Embedded Studio IDE QuickStart

For Nuclei Processor Core

Release 1.1.0

Copyright Notice

The product described herein is subject to continuous development and improvement; information herein is given by Nuclei in good faith but without warranties.

This document is intended only to assist the reader in the use of the product. Nuclei System Technology shall not be liable for any loss or damage arising from the use of any information in this document, or any incorrect use of the product.

Contact Information

Should you have any problems with the information contained herein or any suggestions, please contact Nuclei System Technology by email support@nucleisys.com, or visit “Nuclei User Center” website http://user.nucleisys.com for supports or online discussion.

Revision History

Rev.	Revision Date	Revised Section	Revised Content
1.0.0	2020/02/15	N/A	1. First version as the full English
1.1.0	2020/08/04	ALL	1. Note that path for software and project must be in English 2. Add compile options instructions 3. Modify the compilation option to macro when creating a new project 4. When using j-link, directly modify the debug mode, and do not use GDB connection 5. Update some old figures

Table of Contents

List of Figures

1. Overview

This document will use Embedded Studio as IDE to develop embedded software on the FPGA evaluation board (called Hummingbird Evaluation Kit) with Debugger hardware (called Hummingbird Debugger Kit).

1.1. Introduction of Embedded Studio

SEGGER Embedded Studio (SES for short) developed by SEGGER Company, is a well-known Integrated Development Environment (IDE) for embedded software development.

SES has professional source code development and compilation user interface, powerful debugging features (equipped with famous J-Link).

SES is free for non-commercial usage, with stable cross platform compatibility and flexible configurations.

1.2. Installation of Embedded Studio and Setup for Nuclei

For the detailed installation steps of SES and how to set up the tool chain for Nuclei, please refer to the “Download” page of Nuclei website (http://www.nucleus.com/download.php) to download <Nuclei_SES_IDE_Installation.pdf>.

Note: Make sure the software installation path must be in English.

1.3. Introduction of Hummingbird Evaluation Kit

Nuclei have customized a FPGA evaluation board, called Hummingbird Evaluation Kit as shown in Figure 1-1. This FPGA board can be used as the SoC prototype board directly:

- If the FPGA have been pre-burned (programmed) with “Nuclei evaluation SoC”, this board can be worked as a SoC prototype directly. Since the board has been designed with buttons and extended ports names in line with the SoC GPIO pin name, the

embedded software engineers can directly use this board without knowing any FPGA hardware knowledge.

- To easy the writing, the “Nuclei evaluation SoC” will be shorted as “the SoC” in this document thereby.
- For the detailed introduction of the Hummingbird Evaluation Kit, please refer to the “Development Boards” page of Nuclei website (http://www.nucleisys.com/developboard.php) to download <Nuclei_FPGA_DebugKit_Intro.pdf>.

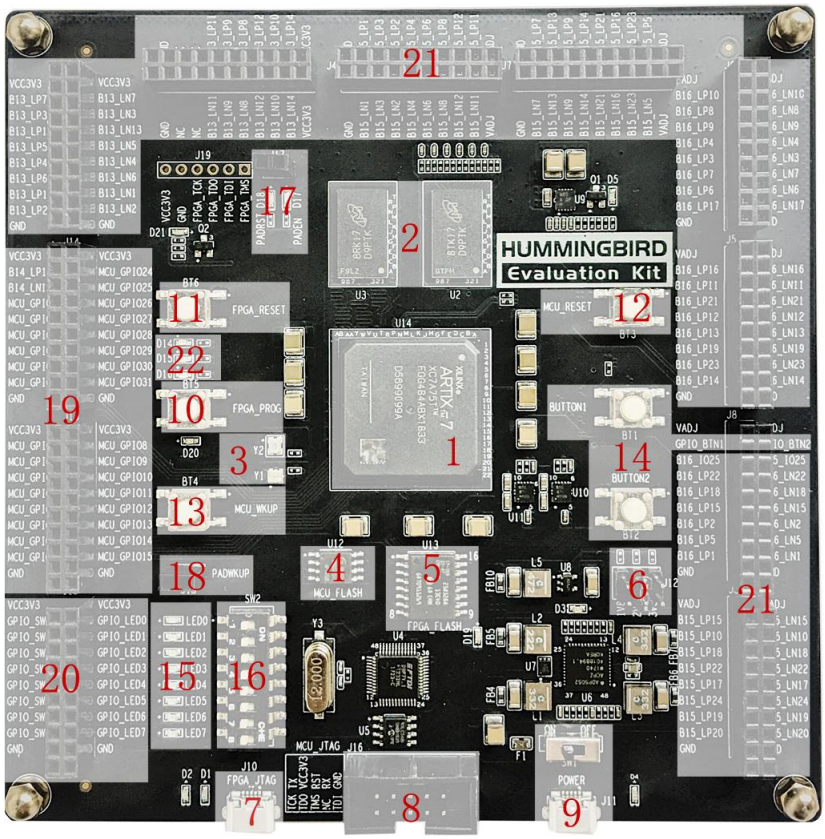

Figure 1-1 Hummingbird Evaluation Kit

1.4. Introduction of Hummingbird Debugger Kit

Nuclei have customized a Debugger hardware (called Hummingbird Debugger Kit), as shown in

Figure 1-2, which can be used to debug the RISC-V core in FPGA prototype or in real chip.

- For the detailed introduction of the Hummingbird Debugger Kit, please refer to the "Development Boards" page of Nuclei website (http://www.nucleisys.com/developboard.php) to download <Nuclei_FPGA_DebugKit_Intro.pdf>.

Figure 1-2 Hummingbird Debugger Kit

2. Create Project

2.1. Overview

There are two ways to create a new project in SES:

- Import project directly from existing project:
  - This is the most common way to create a new project. For example, user A can directly package an existing project; thereafter it can be easily shared and spread. User B can simply import the project on another computer, so as to use it, develop it or create a new project based on it.
- Create project manually without template:
  - This is the most tedious way to create a new project. It requires manual setting of options and paths. Because this way is inconvenient, it is seldom used in practical works. However, by explanations of this way, users can learn how to set various options and paths in details.

These two ways are described at follows.

2.2. Import Project Directly from Existing Project

This section will introduce how to use SES to create a new project by directly importing from an existing project.

This document takes demo-eclic as an example. The project package can be downloaded from Github (https://github.com/riscv-mcu/ses_nuclei_sdk_projects) as depicted in Figure 2-1. Note: this project is for demonstration, for more other examples you can contact Nuclei.

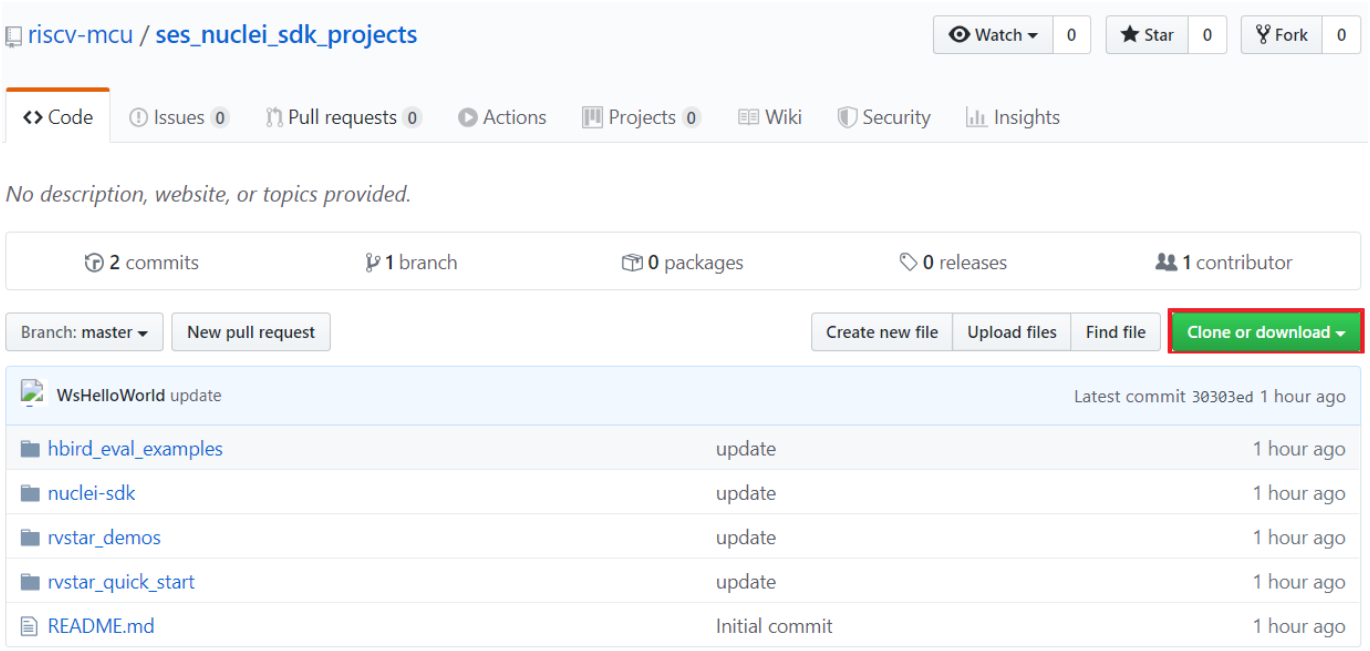

Figure 2-1 Download Project Package on Github

After decompressing the package, the contents are shown in Figure 2-2.

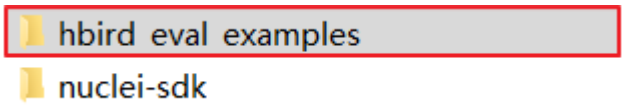

Figure 2-2 The Contents of Package

Open the hbird_eval_examples folder, the contents are as shown in Figure 2-3. Please double click the file which marked in red-box to open SES and import project. If the operation is correct, the interface in Figure 2-4 will appears. The file structure of the project is shown in the red-box.

Note: Make sure the project path must be in English.

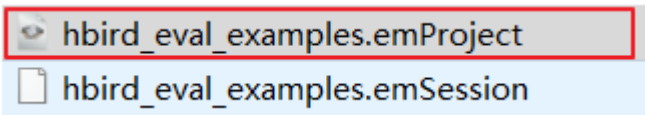

Figure 2-3 The Contents of the hbird_eval_examples Folder

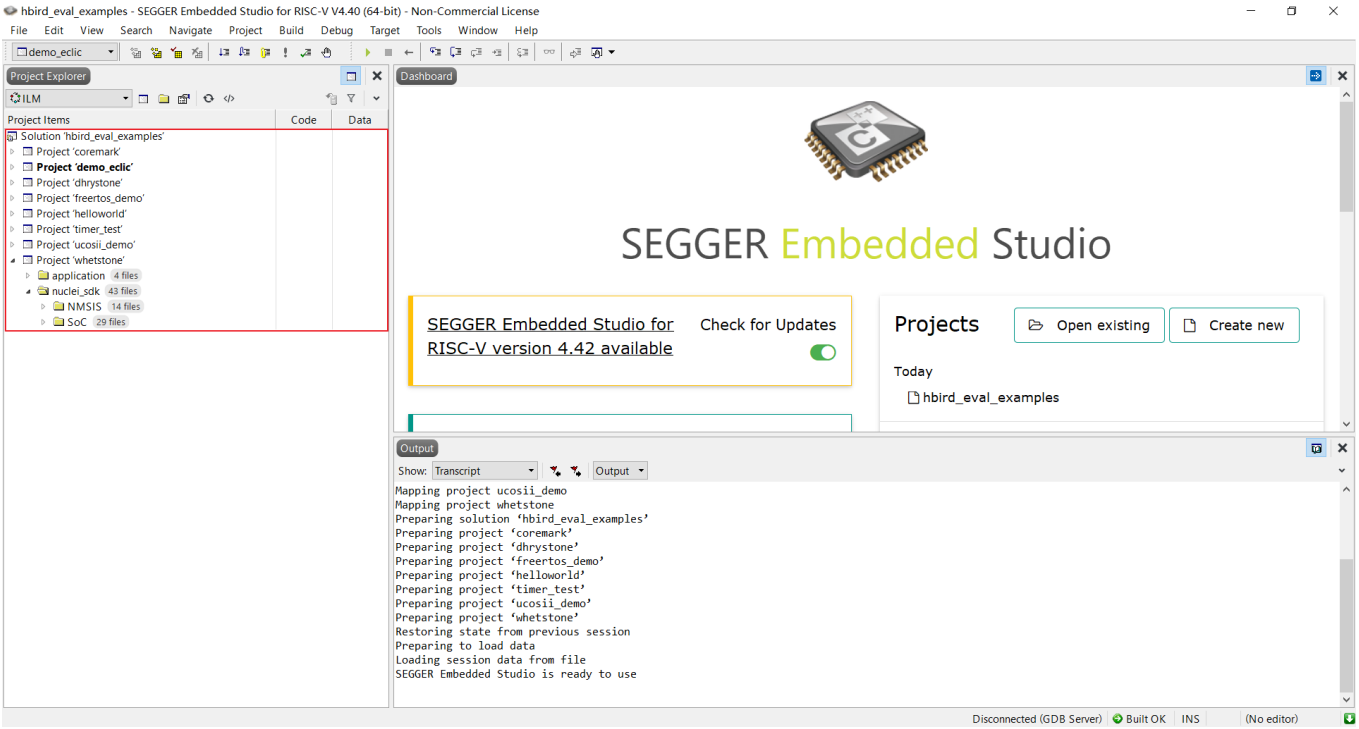

Figure 2-4 File Structure of the Existing Project

The Nuclei Evaluation SoC supports four “Compiling and Downloading” modes, including:

- ILM (Compile to run program from ILM);
- FLASH (Compile to run program from FLASH);
- FLASHXIP (Compile to upload program from FLASH and run from ILM).
- DDR(Compile to run program from DDR)

Please refer to document <Nuclei_Eval_SoC_Intro.pdf> for more information of the Nuclei Evaluation SoC.

The imported project is already set up in advance. As shown in Figure 2-5, the red-box indicates the location where user can switch the “Compiling and Downloading” modes.

