Vivado and SDK Development Flow

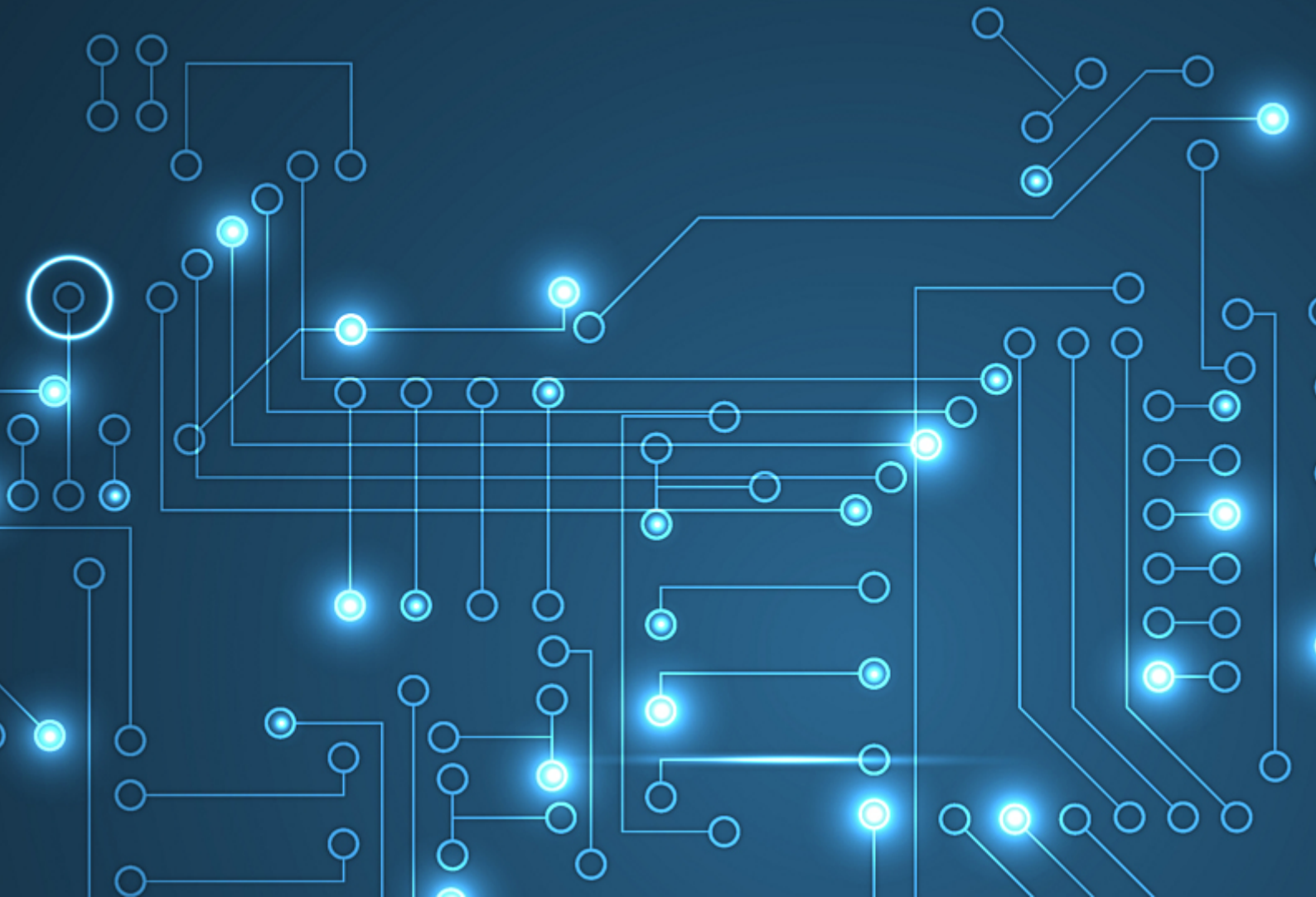
A companion guide to the text book:

A Practical Introduction to the Xilinx Zynq-7000 Adaptive SoC

Author: Derek Murray

Version: 1.0

Date: 29/8/21



Revision History

Version	Date	Comment
1.0	29/8/21	First version.

Table 1. Revision History

Limit of Liability/Disclaimer of Warranty: The author makes no representation or warranty with respect to the accuracy or completeness of the contents of this work and specifically disclaims all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies contained herein may not be suitable for every situation. If improperly wired, circuits described in this work may possibly cause damage to the device and physical injury. The author shall not be liable for damages arising herefrom. The fact that an organisation or website is referred to in this work as a citation and/or a potential source of further information does not mean that the author endorses the information the organisation or website may provide or recommendations it may make. Further, readers should be aware that Internet websites listed in this work may have changed or disappeared between when this work was written and when it is read.

1 Introduction

This is a support document for the text book "A Practical Introduction to the Xilinx Zynq-7000 Adaptive SoC". Both Xilinx SDK and the Vitis unified software platform can be used to develop the projects in the text — this document targets SDK, and it can be used for Vivado/SDK combinations from Version 2017.4 to Version 2019.1. (It might also be applicable for earlier versions — YMMV.)

1.1 Xilinx Software

At the time of writing, archive versions of SDK can be downloaded from the following location: https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vitis/archive-sdk.html

The related Vivado IDE can be downloaded from the following archive page: https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/archive.html

1.2 Guide Layout

Section 2 of this document provides a step-by-step guide to building the hardware project in the main book, and the design details can be found in Chapter 6 of the text.

Section 3 of this document shows how to recreate the software projects in the textbook. It does not cover the technical details of each project; the book will need to be referred to for complete details. Table 2 shows the correlation between the software projects in the textbook and the related section in this guide.

Software Project	Book Chapter	Section in this Guide
1	7	Section 3.1
2	8	Section 3.2
3	9	Section 3.3
4	10	Section 3.4
5	11	Section 3.5
6	12	Section 3.6
7	13	Section 3.7
8	14	Section 3.8
9	16	Section 3.9
10	18	Section 3.10

Table 2. Correlation between the software projects in the textbook and the relevant section in this guide.

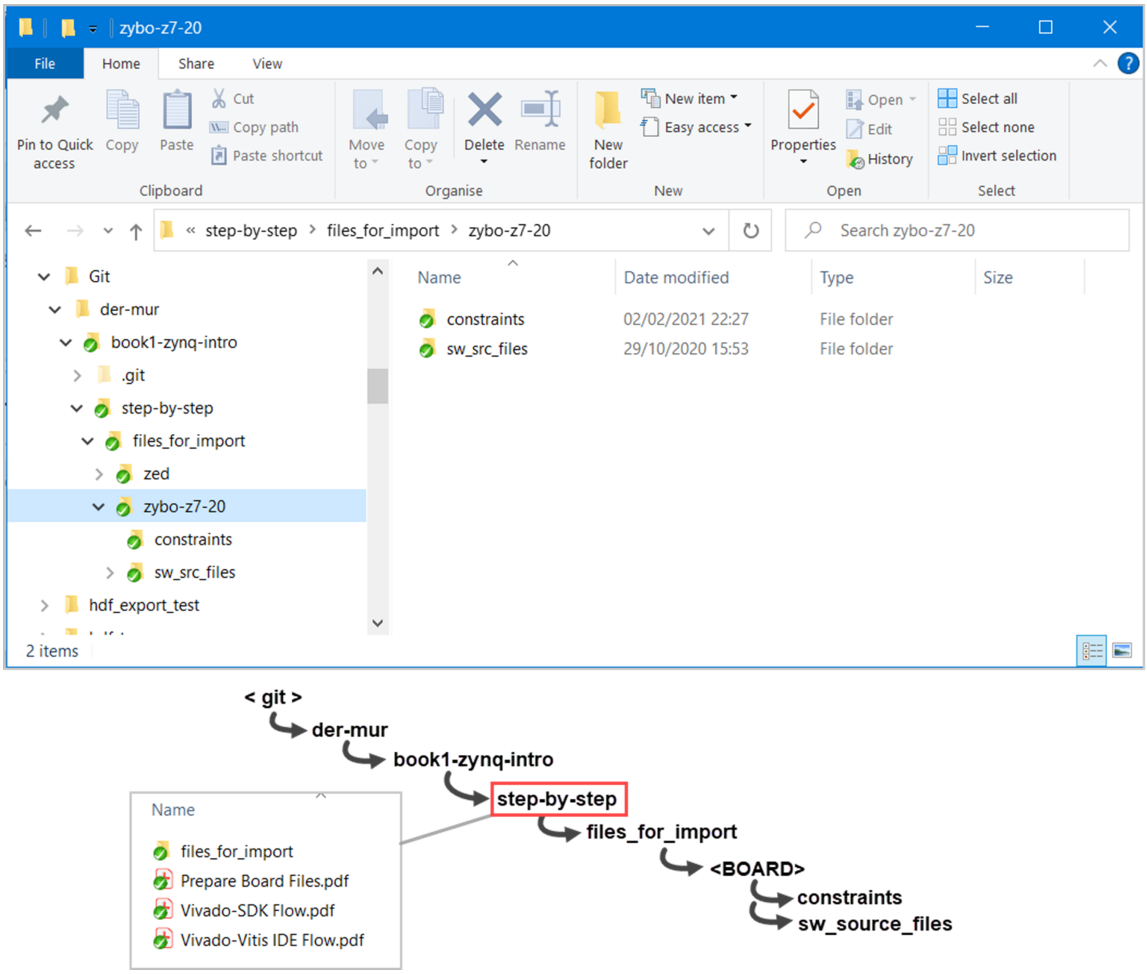


Figure 1. GitHub layout for the support material.

1.3 Accompanying Material

To complete the hardware and software projects in the textbook (and in this guide), some additional resources are required. Constraint files are needed for the FPGA design, and C source files are needed for the software projects — currently, these can be found at the authors GitHub repository:

- https://github.com/der-mur/book1-zynq-intro

Figure 1 illustrates the GitHub layout. The Digilent Zybo-Z7-20 is the main developmental platform used in the text book, although the related resources are also directly compatible with the Zybo-Z7-10. In addition, C code and constraint files have been added for the Digilent/Avnet ZedBoard. If using the Zybo-Z7-20 (or -10), the board set-up for each project can be found in the textbook. The corresponding ZedBoard details can be seen in Table 3, Table 4, and Table 5.

IP Integrator/ XDC Name	Zybo-Z7-20 Board Interface	Zybo-Z7-20 Pin	ZedBoard Board Interface	ZedBoard Pin
TTC0_WAVE0_OUT	Pmod JC- Pin 1	V15	Pmod JB- Pin 1	W12
TTC0_WAVE1_OUT	Pmod JC- Pin 2	W15	Pmod JB- Pin 2	W11
TTC0_WAVE2_OUT	Pmod JC- Pin 3	T11	Pmod JB- Pin 3	V10
TTC1_WAVE0_OUT	Pmod JC- Pin 7	W14	Pmod JB- Pin 7	V12
TTC1_WAVE1_OUT	Pmod JC- Pin 8	Y14	Pmod JB- Pin 8	W10
TTC1_WAVE2_OUT	Pmod JC- Pin 9	T12	Pmod JB- Pin 9	V9
cs_n	Pmod JD- Pin 1	T14	Pmod JD- Pin 1	V7
mosi	Pmod JD- Pin 2	T15	Pmod JD- Pin 2	W7
miso	Pmod JD- Pin 3	P14	Pmod JD- Pin 3	V5
sck	Pmod JD- Pin 4	R14	Pmod JD- Pin 4	V4
PMOD_ACL_INT2	Pmod JD- Pin 7	U14	Pmod JD- Pin 7	W6
PMOD_ACL_INT1	Pmod JD- Pin 8	U15	Pmod JD- Pin 8	W5
gpio_out[4]	Pmod JE- Pin 1	V12	Pmod JC- Pin 1	AB7
gpio_out[5]	Pmod JE- Pin 2	W16	Pmod JC- Pin 2	AB6
gpio_out[6]	Pmod JE- Pin 3	J15	Pmod JC- Pin 3	Y4
gpio_out[7]	Pmod JE- Pin 4	H15	Pmod JC- Pin 4	AA4
gpio_out[0]	LD0	M14	LD0	T22
gpio_out[1]	LD1	M15	LD1	T21
gpio_out[2]	LD2	G14	LD2	U22
gpio_out[3]	LD3	D18	LD3	U21
gpio_in[8]	Pmod JE- Pin 7	V13	Pmod JC- Pin 7	R6
gpio_in[9]	Pmod JE- Pin 8	U17	Pmod JC- Pin 8	T6
gpio_in[10]	Pmod JE- Pin 9	T17	Pmod JC- Pin 9	T4
gpio_in[11]	Pmod JE- Pin 10	Y17	Pmod JC- Pin 10	U4
gpio_in[4]	SW0	G15	SW0	F22
gpio_in[5]	SW1	P15	SW1	G22
gpio_in[6]	SW2	W13	SW2	H22
gpio_in[7]	SW3	T16	SW3	F21
gpio_in[0]	BTN0	K18	BTNU	T18
gpio_in[1]	BTN1	P16	BTNR	R18
gpio_in[2]	BTN2	K19	BTND	R16
gpio_in[3]	BTN3	Y16	BTNL	N15

Table 3. Comparison of Zybo-Z7-20 and ZedBoard board details

Project Function	Zybo-Z7-20 Board Interface	ZedBoard Board Interface
gpio_out[0-3]	LD0 - LD3	LD0 - LD3
gpio_out[4-7]	Pmod JE [1-4]	Pmod JC [1-4]
gpio_in[0]	BTN0	BTNU
gpio_in[1]	BTN1	BTNR
gpio_in[2]	BTN2	BTND
gpio_in[3]	BTN3	BTNL
gpio_in[4-7]	SW0 - SW3	SW0 - SW3
gpio_in[8-11]	Pmod JE [7-10]	Pmod JC [7-10]
TTC0 WAVE Outputs	Pmod JC	Pmod JB
Pmod ACL	Pmod JD	Pmod JD

Table 4. Summary of Zybo-Z7-20 and ZedBoard interface details

MIO	Zybo-Z7-20 Board Interface	ZedBoard Board Interface
7	LD4	LD9
13	Pmod JF - Pin 1	Pmod JE - Pin 1
10	Pmod JF - Pin 2	Pmod JE - Pin 2
11	Pmod JF - Pin 3	Pmod JE - Pin 3
12	Pmod JF - Pin 4	Pmod JE - Pin 4
0	Pmod JF - Pin 7	Pmod JE - Pin 7
9	Pmod JF - Pin 8	Pmod JE - Pin 8
14	Pmod JF - Pin 9	Pmod JE - Pin 9
15	Pmod JF - Pin 10	Pmod JE - Pin 10
50	BTN4	BTN8
51	BTN5	BTN9

Table 5. MIO interface details for the Zybo-Z7-20 and the ZedBoard

1.4 Required Hardware

- Digilent Zybo-Z7-20 (or Zybo-Z7-10).
- The ZedBoard can also be used, as mentioned earlier.
- Digilent Pmod TPH2. (Ideally, three are required.)

- A 16-channel Logic Analyser. (The Digilent Analog Discovery 2 is used in the text, although any suitable logic analyser can be used.)

1.5 Required Software

A terminal emulator such as Tera Term is required when running the software projects in Section 3. (Note that the Xilinx development tools also have a built-in terminal which can be used.)

1.6 Possible Directory Layout

For readers who do not already have a directory structure in place for project development, Figure 2 shows the layout favoured by the author.

- **Level 1**: A base directory called *zynq_proj*
- **Level 2**: The current tool version (2017_4, 2021_1, etc.)
- **Level 3**: The platform (Zybo-Z7-20, Zybo-Z7-10, ZedBoard, etc.)

The directories for individual projects will then be automatically created in Vivado the start of development; see Figure 5, for example, where the project for this guide is created as *hw_proj1*.

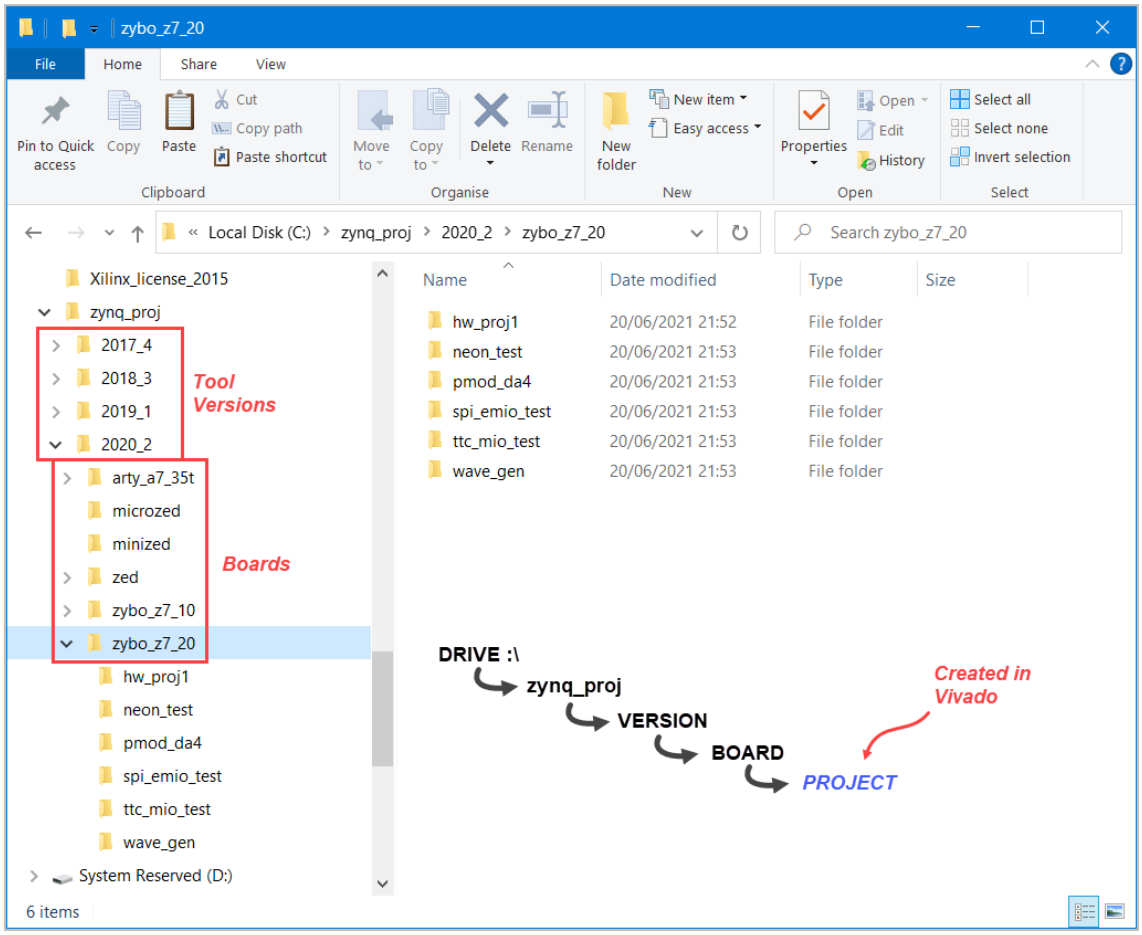


Figure 2. Possible Directory Structure

1.7 Setting Up the Zybo-Z7-20

If using the Zybo-Z7-20 (or Zybo-Z7-10), the reader can optionally follow the steps outlined in an additional document called "*Prepare Board Files.pdf*". (See Figure 1 for the location of this guide in GitHub.) This particular document is not essential, especially if the reader has already set up their platform correctly, although different warnings may appear when working through the steps in the current guide.

1.8 Getting started

1. Download and install Vivado/SDK (see Section 1.1 for software links).
2. Create a directory structure (see Section 1.6 for a possible layout).
3. Optionally, prepare the board files if using the Zybo-Z7-20 or Zybo-Z7-10 (see Section 1.7).
4. Load Vivado and go to Section 2.

A final disclaimer: while every effort has been made to ensure that the steps in this guide are accurate, it is a large document and minor discrepancies may exist (especially as different software versions are supported). If a step is missing, the default option is usually correct!

1.9 Useful Digilent links

- Installing Vivado, Xilinx SDK, and Digilent Board Files

2 Project Design in Vivado

2.1 Create the Project

Figure 3. When Vivado first launches, select Create Project.

Figure 4. Click Next to continue.

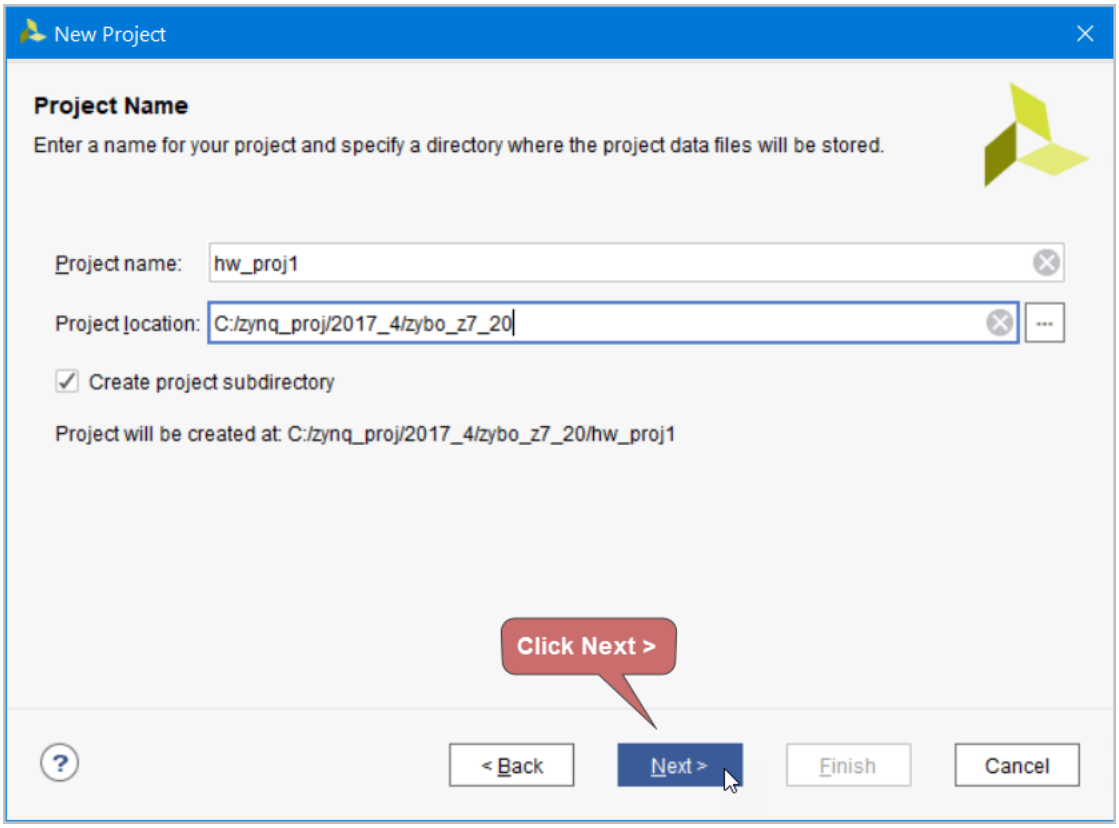


Figure 5. Choose a suitable project location and name ("hw_proj1" is used here).

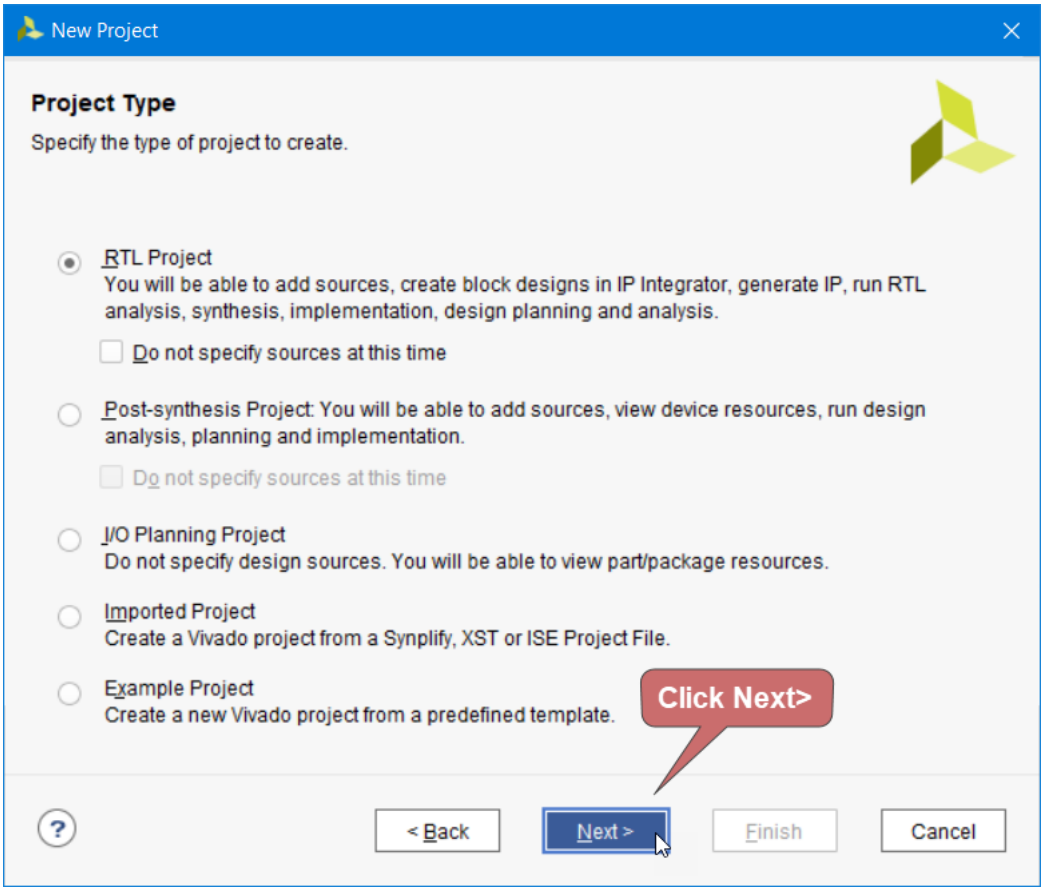


Figure 6. Keep the defaults (RTL Project) and click Next to continue.

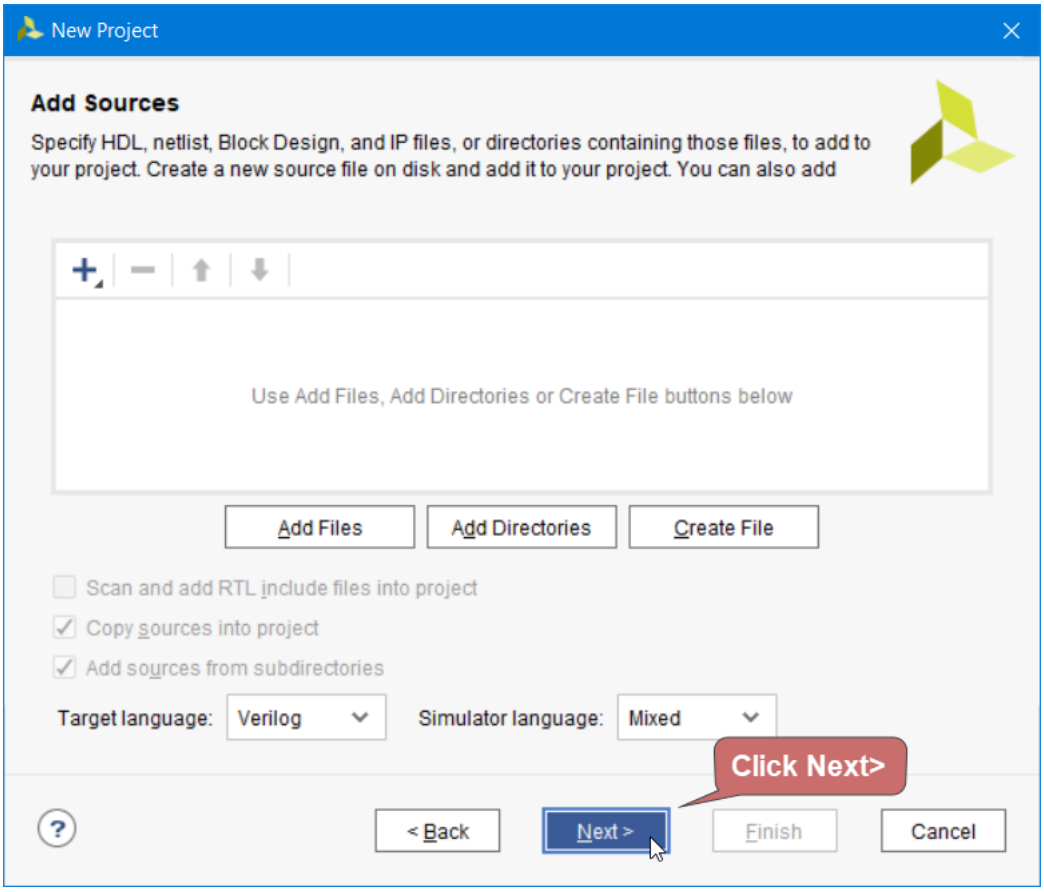


Figure 7. There are no files to add; click Next to continue.

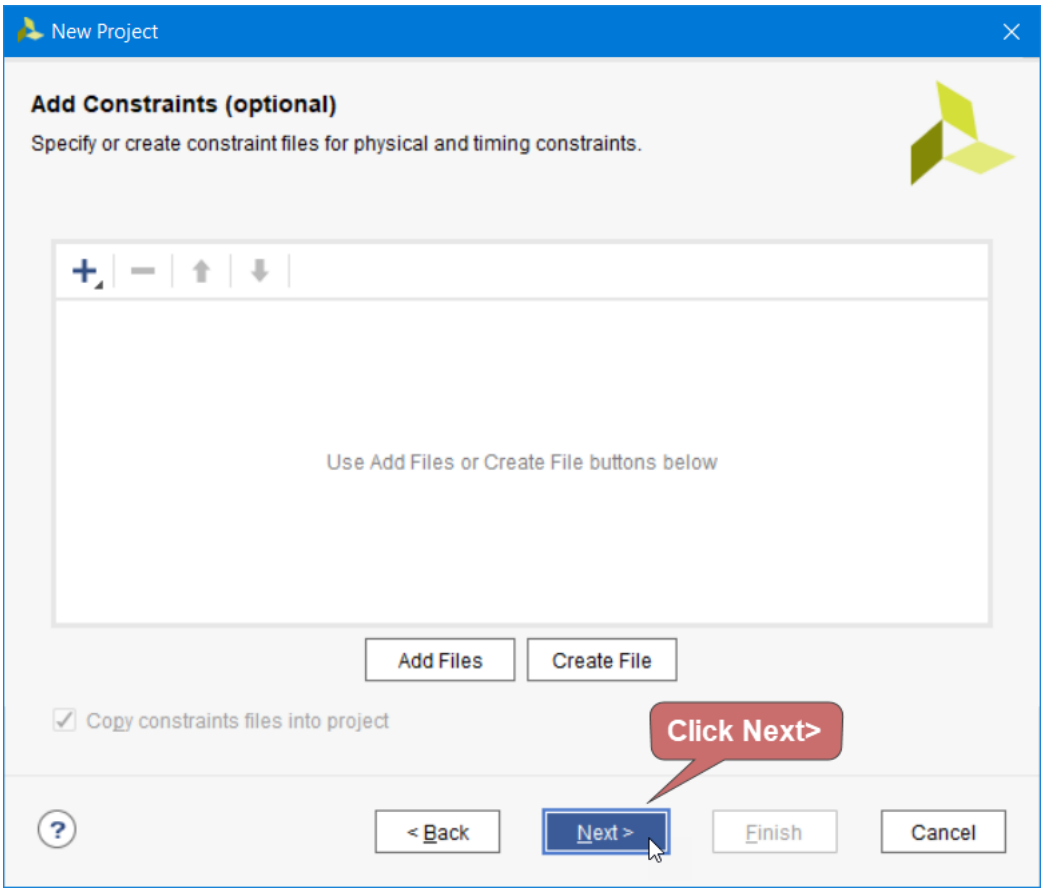


Figure 8. There are no constraints to add; click Next to continue.

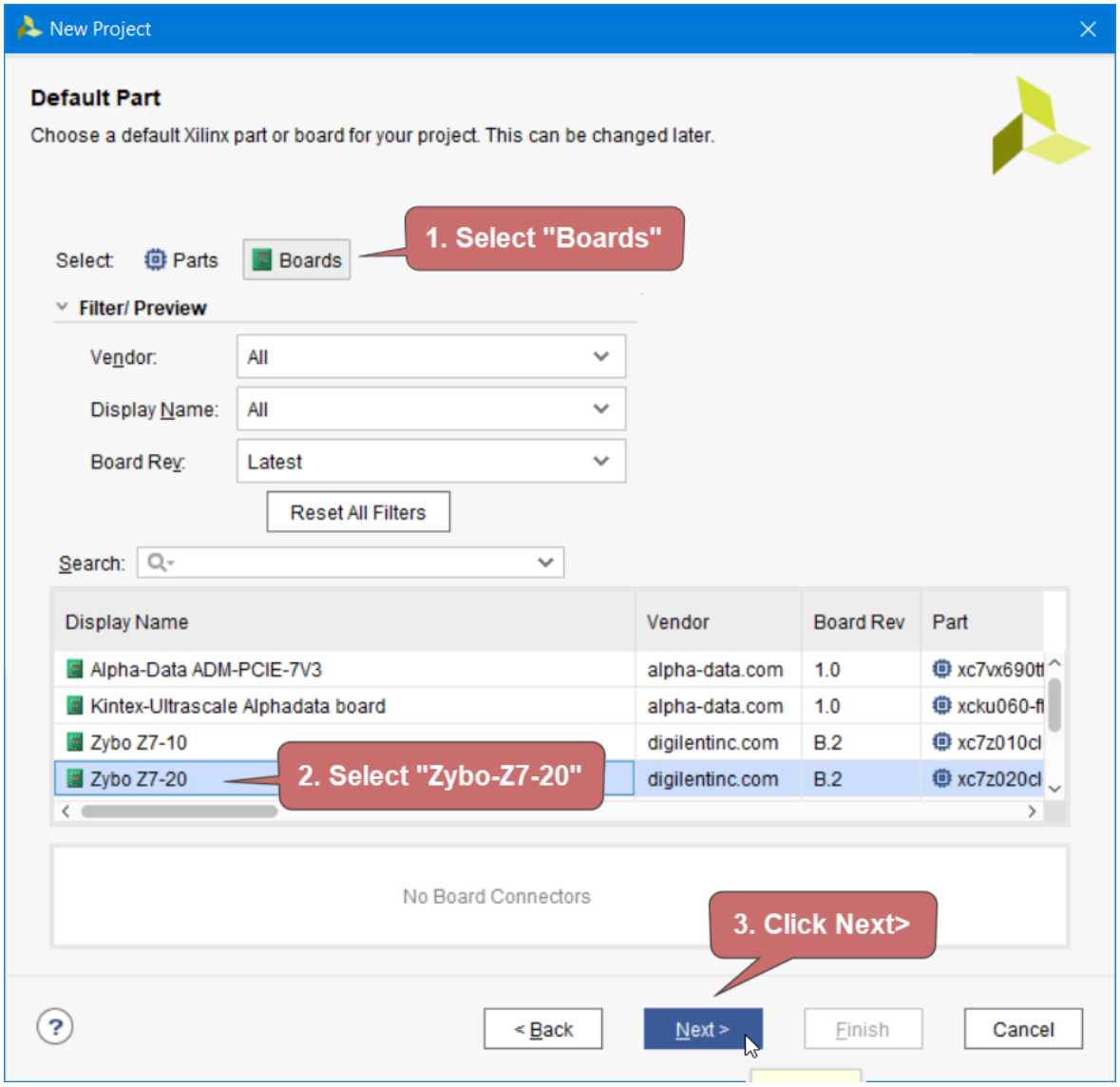


Figure 9. Choose the Digilent Zybo-Z7-20

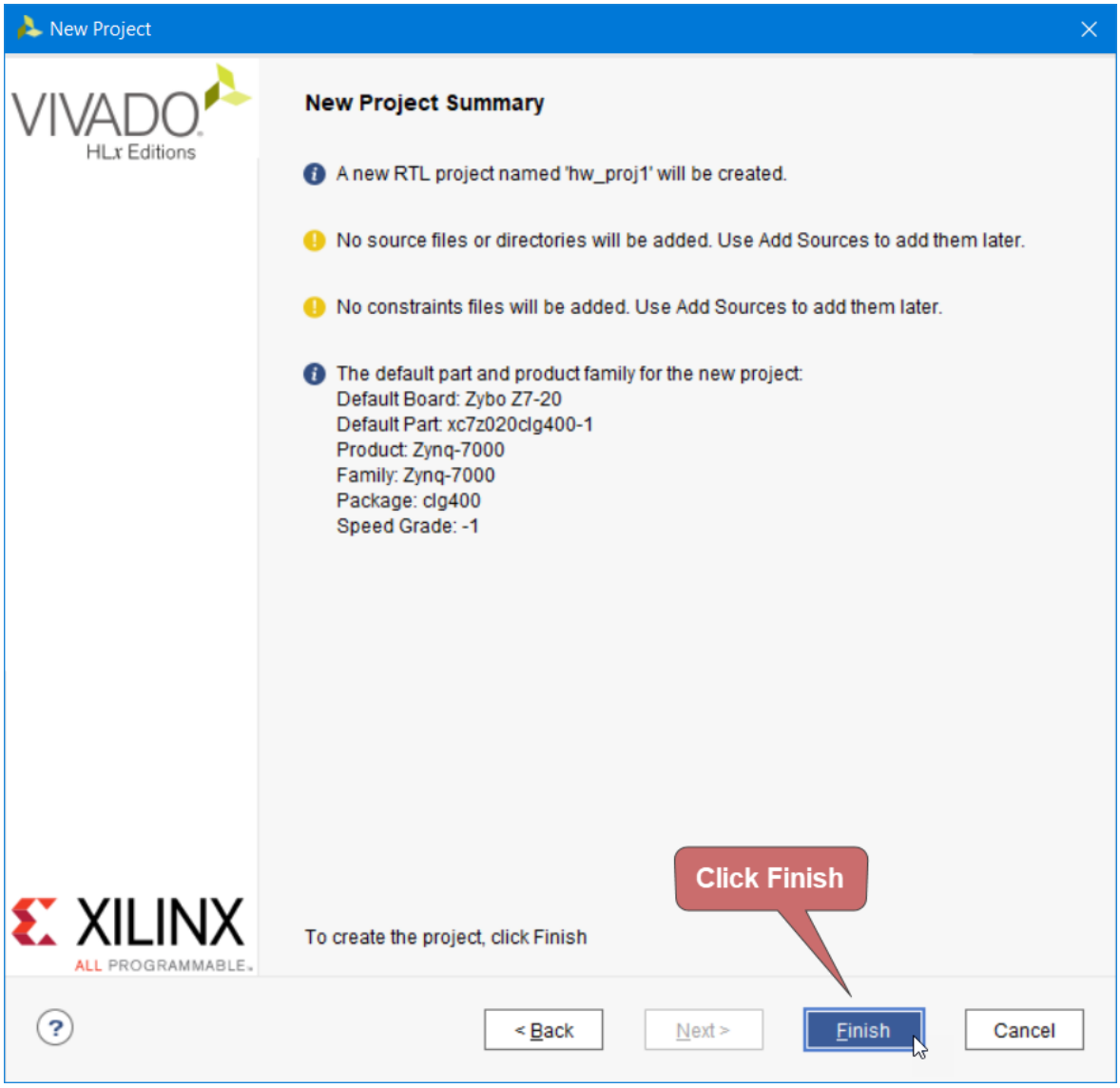


Figure 10. Click Finish.

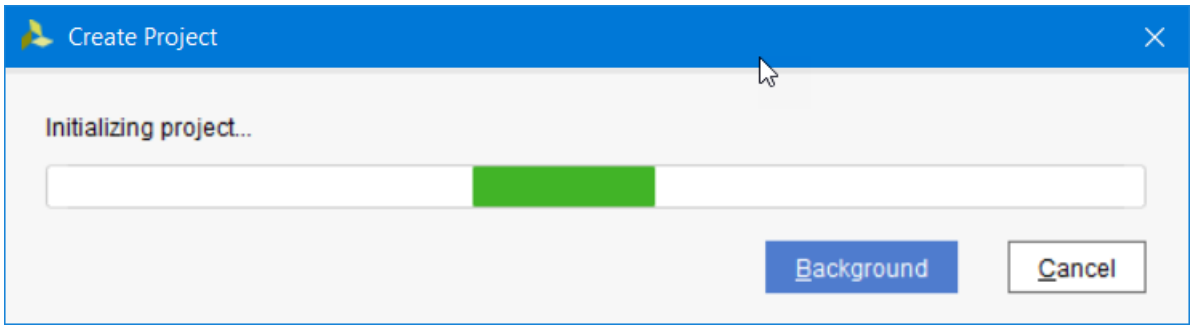


Figure 11. The project will be created...

2.2 Design Entry using IP Integrator

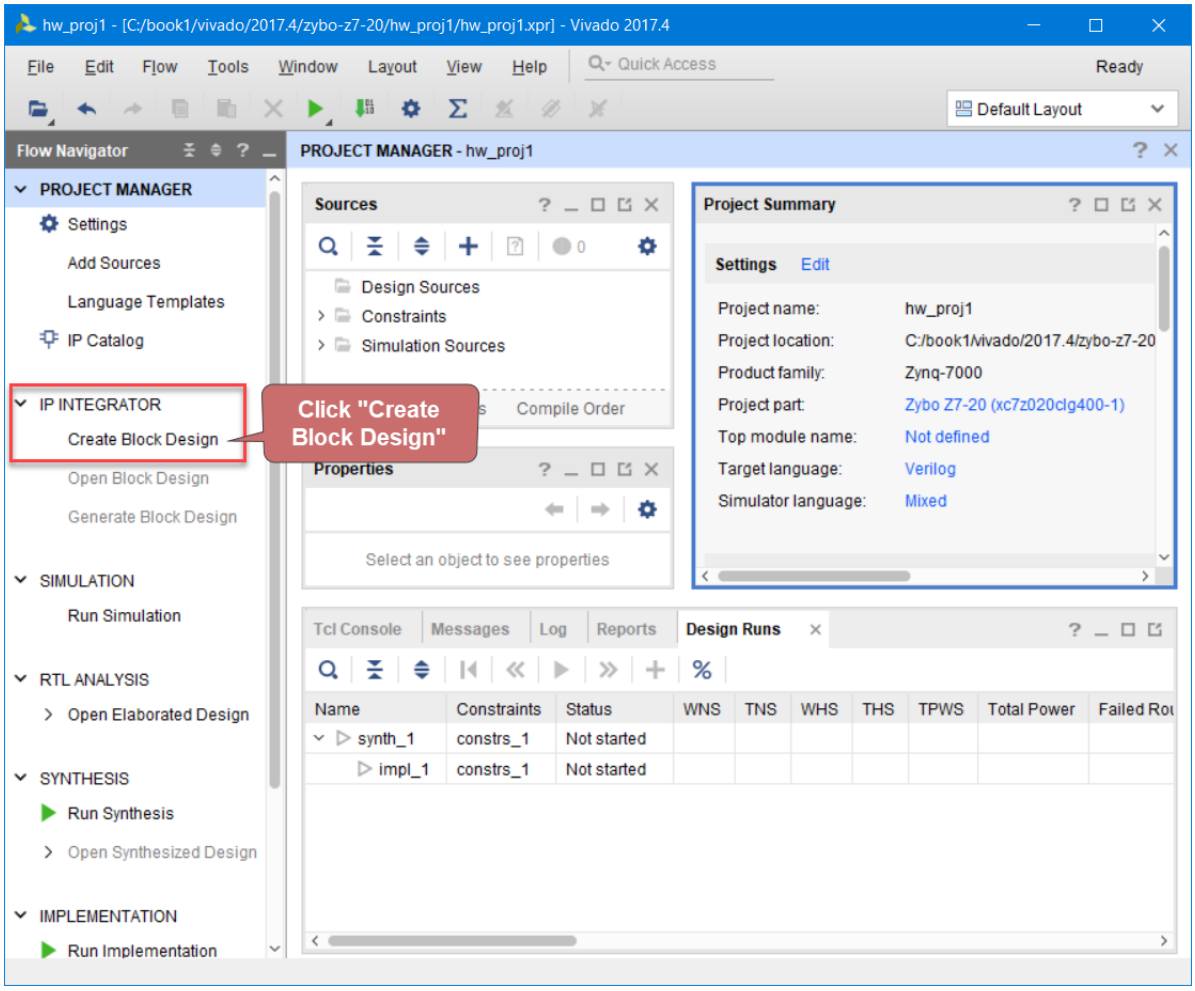


Figure 12. In the Flow Navigator, create the block design canvas.

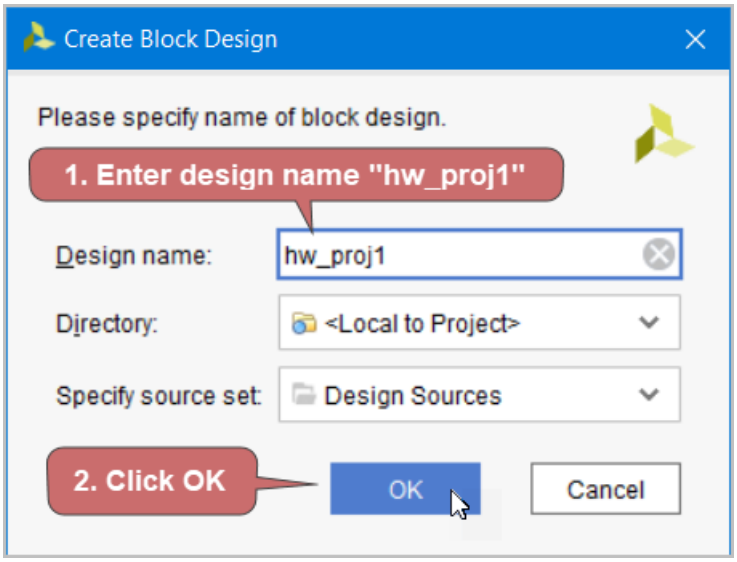


Figure 13. Enter the block design name.

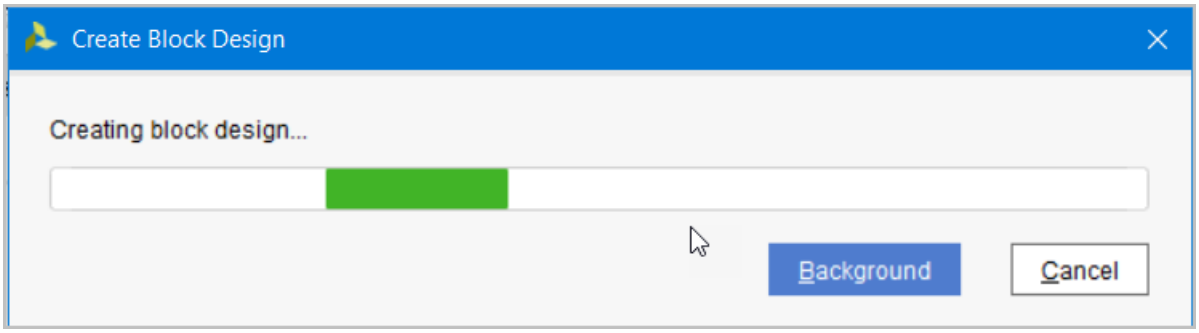


Figure 14. The block design will be created.

2.2.1 Add Zynq-7000 Processing System IP

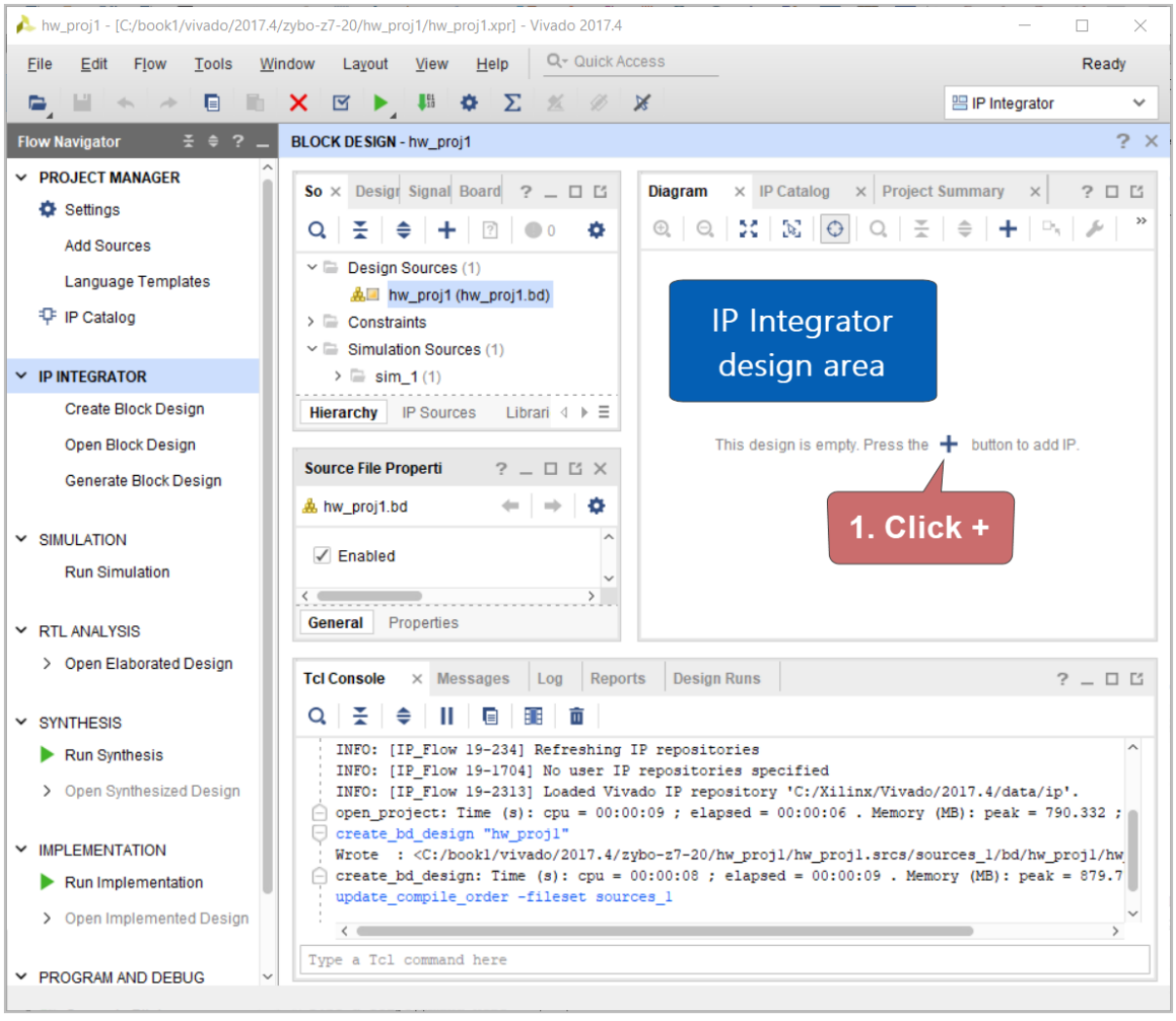


Figure 15. In the design area, click on the "+" icon, or select "CTRL + I"

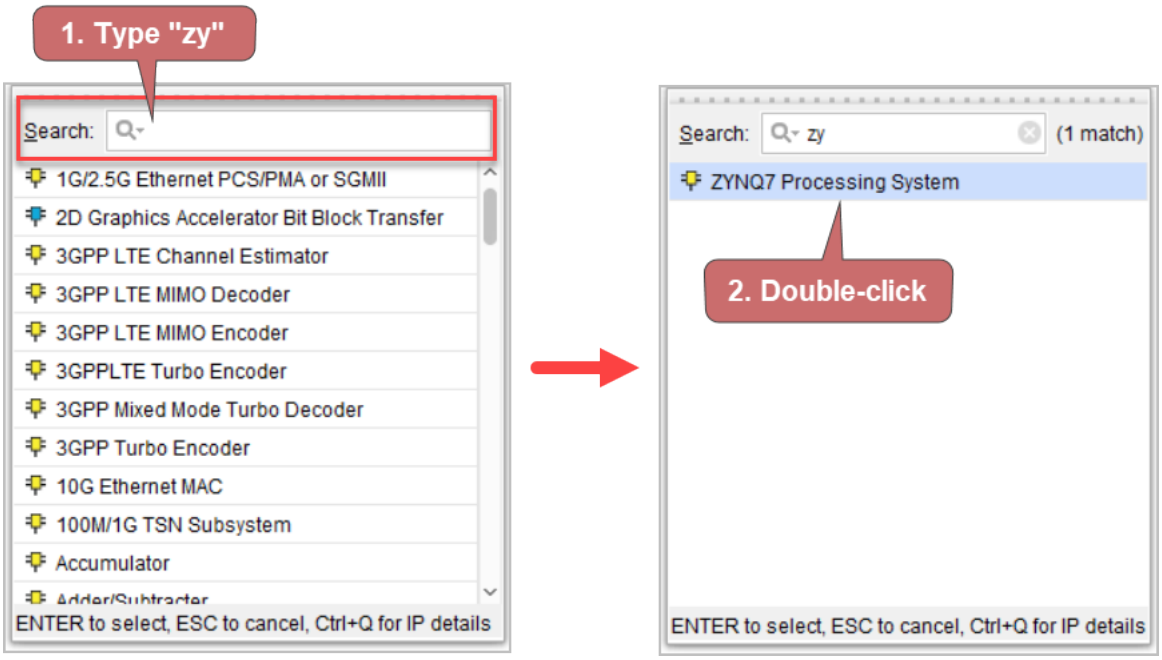


Figure 16. Find the Zynq-7000 Processing System IP

2.2.2 Generate Default Board Connections

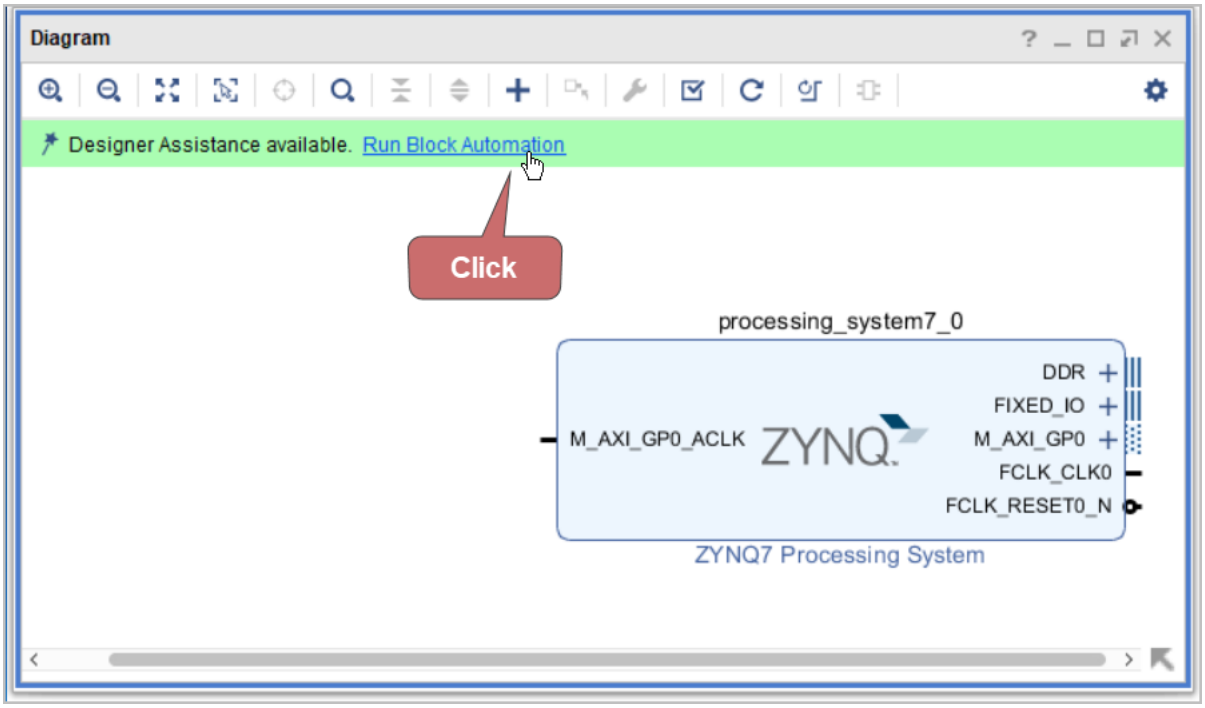


Figure 17. Click on the Run Block Automation option

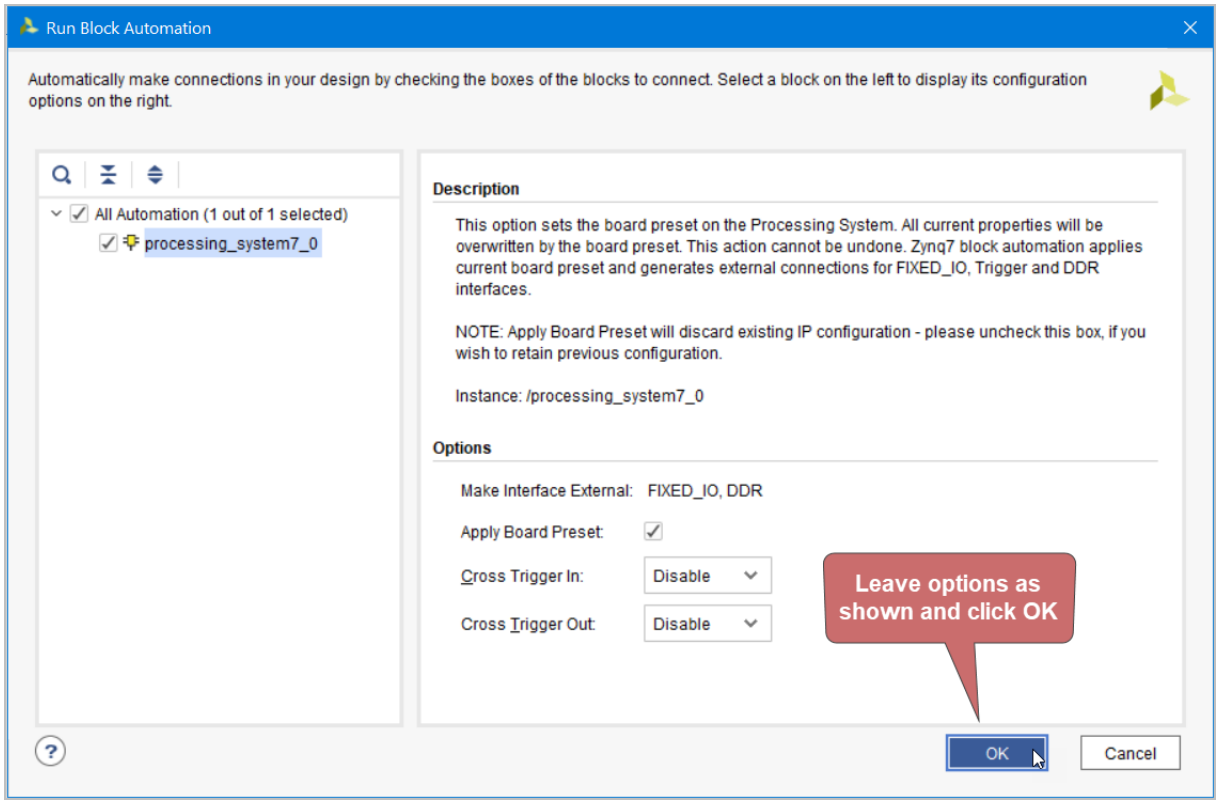


Figure 18. Select the default options and click OK.

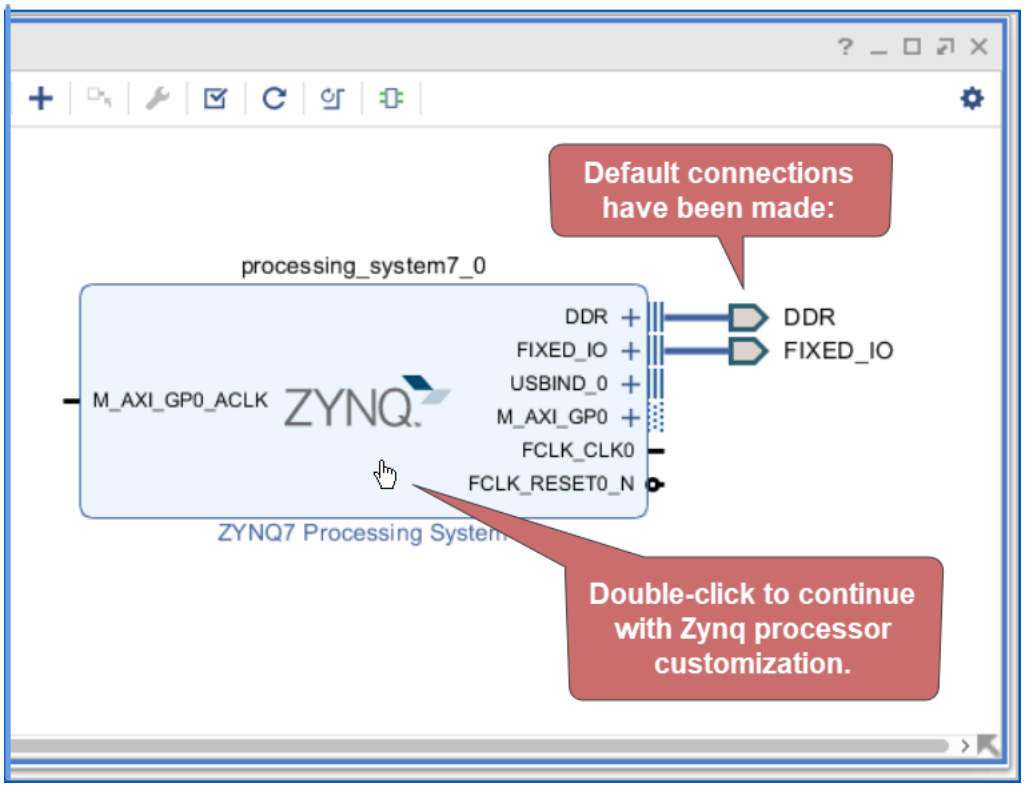


Figure 19. Double-click the Zynq IP to add more configuration options.

2.2.3 Customise the Processing System for our Project

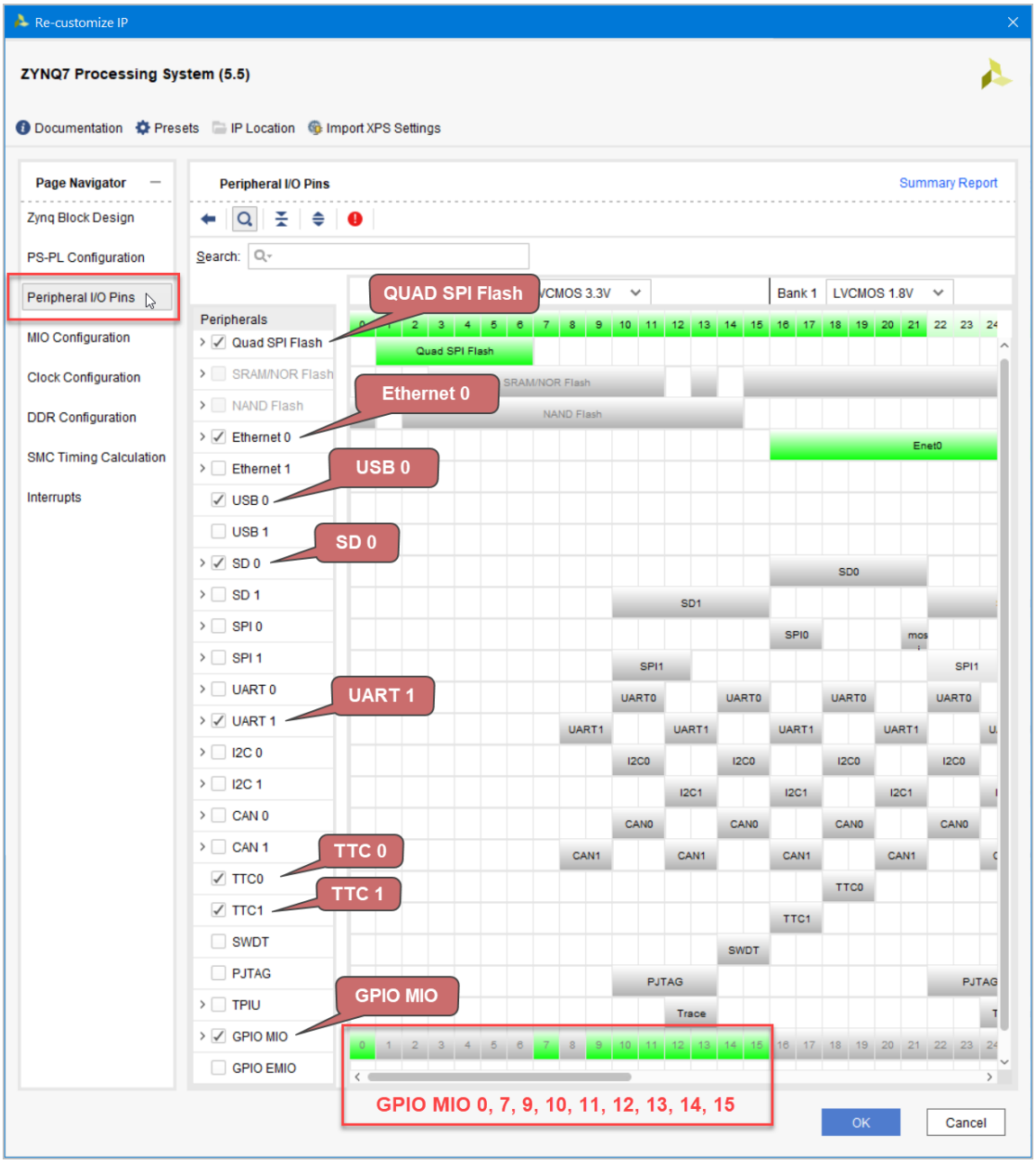


Figure 20. Peripheral I/O Pins (1). The settings should be as shown.

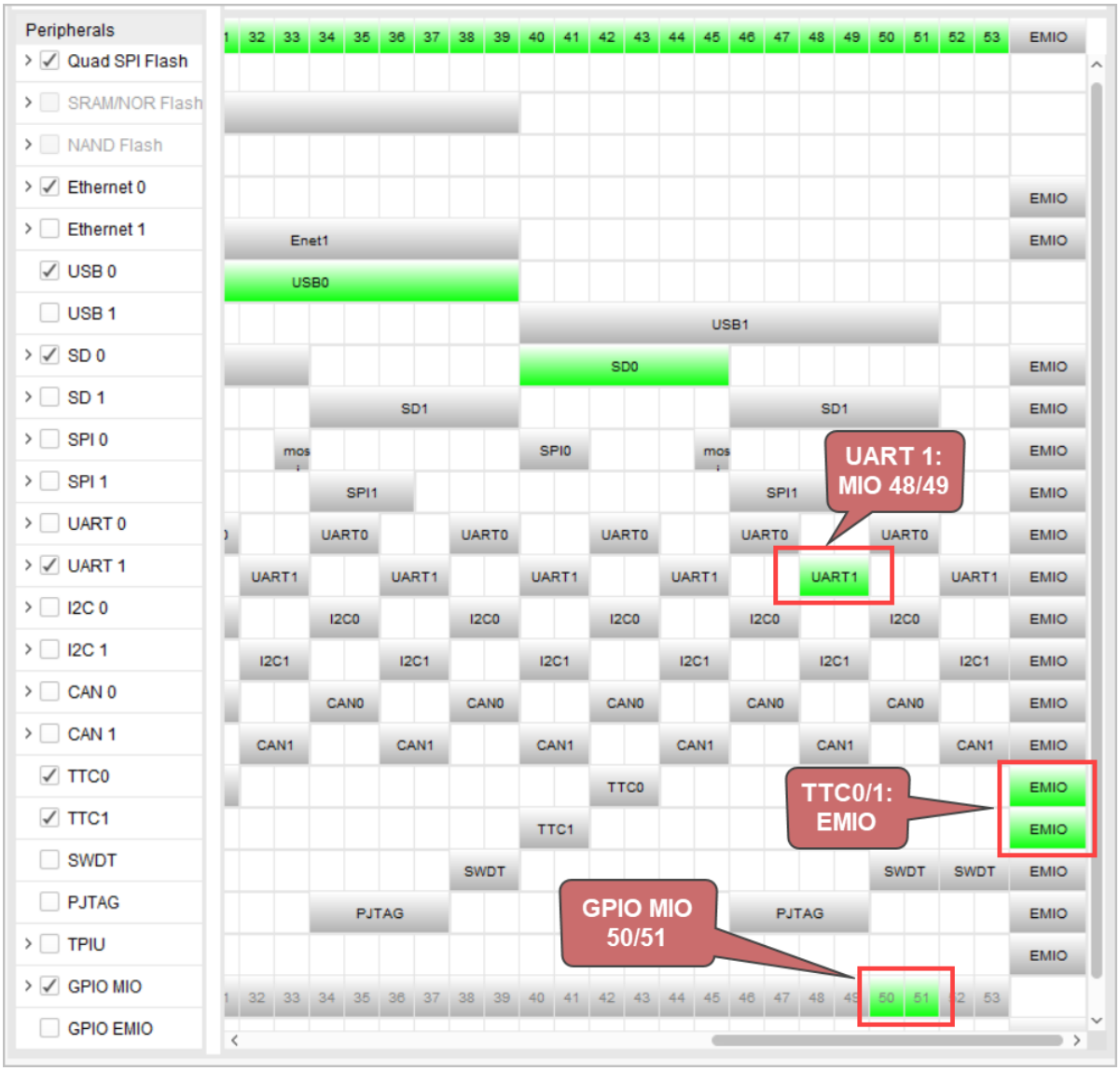


Figure 21. Enable TTC0 and TTC1 (the other settings should already be as shown).

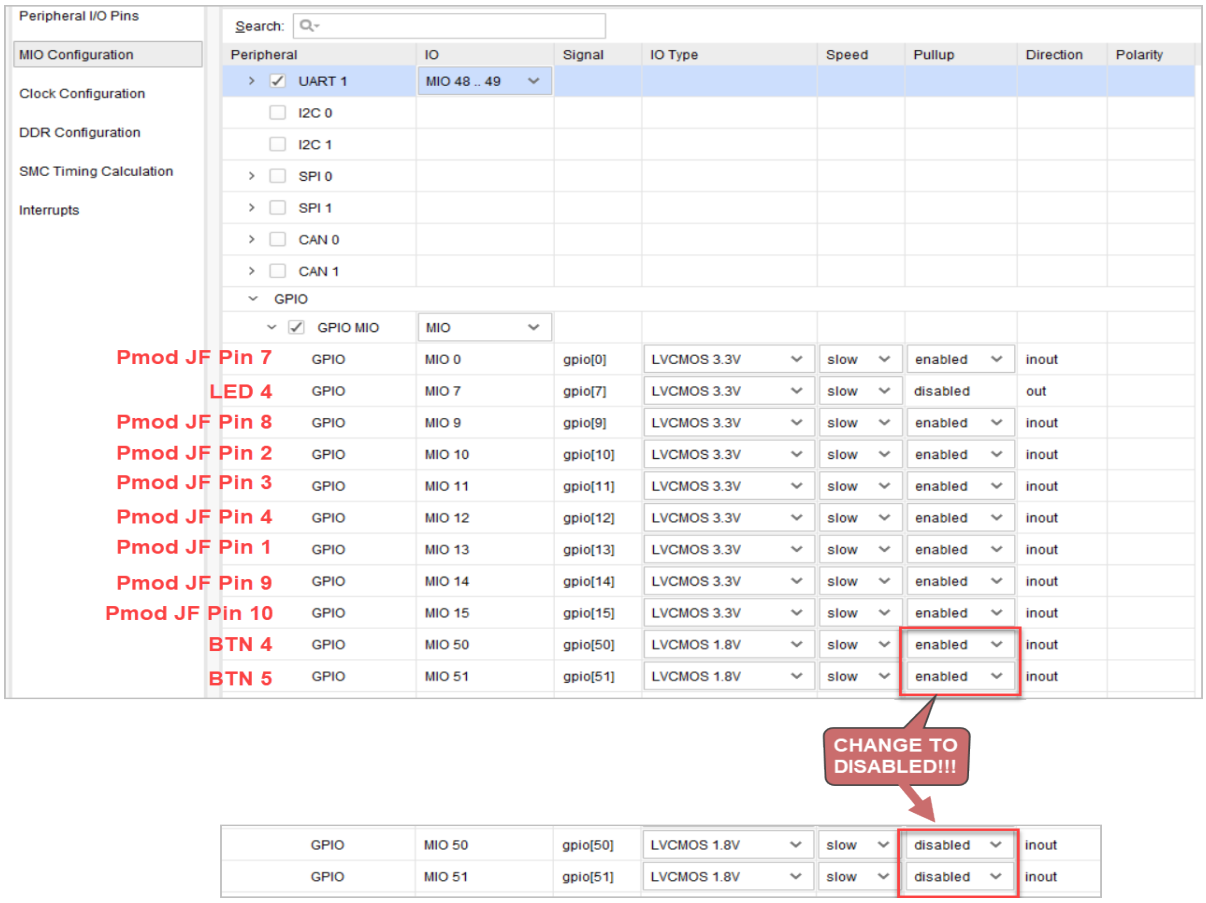


Figure 22. In the MIO Configuration Tab, the pull-ups for MIO50/MIO51 (BTN4/BTN5) can be disabled (if not already configured in the way).

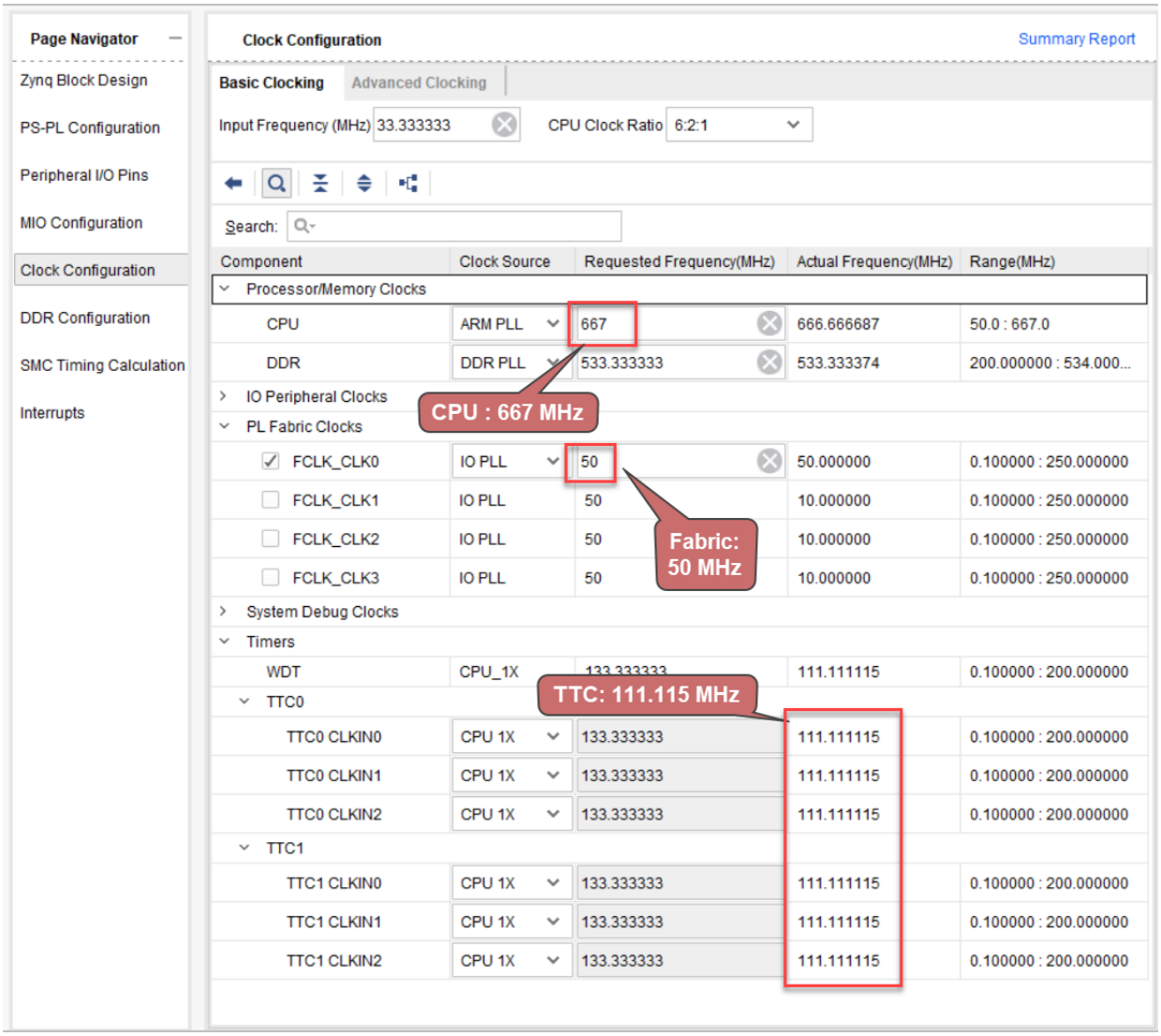


Figure 23. In the Clock Configuration Tab, verify that the clock settings are as shown.

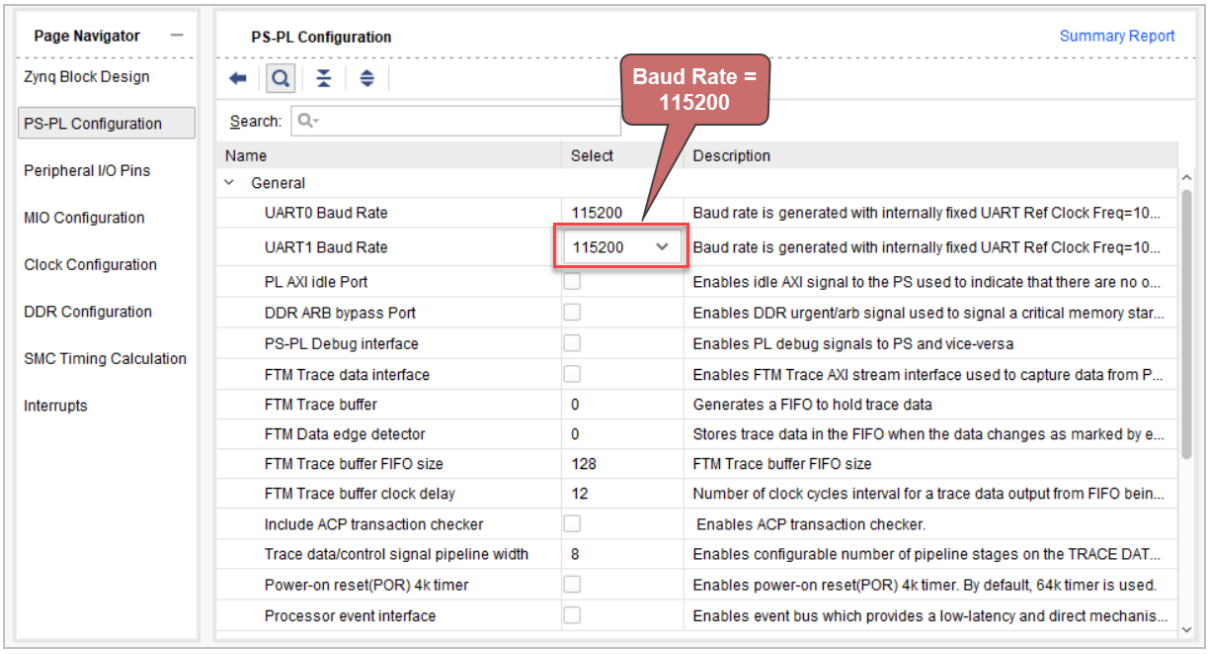


Figure 24. Check that the UART Baud Rate is 115200 baud.

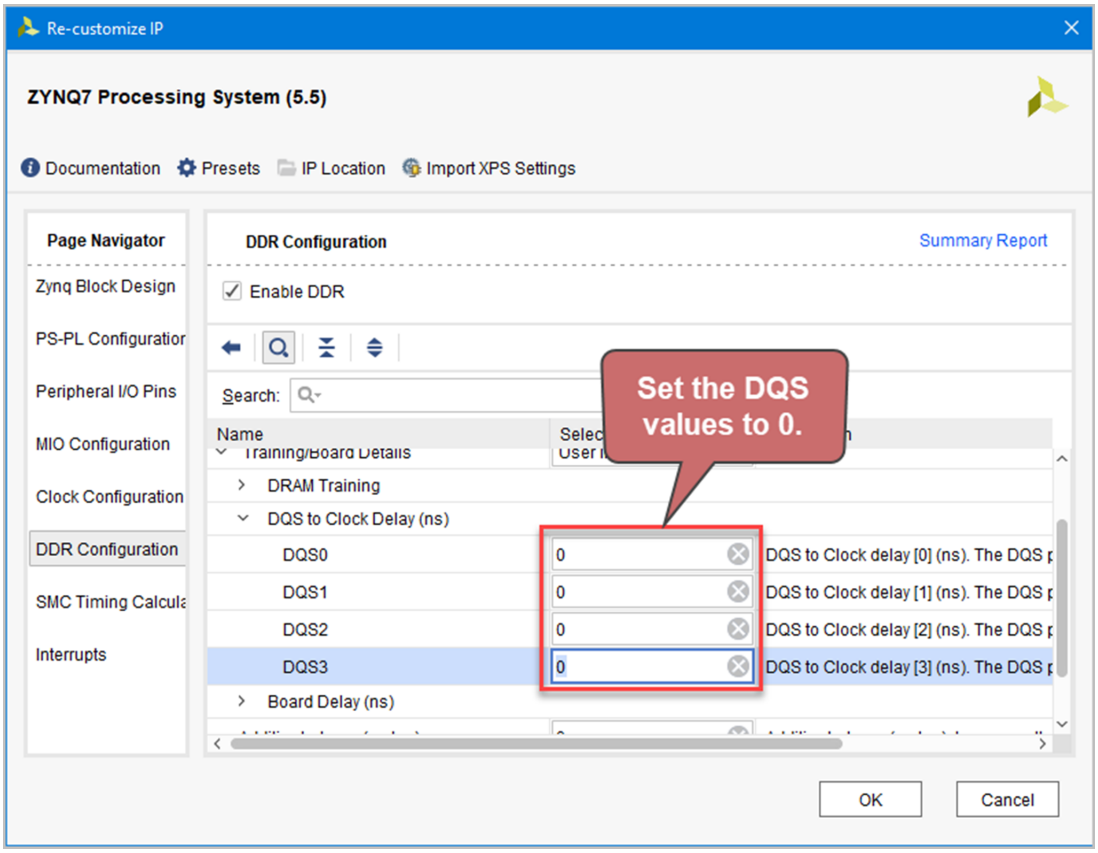


Figure 25. In the DDR Configuration tab, the DQS to Clock Delay settings can all be changed to 0.

