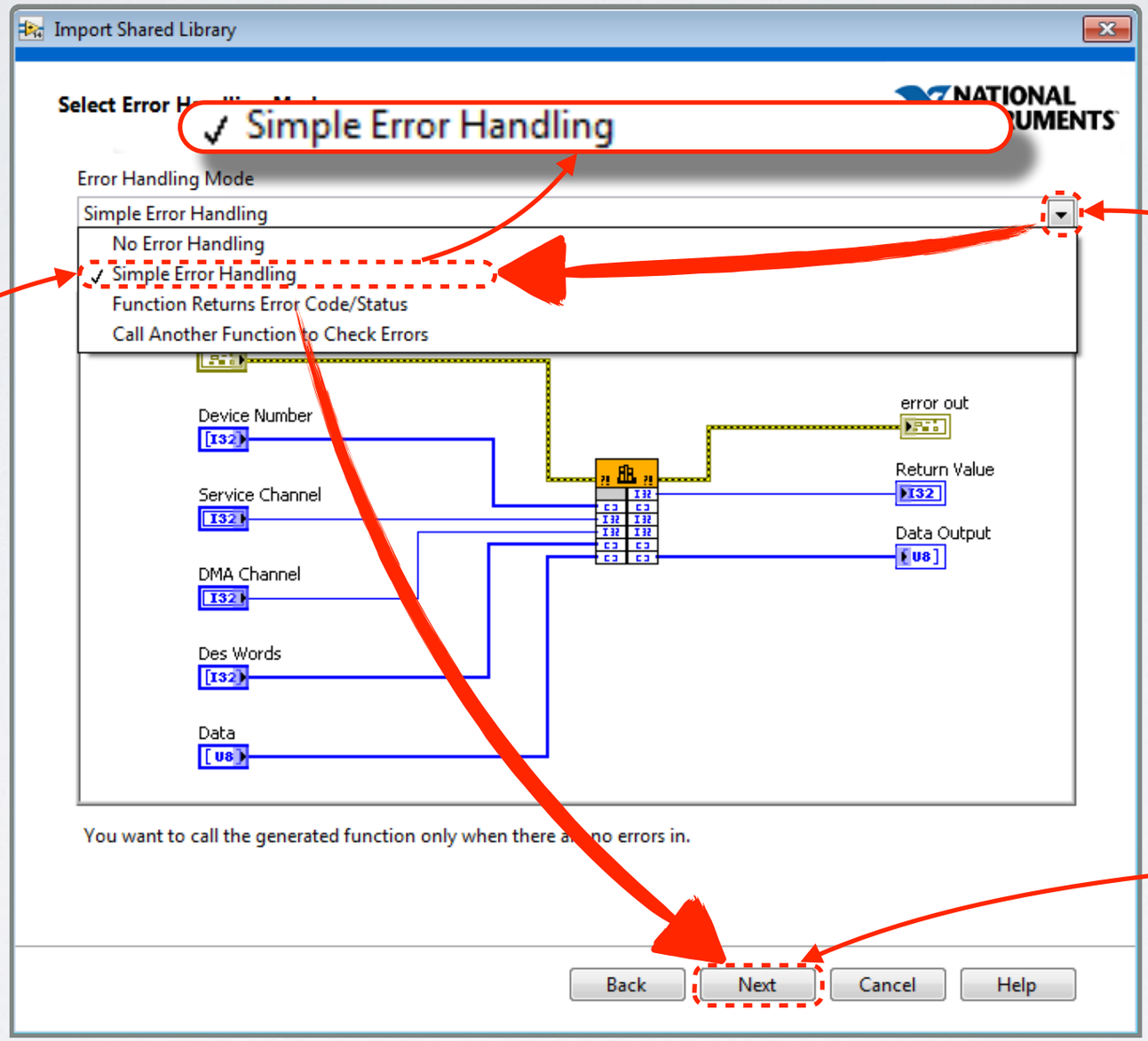
Verify that all checkboxes are selected.



Select "Next"

IMPORT SHARED LIBRARY

SELECT ERROR HANDLING MODE



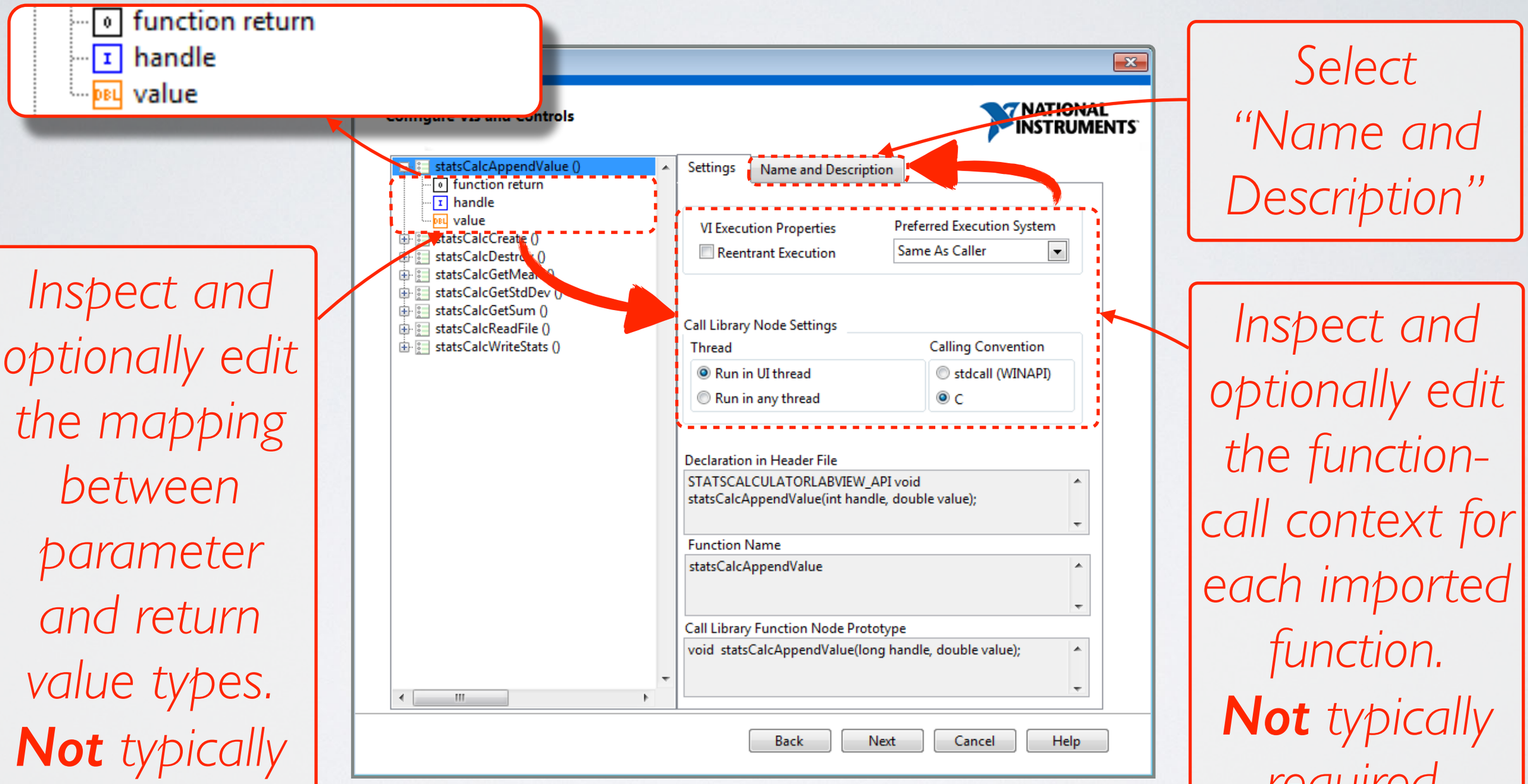
Select "Simple Error Handling"

Open Menu

Select "Next"

IMPORT SHARED LIBRARY

CONFIGURE VIS AND CONTROLS



Select "Name and Description"

Inspect and optionally edit the mapping between parameter and return value types. ***Not*** *typically required.*