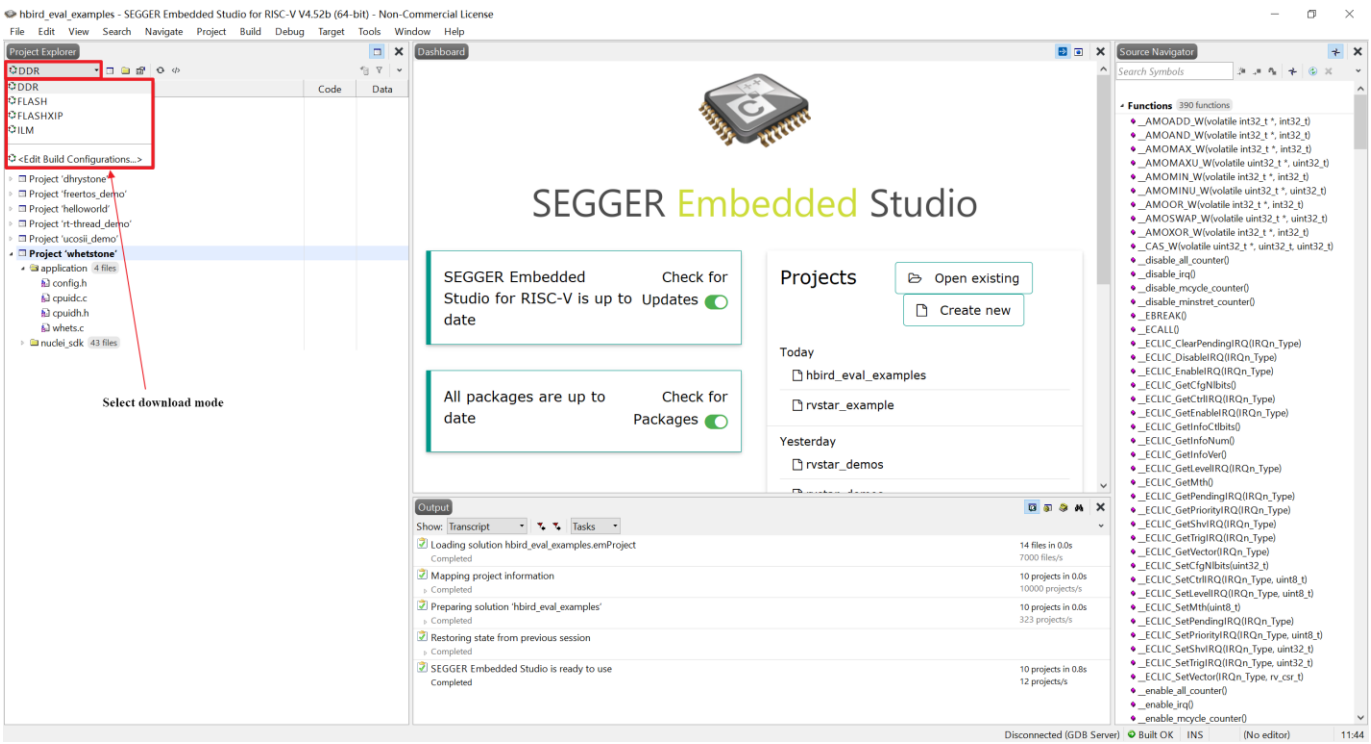

Figure 2-5 Switch the Compiling and Downloading Modes

2.3. Manually Create Project without Template

This section describes how to manually create a project in SES.

As described in Section 2.1, this method is very tedious, which needs to manually set various options and paths. So, it is rarely used in practical work.

This document explains this method in details only to help users understand how to set up options and paths. For the user just want to quick start, this section can be skipped.

The detailed steps are as follows.

2.3.1. Manually Create Project

- Select "File --> New Project" in the menu bar, as shown in Figure 2-6.

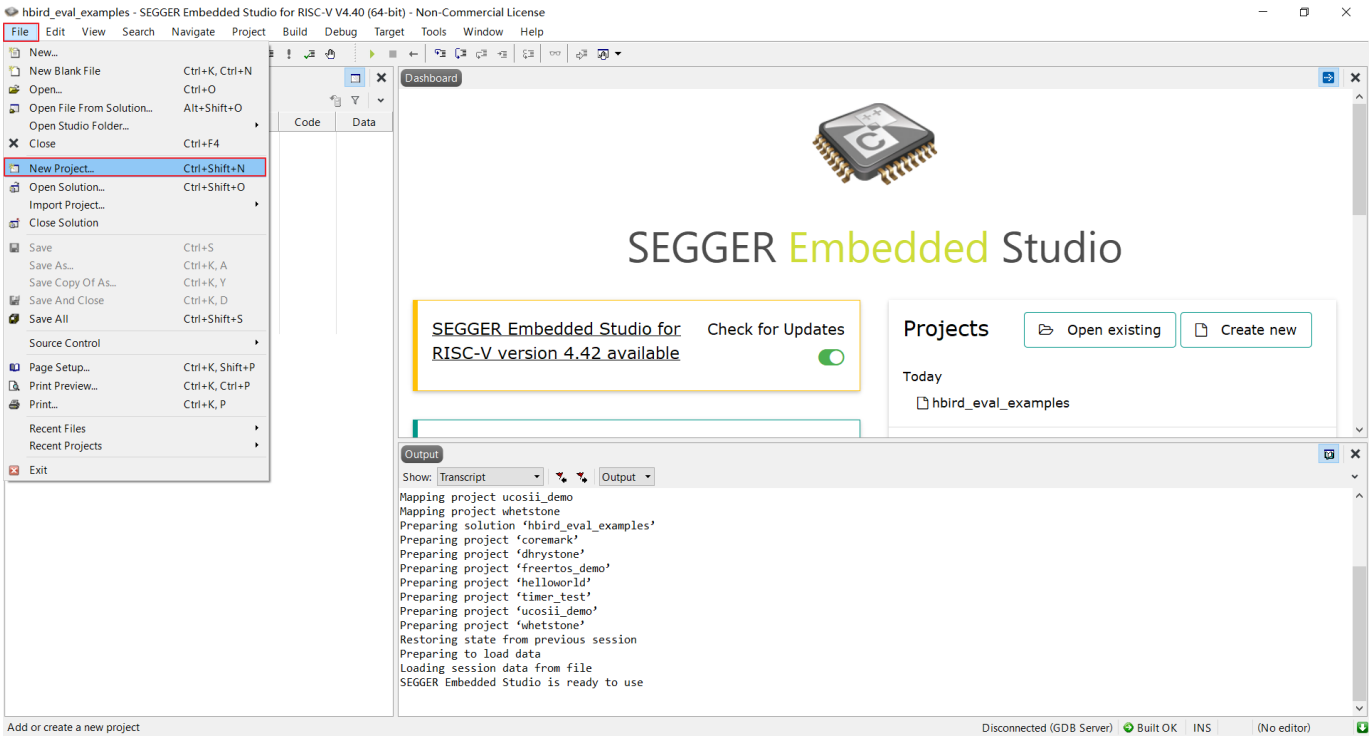

Figure 2-6 Create New Project

- Click "New Project" to pop up the window as shown in Figure 2-7, select the first option, enter the project name and select the folder location, then click next.

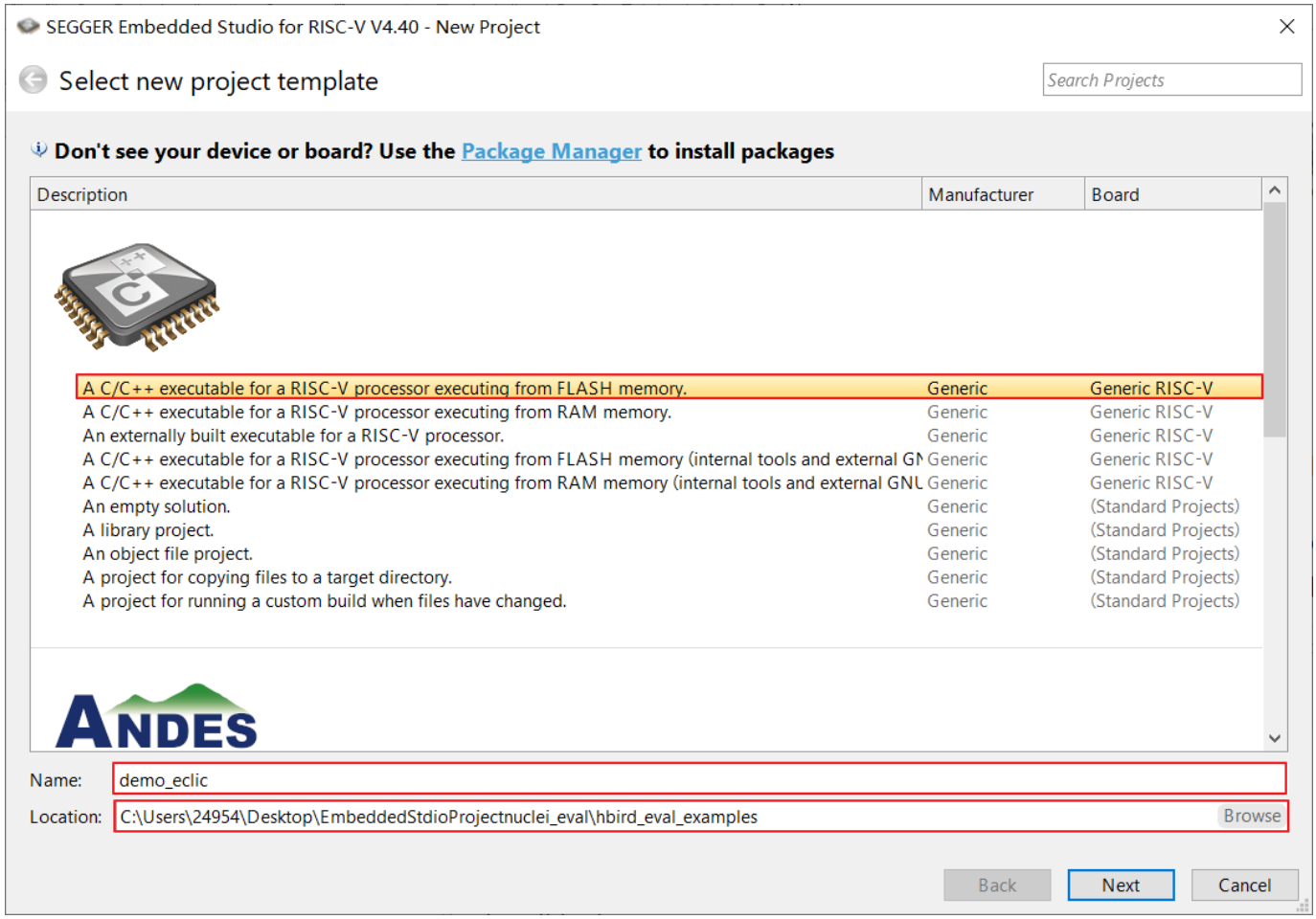

Figure 2-7 Project Name and Location

- Select n307 (as an example) as the target device, as shown in Figure 2-8, then click next.

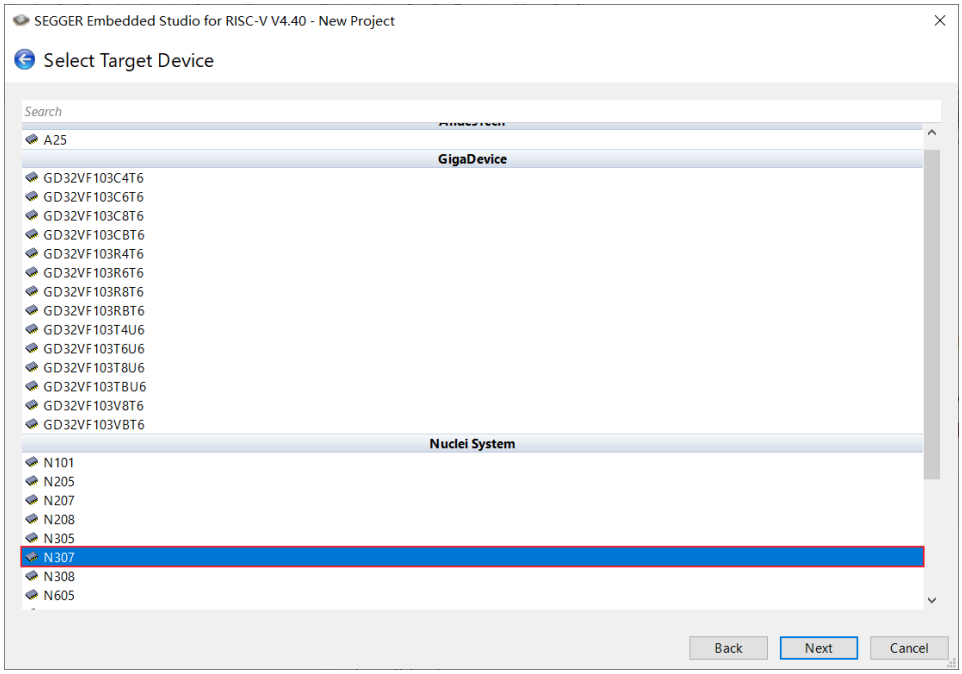

Figure 2-8 Choose Target Device

- Set to add default files, as shown in Figure 2-9. In this step, uncheck all files, and then click next.
- The new page does not need to be modified. Click “Finish” to finish.

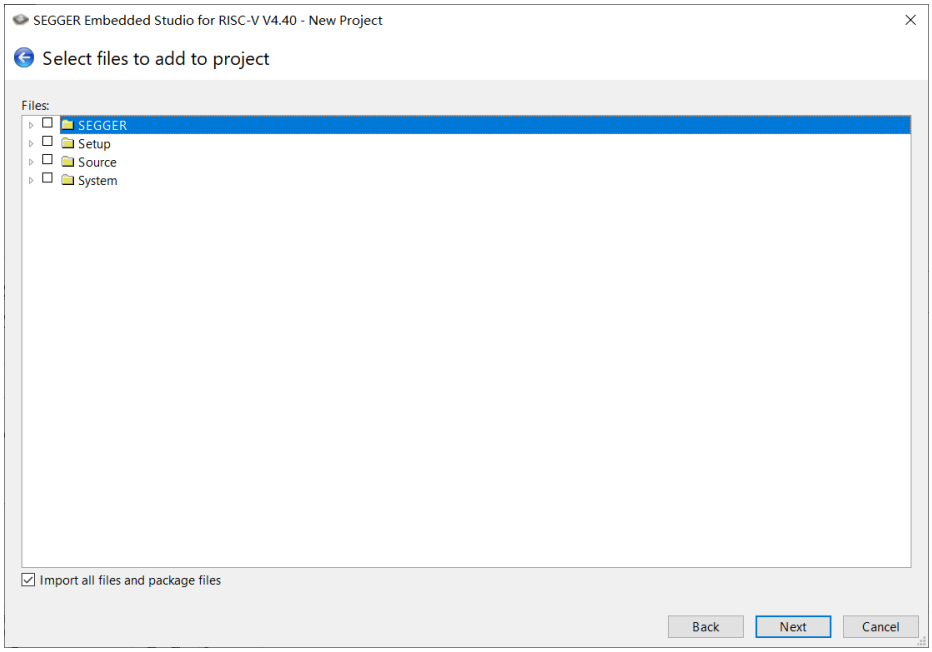

Figure 2-9 Add Default Files

2.3.2. Add the Source Codes for Project

- Please create new folders and add files according to the actual file structure of project package downloaded in Section 2.2.
- Right click the new project created in the previous section to open the right-click menu, and select “New Folder”, as shown in Figure 2-10.
  - As an example, please create two new folders and name them with “application” and “nuclei_sdk”, where “application” folder is used to save application codes, and “nuclei_sdk” folder is used to save Nuclei-SDK.
- Right click the newly created folder and select “Add Existing File” to add files to the project, as shown in Figure 2-11.
  - Note: Please add only the contents under the hbird folder in SoC directory. The final project directory is shown in Figure 2-12.