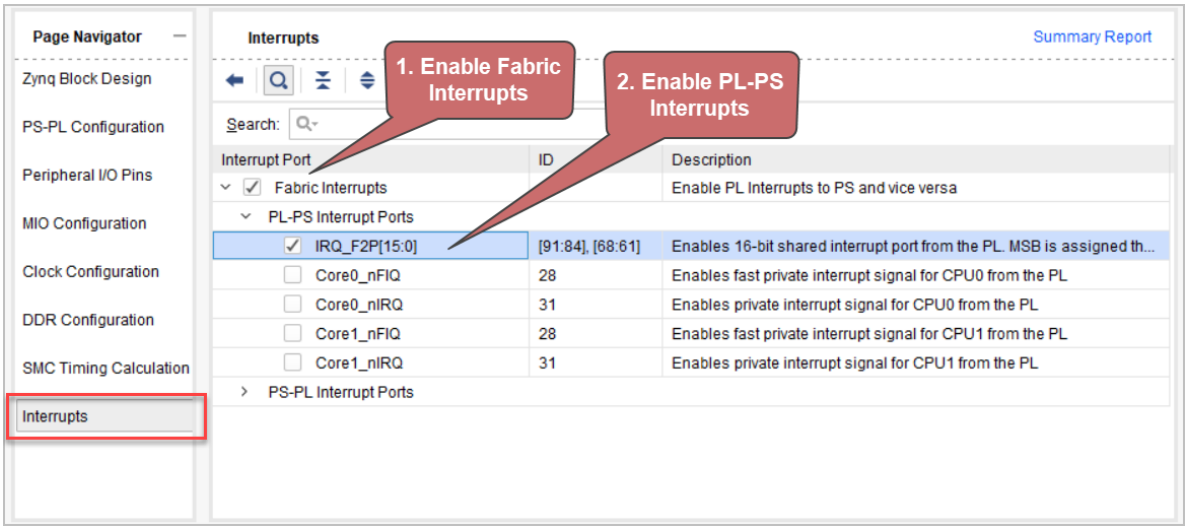


Figure 26. In the Interrupts tab, enable the IRQ_F2P[15:0] interrupts.

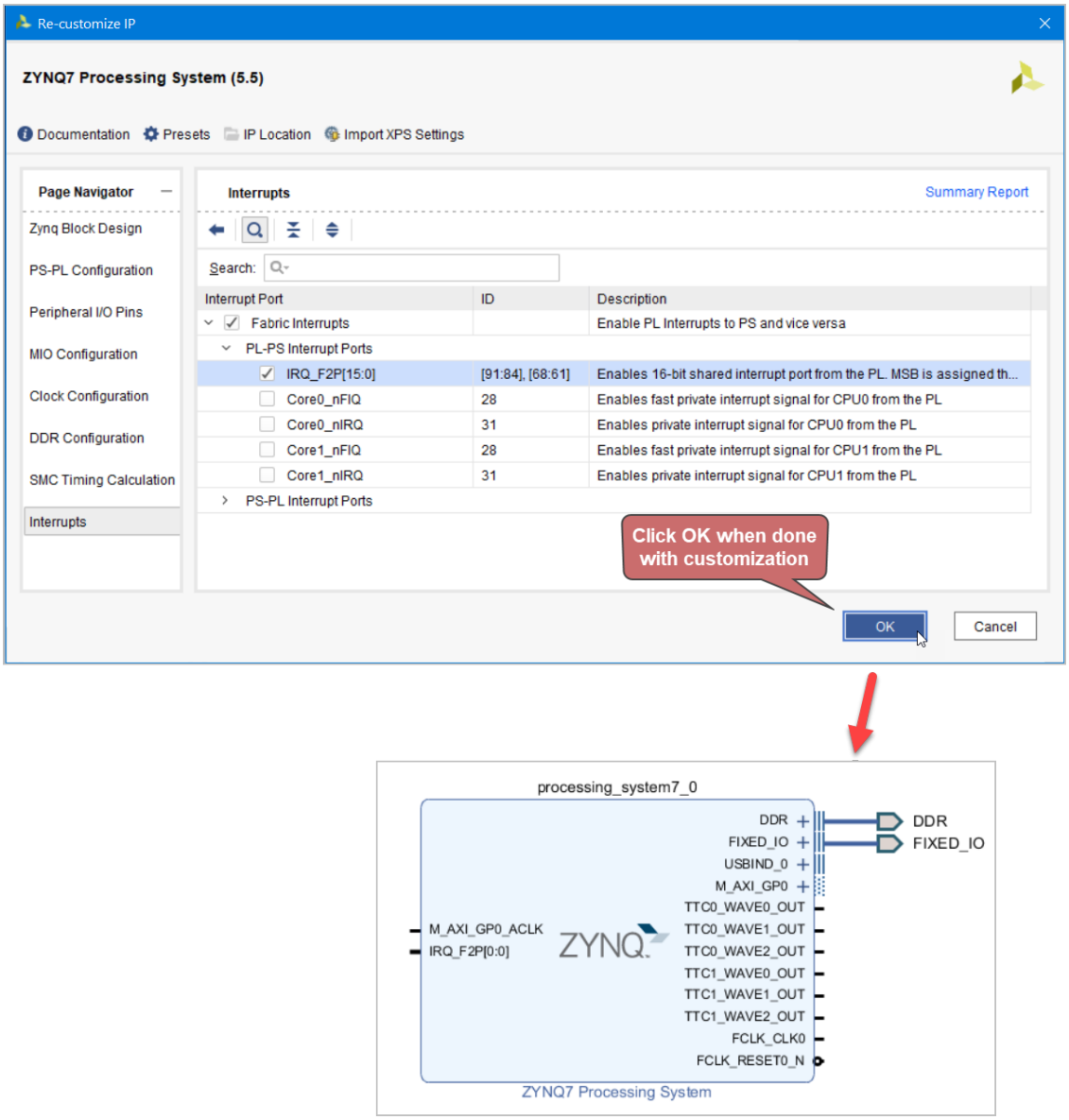


Figure 27. Click OK to apply customisation.

2.2.4 Add TTC Output Pins

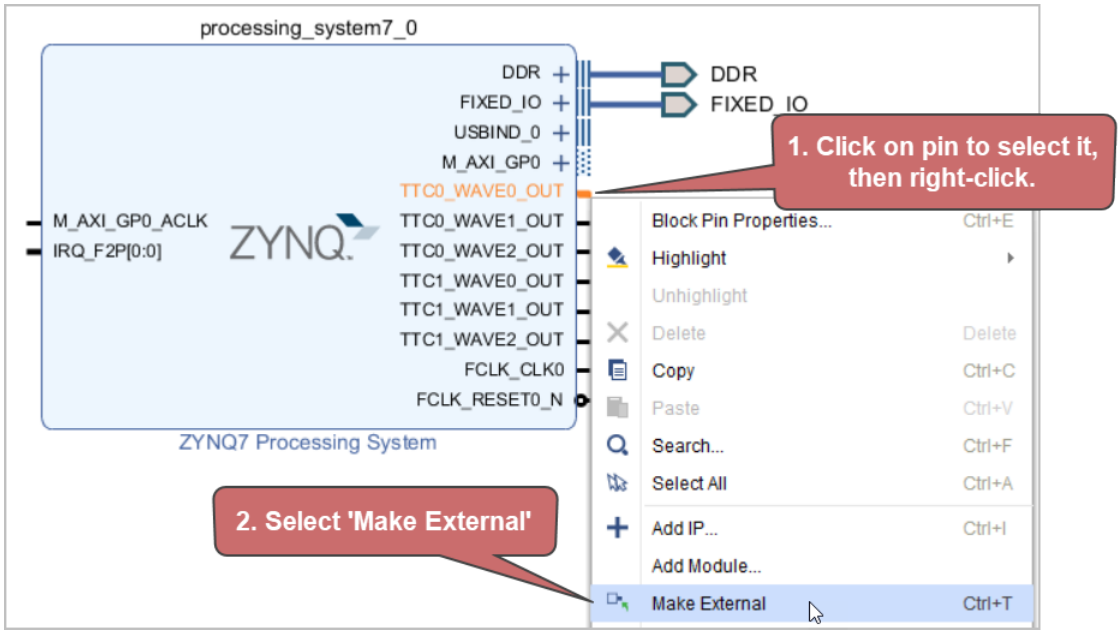


Figure 28. Add TTC pins by right-clicking and selecting "Make External"

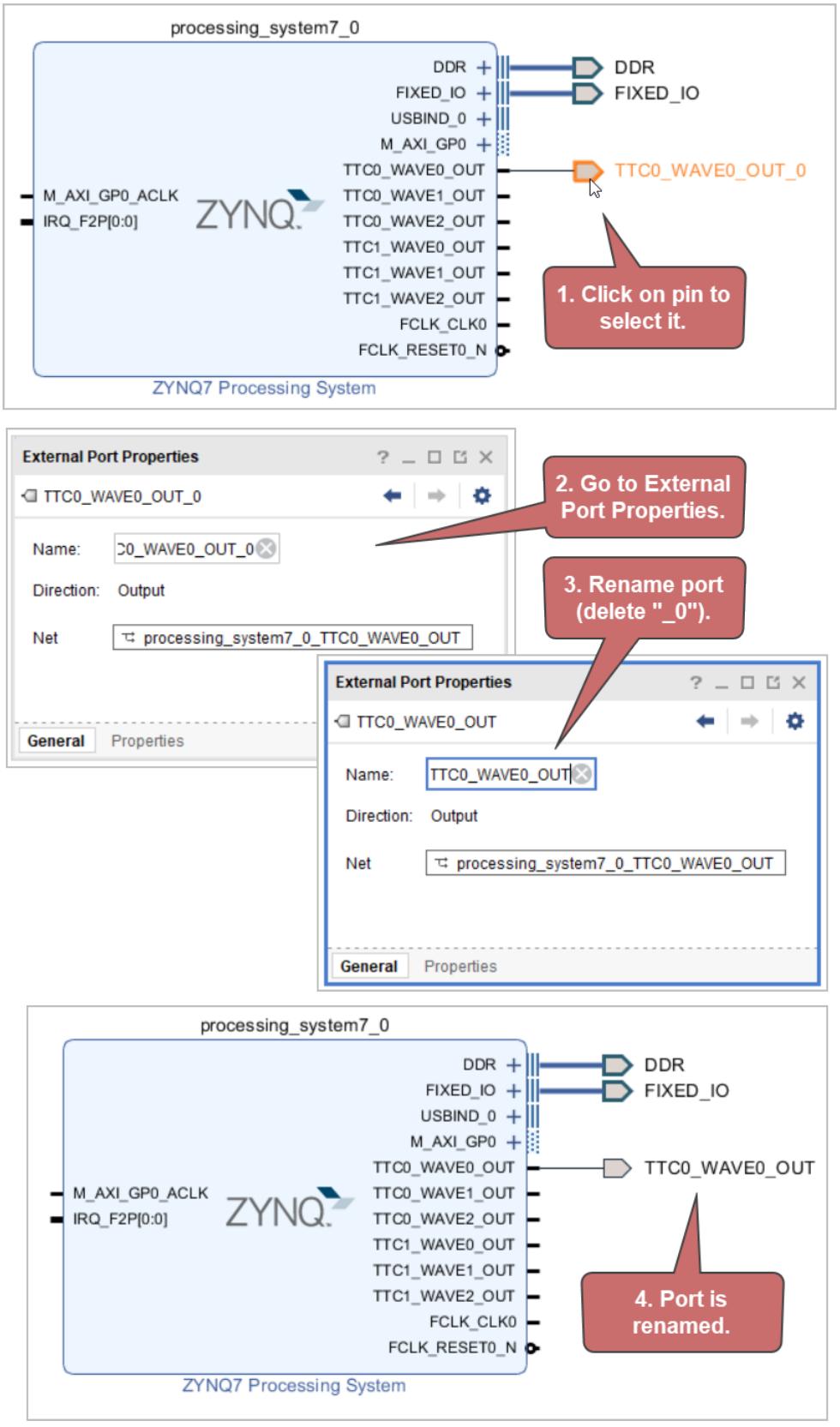


Figure 29. Rename the pins: delete "_0".

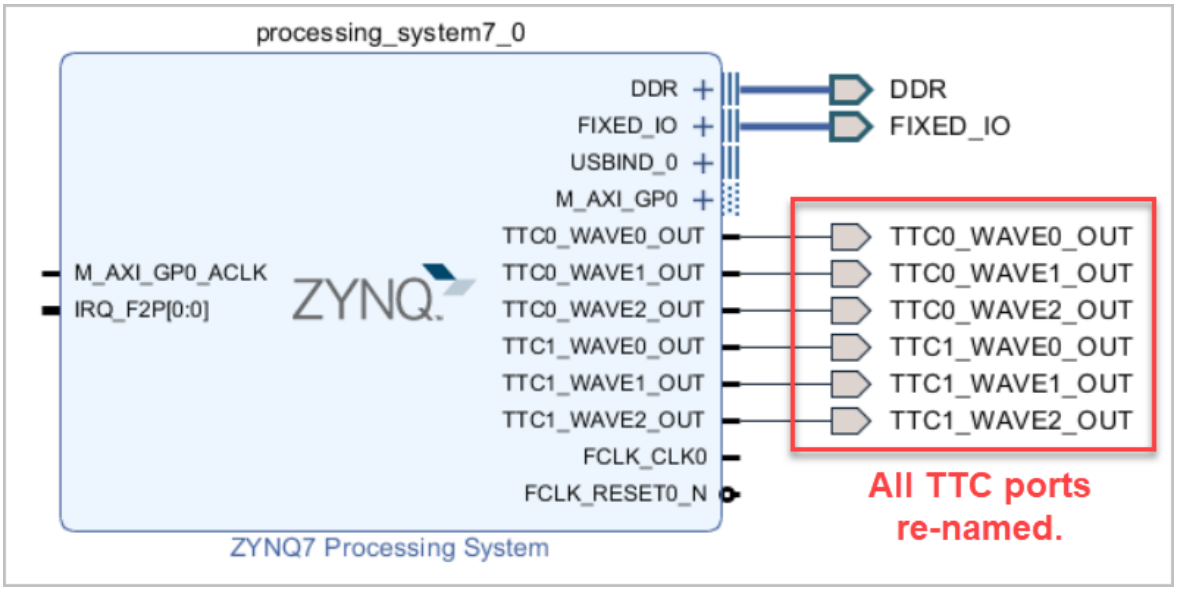


Figure 30. Repeat for all TTC pins

2.2.5 Add AXI GPIO IP

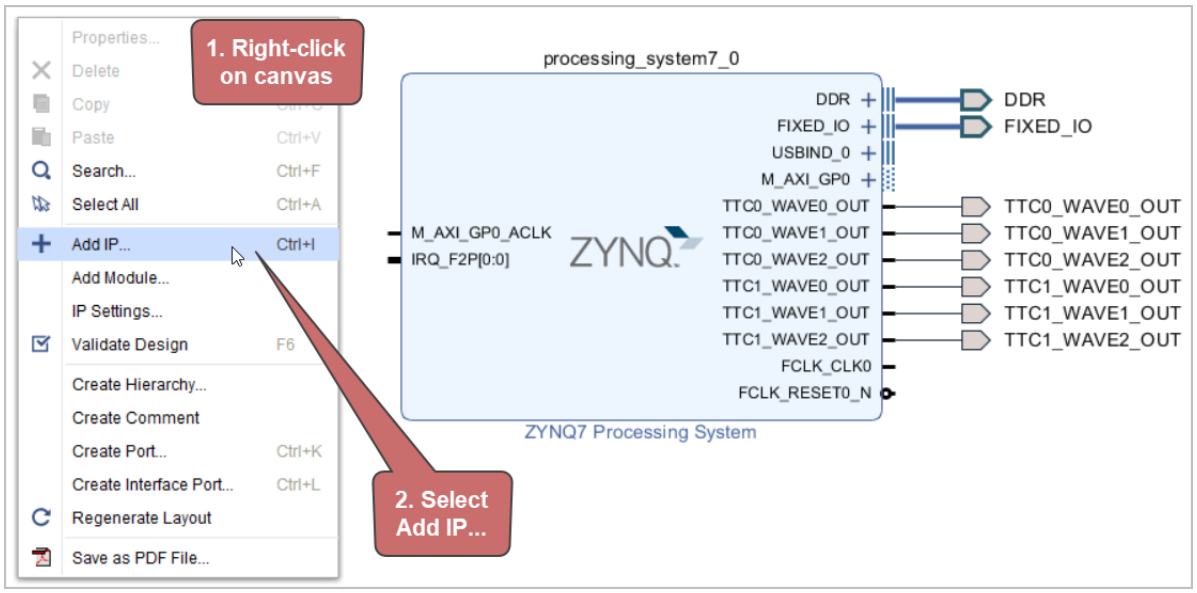


Figure 31. Add the AXI GPIO IP (1).

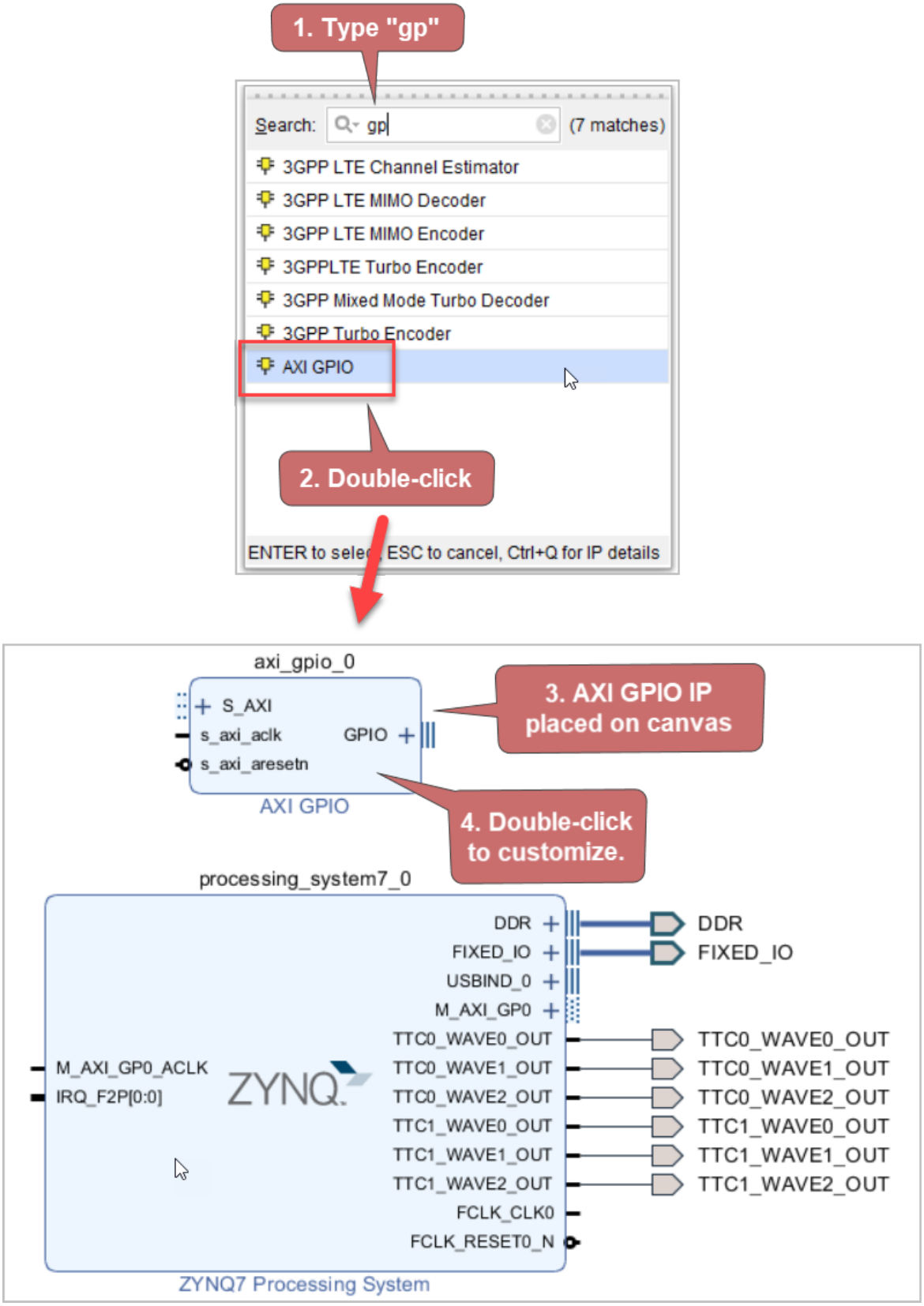


Figure 32. Add the AXI GPIO IP (2).

2.2.5.1 Customise the AXI GPIO IP

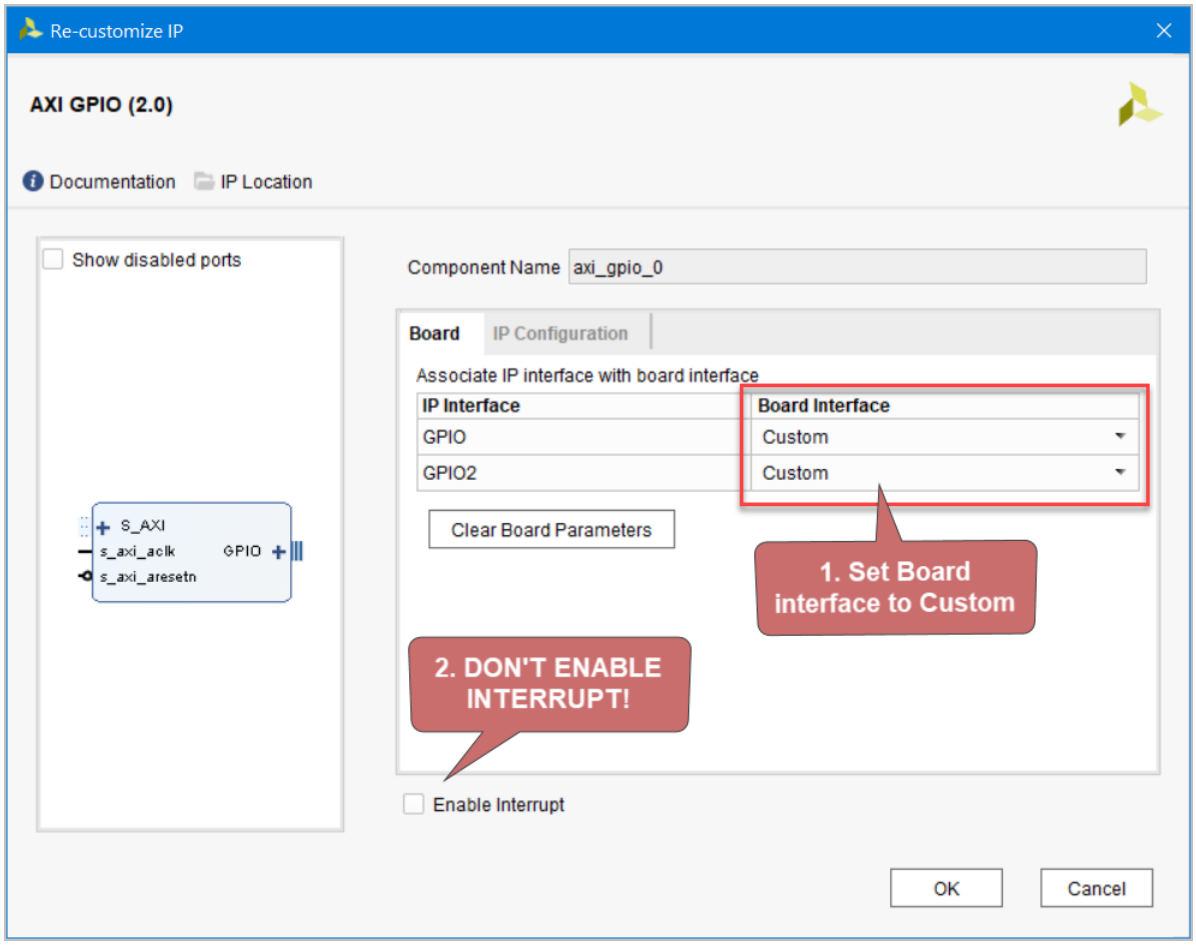


Figure 33. Set the GPIO Board options

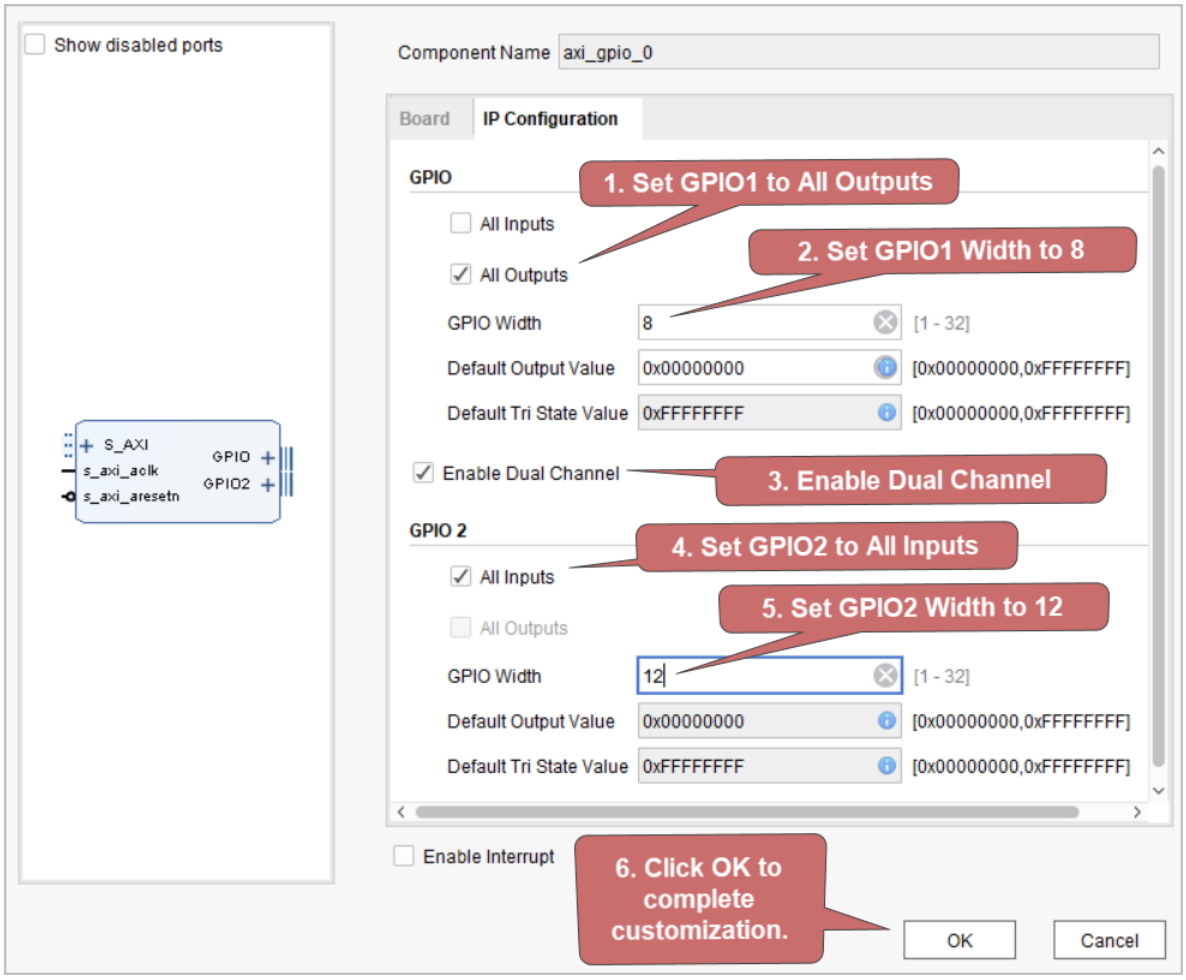


Figure 34. Set GPIO IP Configuration.

2.2.5.2 Add GPIO Input/Output Pins

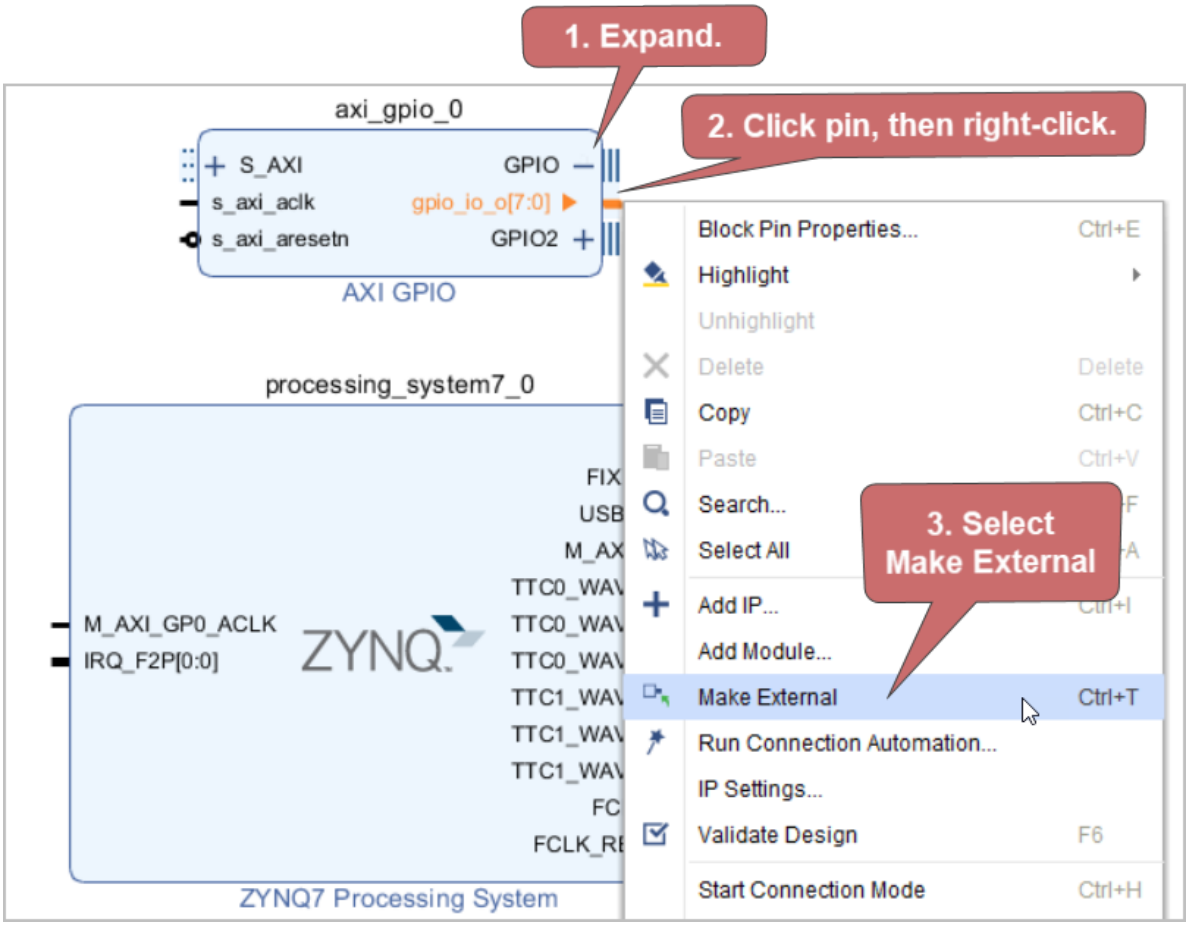


Figure 35. Add the output pins for channel 1.

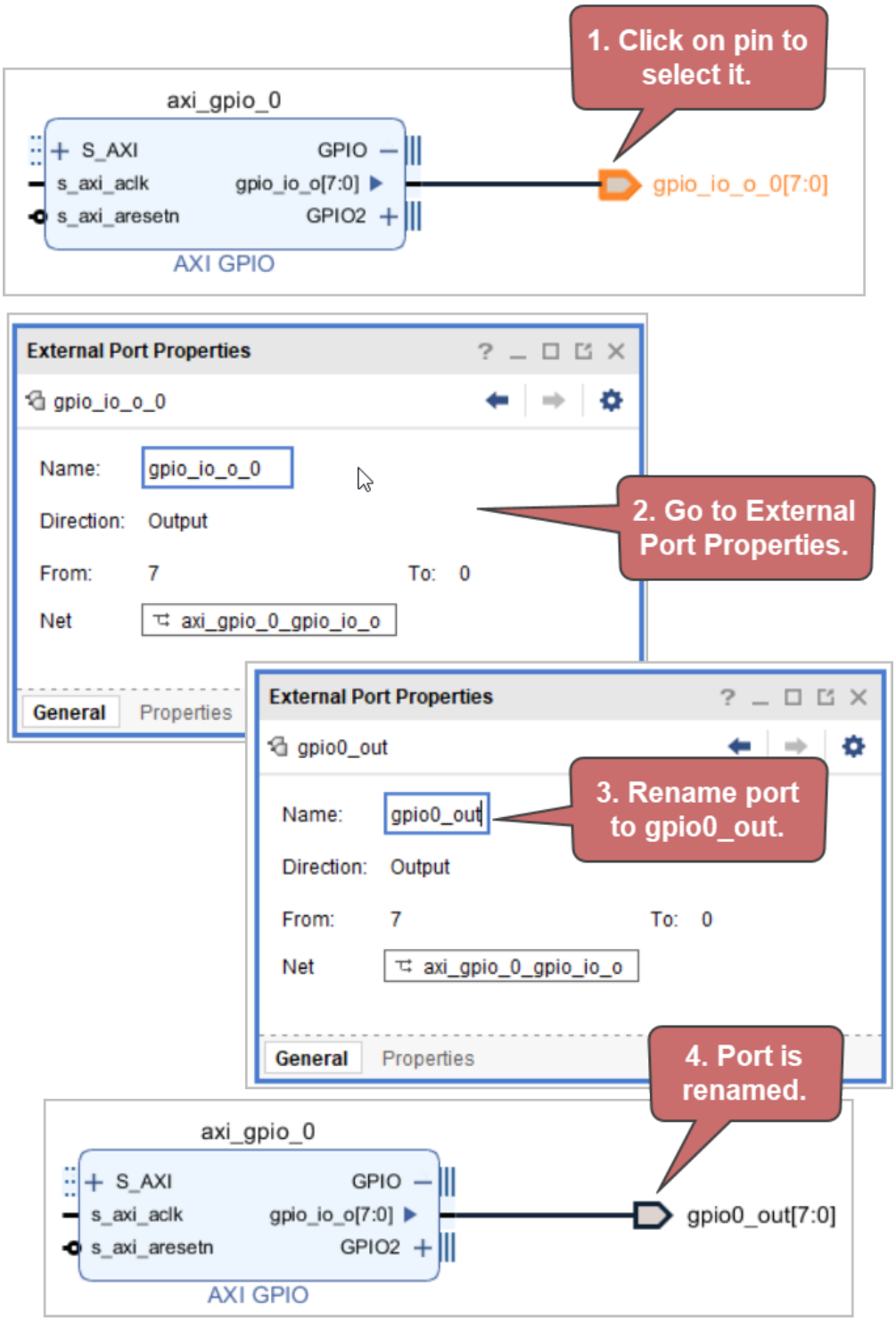


Figure 36. Rename the output pins.

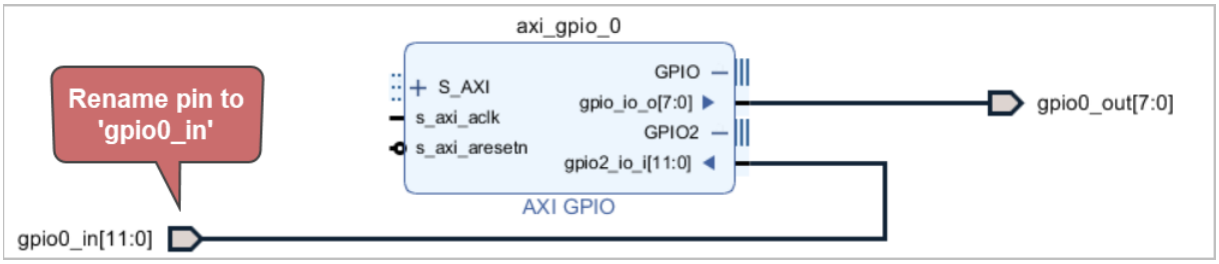


Figure 37. Repeat for the channel 2 input pins.

2.2.5.3 Run Connection Automation to Update Block Design

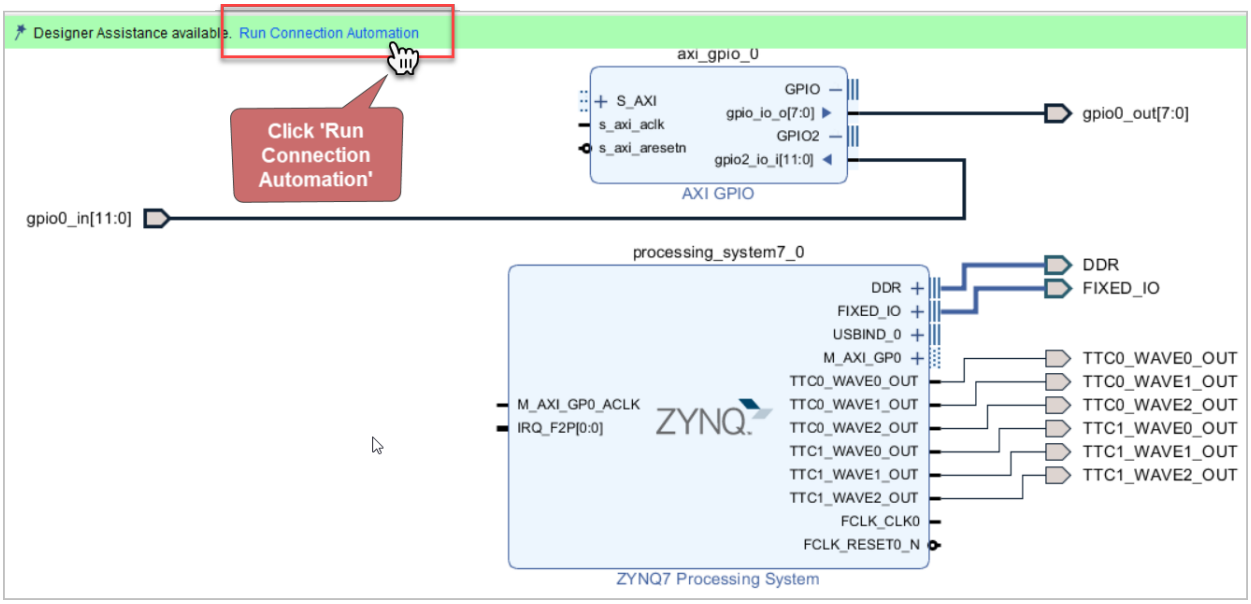


Figure 38. Run Connection Automation.

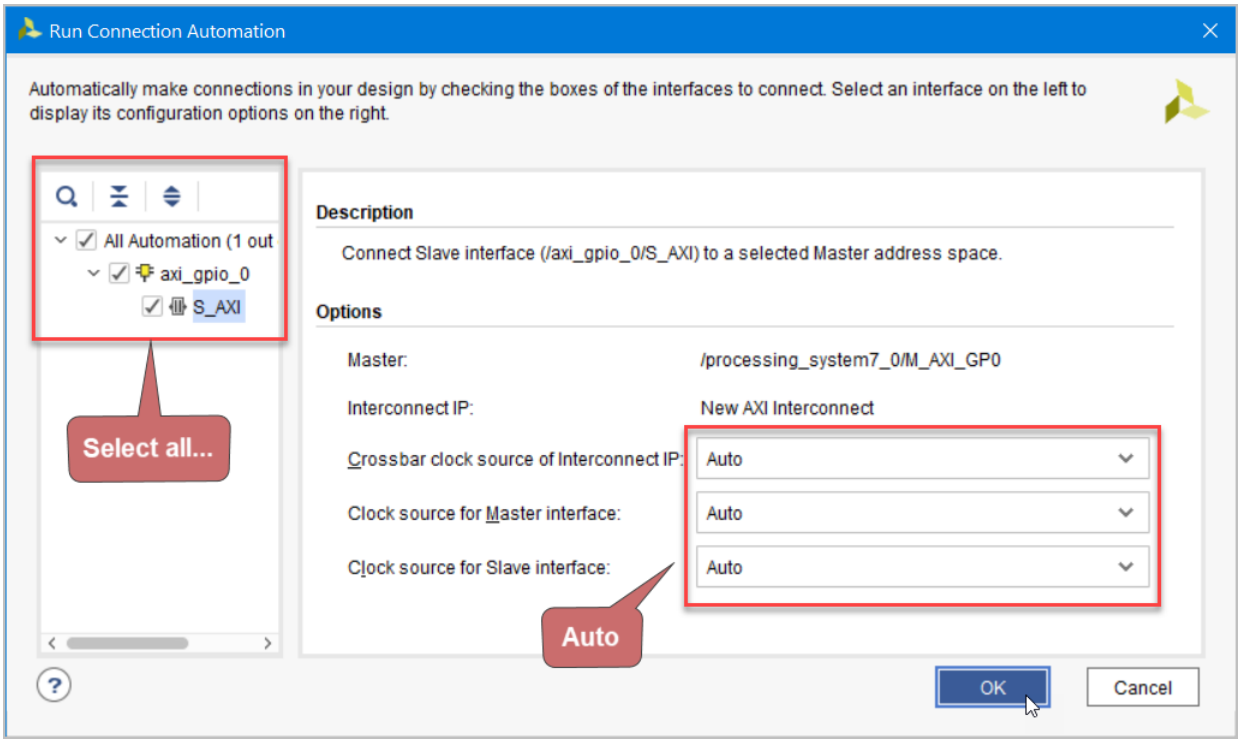


Figure 39. Ensure the options are set to Auto, and click OK.

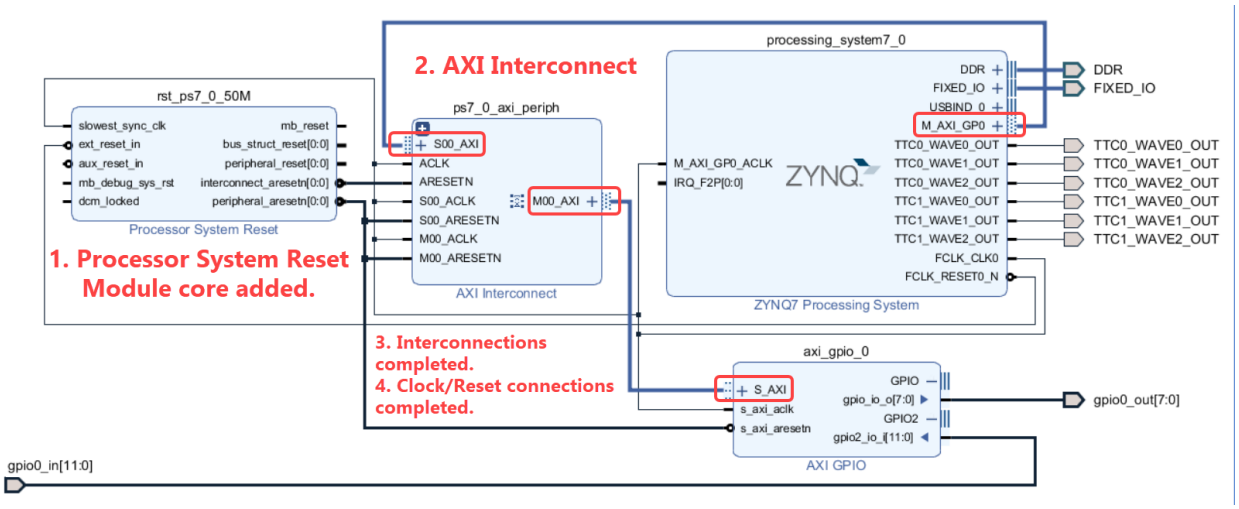


Figure 40. The updated block diagram is as shown.

2.2.5.4 Validate Design

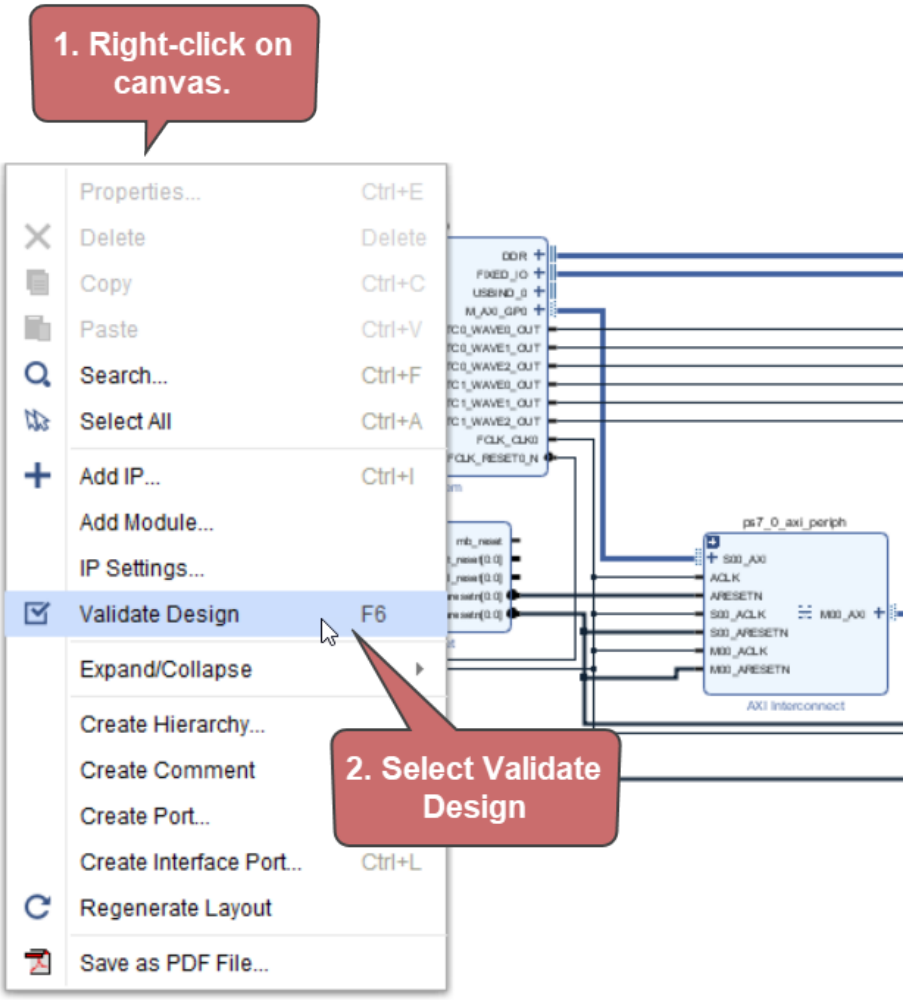


Figure 41. Validate the design.

Figure 42. Click OK to continue.

2.2.6 Add AXI Quad SPI IP

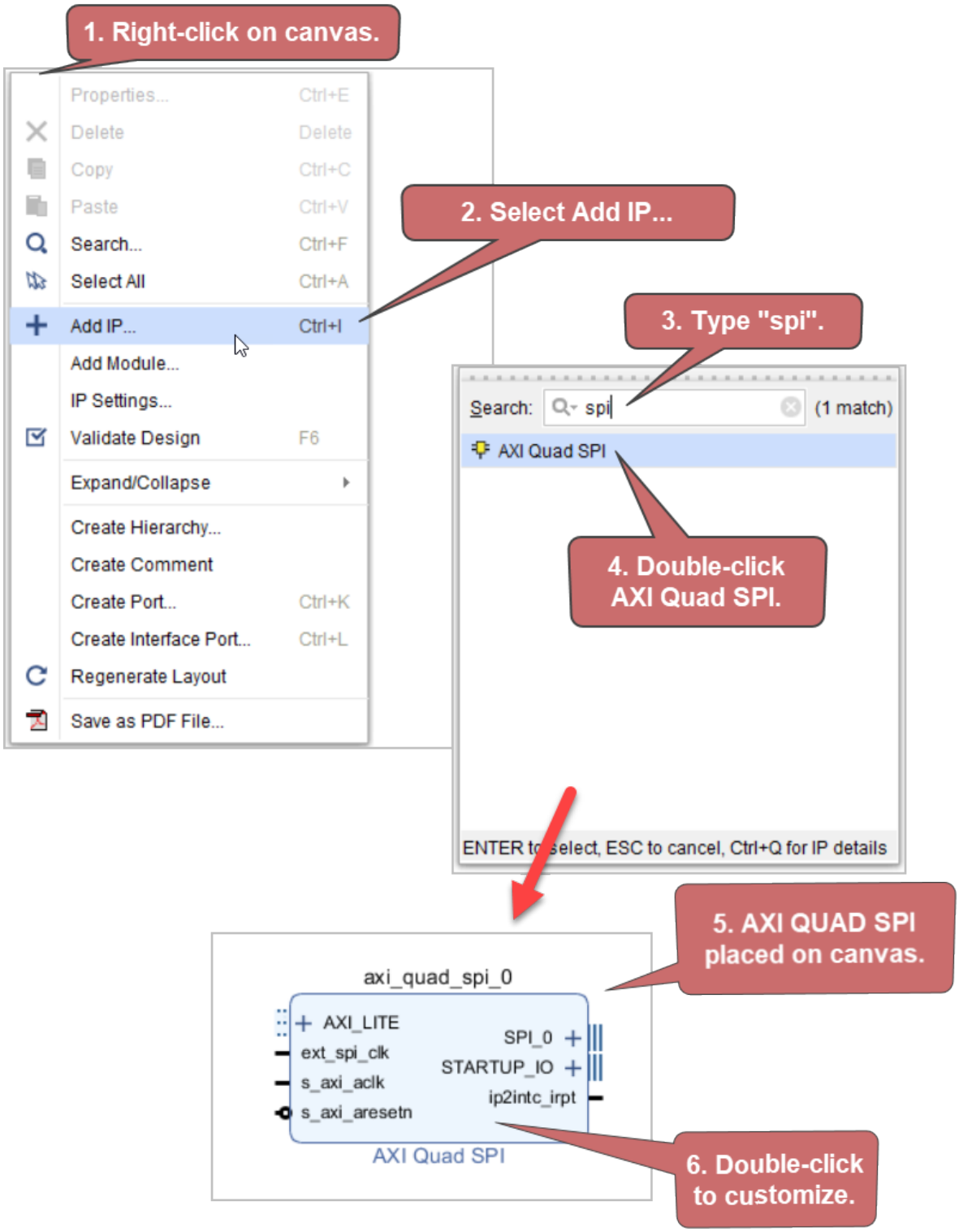


Figure 43. Add the AXI Quad SPI module.

2.2.6.1 Customise AXI Quad SPI IP

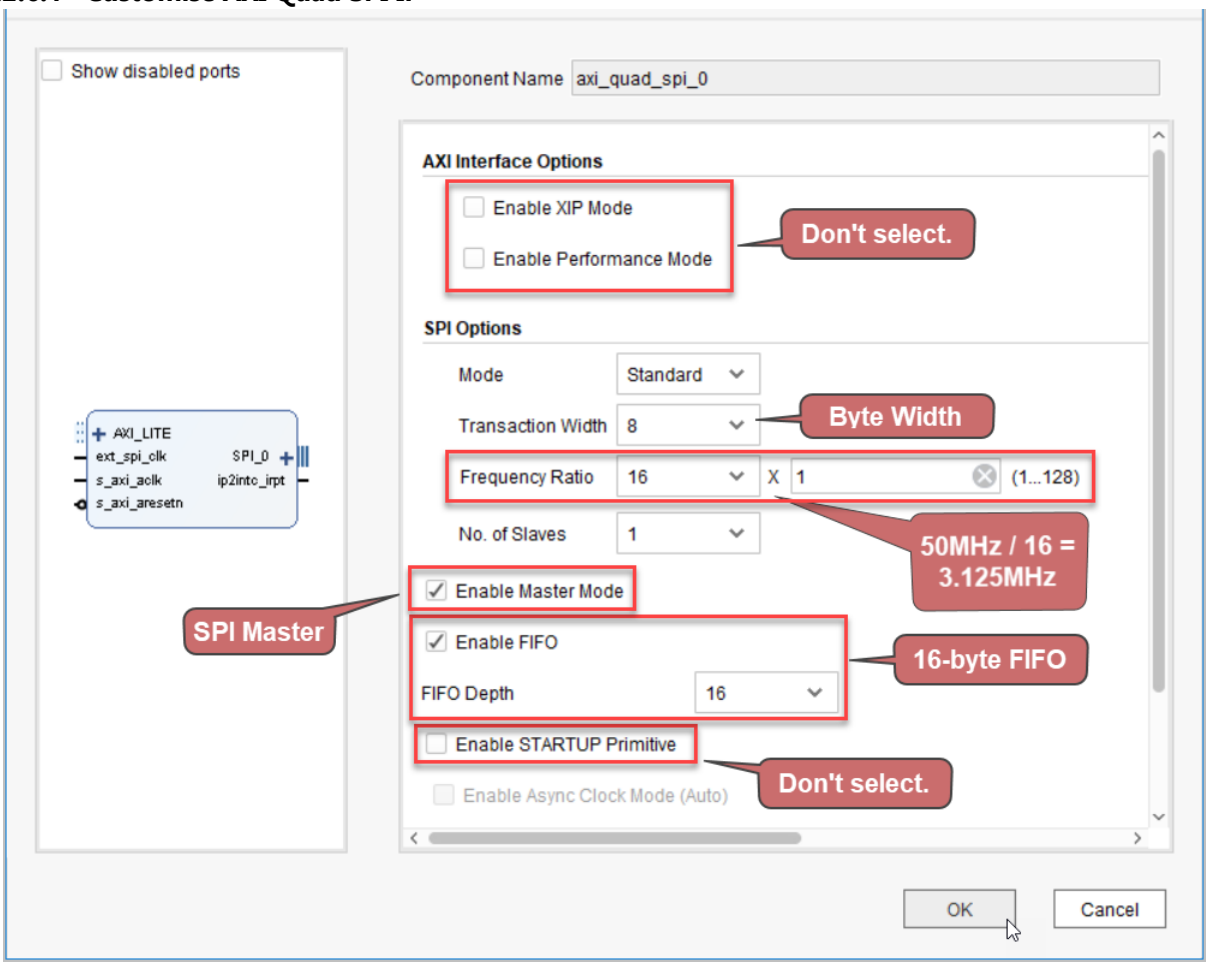


Figure 44. Customise the AXI Quad SPI as shown and click OK.

2.2.6.2 Add SPI Input/Output Pins

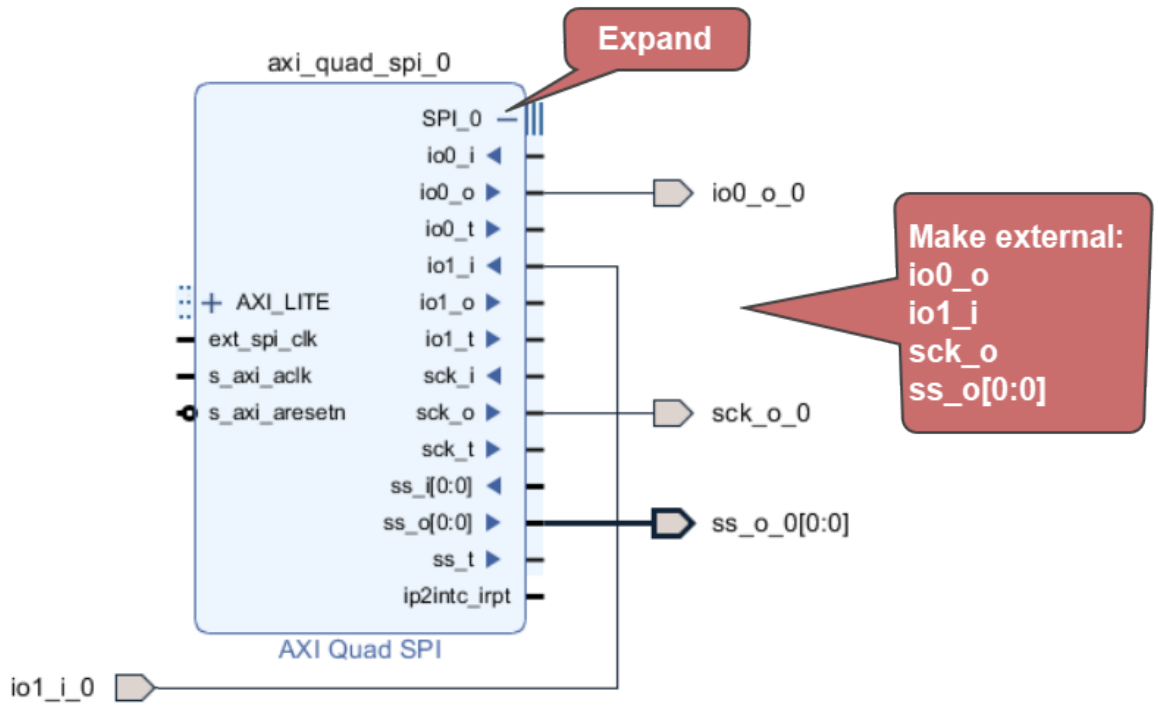


Figure 45. Add individual SPI pins

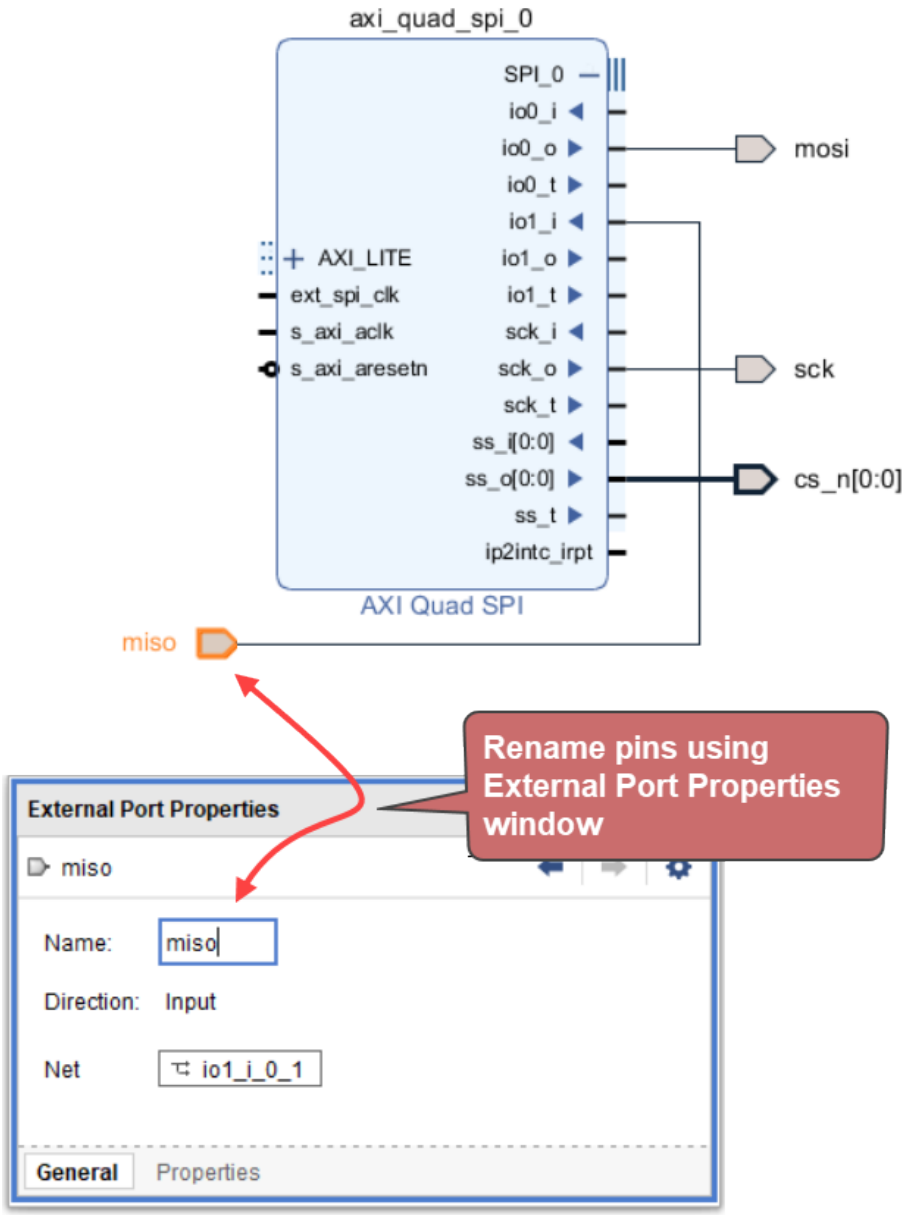


Figure 46. Rename the pins to match standard SPI names.

2.2.6.3 Run Connection Automation to Update Block Design

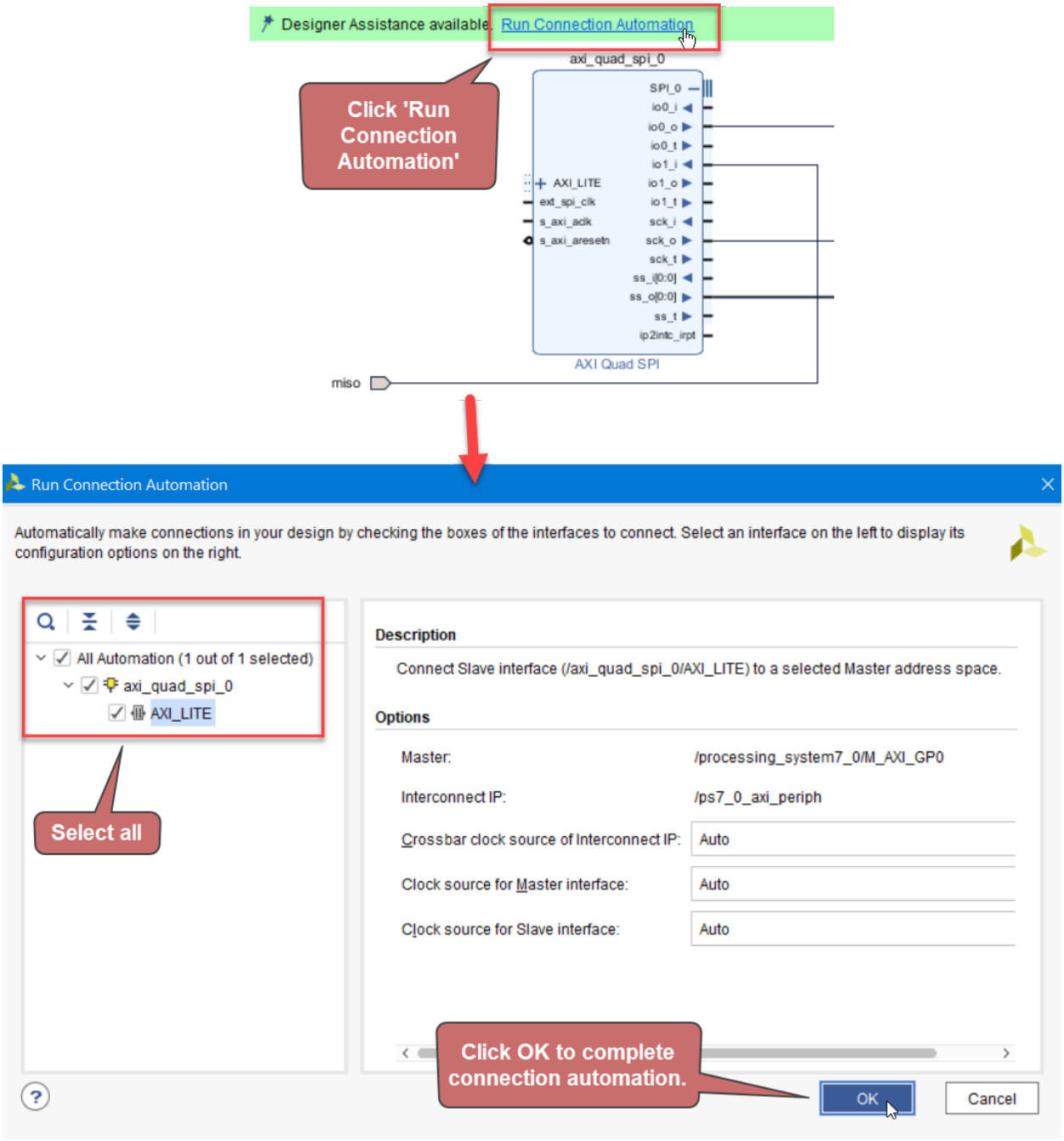


Figure 47. Run Connection Automation again.

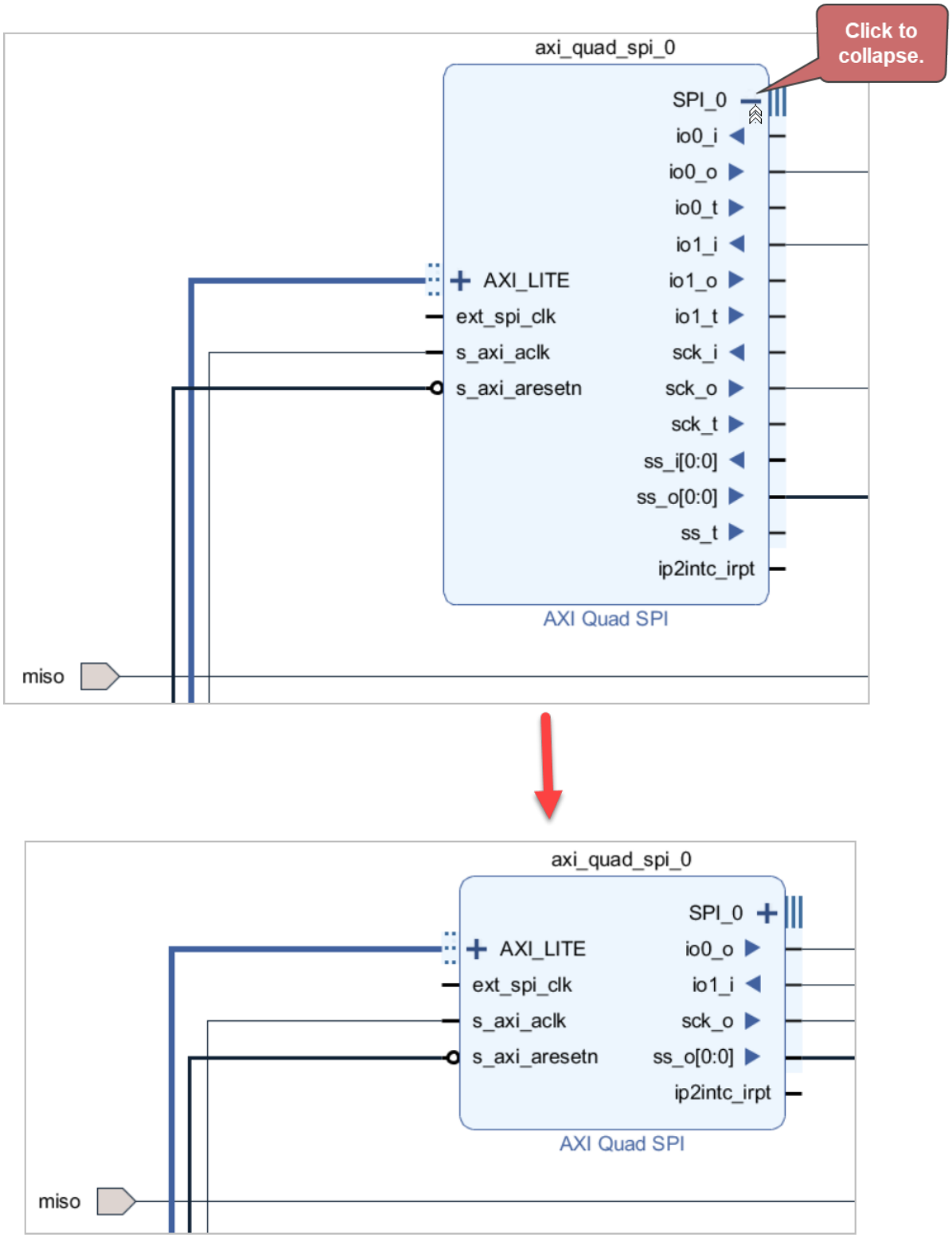


Figure 48. Tidy up block diagram by collapsing SPI_0 port.

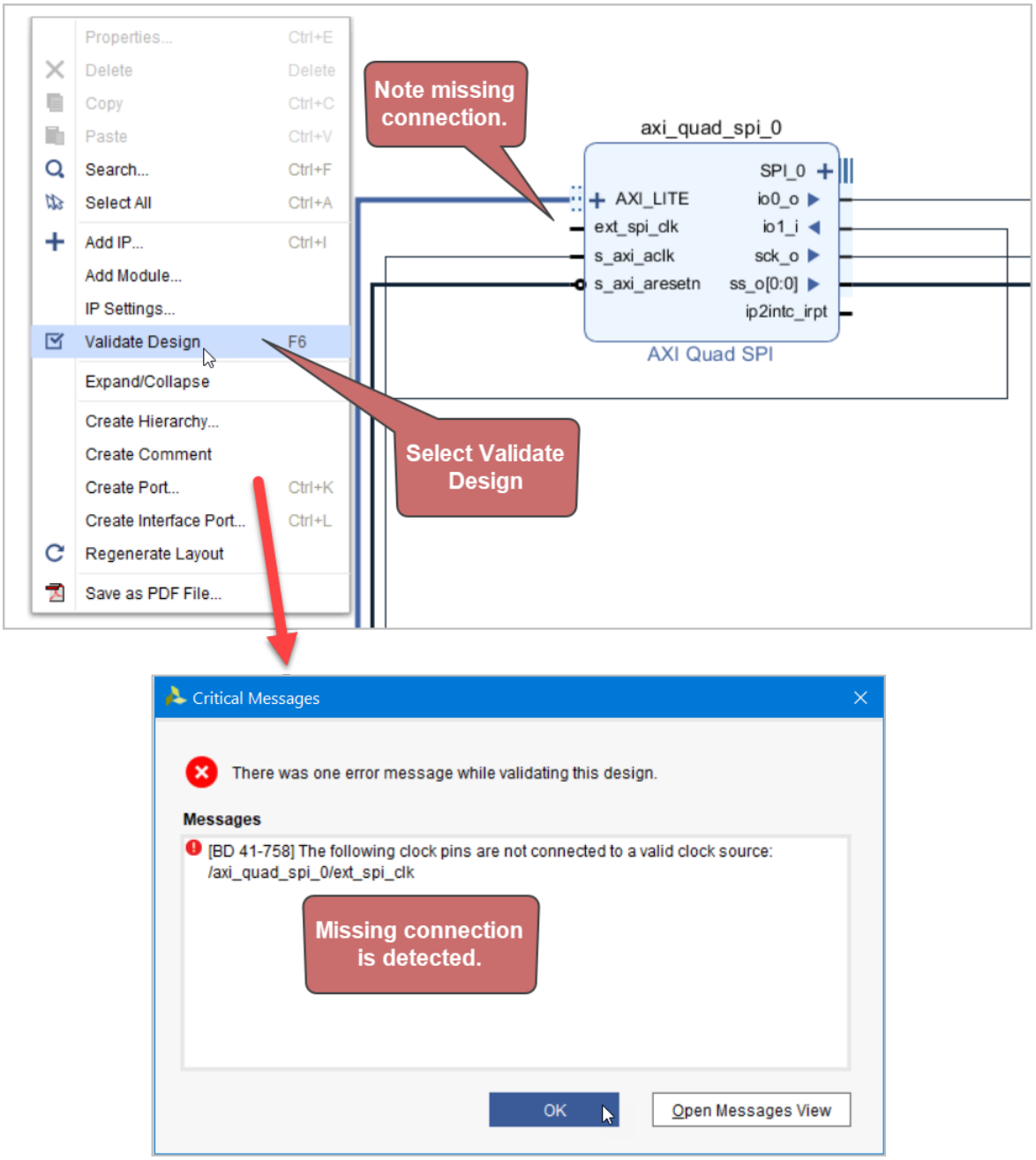


Figure 49. Validate the design; an error should be flagged.

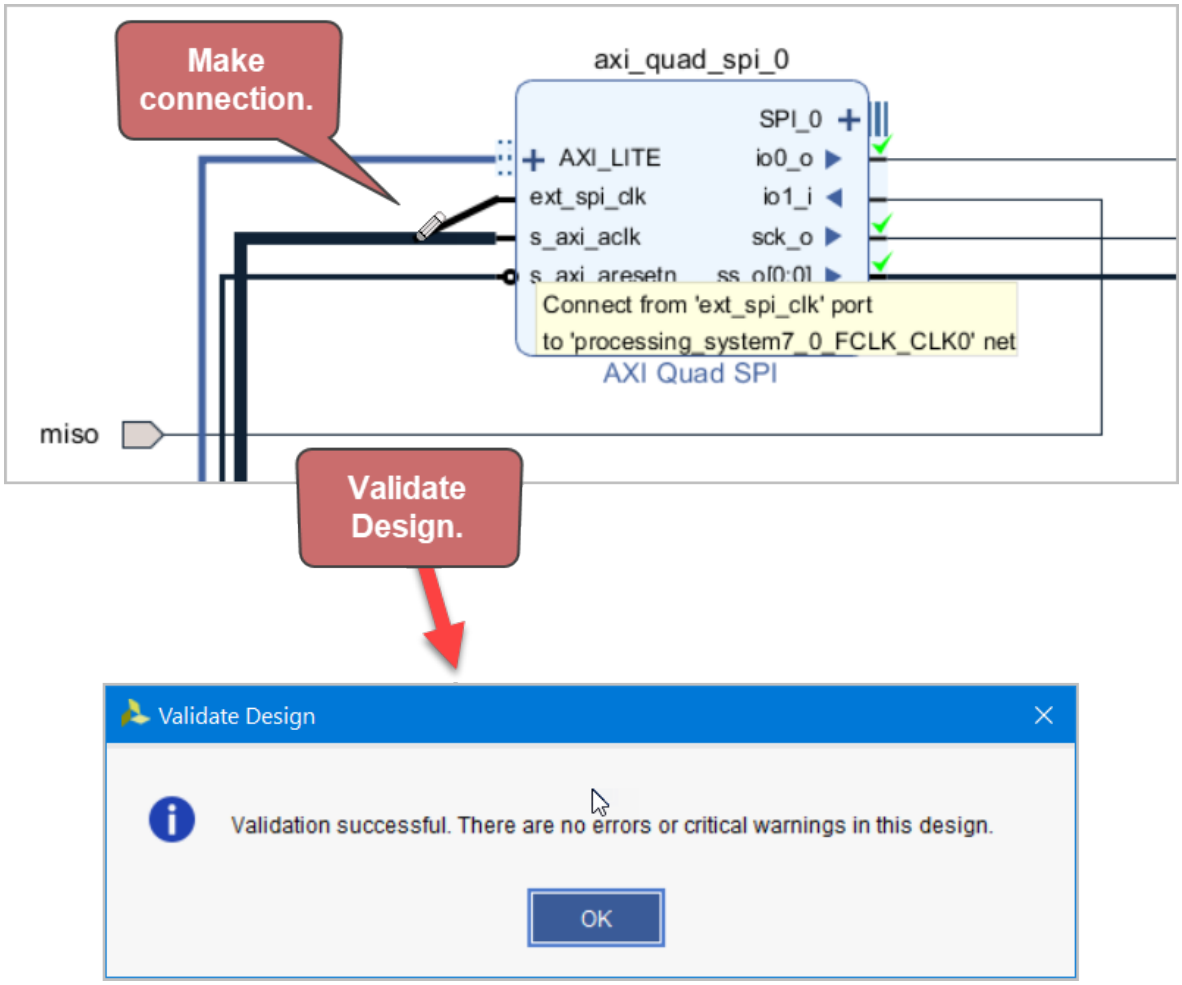


Figure 50. Fix the error by connecting ext_spi_clk to s_axi_aclk.

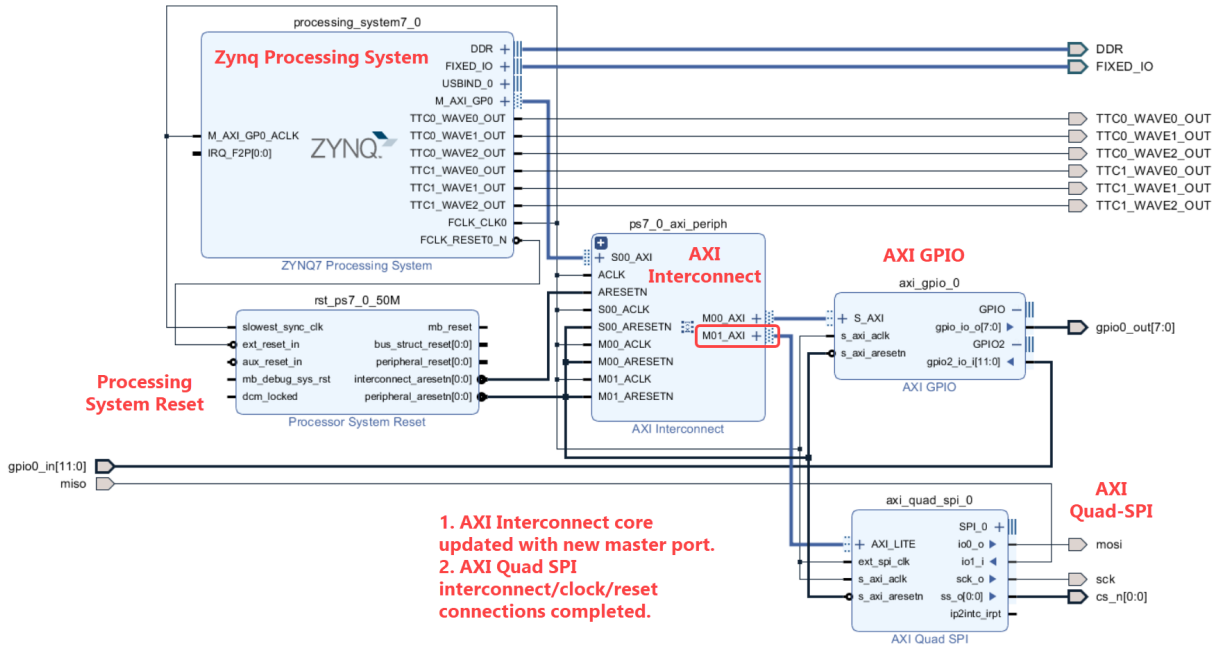


Figure 51. Block diagram at this stage of the design.