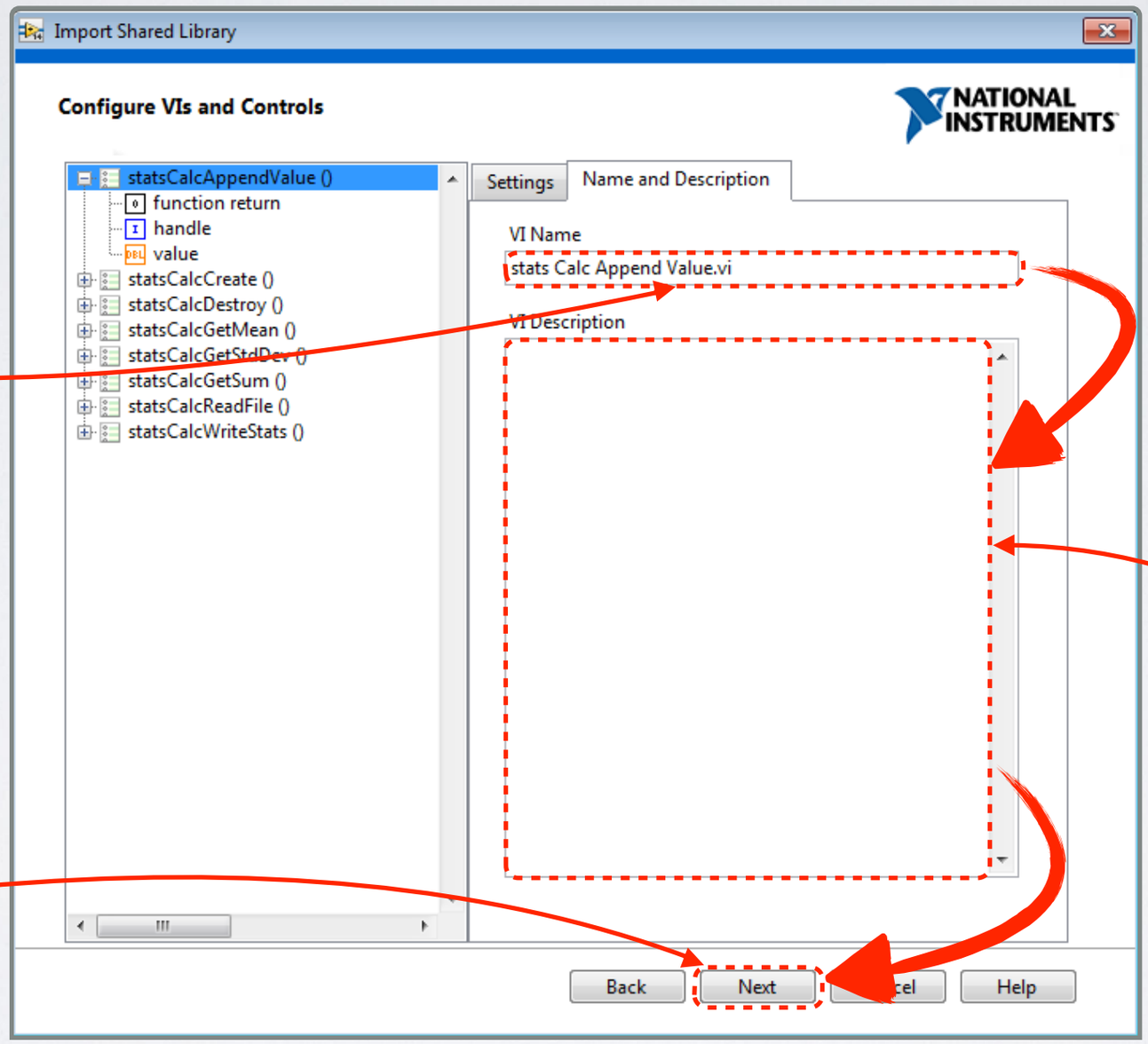
Inspect and optionally edit the function-call context for each imported function. ***Not*** *typically required.*

IMPORT SHARED LIBRARY

CONFIGURE VIS AND CONTROLS

Optionally specify a more user-friendly name for the VI that is mapped to the imported function.

Select "Next"

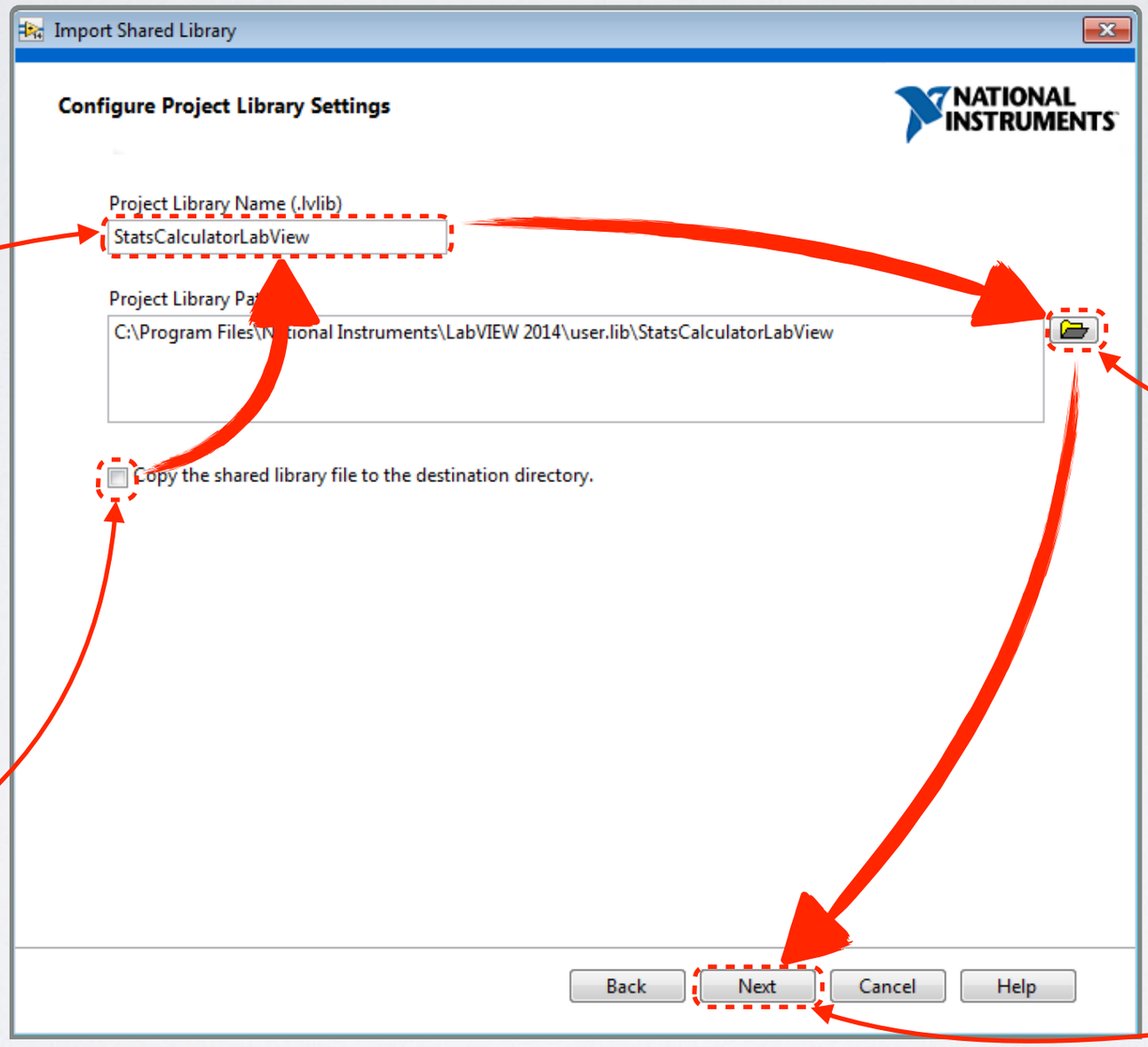


Optionally provide a description of the imported function's operation.

IMPORT SHARED LIBRARY

CONFIGURE PROJECT LIBRARY SETTINGS

Optionally alter the name of the LabVIEW VI library that will be generated.



Optionally browse to select an alternative path for the LabVIEW VI library that will be generated.