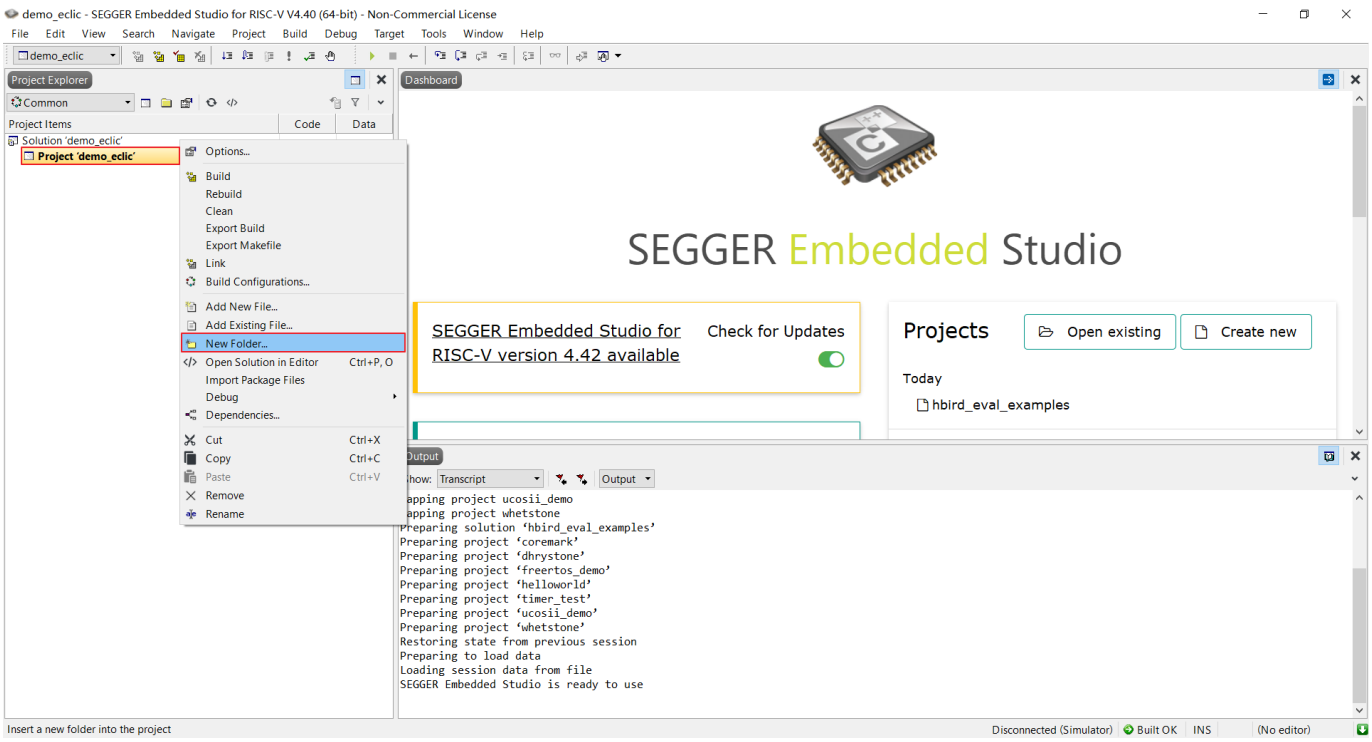

Figure 2-10 Add Folder to the Project

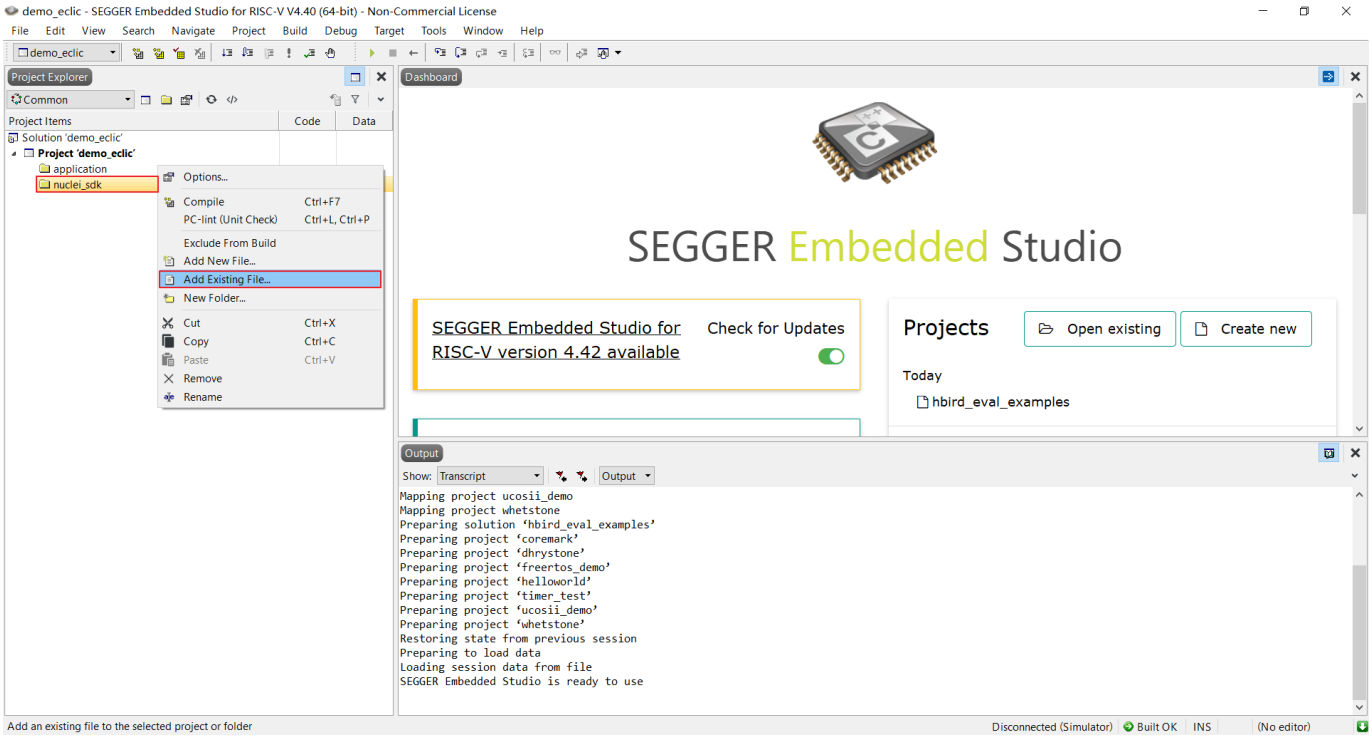

Figure 2-11 Add File to the Project

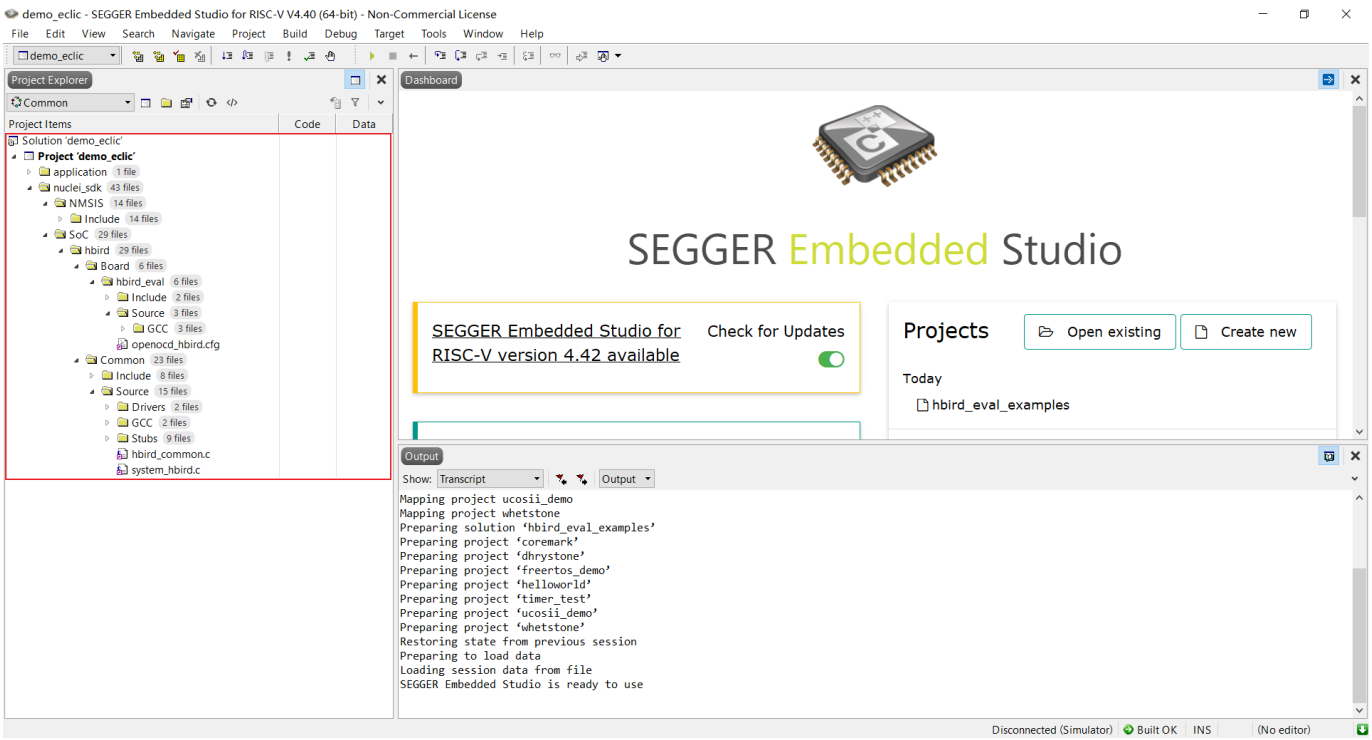

Figure 2-12 The Structure of the New Project in SES

2.3.3. Set the Compile and Link Options for Project

The Nuclei Evaluation SoC supports four "Compiling and Downloading" modes, including:

- ILM (Compile to run program from ILM);
- FLASH (Compile to run program from FLASH);
- FLASHXIP (Compile to upload program from FLASH and run from ILM).
- DDR(Compile to run program from DDR)

Please refer to document <Nuclei_Eval_SoC_Intro.pdf> for more information of the Nuclei Evaluation SoC.

This document takes ILM mode as an example, the setting steps are as follows.

- Select "Project --> Options" in the menu bar to open the project settings pop-up window, as shown in Figure 2-13.

- Modify the “Project Type” to “External Built Executable” in the “Build” column, as shown in Figure 2-14. Click the OK button in the pop-up window to save the settings.

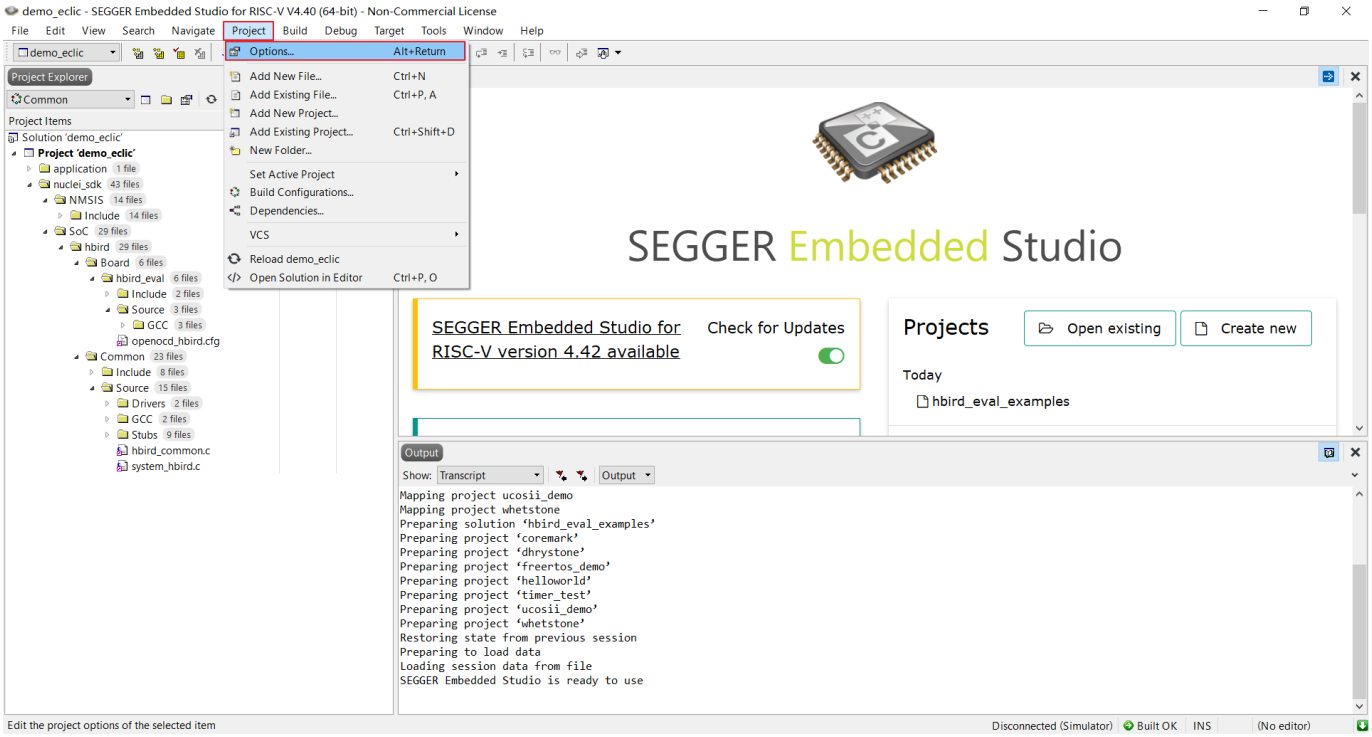

Figure 2-13 Open Project Configuration

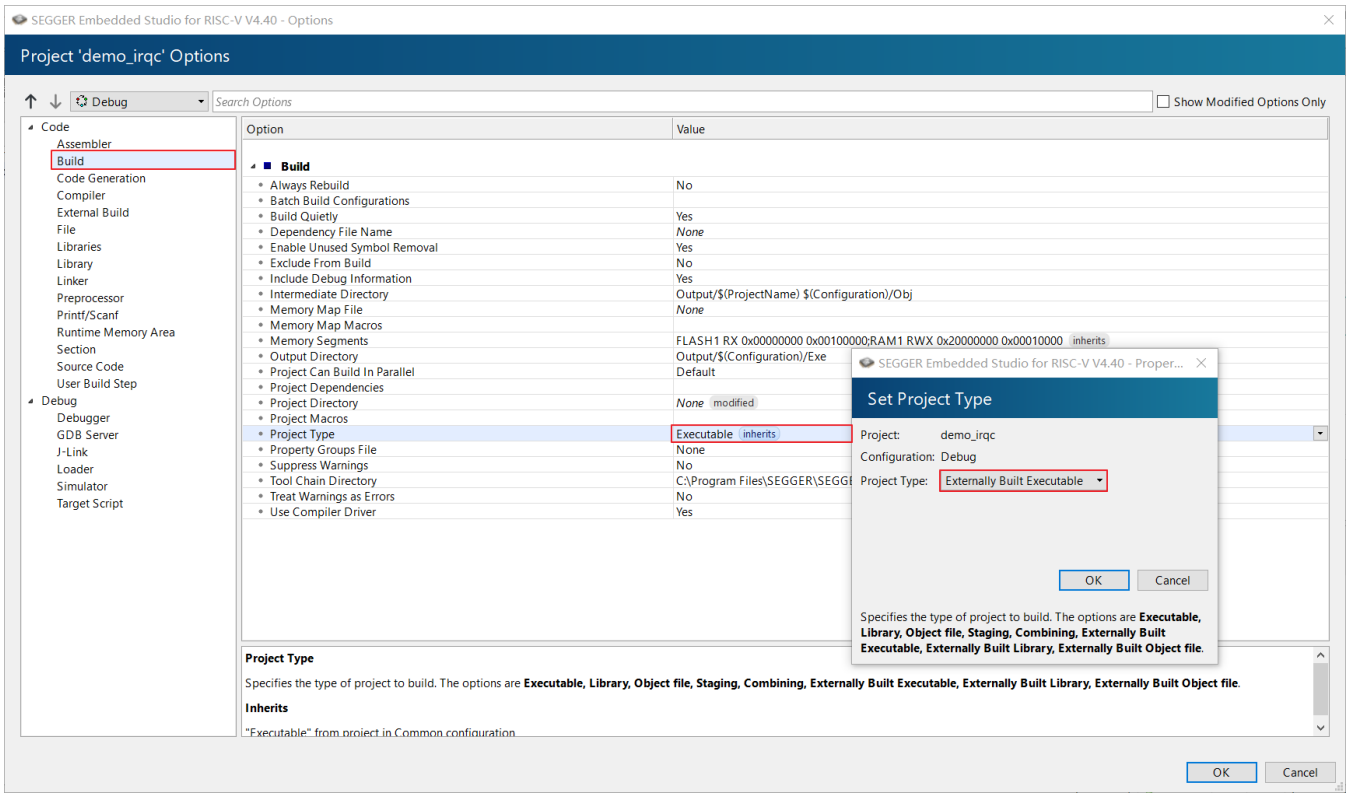

Figure 2-14 Change Project Type

- Select "Project --> Options" in the menu bar again to open the project settings pop-up window. Modify the “Tool Chain Directory” under the “Build” column to “$(StudioDir)/Nuclei_Toolchain/gcc/bin”, as shown in Figure 2-15.