2.2.7 Add Concat IP for PmodACL Interrupts

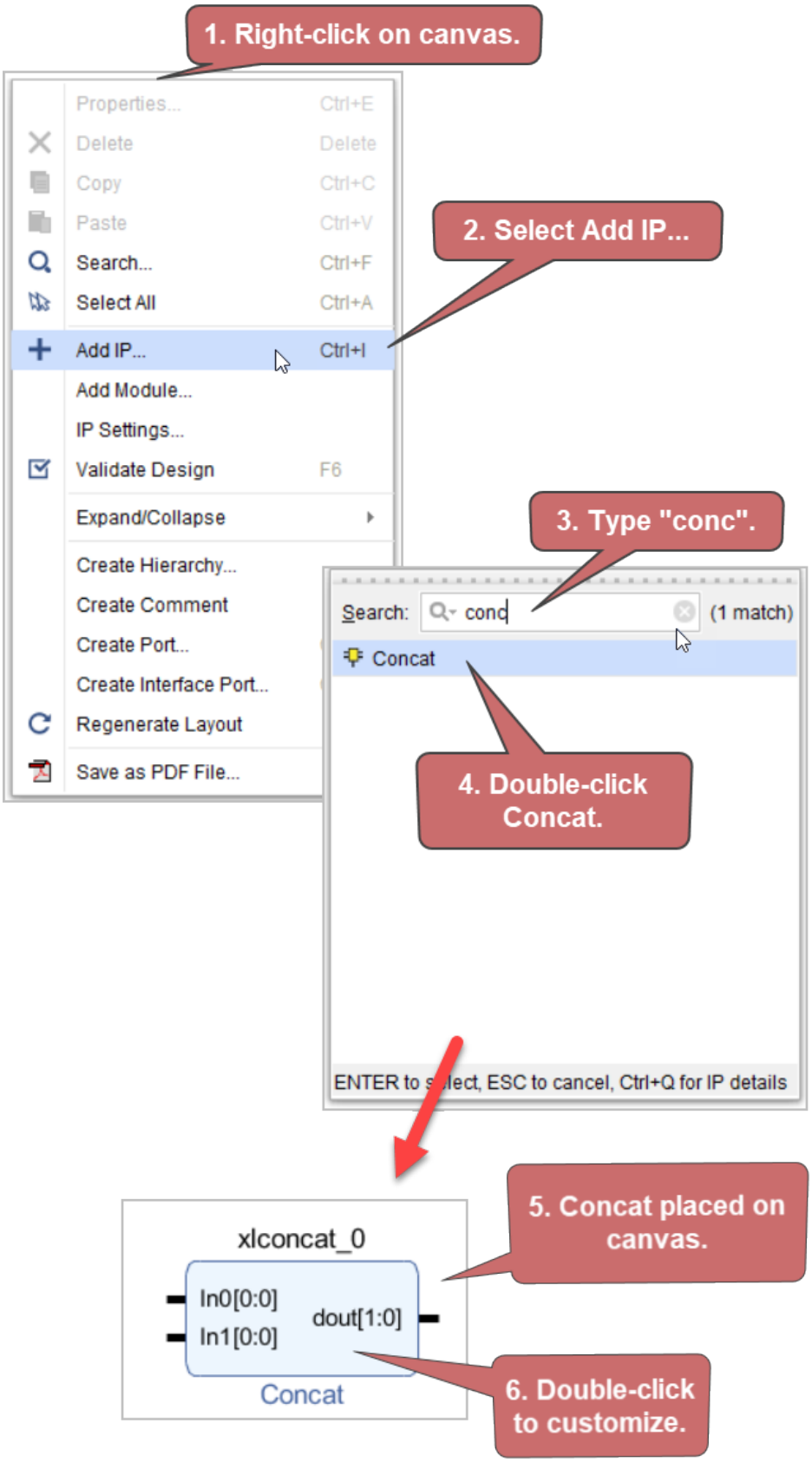


Figure 52. Add the Concat Utility IP for interrupt logic.

2.2.7.1 Customise Concat IP

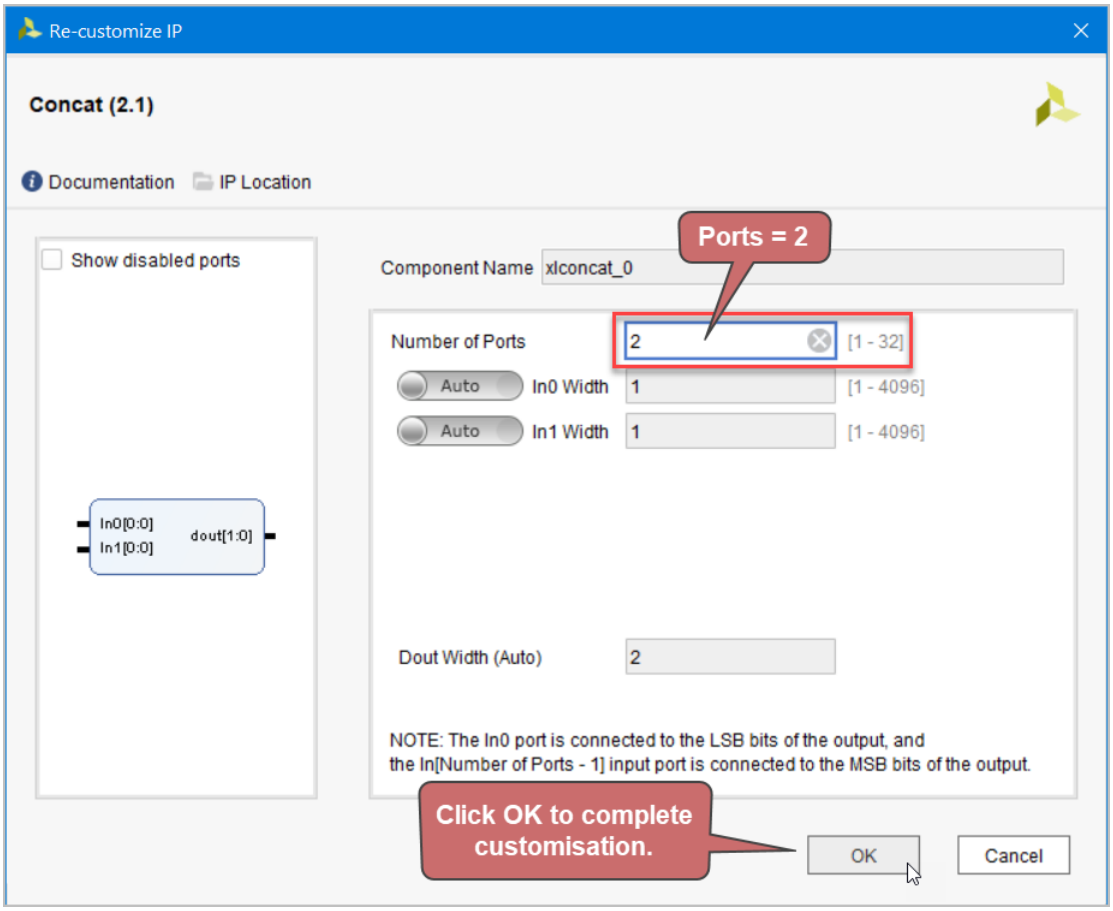


Figure 53. Ensure that two ports are selected and click OK

2.2.7.2 Add PmodACL Interrupt Pins

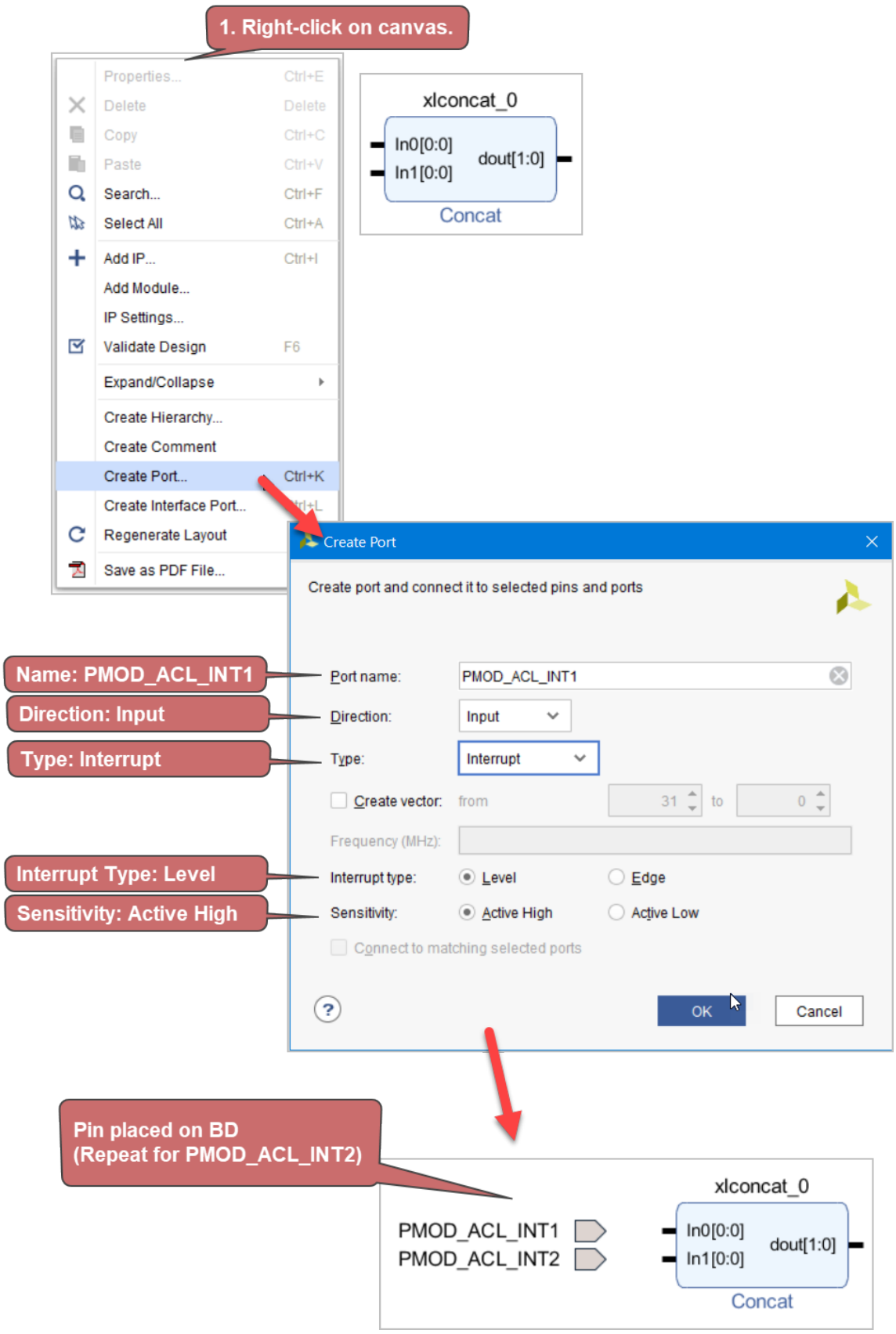


Figure 54. Add the Pmod ACL interrupt pins.

2.2.7.3 Make PmodACL Interrupt Pin Connections

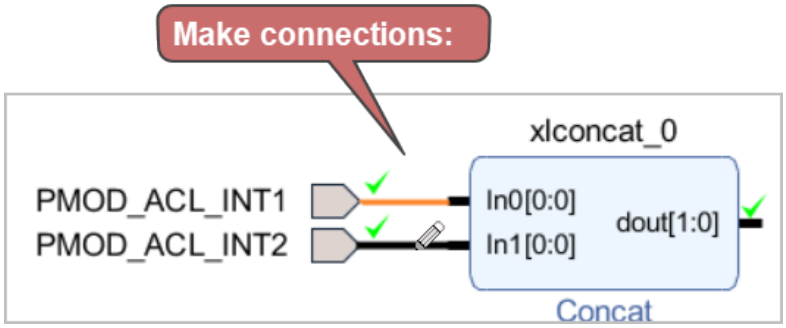


Figure 55. Make the interrupt pin connections.

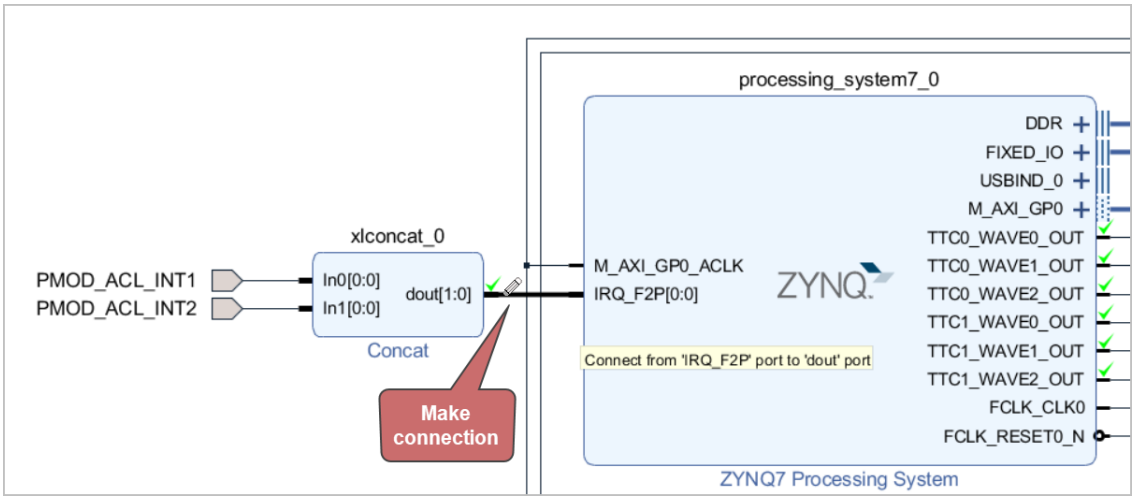


Figure 56. Connect the Concat IP output to IRQ_F2P port on the Zynq IP.

2.2.7.4 Validate Design

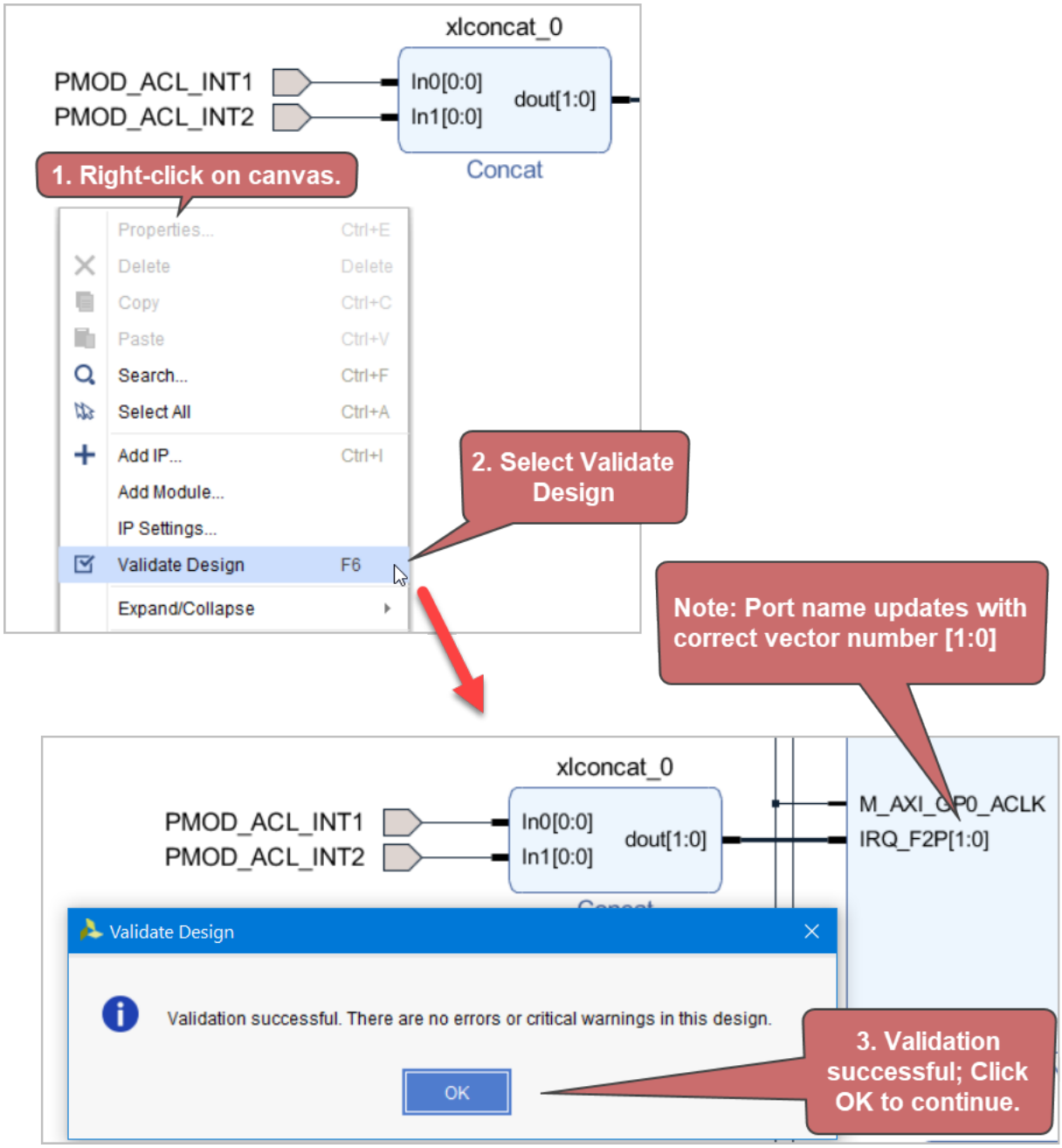


Figure 57. Validate the design for the final time.

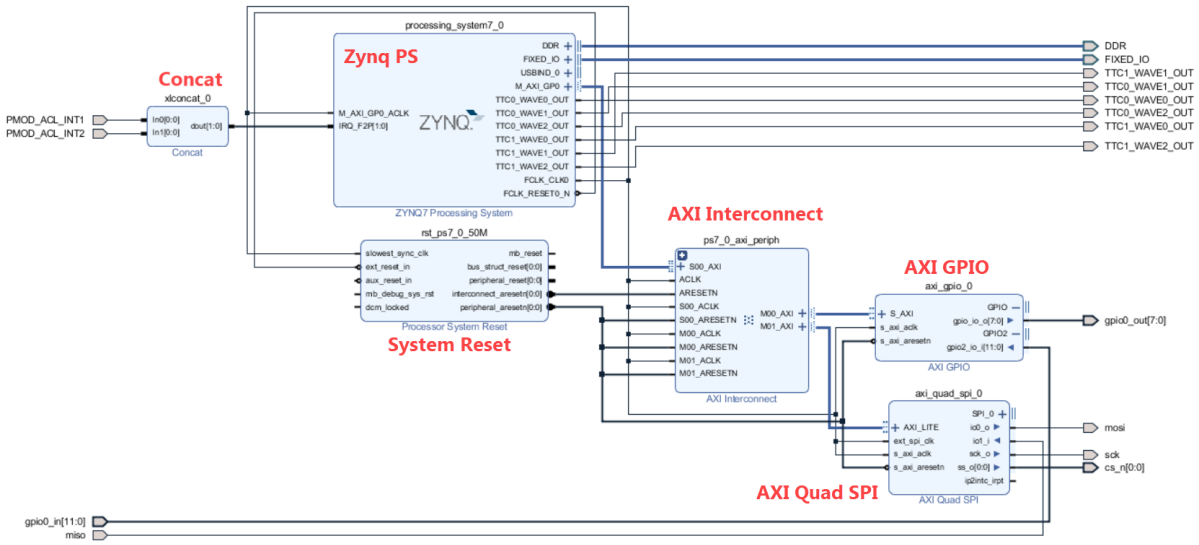


Figure 58. The final block diagram.

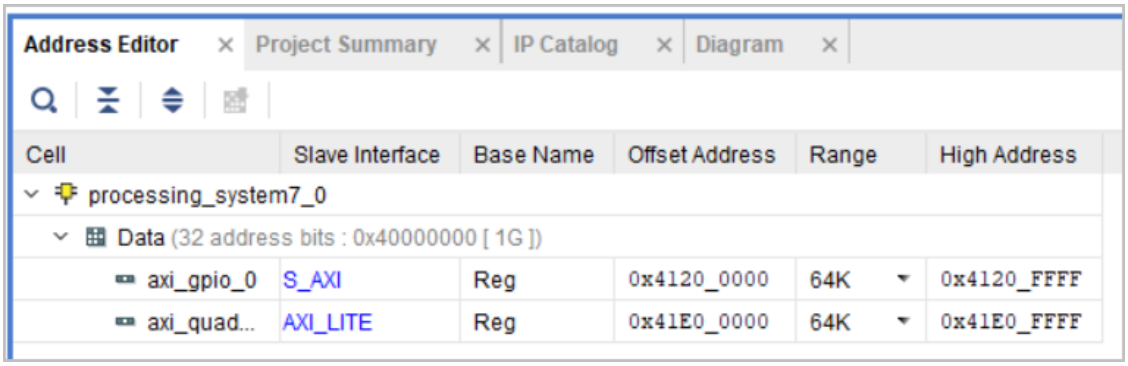


Figure 59. The address information for the AXI GPIO and AXI Quad IP can be viewed in the Address Editor tab.

2.3 Finalize the Design

2.3.1 Generate Output Products

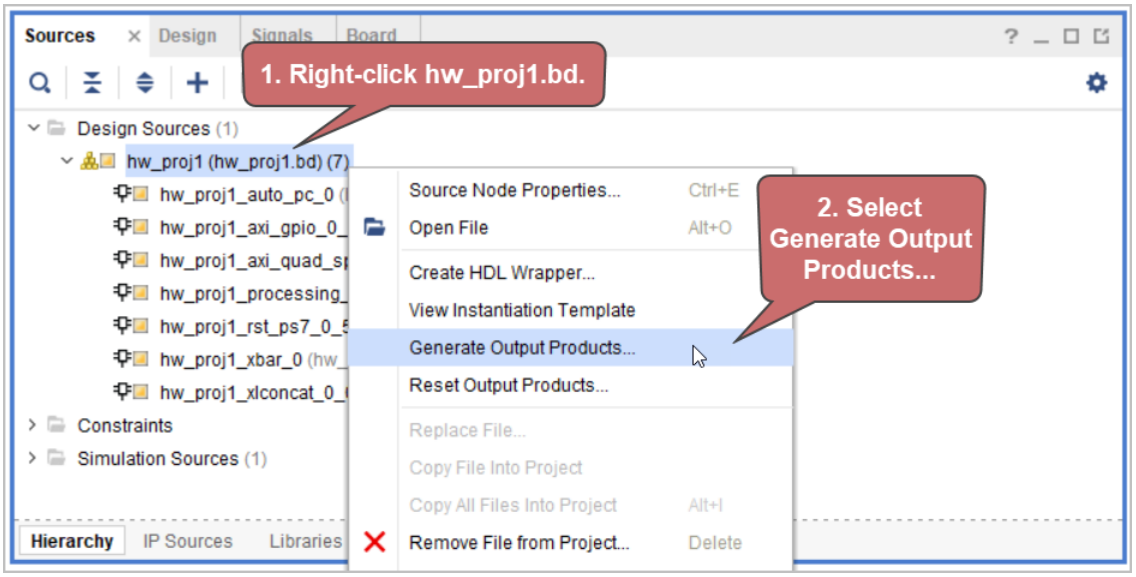


Figure 60. Select Generate Output Products...

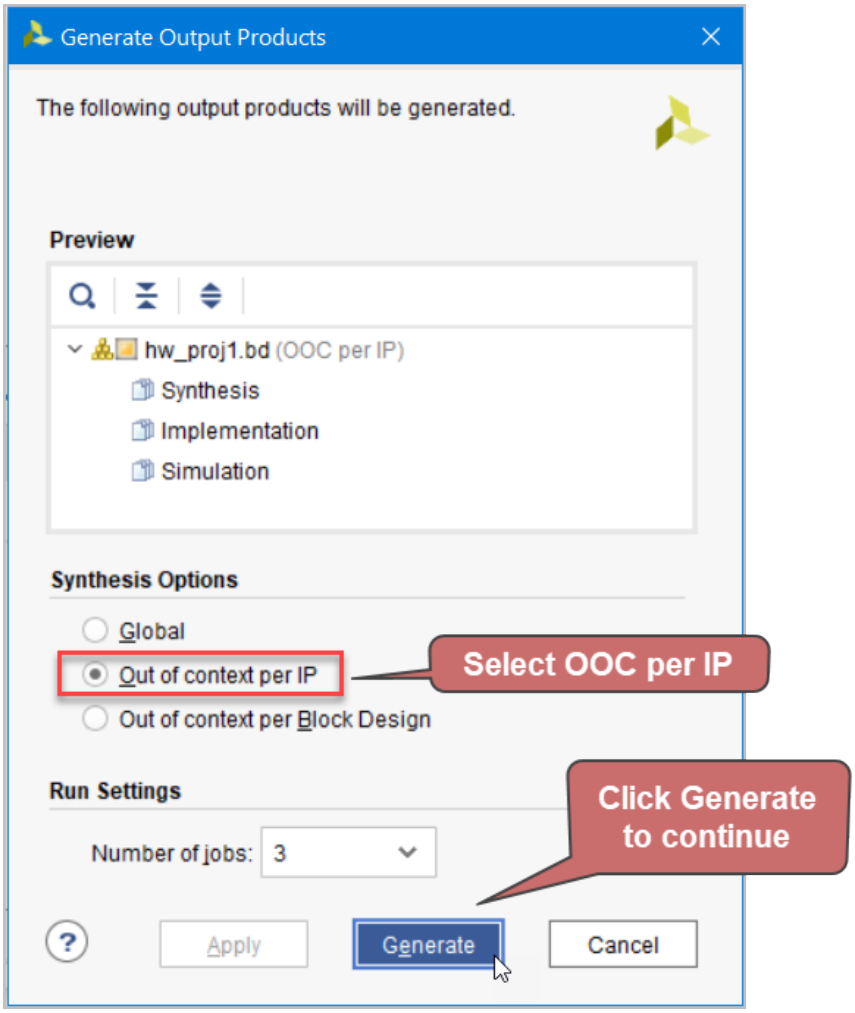


Figure 61. Ensure OOC per IP is selected, and click Generate.

Figure 62. OOC can be set to run in the background.

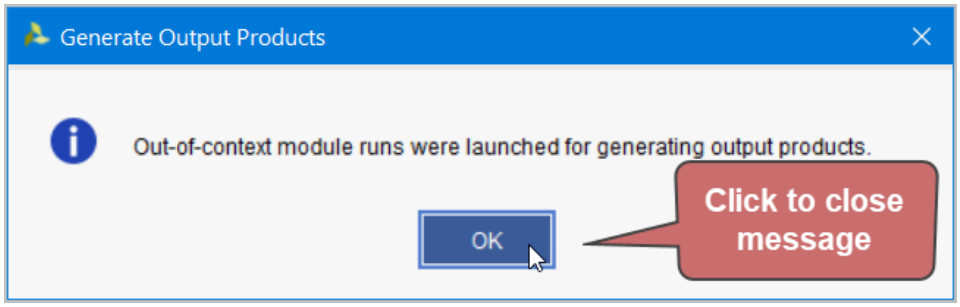


Figure 63. Click OK when the Generate Output Products dialog appears.

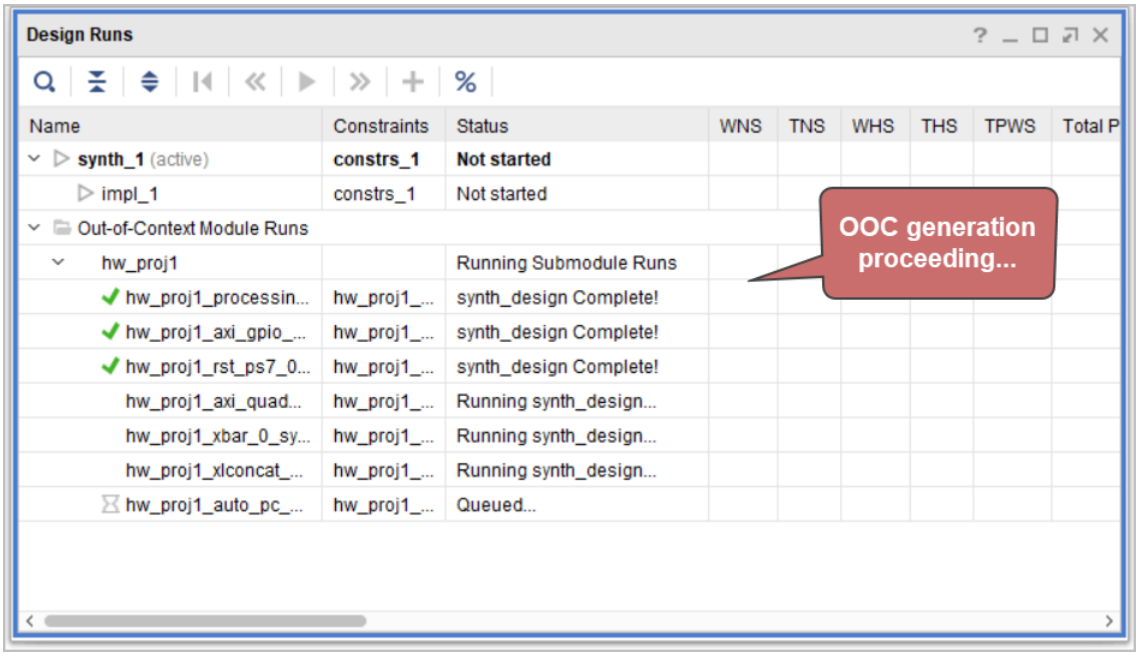


Figure 64. OOC generation runs...

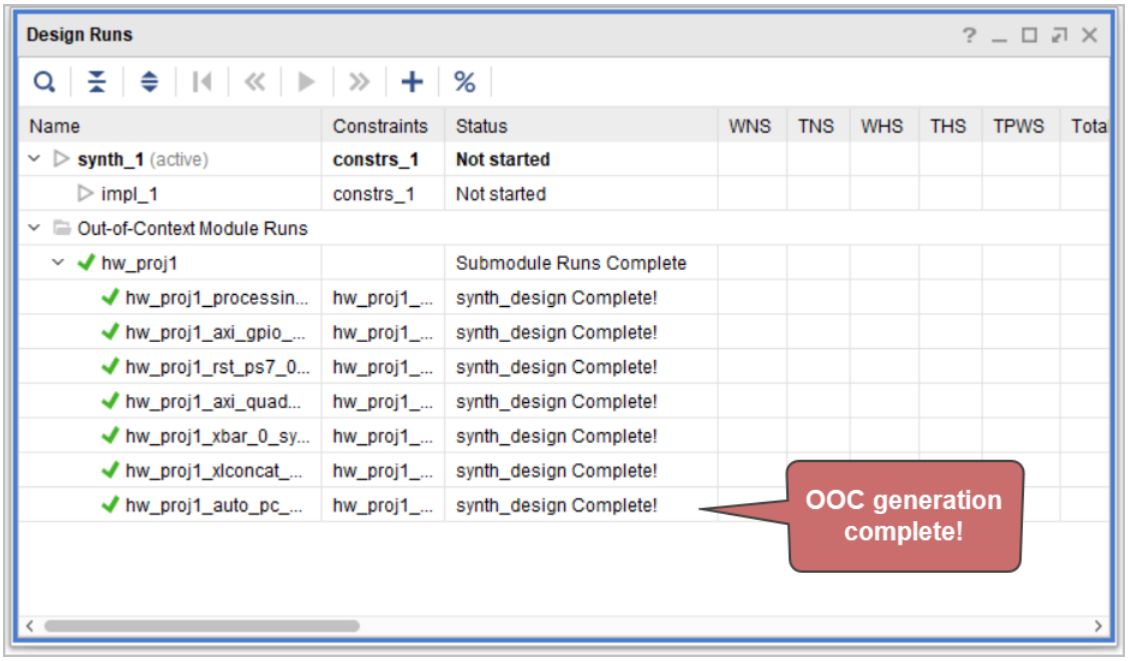


Figure 65. OOC Generation is complete.

2.3.2 Create HDL Wrapper

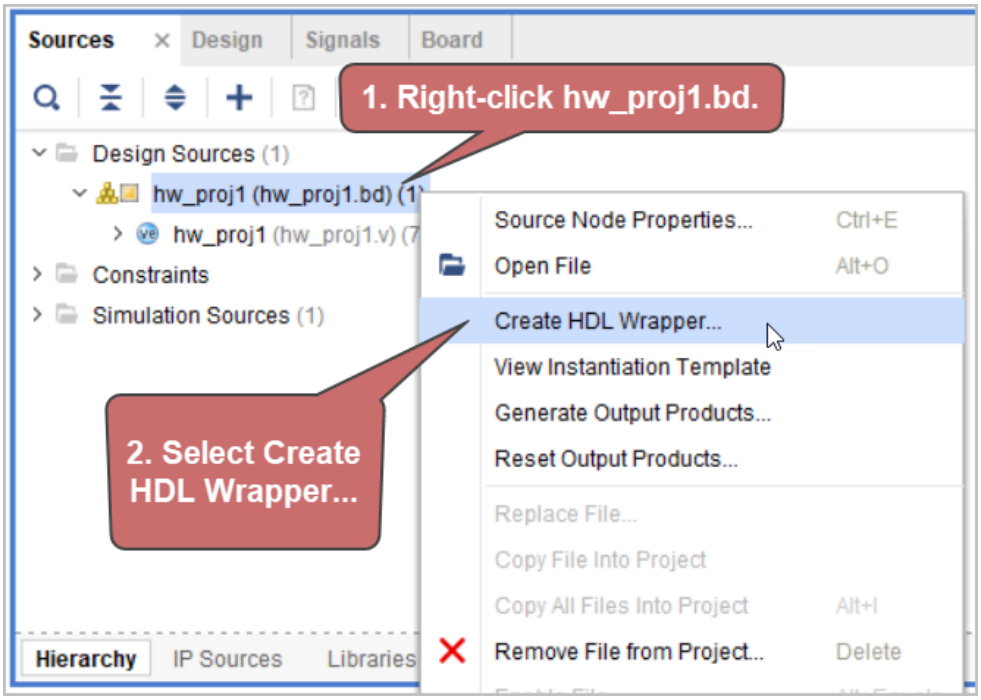


Figure 66. Create the HDL top-level wrapper.

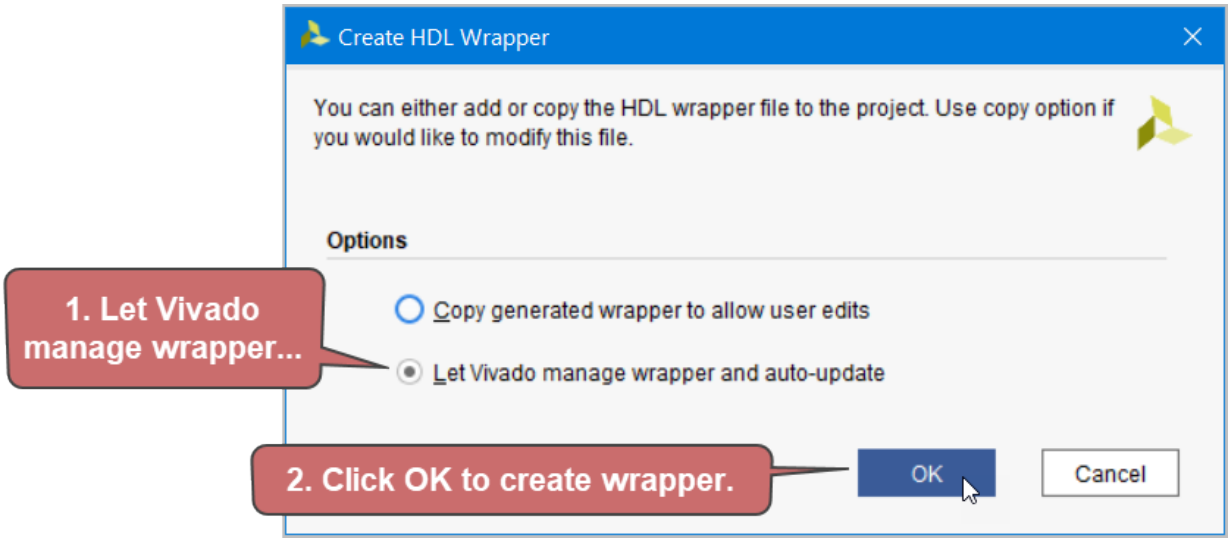


Figure 67. Let Vivado manage the wrapper, and click OK to continue.

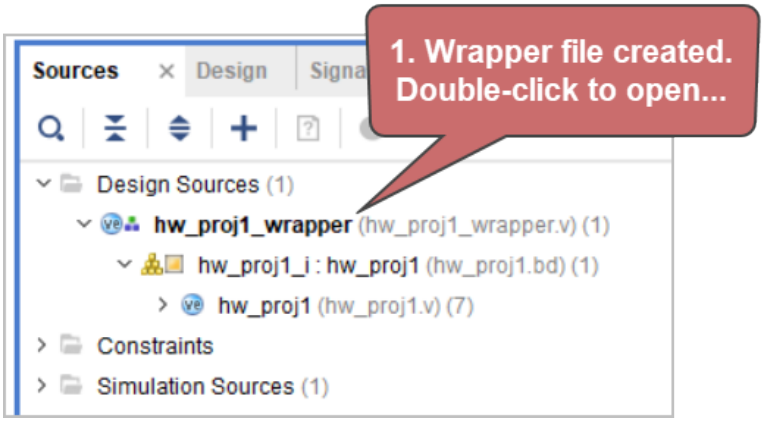


Figure 68. When created, double-click the wrapper to inspect it.

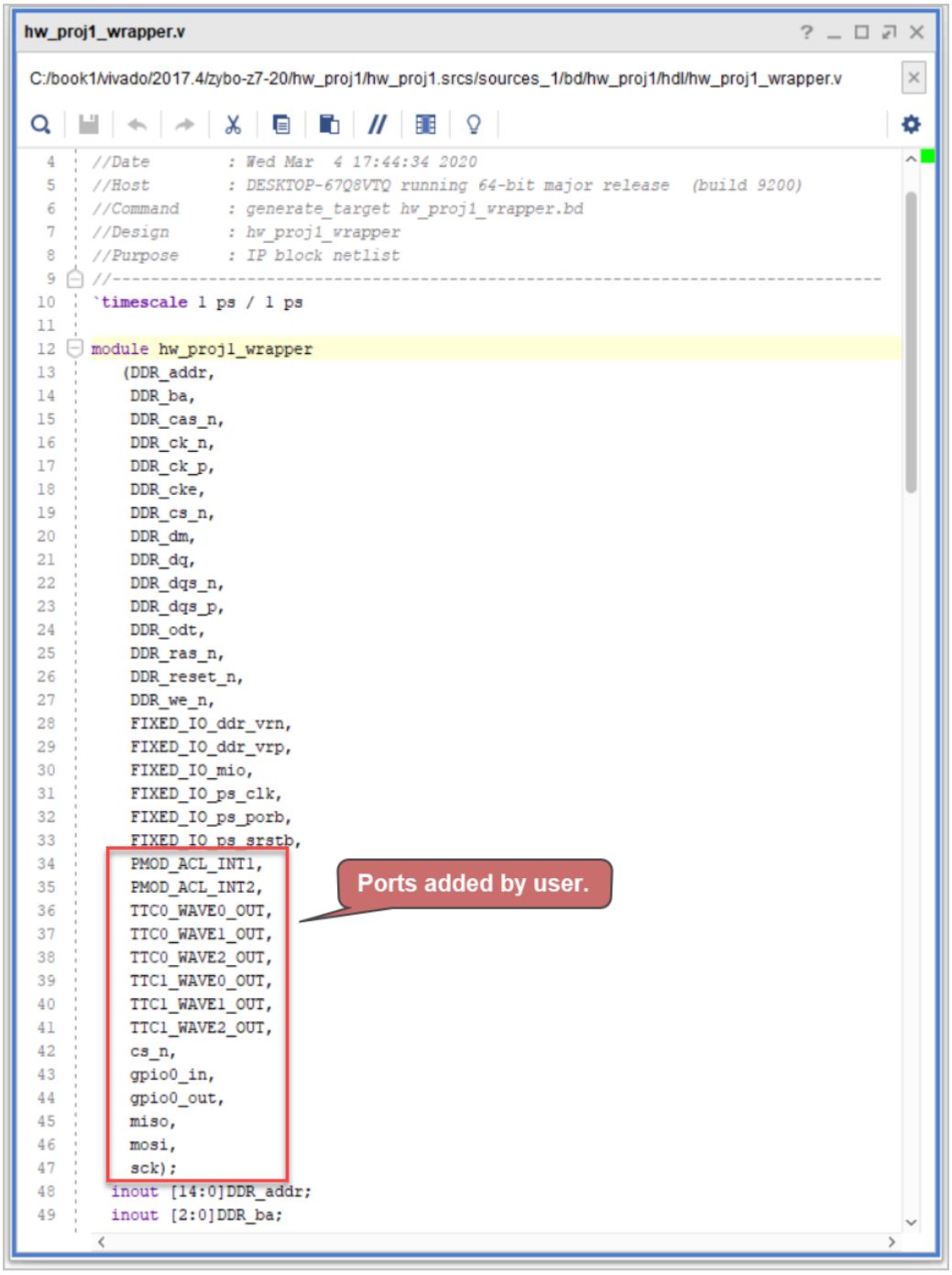


Figure 69. Check that the ports added by the user are present.

2.3.3 Add Constraints for the Project

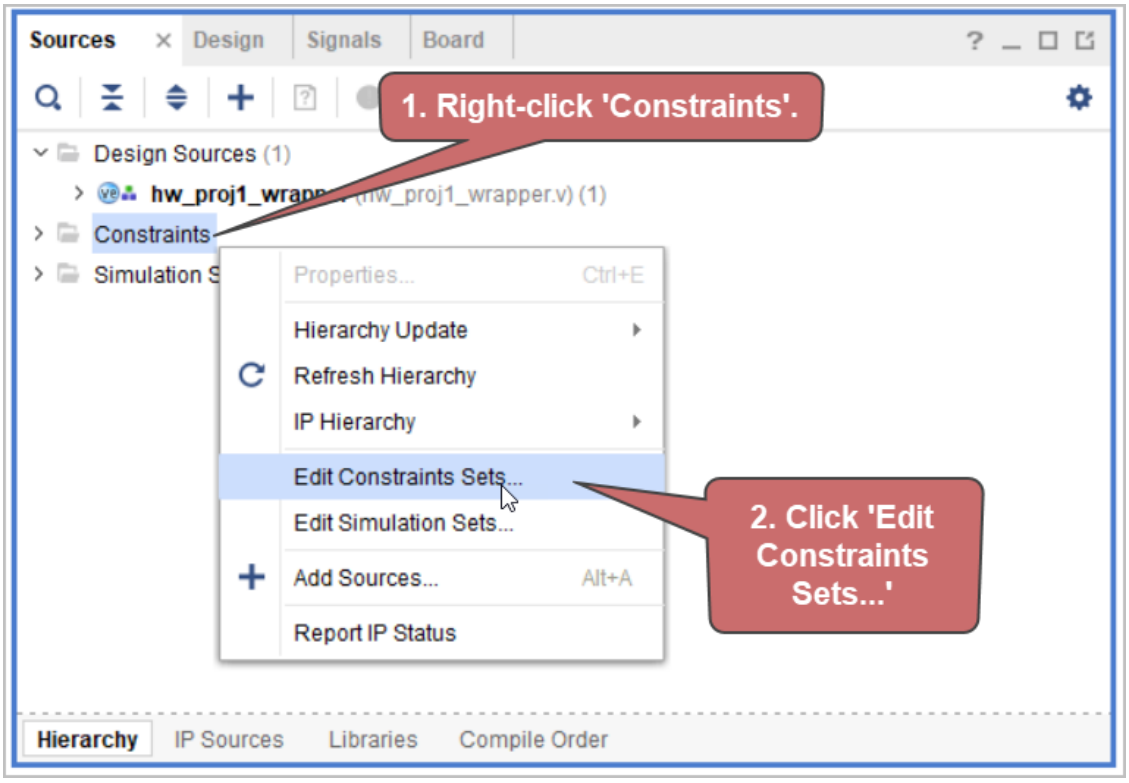


Figure 70. The first step in adding constraints (Right-click, and select "Edit Constraints Sets")

Figure 70 shows the first step in adding constraints for the project. Two options are then available to create the constraints file. The first is to simply to copy the constraints file into the project (see Section 2.3.3.1). The second is to create a new file and add the constraints manually (see Section 2.3.3.2). Each is covered in the coming pages.

Note that if the ZedBoard is being used, then the first option should be used. Simply choose the constraints file for the ZedBoard instead of the file for the Zybo-Z7 in Figure 72.

(See Section 1.3 for details on where to find the constraint files in GitHub.)

2.3.3.1 Option 1: Copy the constraints into the project

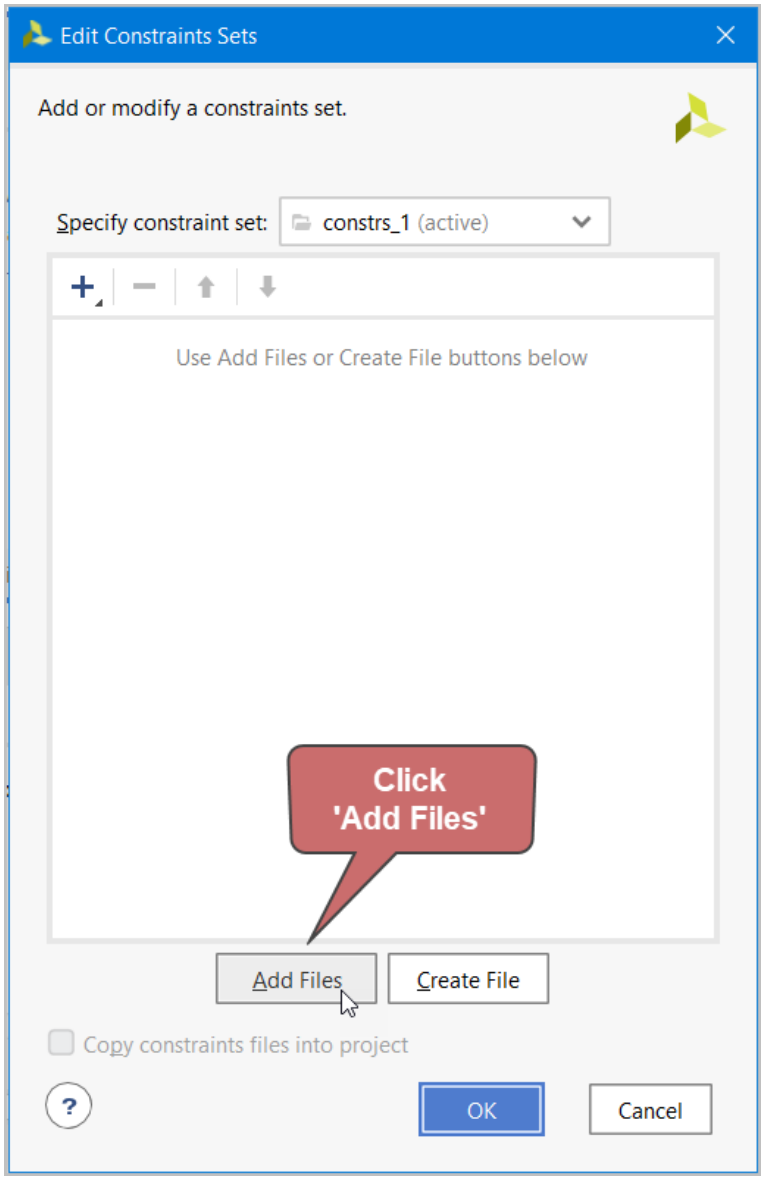


Figure 71. Click Add Files

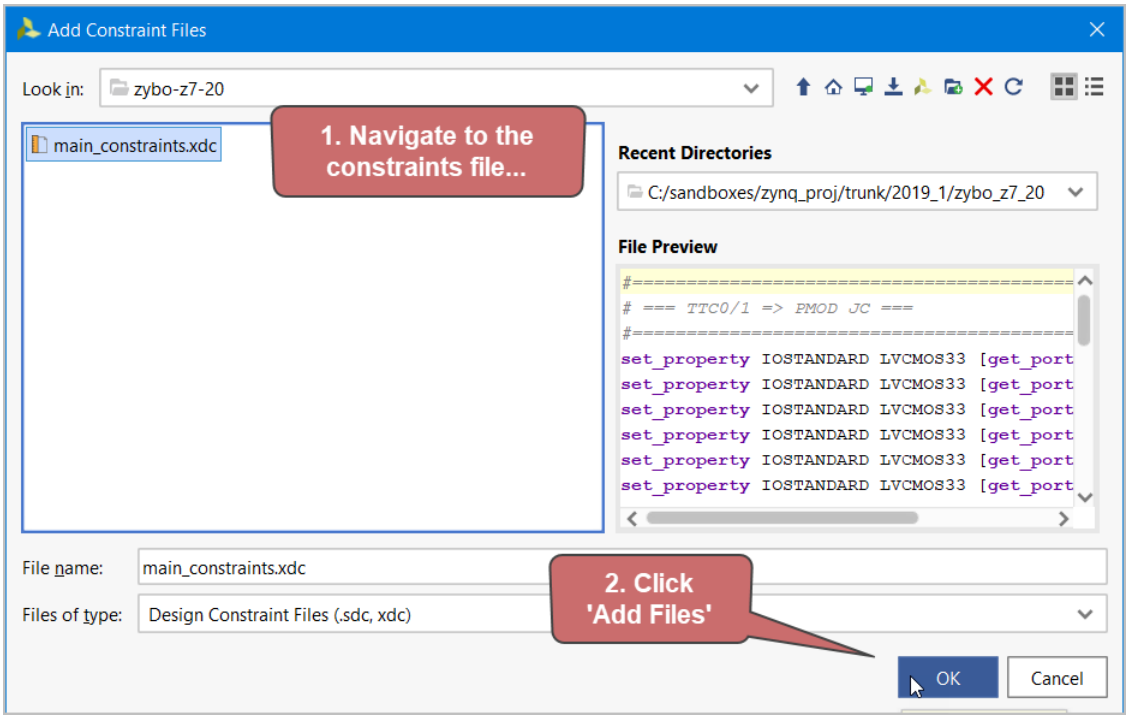


Figure 72. Browse to the supplied constraints file, and click OK to add it. (NOTE: for ZedBoard, browse to the 'zed' directory'_

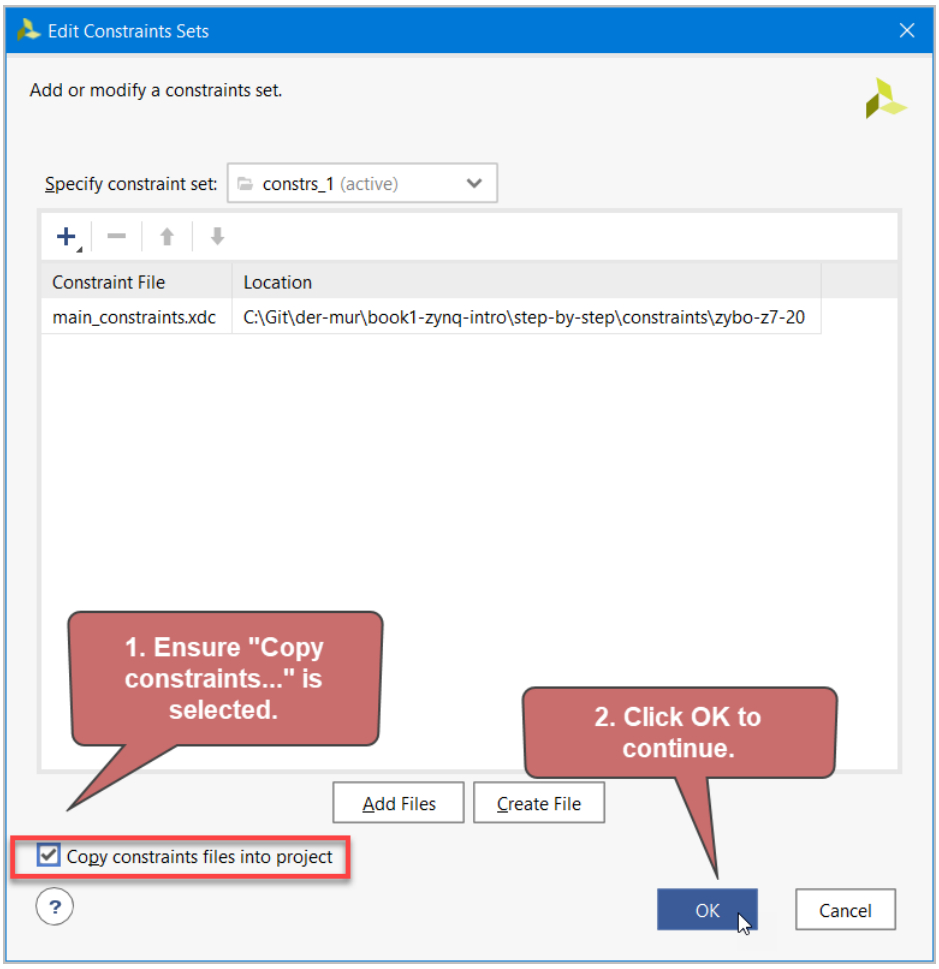


Figure 74. Click OK to continue

2.3.3.2 Option 2: Create the File and Add Constraints

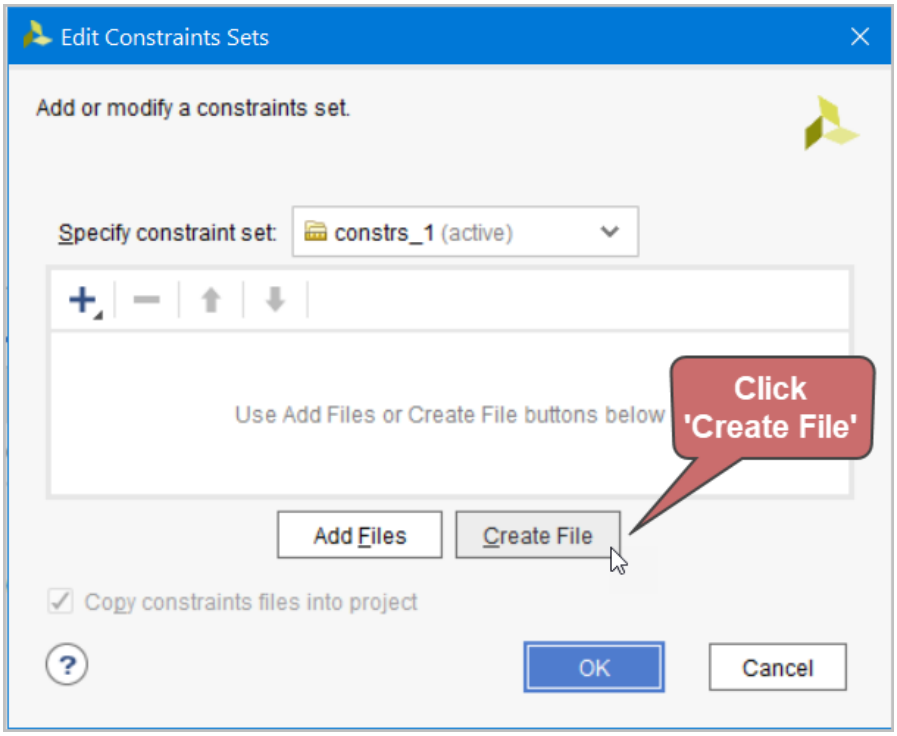


Figure 75. Create the file.

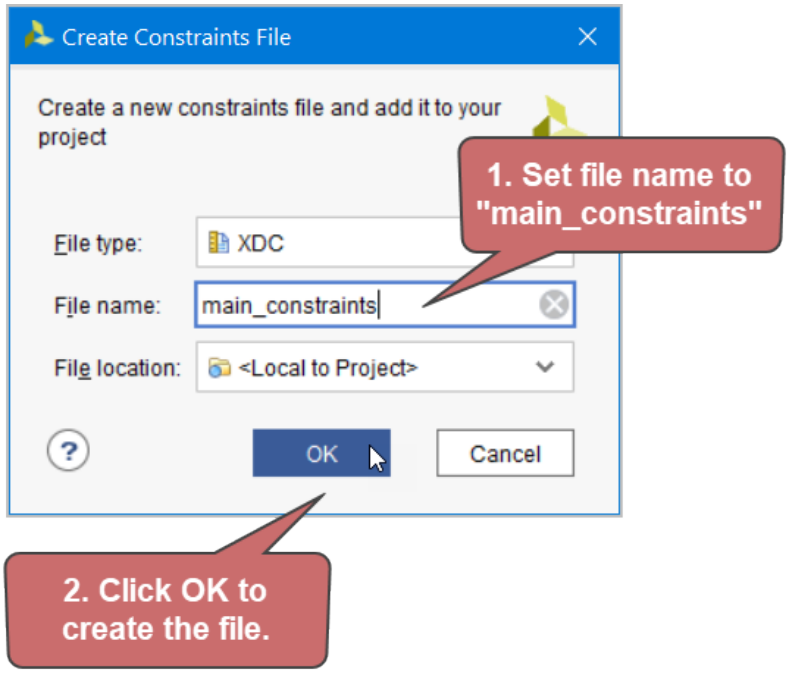


Figure 76. Name the file to "main_constraints" and click OK.

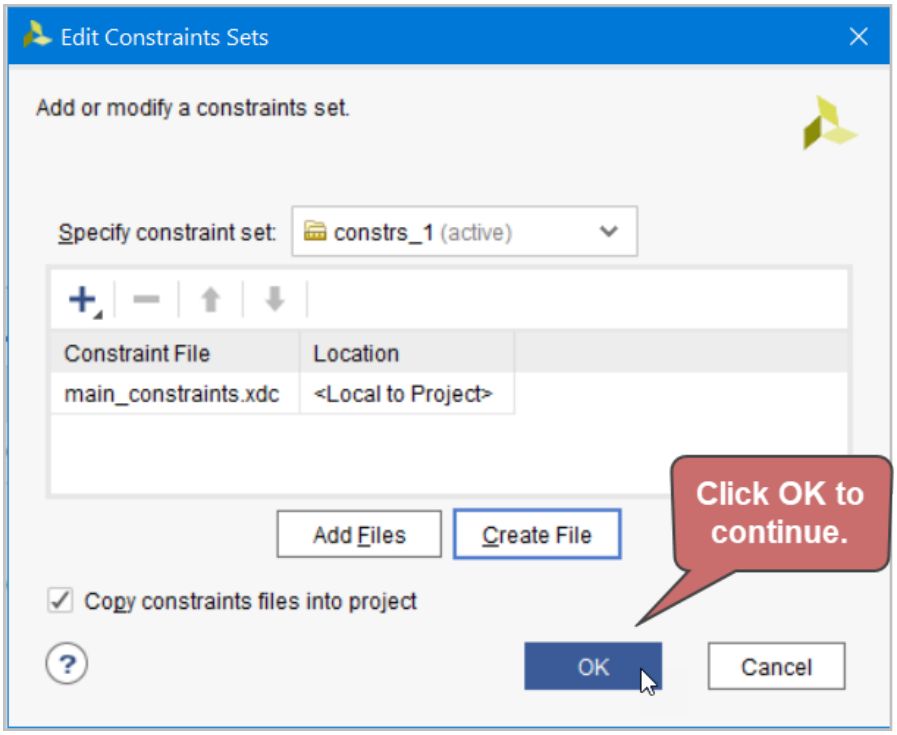


Figure 77. Click OK to add it to the project.

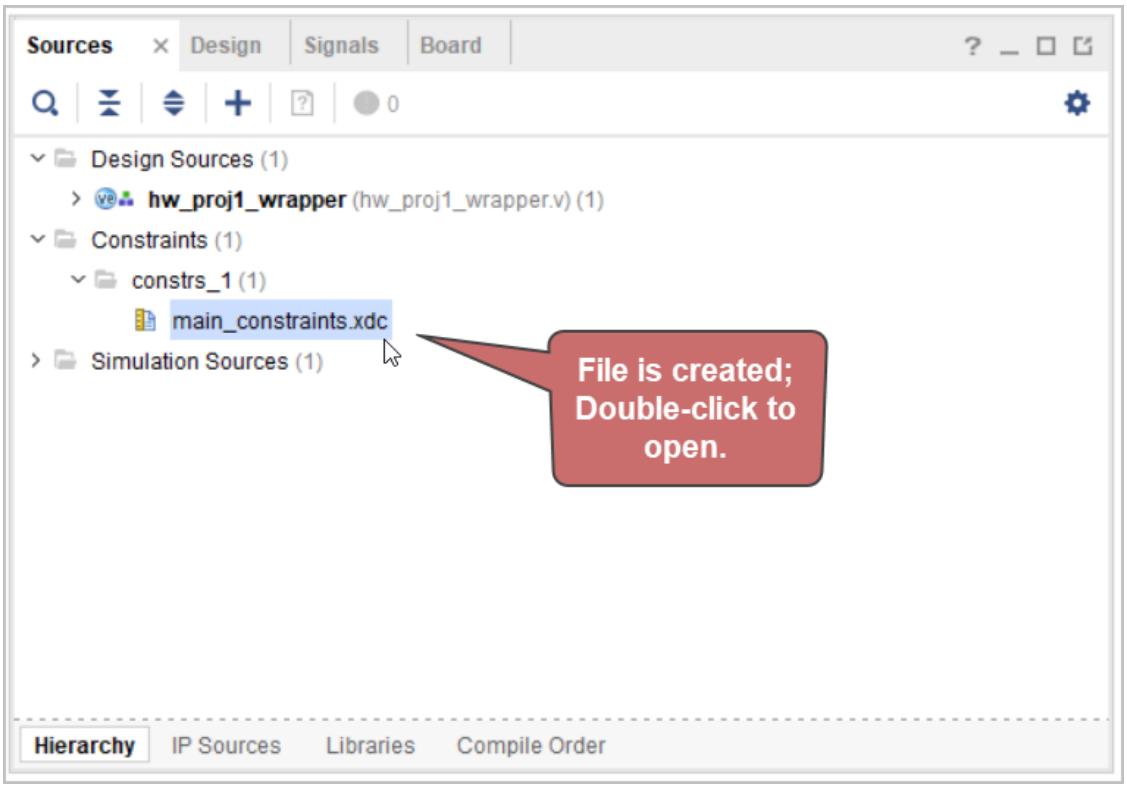


Figure 78. Open the constraints file.

Add (copy) the constraints as shown in the next two pages. (Note that these constraints are specifically for the Zybo-Z7-20 and Zybo-Z7-10 platforms! Do not use for ZedBoard.)

```
#============================================================#
# === AXI GPIO0 ===
#============================================================#
set_property IOSTANDARD LVCMOS33 [get_ports gpio0_out*]
set_property IOSTANDARD LVCMOS33 [get_ports gpio0_in*]

#===============
# gpio0_out[7:0]
#===============
# Board LEDs: LD0 - LD3
set_property PACKAGE_PIN M14 [get_ports {gpio0_out[0]}]
set_property PACKAGE_PIN M15 [get_ports {gpio0_out[1]}]
set_property PACKAGE_PIN G14 [get_ports {gpio0_out[2]}]
set_property PACKAGE_PIN D18 [get_ports {gpio0_out[3]}]

# PMOD JE Pins 1-4
set_property PACKAGE_PIN V12 [get_ports {gpio0_out[4]}]
set_property PACKAGE_PIN W16 [get_ports {gpio0_out[5]}]
set_property PACKAGE_PIN J15 [get_ports {gpio0_out[6]}]
set_property PACKAGE_PIN H15 [get_ports {gpio0_out[7]}]

#===============
# gpio0_in[11:0]
#===============
# Board Push-buttons: BTN0 - BTN3
set_property PACKAGE_PIN K18 [get_ports {gpio0_in[0]}]
set_property PACKAGE_PIN P16 [get_ports {gpio0_in[1]}]
set_property PACKAGE_PIN K19 [get_ports {gpio0_in[2]}]
set_property PACKAGE_PIN Y16 [get_ports {gpio0_in[3]}]

# Board Switches: SW0 - SW3
set_property PACKAGE_PIN G15 [get_ports {gpio0_in[4]}]
set_property PACKAGE_PIN P15 [get_ports {gpio0_in[5]}]
set_property PACKAGE_PIN W13 [get_ports {gpio0_in[6]}]
set_property PACKAGE_PIN T16 [get_ports {gpio0_in[7]}]

# PMOD JE Pins 5-8
set_property PACKAGE_PIN V13 [get_ports {gpio0_in[8]}]
set_property PACKAGE_PIN U17 [get_ports {gpio0_in[9]}]
set_property PACKAGE_PIN T17 [get_ports {gpio0_in[10]}]
set_property PACKAGE_PIN Y17 [get_ports {gpio0_in[11]}]
```

```
#=========================================================#
# === TTC0/1 => PMOD JC ===
#=========================================================#
set_property IOSTANDARD LVCMOS33 [get_ports TTC0_WAVE0_OUT]
set_property IOSTANDARD LVCMOS33 [get_ports TTC0_WAVE1_OUT]
set_property IOSTANDARD LVCMOS33 [get_ports TTC0_WAVE2_OUT]
set_property IOSTANDARD LVCMOS33 [get_ports TTC1_WAVE0_OUT]
set_property IOSTANDARD LVCMOS33 [get_ports TTC1_WAVE1_OUT]
set_property IOSTANDARD LVCMOS33 [get_ports TTC1_WAVE2_OUT]

set_property PACKAGE_PIN V15 [get_ports TTC0_WAVE0_OUT]
set_property PACKAGE_PIN W15 [get_ports TTC0_WAVE1_OUT]
set_property PACKAGE_PIN T11 [get_ports TTC0_WAVE2_OUT]
# PIN 4 (T10) NOT USED
set_property PACKAGE_PIN W14 [get_ports TTC1_WAVE0_OUT]
set_property PACKAGE_PIN Y14 [get_ports TTC1_WAVE1_OUT]
set_property PACKAGE_PIN T12 [get_ports TTC1_WAVE2_OUT]
# PIN 10 (U12) NOT USED

#=========================================================#
# === PmodACL => PMOD JD ===
#=========================================================#
set_property IOSTANDARD LVCMOS33 [get_ports cs_n]
set_property IOSTANDARD LVCMOS33 [get_ports mosi]
set_property IOSTANDARD LVCMOS33 [get_ports miso]
set_property IOSTANDARD LVCMOS33 [get_ports sclk]
set_property IOSTANDARD LVCMOS33 [get_ports PMOD_ACL_INT2]
set_property IOSTANDARD LVCMOS33 [get_ports PMOD_ACL_INT1]

# cs_n => PMOD JD PIN 1 = T14
# mosi => PMOD JD PIN 2 = T15
# miso => PMOD JD PIN 3 = P14
# sclk => PMOD JD PIN 4 = R14
# PMOD_ACL_INT2 => PMOD JD PIN 7 = U14
# PMOD_ACL_INT1 => PMOD JD PIN 8 = U15
set_property PACKAGE_PIN T14 [get_ports cs_n]
set_property PACKAGE_PIN T15 [get_ports mosi]
set_property PACKAGE_PIN P14 [get_ports miso]
set_property PACKAGE_PIN R14 [get_ports sclk]
set_property PACKAGE_PIN U14 [get_ports PMOD_ACL_INT2]
set_property PACKAGE_PIN U15 [get_ports PMOD_ACL_INT1]
```

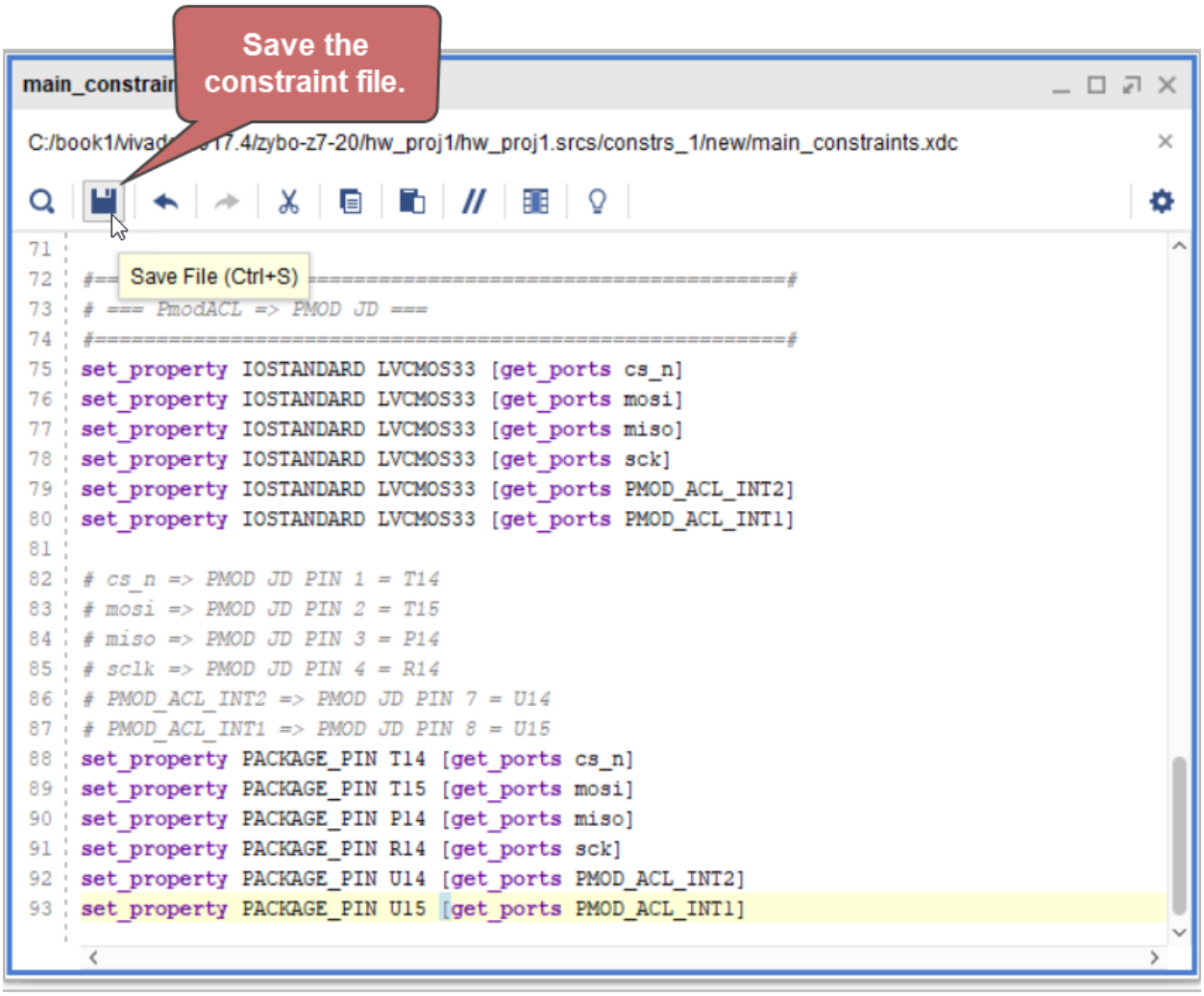


Figure 79. Save the constraints file.

2.4 Design Generation

2.4.1 Synthesize the Design

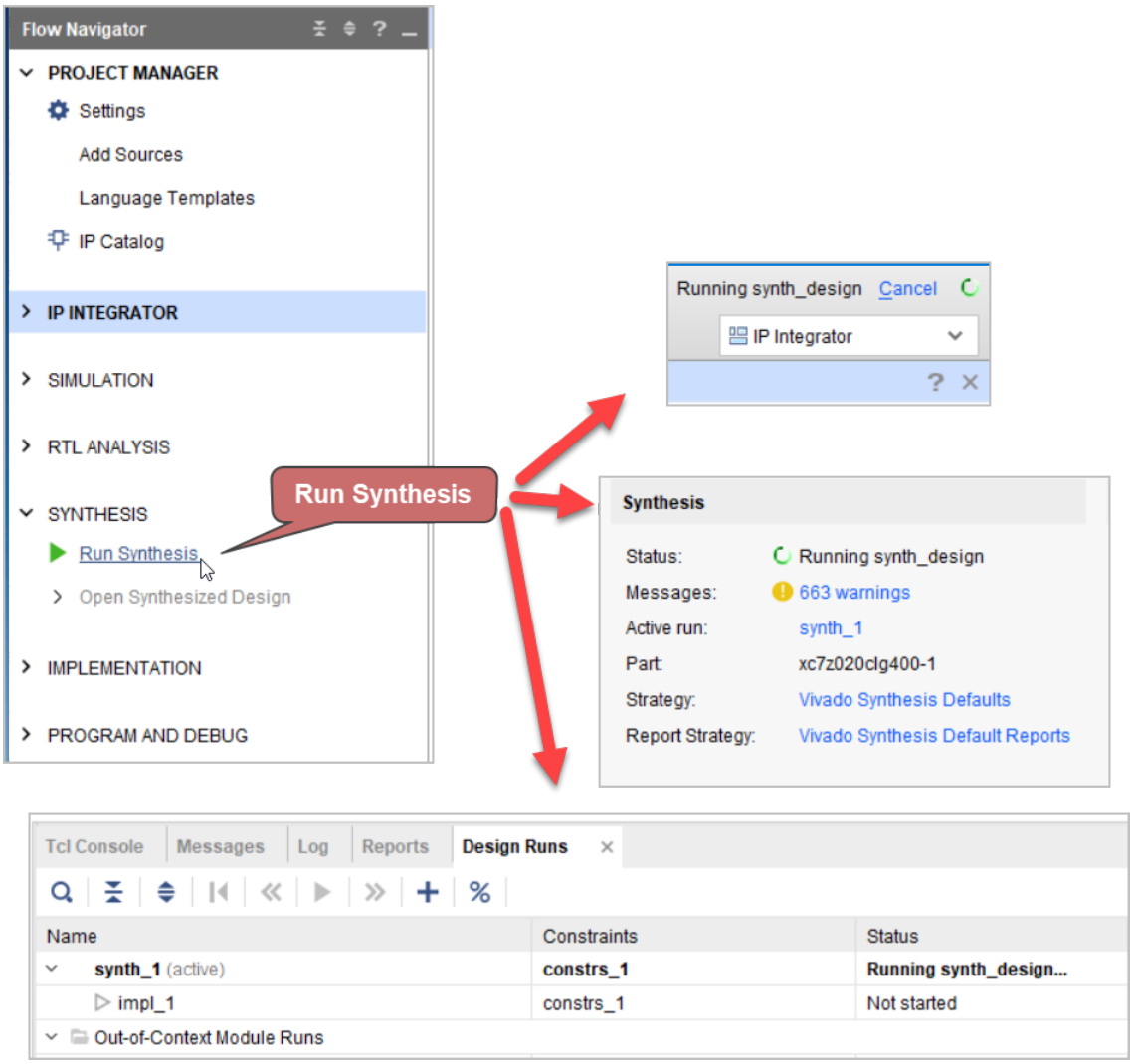


Figure 80. Run design synthesis.

2.4.2 Implement the Design

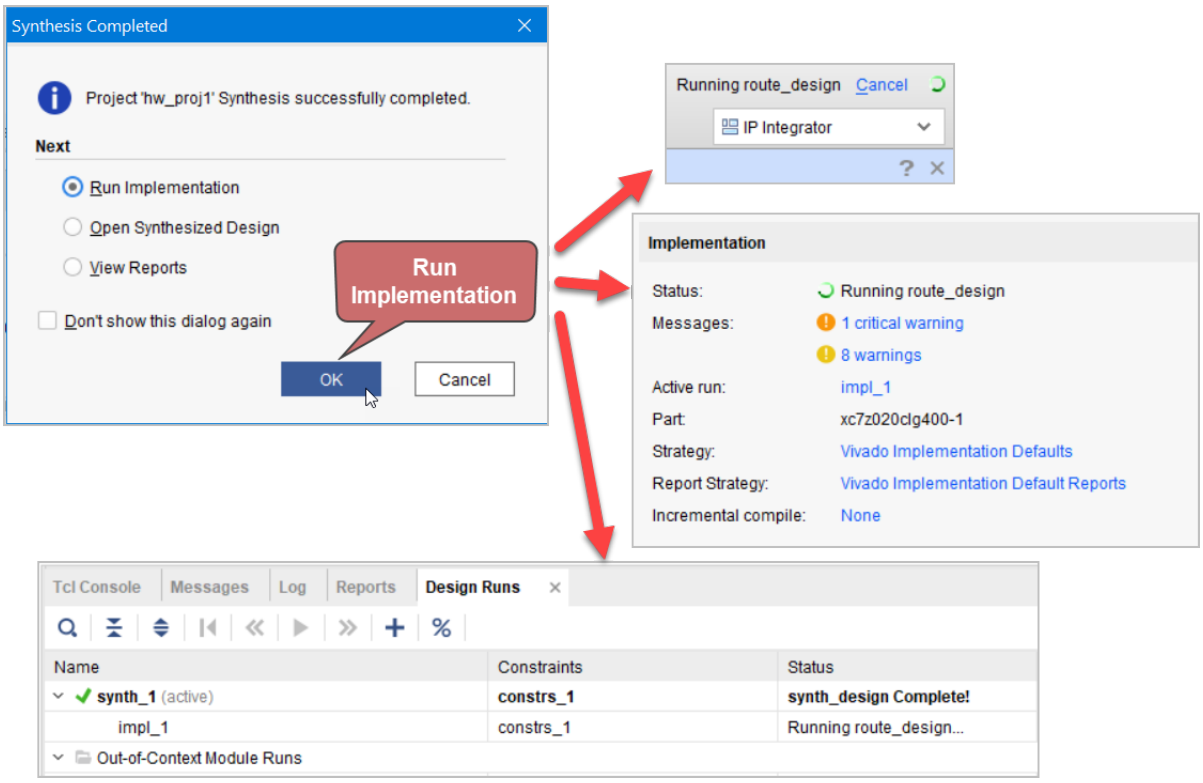


Figure 81. Implement the design.

2.4.3 Generate the Bitstream

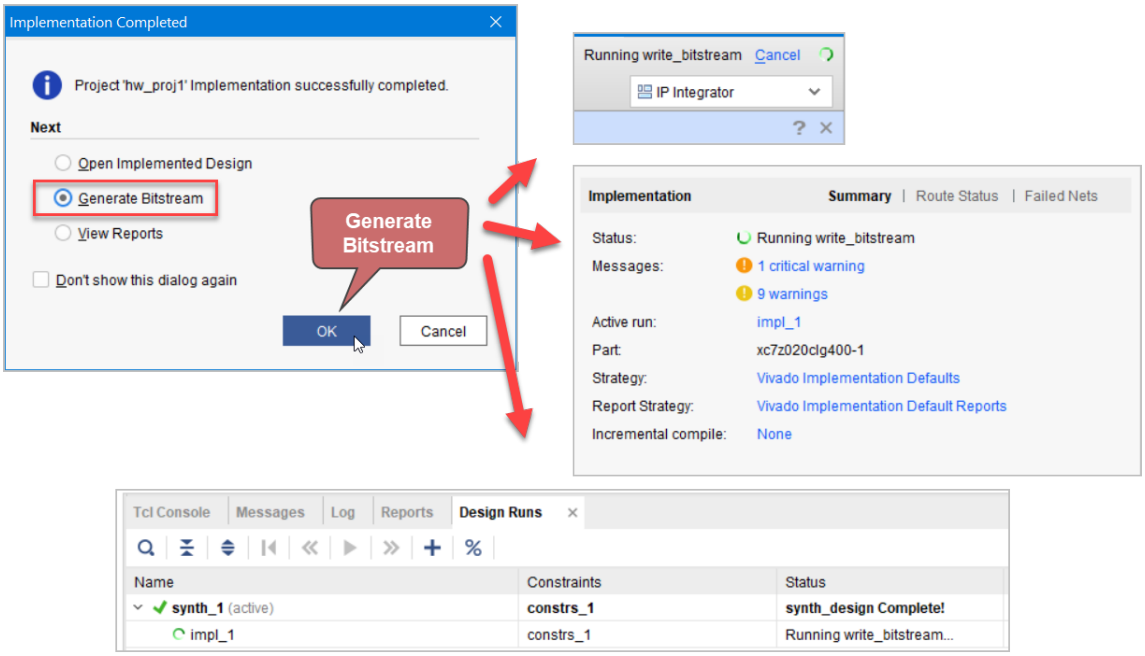


Figure 82. Generate the bitstream.

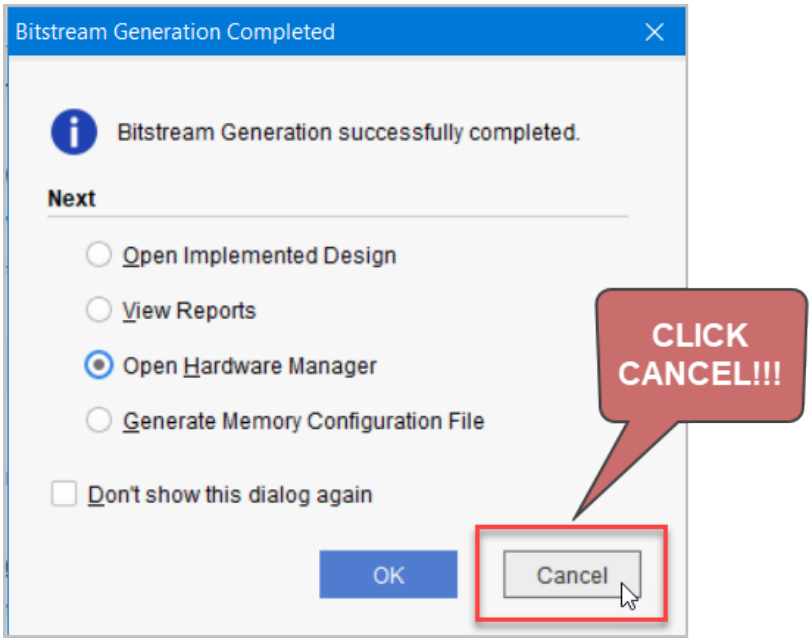


Figure 83. When the bitstream is generated, click Cancel (unless the user wants to open the HW Manager).