*Verify that checkbox is **not** selected.*

Select "Next"

IMPORT SHARED LIBRARY

GENERATION SUMMARY

Input file locations

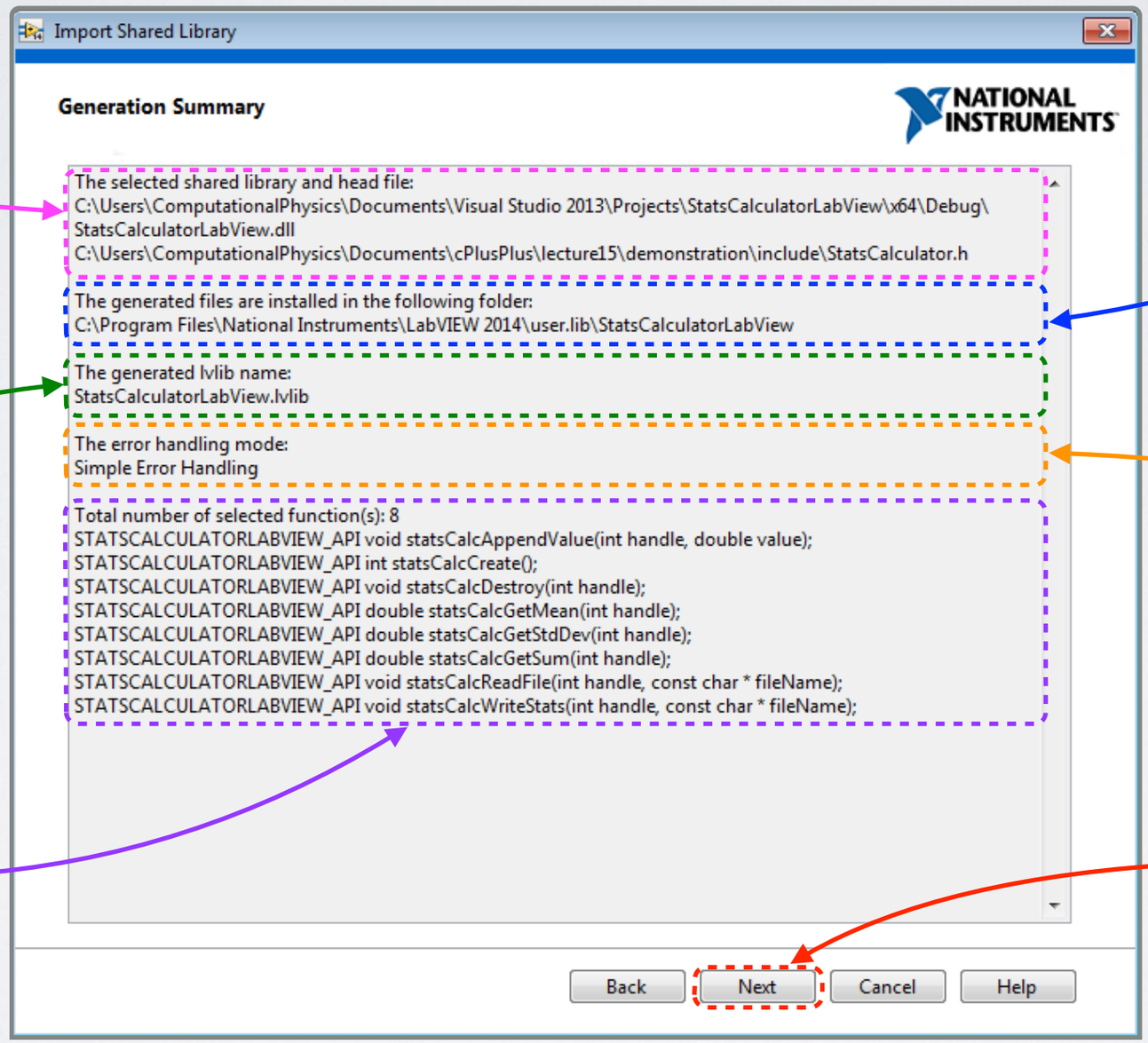
Generated LabVIEW library name.

Imported C++ function list.

Generated files location.

Chosen error-handling mode.

Select "Next"

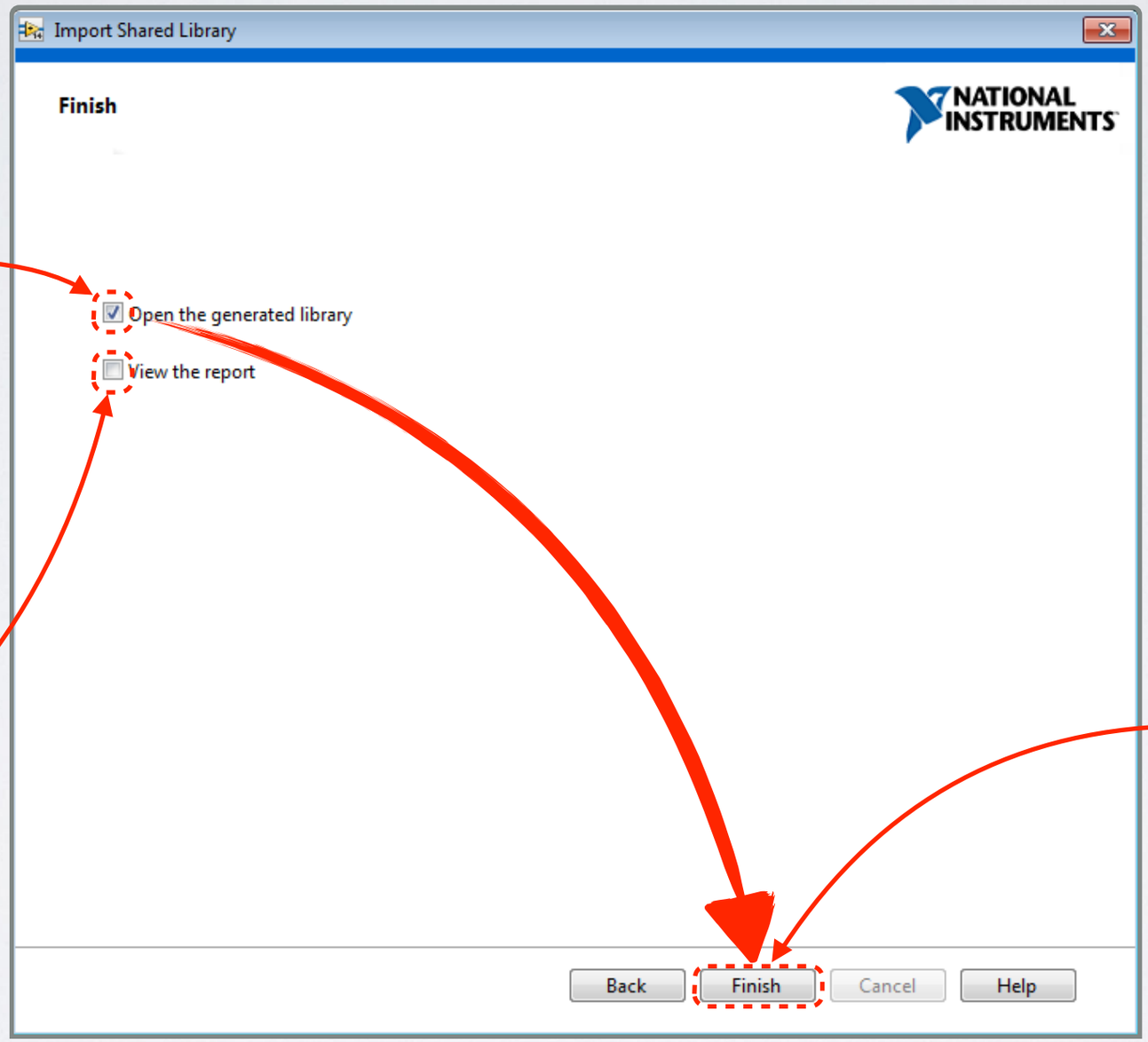


IMPORT SHARED LIBRARY

FINISH

Optionally inspect the VIs provided by the generated LabVIEW VI library.