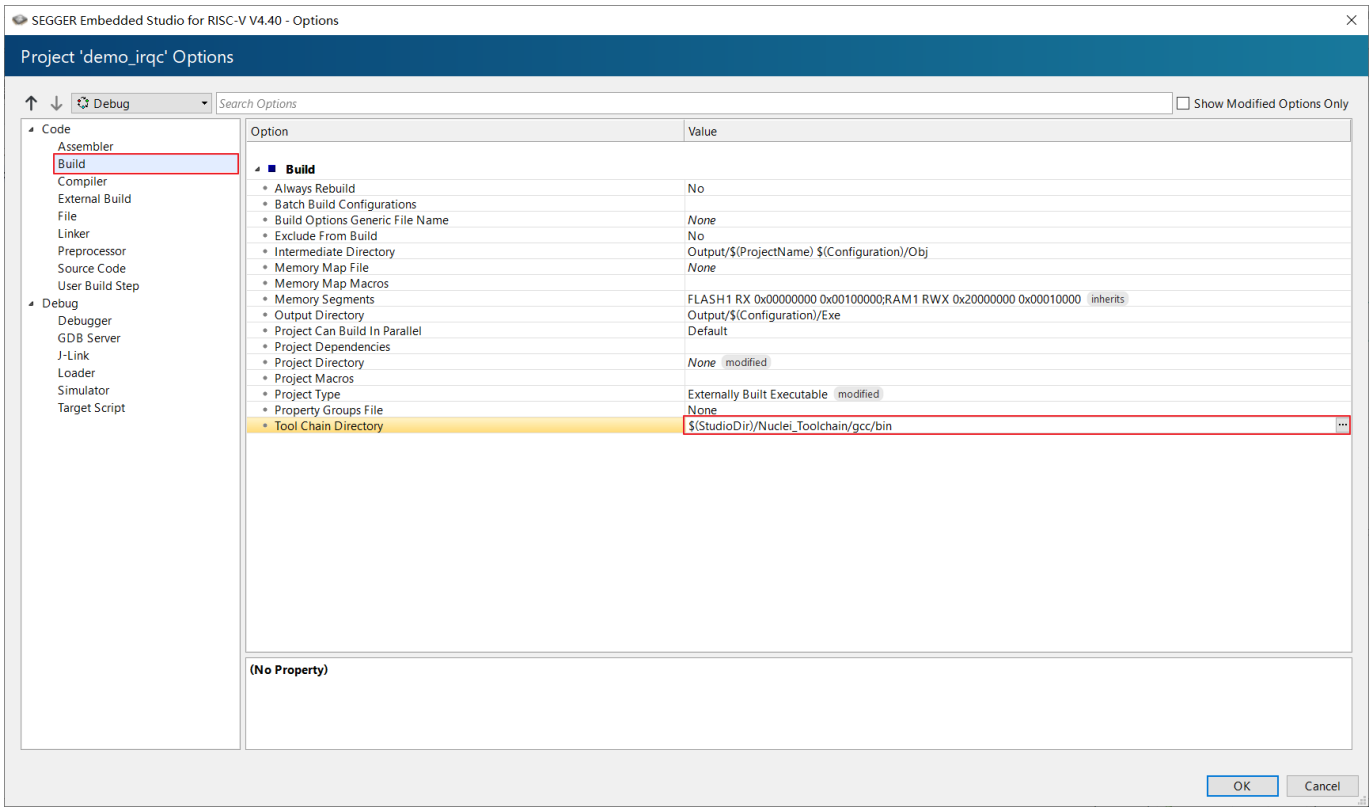

Figure 2-15 Change Tool Chain Directory

- Modify the compilation options, as shown in Figure 2-16. Fill in the compilation instructions as follows.

  Assemble Command："$(ToolChainDir)/riscv-nuclei-elf-gcc" $(CORE_FLAGS) $(COMMON_FLAGS) $(GC_CFLAGS) $(AsmOptions) $(Defines) $(Includes) -MD -MF "$(RelDependencyPath)" -c -o "$(RelTargetPath)" "$(RelInputPath)"

  C Compile Command："$(ToolChainDir)/riscv-nuclei-elf-gcc" $(CORE_FLAGS) $(COMMON_FLAGS) $(GC_CFLAGS) $(COptions) $(COnlyOptions) $(Defines) $(Includes) -MD -MF "$(RelDependencyPath)" -c -o "$(RelTargetPath)"

"$(RelInputPath)"

C++ Compile Command："$(ToolChainDir)/riscv-nuclei-elf-g++" $(CORE_FLAGS) $(COMMON_FLAGS) $(GC_CFLAGS) $(COptions) $(CppOnlyOptions) $(Defines) $(Includes) -MD -MF "$(RelDependencyPath)" -c -o "$(RelTargetPath)" "$(RelInputPath)"

Link Command："$(ToolChainDir)/riscv-nuclei-elf-gcc" $(CORE_FLAGS) $(COMMON_FLAGS) $(GC_LDFLAGS) $(NEWLIB_LDFLAGS) $(EXTRA_LDFLAGS) --specs=nosys.specs -MMD -MT $(ProjectName)$(EXE) -MF $(ProjectName)$(EXE).d $(Objects) -o "$(OutDir)/$(ProjectName)$(EXE)" -T "$(RelLinkerScriptPath)" -lstdc++ -nostartfiles -Wl,-M,-Map="$(RelMapPath)" $(LinkOptions)

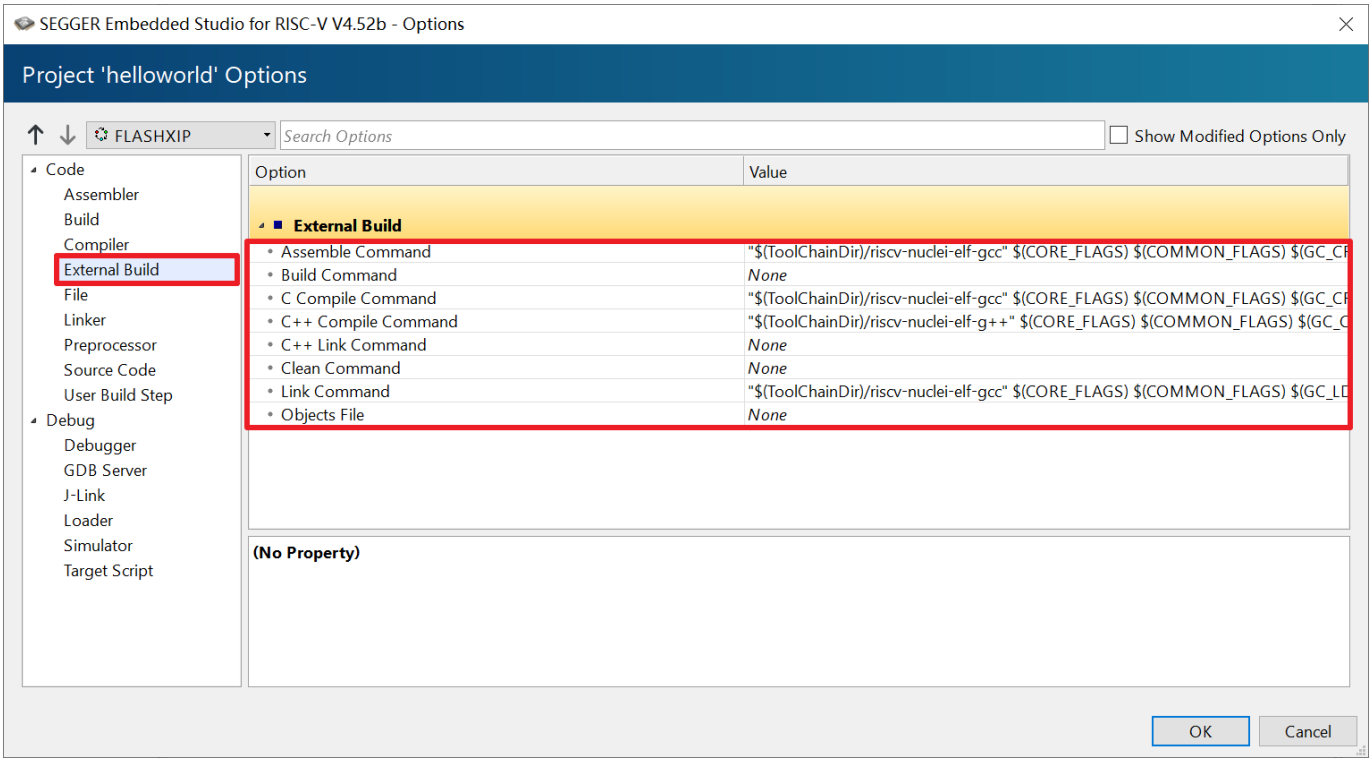

Figure 2-16 Change compilation options

- Modify the Project Macros. As shown in Figure 2-17 , Open Build options and modify the Project Macros option to the following. Note that each Macro should be separated by a new line.

  CORE_FLAGS=-march=rv32imafc -mabi=ilp32f -mcmodel=medany

COMMON_FLAGS=-g -fno-common
-DDOWNLOAD_MODE=DOWNLOAD_MODE_FLASHXIP

GC_CFLAGS=-ffunction-sections -fdata-sections

GC_LDFLAGS=-Wl,--gc-sections -Wl,--check-sections

NEWLIB_LDFLAGS=--specs=nano.specs

EXTRA_LDFLAGS=-u _isatty -u _write -u _sbrk -u _read -u _close -u _fstat -u _lseek

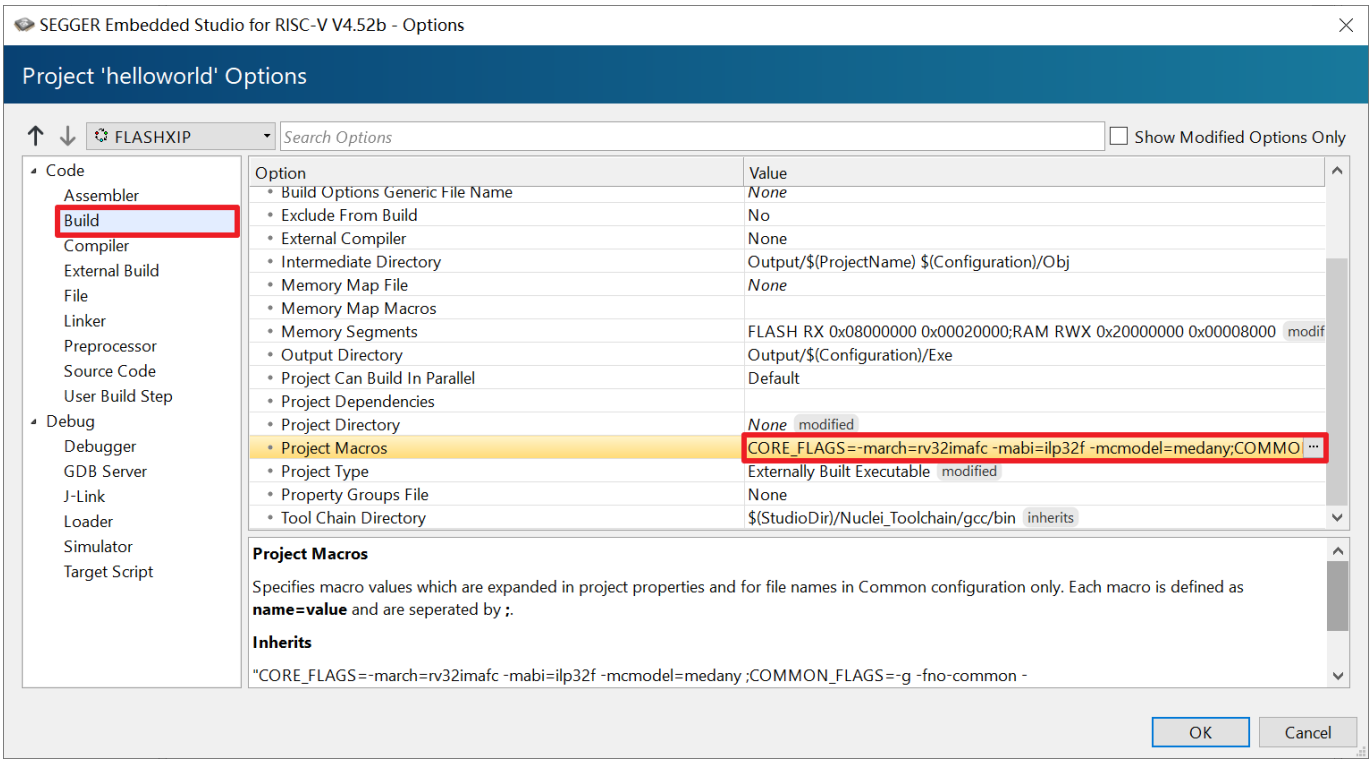

Figure 2-17 Change Project Macros

- Modify the “Use Manual Linker Script” option under “Linker” column to “Yes”, as shown in Figure 2-18. When the option is set to “Yes”, a new option is added, as shown in Figure 2-19. Enter the path of link script in this option.