2.5 Hardware Hand-off

2.5.1 Export the Design

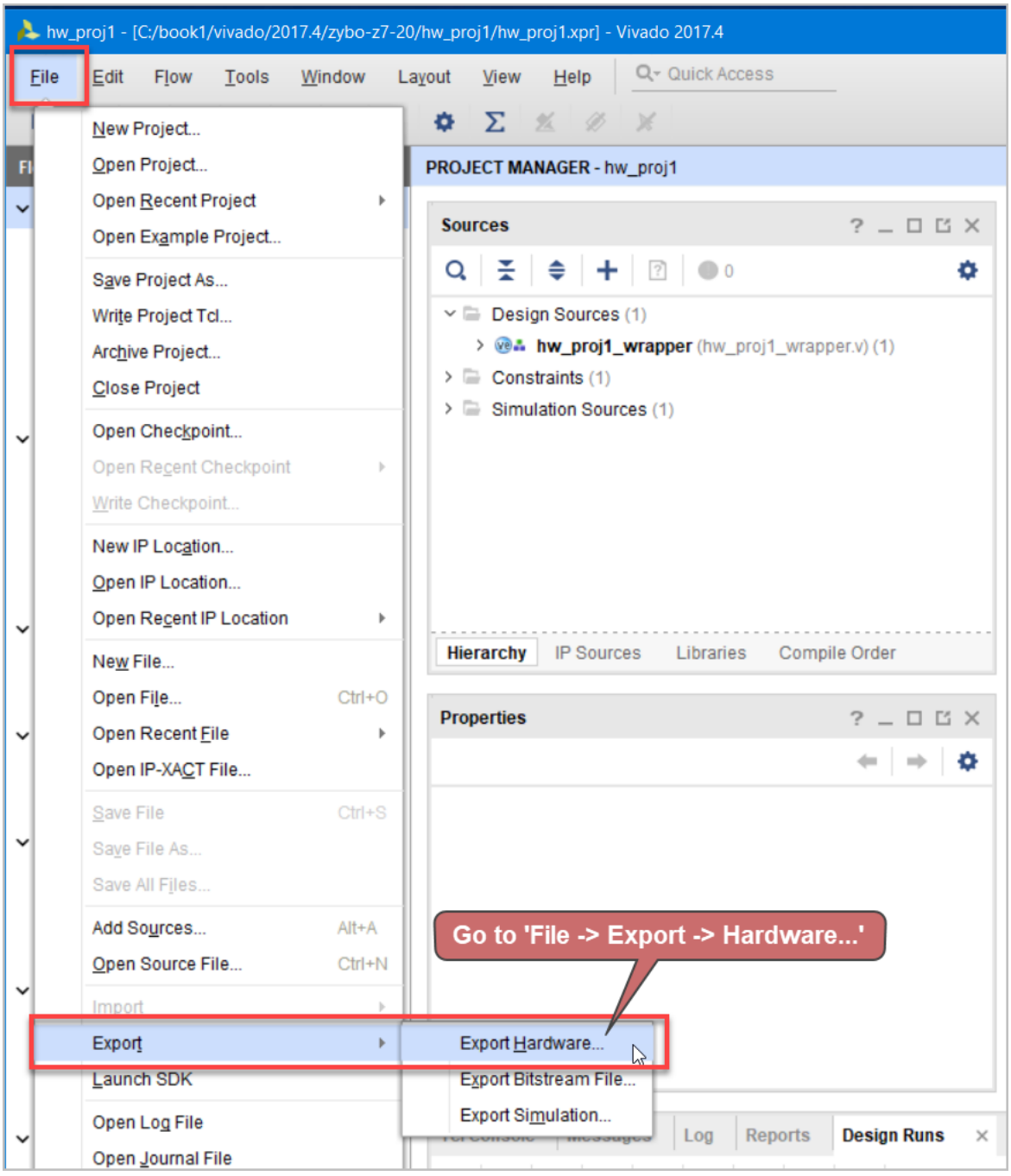


Figure 84. Export the design

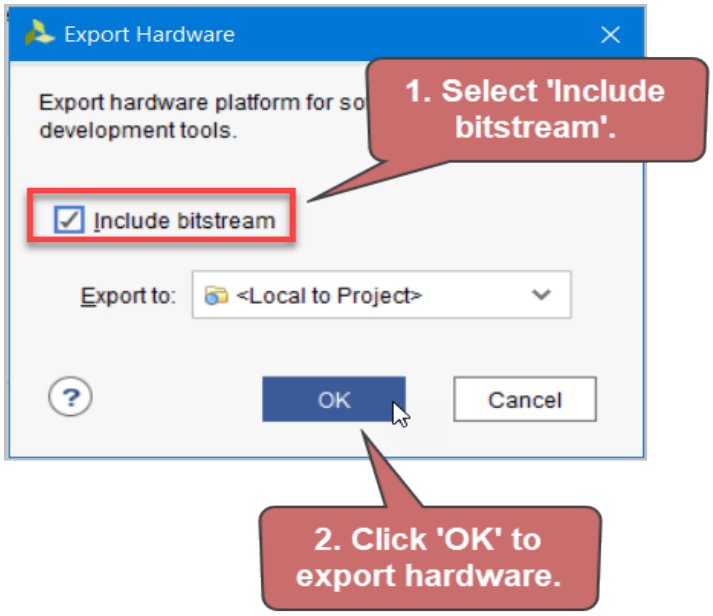


Figure 85. Include the bitstream and click OK.

2.5.2 Launch SDK

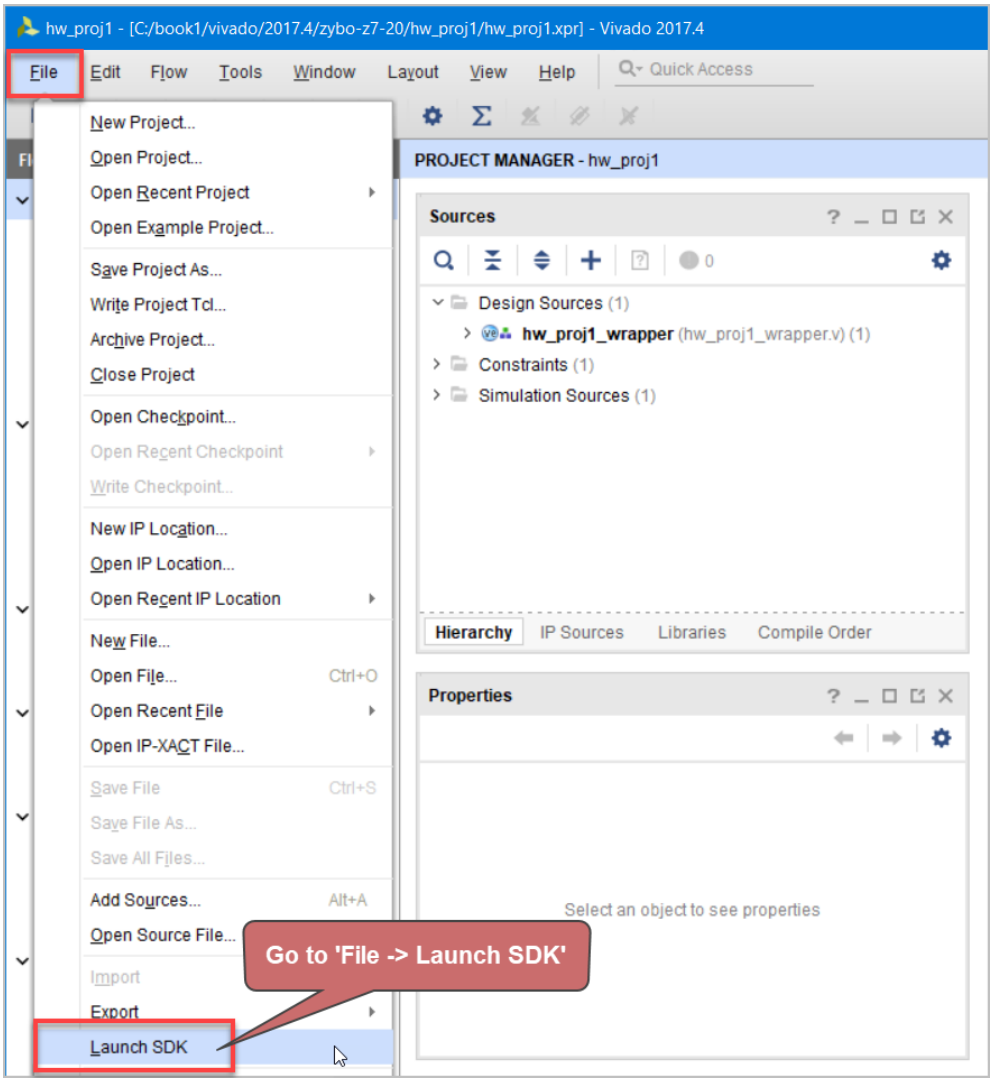


Figure 86. Launch SDK.

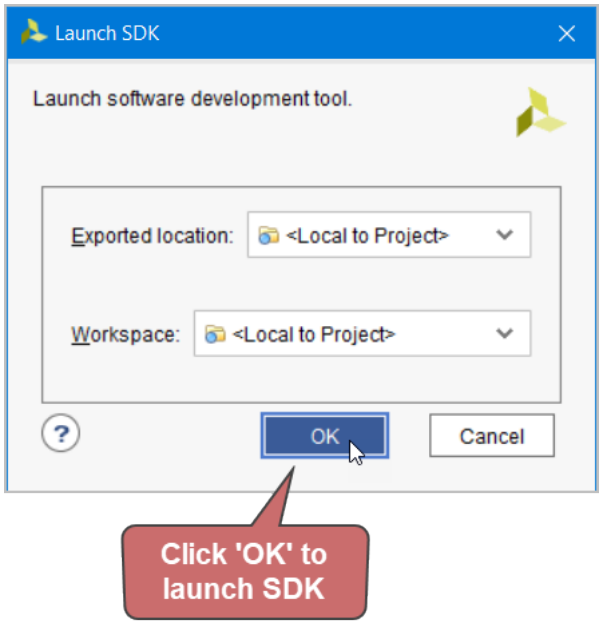


Figure 87. Click OK to launch SDK.

3 Software Development in Xilinx SDK

Figure 88. The Eclipse SDK opens after a short time (the user should not interfere during this time!). The hardware description file is automatically extracted in to the workspace.

3.1 Software Project 1: Hello World

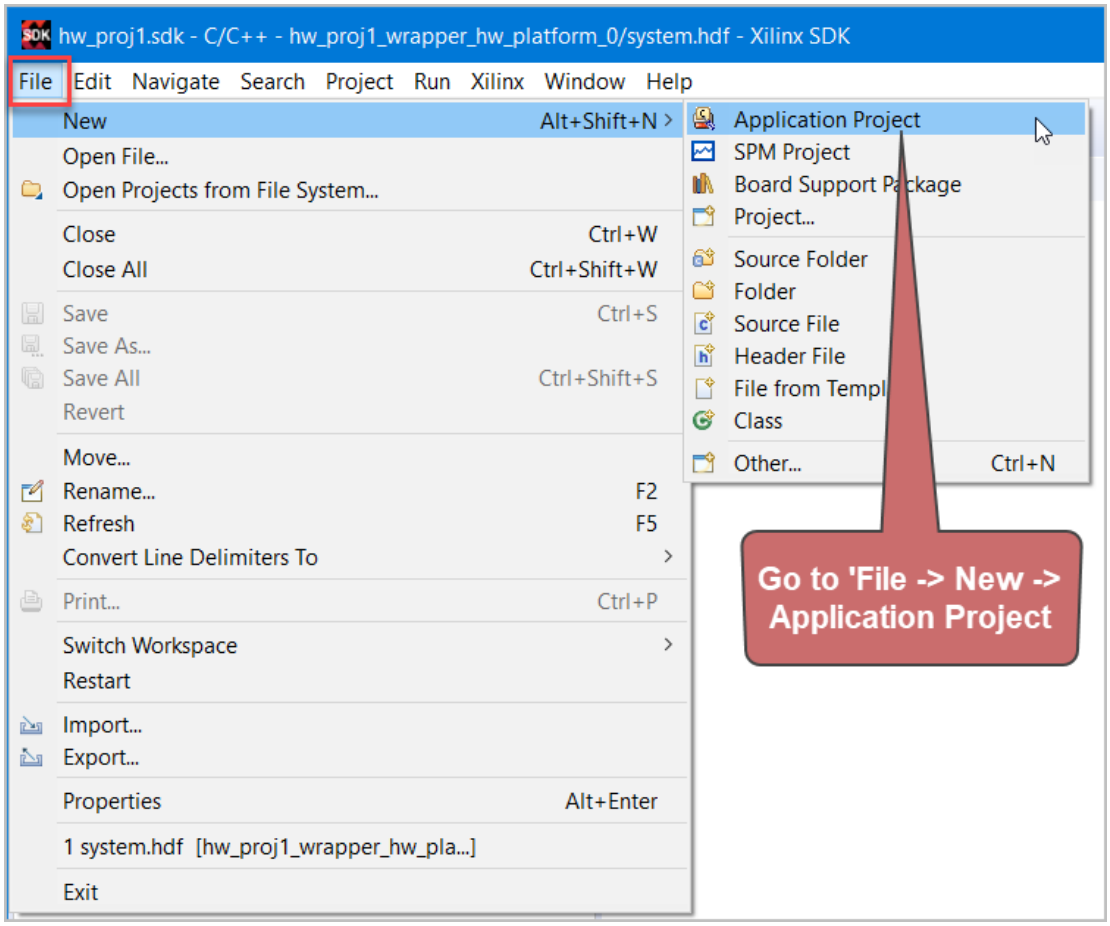


Figure 89. Go to File -> New -> Application Project

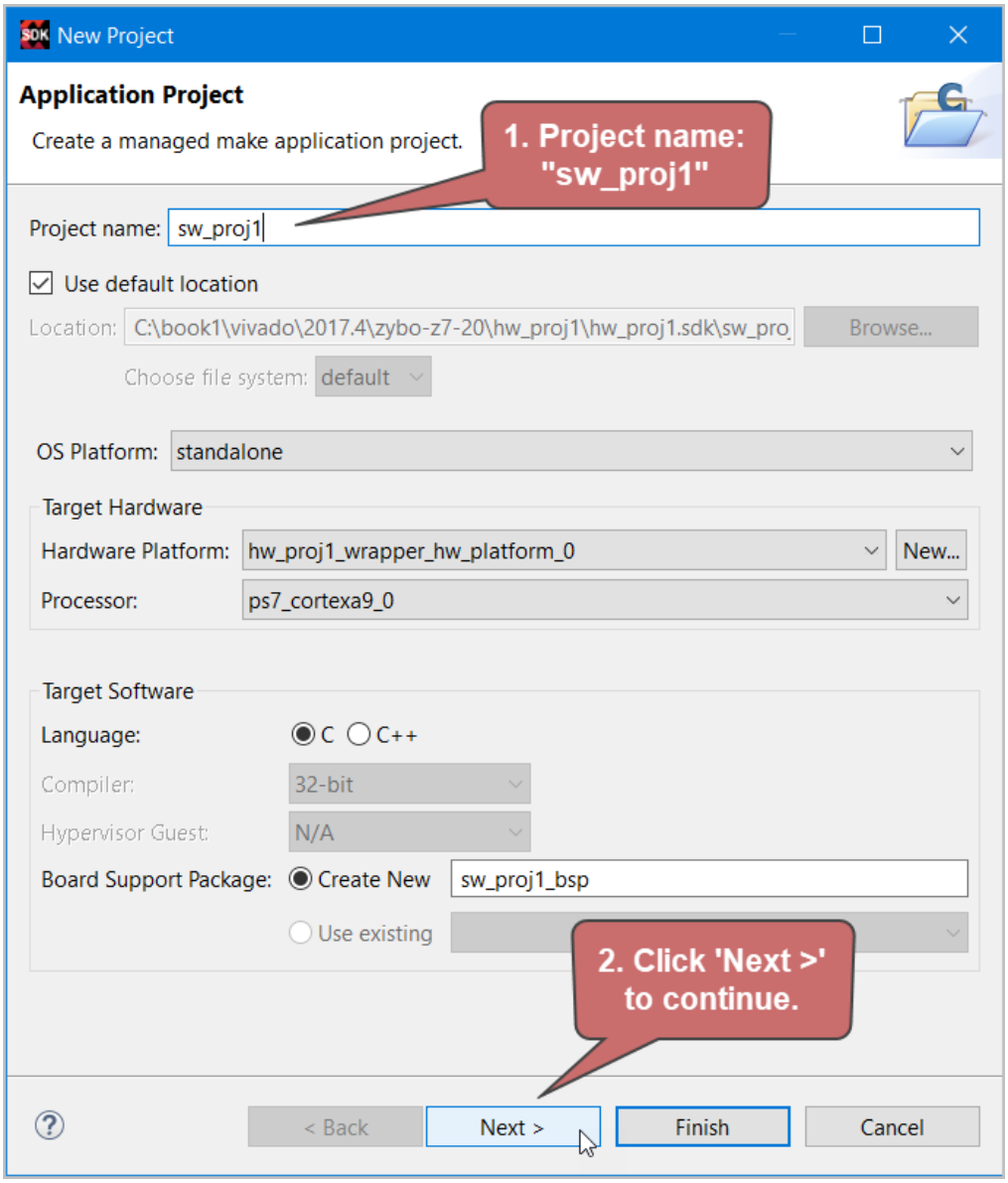


Figure 90. Call the project "sw_proj1" and click Next >.

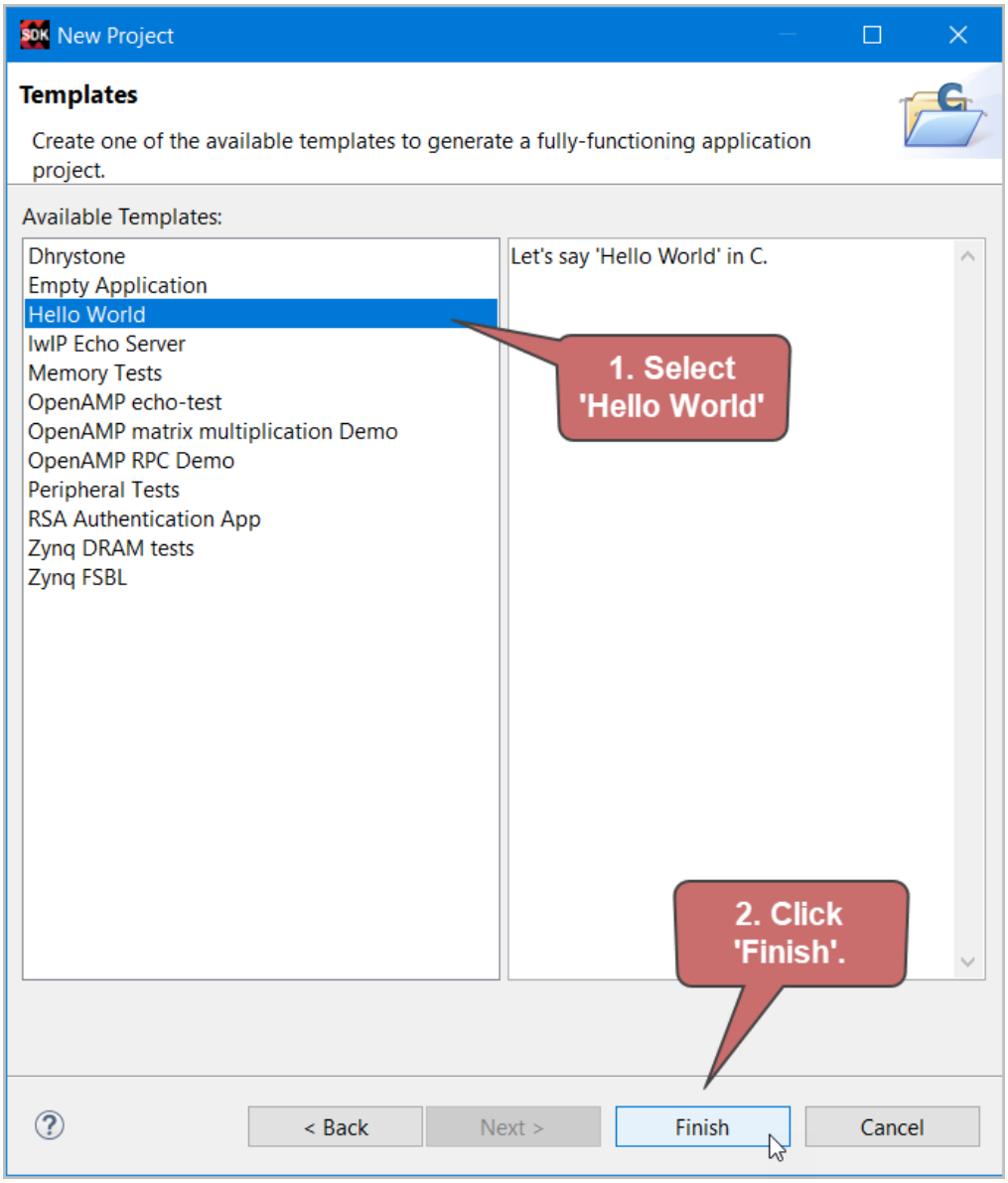


Figure 91. Ensure the Hello World template is selected, and click Finish.

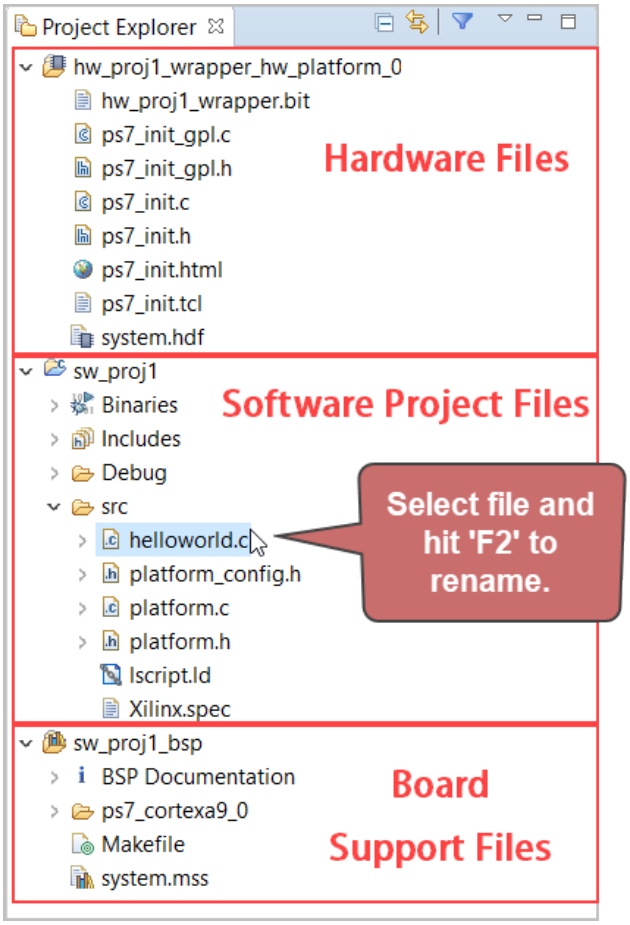


Figure 92. Rename the main source file

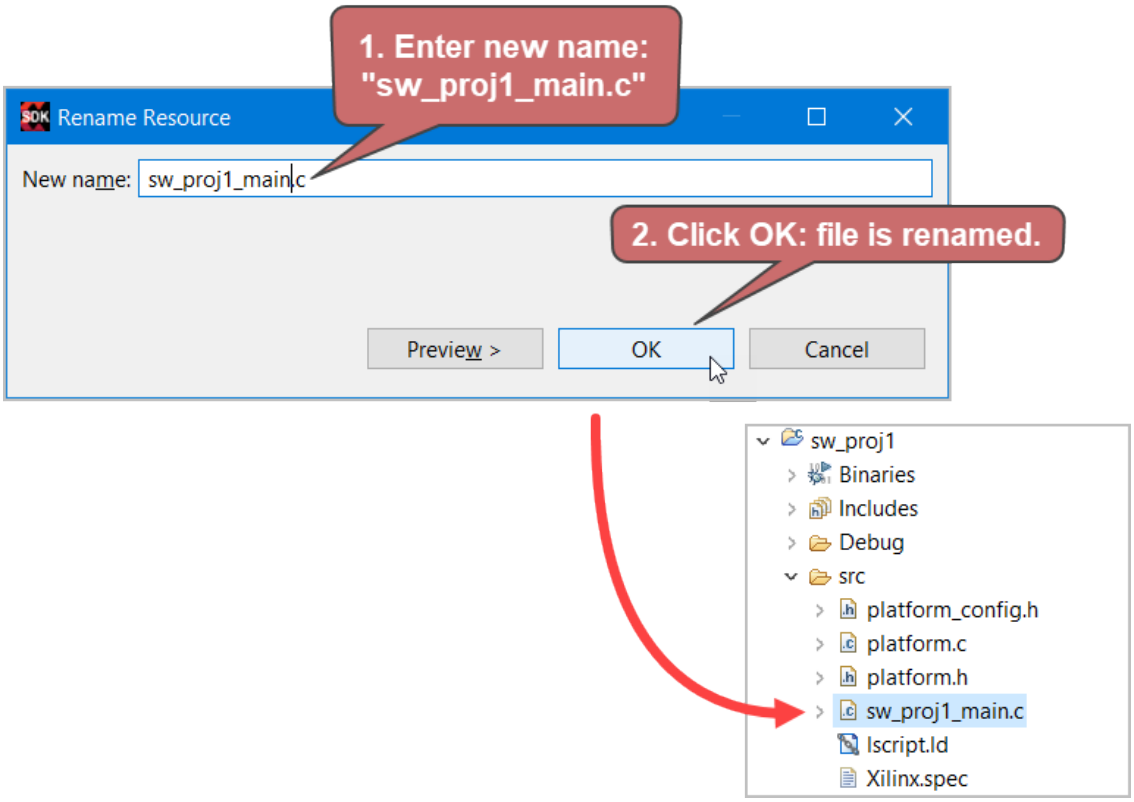


Figure 93. Change the project name to "sw_proj1_main.c"

```c
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"


int main()
{
    init_platform();

    print("Hello World\n\r");

    cleanup_platform();
    return 0;
}
```

3.1.1 Prepare the Platform (Zybo-Z7-2o or Zybo-Z7-10)

- Ensure that the boot mode is set to JTAG. (Header J5 must have a jumper across pins 1-2.)
- Connect a Micro-B USB cable to J12.
- Power up the platform (SW4).

3.1.2 Open Console Application

Here, Tera Term is used, although the console in the XSDK could also be used (or indeed any other suitable terminal program). Suitable settings are shown in the following figures. The terminal and serial port settings are important, but the window and font settings are optional.

Note that in this particular case, the platform is found on COM Port 6, but this will vary from system to system.

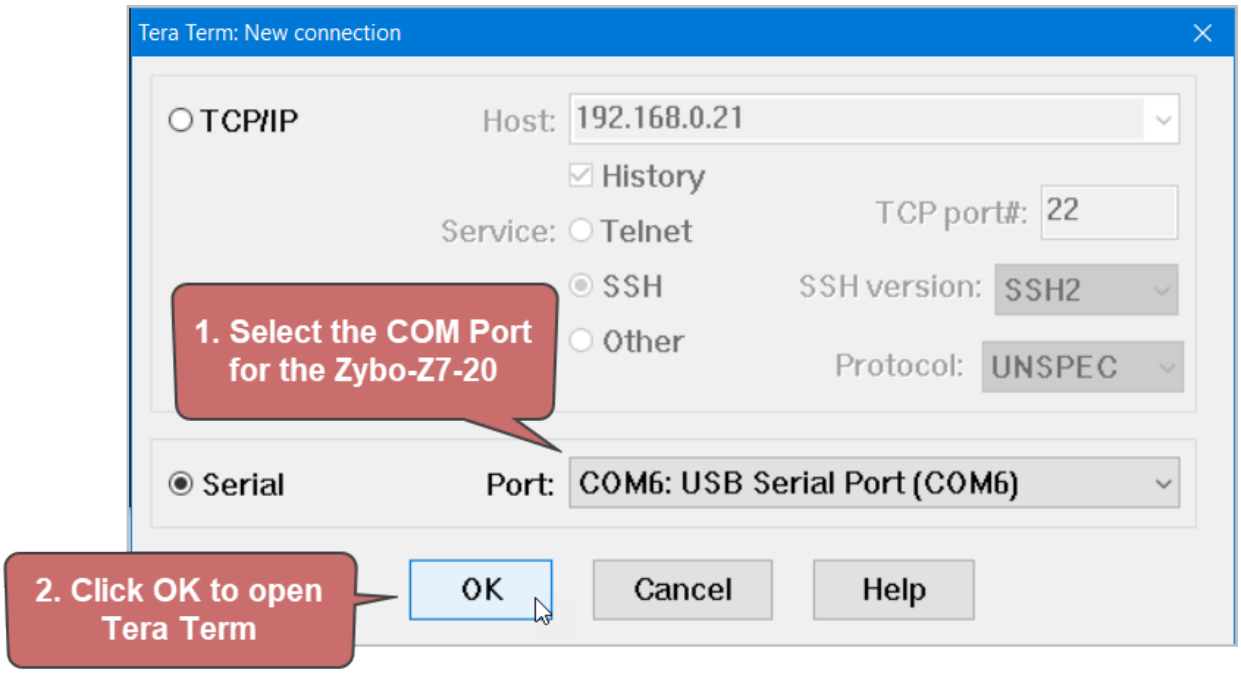


Figure 94. Open the connection to the platform.

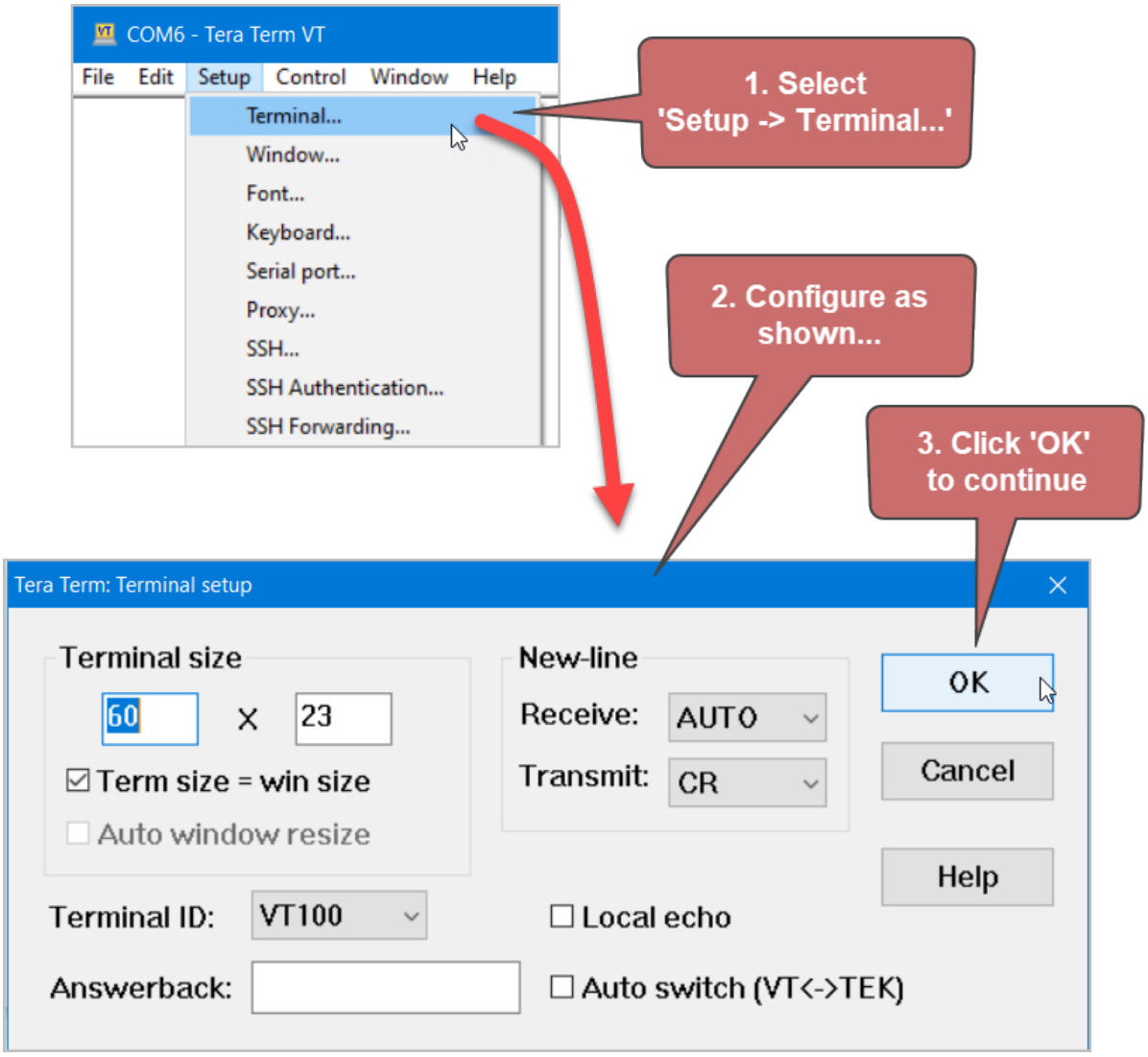


Figure 95. Configure the terminal. The New-line settings in particular are important.

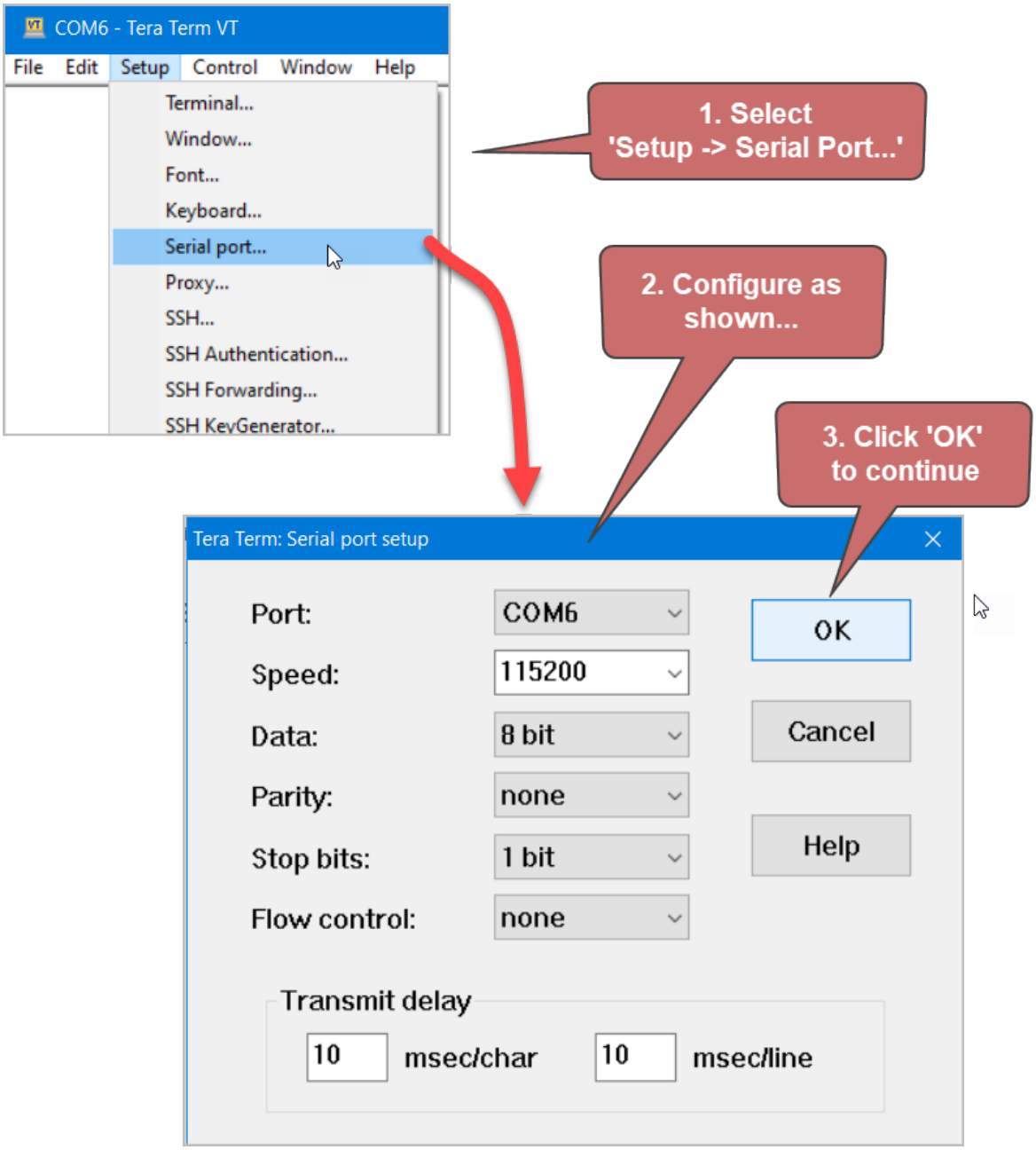


Figure 96. Configure the serial port settings.

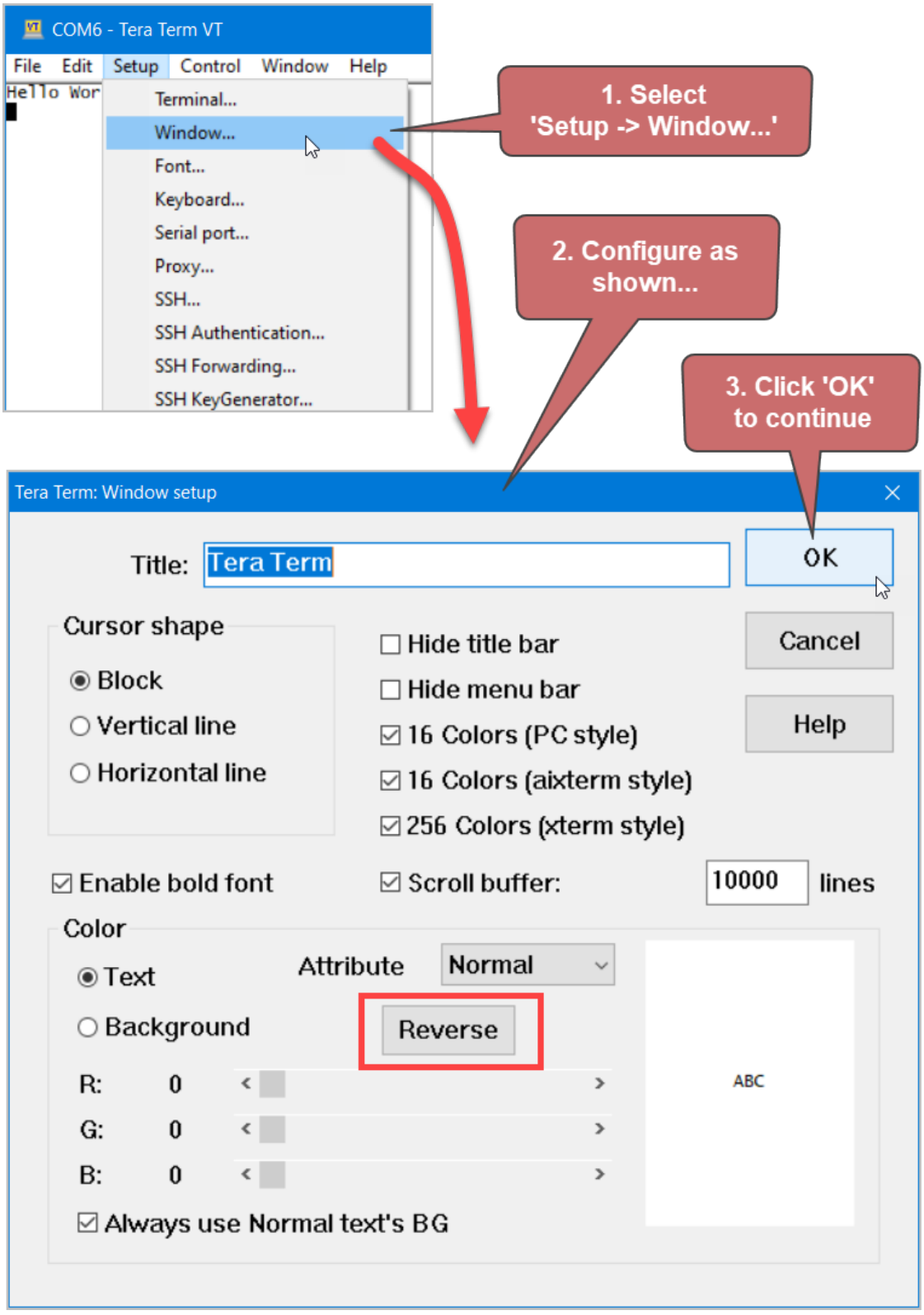


Figure 97. Optionally, reconfigure the window with a white background and black text.

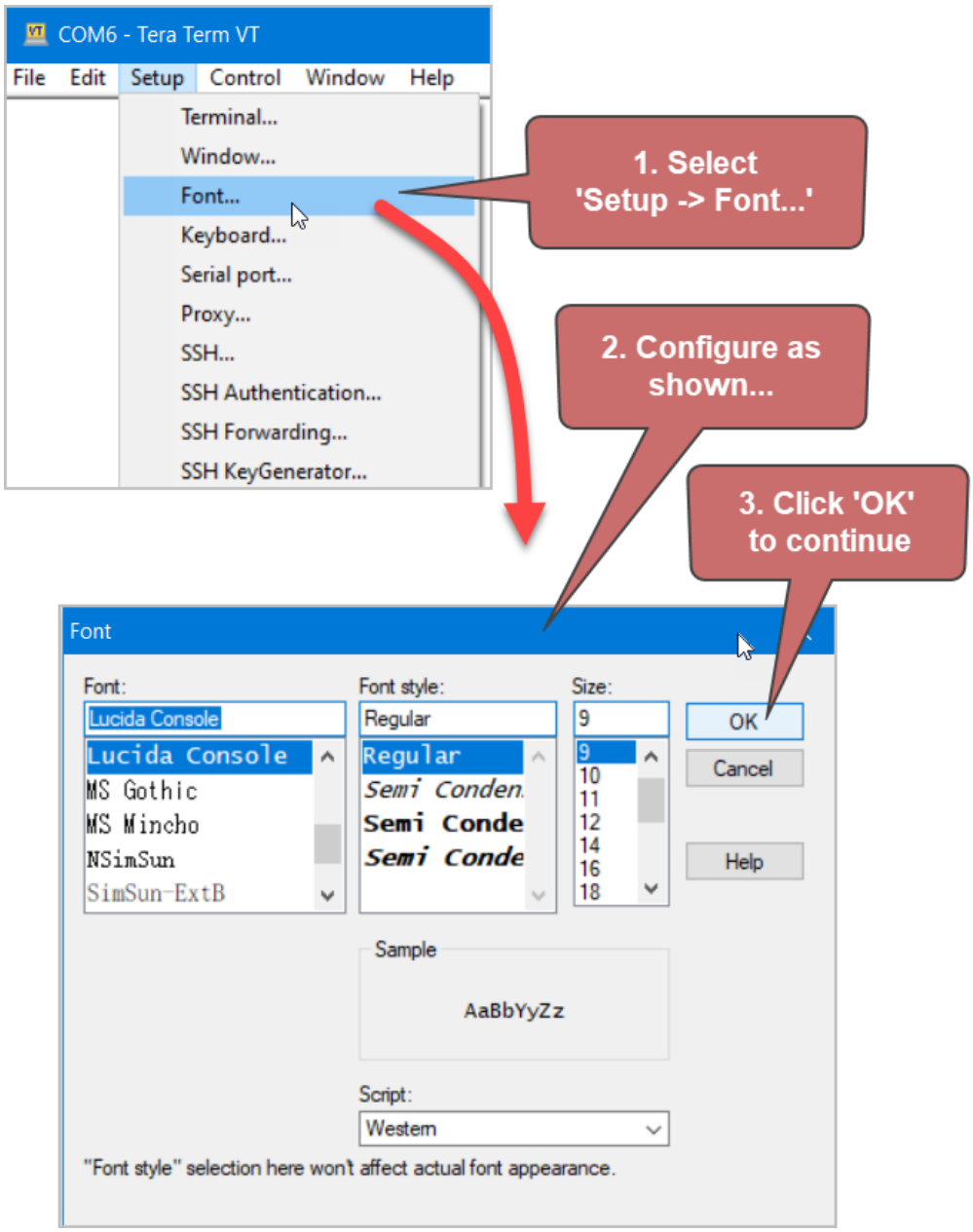


Figure 98. Optionally, change the font.

3.1.3 Run the Program

First, open the XSCT console. This is not a necessary step, but it does provide useful information on what happens when the program is launched.

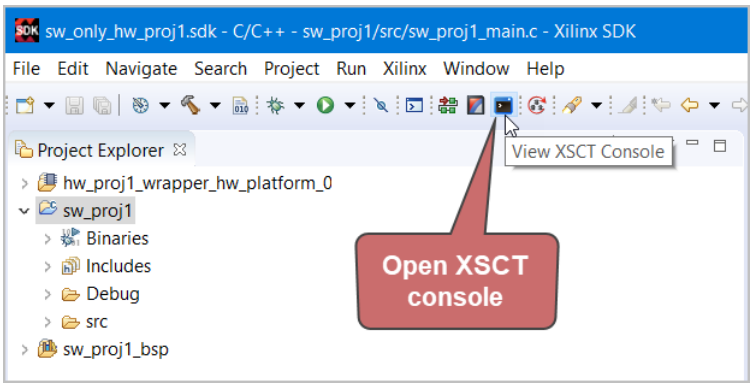


Figure 99. Open the XSCT console

To run the program, the first step is to create a Run configuration. Right-click on the project, and select 'Run As -> Run Configurations...' option.

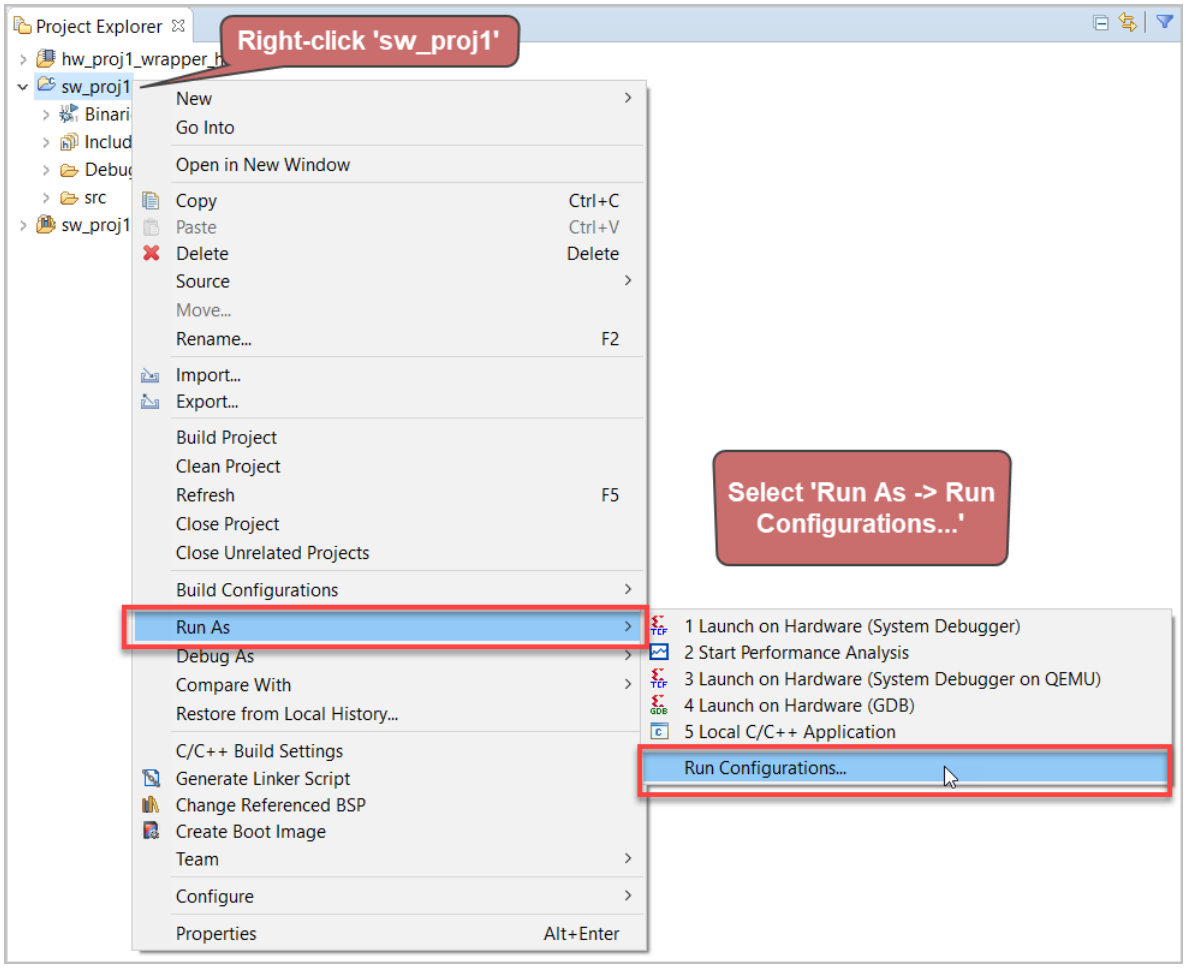


Figure 100. Right-click the software project to create a Run Configuration

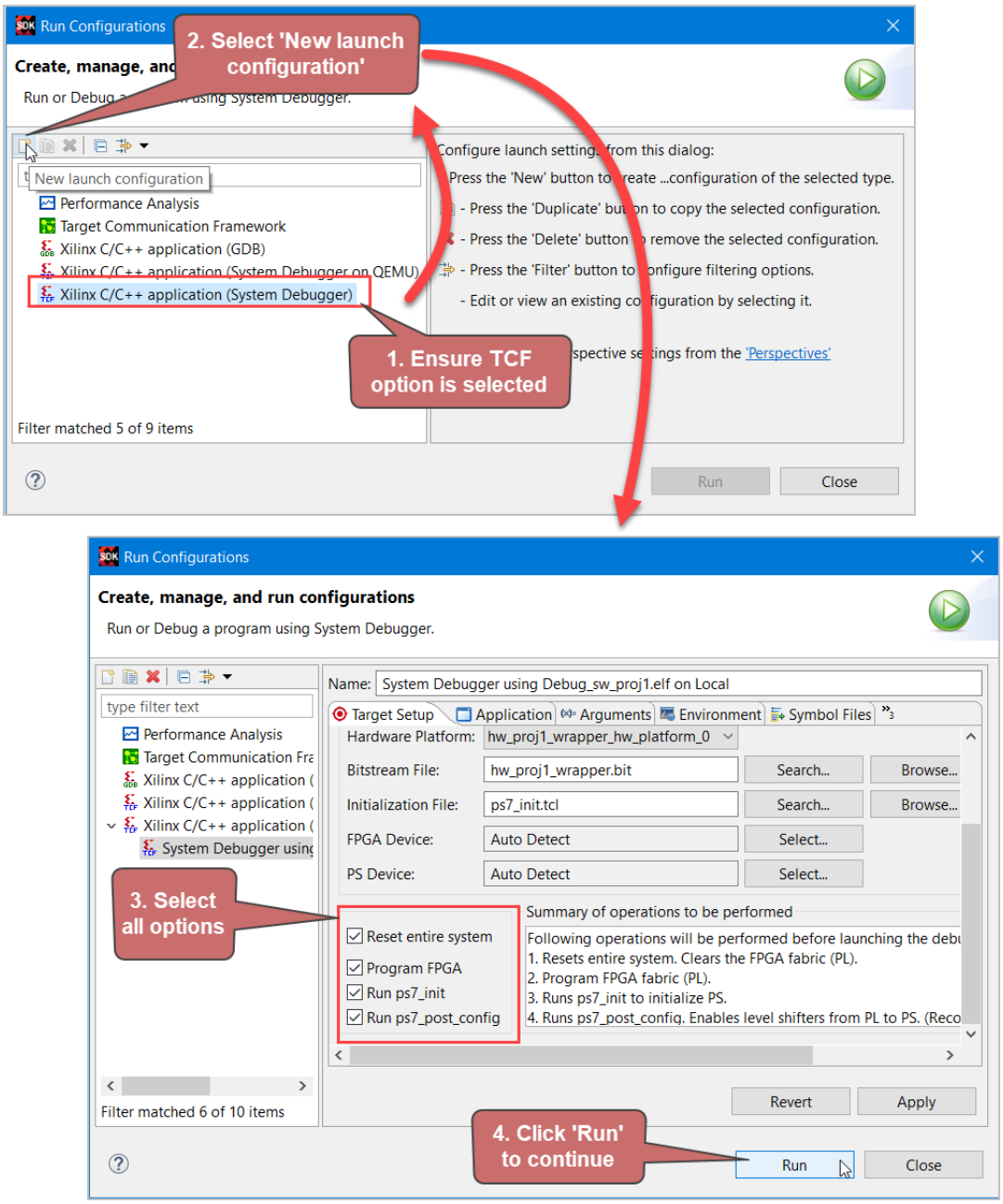


Figure 101. Create a TCF (i.e. System Debugger) configuration, and execute the program.

1. Ensure the Xilinx C/C++ application (System Debugger) is selected.
2. Click on New Launch Configuration.
3. Select all options:
    a. Reset entire system
    b. Program FPGA
    c. Run ps7_init
    d. Run ps7_post_config

4. Click on 'Run' to continue.

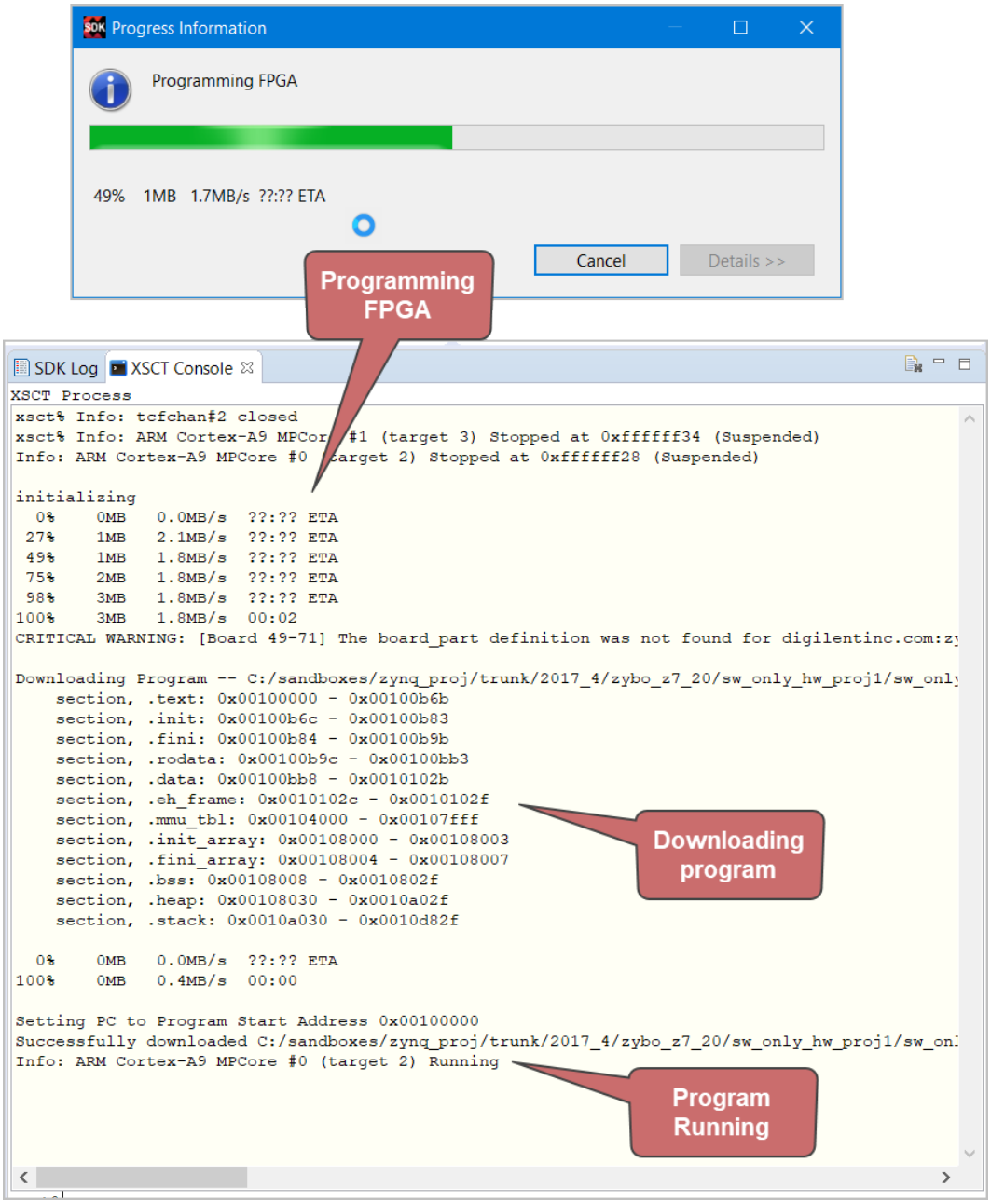


Figure 102. The FPGA is programmed, and the application is downloaded.

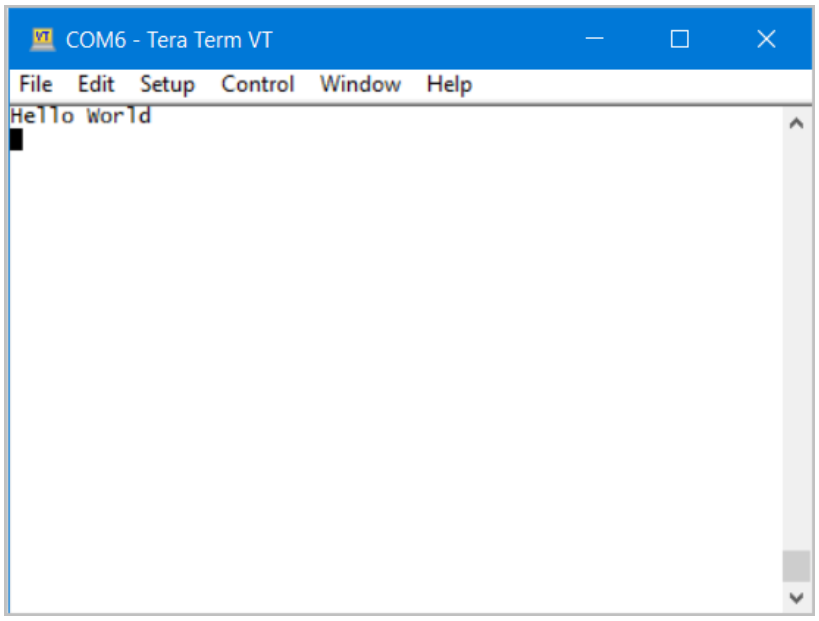


Figure 103. The program output is displayed in the console.

The “Hello World” message is displayed in the console.

3.2 Software Project 2: Zynq GPIO

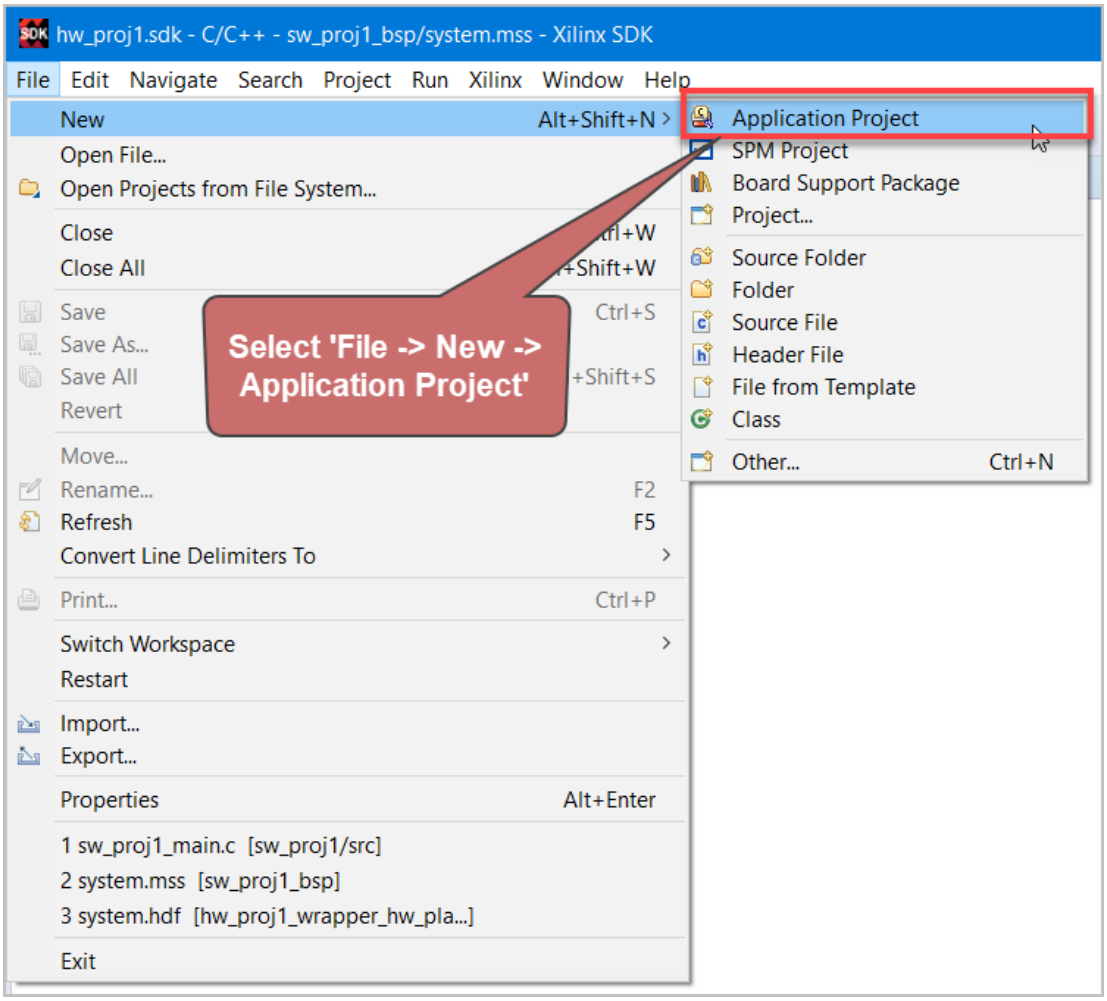


Figure 104. Select File->New-> Application Project

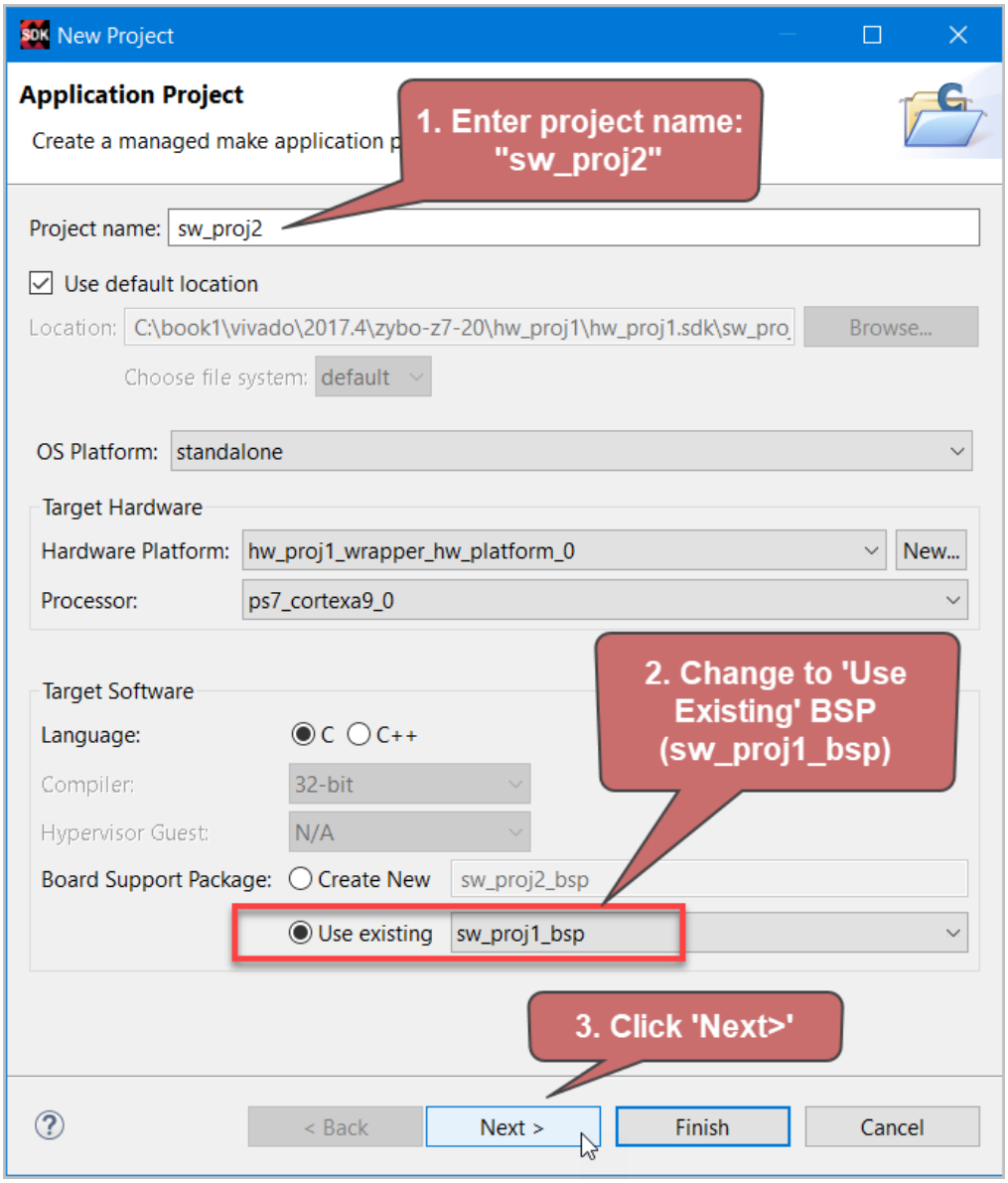


Figure 105. Enter the project details (part 1)

1. Enter the project name: sw_proj2
2. Board Support Package: Use existing
3. Click Next (don't click Finish!)

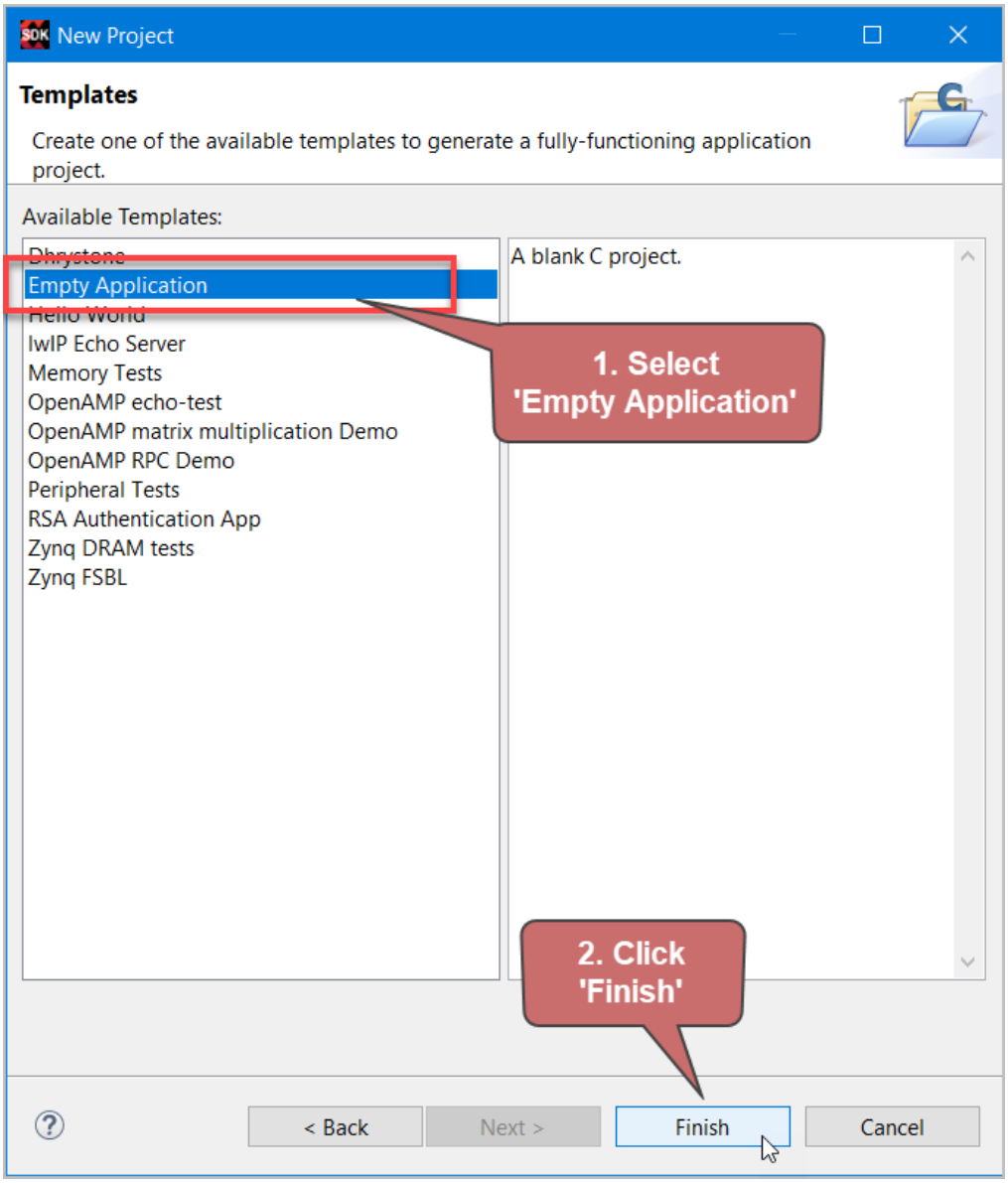


Figure 106. Enter the project details (part 2)

1. Select Empty Application
2. Click Finish

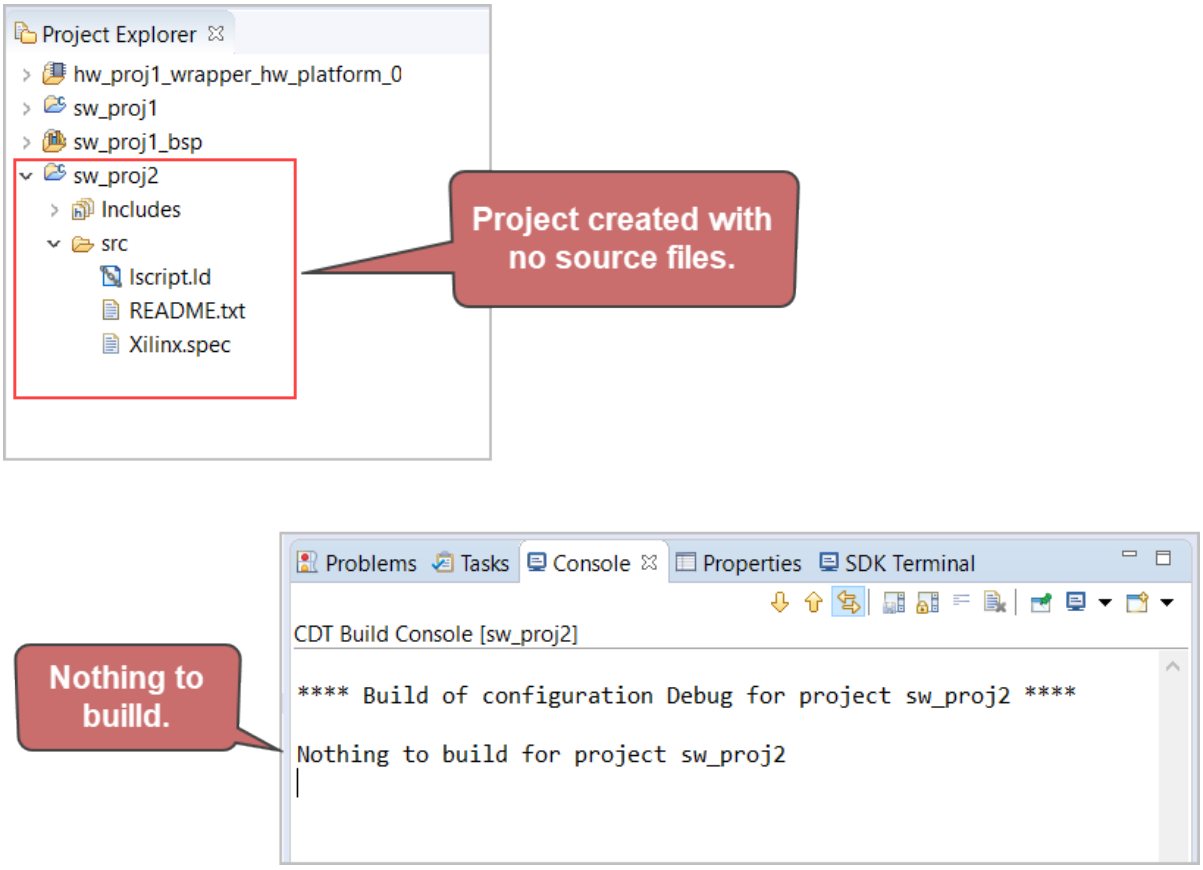


Figure 108. Project created with no source files

A project named “sw_proj2” is created with no project files. The CDT Build Console confirms that there is nothing to build.

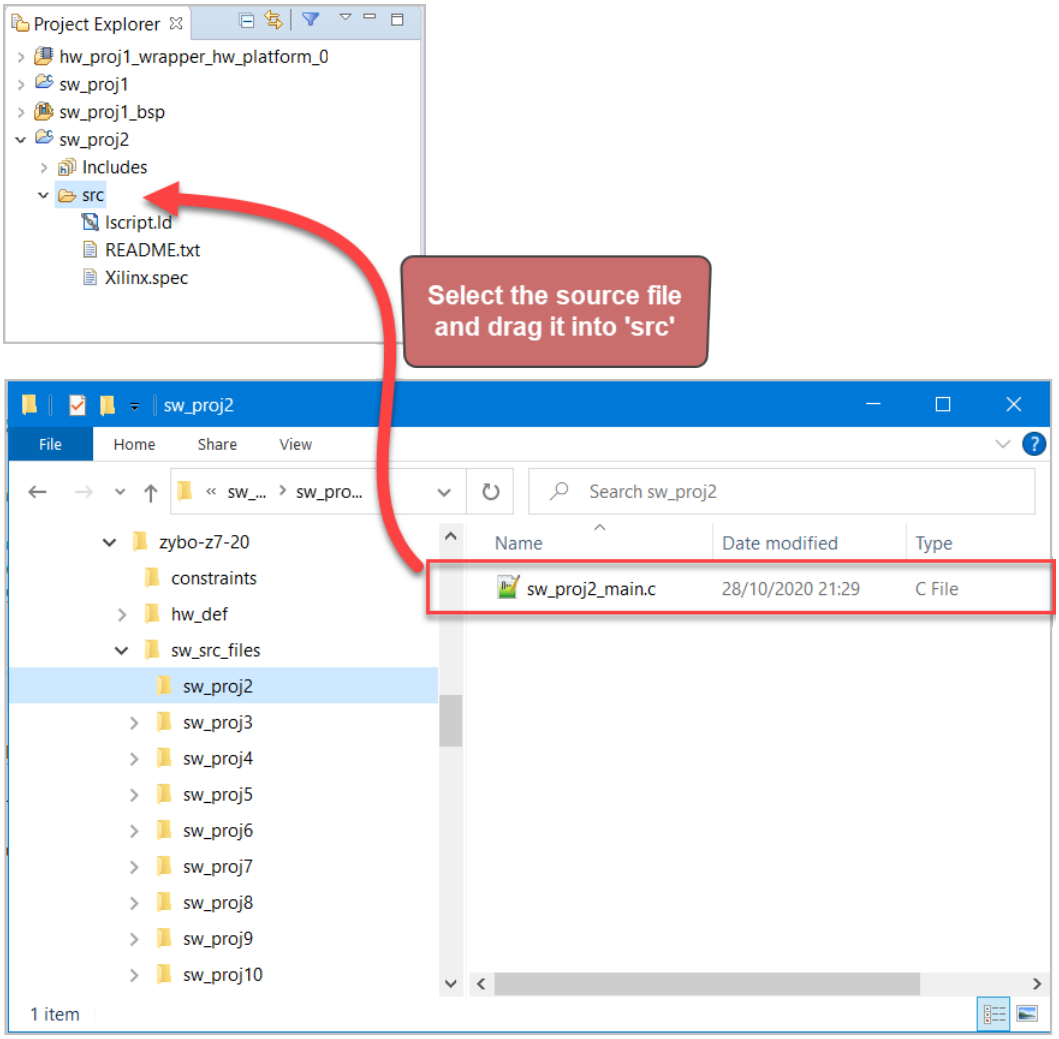


Figure 109. Drag the source file from Windows Explorer into the SDK project

Open the location where the source file for software project 2 is saved on your PC. Drag the file from Windows explorer directly into the "**src**" folder in SDK.

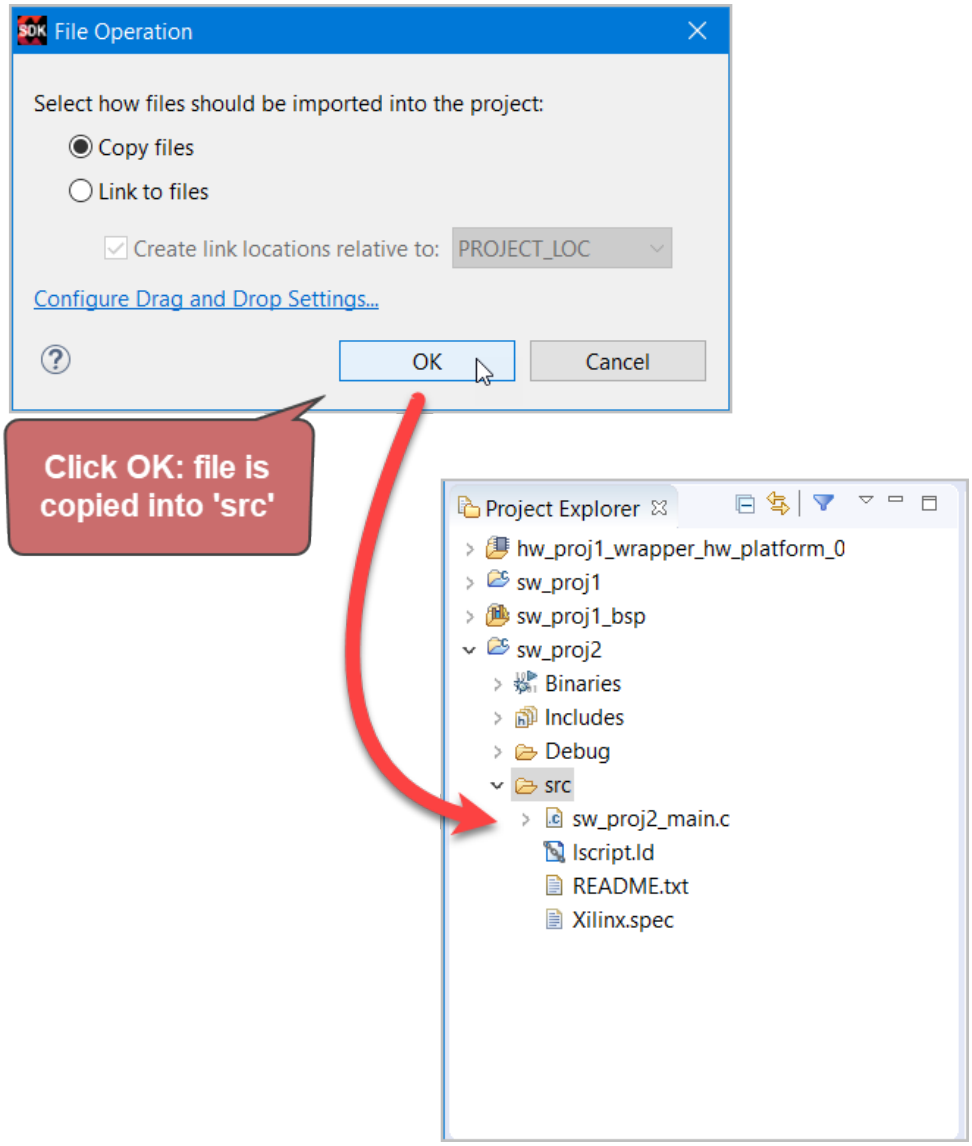


Figure 110. Click OK, and the file will be copied into the project.

In the File Operation dialog box, ensure “Copy files” is checked, and click OK.

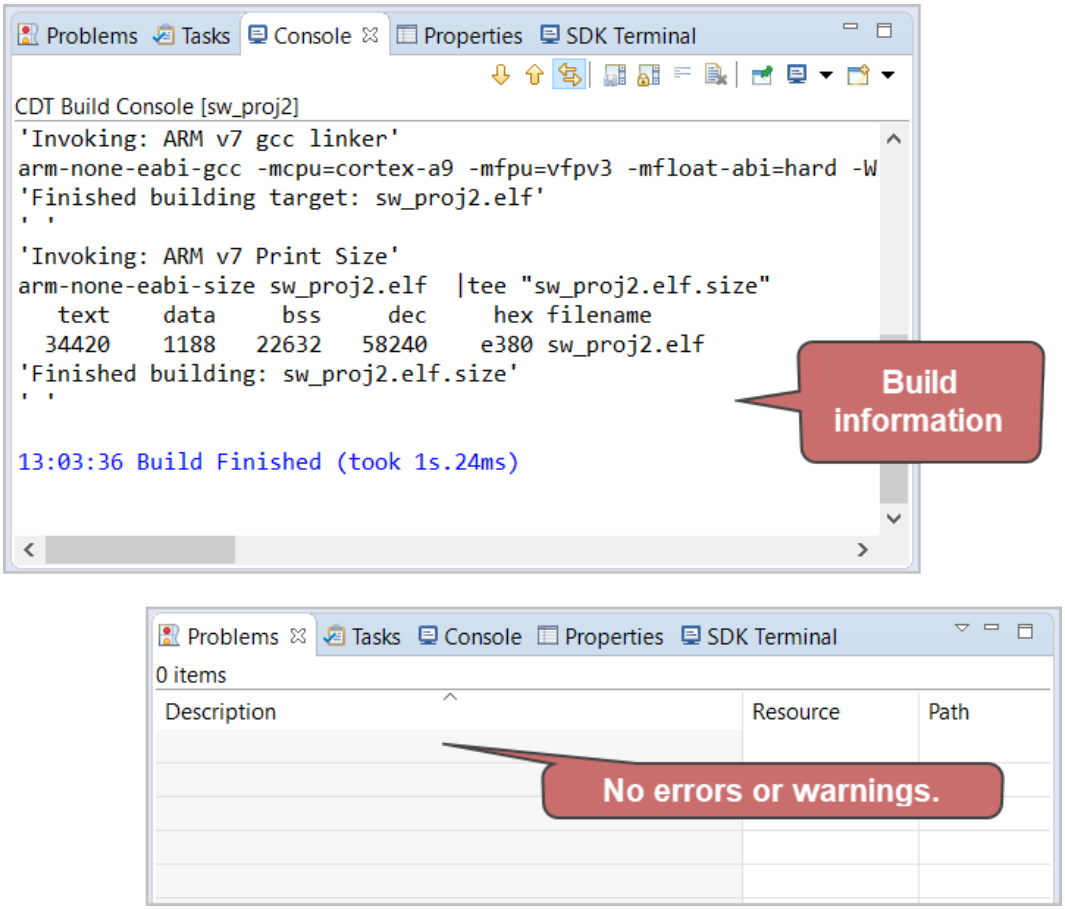


Figure 111. The project should build successfully

If "Build Automatically" is selected, the project should build successfully, with no errors or warnings.