Optionally view a report of the generation outcome

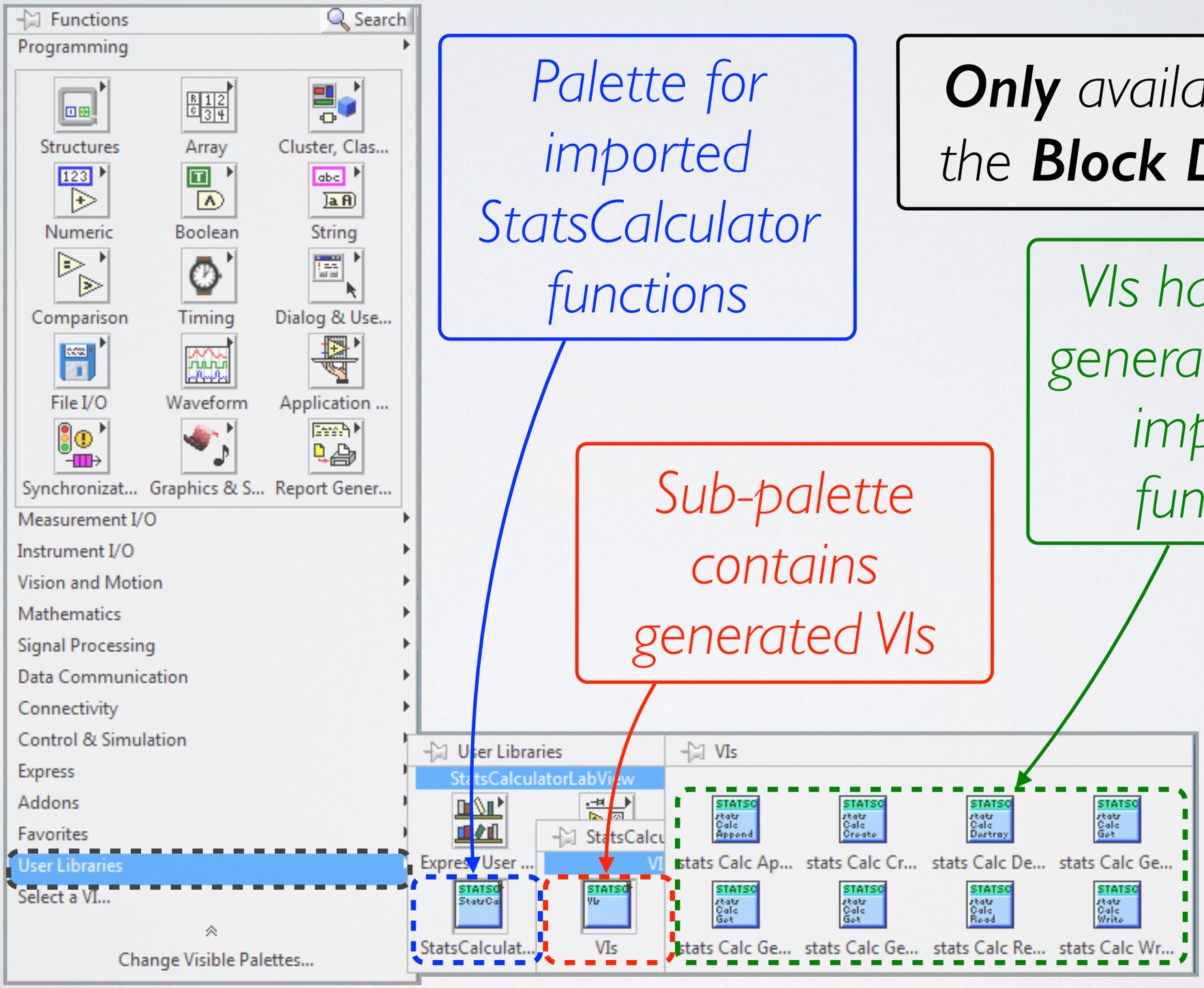


Select "Next"

STEP 7:

USE THE VIS

THE USER LIBRARIES PALETTE



USING THE VIS

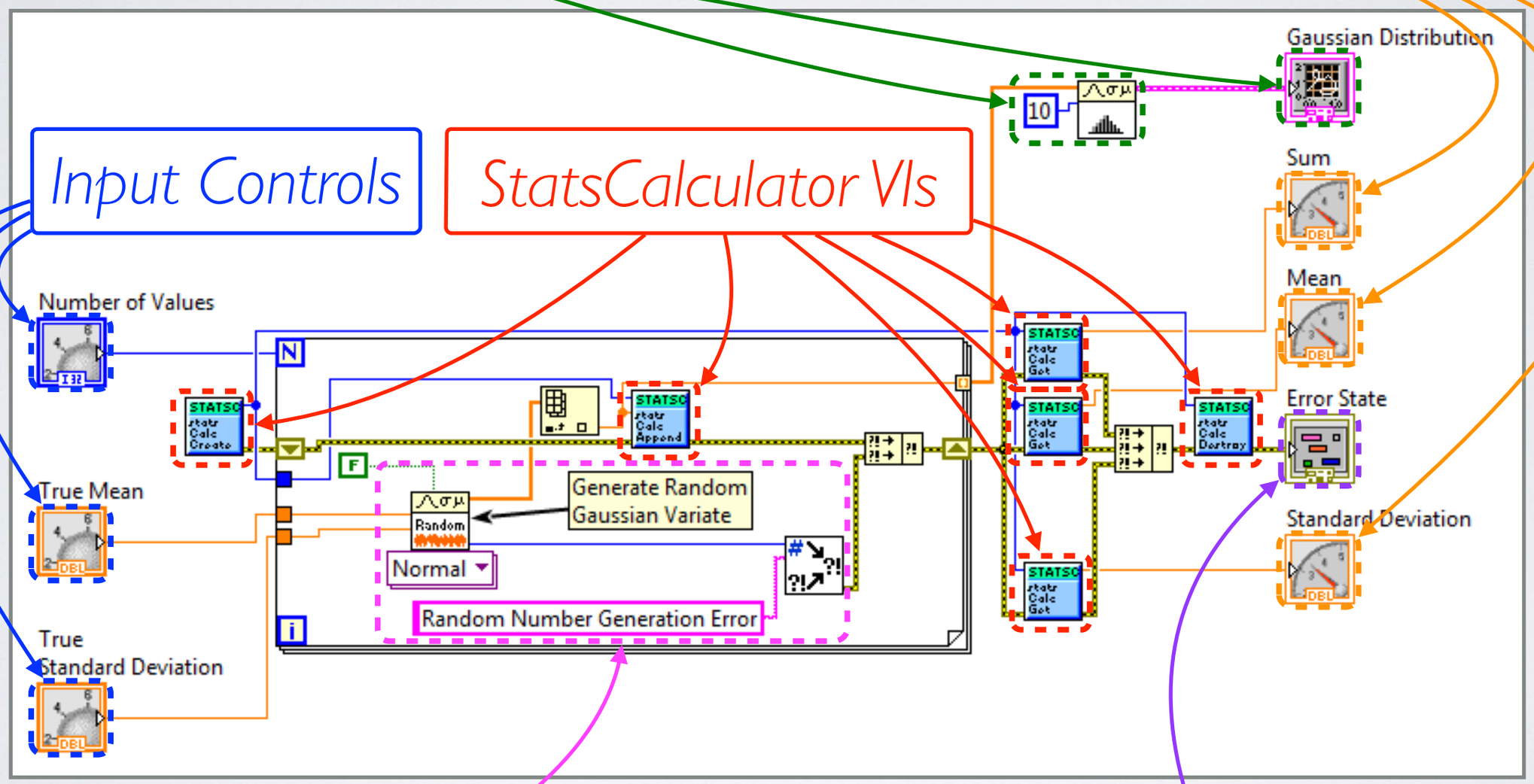
BLOCK DIAGRAM

Histogram and display data

Display StatsCalculator VI Output

Input Controls

StatsCalculator VIs

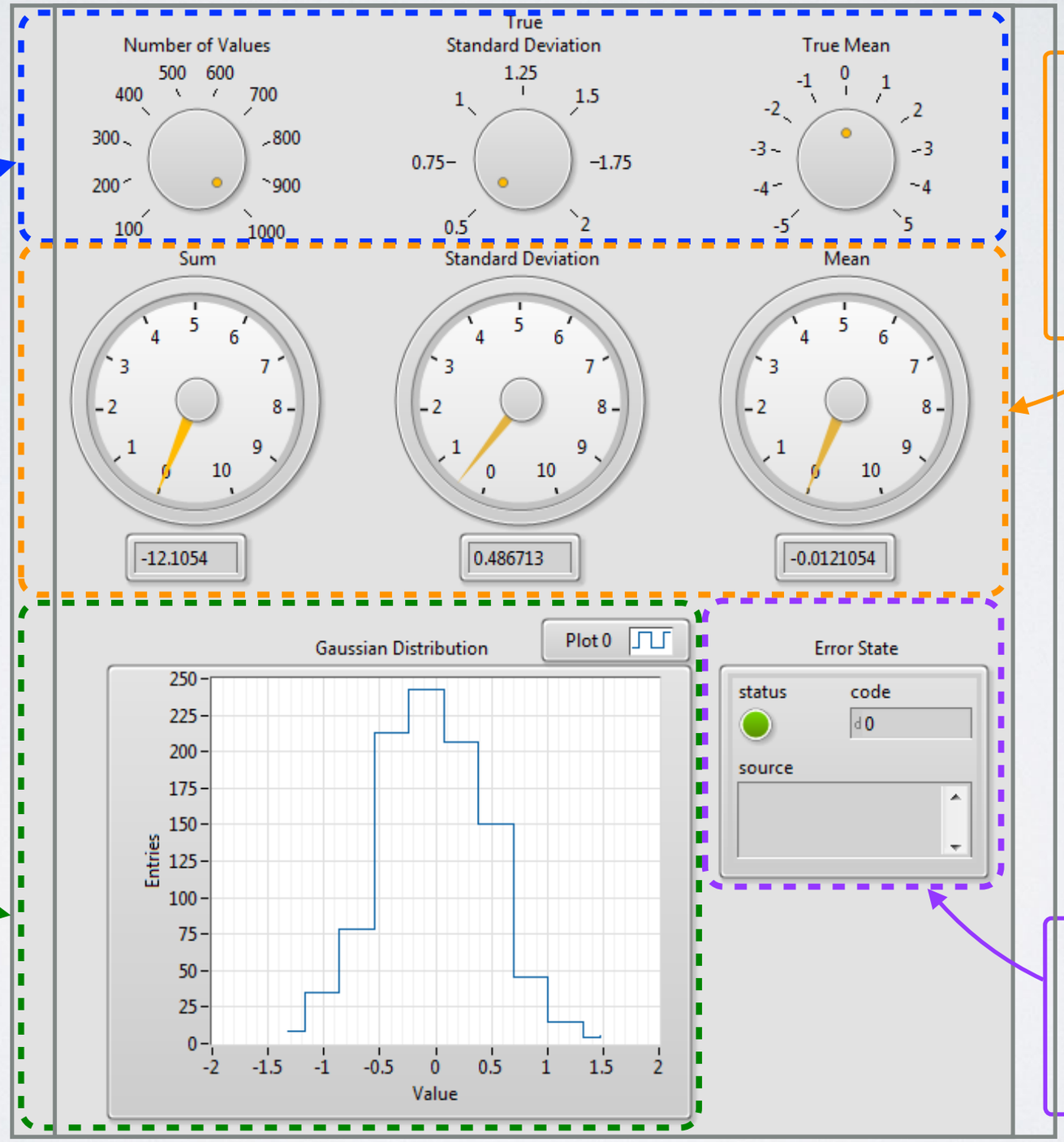


Generate Normal Random Variates

Display Error State

USING THE VIS

FRONT PANEL



Input Controls

Display StatsCalculator VI Output

Histogram and display data

Display Error State

CREATING A NEW VI FROM THE BLOCK DIAGRAM

ENCAPSULATION OF FUNCTIONALITY

- When building **non-trivial** programs using LabVIEW, the **complexity** of the Block Diagram rapidly **increases**.
- Often, the Block Diagram can be meaningfully **subdivided** into **functionally distinct** subsets of elements.

☞ LabVIEW allows these subsets to be aggregated and saved as **new VIs**.

- In addition to simplifying the block diagram the newly created VIs can be **reused** in other LabVIEW programs.
- This is similar to the **encapsulation of functionality** provided by Object Orientated programming languages.

DEMONSTRATION

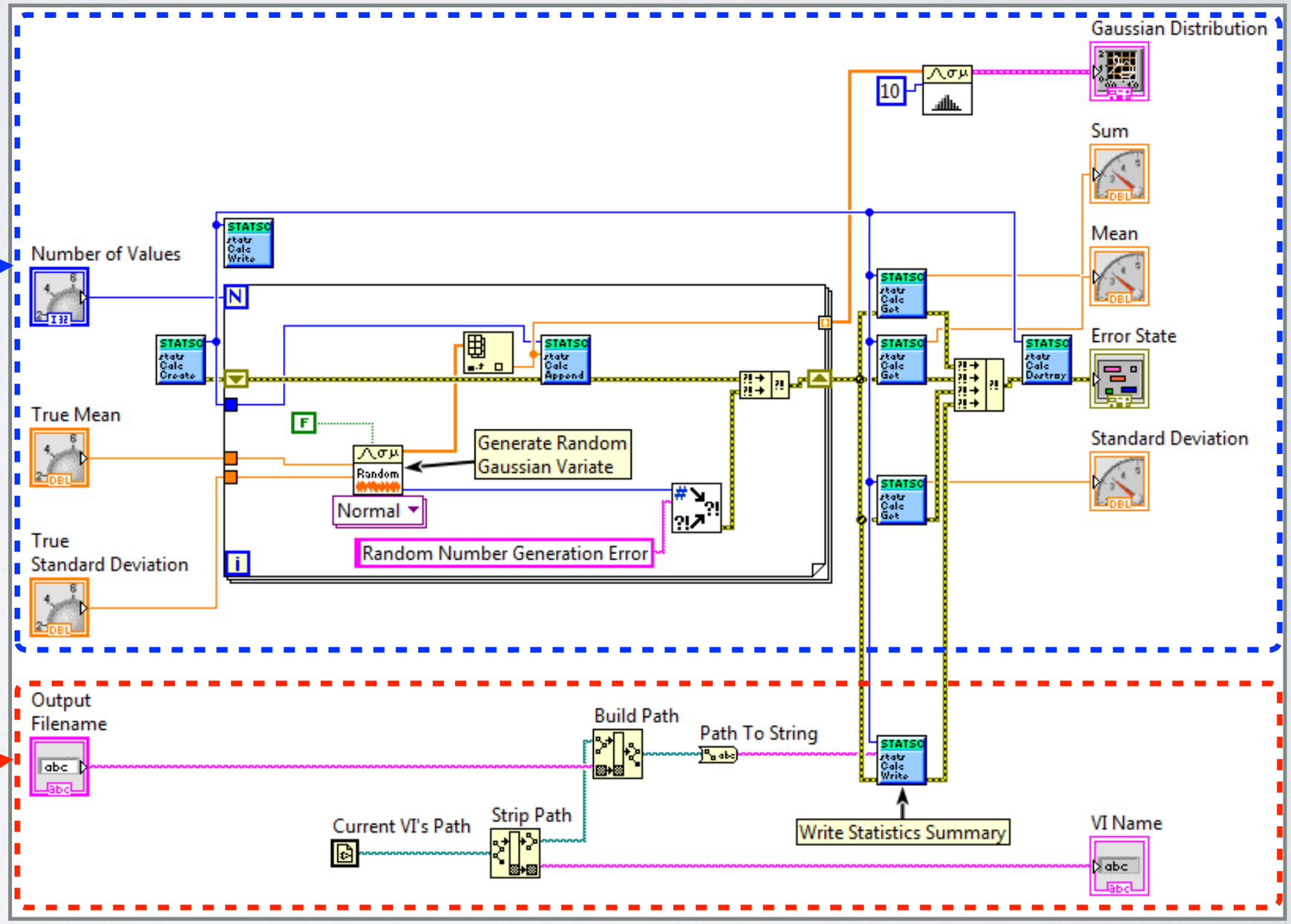
Creating a **New VI** from a **Subset of the Block Diagram**

CREATING A NEW VI

IDENTIFY LOGICALLY DISTINCT CODE

Data generation, analysis and display logic.

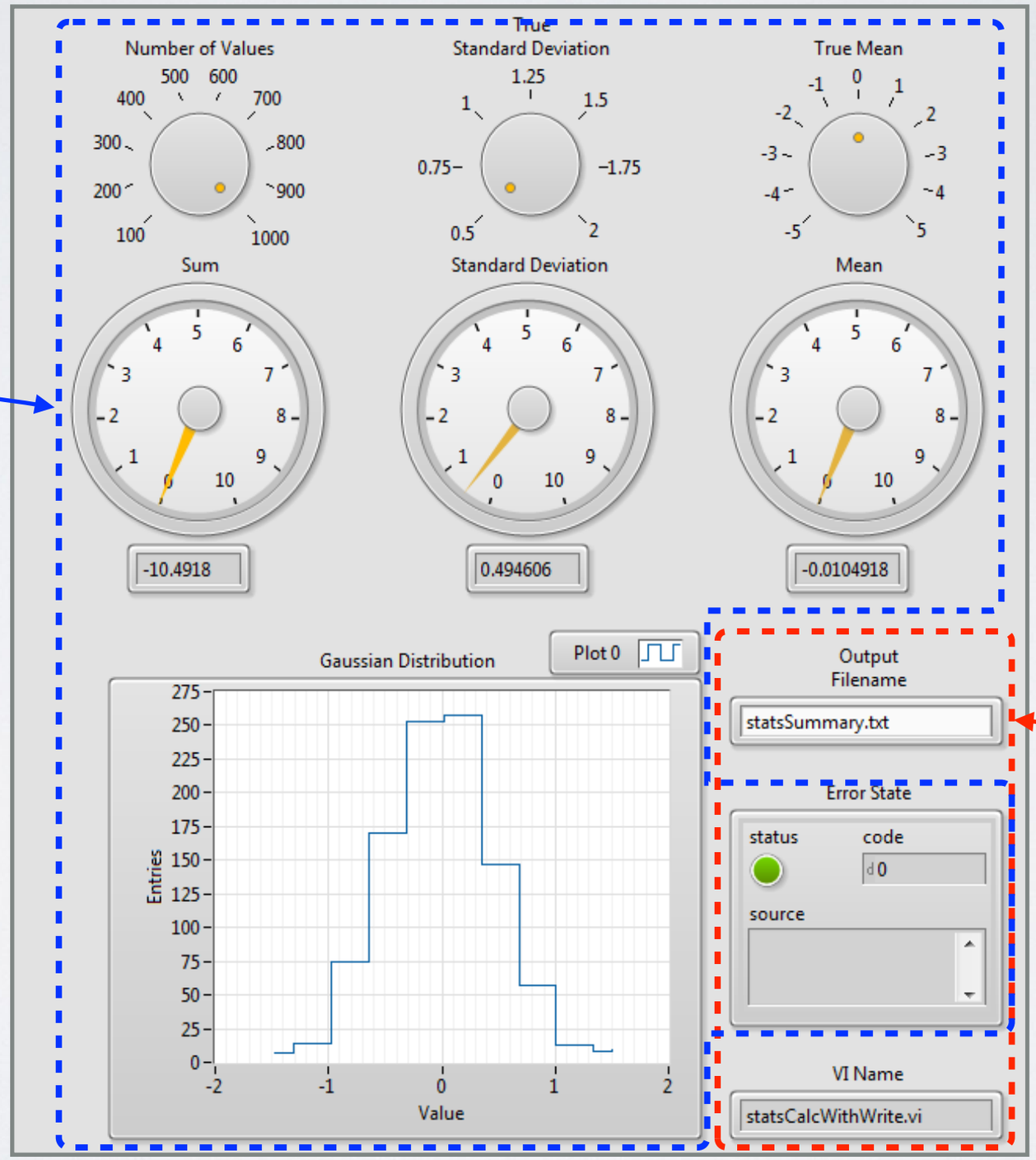
File output and path assembly logic.