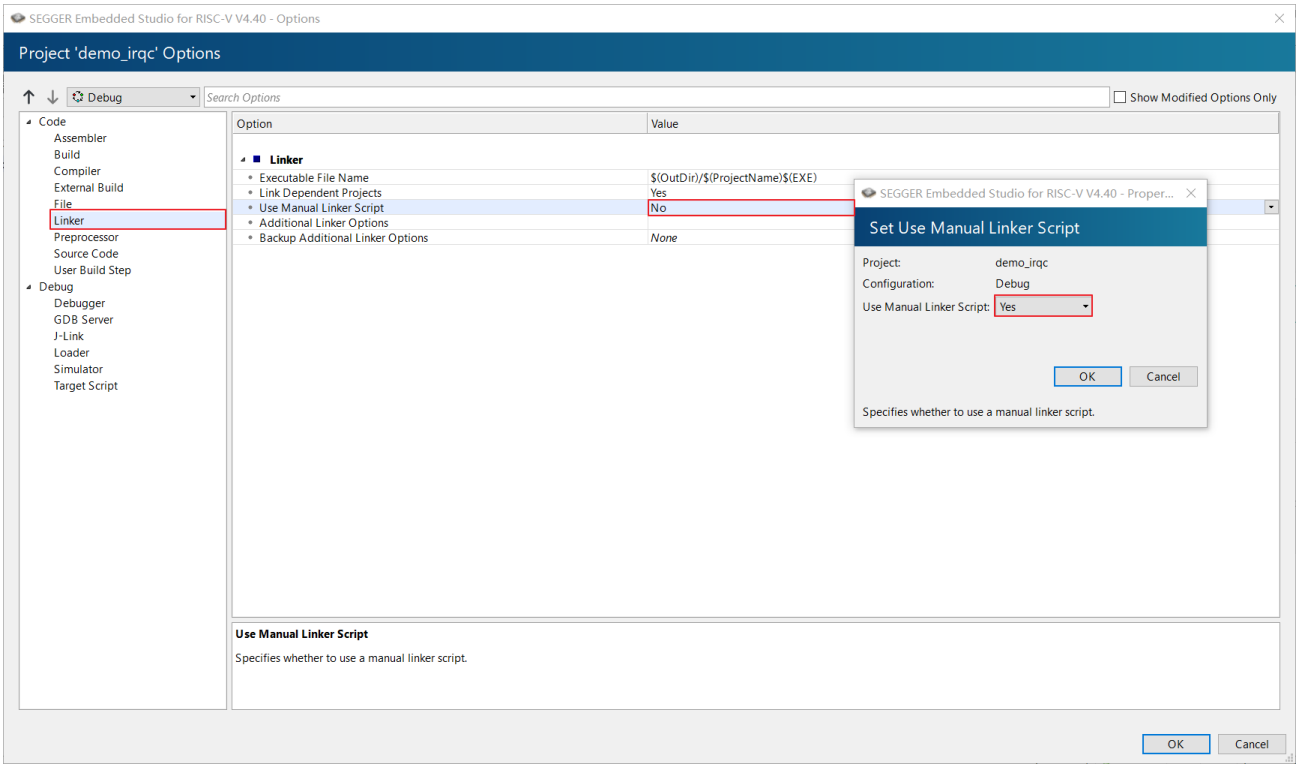

Figure 2-18 Set Use Manual Linker Script Option to Yes

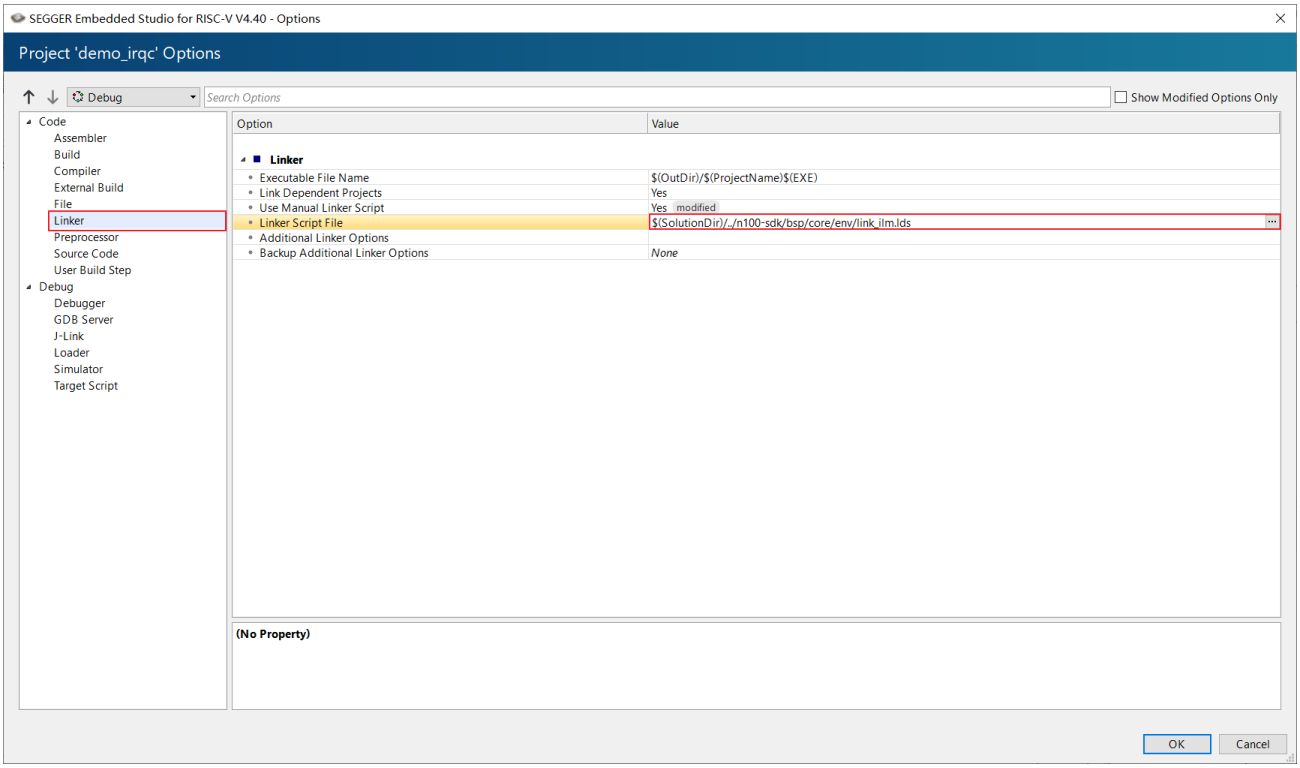

Figure 2-19 Set Linker Script Directory

2.3.4. Set the Include File for Project

- Select "Project --> Options" in the menu bar to open the project settings pop-up window, as shown in Figure 2-20. Under the “Preprocessor” option, enter the required header file path for the project, including nuclei_sdk header file path and application header file path.

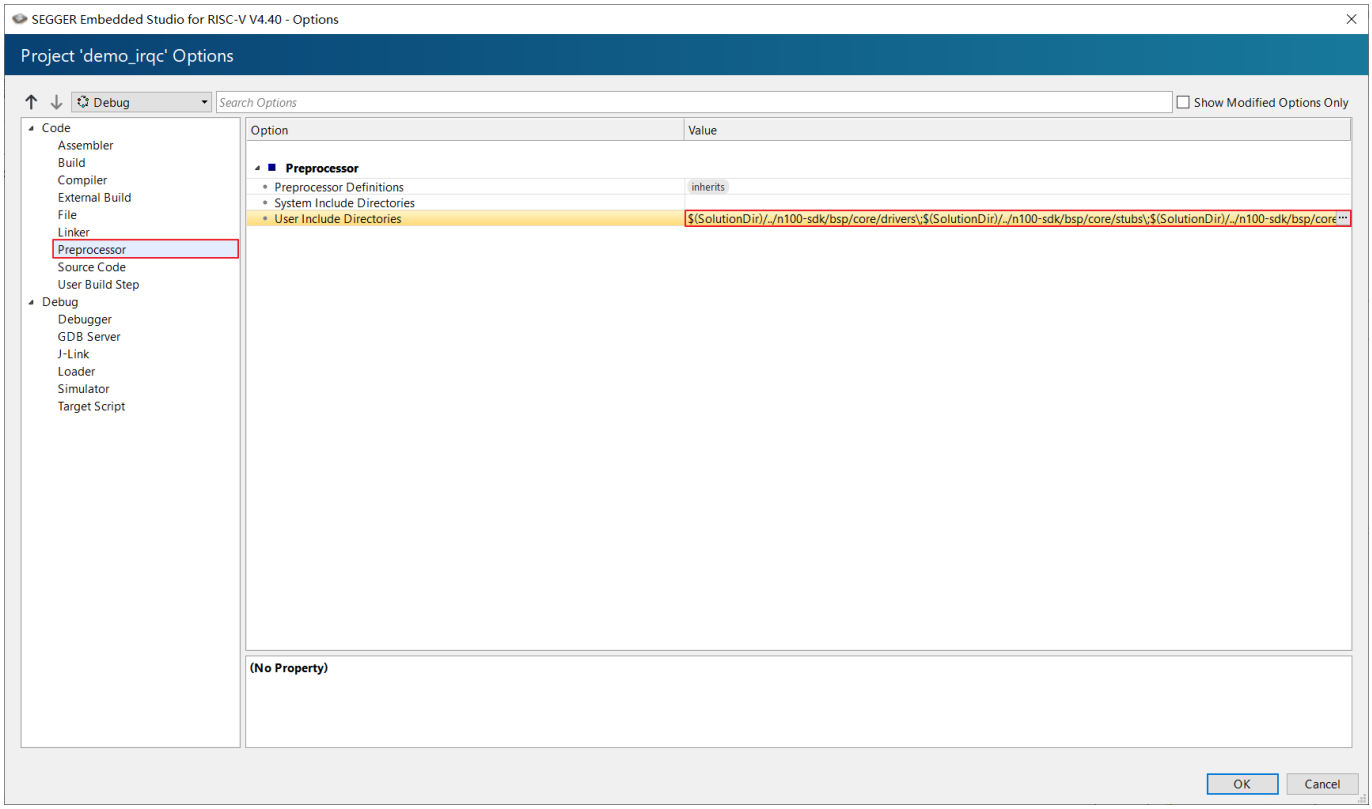

Figure 2-20 Project’s Header File Path

3. Compile Project

3.1. Set Compile Options

Whether you are using an existing project or creating a project manually, please do not modify the content of code > external build unless necessary. This article takes the project imported from an existing project as an example to introduce how to modify the compilation options.

As shown in Figure 3-1, to open Project Options pop-out window, right click in the current project to open the right-click menu bar, then select Options. As shown in Figure 3-2, click the up arrow to enter the Solution Options. As shown in Figure 3-3, in “Code > Build > Project Macros” can find “CORE_FLAGS, COMMON_FLAGS, GC_CFLAGS, GC_LDFLAGS, NEWLIB_LDFLAGS and EXTRA_LDFLAGS”. Each macro is explained separately below.

- CORE_FLAGS: Saves Core related options
- COMMON_FLAGS: Saves common options
- GC_CFLAGS: Compile options when using “gc sections”
- GC_LDFLAGS: Link options when using “gc sections”
- NEWLIB_LDFLAGS: Link options when using NEWLIB
- EXTRA_LDFLAGS: Extra Link options

If you want to modify the above compilation options, you can modify the project settings options. Open the project settings page, as shown in Figure 3-4, and modify the macro at the Project level. It should be noted that the Project level will overwrite the Solution level for the macro with the same name. Therefore, if you want to add compilation options in current macros, please make sure to copy the original macros first. As shown in Figure 3-5, here we change optimization level to “-O2”. Because the Project level will overwrite the Solution level, we should copy the contents of “COMMON_FLAGS” first. Then we add “-O2” at front. Remember to add space between each option.