[Exceptions: SDK 2018.3/SDK 2019.1 may indicate a warning as follows, which can be ignored:

#pragma message: For the sleep routines, Global timer is being used]

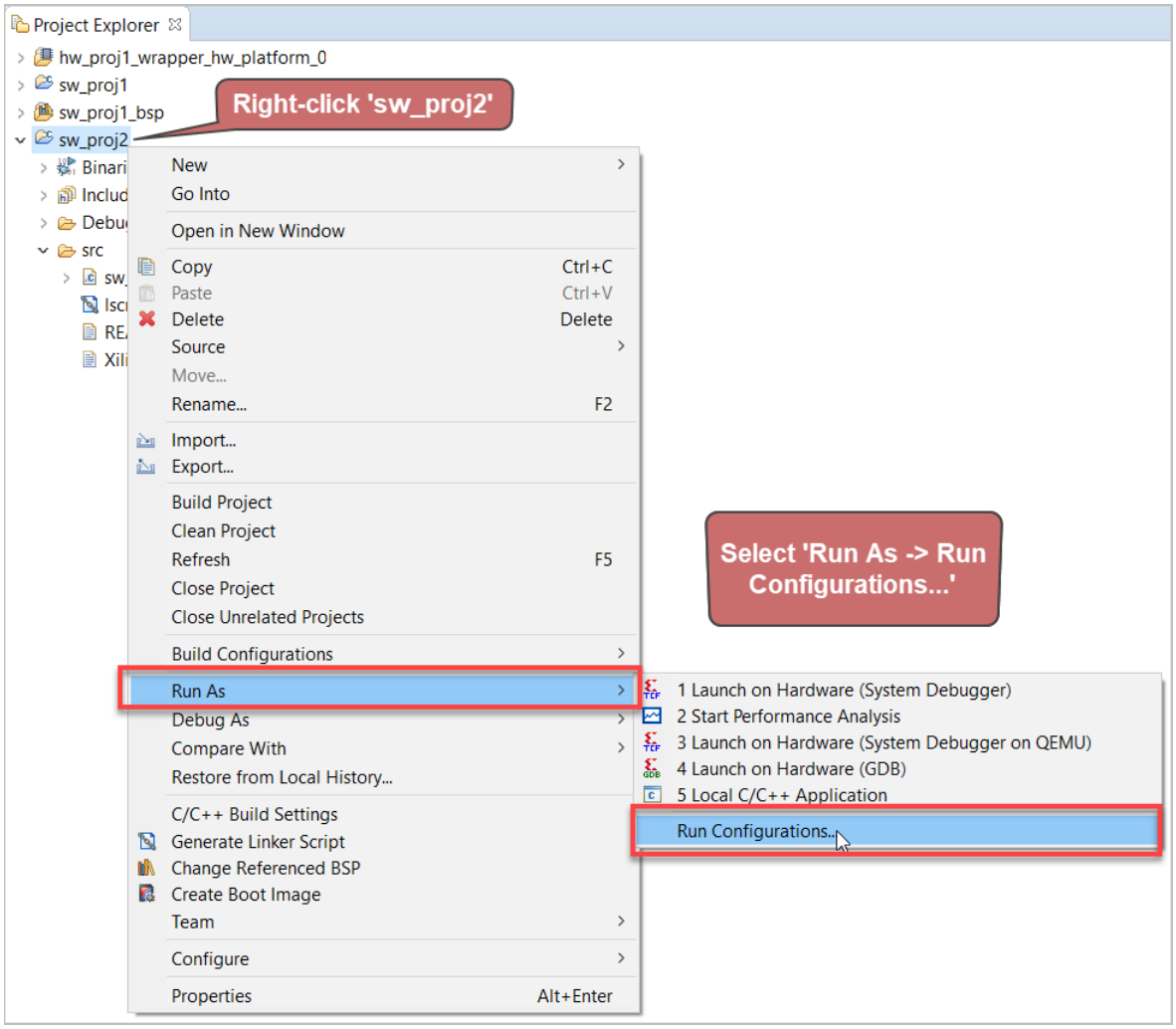


Figure 112. Right-click the software project and create a Run Configuration

To run the program, the first step is to create a Run configuration. Right-click on the project, and select the 'Run Configurations...' option.

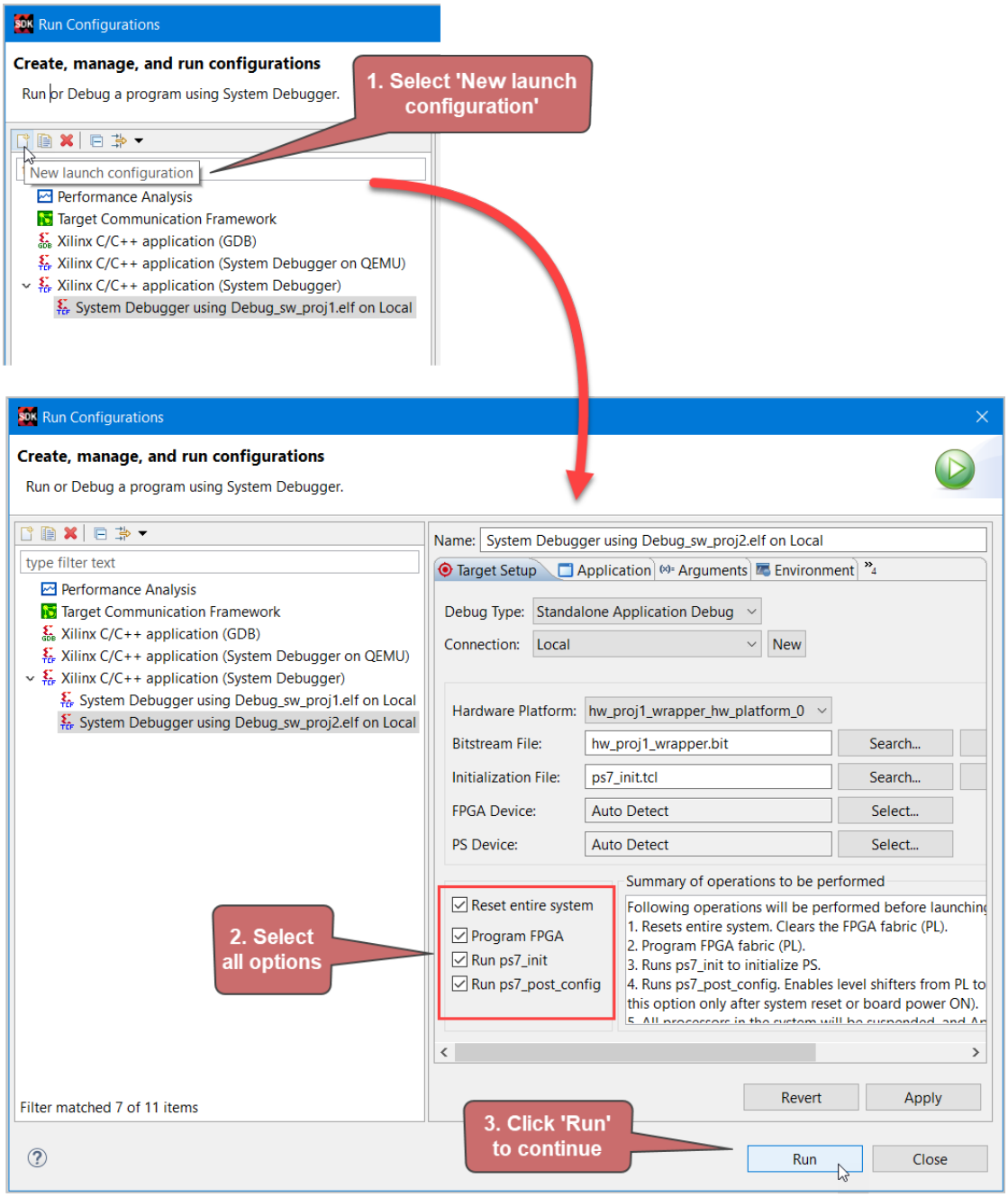


Figure 113. Create a TCF (i.e. System Debugger) option and click Run to execute the program.

1. Ensure the Xilinx C/C++ application (System Debugger) is selected.
2. Click on New Launch Configuration.
3. Select all options:
    a. Reset entire system
    b. Program FPGA
    c. Run ps7_init
    d. Run ps7_post_config

4. Click on 'Run' to continue.

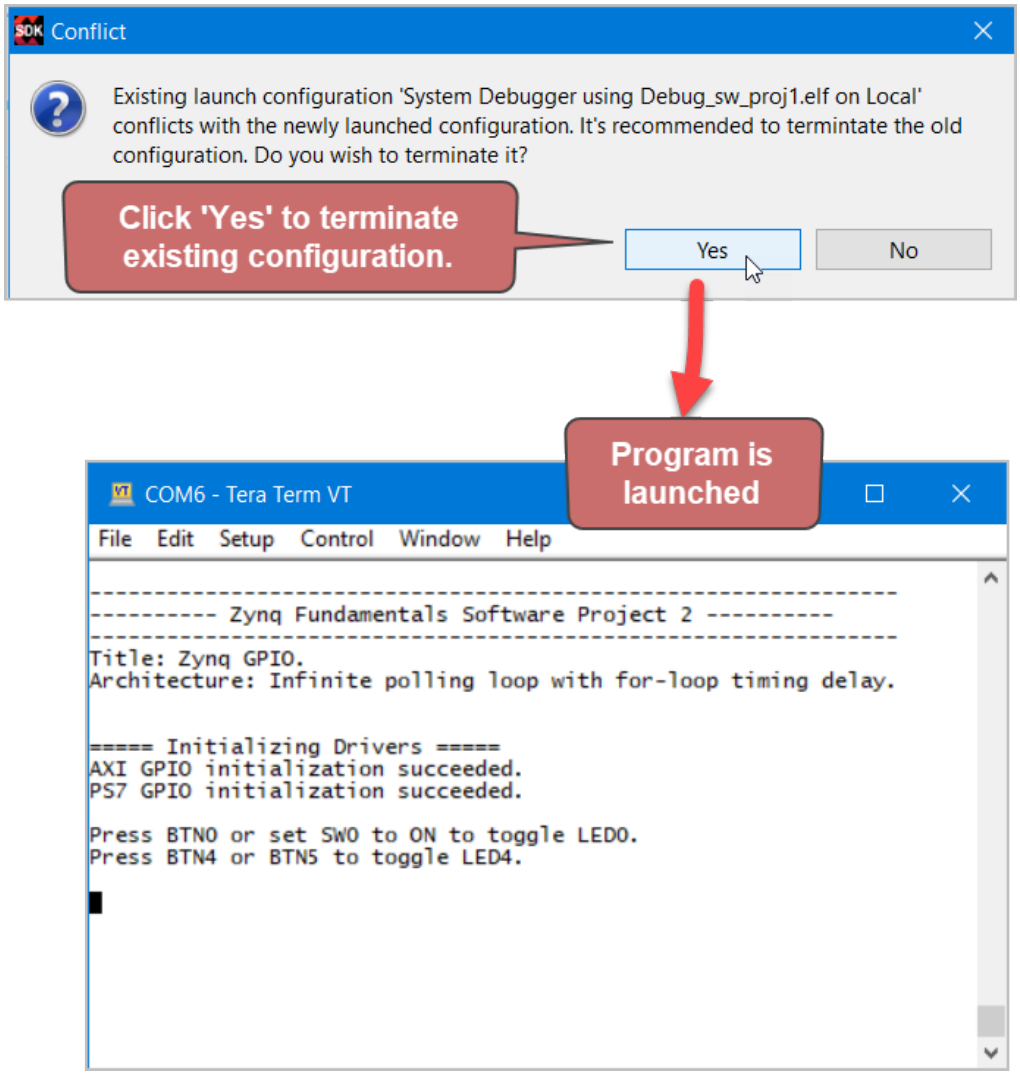


Figure 114. Terminal output for Software Project 2

If prompted, select 'Yes' to terminate any existing configuration. The FPGA will be programmed and the application will be downloaded to the processor. The terminal program should display the results for launching Software Project 2.

3.3 Software Project 3: Structured Program

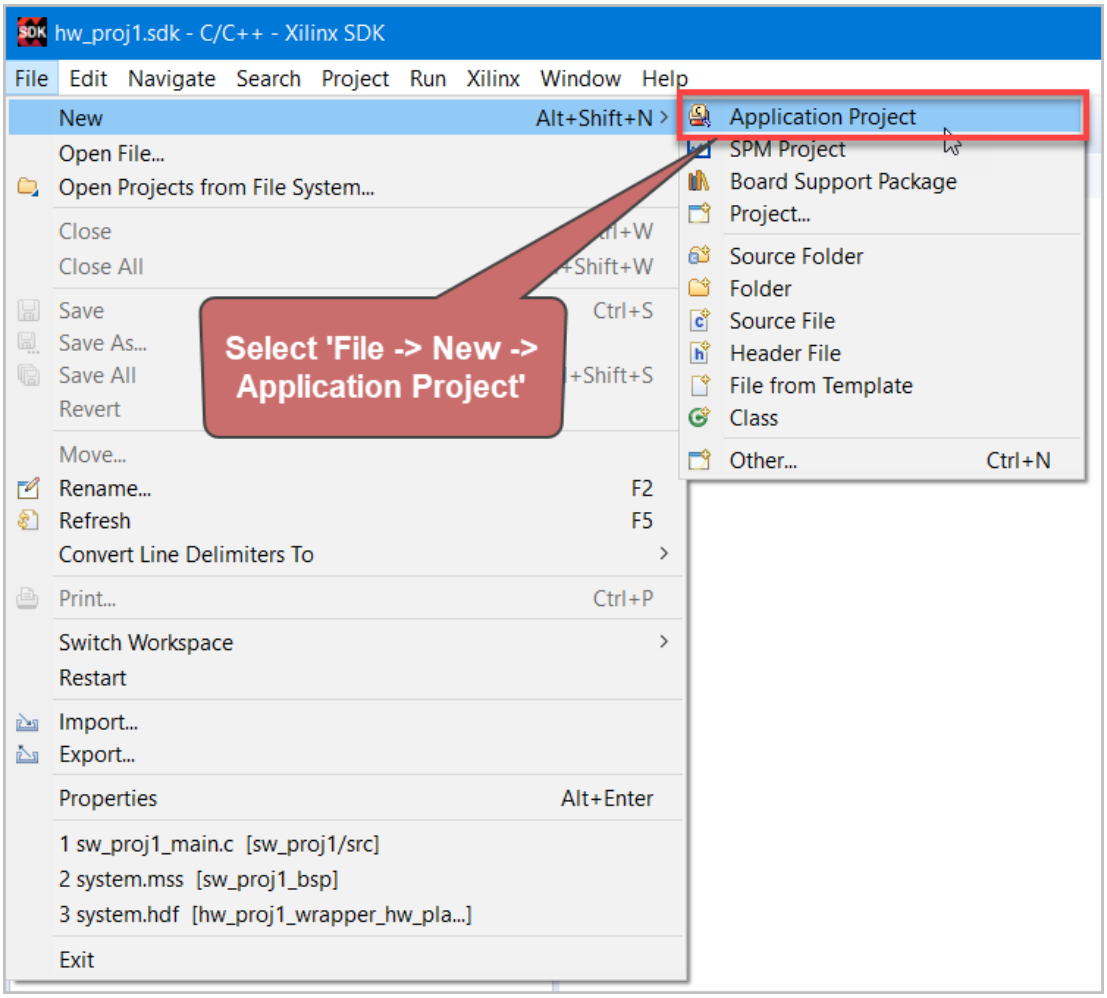


Figure 115. Select File->New-> Application Project

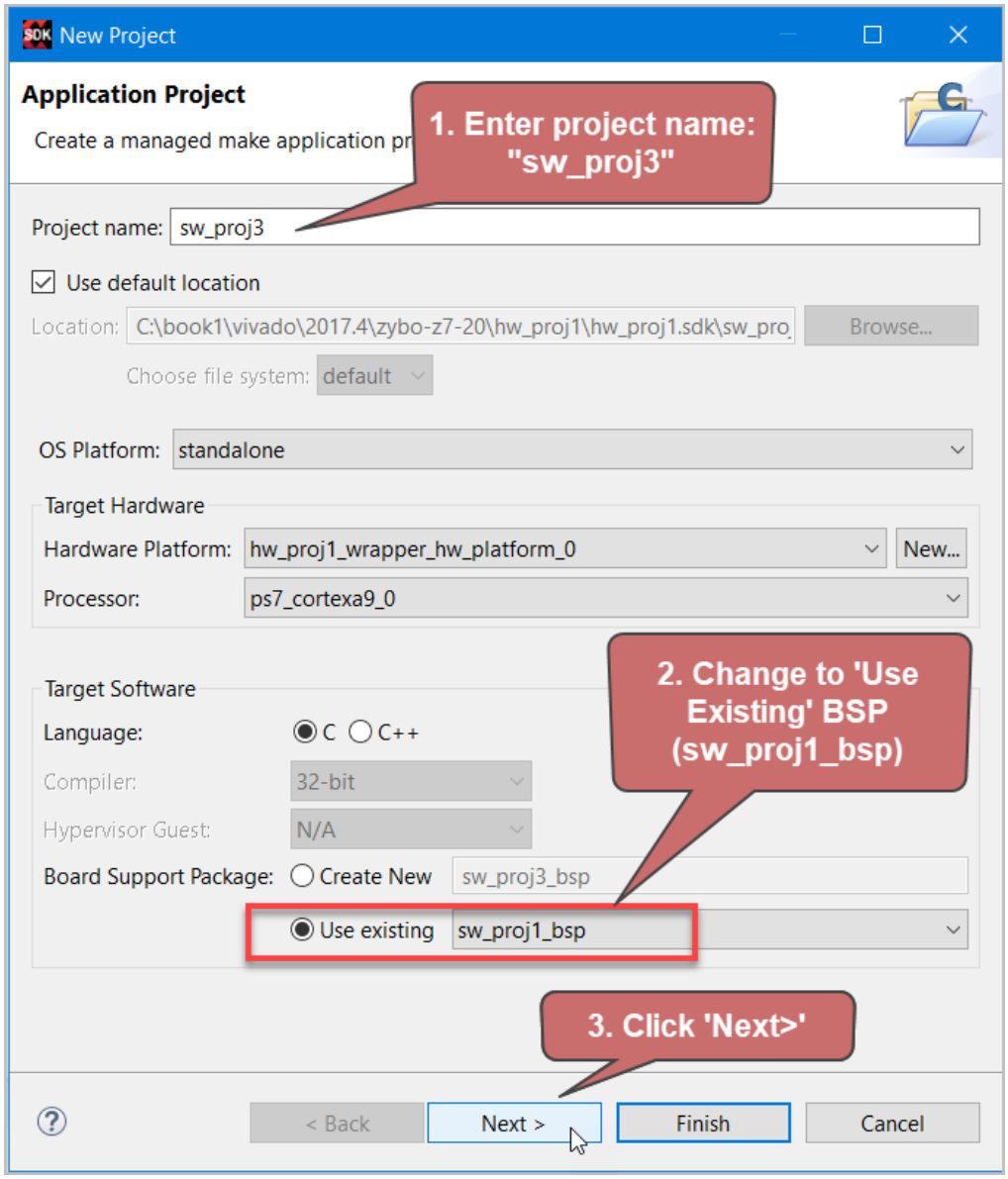


Figure 116. Enter the project details (part 1)

1. Enter the project name: sw_proj3
2. Board Support Package: Use existing
3. Click Next (don't click Finish!)

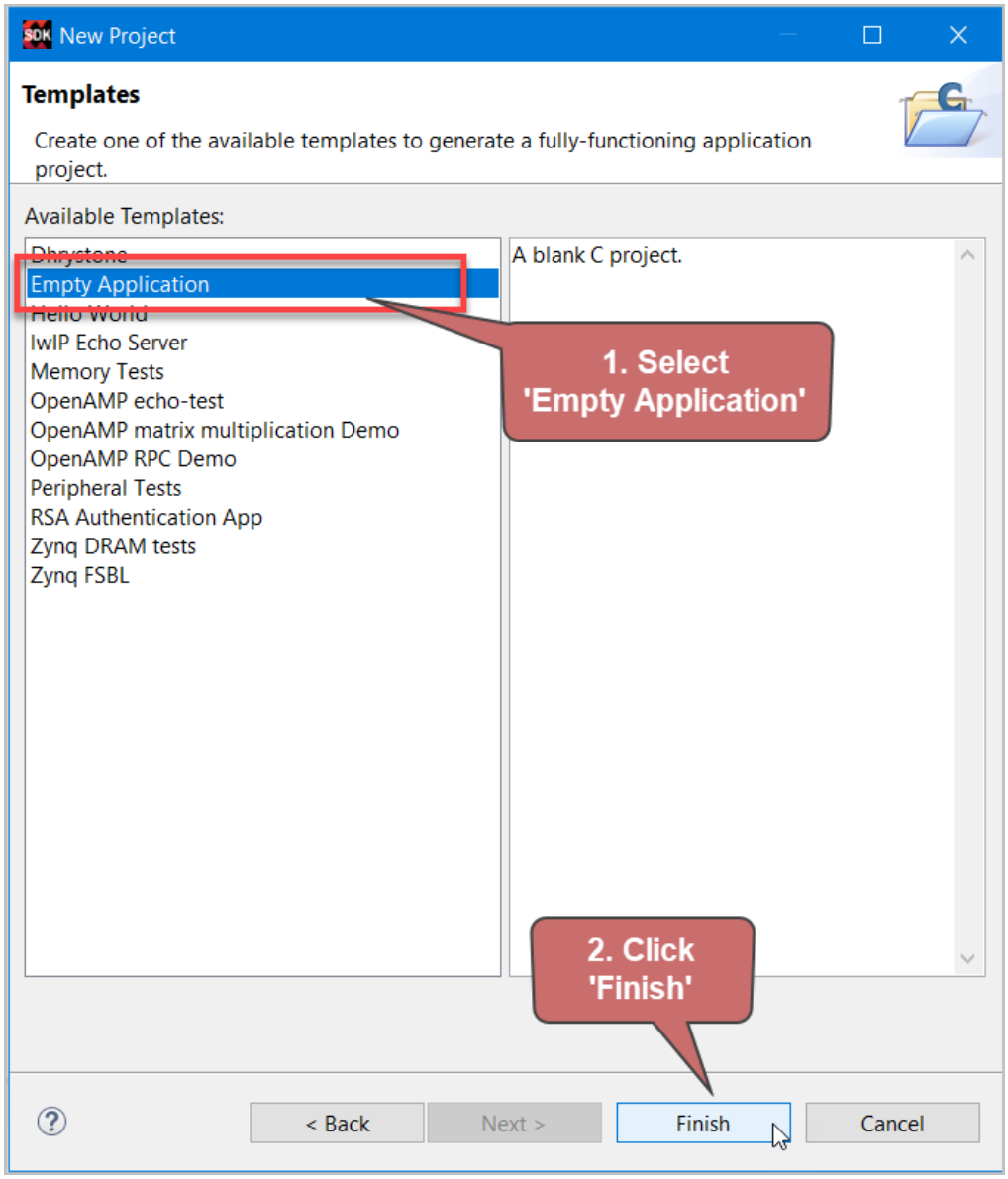


Figure 117. Enter the project details (part 2)

1. Select Empty Application
2. Click Finish

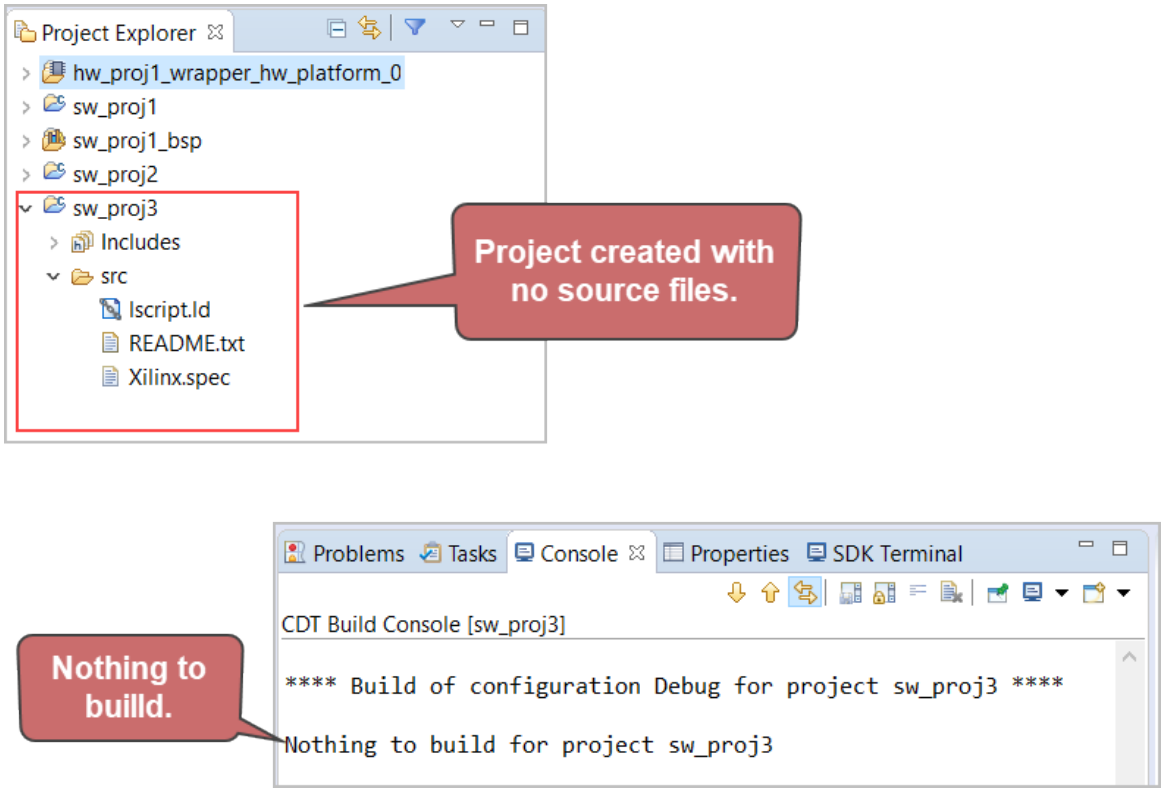


Figure 119. Project created with no source files

A project named “sw_proj3” is created with no project files. The CDT Build Console confirms that there is nothing to build.

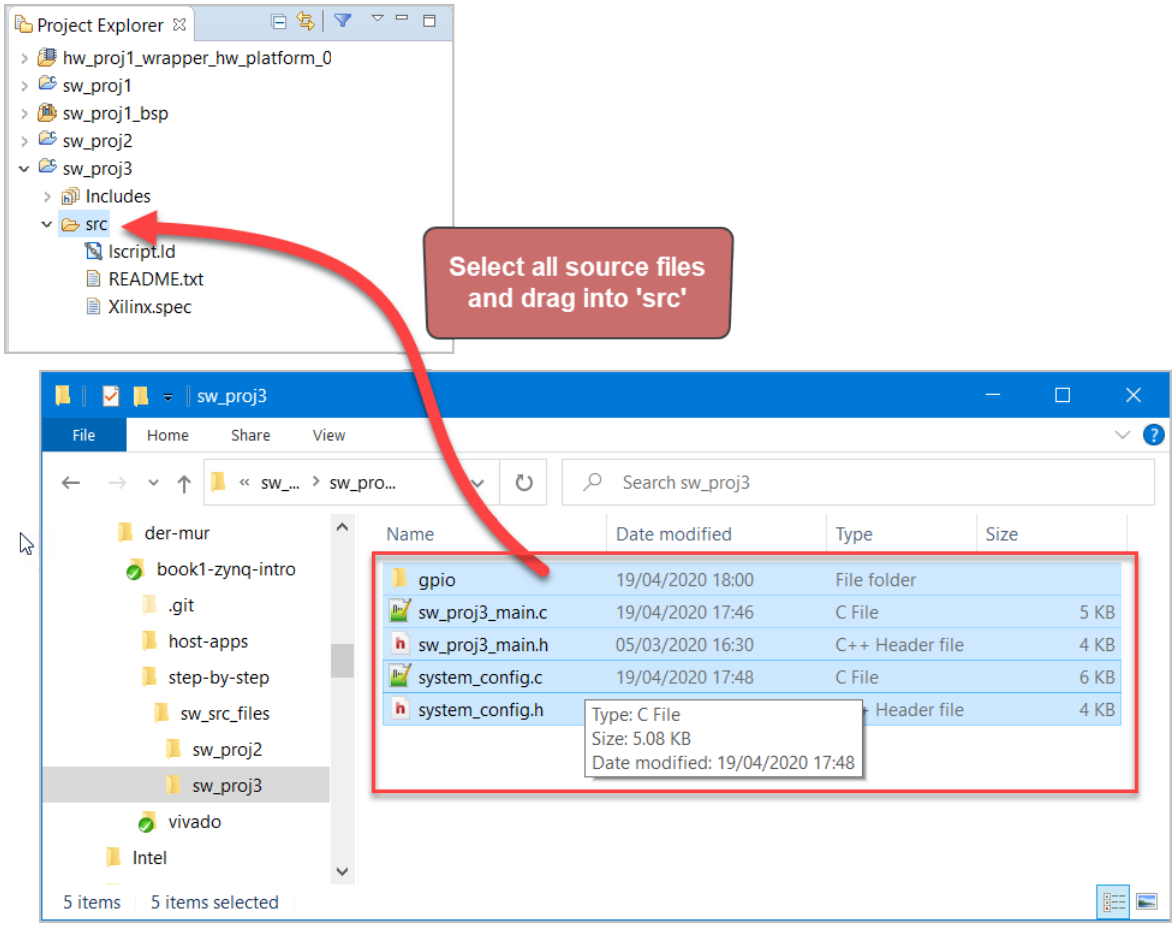


Figure 120. Drag the source files from Windows Explorer into the SDK project

Open the location where the source files for software project 3 are saved on your PC. Drag all the files and directory and from Windows explorer directly into the "**src**" folder in SDK.

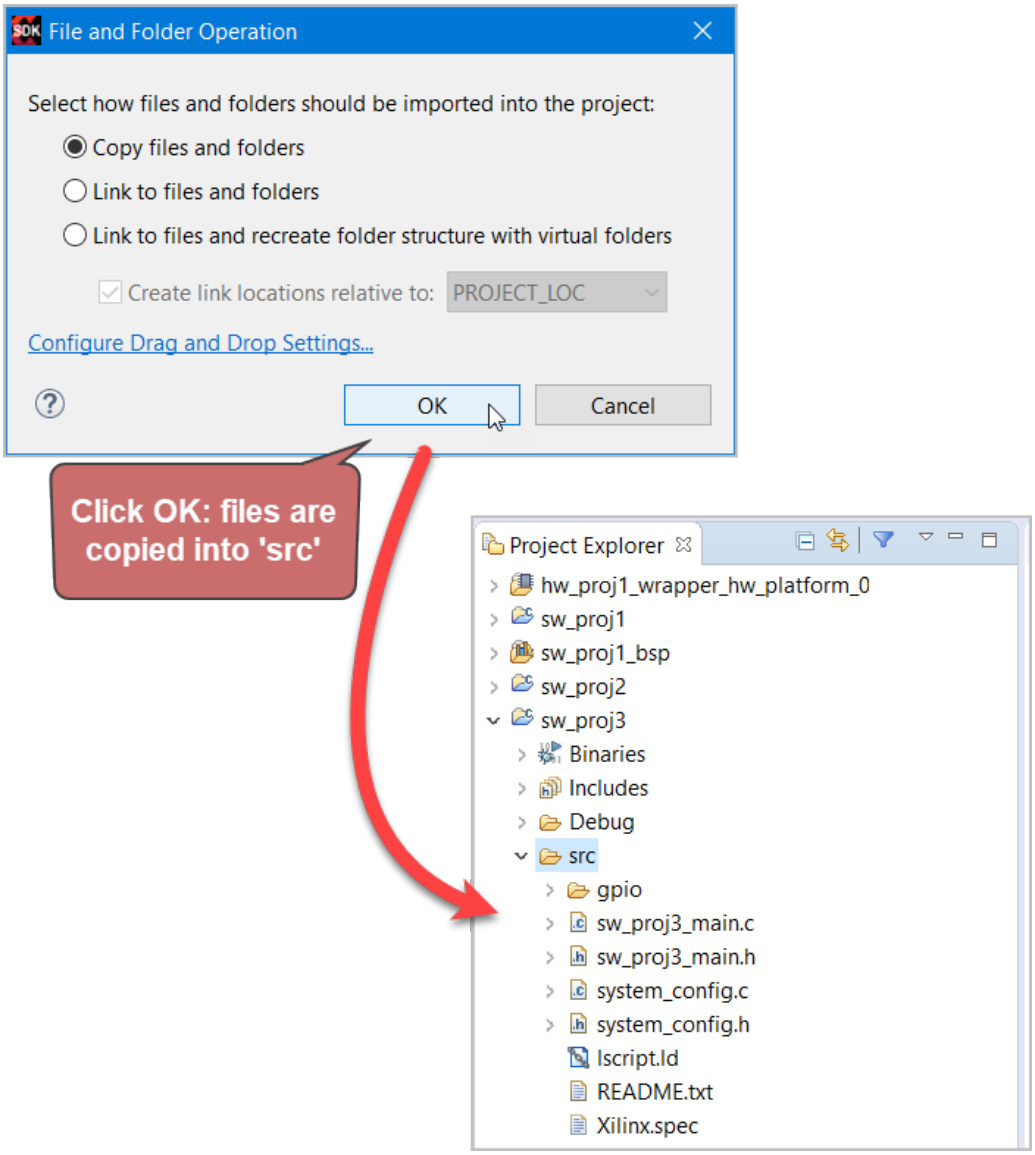


Figure 121. Click OK, and the files will be copied into the project.

In the File Operation dialog box, ensure “Copy files and folders” is checked, and click OK.

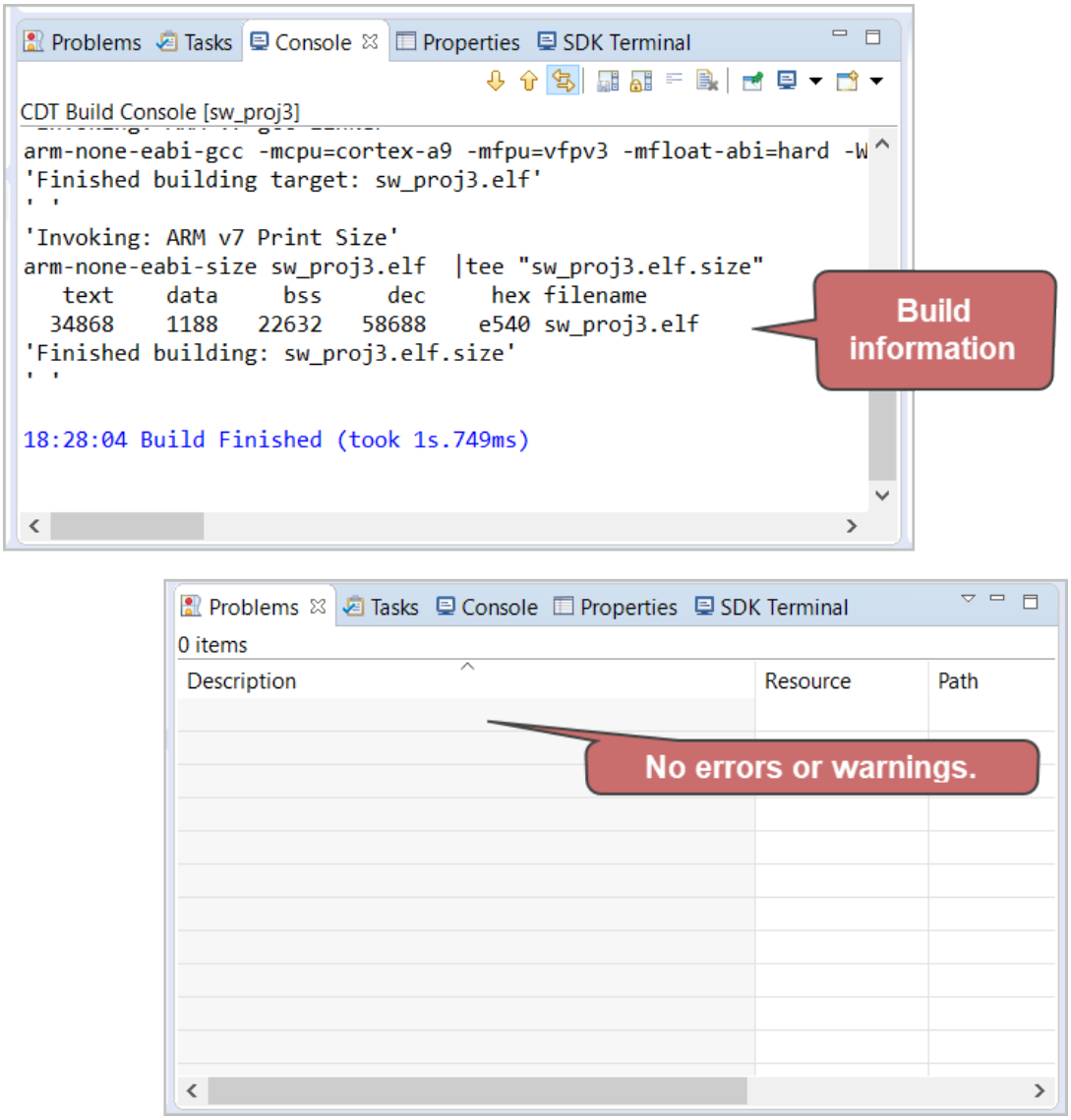


Figure 122. The project should build successfully

If "Build Automatically" is selected, the project should build successfully, with no errors or warnings.

[Exceptions: SDK 2018.3/SDK 2019.1 may indicate a warning as follows, which can be ignored:

#pragma message: For the sleep routines, Global timer is being used]

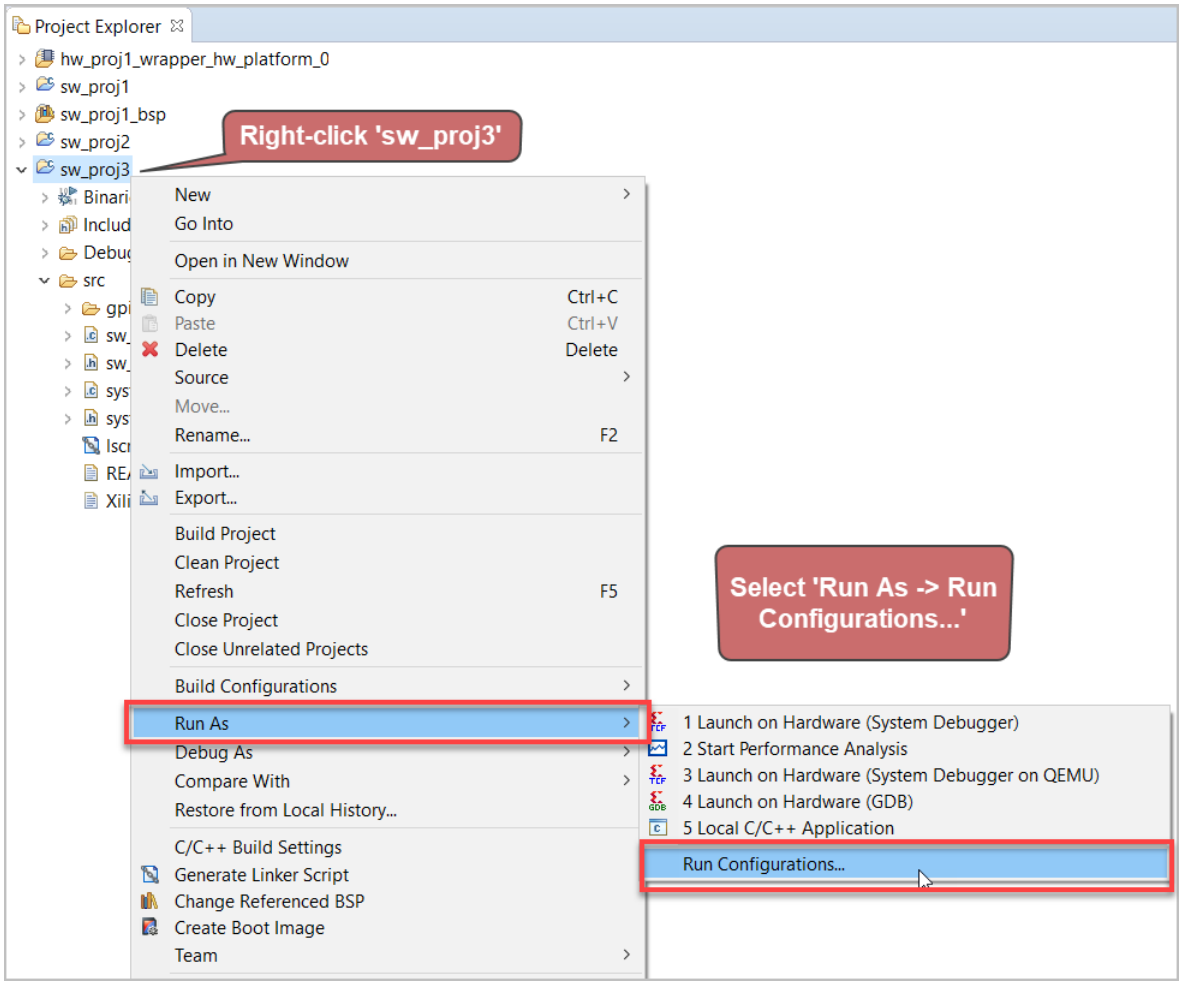


Figure 123. Right-click the software project and create a Run Configuration

To run the program, the first step is to create a Run configuration. Right-click on the project, and select the 'Run Configurations...' option.

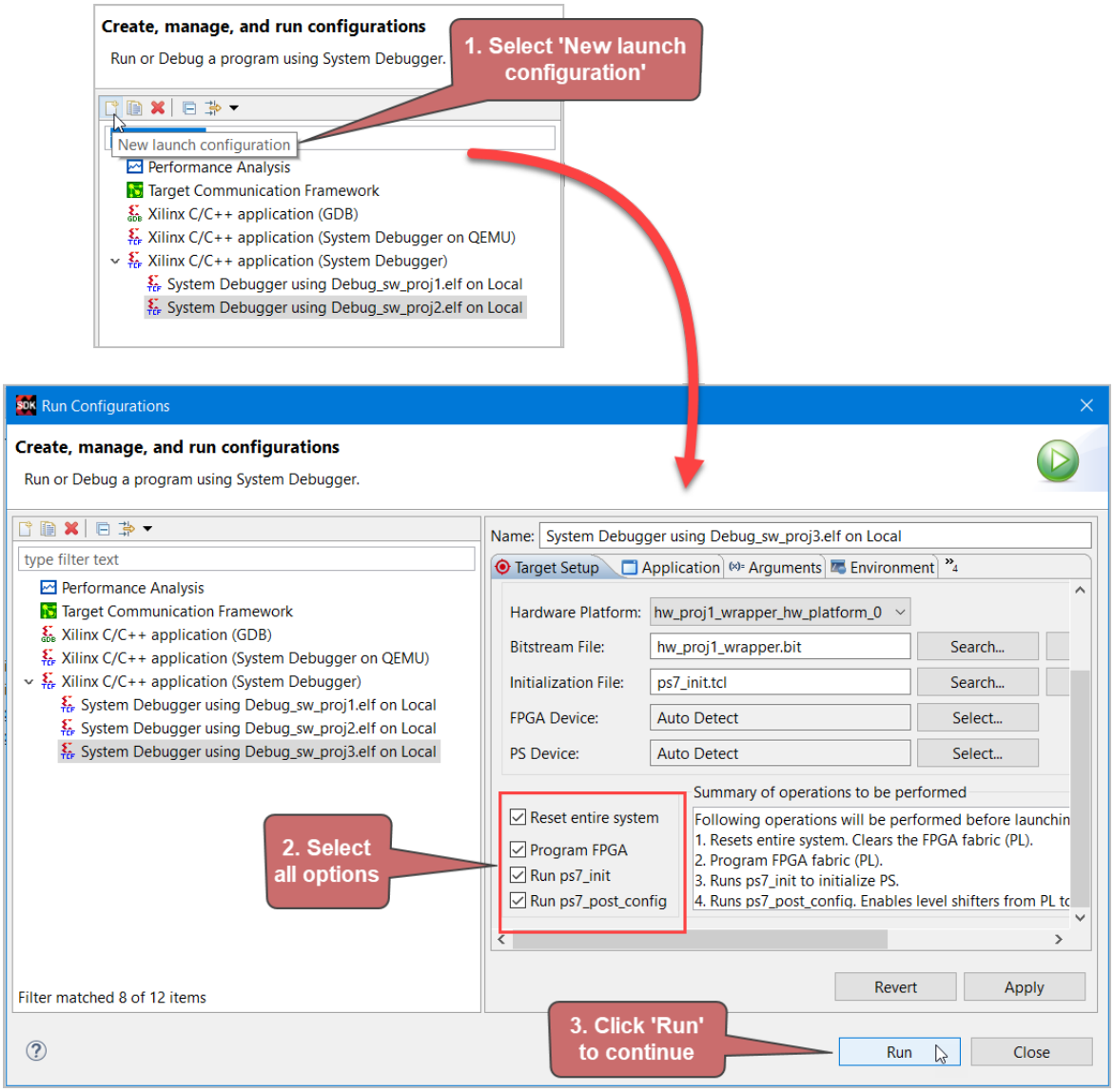


Figure 124. Create a TCF (i.e. System Debugger) option and click Run to execute the program.

1. Ensure the Xilinx C/C++ application (System Debugger) is selected.
2. Click on New Launch Configuration.
3. Select all options:
    a. Reset entire system
    b. Program FPGA
    c. Run ps7_init
    d. Run ps7_post_config

4. Click on 'Run' to continue.

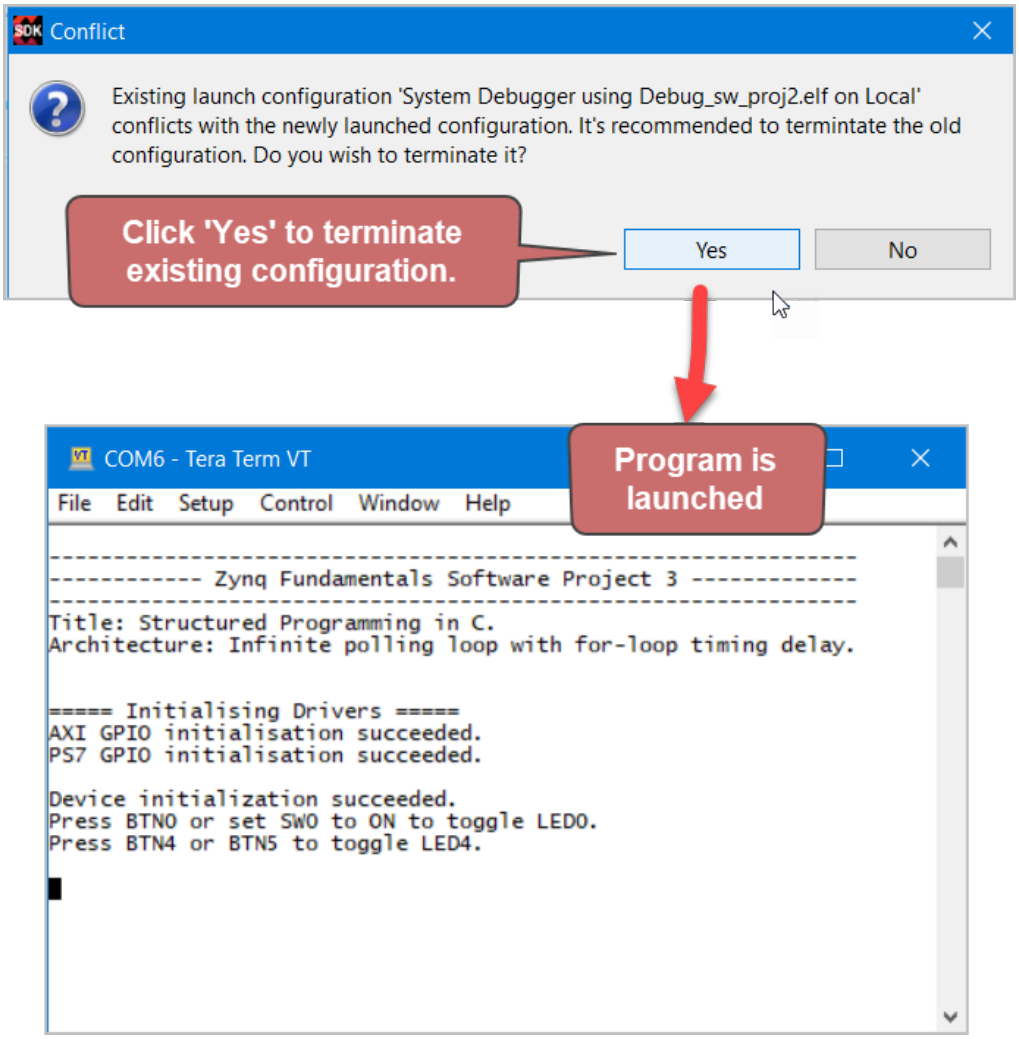


Figure 125. Terminal output for Software Project 3

If prompted, select 'Yes' to terminate any existing configuration. The FPGA will be programmed and the application will be downloaded to the processor. The terminal program should display the results for launching Software Project 3.

3.4 Software Project 4: SCU Timing

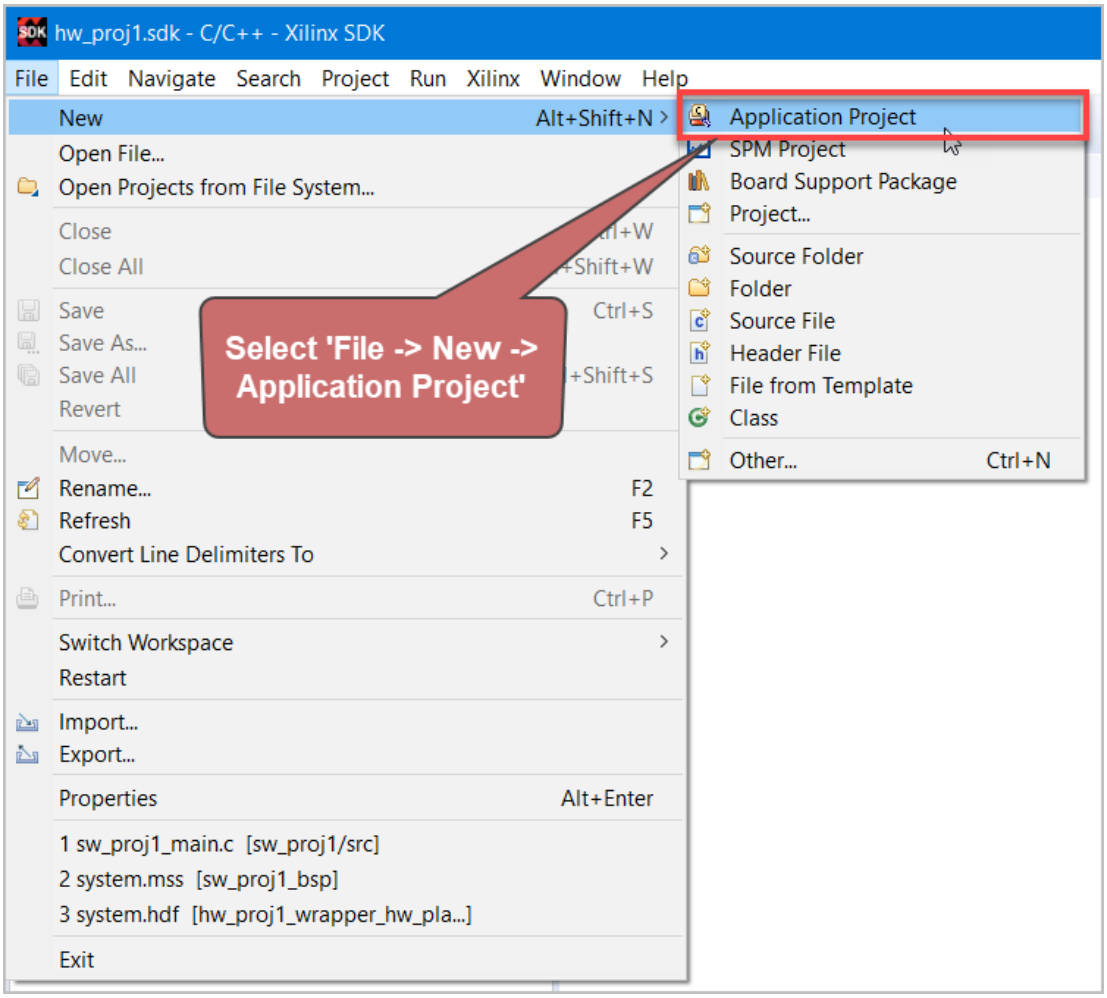


Figure 126. Select File->New-> Application Project

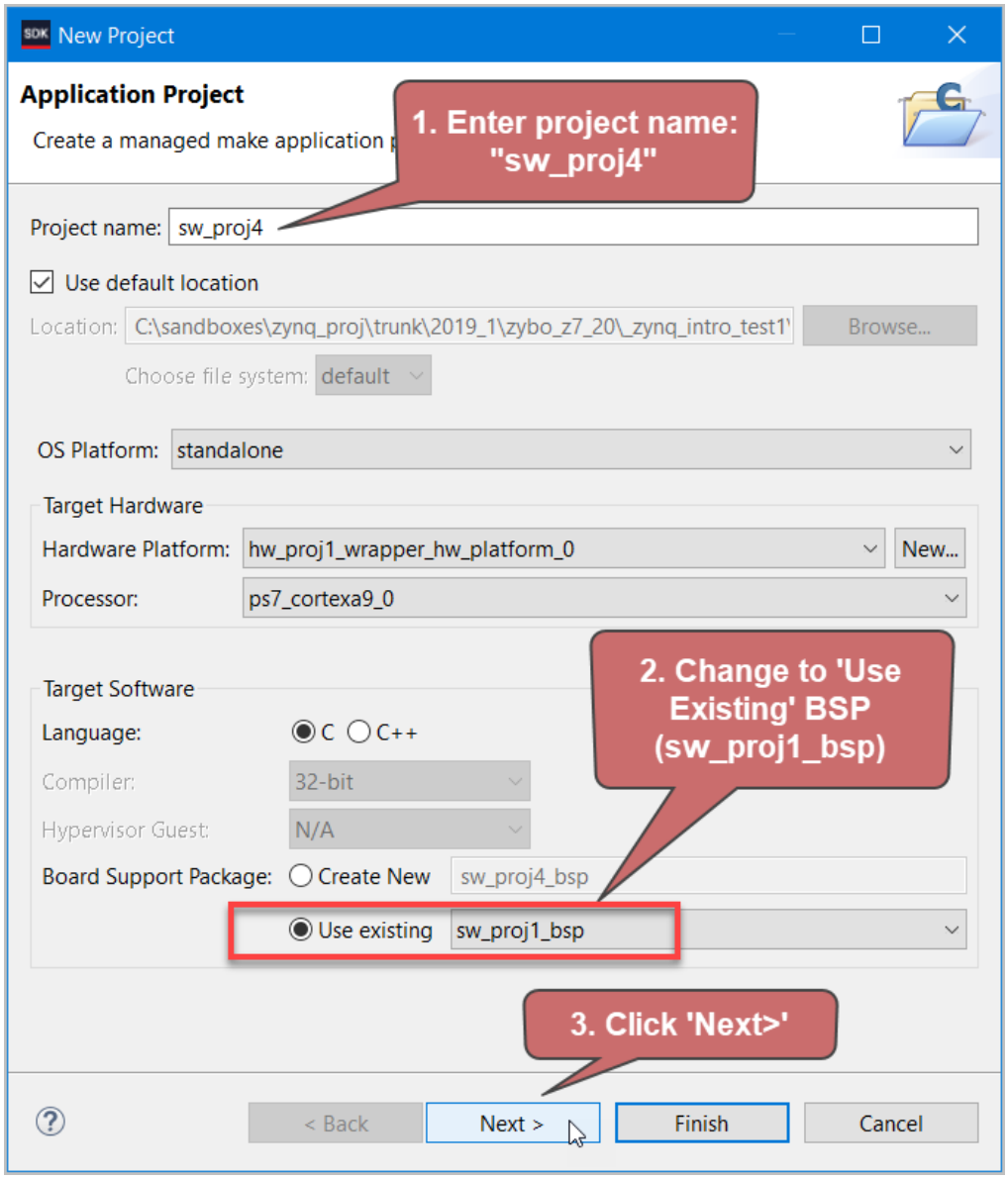


Figure 127. Enter the project details (part 1)

1. Enter the project name: sw_proj4
2. Board Support Package: Use existing
3. Click Next (don't click Finish!)

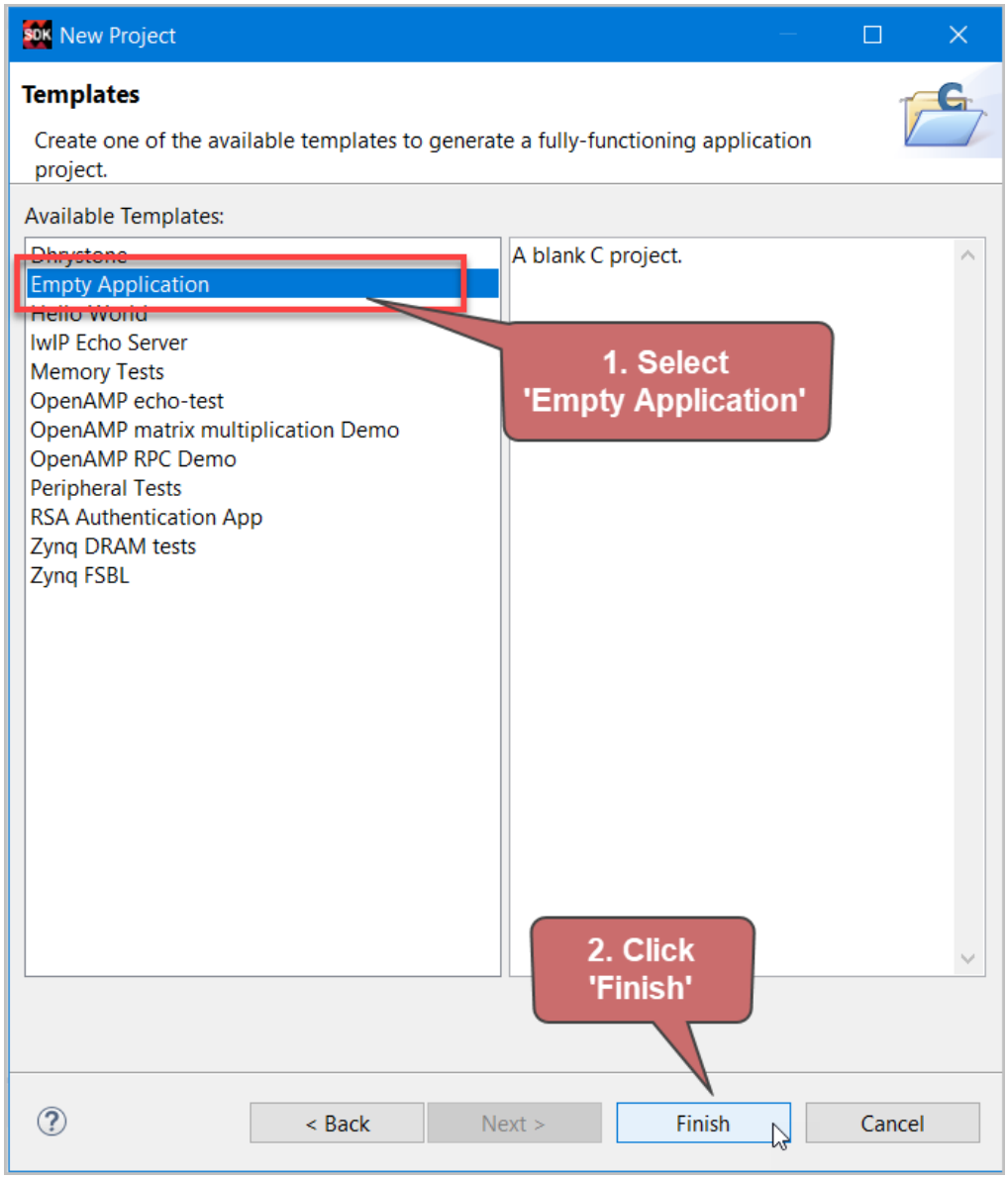


Figure 128. Enter the project details (part 2)

1. Select Empty Application
2. Click Finish

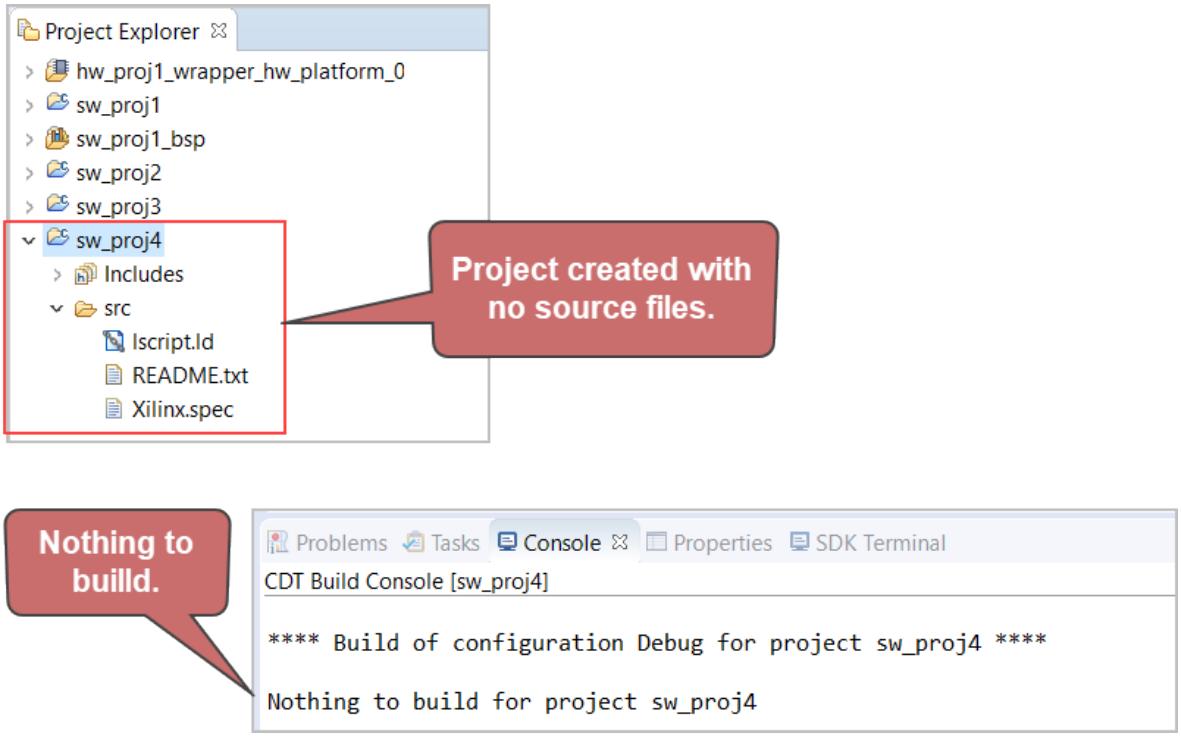


Figure 130. Project created with no source files

A project named “sw_proj4” is created with no project files. The CDT Build Console confirms that there is nothing to build.

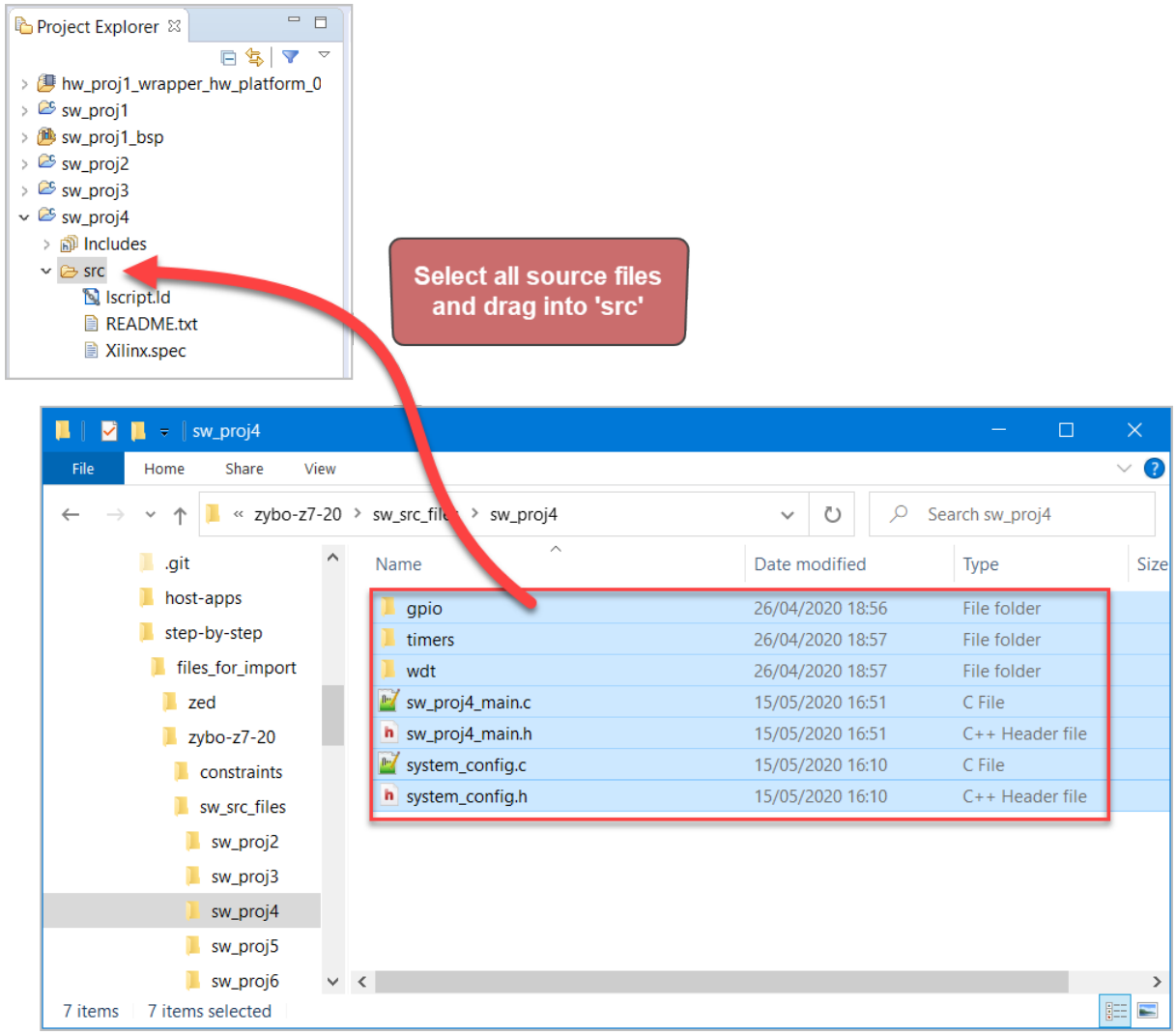


Figure 131. Drag the source files from Windows Explorer into the SDK project

Open the location where the source files for software project 4 are saved on your PC. Drag all the files and directory and from Windows explorer directly into the "**src**" folder in SDK.

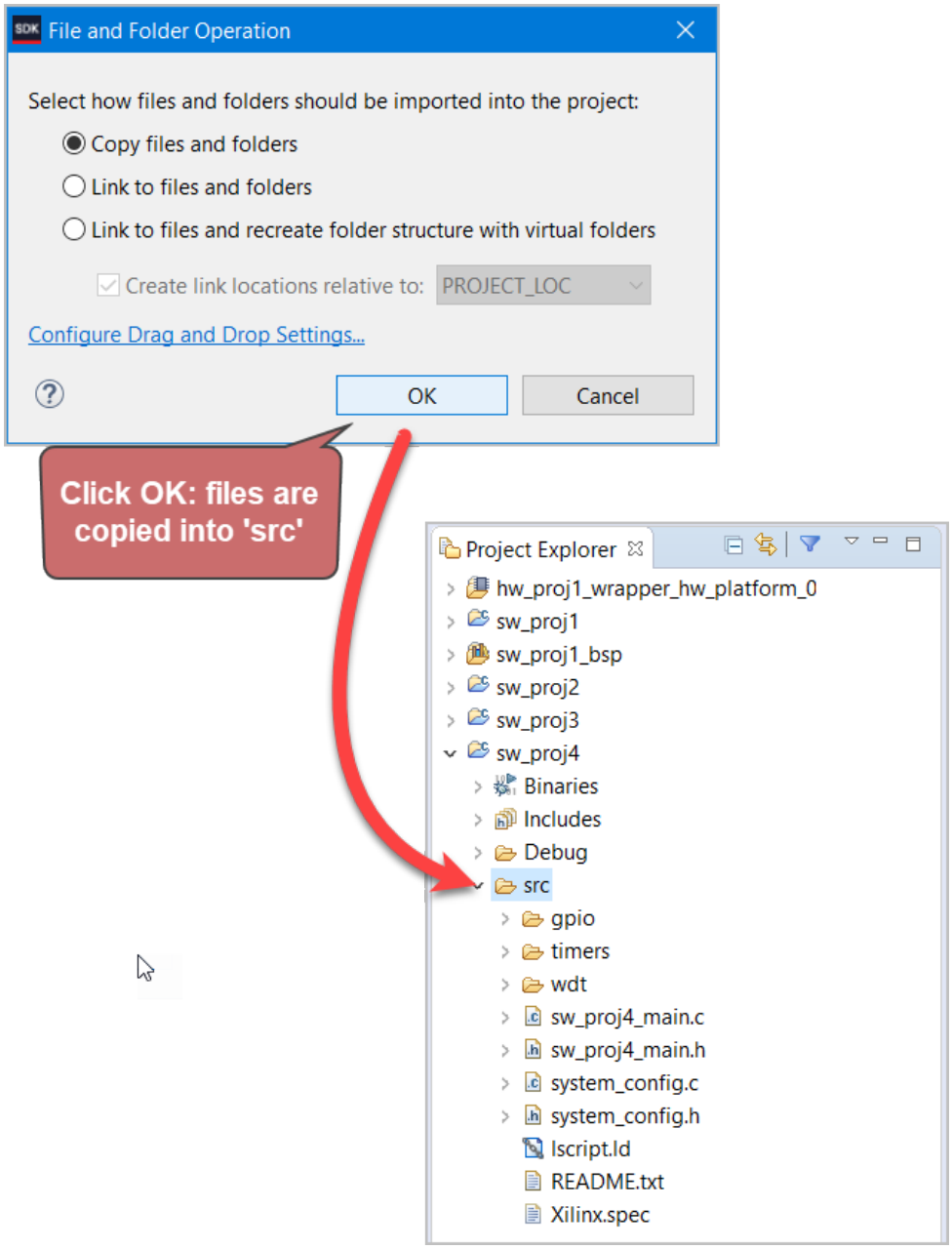


Figure 132. Click OK, and the files will be copied into the project.

In the File Operation dialog box, ensure “Copy files and folders” is checked, and click OK.

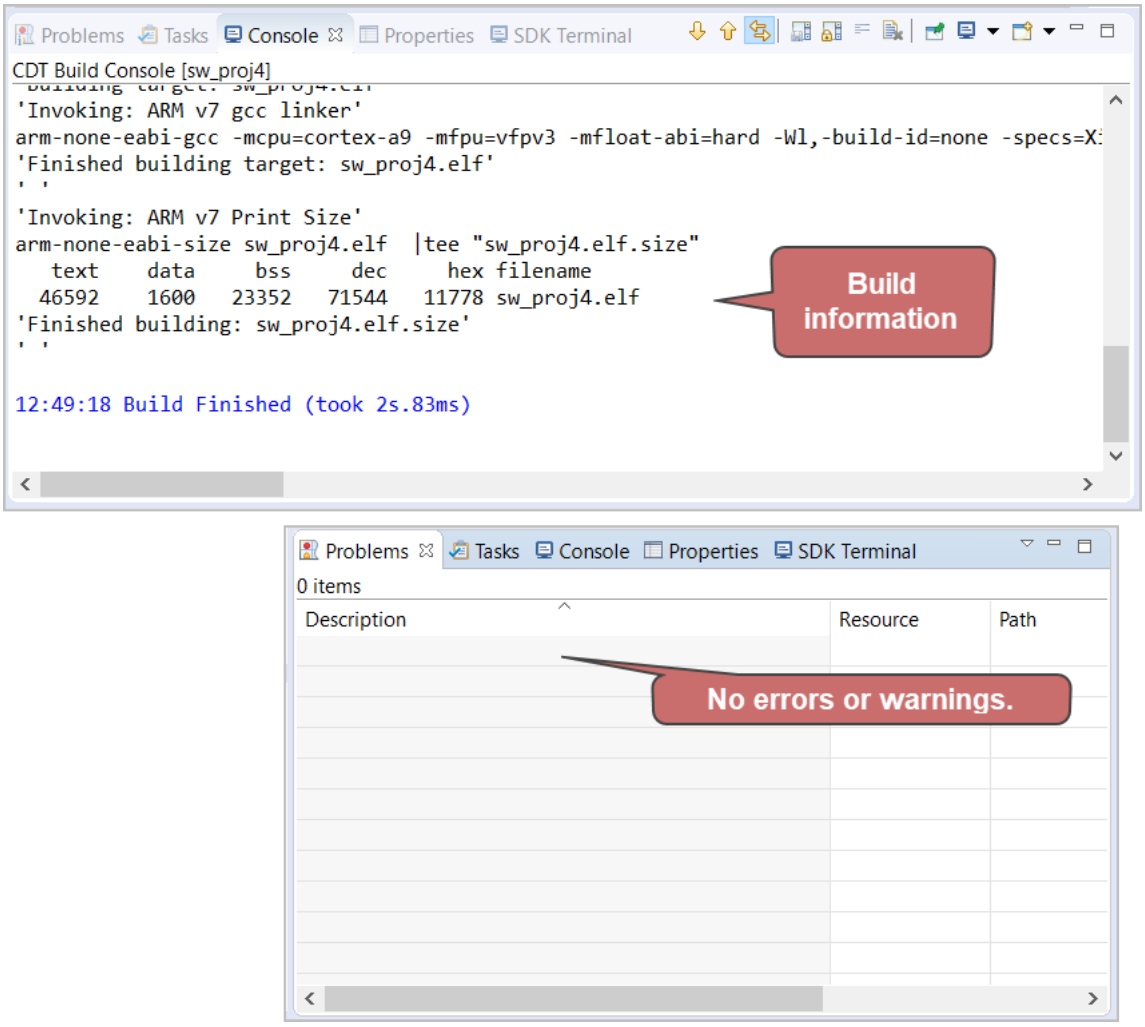


Figure 133. The project should build successfully

If "Build Automatically" is selected, the project should build successfully, with no errors or warnings.

[Exceptions: SDK 2018.3/SDK 2019.1 may indicate a warning as follows, which can be ignored:

#pragma message: For the sleep routines, Global timer is being used

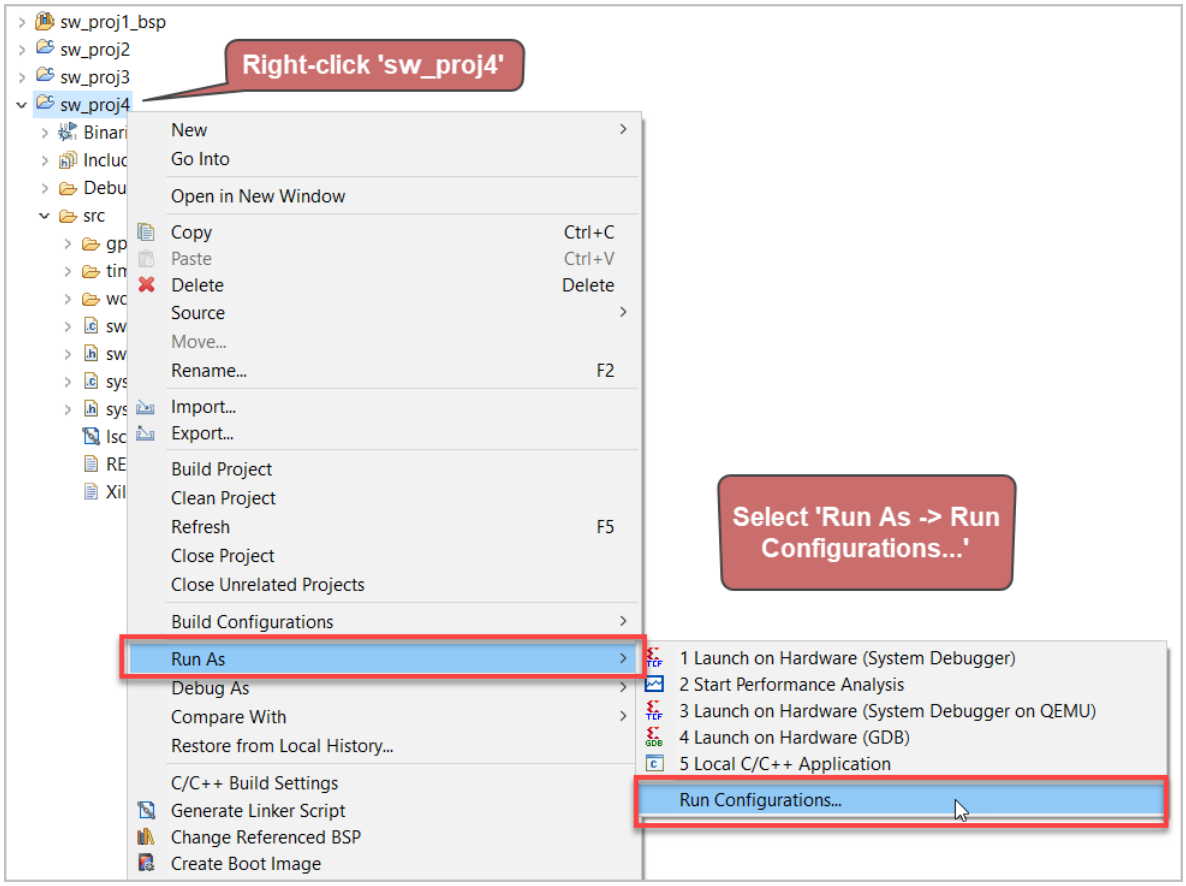


Figure 134. Right-click the software project and create a Run Configuration

To run the program, the first step is to create a Run configuration. Right-click on the project, and select the 'Run Configurations...' option.

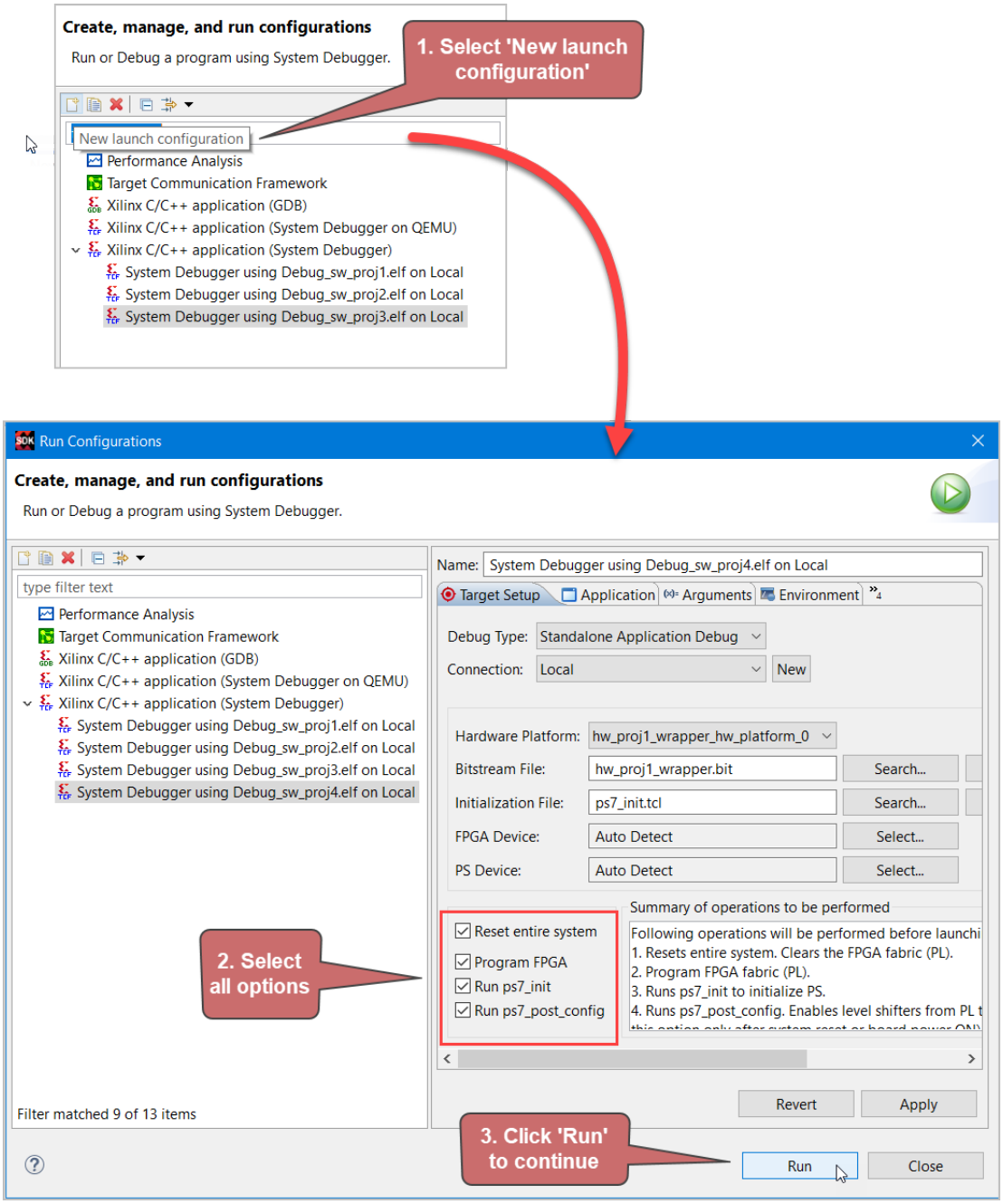


Figure 135. Create a TCF (i.e. System Debugger) option and click Run to execute the program.