CREATING A NEW VI

IDENTIFY LOGICALLY DISTINCT CODE

Data generation, analysis and display logic.



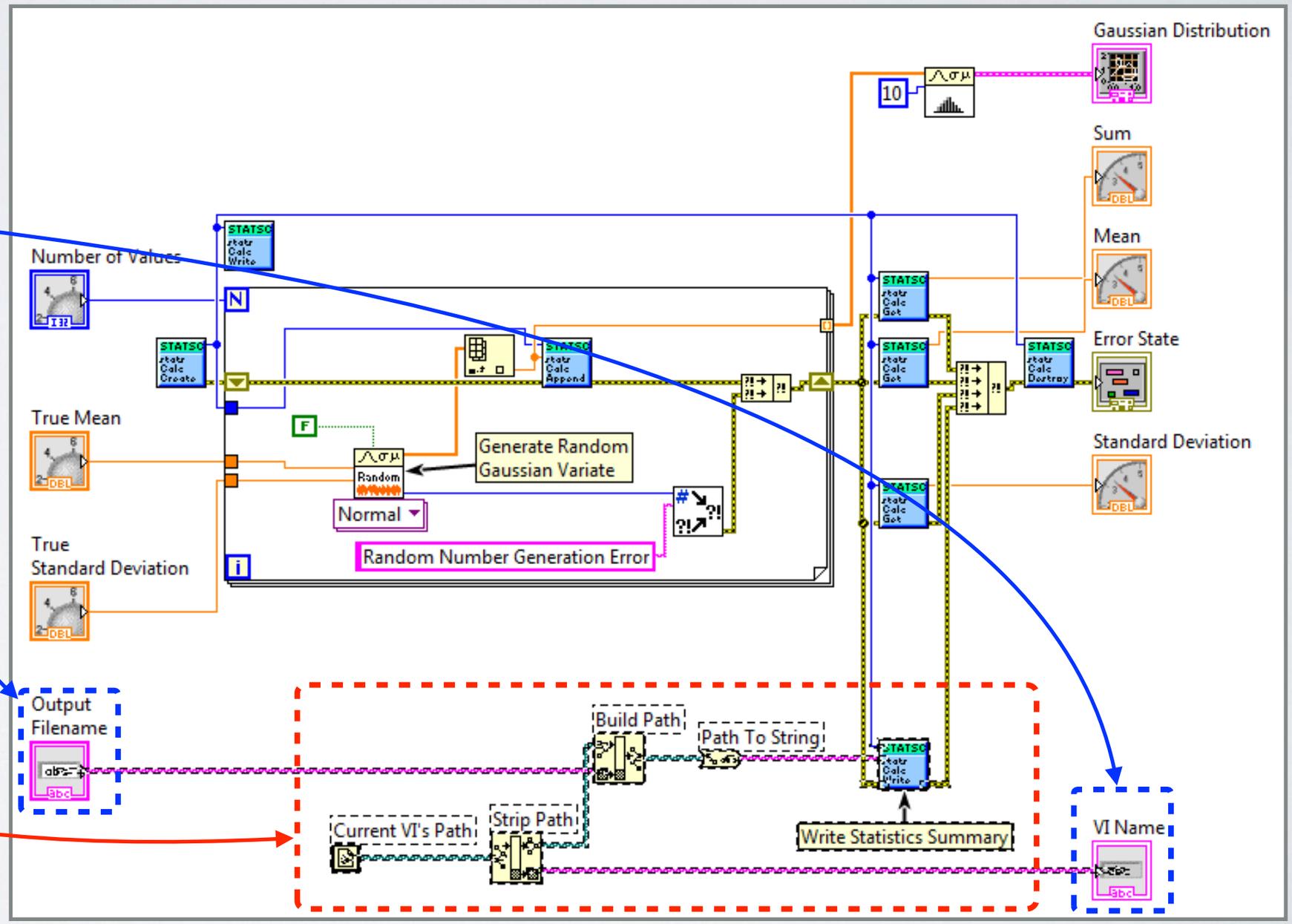
File output and path assembly logic.

CREATING A NEW VI

SELECT BLOCK DIAGRAM ELEMENTS TO MERGE

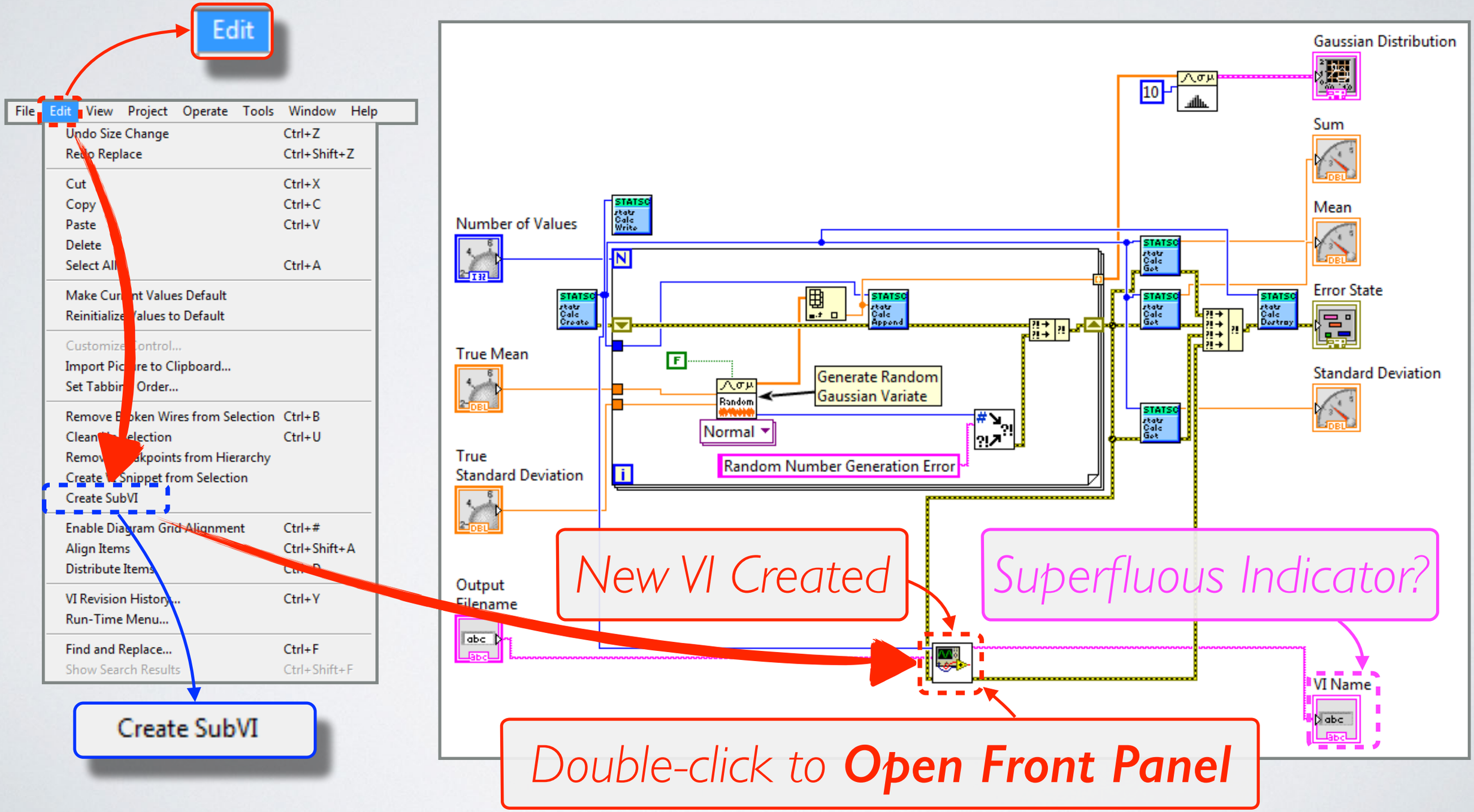
Omit ***controls*** *and* ***indicators****.*