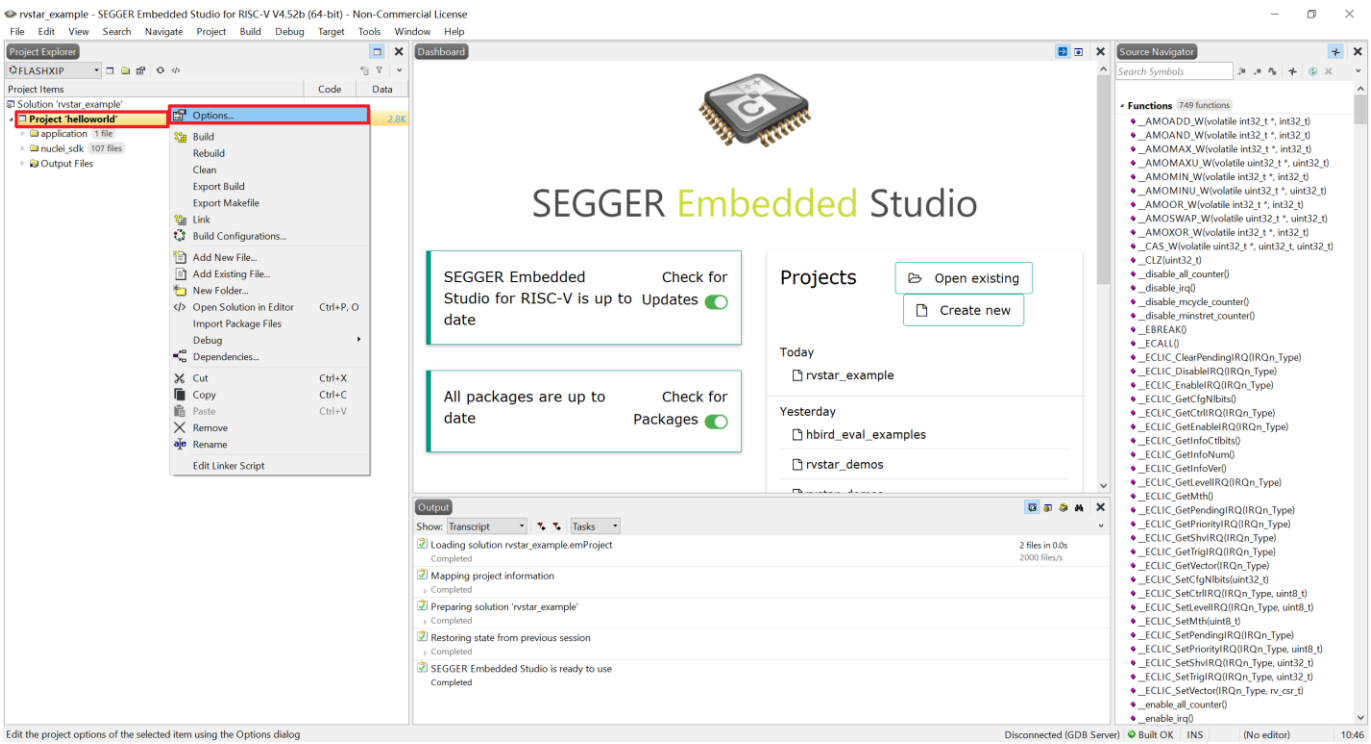

Figure 3-1 Open Project Options

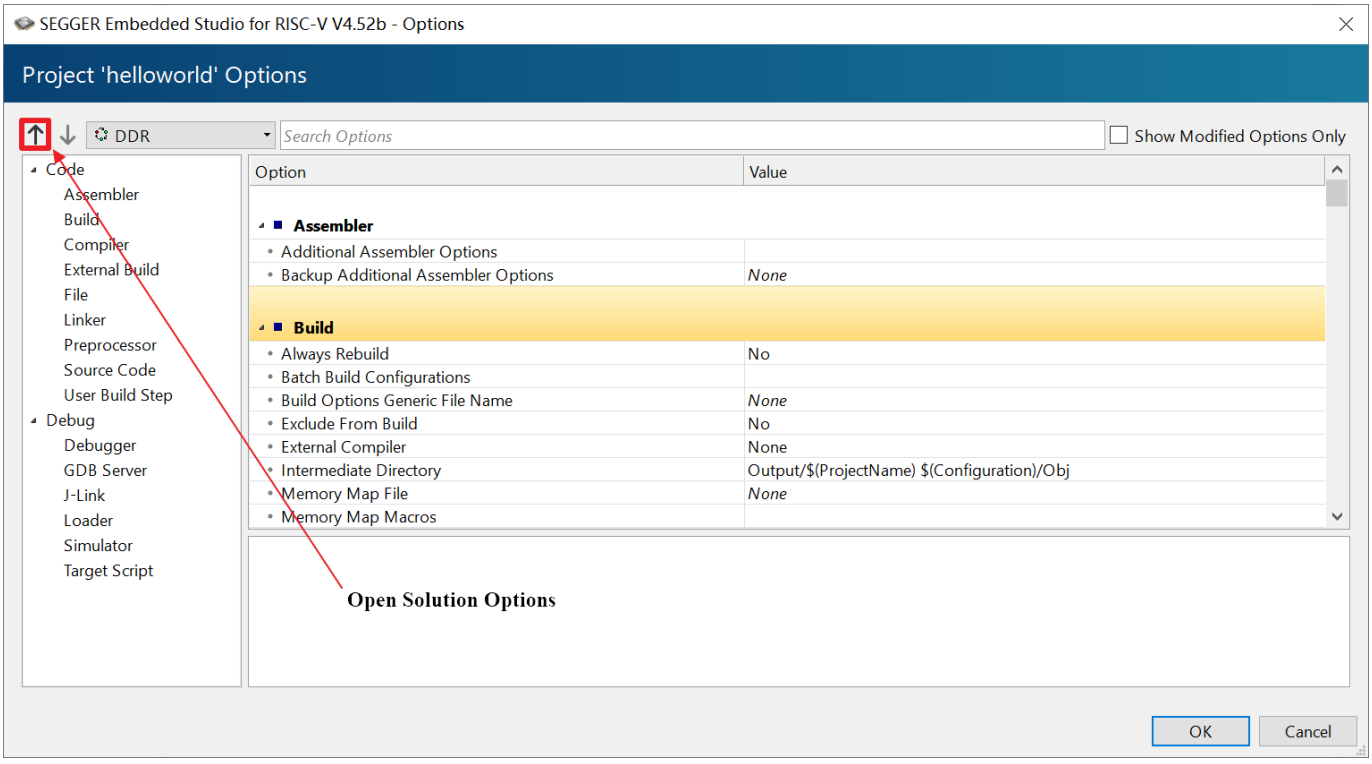

Figure 3-2 Open Solution Options

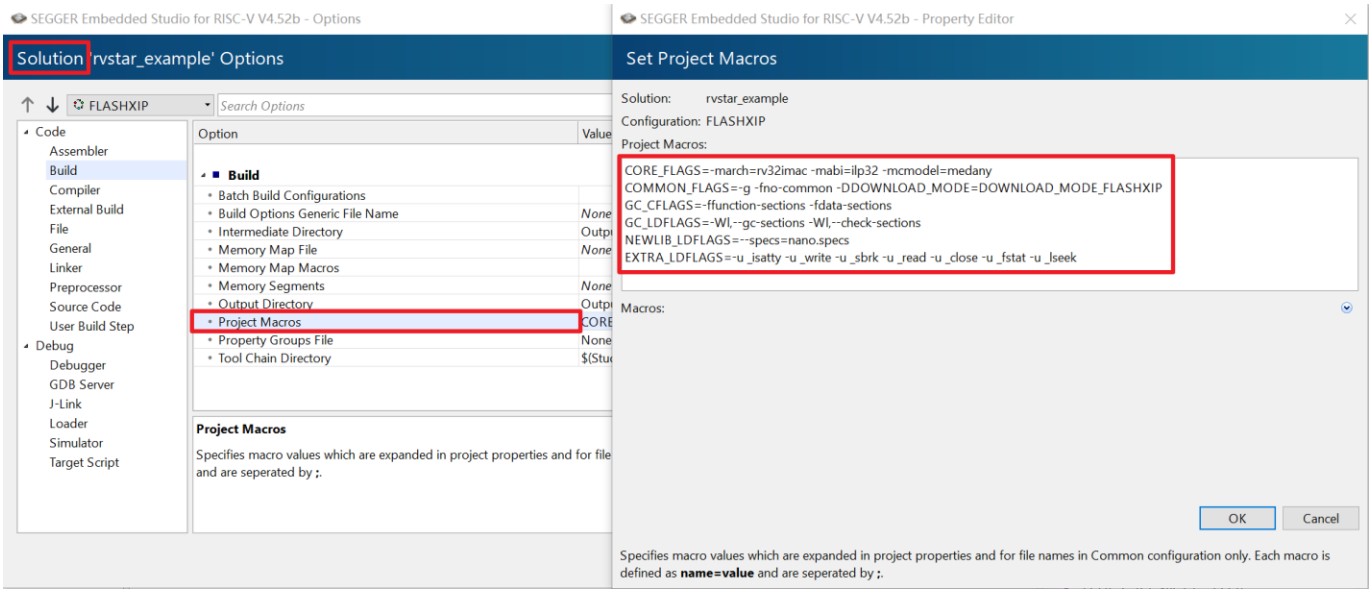

Figure 3-3 View Macros Defined in Solution

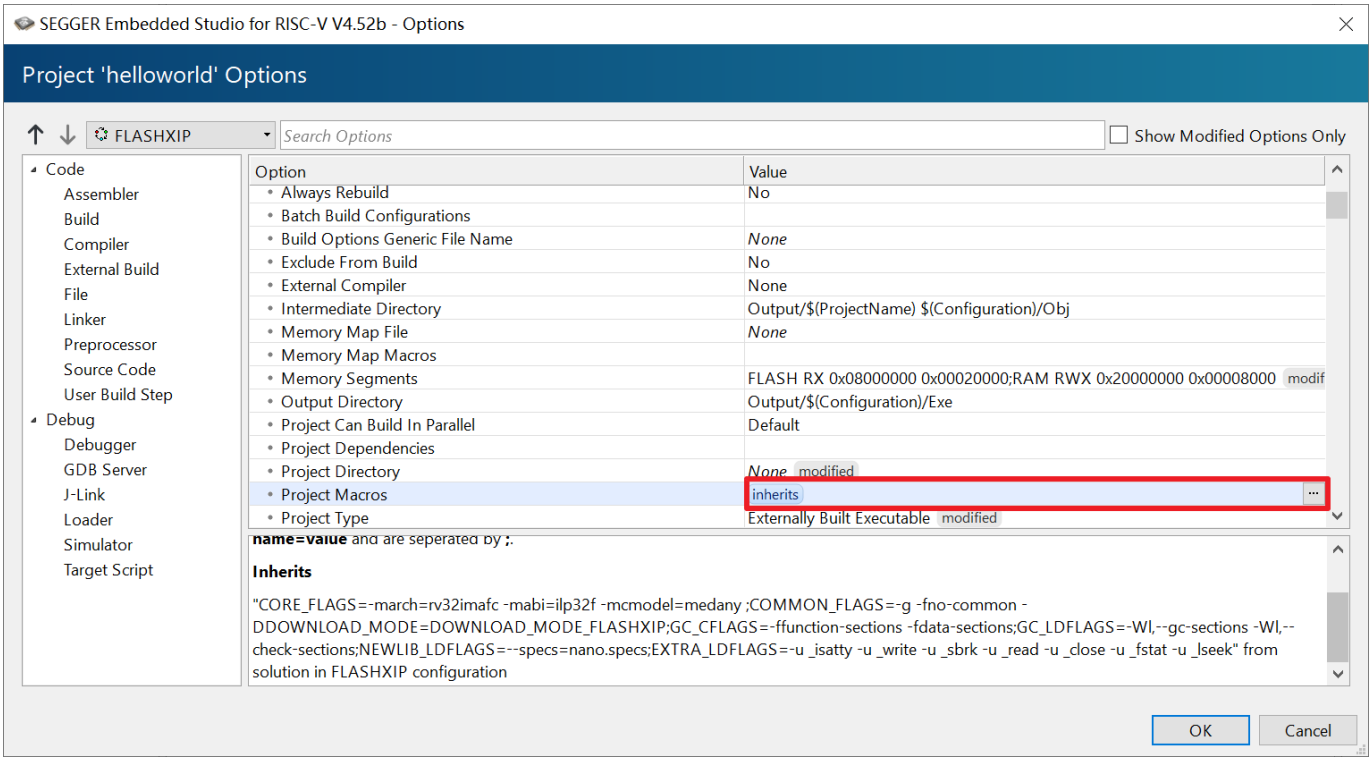

Figure 3-4 Change Project Macros

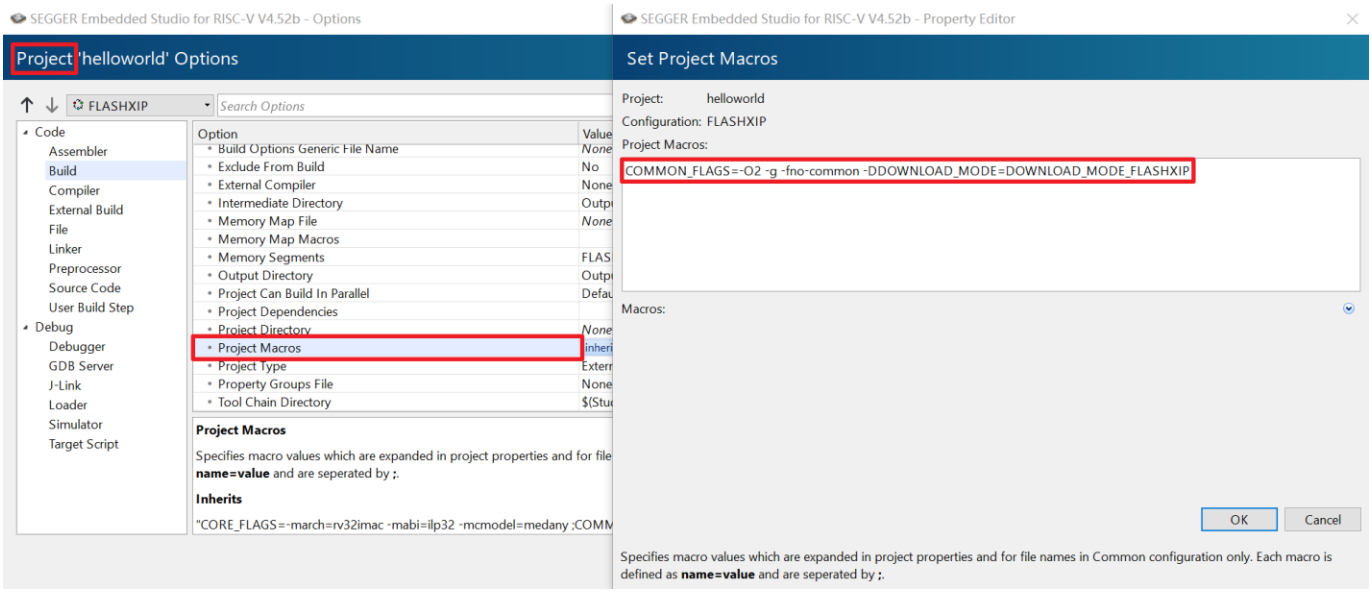

Figure 3-5 Change "COMMON_FLAGS" In Project Level

If the compilation options to be added are not in the above categories, you can add options by yourselves. Make sure each command takes a line in the pop-out window.

Add assembler options in "Code > Assembler > Additional Assembler Options".

Add C compiler only options in "Code > Compiler > Additional C Compiler Only Options".

Add C/C++ compiler only options in "Code > Compiler > Additional C/C++ Compiler Option".

Add C++ compiler only options in "Code > Compiler > Additional C++ Compiler Only Options".

Add Link options in "Code > Linker > Additional Linker Options".

3.2. Compile Project demo_eclic in SES

This section takes the imported project demo_eclic (described in Section 2.2) as an example to introduce how to compile the project and generate Elf file in SES.

After the "compiling and downloading modes (DDR, ILM, FLASH or FLASHXIP)" is selected, to compile the current project, press F7 directly on the keyboard or click the first option under "Build" in the menu bar, as shown in Figure 3-6.

After the compilation is successful, the user interface is as shown in Figure 3-7.