1. Ensure the Xilinx C/C++ application (System Debugger) is selected.
2. Click on New Launch Configuration.
3. Select all options:
    a. Reset entire system
    b. Program FPGA
    c. Run ps7_init
    d. Run ps7_post_config

4. Click on 'Run' to continue.

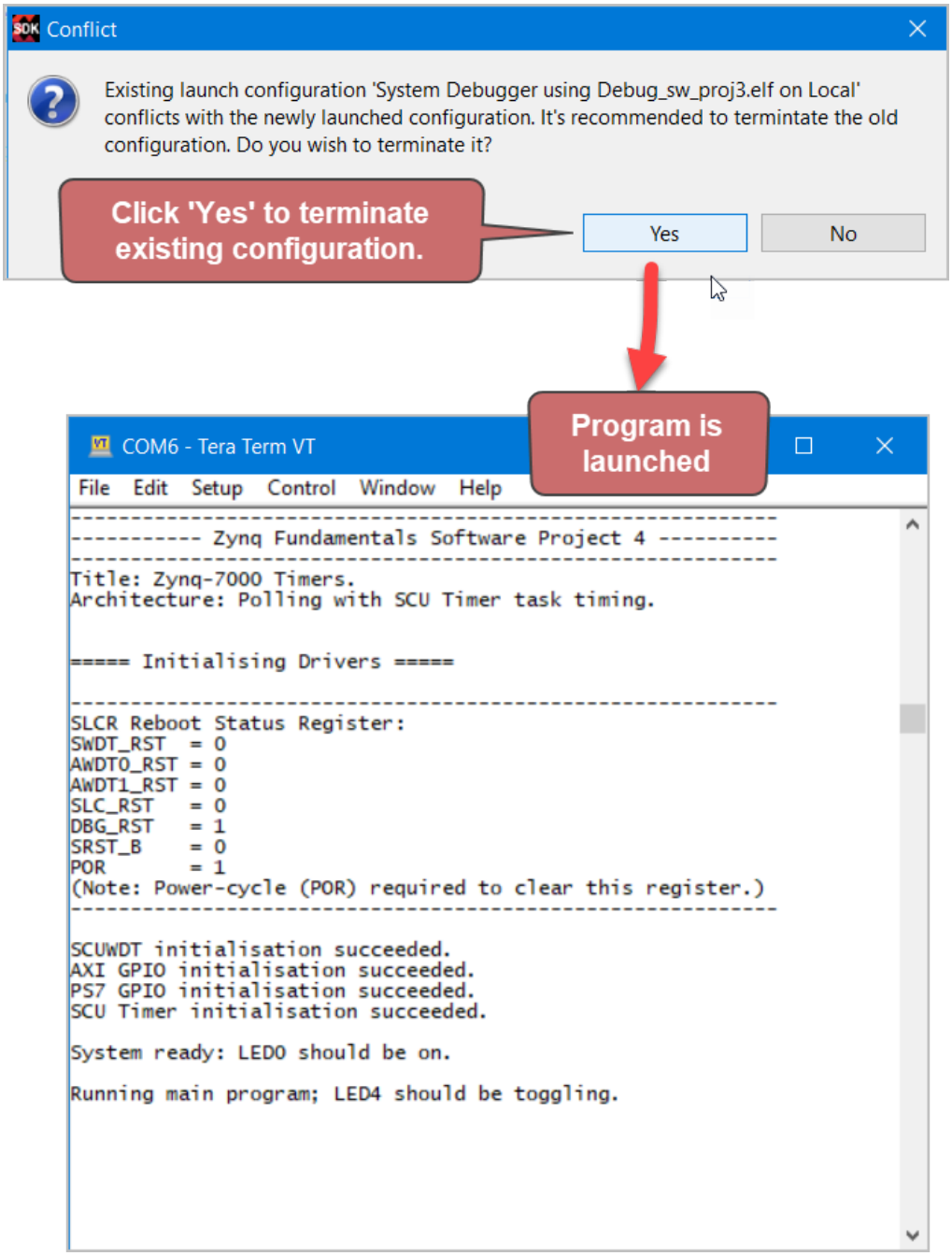


Figure 136. Terminal output for Software Project 4

If prompted, select 'Yes' to terminate any existing configuration. The FPGA will be programmed and the application will be downloaded to the processor. The terminal program should display the results for launching Software Project 4.

3.5 Software Project 5: Zynq Interrupts

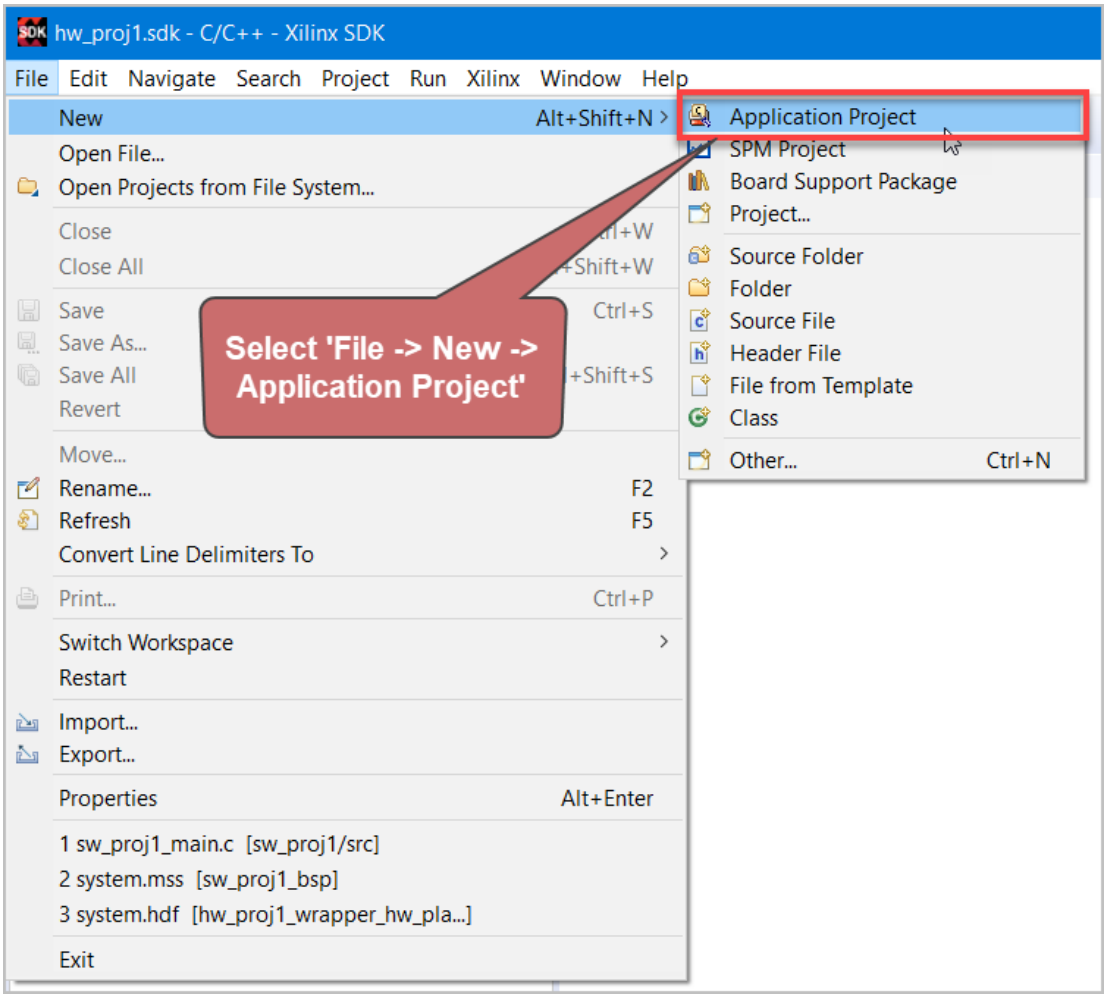


Figure 137. Select File->New-> Application Project

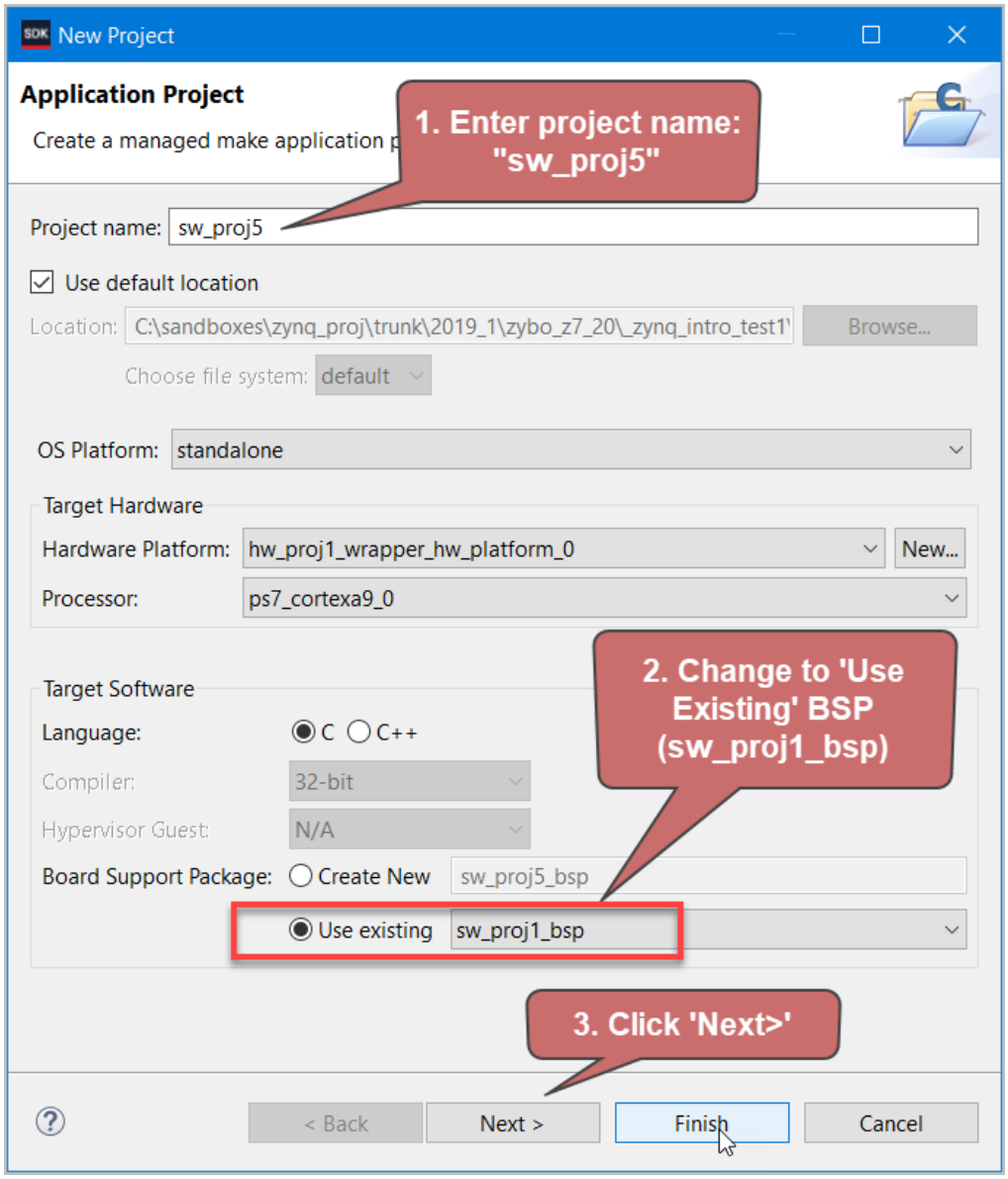


Figure 138. Enter the project details (part 1)

1. Enter the project name: sw_proj5
2. Board Support Package: Use existing
3. Click Next (don't click Finish!)

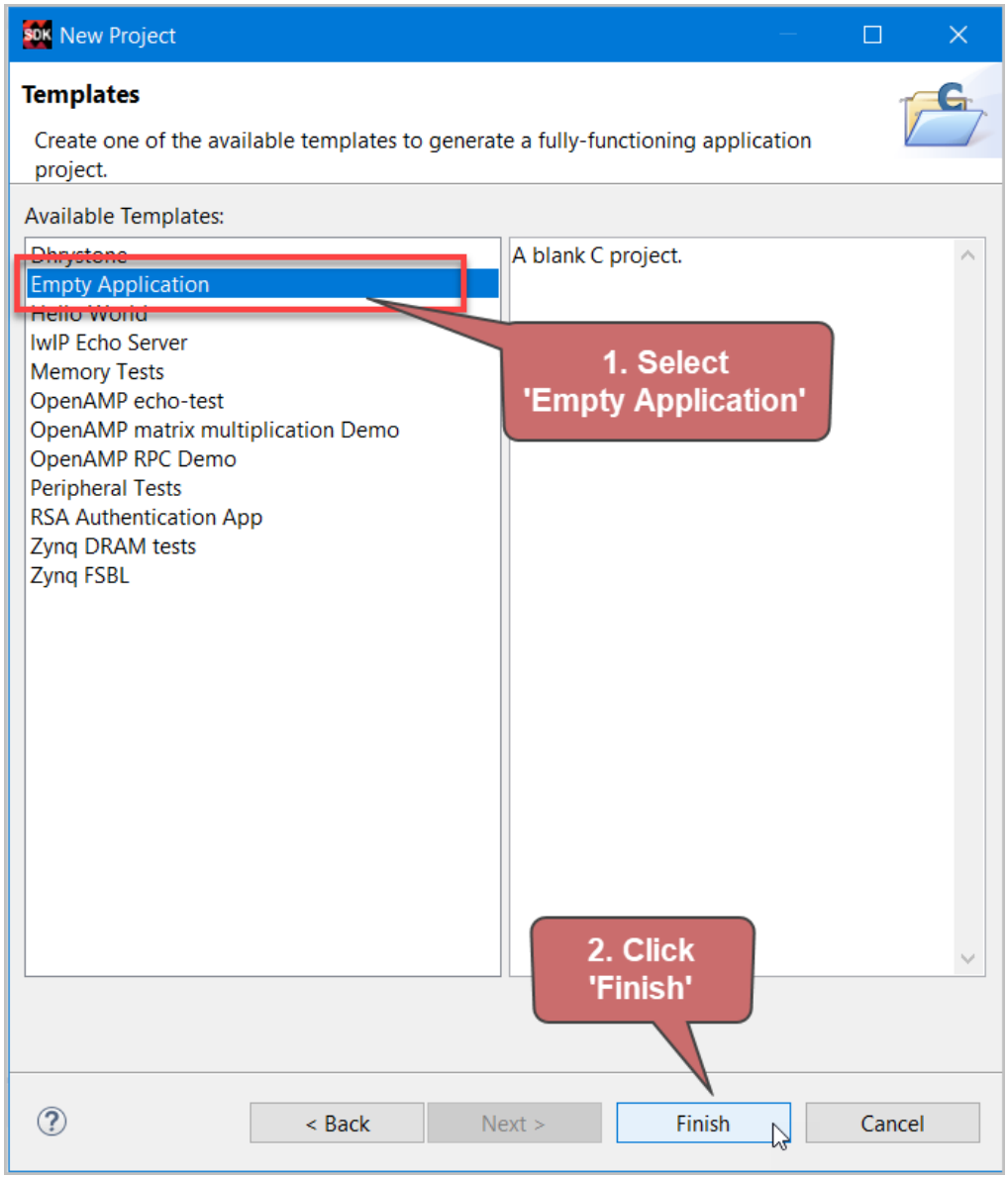


Figure 139. Enter the project details (part 2)

1. Select Empty Application
2. Click Finish

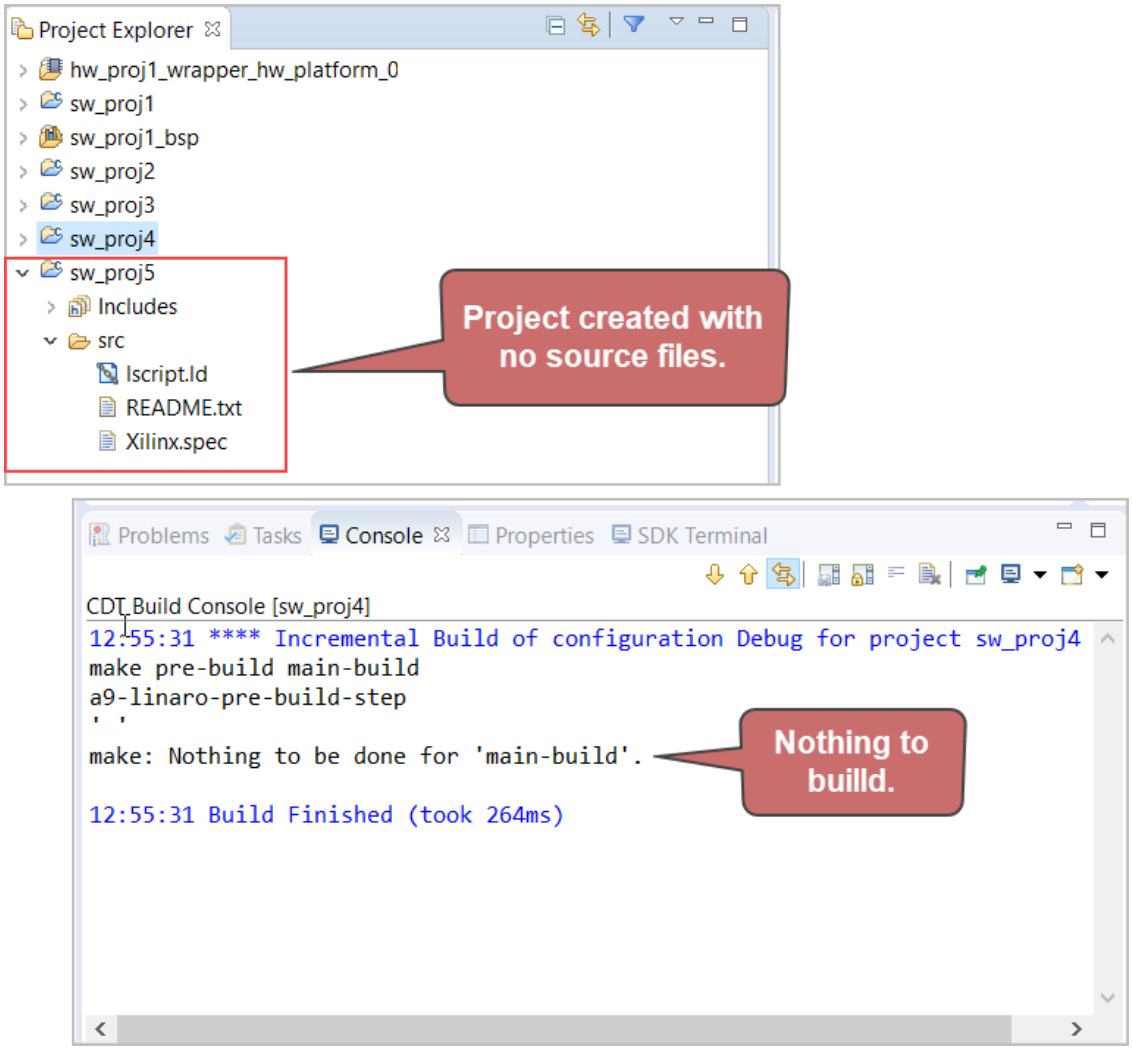


Figure 141. Project created with no source files

A project named "sw_proj5" is created with no project files. The CDT Build Console confirms that there is nothing to build.

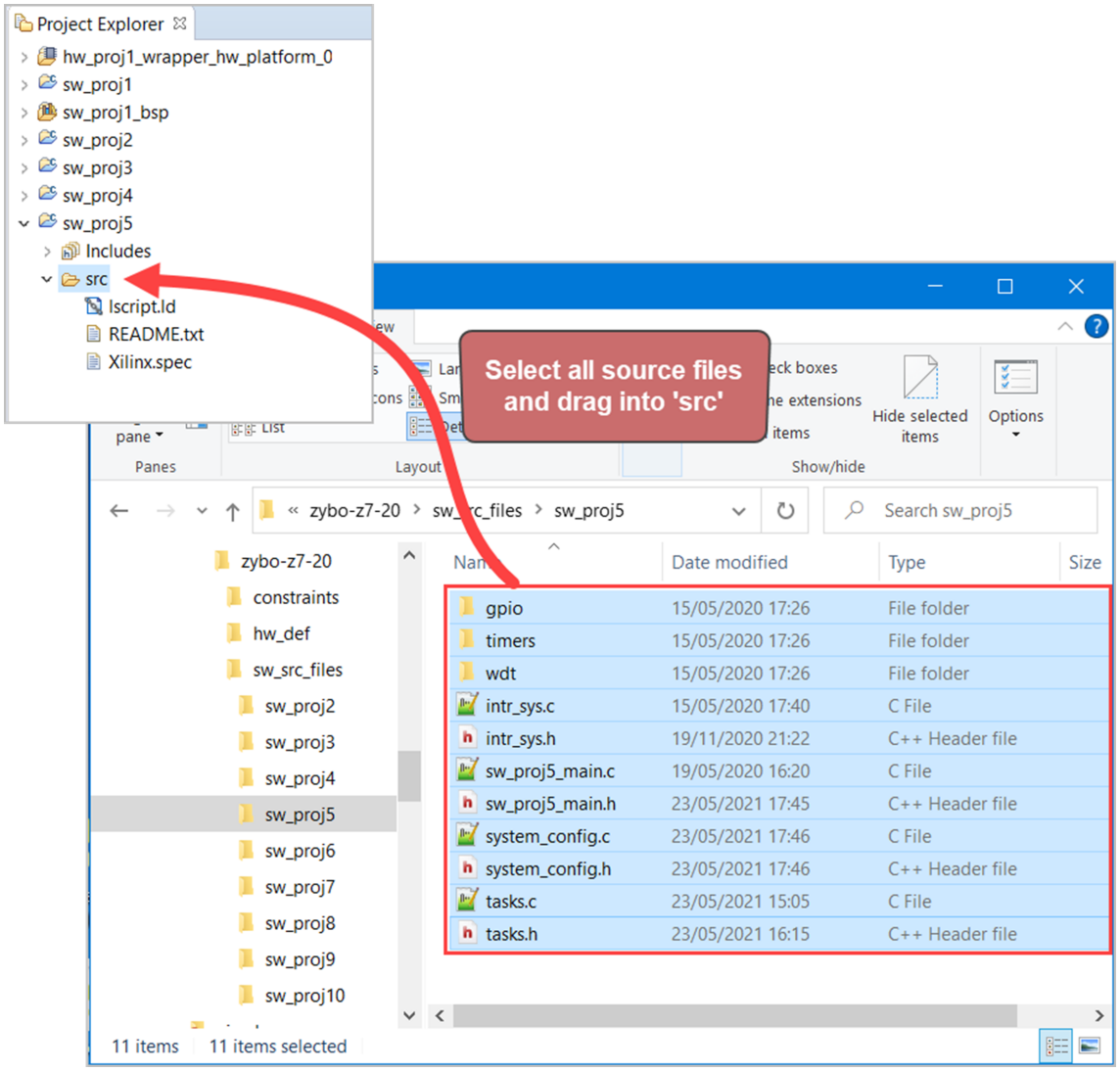


Figure 142. Drag the source files from Windows Explorer into the SDK project

Open the location where the source files for software project 5 are saved on your PC. Drag all the files and directory and from Windows explorer directly into the "**src**" folder in SDK.

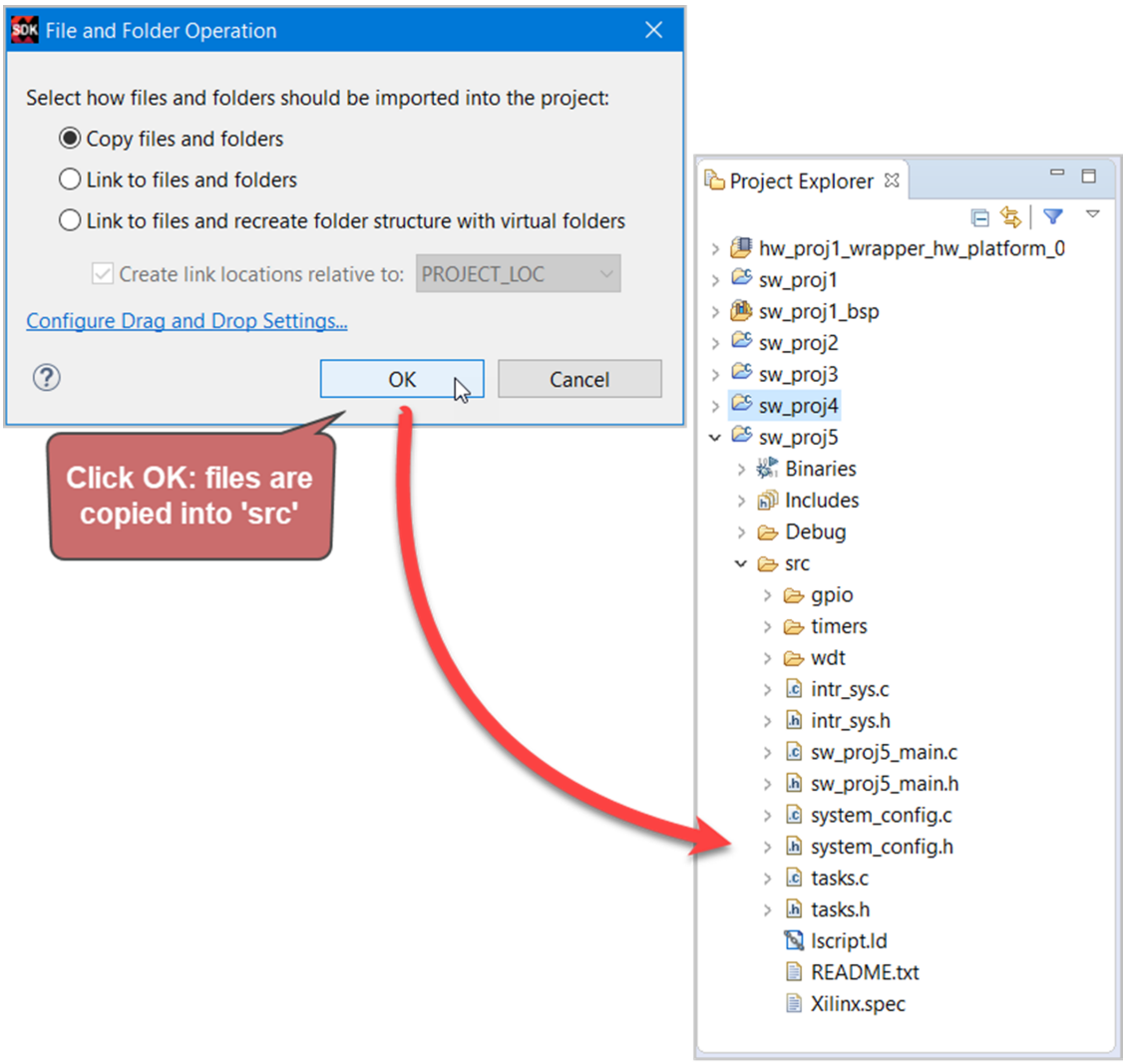


Figure 143. Click OK, and the files will be copied into the project.

In the File Operation dialog box, ensure “Copy files and folders” is checked, and click OK.

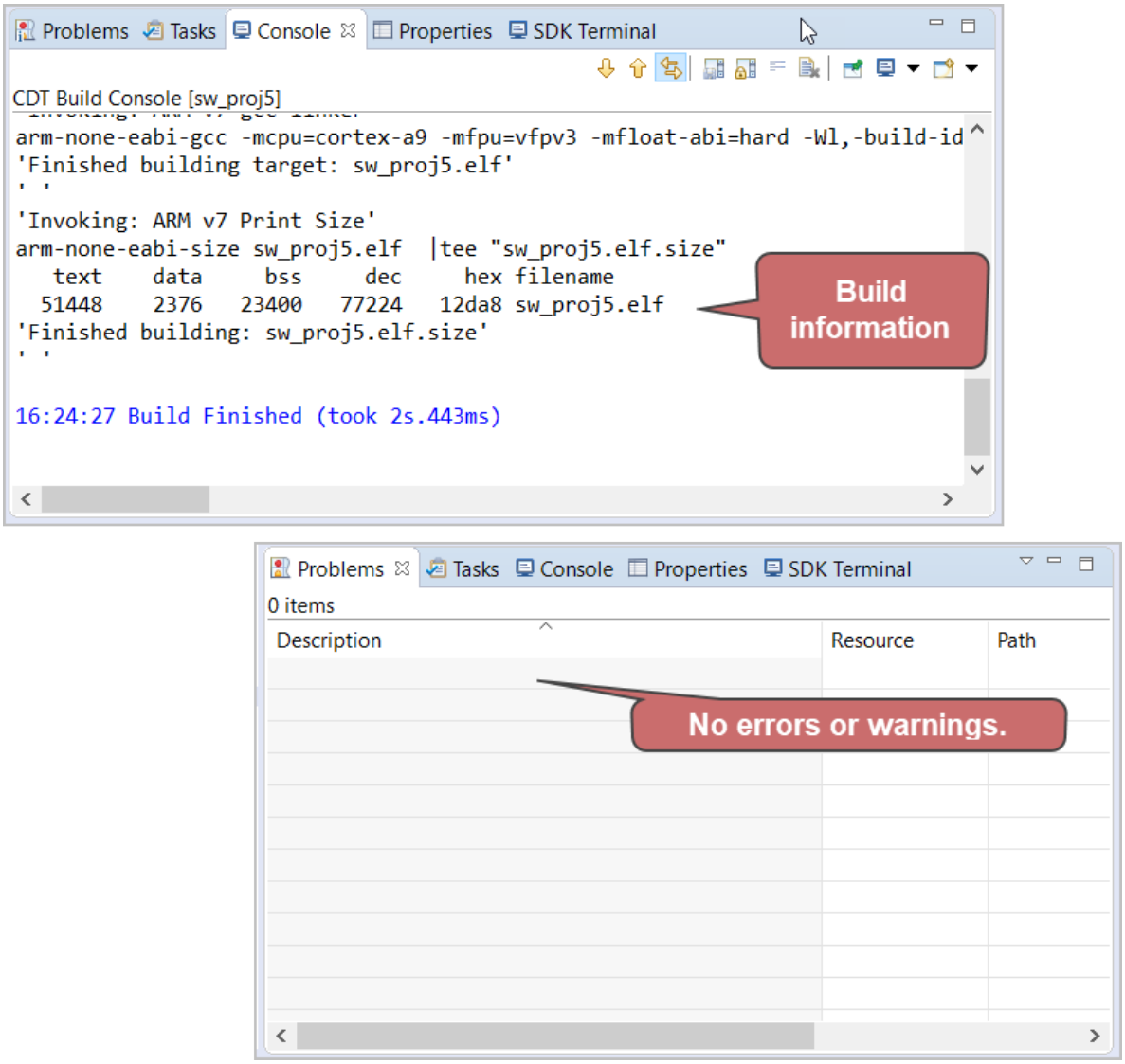


Figure 144. The project should build successfully

If "Build Automatically" is selected, the project should build successfully, with no errors or warnings.

[Exceptions: SDK 2018.3/SDK 2019.1 may indicate a warning as follows, which can be ignored:

#pragma message: For the sleep routines, Global timer is being used]

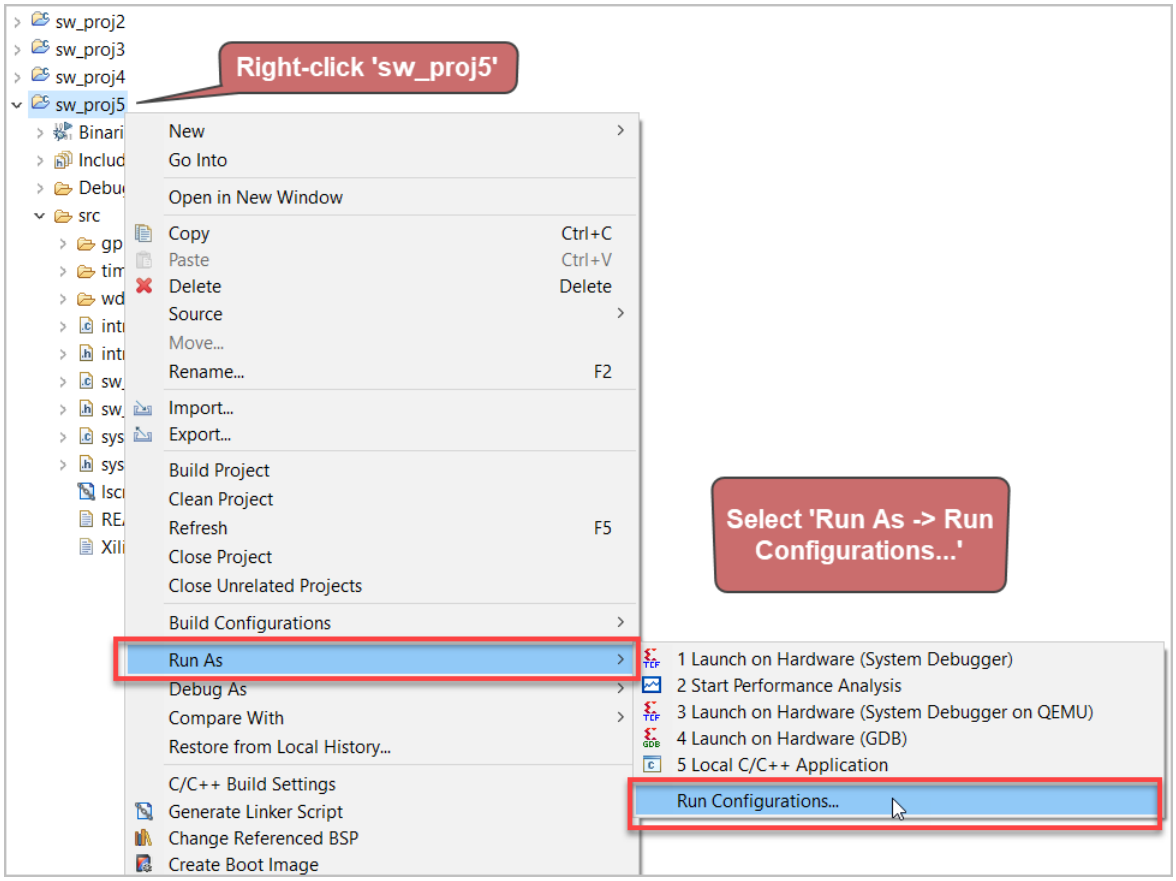


Figure 145. Right-click the software project and create a Run Configuration

To run the program, the first step is to create a Run configuration. Right-click on the project, and select the 'Run Configurations...' option.

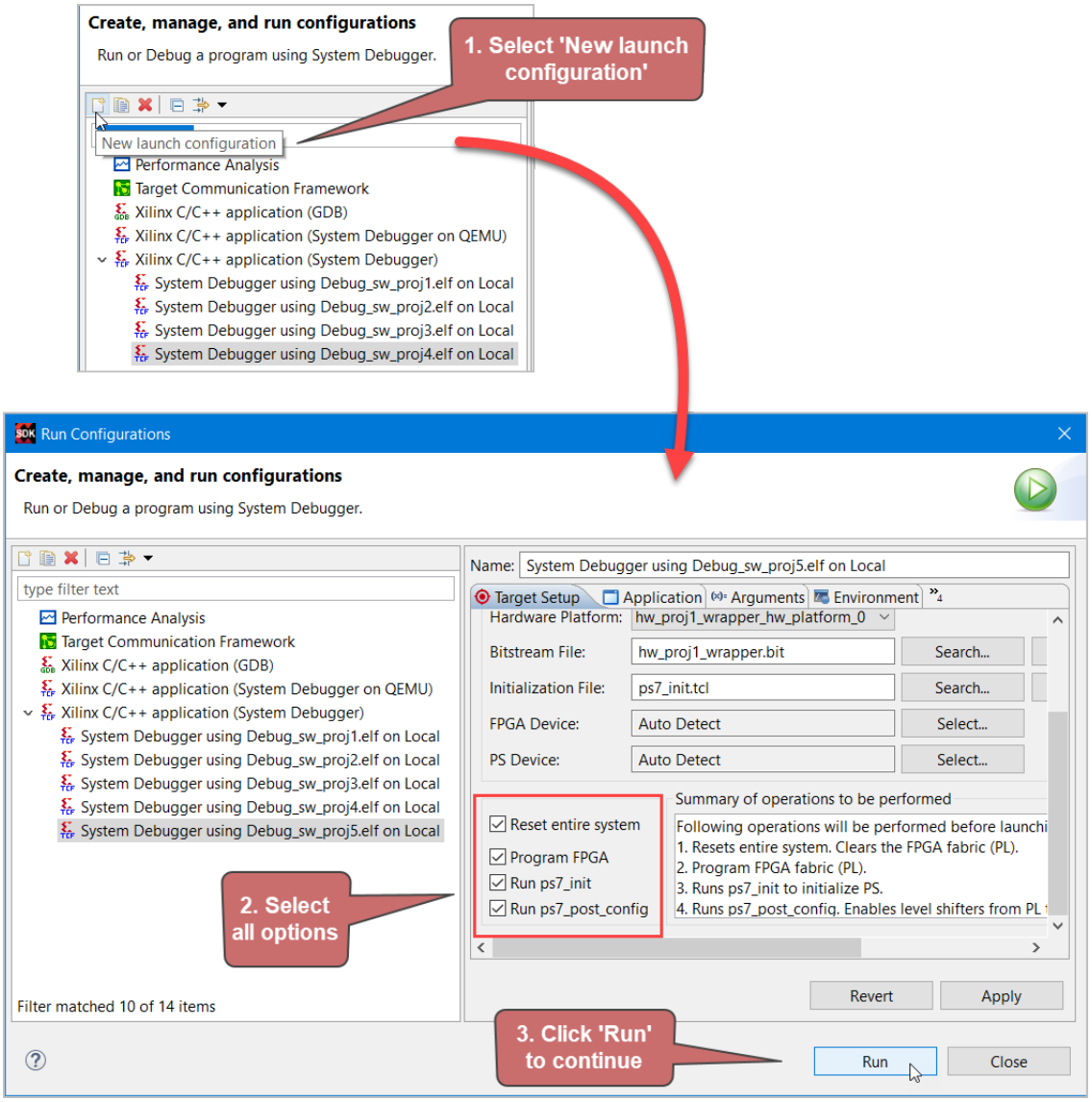


Figure 146. Create a TCF (i.e. System Debugger) option and click Run to execute the program.

Figure 146. Create a TCF (i.e. System Debugger) option and click Run to execute the program.

1. Ensure the Xilinx C/C++ application (System Debugger) is selected.
2. Click on New Launch Configuration.
3. Select all options:
    a. Reset entire system
    b. Program FPGA
    c. Run ps7_init
    d. Run ps7_post_config

4. Click on 'Run' to continue.

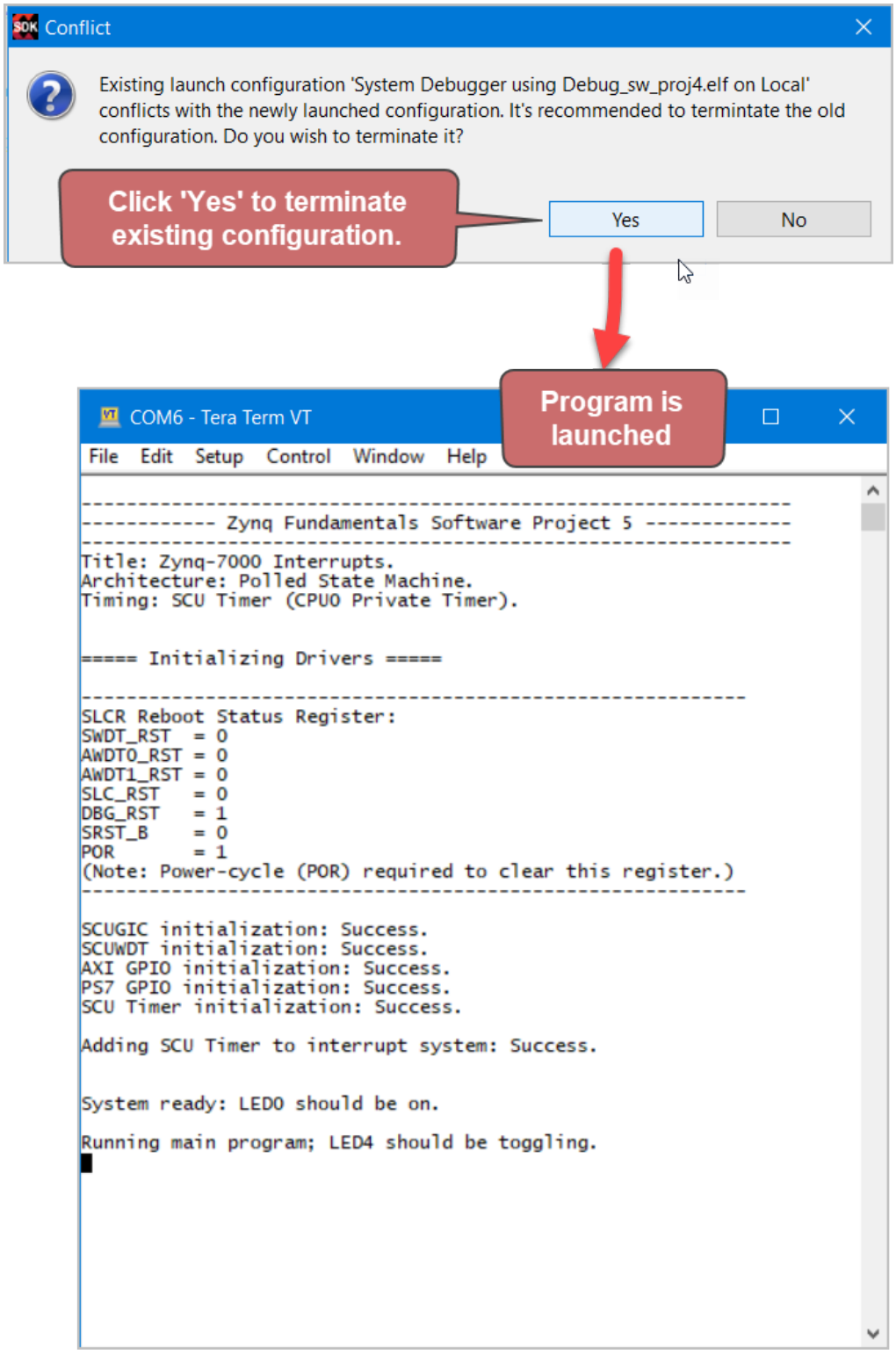


Figure 147. Terminal output for Software Project 5

If prompted, select 'Yes' to terminate any existing configuration. The FPGA will be programmed and the application will be downloaded to the processor. The terminal program should display the results for launching Software Project 5.

3.6 Software Project 6: TTC Timing

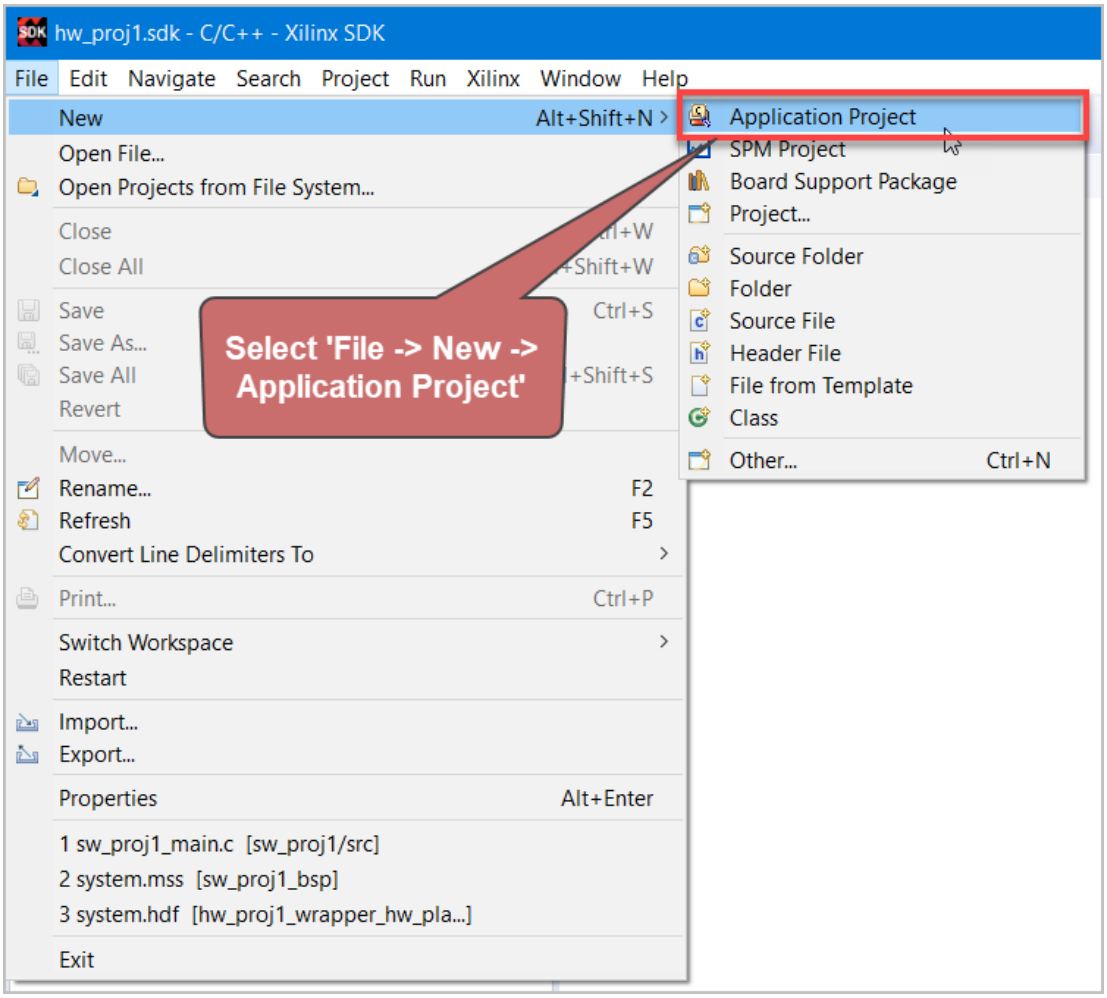


Figure 148. Select File->New-> Application Project

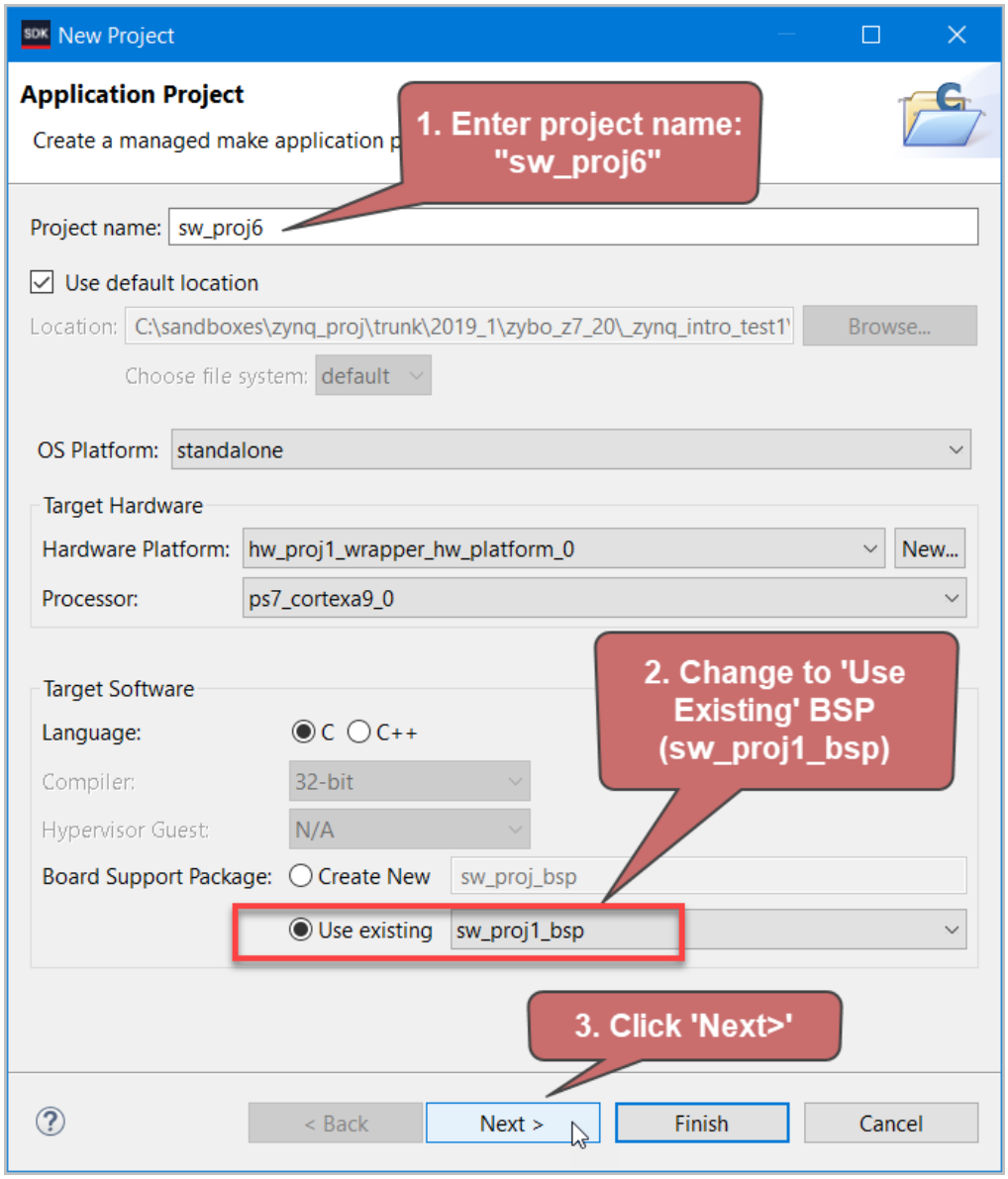


Figure 149. Enter the project details (part 1)

1. Enter the project name: sw_proj6
2. Board Support Package: Use existing
3. Click Next (don't click Finish!)

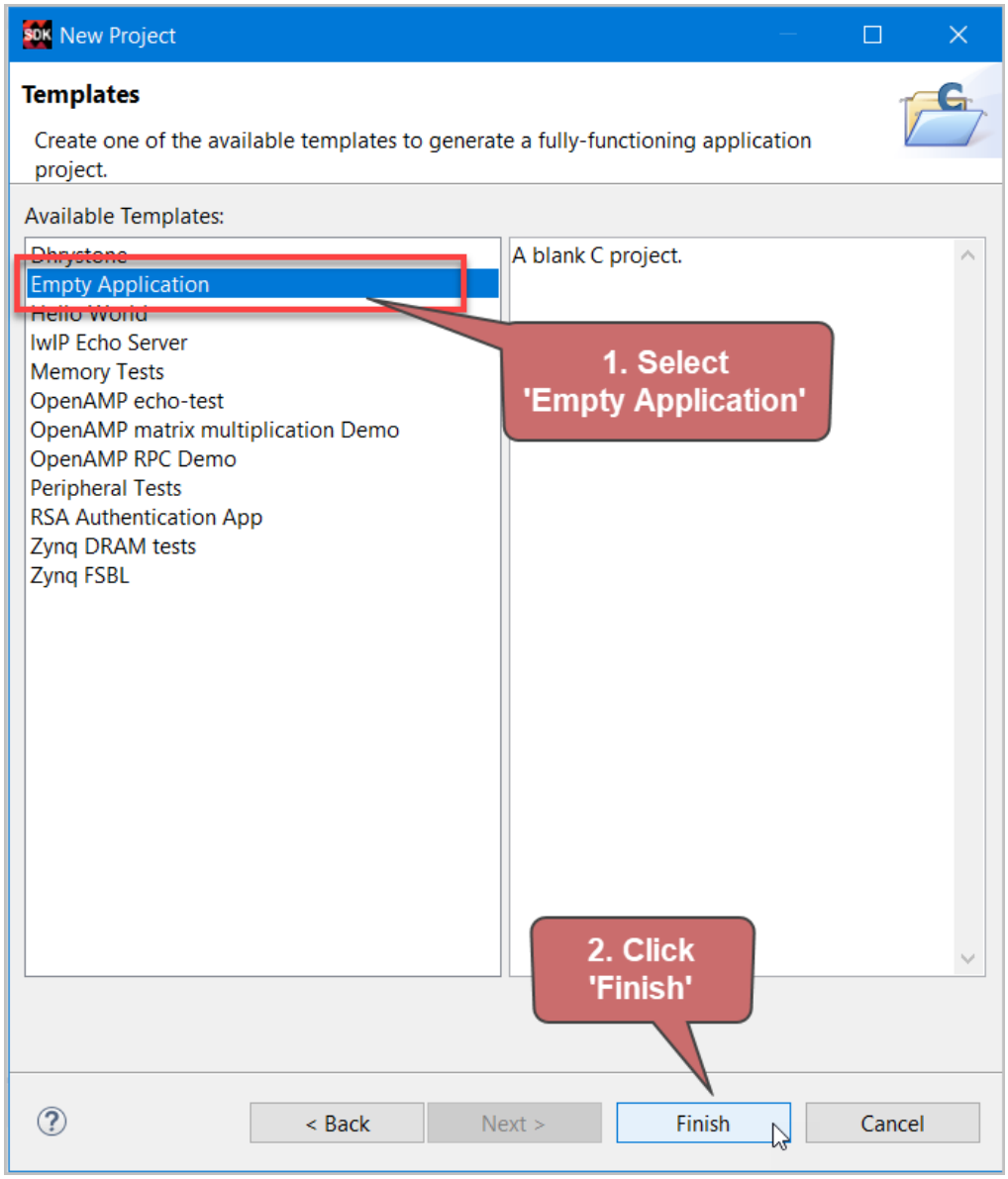


Figure 150. Enter the project details (part 2)

1. Select Empty Application
2. Click Finish

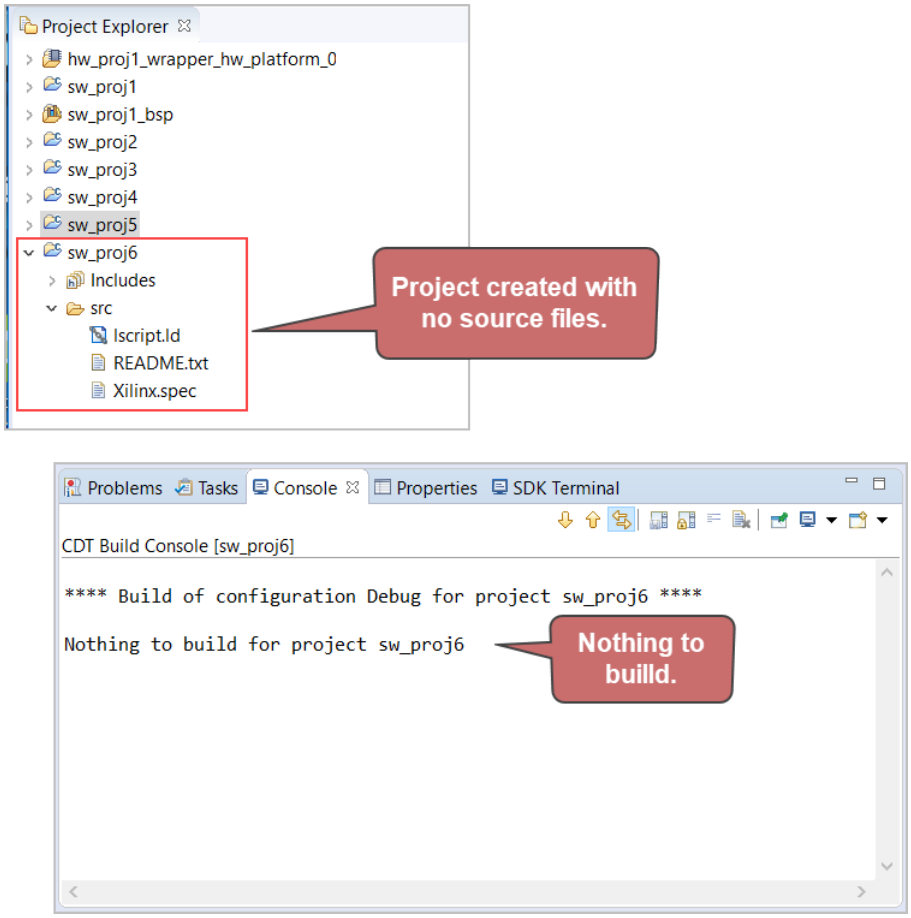


Figure 152. Project created with no source files

A project named “sw_proj6” is created with no project files. The CDT Build Console confirms that there is nothing to build.

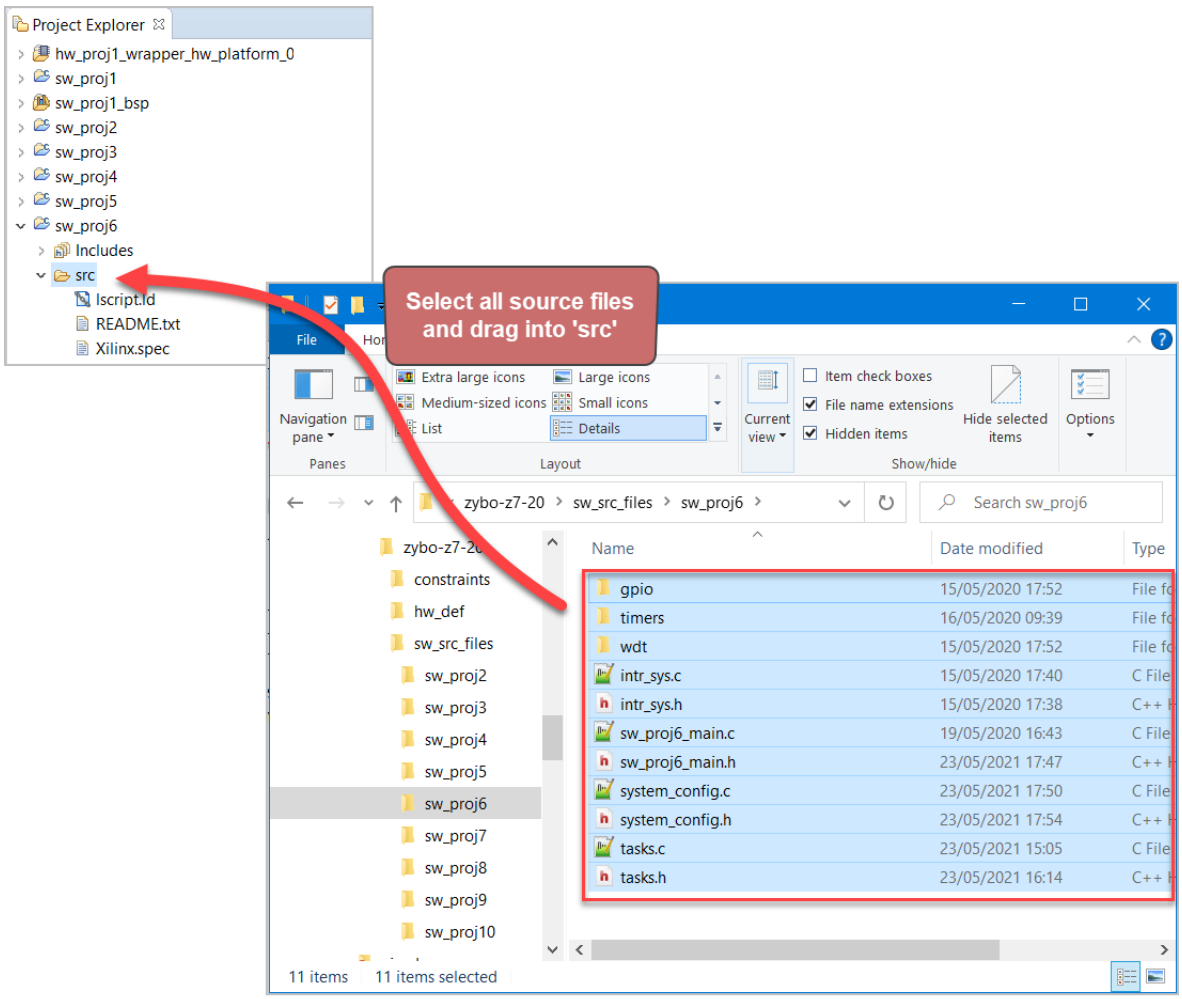


Figure 153. Drag the source files from Windows Explorer into the SDK project

Open the location where the source files for software project 6 are saved on your PC. Drag all the files and directory and from Windows explorer directly into the "**src**" folder in SDK.

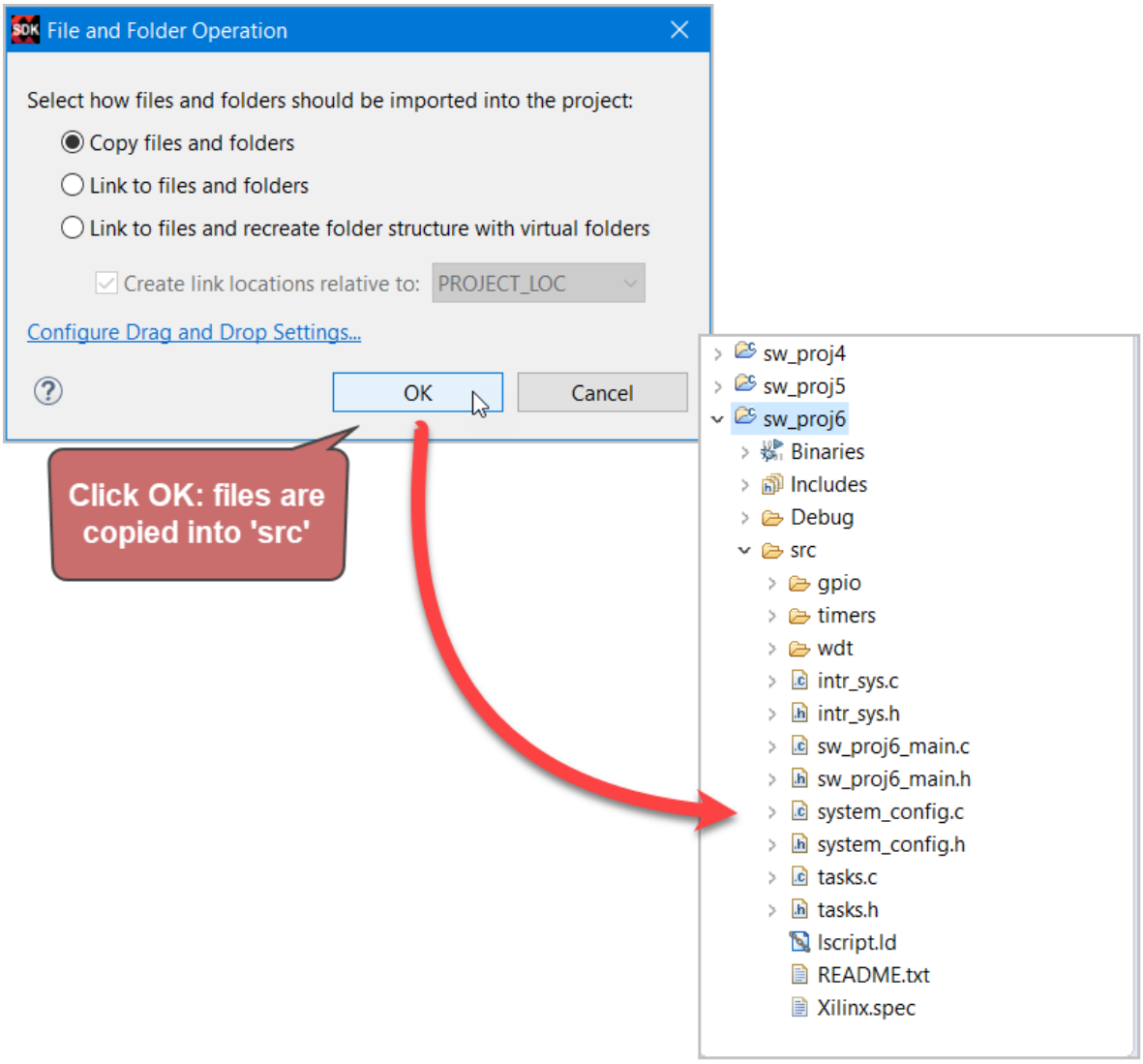


Figure 154. Click OK, and the files will be copied into the project.

In the File Operation dialog box, ensure “Copy files and folders” is checked, and click OK.

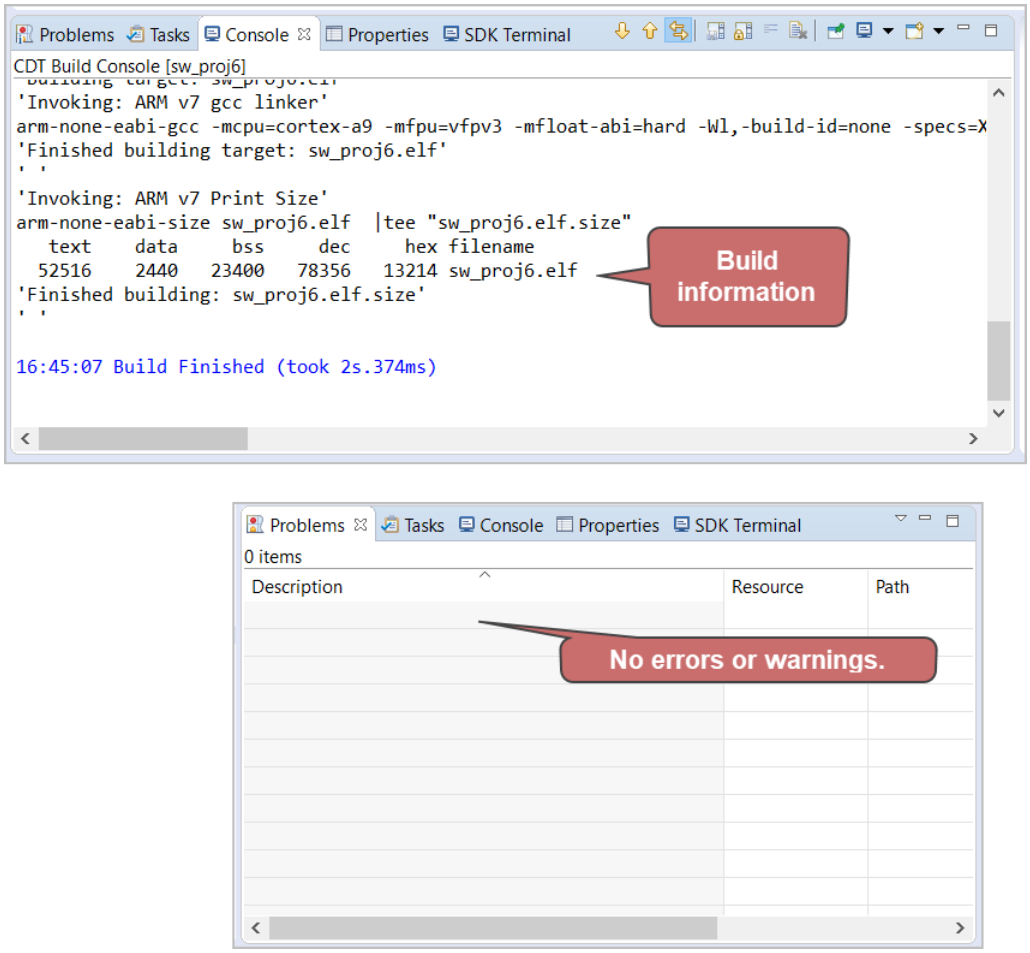


Figure 155. The project should build successfully

If "Build Automatically" is selected, the project should build successfully, with no errors or warnings.

[Exceptions: SDK 2018.3/SDK 2019.1 may indicate a warning as follows, which can be ignored:

#pragma message: For the sleep routines, Global timer is being used]

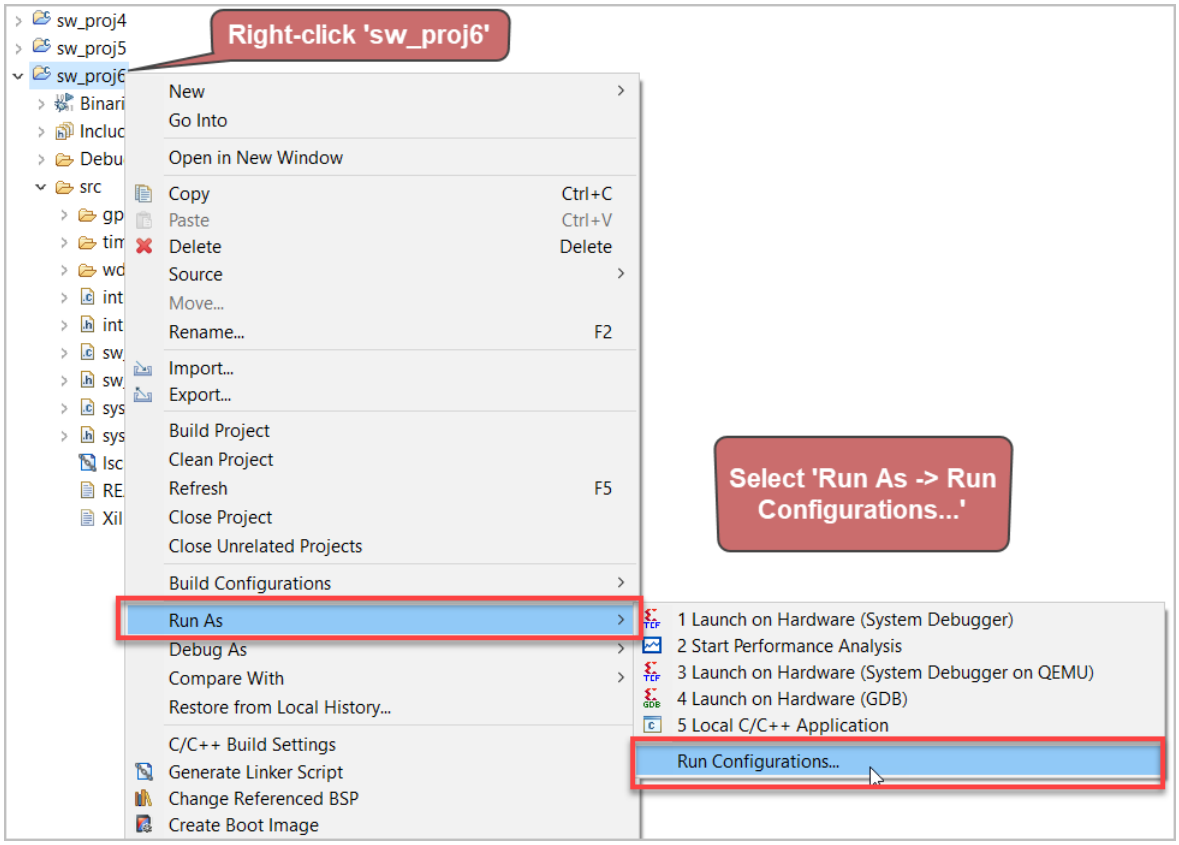


Figure 156. Right-click the software project and create a Run Configuration

To run the program, the first step is to create a Run configuration. Right-click on the project, and select the 'Run Configurations...' option.

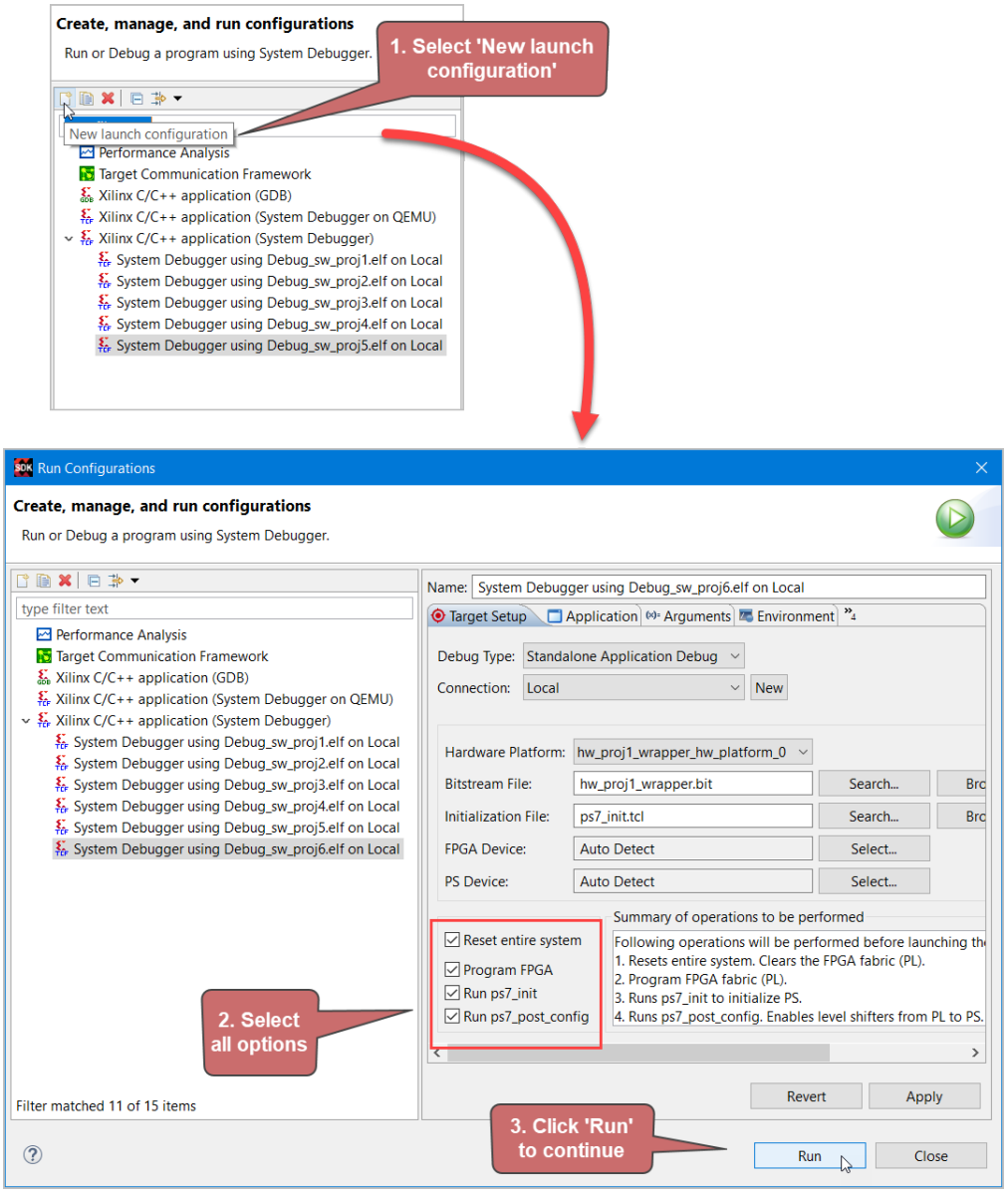


Figure 157. Create a TCF (i.e. System Debugger) option and click Run to execute the program.

1. Ensure the Xilinx C/C++ application (System Debugger) is selected.
2. Click on New Launch Configuration.
3. Select all options:
    a. Reset entire system
    b. Program FPGA
    c. Run ps7_init
    d. Run ps7_post_config

4. Click on 'Run' to continue.

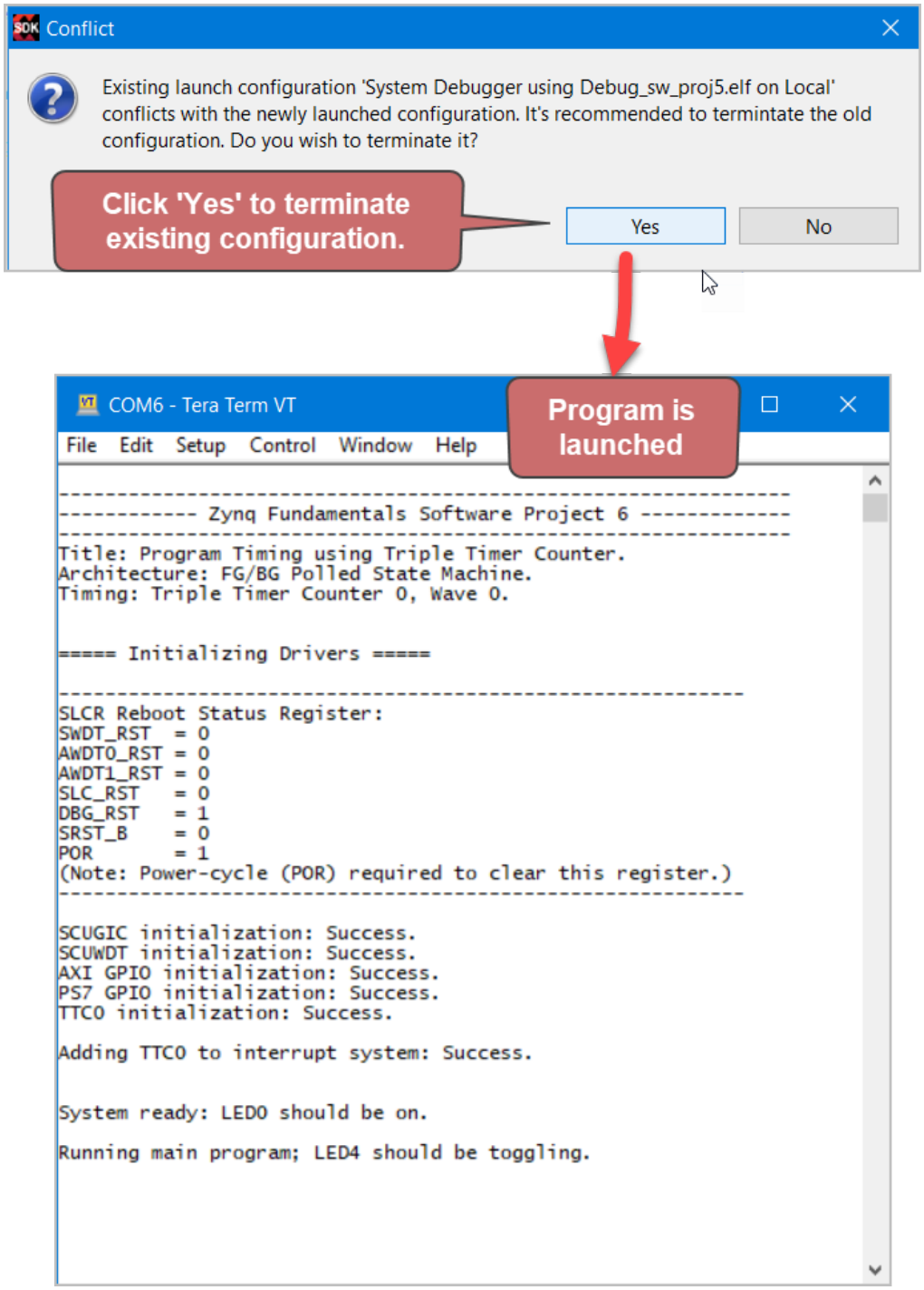


Figure 158. Terminal output for Software Project 6

If prompted, select 'Yes' to terminate any existing configuration. The FPGA will be programmed and the application will be downloaded to the processor. The terminal program should display the results for launching Software Project 6.