Select ***functional*** *elements*



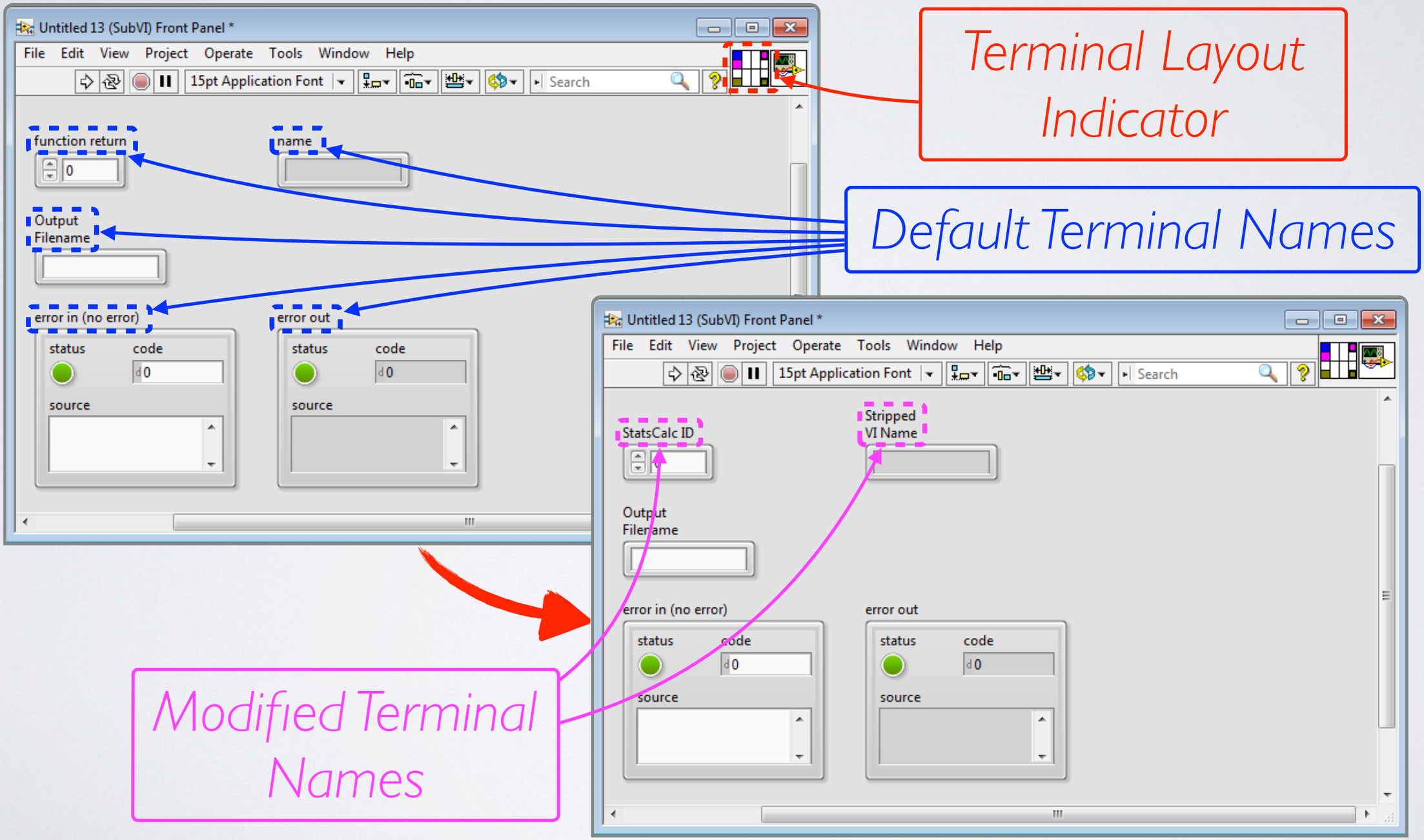
CREATING A NEW VI

SELECT *EDIT* → *CREATE SUBVI*



CREATING A NEW VI

MODIFY TERMINAL NAMES

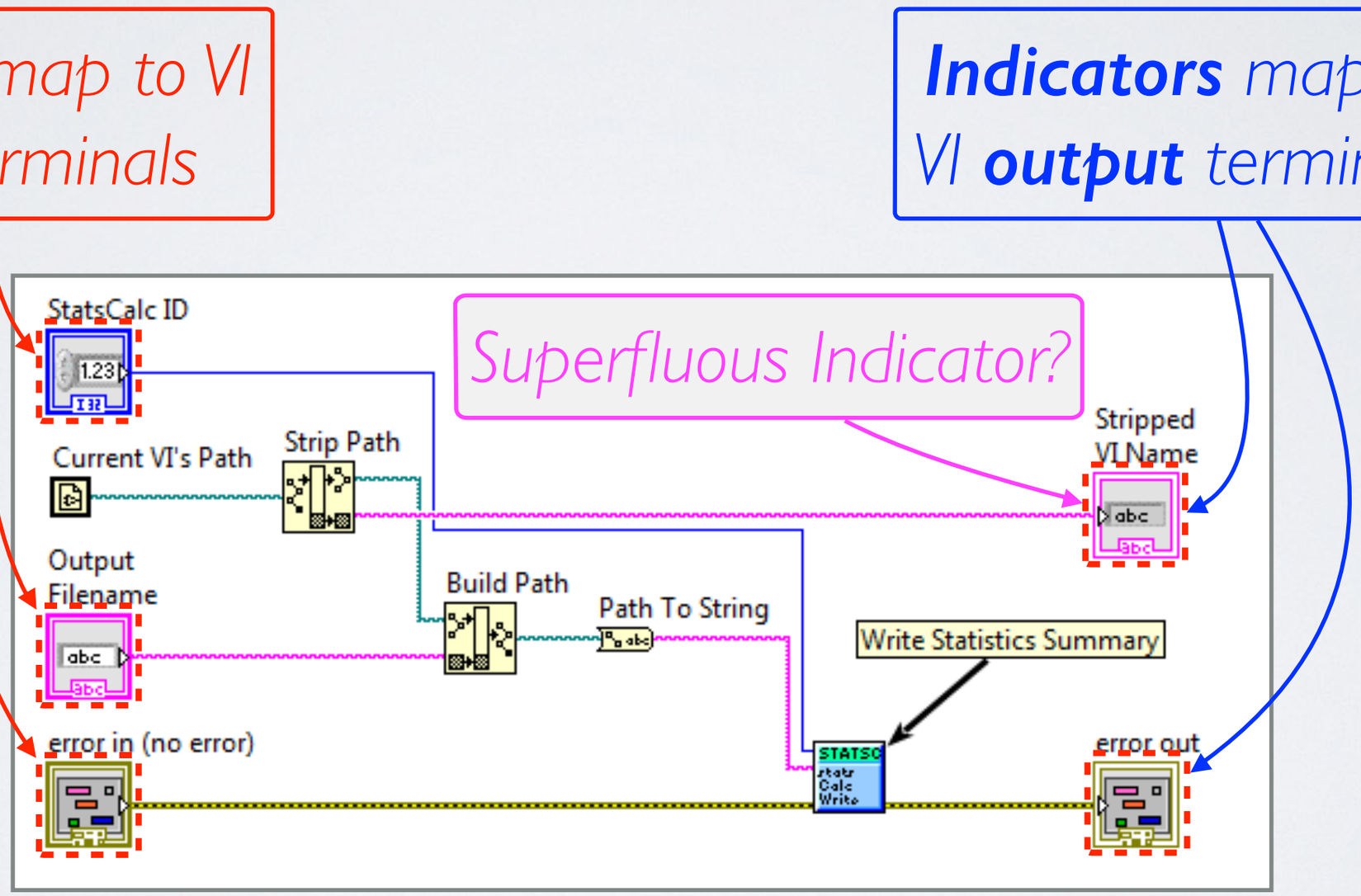


CREATING A NEW VI

INSPECT NEW VI BLOCK DIAGRAM

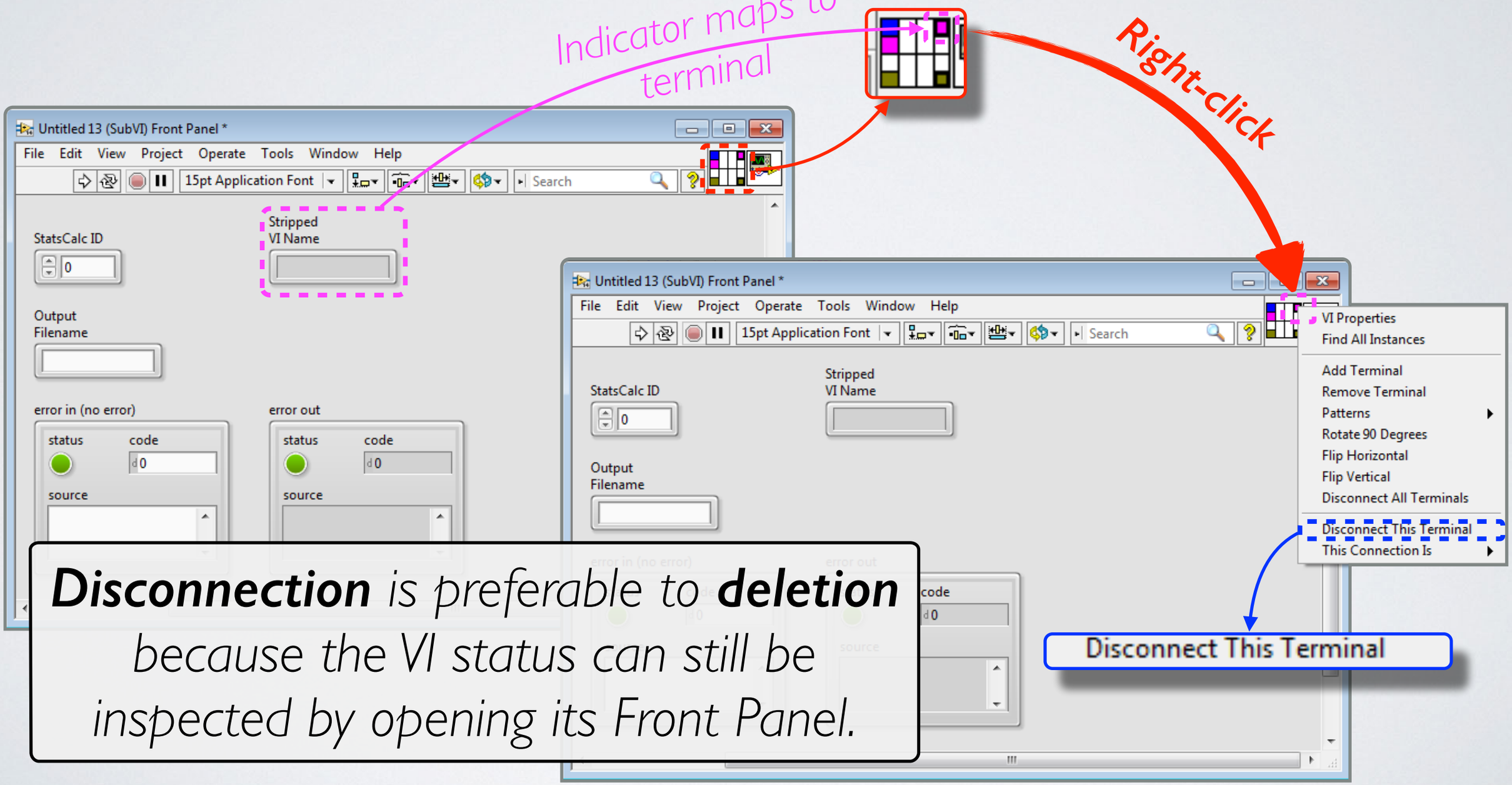
Controls *map to VI* ***input*** *terminals*

Indicators *map to VI* ***output*** *terminals*



CREATING A NEW VI

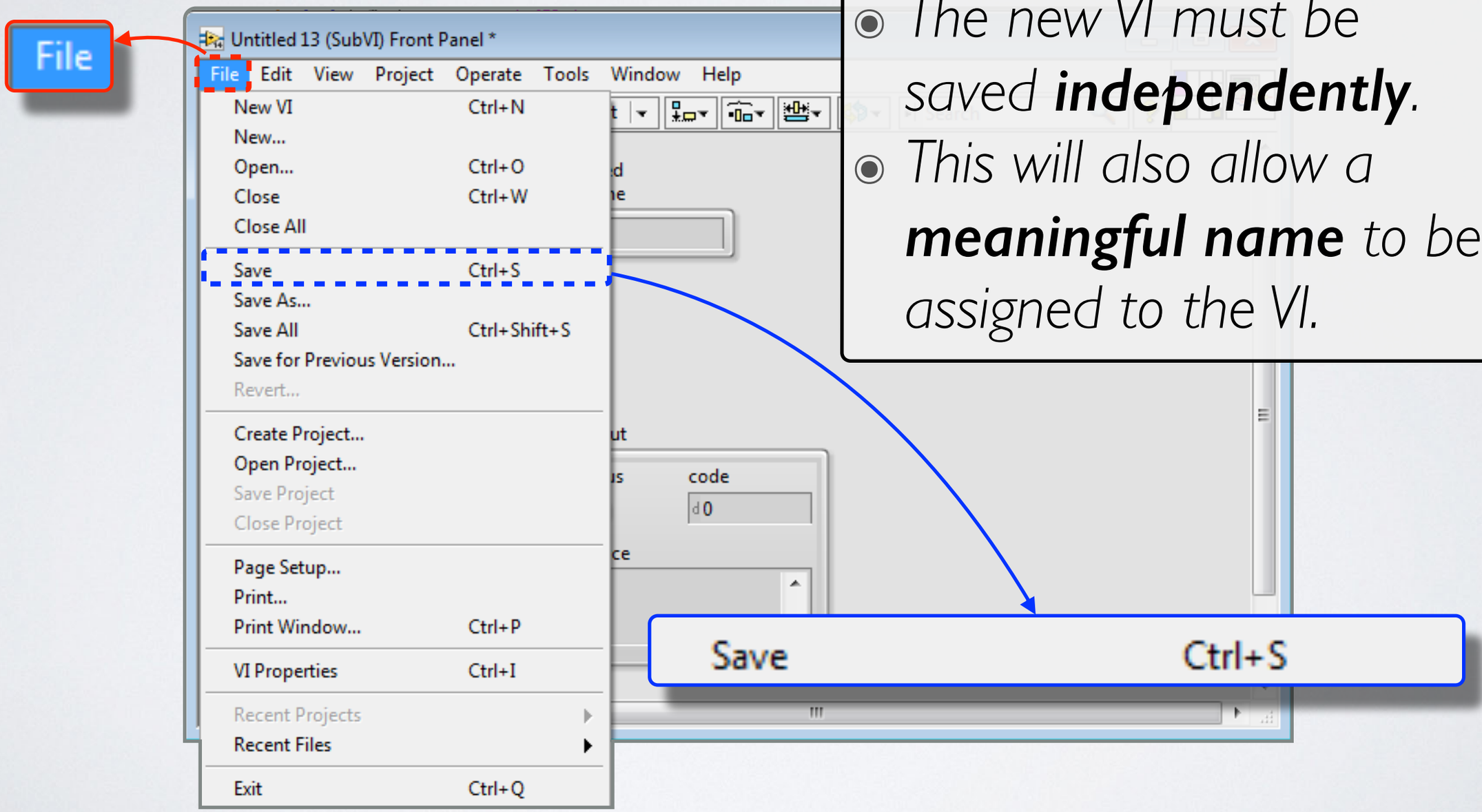
DISCONNECT SUPERFLOUS TERMINAL



Disconnection *is preferable to* ***deletion*** *because the VI status can still be inspected by opening its Front Panel.*

CREATING A NEW VI

SAVE THE NEW VI



- *The new VI must be saved **independently**.*
- *This will also allow a **meaningful name** to be assigned to the VI.*

RECOMMENDED READING

LabVIEW Manual: Creating SubVIs from Selections

(http://zone.ni.com/reference/en-XX/help/371361L-01/lvhowto/creatingsubvisfromviselec/)

LabVIEW Manual: System Exec VI

(http://zone.ni.com/reference/en-XX/help/371361L-01/glang/system_exec/)

LabVIEW Manual: Formula Node

(http://zone.ni.com/reference/en-XX/help/371361L-01/glang/formula_node/)

LabVIEW Manual: Using the Import Shared Library Wizard

(http://zone.ni.com/reference/en-XX/help/371361L-01/lvexcodeconcepts/importing_shared_library/)

LECTURE 15 HOMEWORK

Review the ***Recommended Reading*** items listed on the previous slide.

Continue to refine your **final project proposal** and continue working on your **final project**.