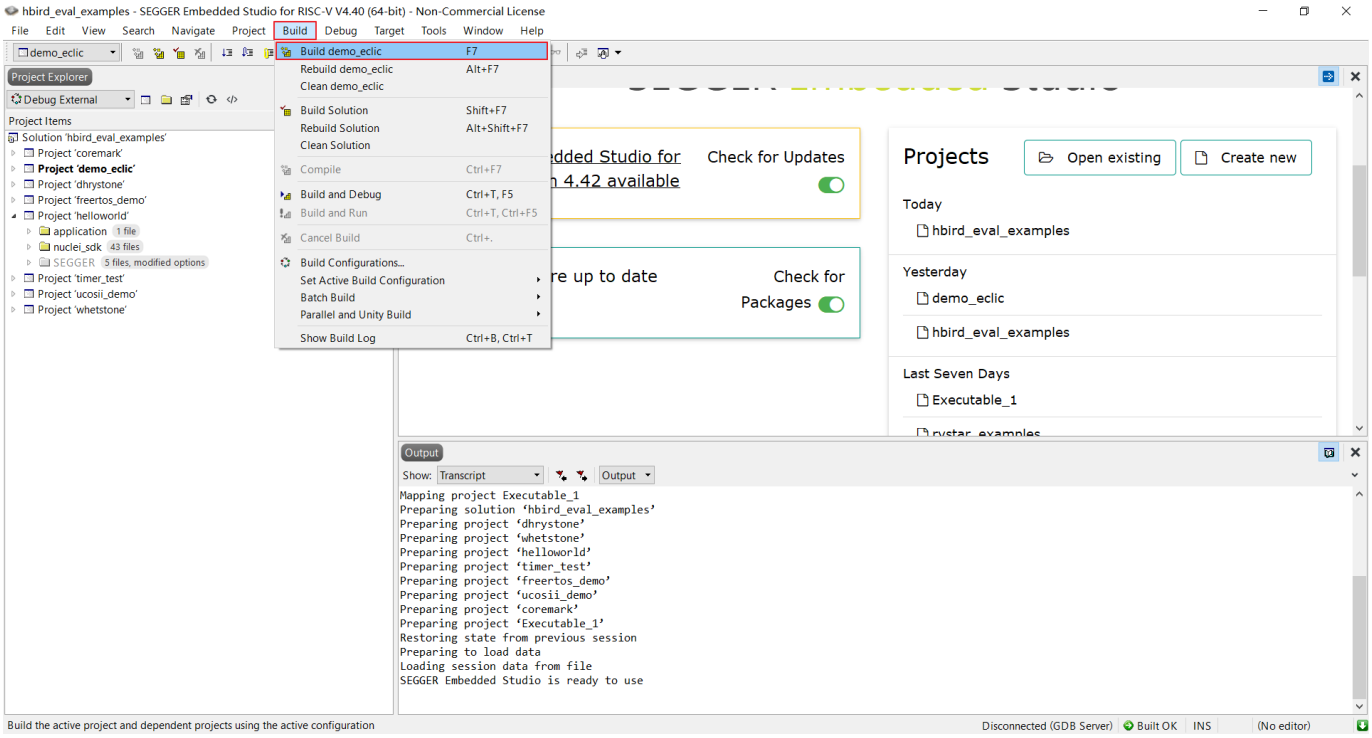

Figure 3-6 Compile Through Build Option in Menu Bar

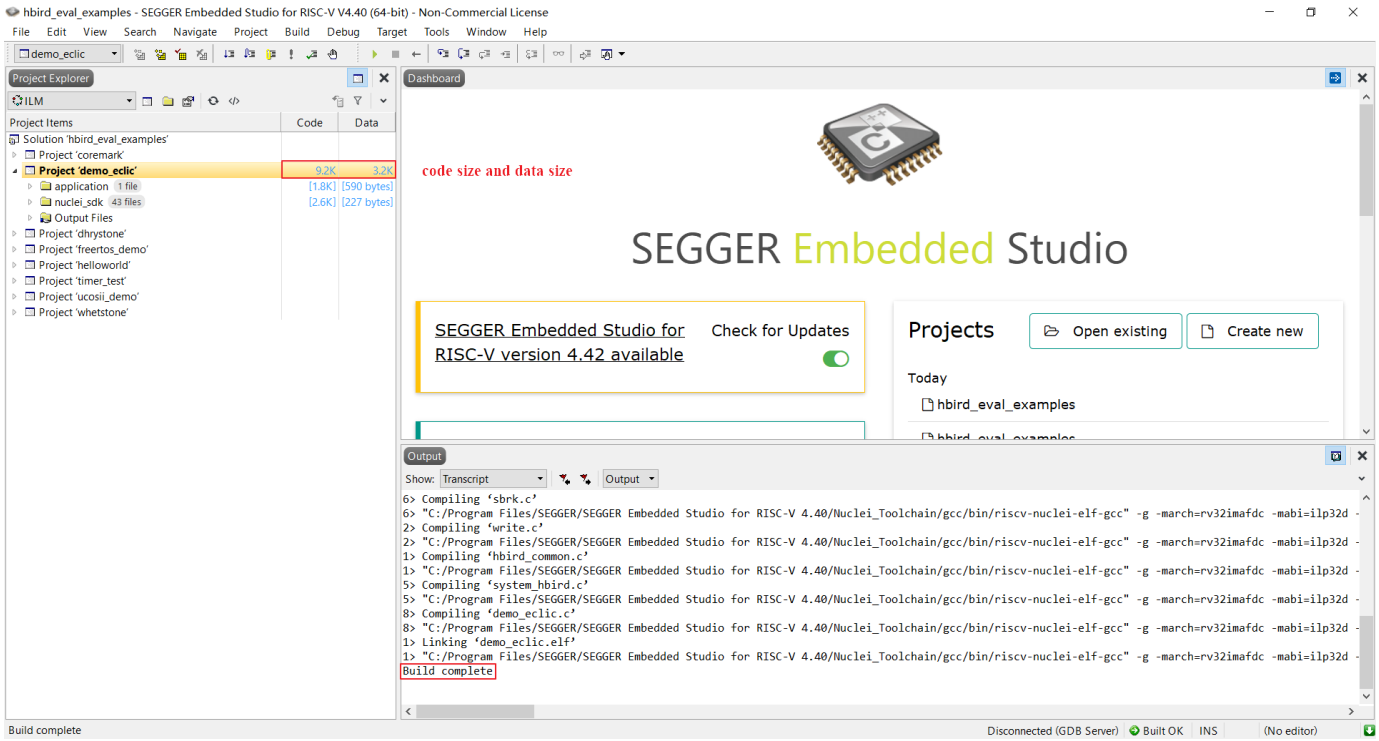

Figure 3-7 Compiled Successfully

4. Download and Run Project

4.1. Set GDB Server according to Debugger Type

SES supports two Debugger types for Nuclei:

- Debug with its native J-Link debugger.
- Debug with Hummingbird Debugger Kit.

Before debugging or downloading the program, user needs to set up GDB Server according to the debugger type.

The steps to set up GDB Server (according to the Debugger type) are detailed as follows.

4.1.1. Debug with Hummingbird Debugger Kit

For the details of Hummingbird Debugger Kit, please refer to Section 1.4.

The Hummingbird Debugger Kit is connected to the host PC through USB, therefore, USB driver needs to be installed. About how to connect Hummingbird Debugger Kit, and how to install its driver, please refer to the document <Nuclei_FPGA_DebugKit_Intro.pdf> which can be downloaded from “Development Boards” page of Nuclei website (http://www.nucleisys.com/developboard.php).

The final hardware connection is shown in Figure 4-1. After the correct connection, turn on the power switch to connect board.

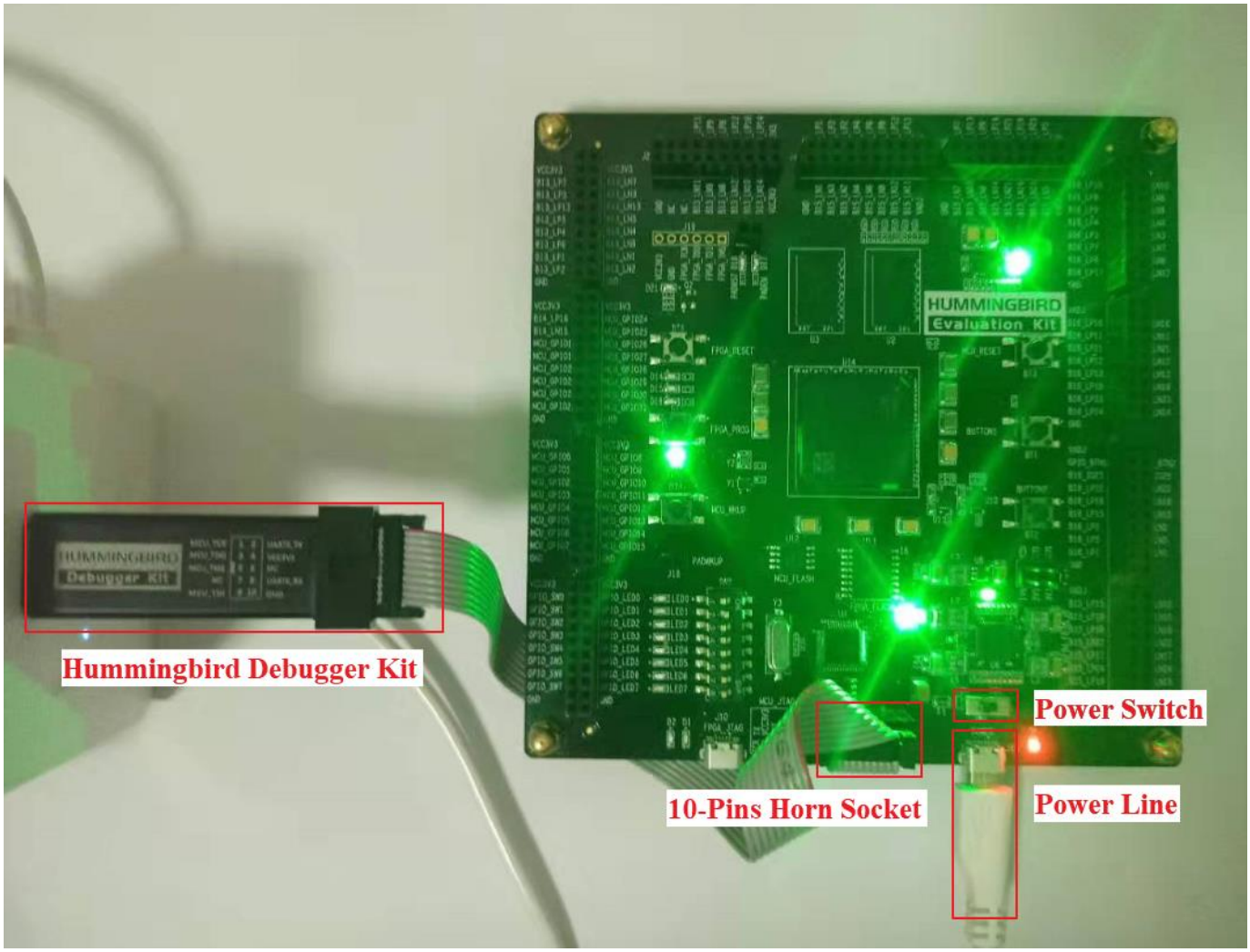

Figure 4-1 Hardware Connection of Hummingbird Debugger Kit

The steps to set up GDB Server for Hummingbird Debugger Kit are as follows:

- Click “Options” under “Project” option in the menu bar, as shown in Figure 4-2.
- Change the “Target Connection” option to “GDB Server” under the “Debugger” column, as shown in Figure 4-3.

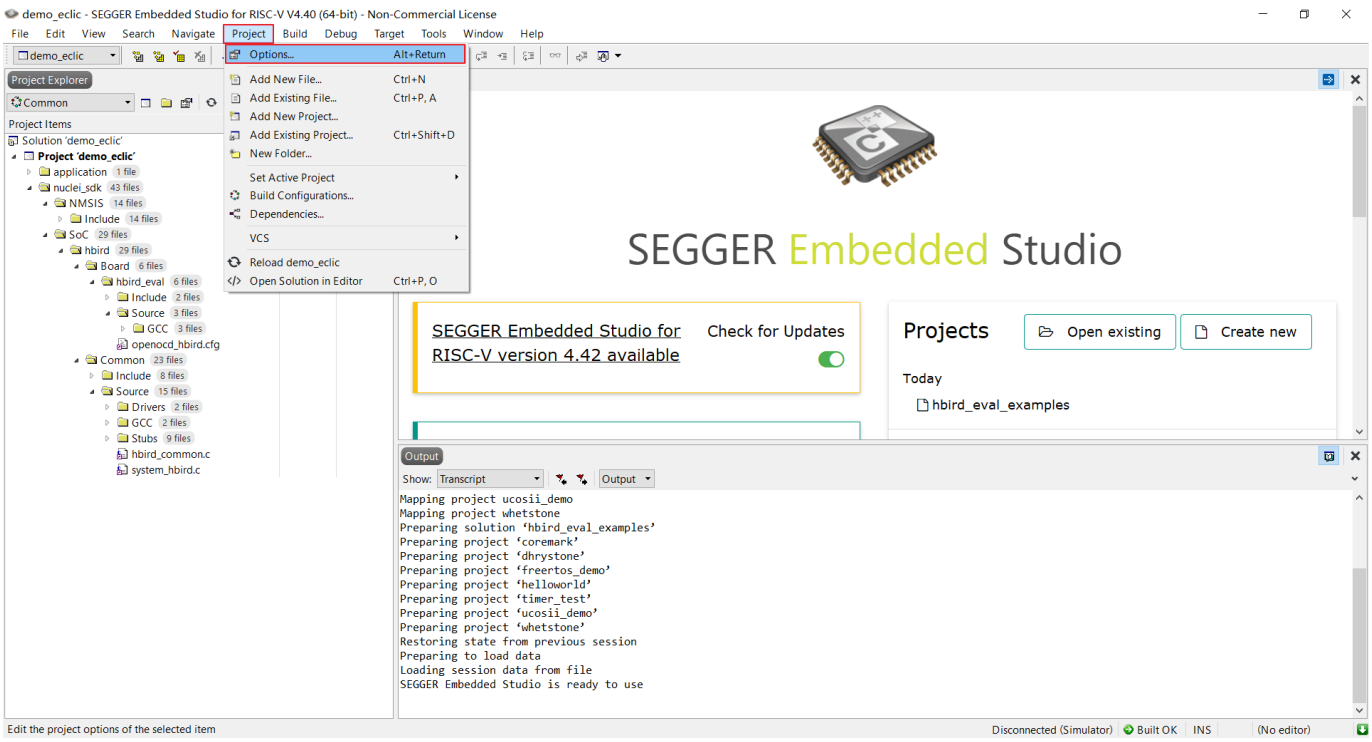

Figure 4-2 Open Project Configuration

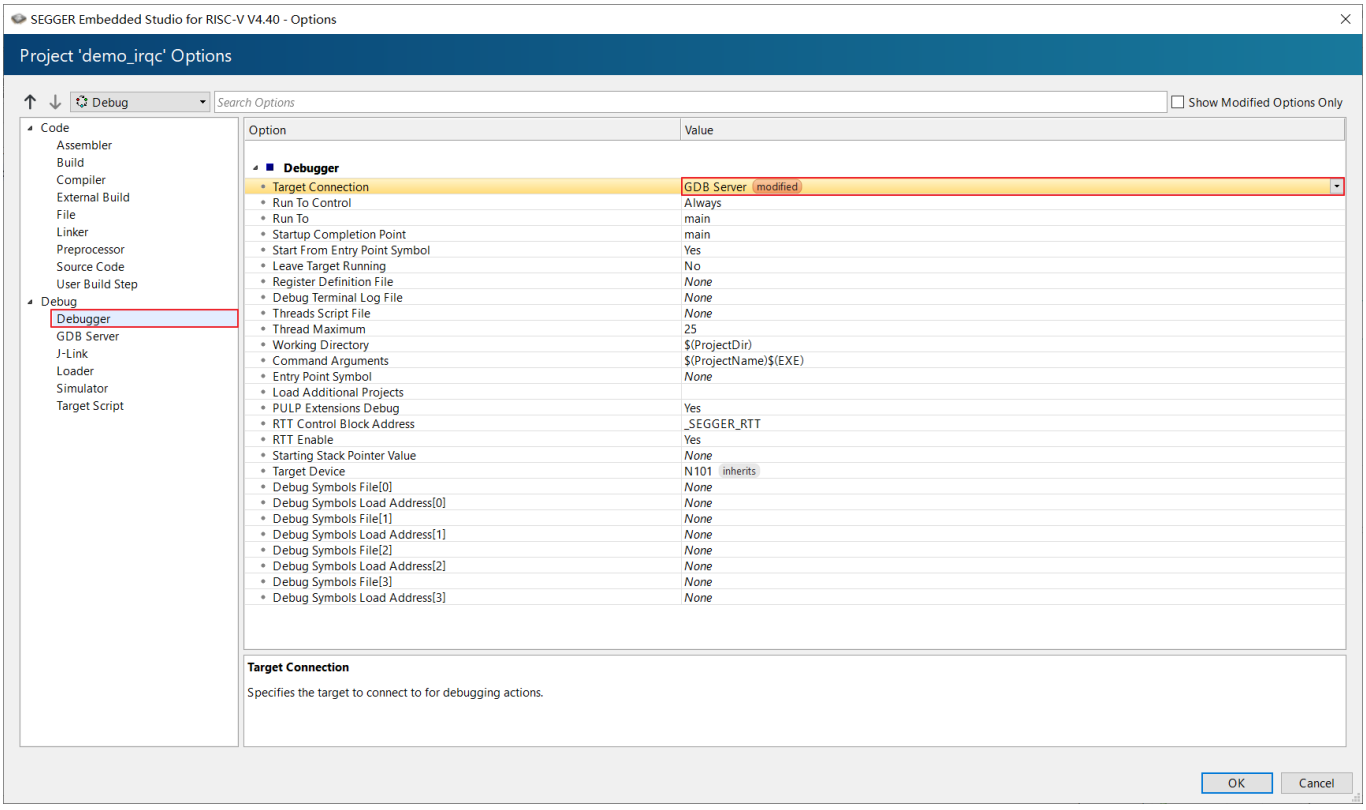

Figure 4-3 Set to Use GDB Server

- As shown in Figure 4-4, There are three settings under the “GDB Server” option should be changed:
  - Change the type to OpenOCD.
  - Change the “GDB Server Command Line” to "$(studiodir)/Nuclei_Toolchain/OpenOCD/bin/OpenOCD" -f, and add the path of OpenOCD's configuration file. Here you should select the configuration file corresponding to the Downloading mode (DDR, ILM, FlASH, or FLASHXIP).
  - Modify “Auto Start GDB Server” to “Yes”.
- After the above modifications, click OK to save the project settings.