3.7 Software Project 7: Button De-bouncing

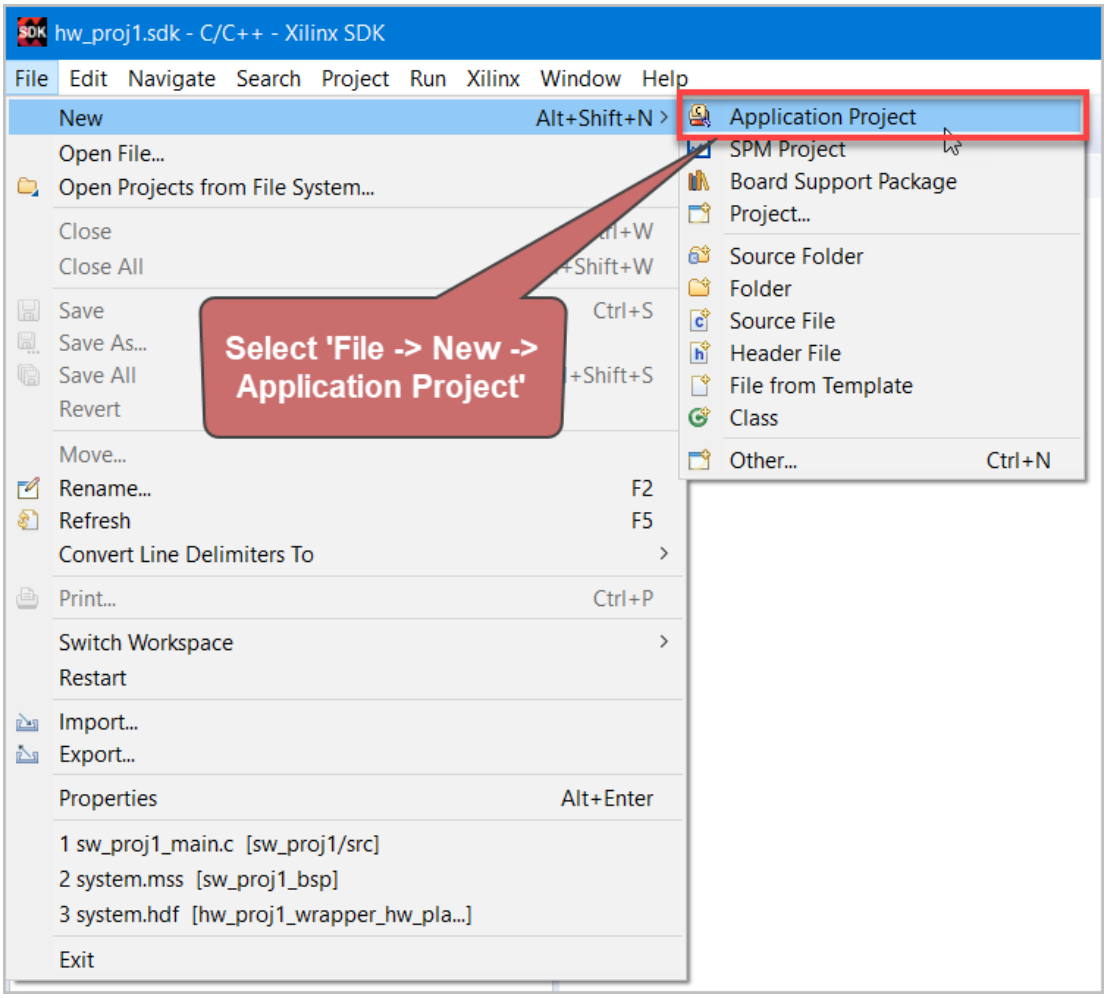


Figure 159. Select File->New-> Application Project

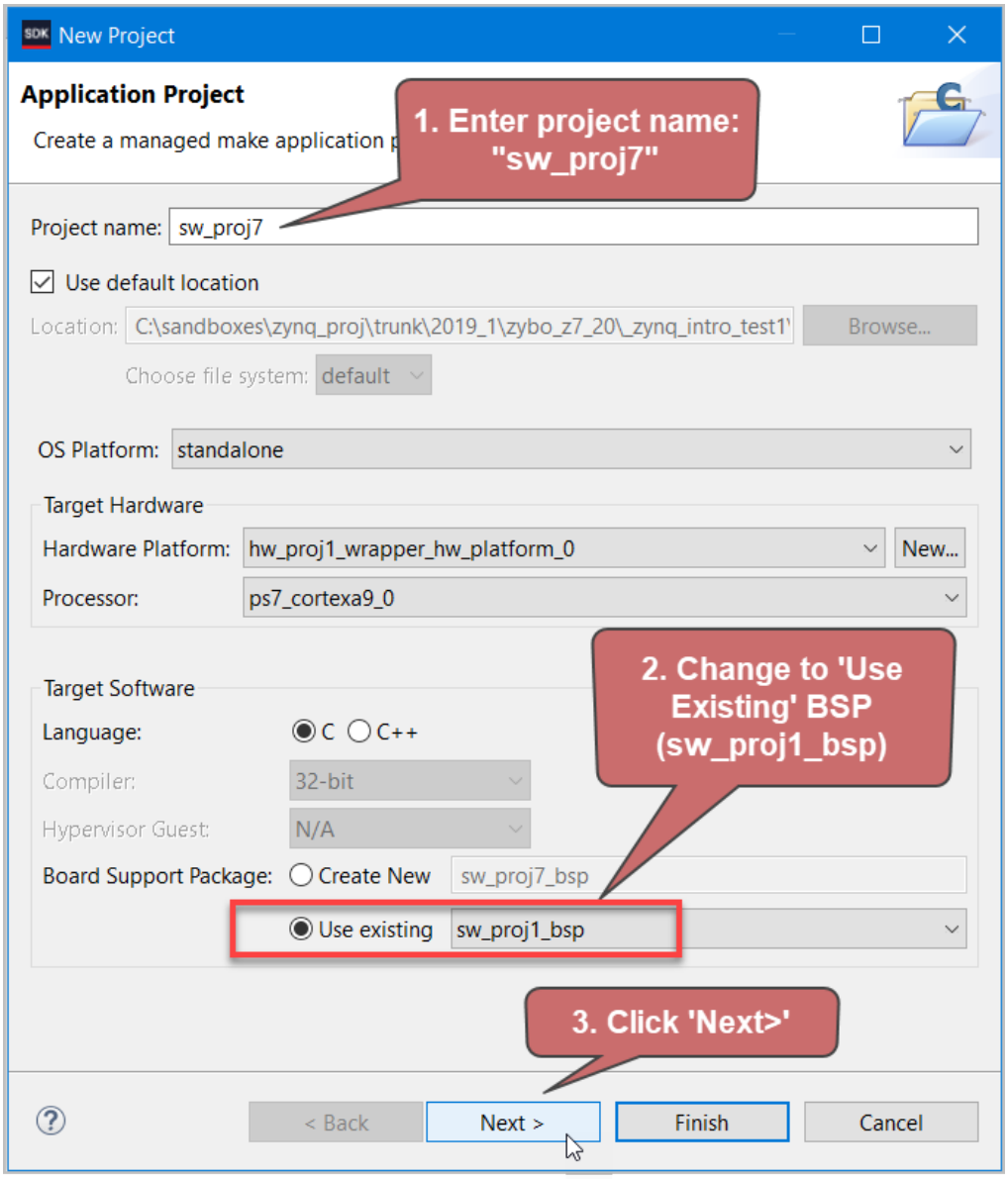


Figure 160. Enter the project details (part 1)

1. Enter the project name: sw_proj7
2. Board Support Package: Use existing
3. Click Next (don't click Finish!)

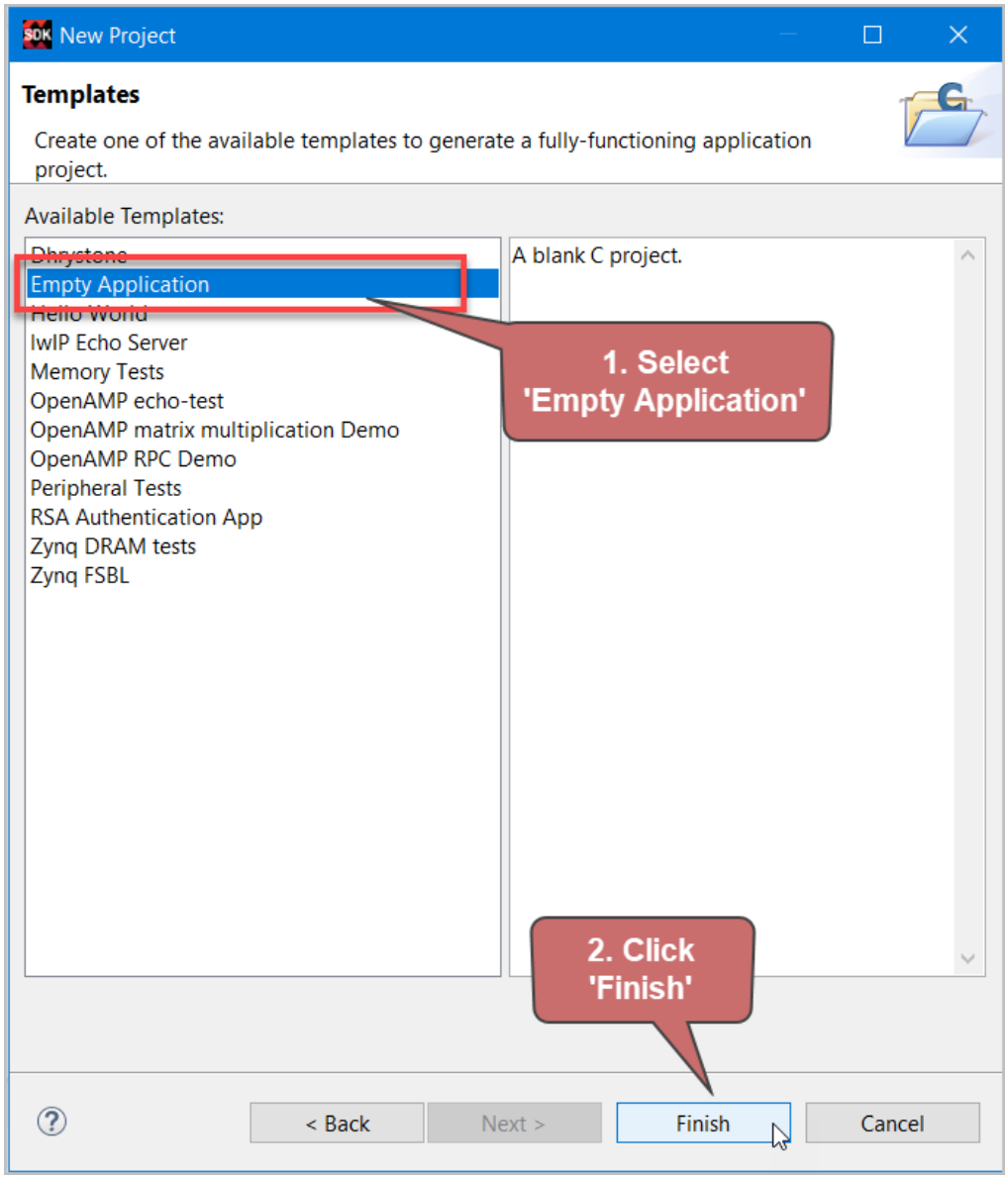


Figure 161. Enter the project details (part 2)

1. Select Empty Application
2. Click Finish

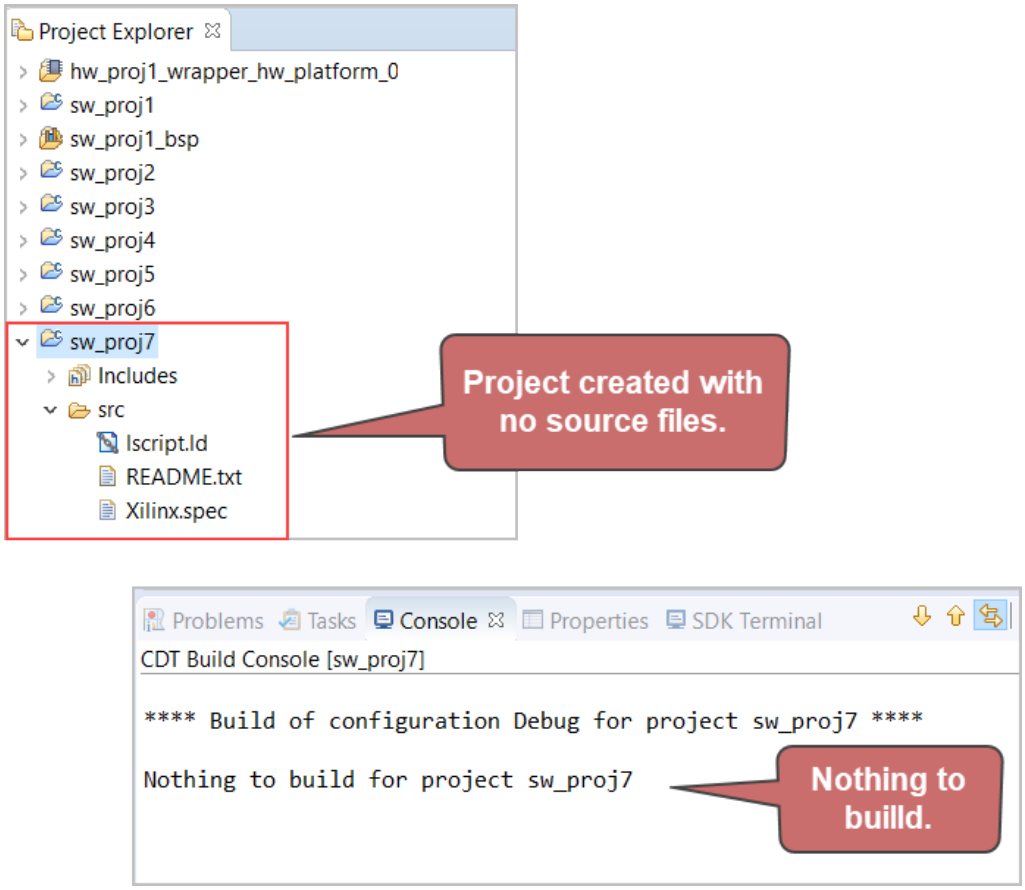


Figure 163. Project created with no source files

A project named “sw_proj7” is created with no project files. The CDT Build Console confirms that there is nothing to build.

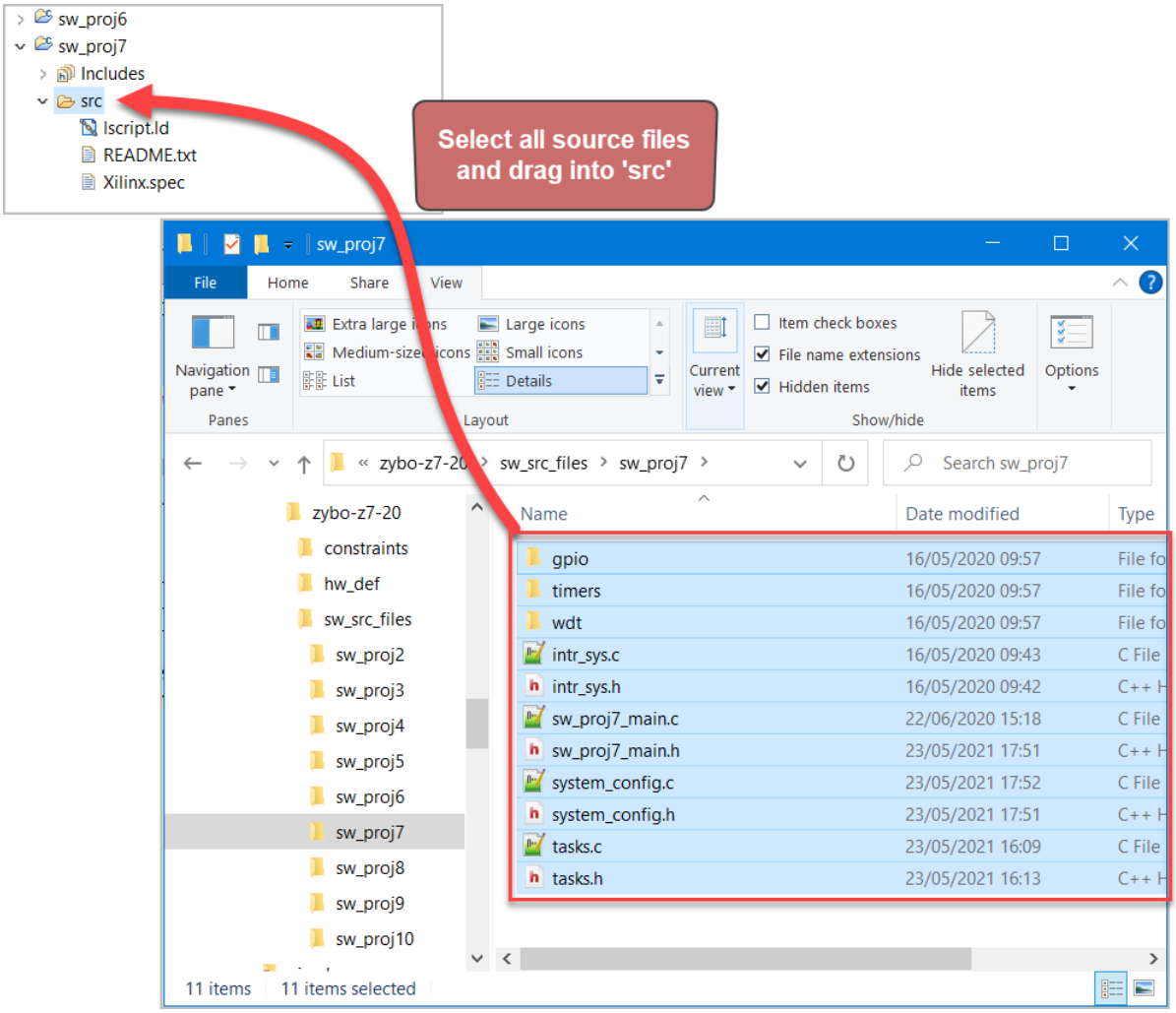


Figure 164. Drag the source files from Windows Explorer into the SDK project

Open the location where the source files for software project 7 are saved on your PC. Drag all the files and directory and from Windows explorer directly into the "**src**" folder in SDK.

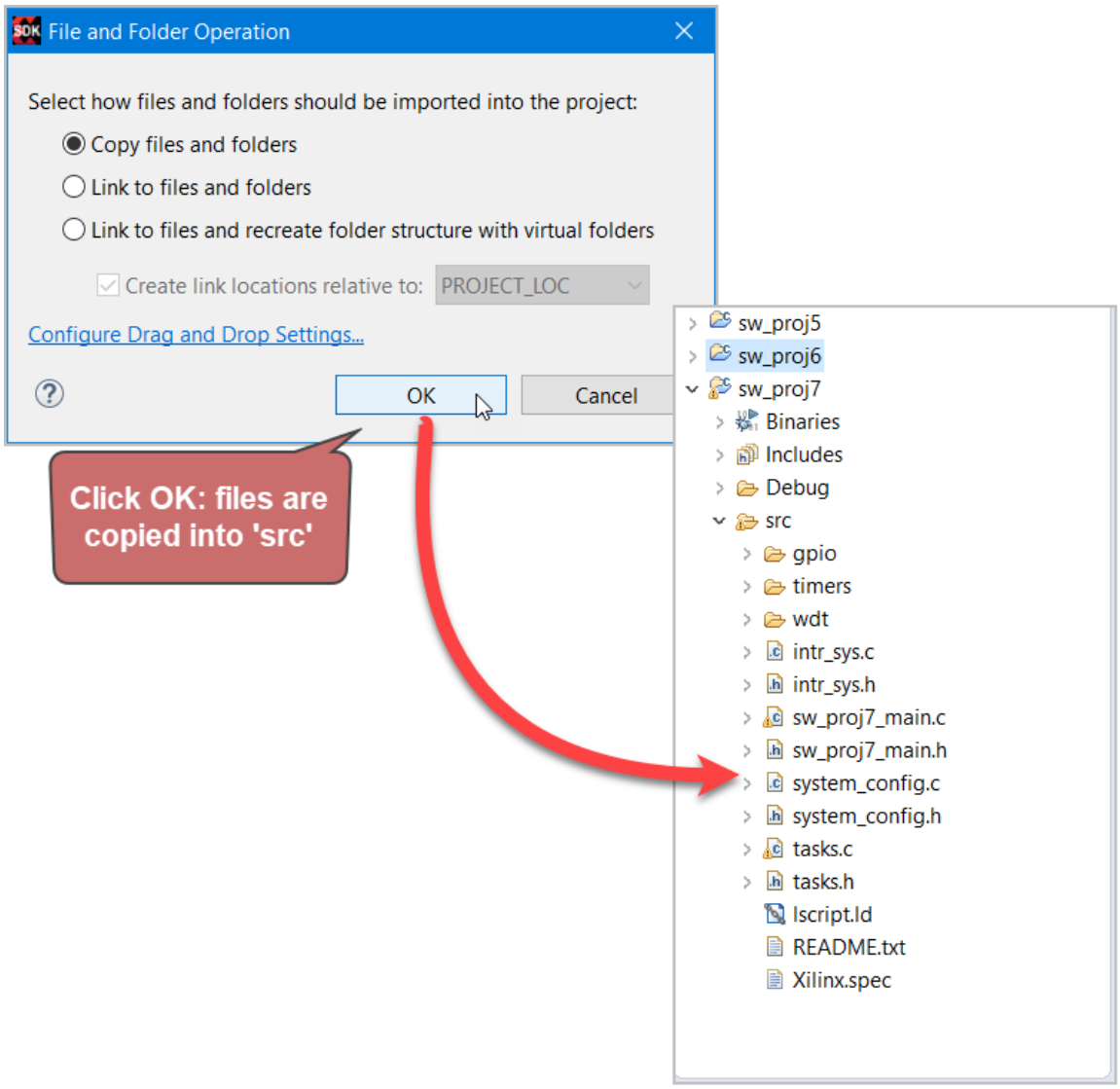


Figure 165. Click OK, and the files will be copied into the project.

In the File Operation dialog box, ensure “Copy files and folders” is checked, and click OK.

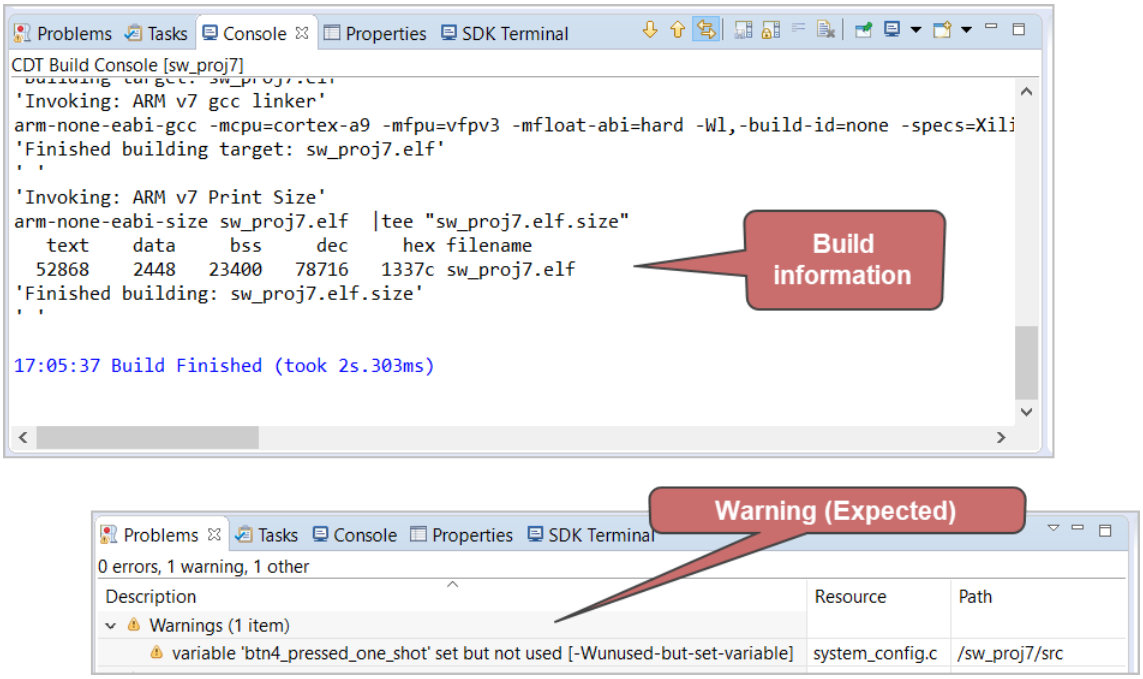


Figure 166. The project should build successfully, but a warning will be generated (this is expected).

If “Build Automatically” is selected, the project should build successfully, although for this project, a warning is expected, as follows:

- variable 'btn4_pressed_one_shot' set but not used

[SDK 2018.3/SDK 2019.1 may also indicate an information message as follows, which can be ignored:

#pragma message: For the sleep routines, Global timer is being used]

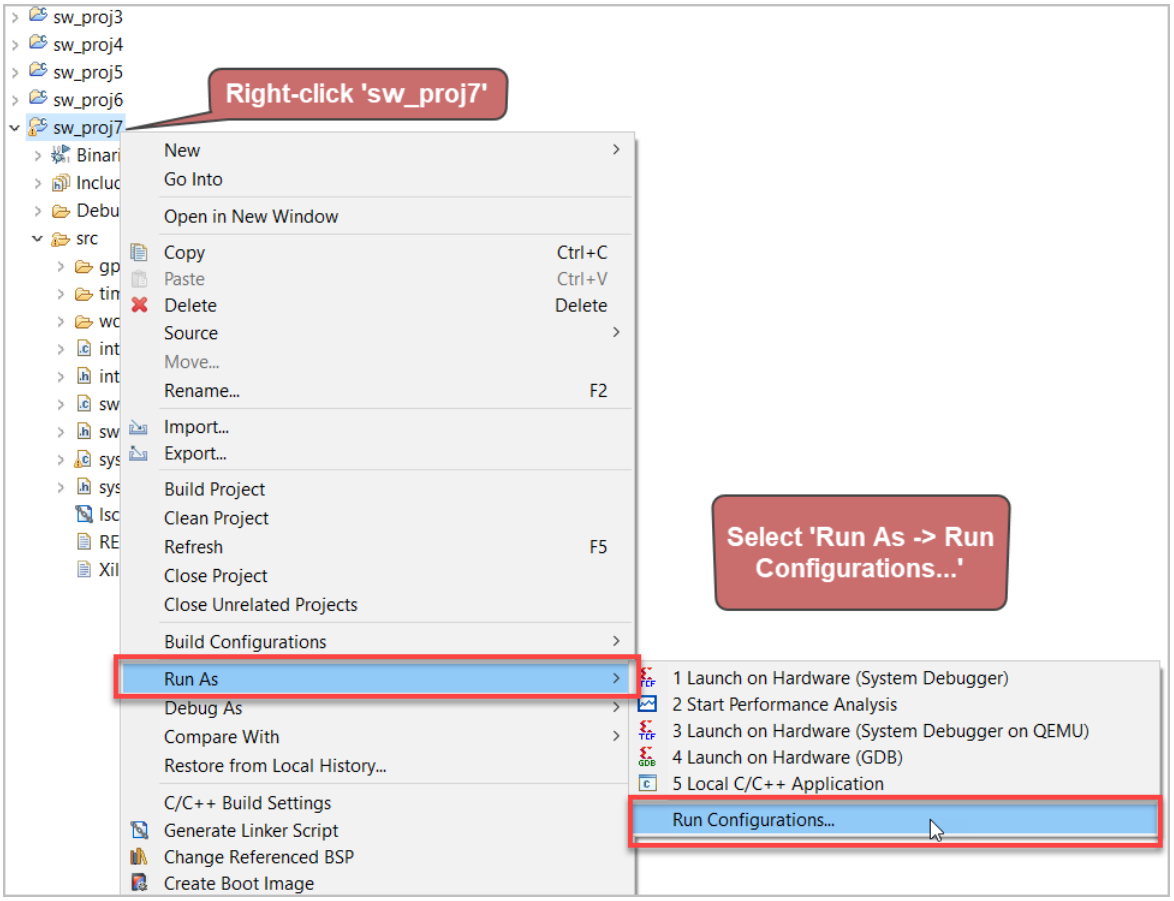


Figure 167. Right-click the software project and create a Run Configuration

To run the program, the first step is to create a Run configuration. Right-click on the project, and select the 'Run Configurations...' option.

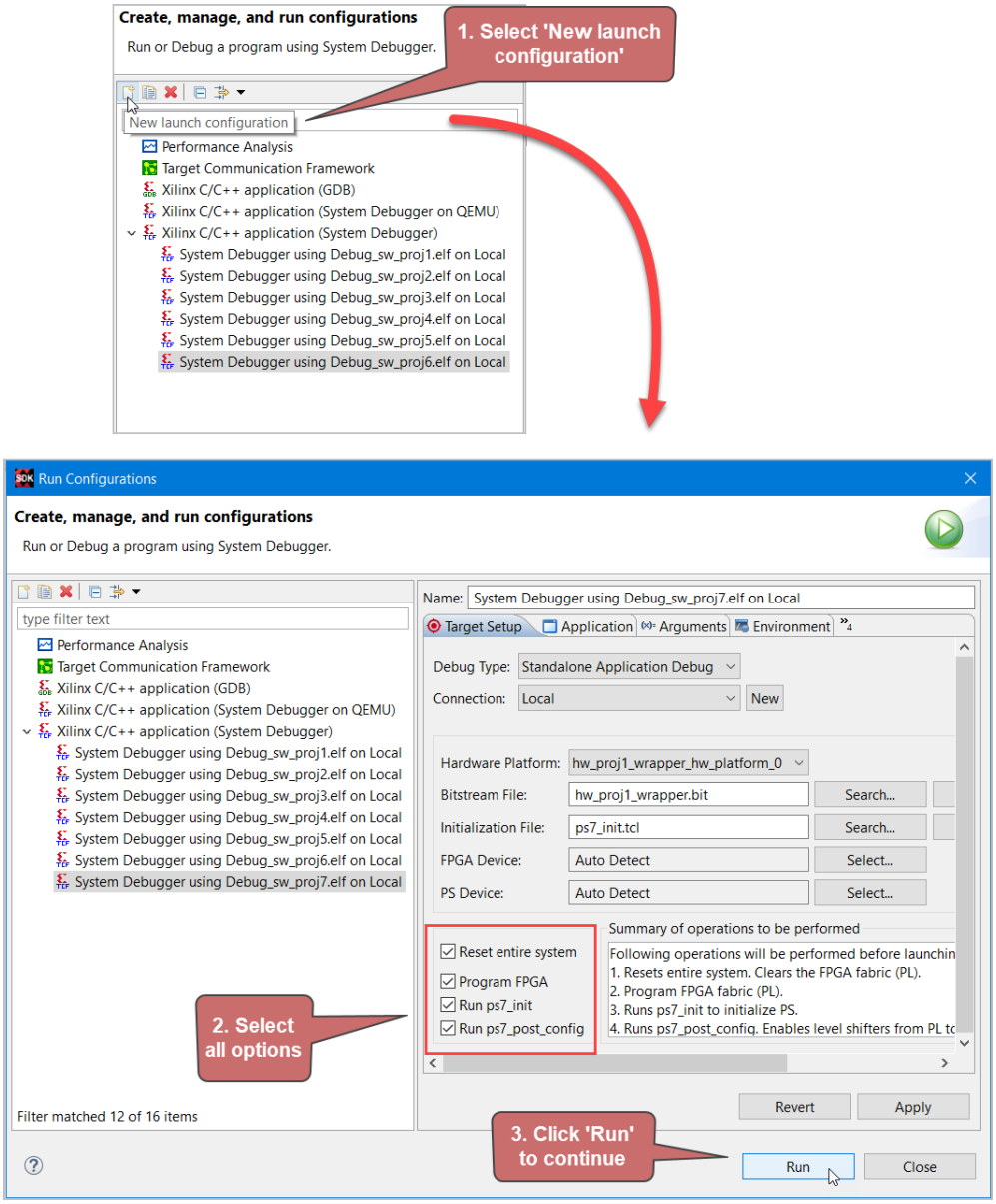


Figure 168. Create a TCF (i.e. System Debugger) option and click Run to execute the program.

1. Ensure the Xilinx C/C++ application (System Debugger) is selected.
2. Click on New Launch Configuration.
3. Select all options:
    a. Reset entire system
    b. Program FPGA
    c. Run ps7_init
    d. Run ps7_post_config

4. Click on 'Run' to continue.

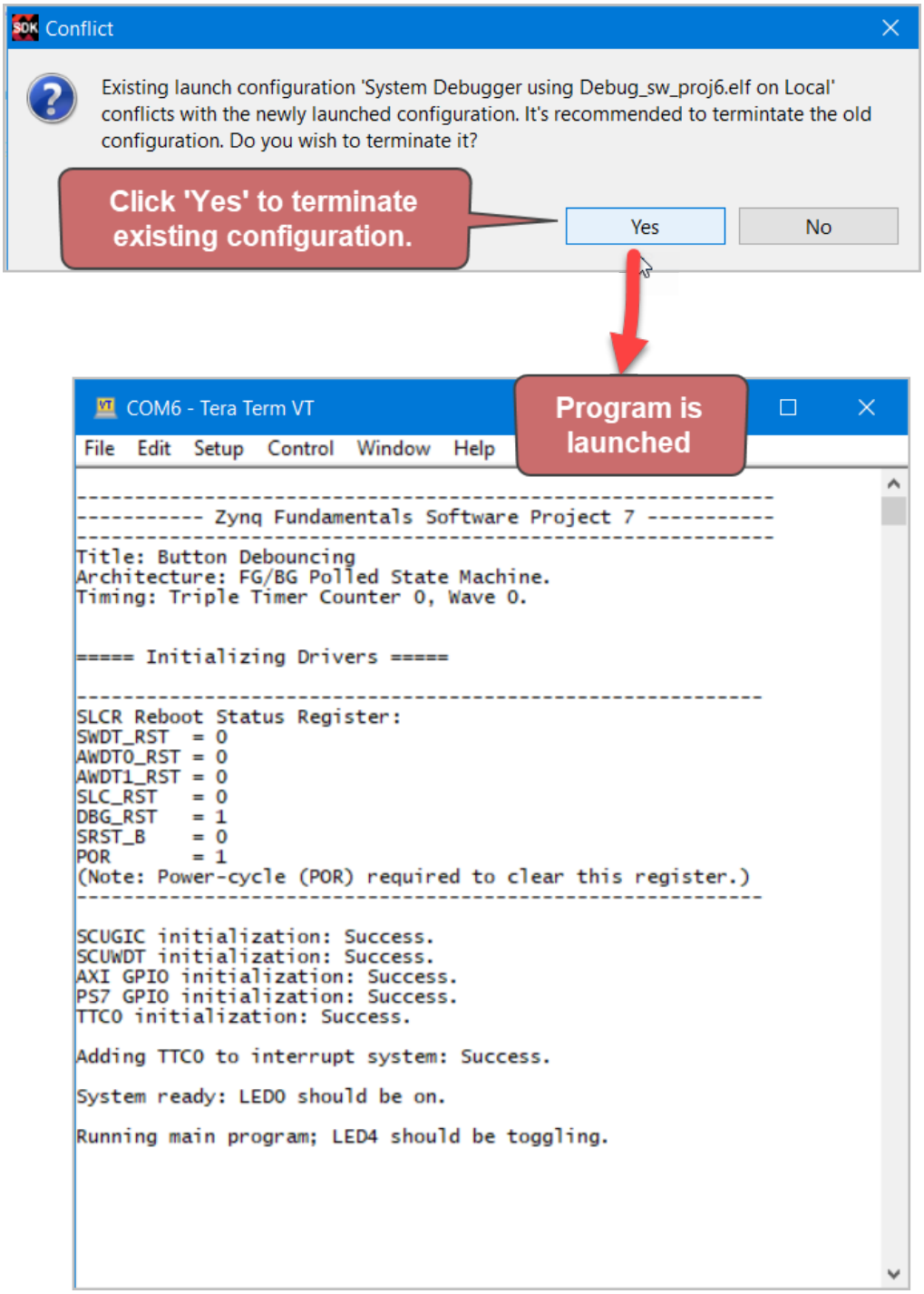


Figure 169. Terminal output for Software Project 7

If prompted, select 'Yes' to terminate any existing configuration. The FPGA will be programmed and the application will be downloaded to the processor. The terminal program should display the results for launching Software Project 7.

3.8 Software Project 8: UART Command Handler

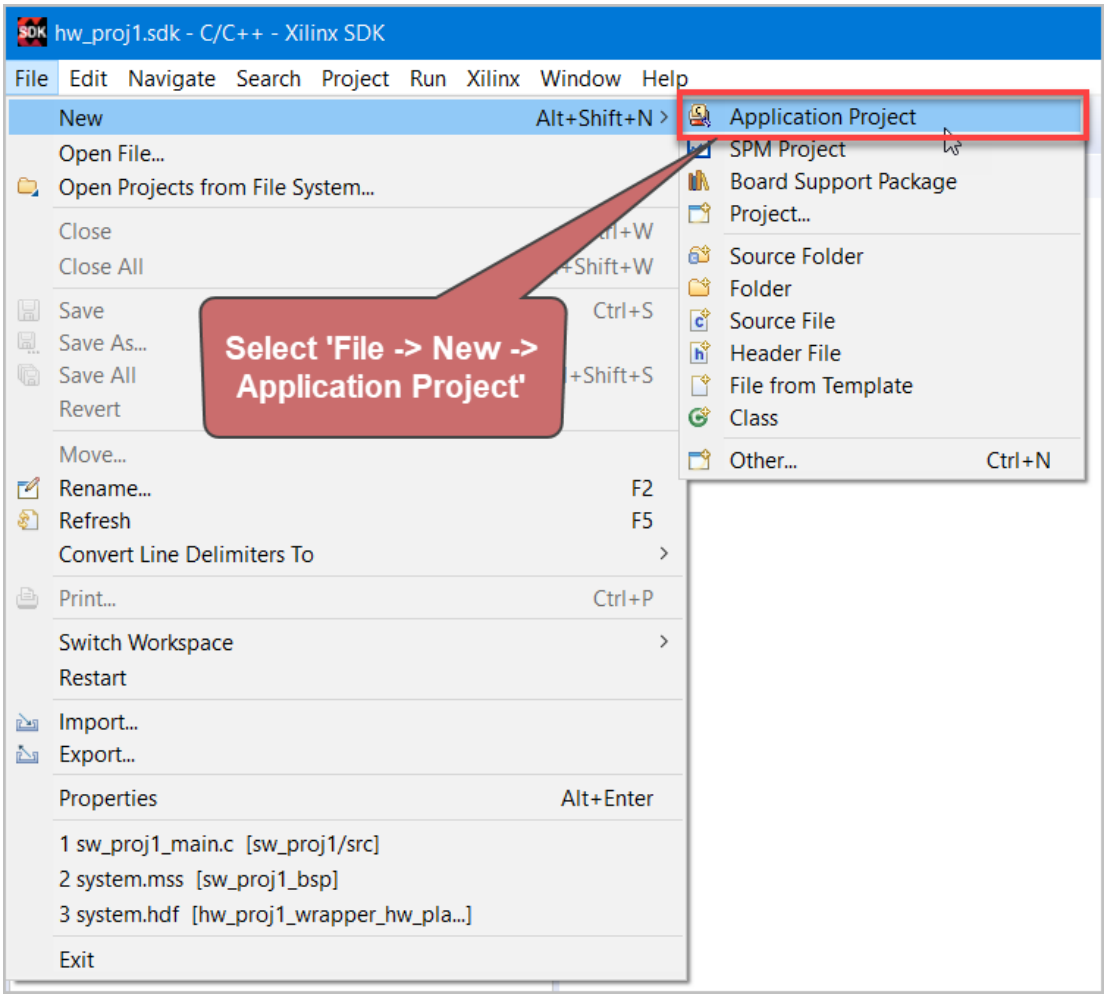


Figure 170. Select File->New-> Application Project

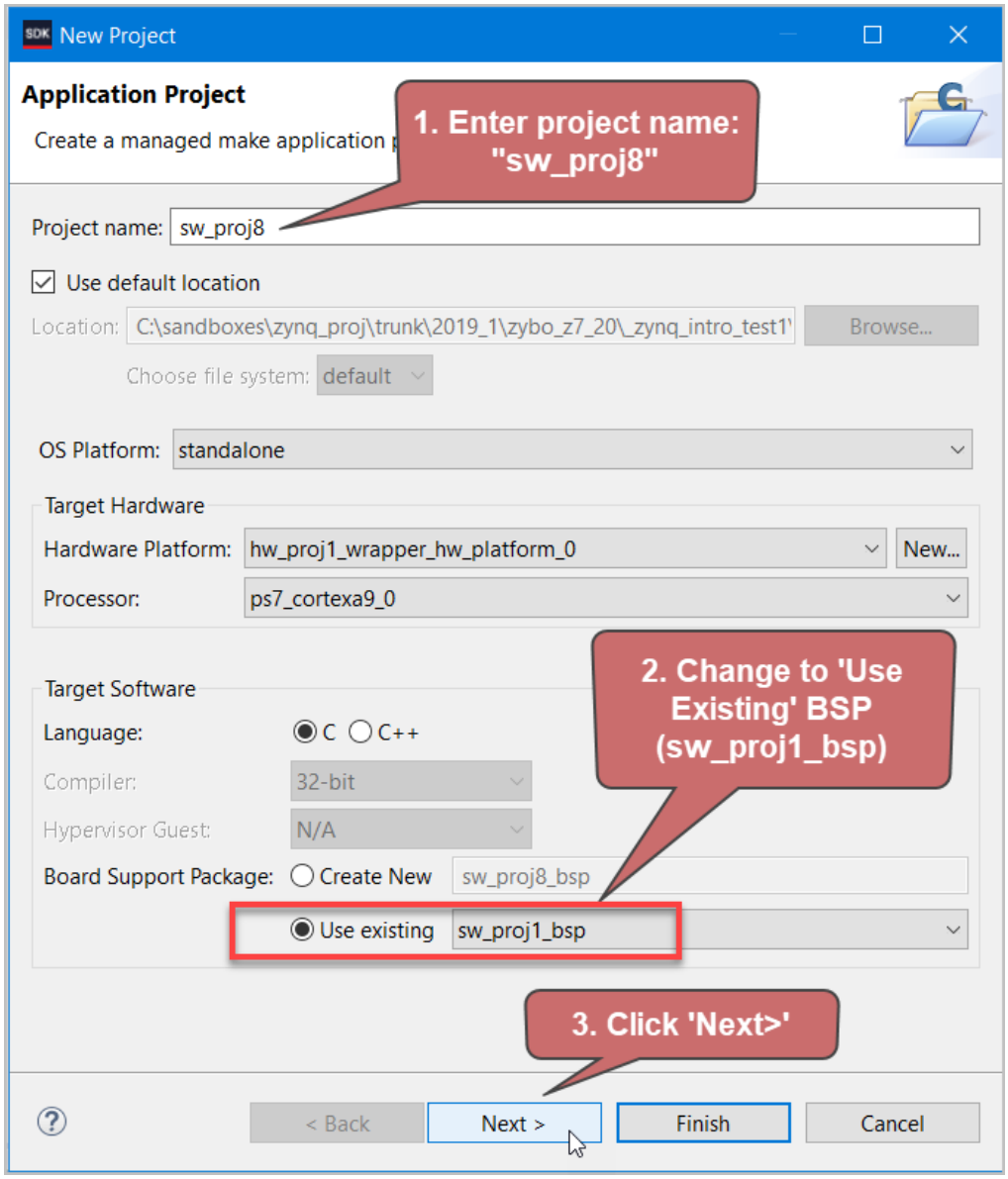


Figure 171. Enter the project details (part 1)

1. Enter the project name: sw_proj8
2. Board Support Package: Use existing
3. Click Next (don't click Finish!)

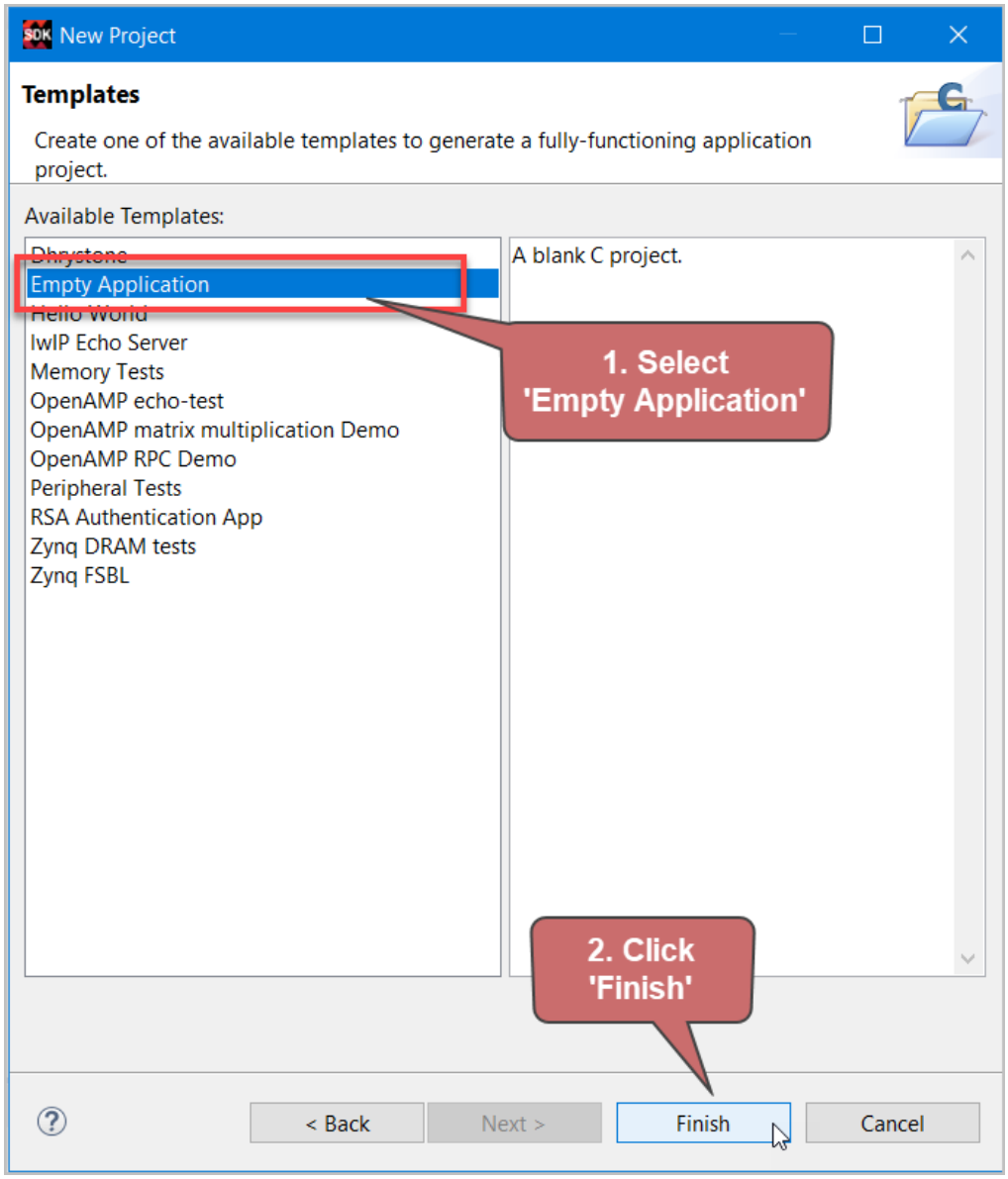


Figure 172. Enter the project details (part 2)

1. Select Empty Application
2. Click Finish

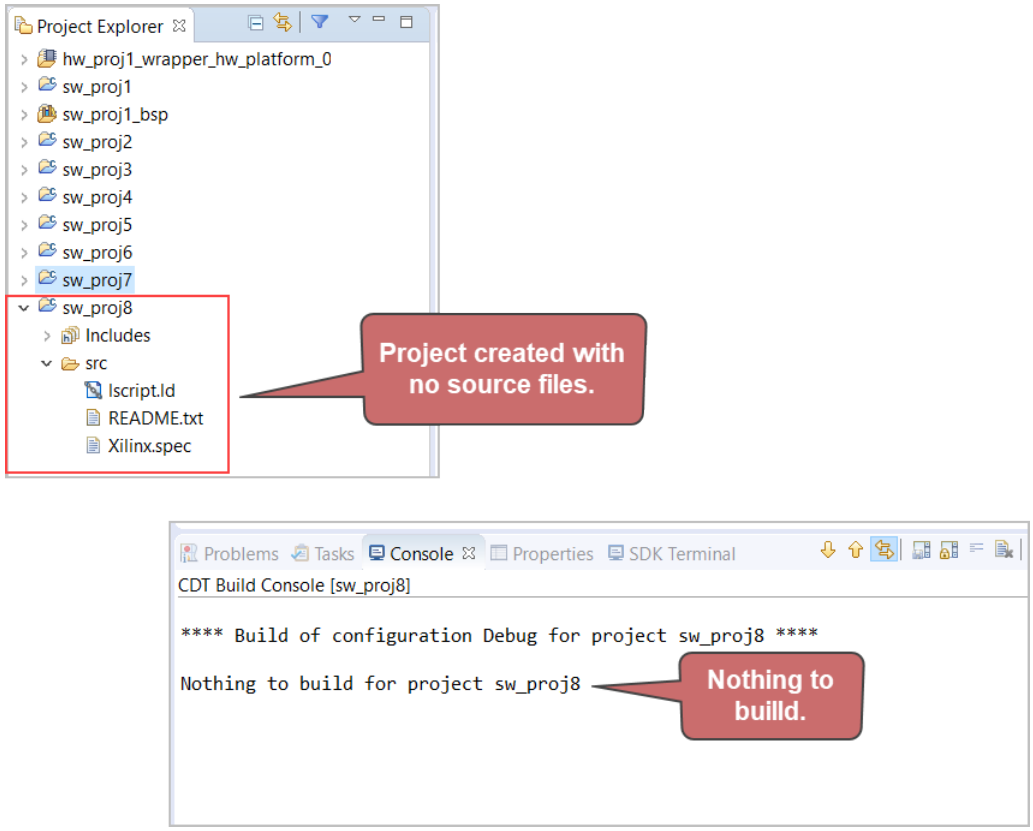


Figure 174. Project created with no source files

A project named “sw_proj8” is created with no project files. The CDT Build Console confirms that there is nothing to build.

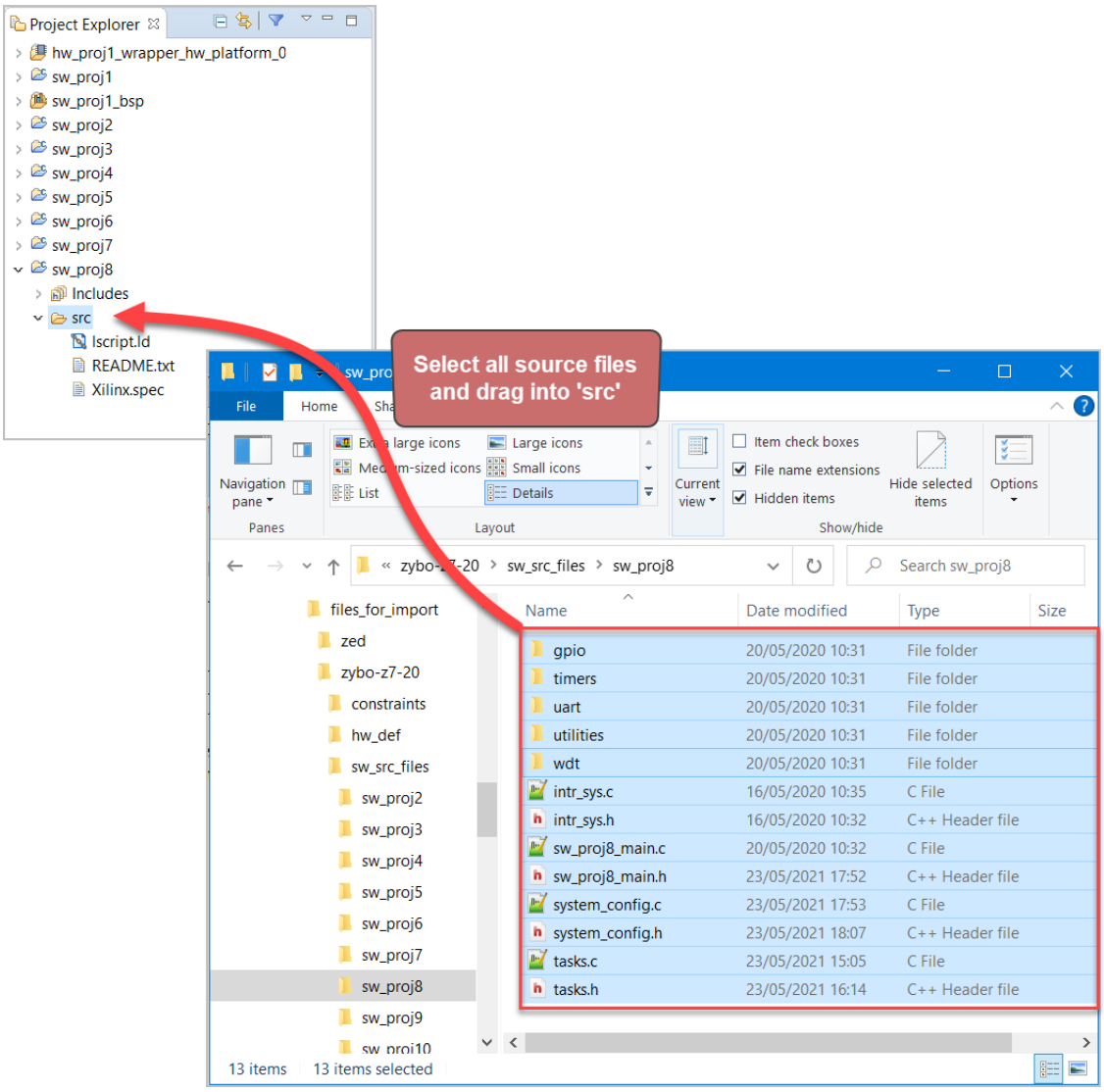


Figure 175. Drag the source files from Windows Explorer into the SDK project

Open the location where the source files for software project 8 are saved on your PC. Drag all the files and directory and from Windows explorer directly into the "**src**" folder in SDK.

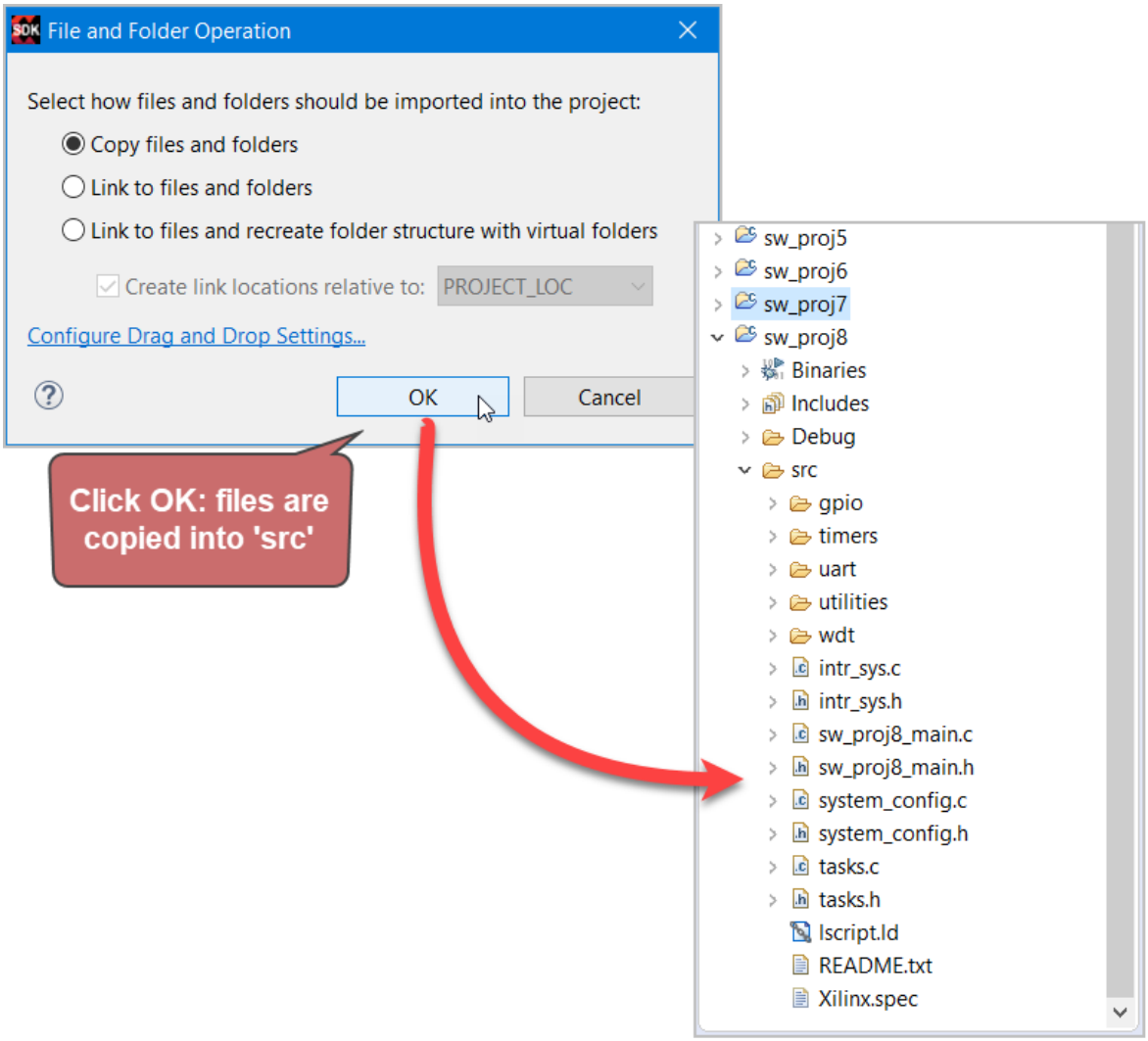


Figure 176. Click OK, and the files will be copied into the project.

In the File Operation dialog box, ensure “Copy files and folders” is checked, and click OK.

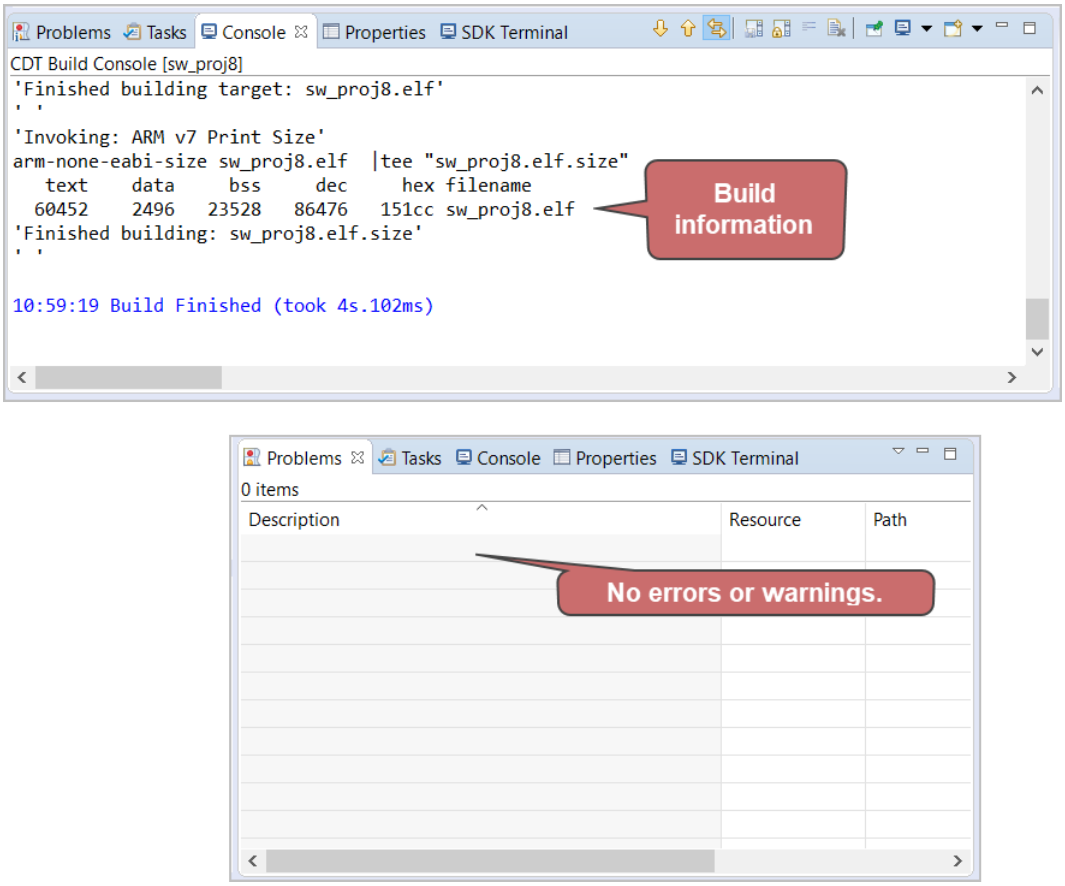


Figure 177. The project should build successfully.

If "Build Automatically" is selected, the project should build successfully, with no errors or warnings.

[Exceptions: SDK 2018.3/SDK 2019.1 may indicate a warning as follows, which can be ignored:

#pragma message: For the sleep routines, Global timer is being used

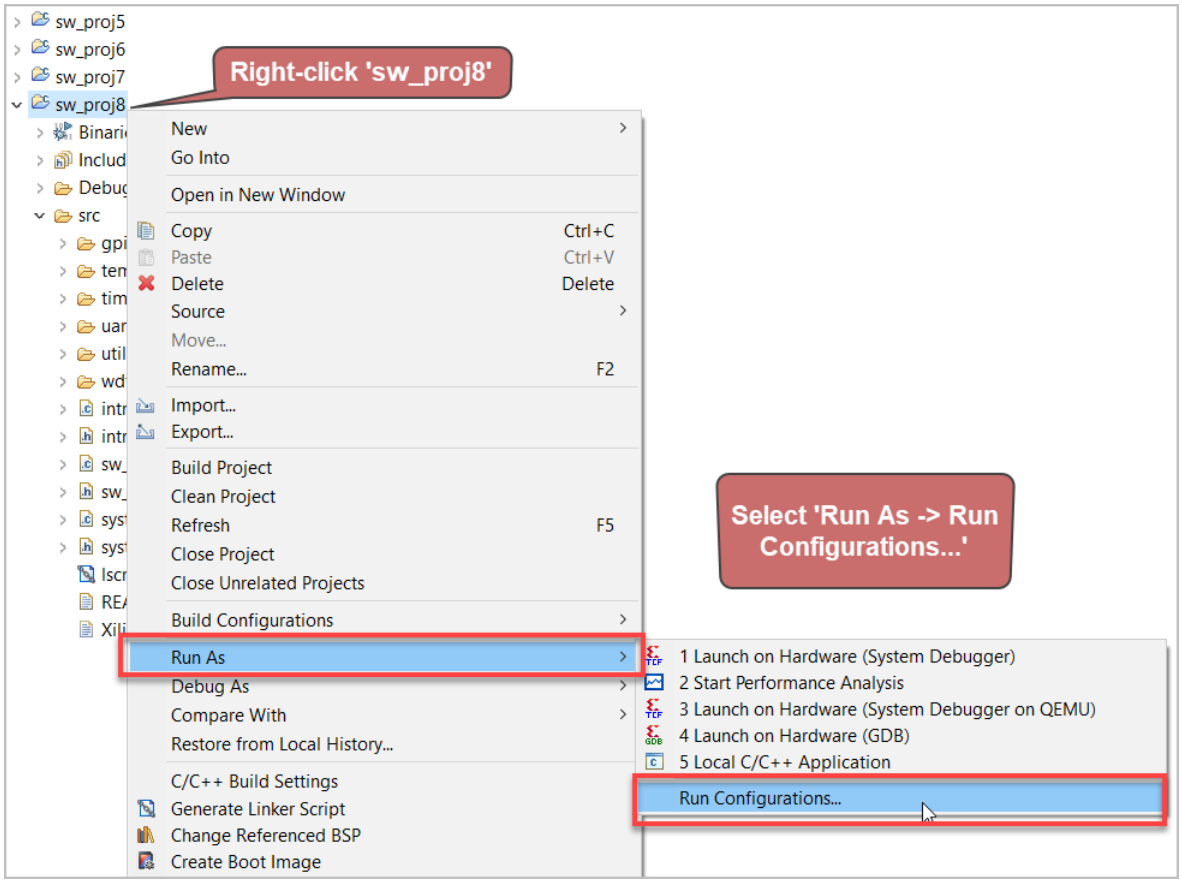


Figure 178. Right-click the software project and create a Run Configuration

To run the program, the first step is to create a Run configuration. Right-click on the project, and select the 'Run Configurations...' option.

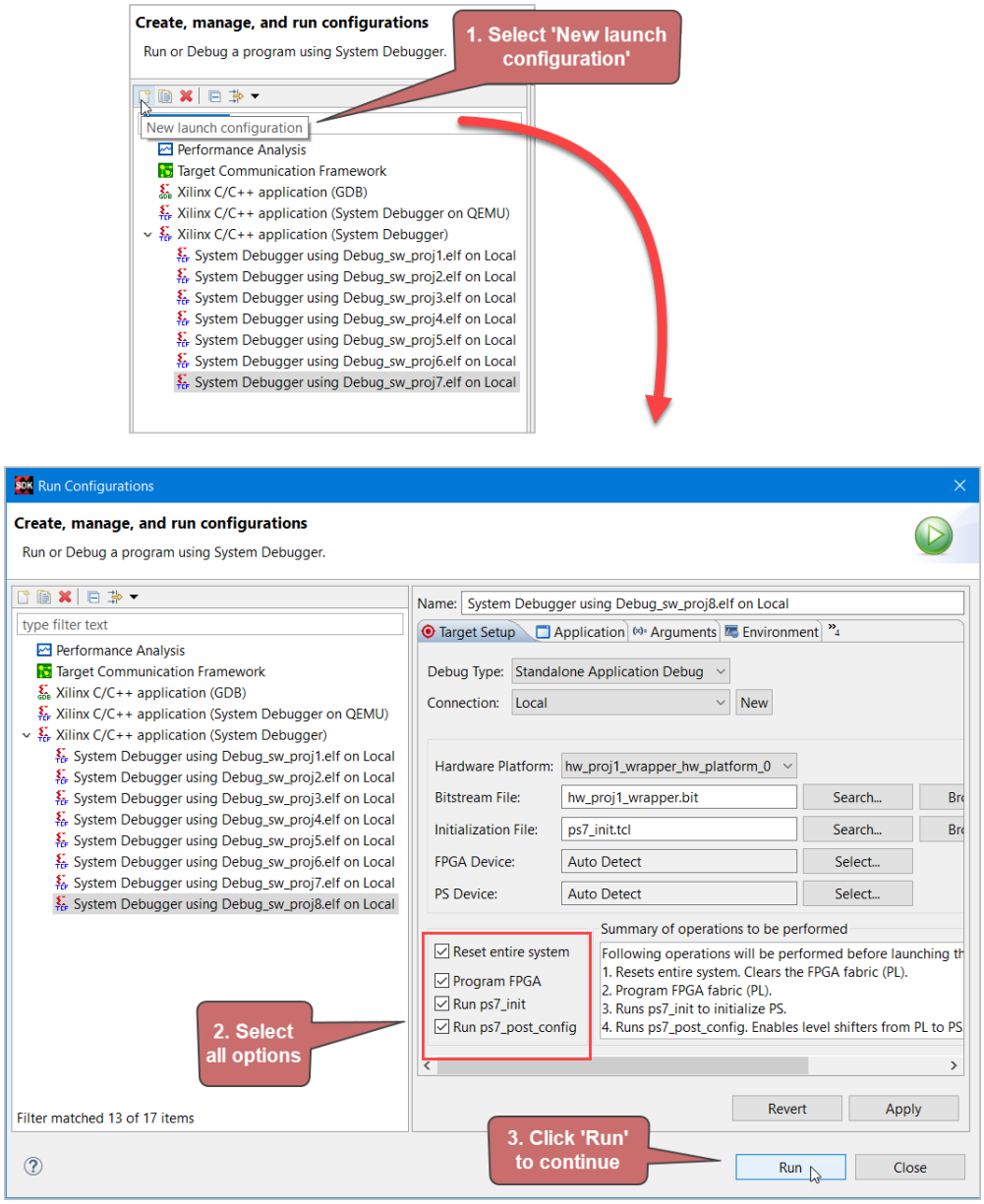


Figure 179. Create a TCF (i.e. System Debugger) option and click Run to execute the program.

1. Ensure the Xilinx C/C++ application (System Debugger) is selected.
2. Click on New Launch Configuration.
3. Select all options:
    a. Reset entire system
    b. Program FPGA
    c. Run ps7_init
    d. Run ps7_post_config

4. Click on 'Run' to continue.

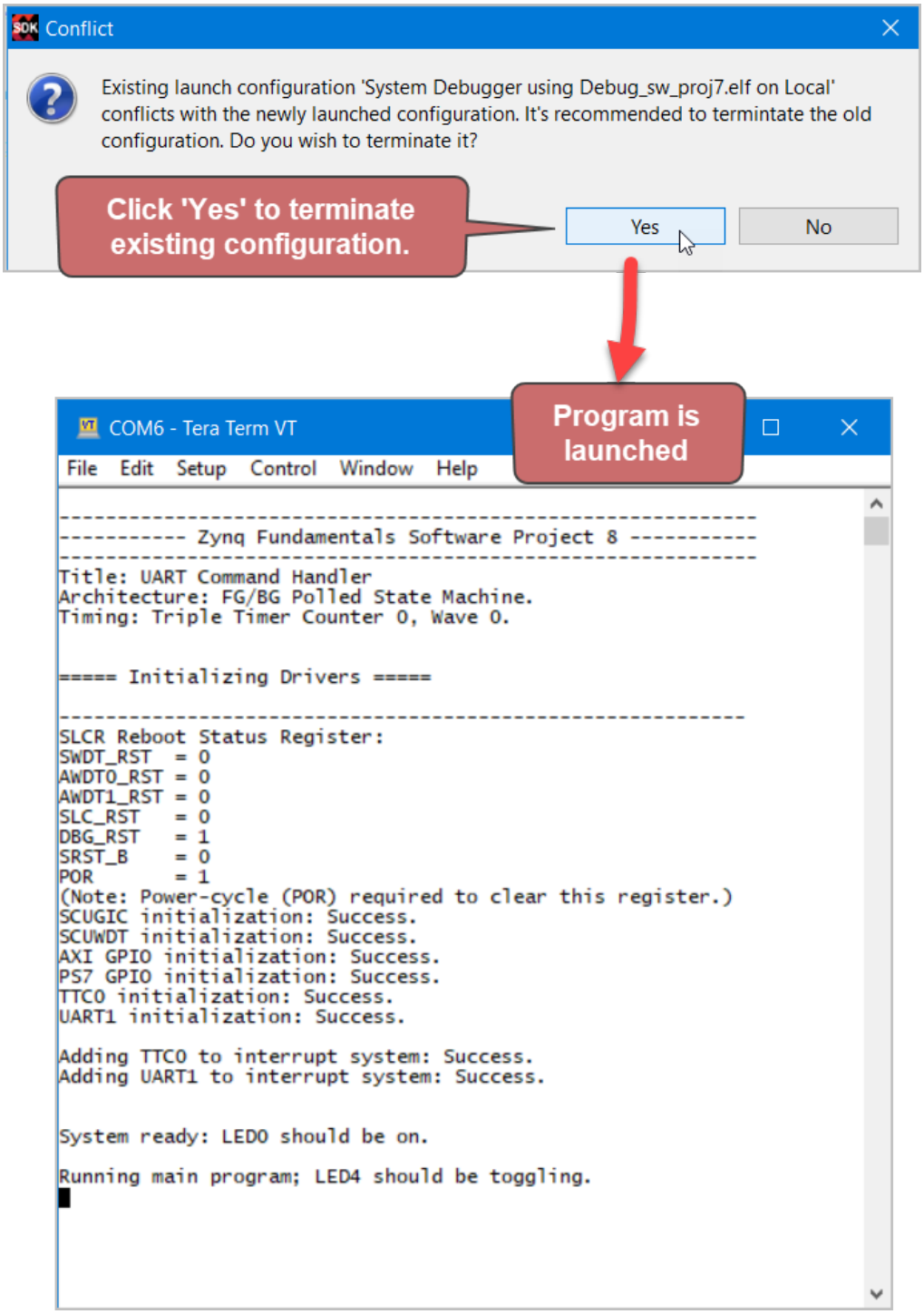


Figure 180. Terminal output for Software Project 8

If prompted, select 'Yes' to terminate any existing configuration. The FPGA will be programmed and the application will be downloaded to the processor. The terminal program should display the results for launching Software Project 8.

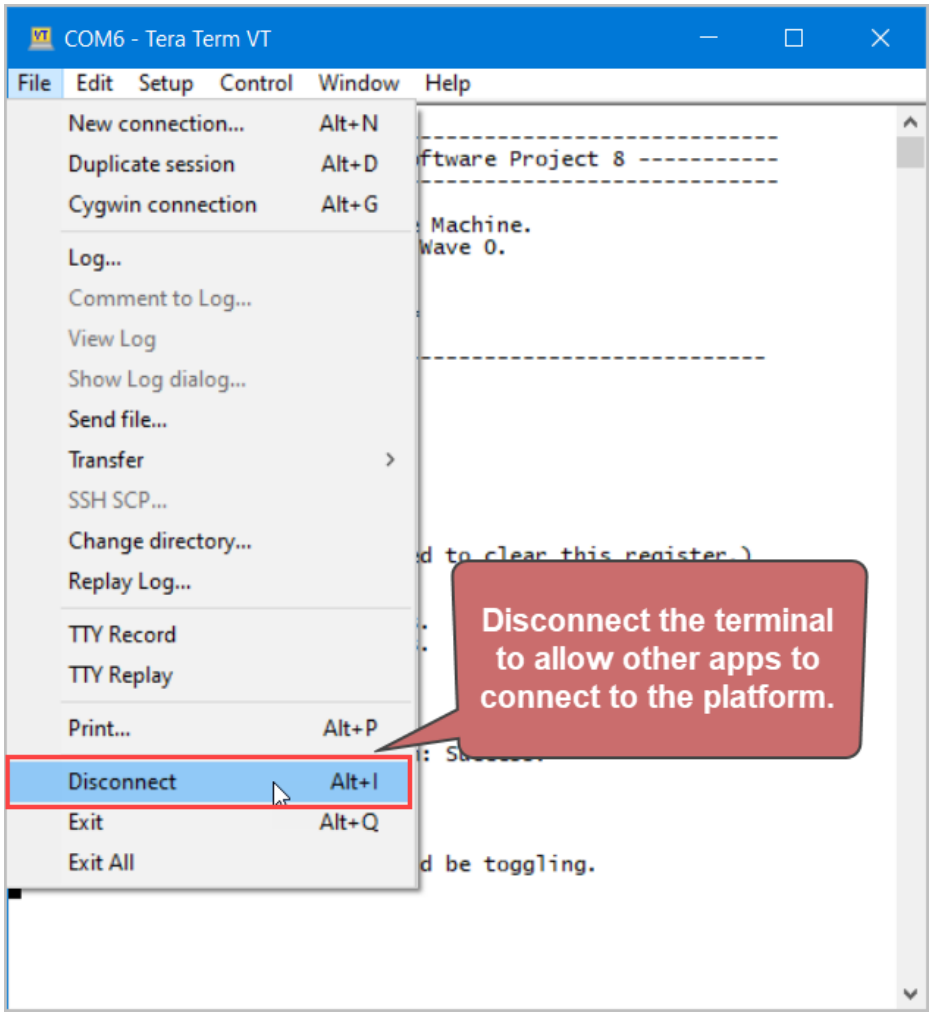


Figure 181. Disconnect the platform.

If the user wants to communicate with the command handler on the platform using a host application, the terminal needs to be disconnected.

3.9 Software Project 9: Programmable Logic Interrupts

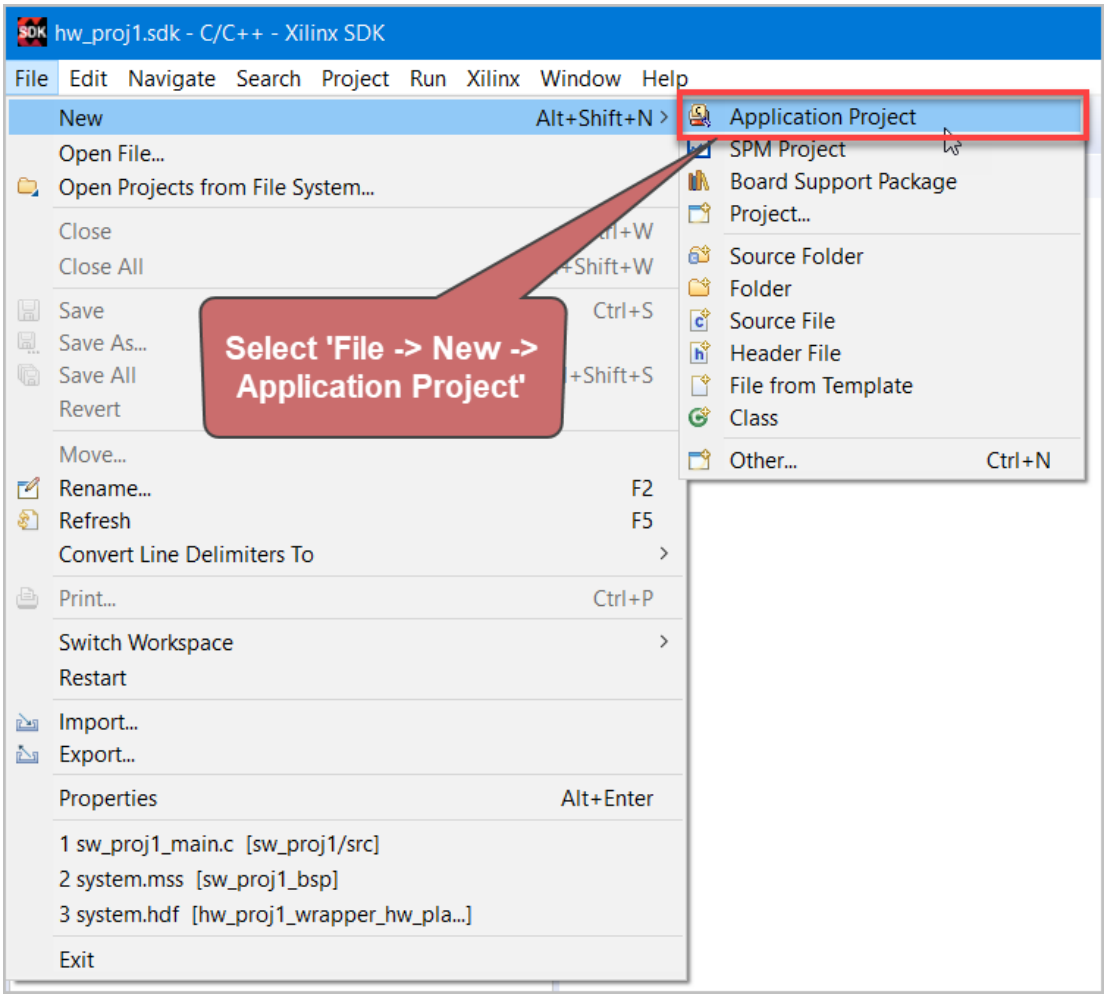


Figure 182. Select File->New-> Application Project

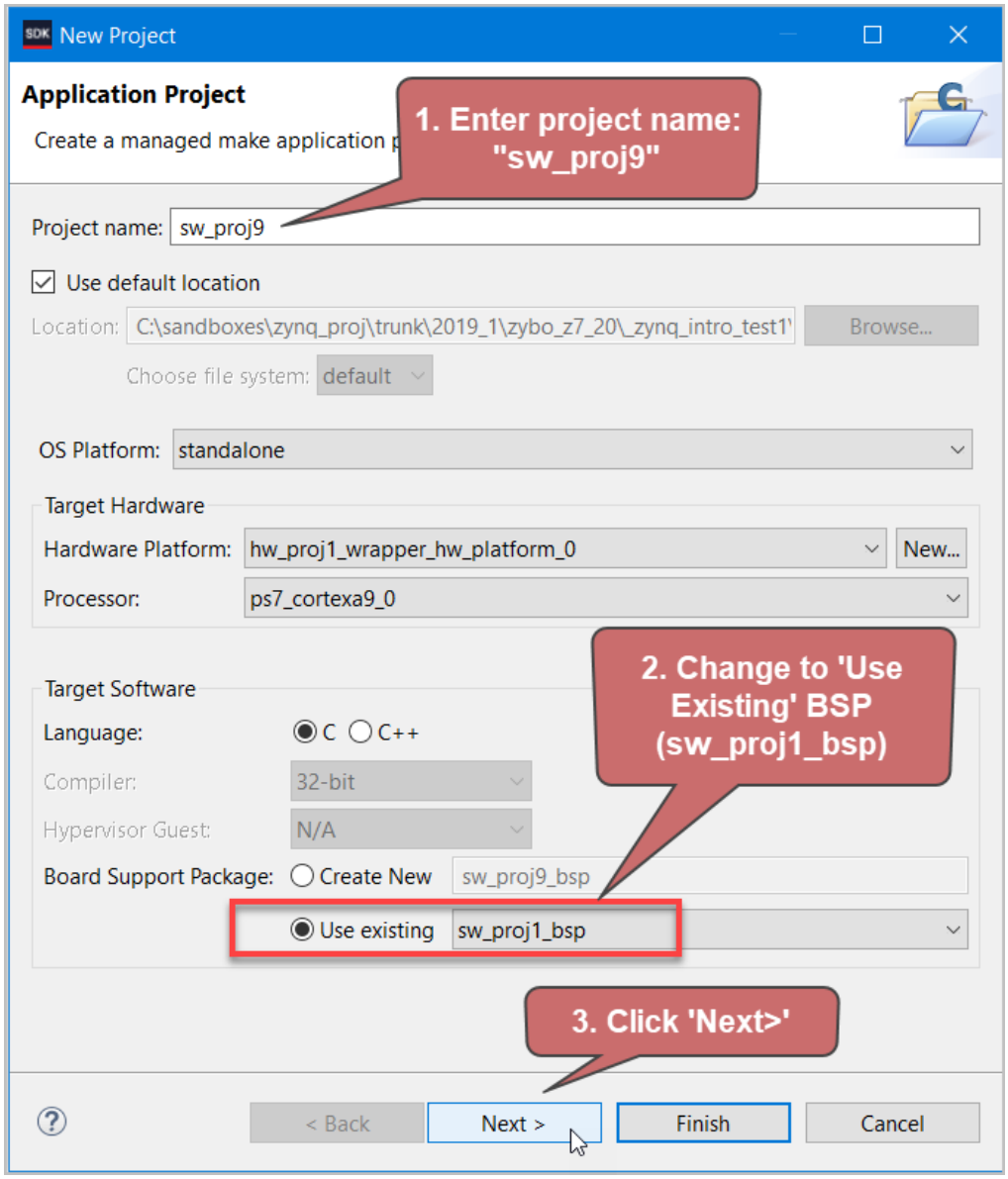


Figure 183. Enter the project details (part 1)

1. Enter the project name: sw_proj9
2. Board Support Package: Use existing
3. Click Next (don't click Finish!)

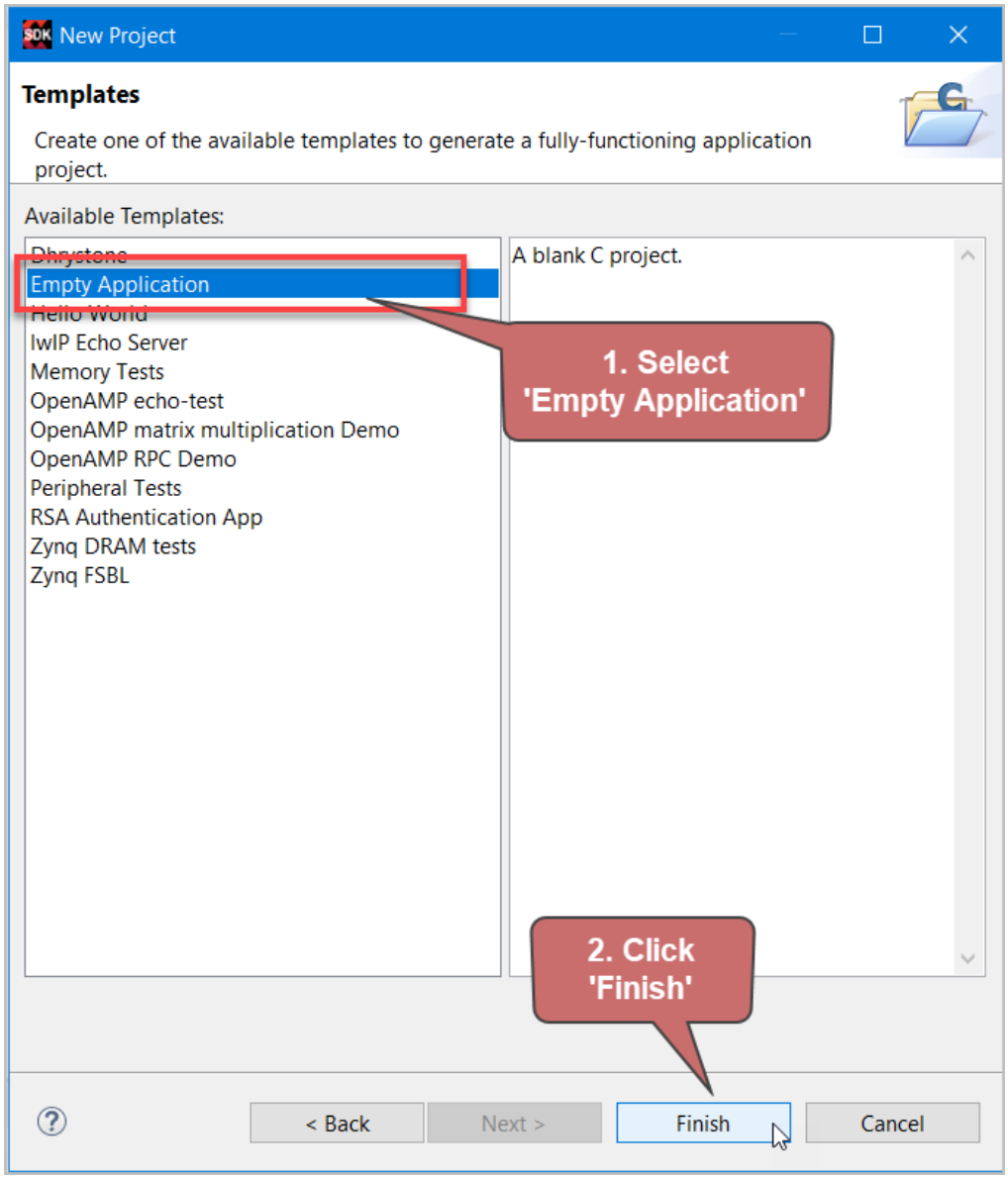


Figure 184. Enter the project details (part 2)

1. Select Empty Application
2. Click Finish

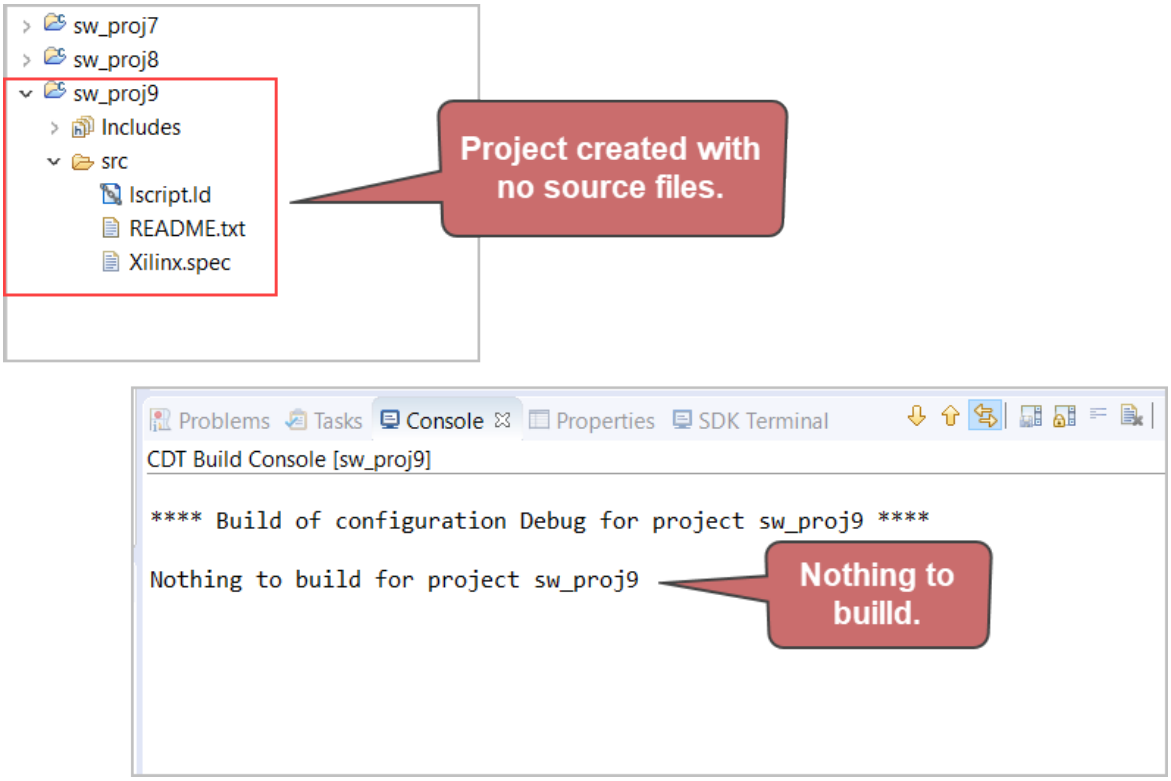


Figure 186. Project created with no source files

A project named “sw_proj9” is created with no project files. The CDT Build Console confirms that there is nothing to build.

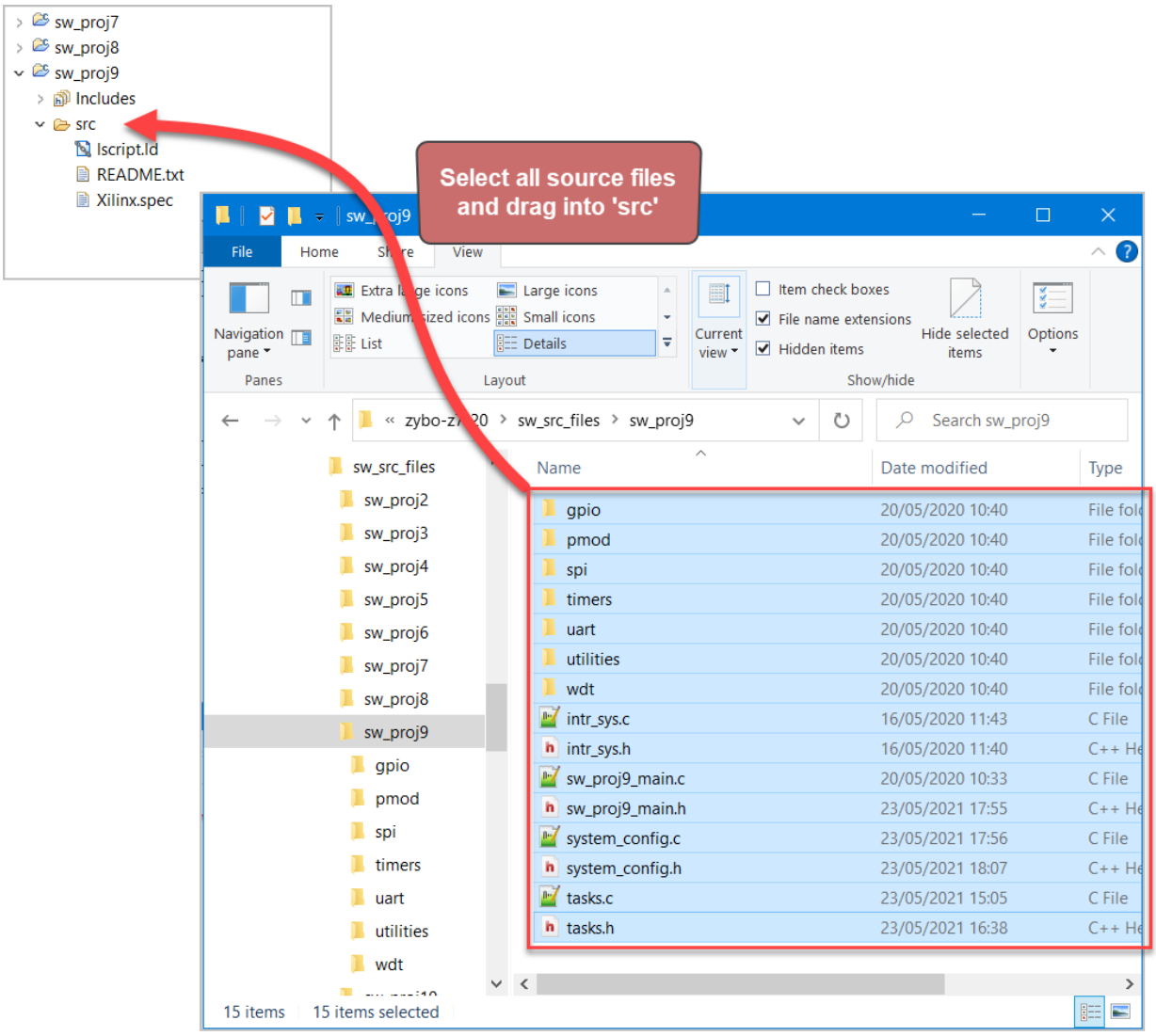


Figure 187. Drag the source files from Windows Explorer into the SDK project

Open the location where the source files for software project 9 are saved on your PC. Drag all the files and directory and from Windows explorer directly into the "**src**" folder in SDK.

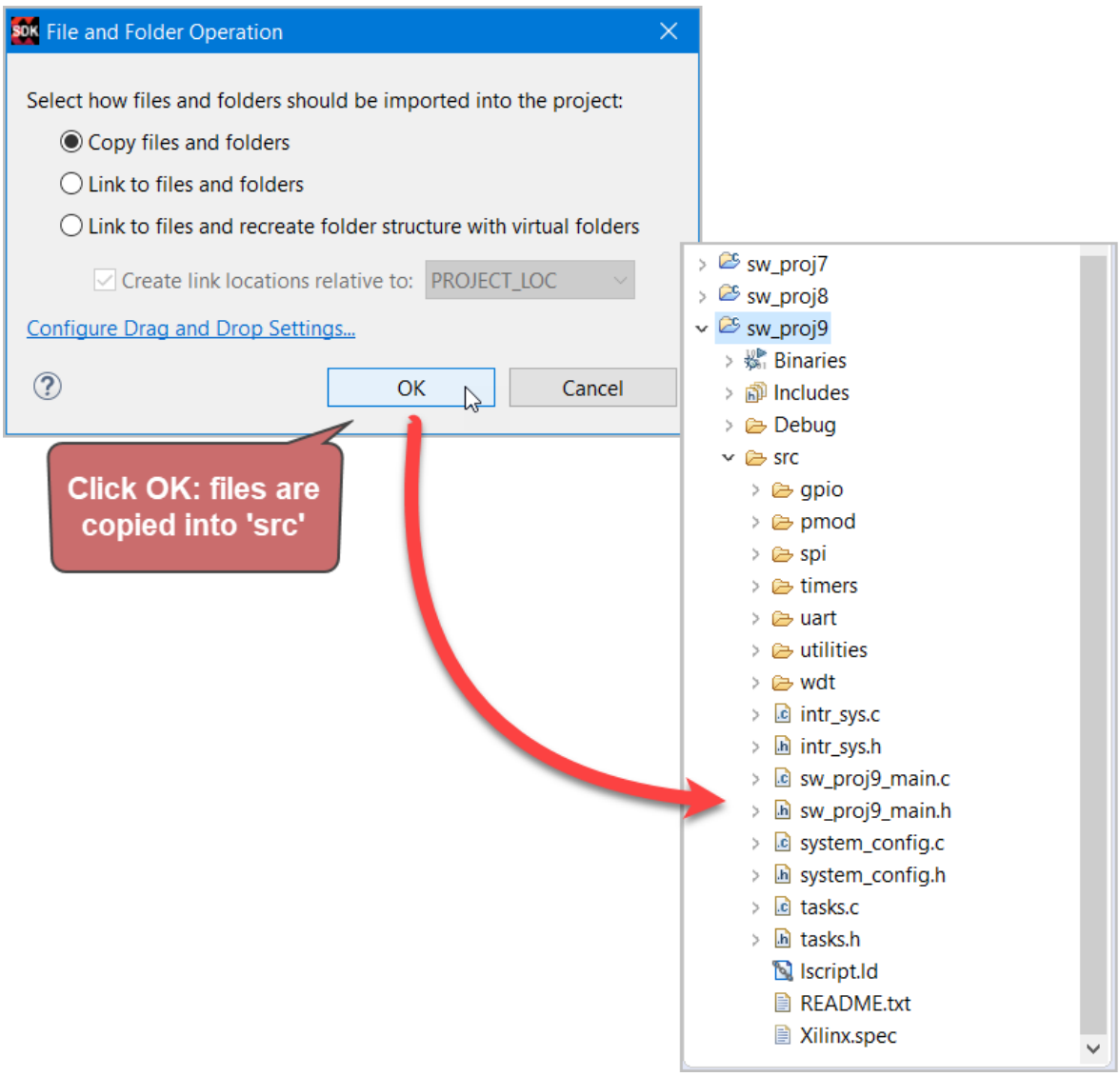


Figure 188. Click OK, and the files will be copied into the project.

In the File Operation dialog box, ensure “Copy files and folders” is checked, and click OK.

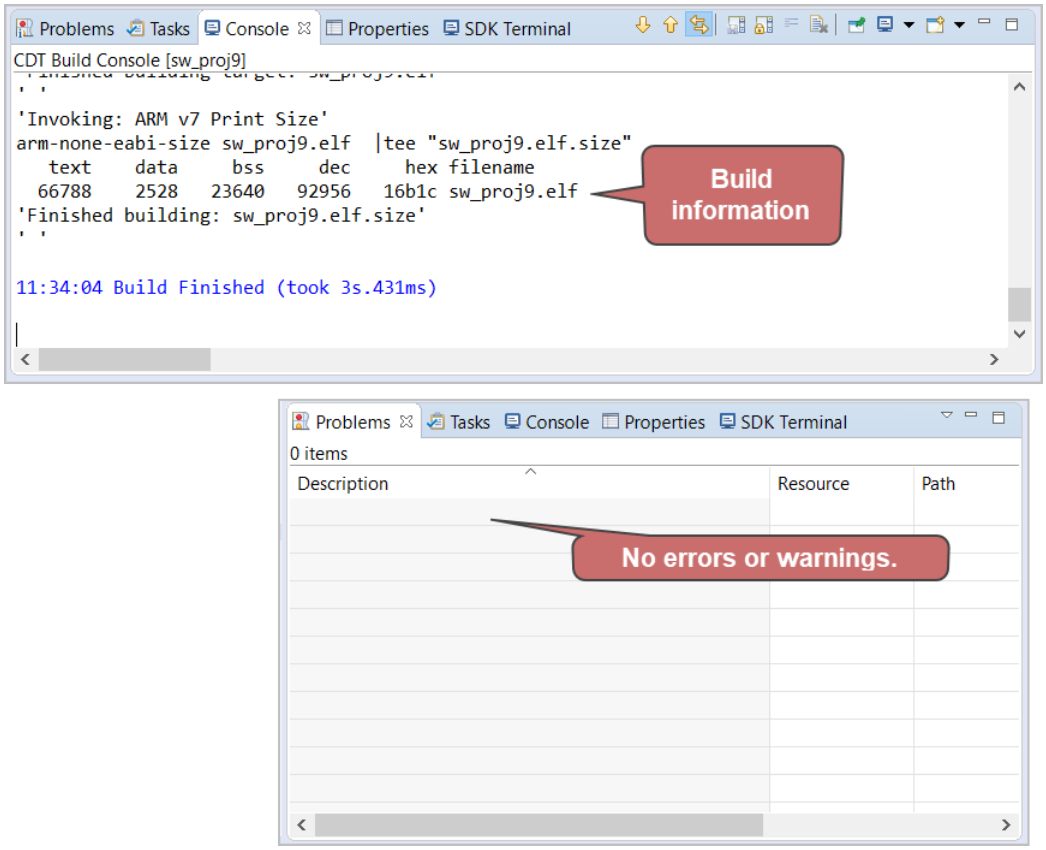


Figure 189. The project should build successfully

If "Build Automatically" is selected, the project should build successfully, with no errors or warnings.

[Exceptions: SDK 2018.3/SDK 2019.1 may indicate a warning as follows, which can be ignored:

#pragma message: For the sleep routines, Global timer is being used]

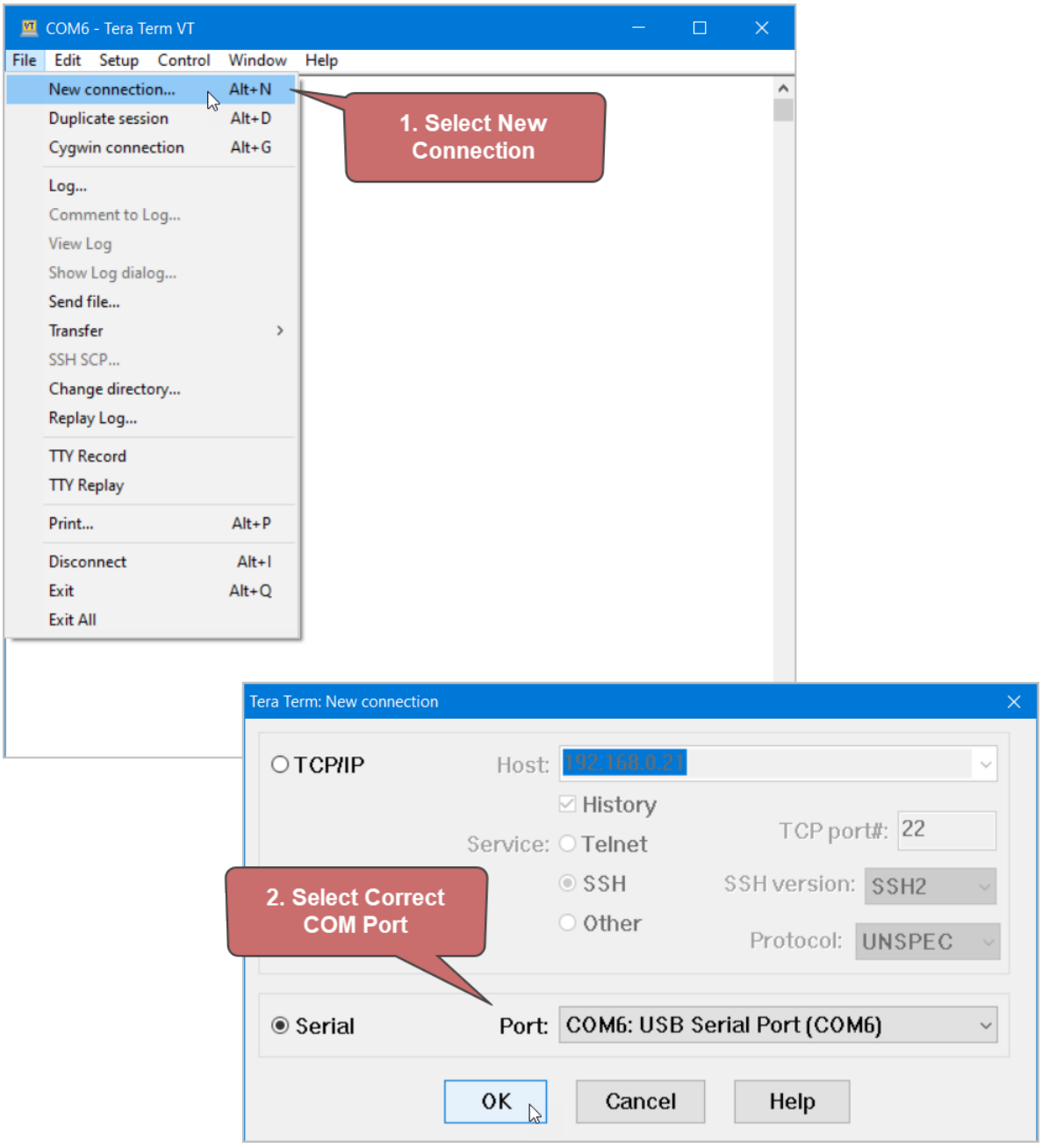


Figure 190. Reconnect terminal to platform if disconnected in last project.

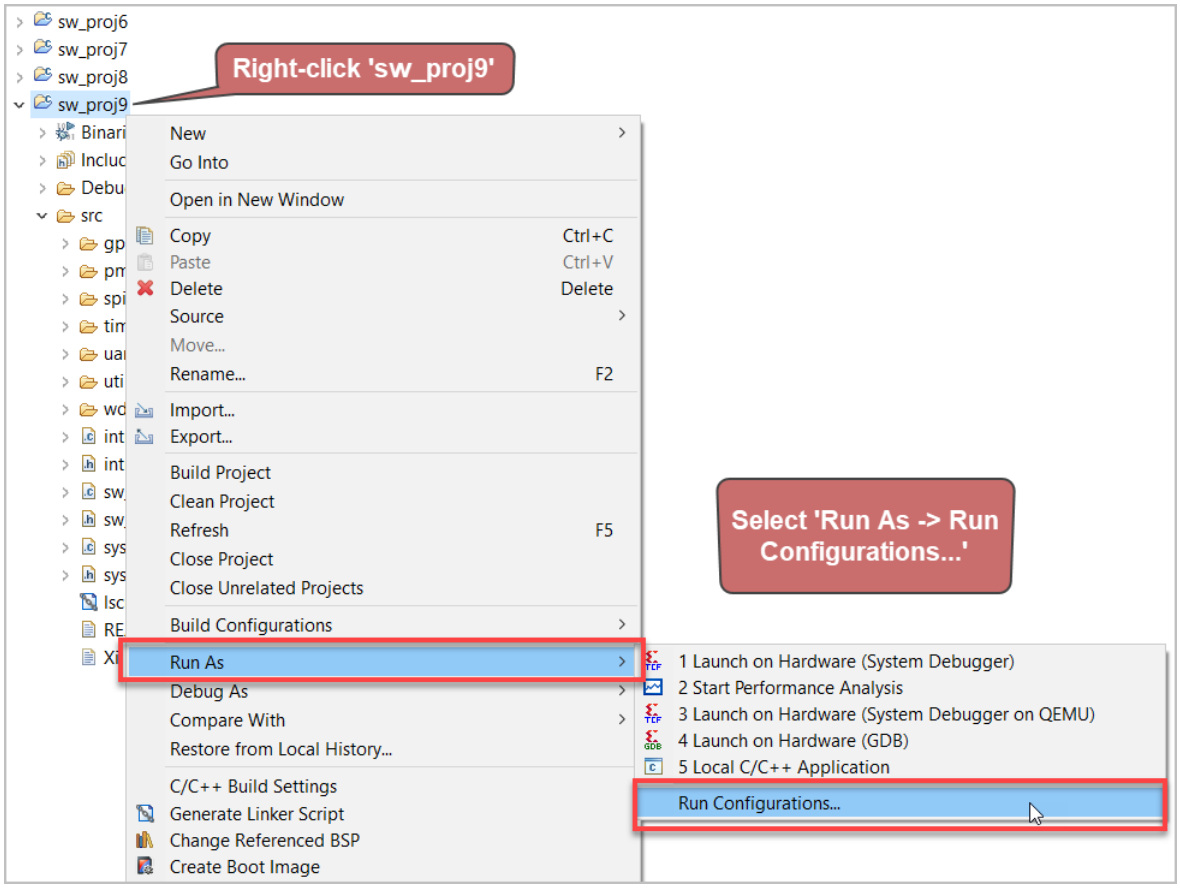


Figure 191. Right-click the software project and create a Run Configuration

To run the program, the first step is to create a Run configuration. Right-click on the project, and select the 'Run Configurations...' option.

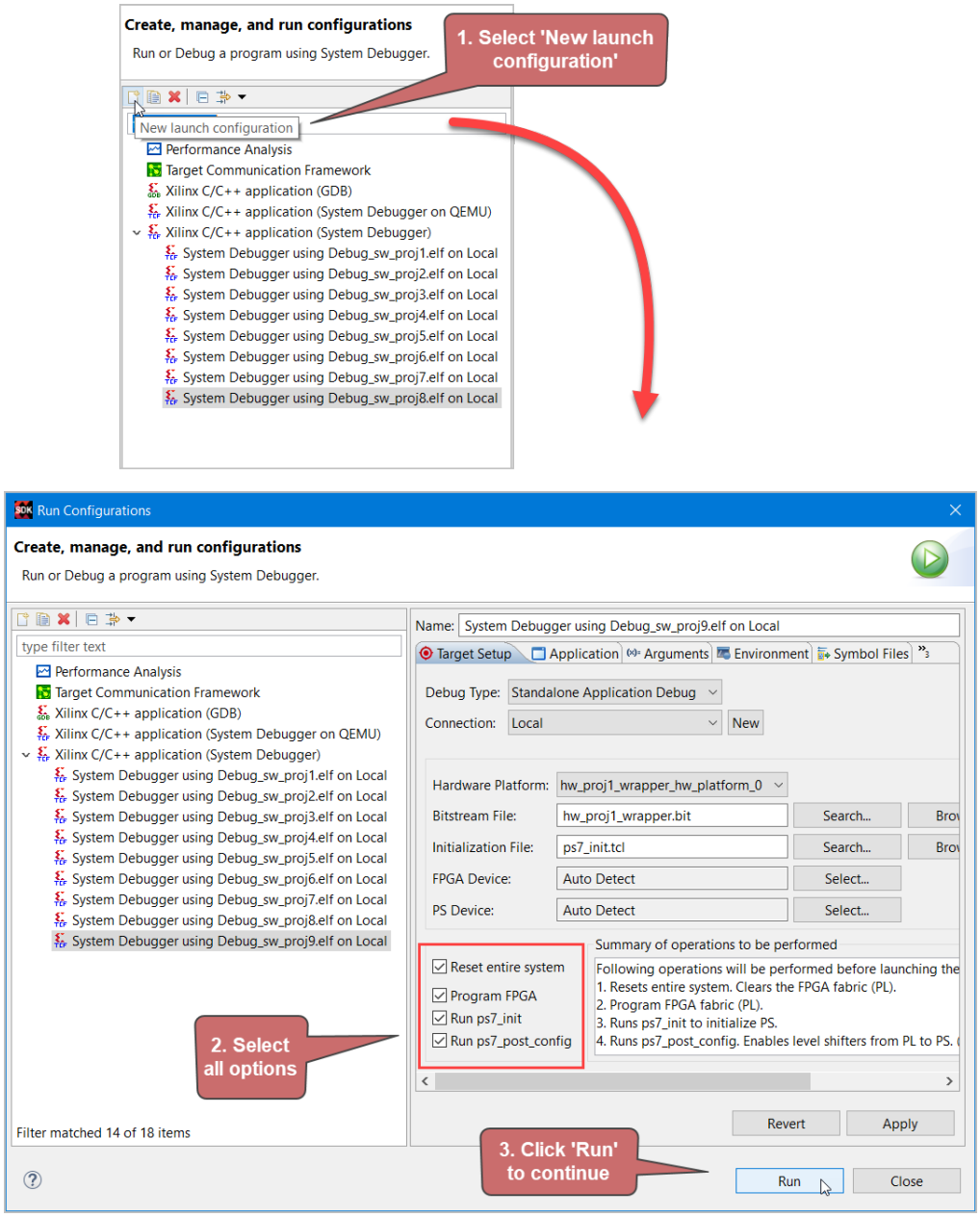


Figure 192. Create a TCF (i.e. System Debugger) option and click Run to execute the program.

1. Ensure the Xilinx C/C++ application (System Debugger) is selected.
2. Click on New Launch Configuration.
3. Select all options:
    a. Reset entire system
    b. Program FPGA
    c. Run ps7_init
    d. Run ps7_post_config

4. Click on 'Run' to continue.

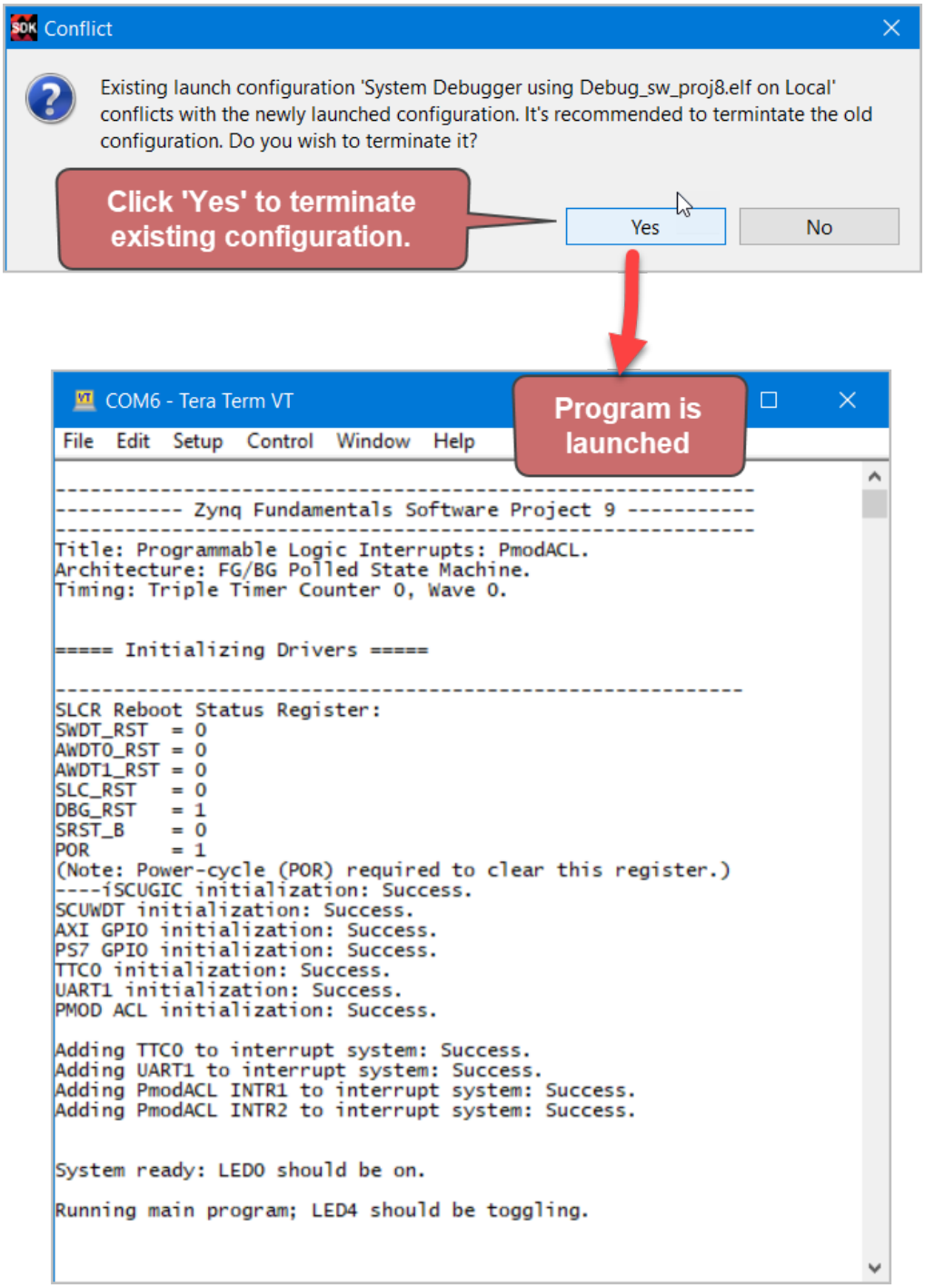


Figure 193. Terminal output for Software Project 9

If prompted, select 'Yes' to terminate any existing configuration. The FPGA will be programmed and the application will be downloaded to the processor. The terminal should display the results for launching Software Project 9.

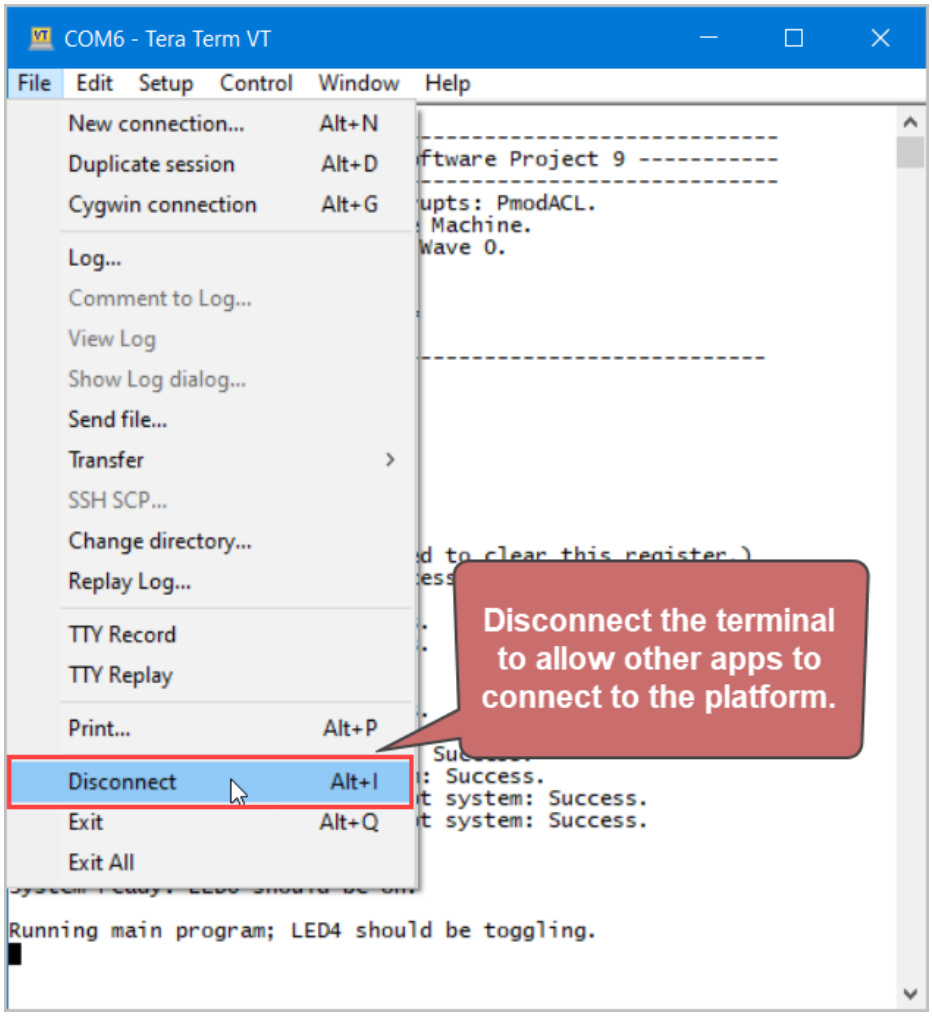


Figure 194. Disconnect the platform.

If the user wants to communicate with the command handler on the platform using a host application, the terminal needs to be disconnected.

3.10 Software Project 10: Additional Interrupt Topics

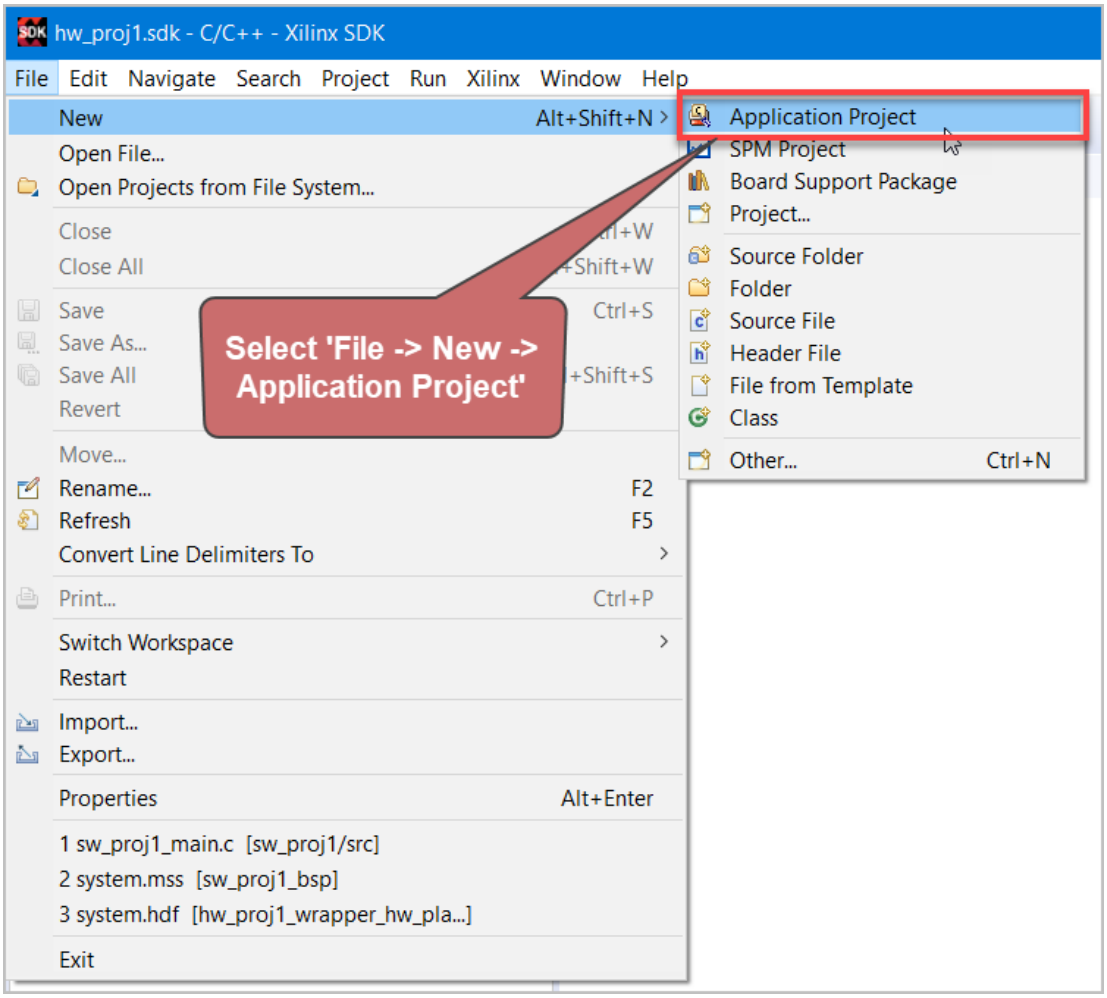


Figure 195. Select File->New-> Application Project

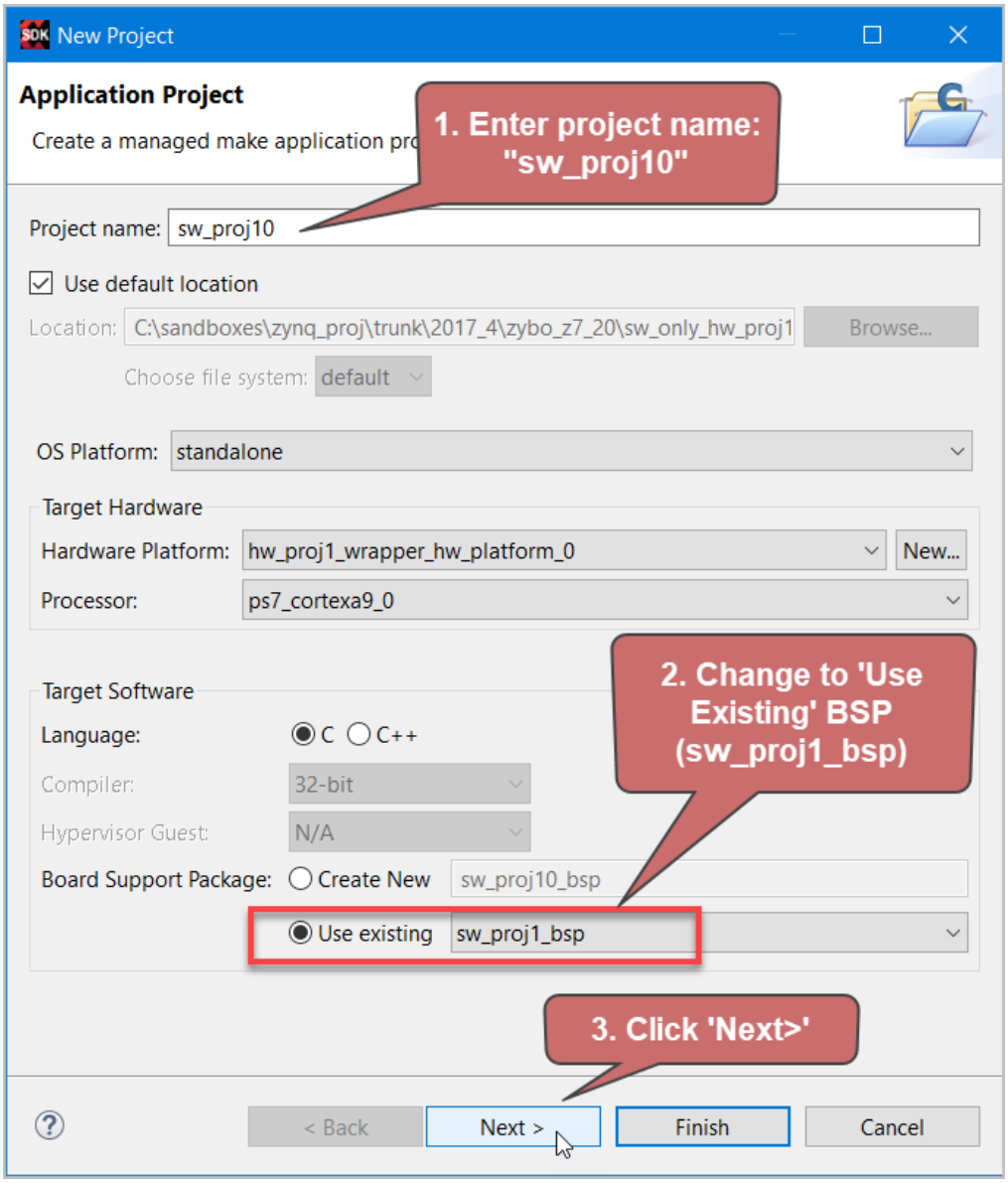


Figure 196. Enter the project details (part 1)

1. Enter the project name: sw_proj10
2. Board Support Package: Use existing
3. Click Next (don't click Finish!)

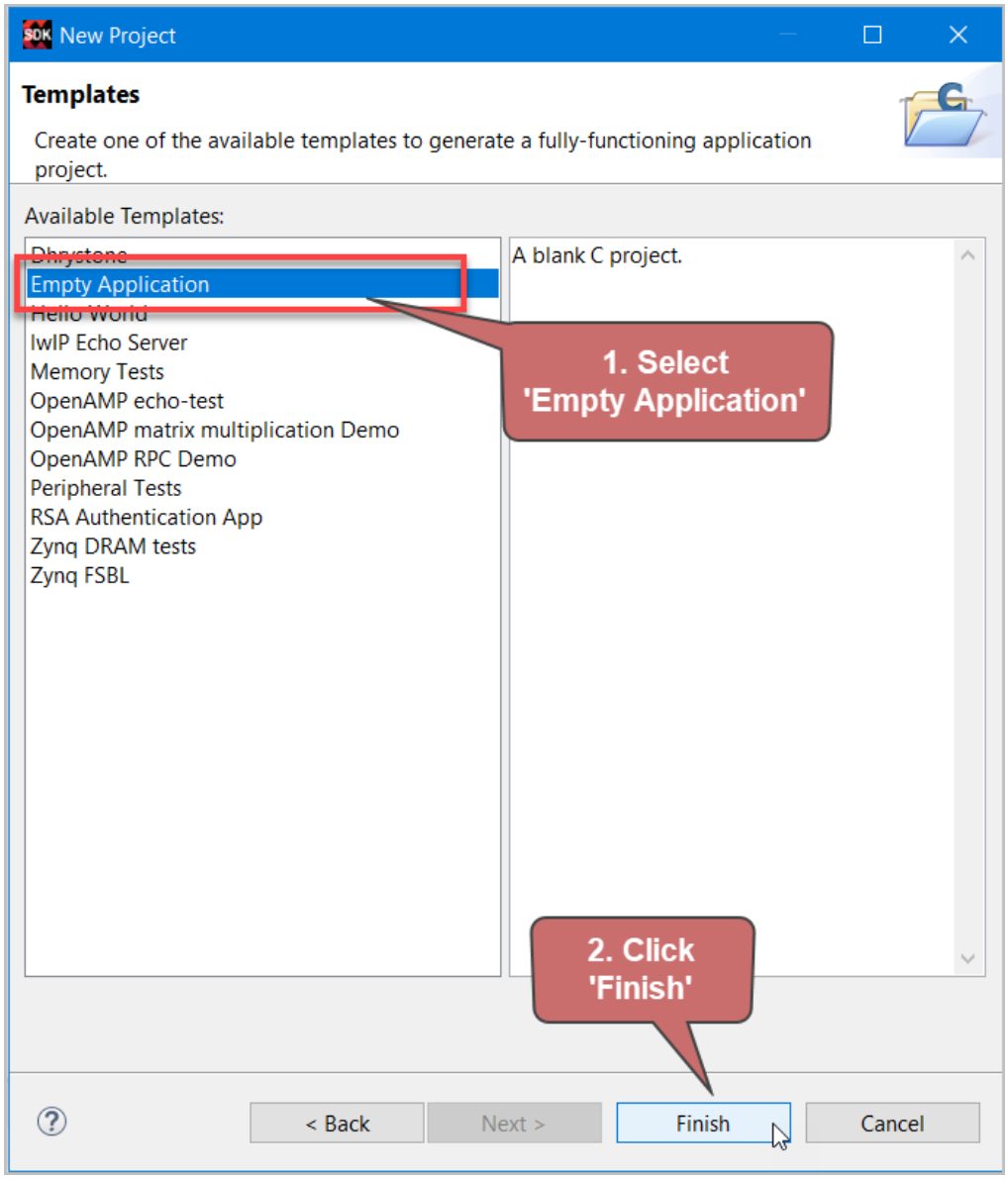


Figure 197. Enter the project details (part 2)

1. Select Empty Application
2. Click Finish

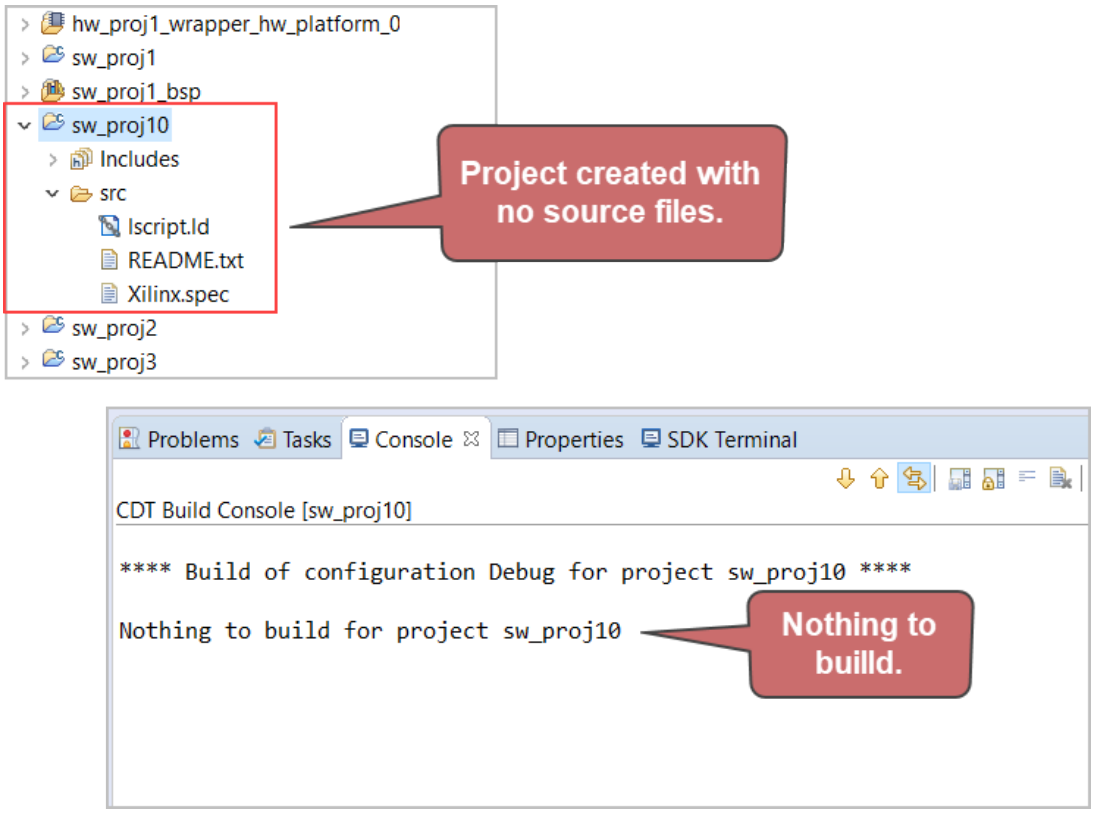


Figure 199. Project created with no source files

A project named "sw_proj10" is created with no project files. The CDT Build Console confirms that there is nothing to build.

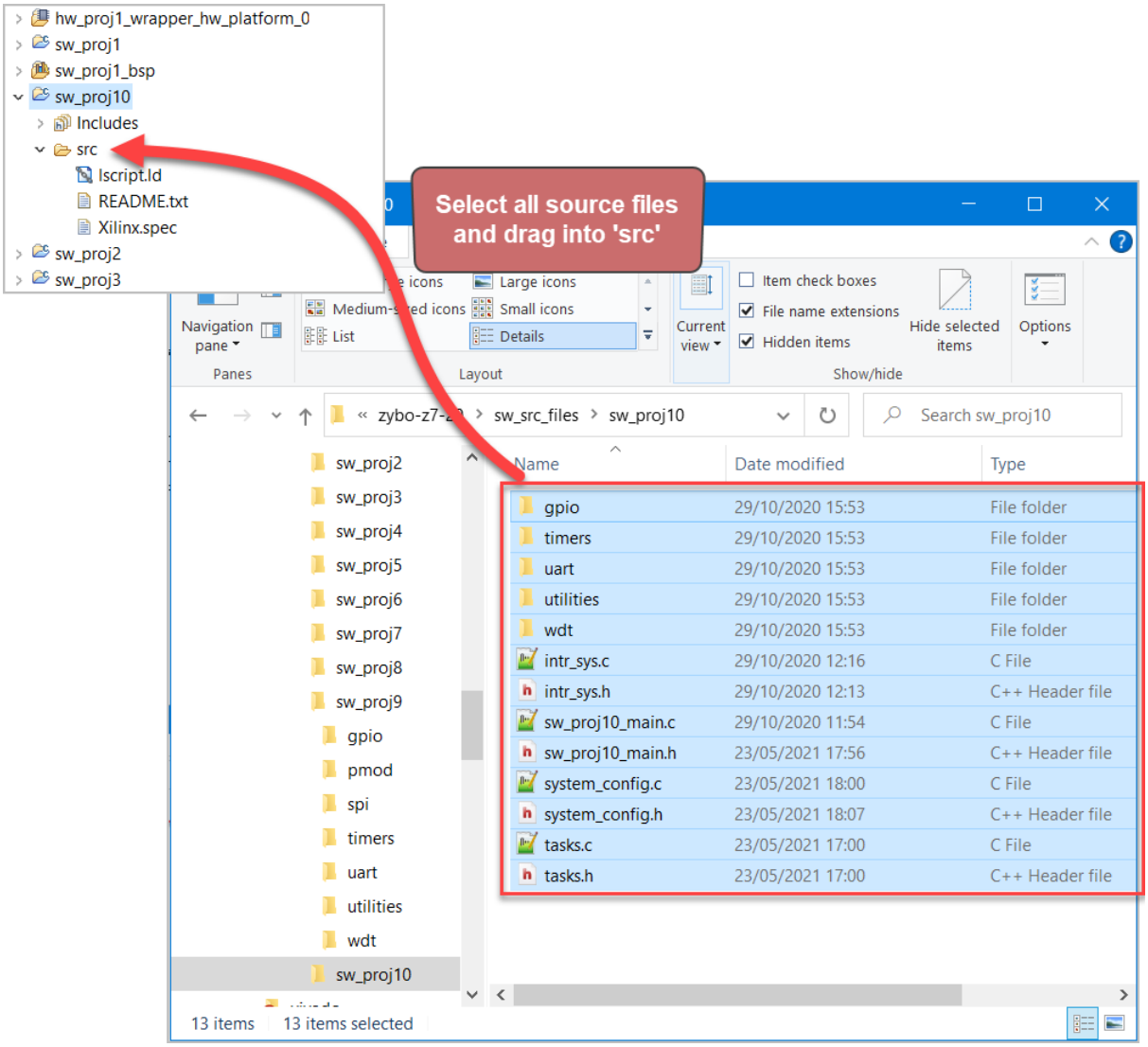


Figure 200. Drag the source files from Windows Explorer into the SDK project

Open the location where the source files for software project 10 are saved on your PC. Drag all the files and directory and from Windows explorer directly into the "**src**" folder in SDK.

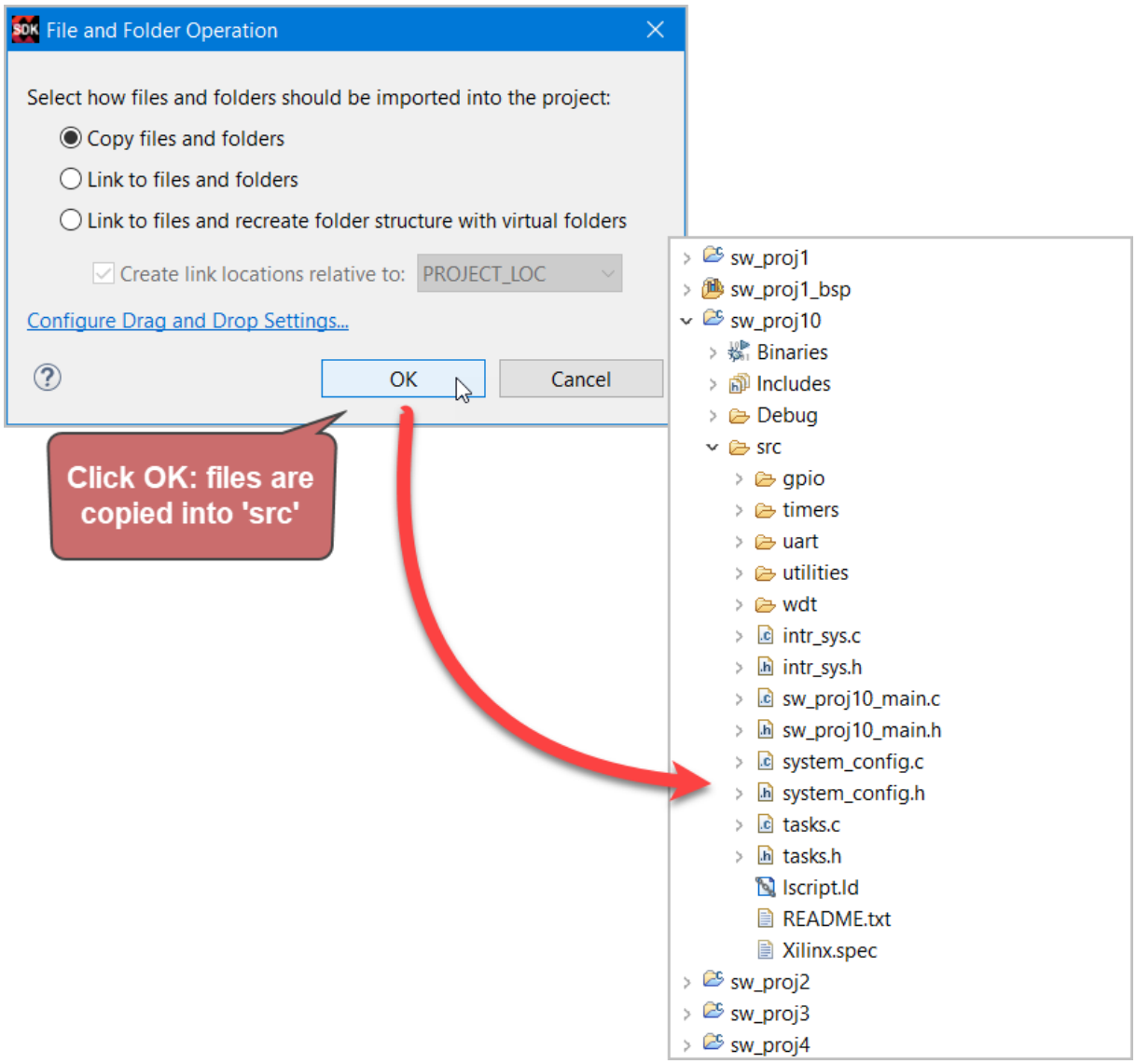


Figure 201. Click OK, and the files will be copied into the project.

In the File Operation dialog box, ensure “Copy files and folders” is checked, and click OK.

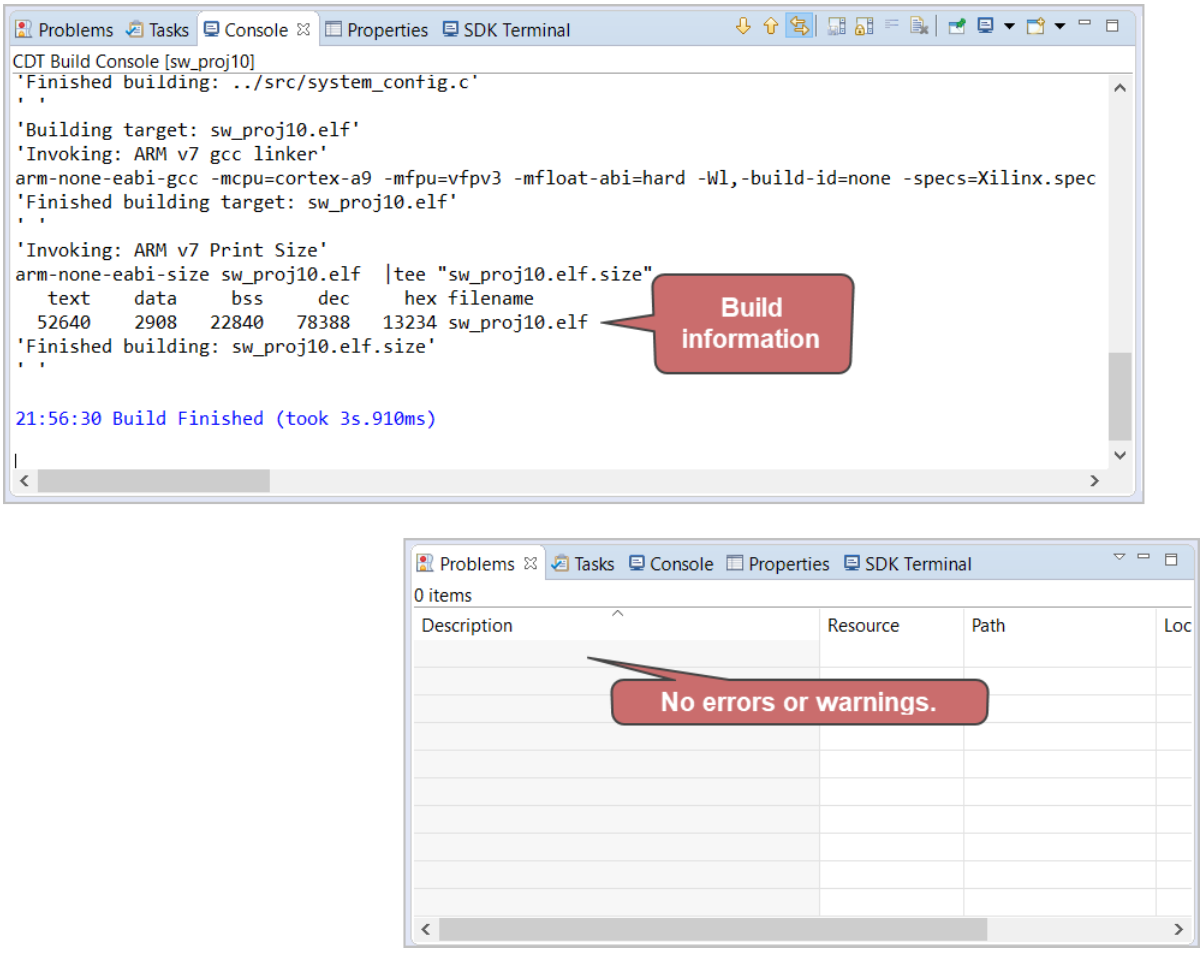


Figure 202. The project should build successfully

If "Build Automatically" is selected, the project should build successfully, with no errors or warnings.

[Exceptions: SDK 2018.3/SDK 2019.1 may indicate a warning as follows, which can be ignored:

#pragma message: For the sleep routines, Global timer is being used]

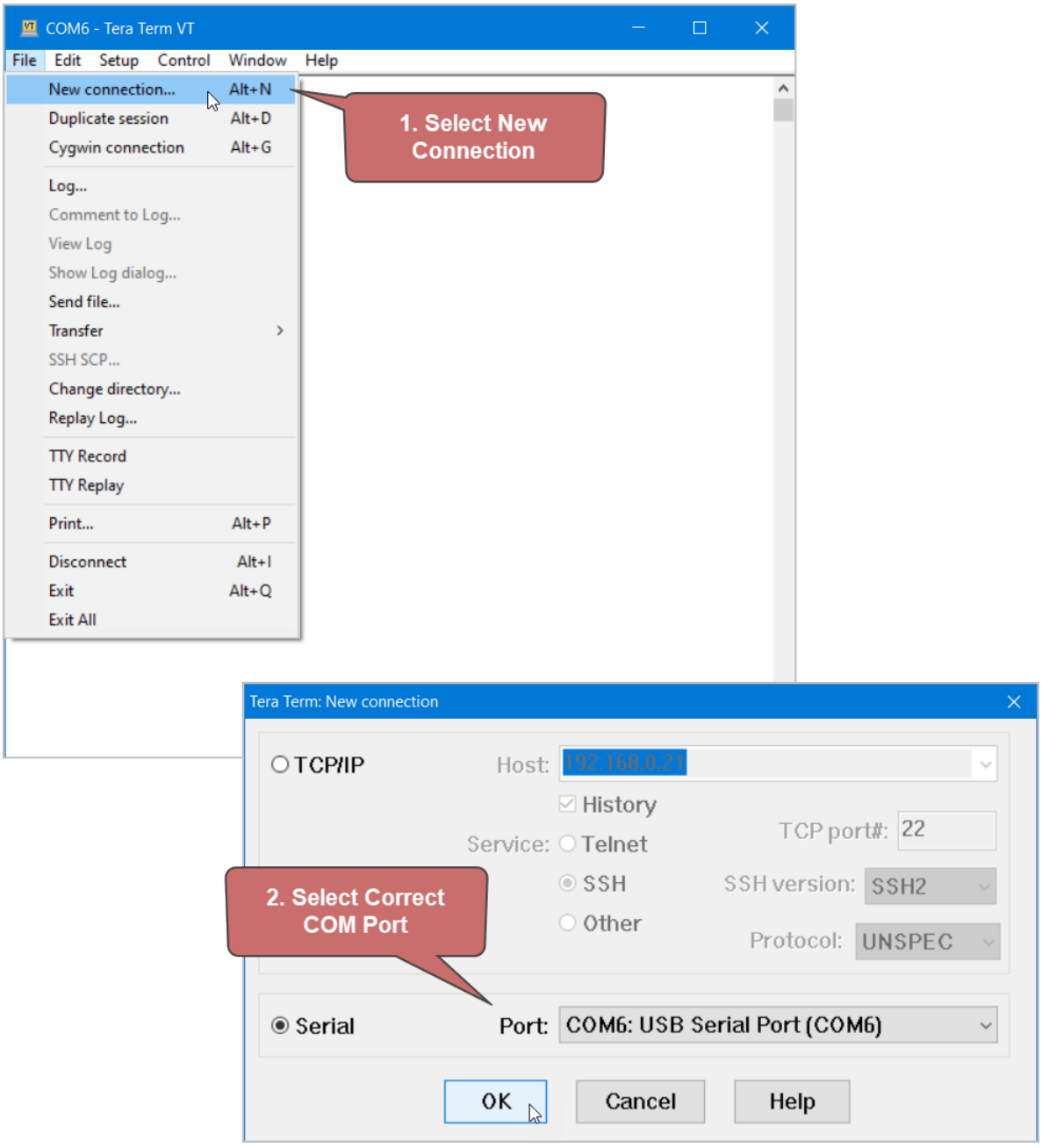


Figure 203. Reconnect terminal to platform if disconnected in last project.

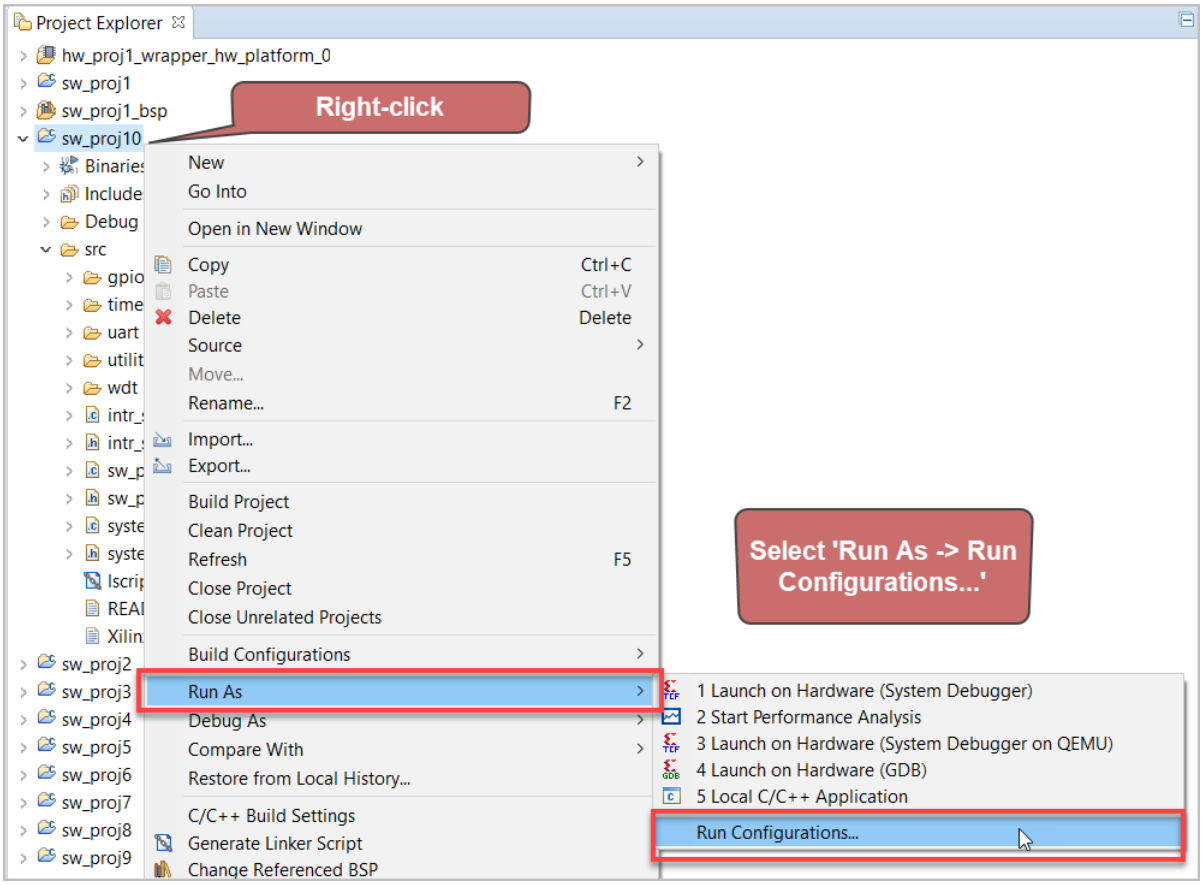


Figure 204. Right-click the software project and create a Run Configuration

To run the program, the first step is to create a Run configuration. Right-click on the project, and select the 'Run Configurations...' option.

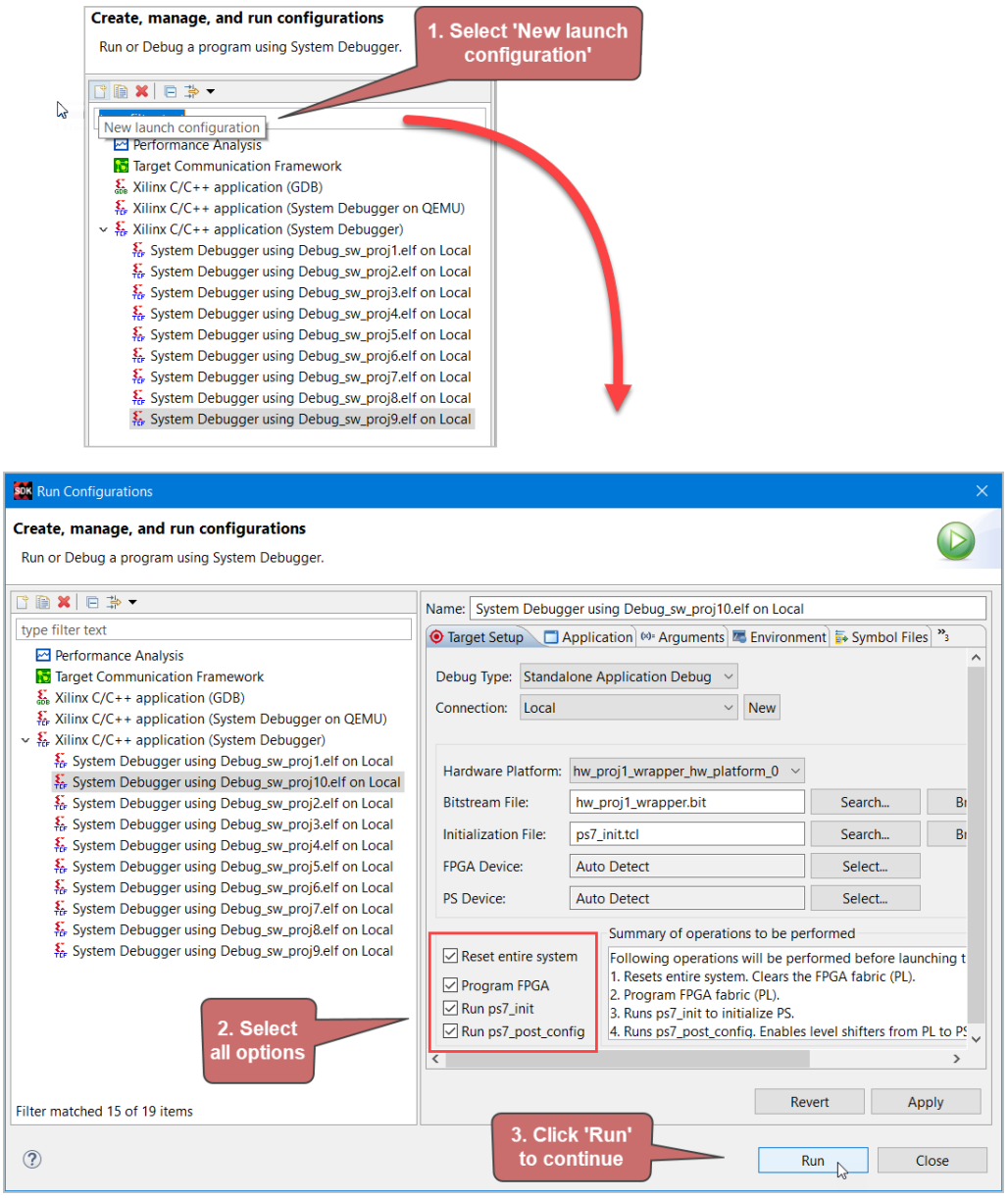


Figure 205. Create a TCF (i.e. System Debugger) option and click Run to execute the program.

1. Ensure the Xilinx C/C++ application (System Debugger) is selected.
2. Click on New Launch Configuration.
3. Select all options:
    a. Reset entire system
    b. Program FPGA
    c. Run ps7_init
    d. Run ps7_post_config

4. Click on 'Run' to continue.

If the terminal was disconnected in the last project (to run the LabVIEW/Python applications) then the Tera Term connection should be re-opened.

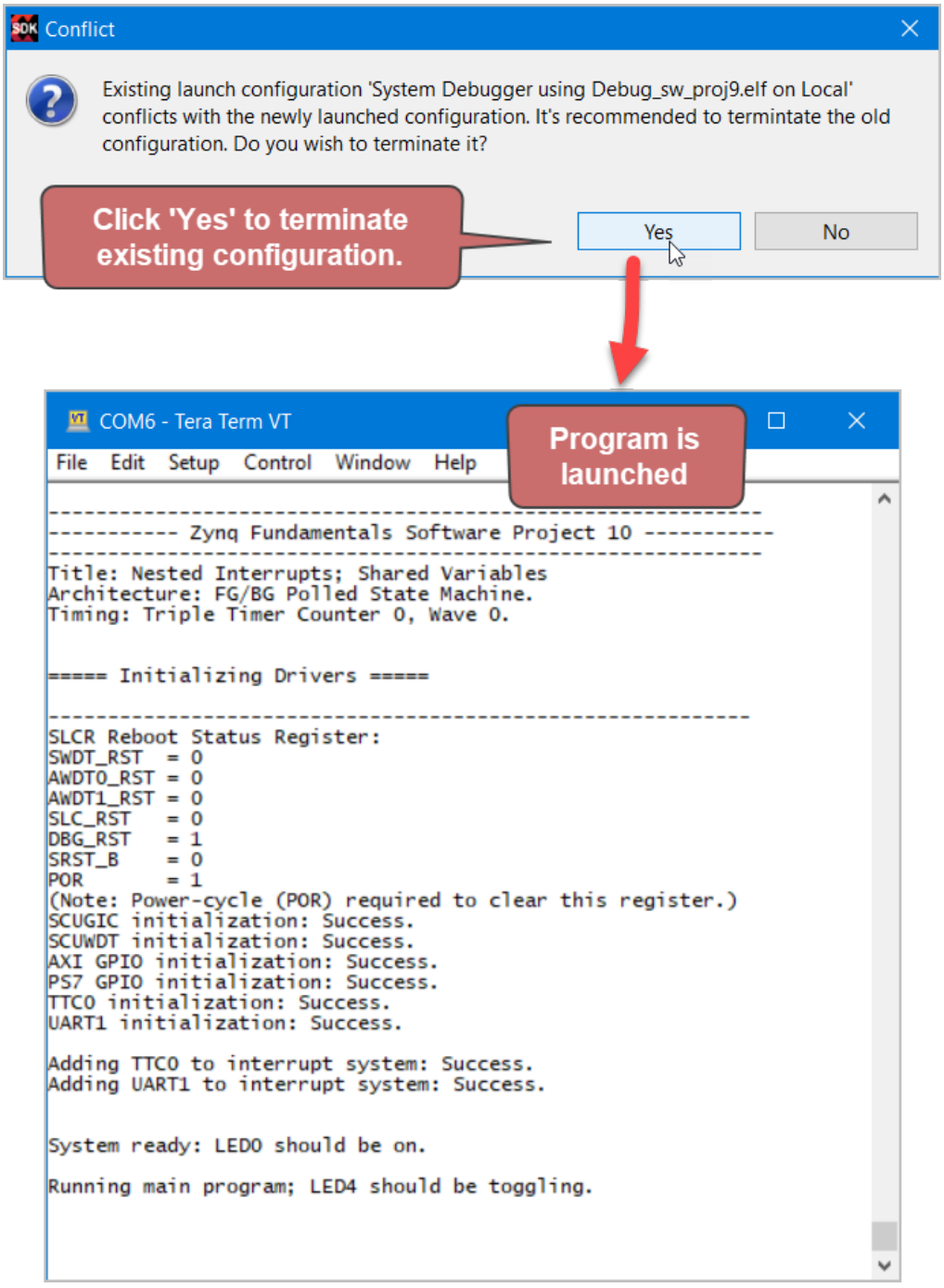


Figure 206. Terminal output for Software Project 10

If prompted, select 'Yes' to terminate any existing configuration. The FPGA will be programmed and the application will be downloaded to the processor. The terminal program should display the results for launching Software Project 10.

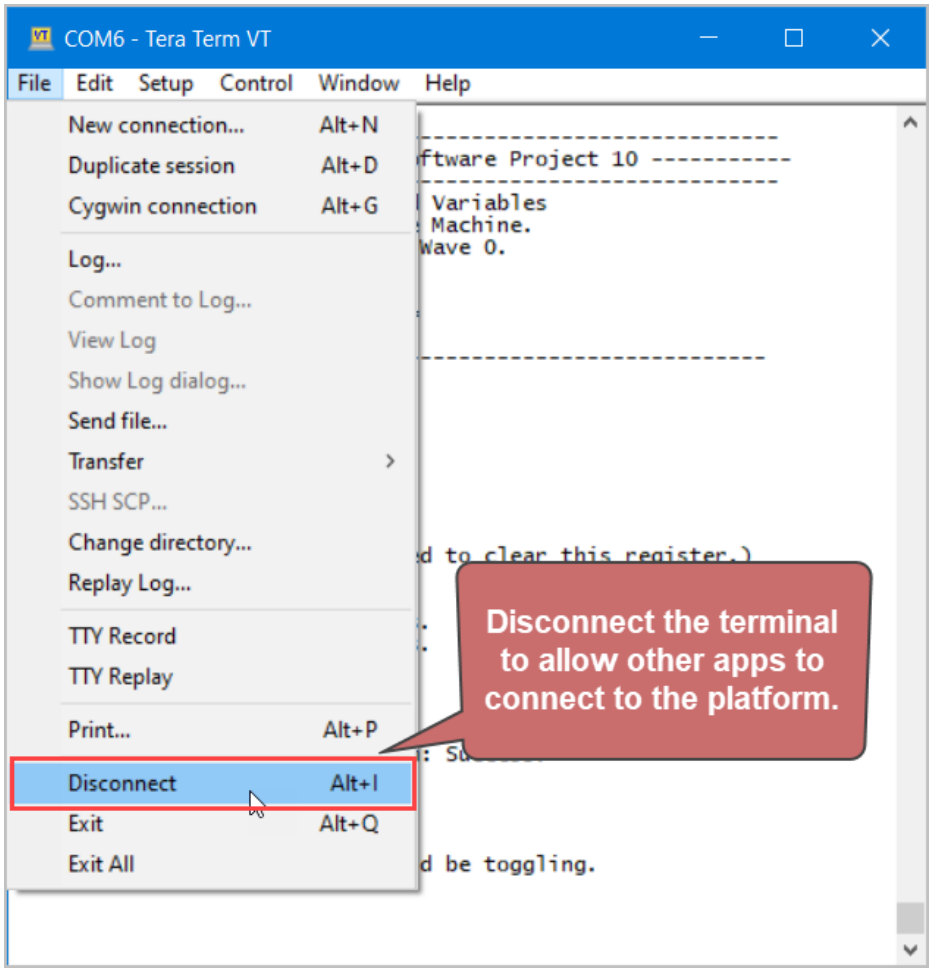


Figure 207. Disconnect the platform.

If the user wants to communicate with the command handler on the platform using a host application, the terminal needs to be disconnected.