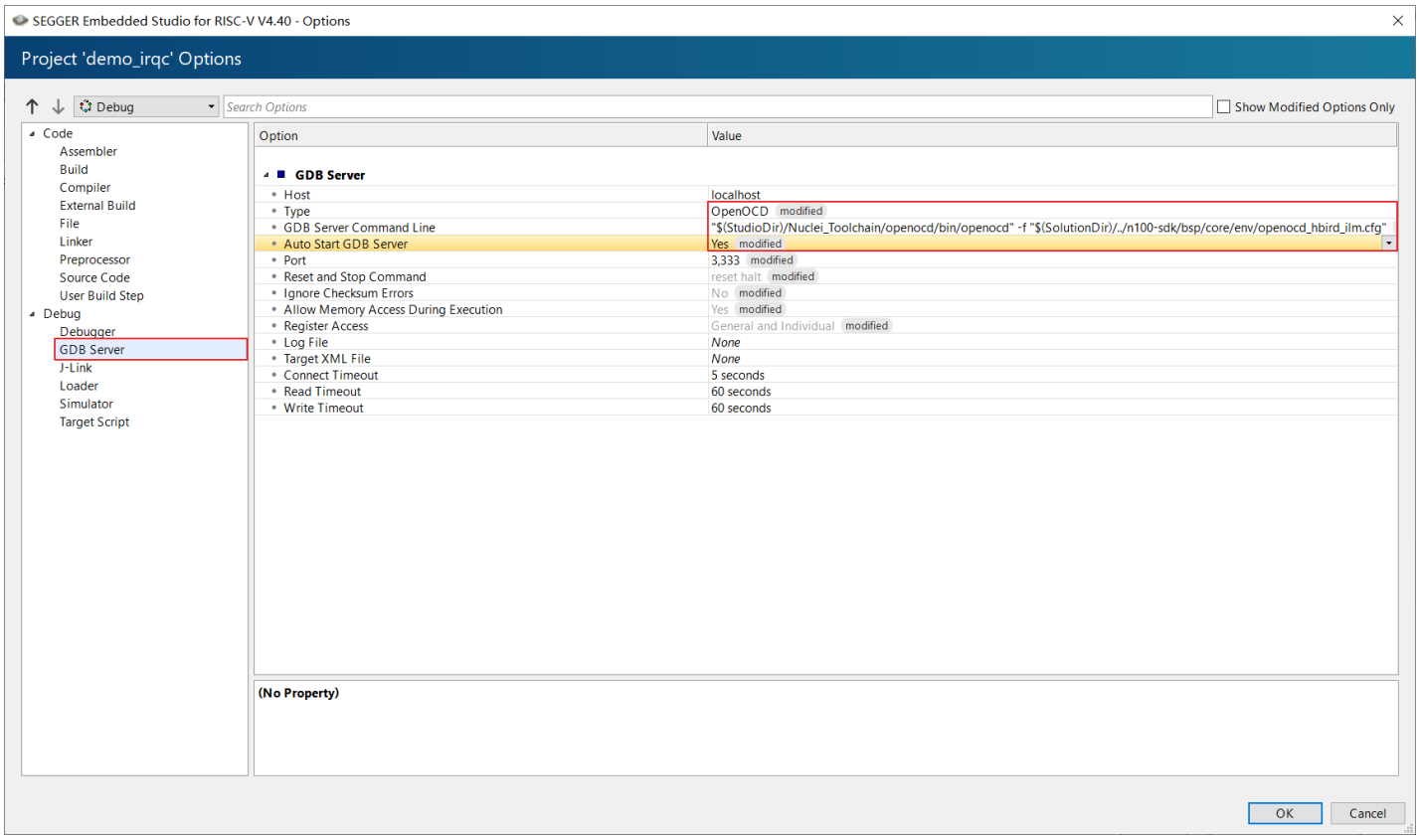

Figure 4-4 Use Openocd to Connect

4.1.2. Debug with J-Link

SES certainly supports the J-Link Debugger Probe. Note: when using J-Link for debugging, only

ILM mode can be supported. For more detailed reason about this problem, please contact Nuclei.

The key points for connecting of J-Link are as below:

- Prepare several jumper wires.
- The pin diagram of J-Link is as shown in Figure 4-5, with the red-box marked as the pin to be connected.
- The pin diagram of Hummingbird Evaluation Kit FPGA board is next to the 10-Pins Horn Socket as shown in Figure 4-6.
- Connect other pins first, and then lastly connect the “VTref” pin to VCC3V3 on Hummingbird Evaluation Kit.
- The hardware connection of J-Link is as shown in Figure 4-6. If the connection is correct, turn on the power switch.

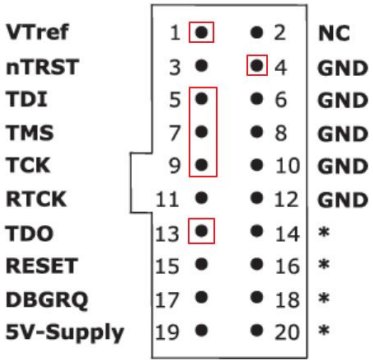

Figure 4-5 The Pin Diagram of J-Link

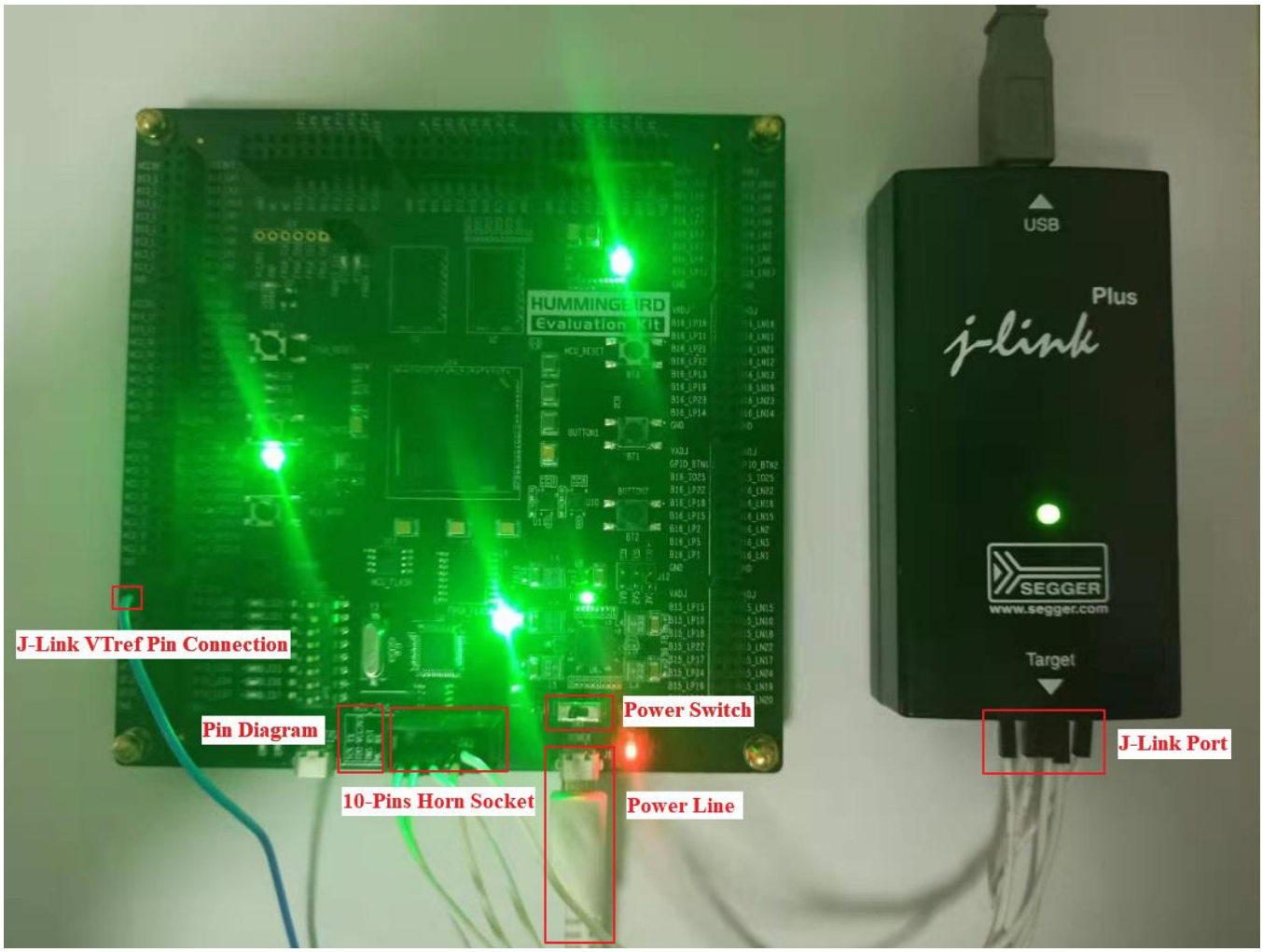

Figure 4-6 Hardware Connection of J-LINK

The steps to set up GDB Server for J-Link are as follows:

- Click “Options” under “Project” option in the menu bar, as shown in Figure 4-2.
- Switch the “Target Connection” option to “J-Link” under the debugger column, as shown in Figure 4-7.

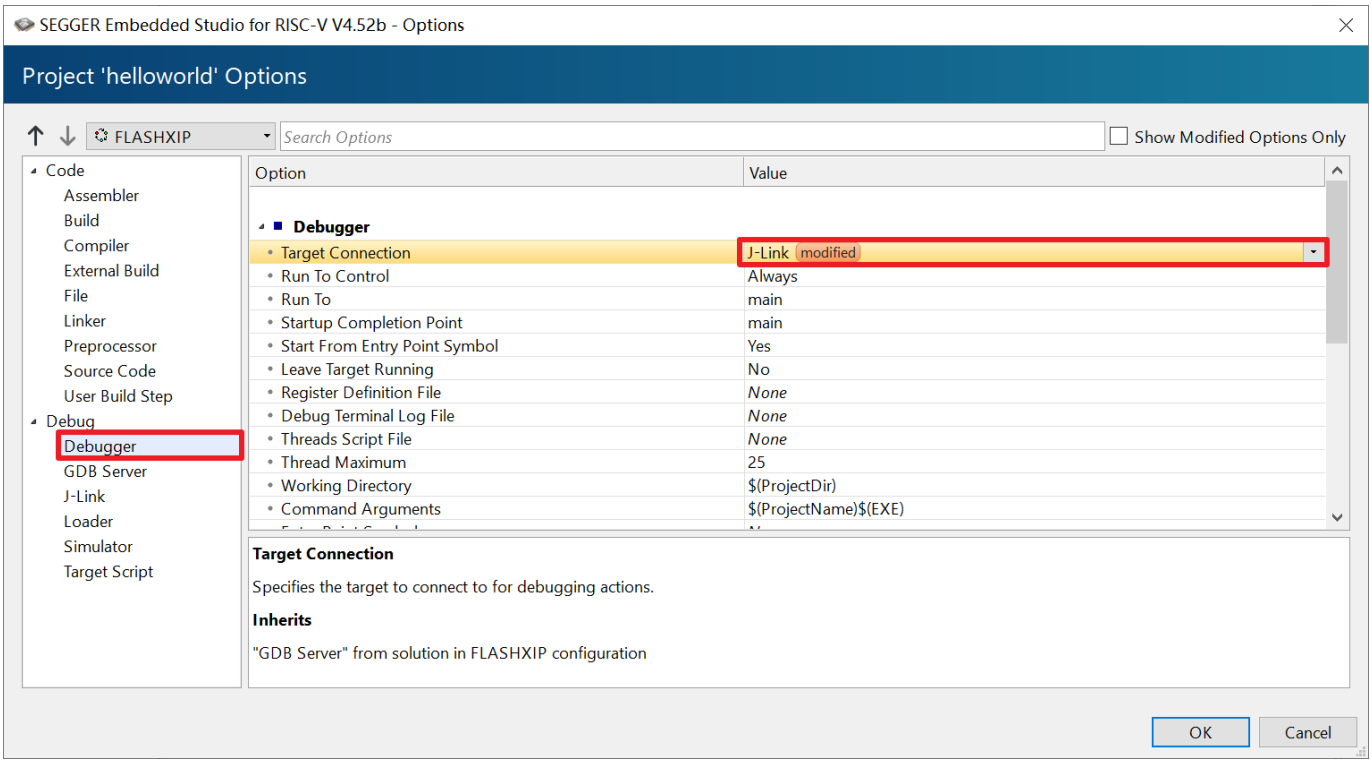

Figure 4-7 Modify Project Settings to Use J-Link

4.2. Set Printout Mode for printf according to Debugger Type

Using printf function can help developers to confirm the running status of programs efficiently. Because embedded system usually does not equip with display screen, it is necessary to redirect the printf function to the host PC.

SES supports two Printout modes for Nuclei:

- Printout through UART Serial Port from Hummingbird Debugger Kit.
- Printout through RTT functions from native J-link debugger.

The steps to set up Printout modes (according to the Debugger type) are detailed as follows.

4.2.1. Printout through Serial Port

In embedded system, UART port of SoC is commonly used to connect COM port of host PC (or USB port of host PC after UART is converted to USB) for debugging, so that printf function in

embedded system can be redirected and printed to display screen of host PC.

The Hummingbird Debugger Kit has a FT2232 chip inside it to convert the UART to USB, and then connect to host PC, which will be recognized as COM port at host PC (if the driver is installed correctly). About how to connect Hummingbird Debugger Kit, and how to install its driver, please refer to the document <Nuclei_FPGA_DebugKit_Intro.pdf> which can be downloaded from “Development Boards” page of Nuclei website (http://www.nucleisys.com/developboard.php).

The setting steps at SES to grab this COM port are as follows:

- After connecting the Hummingbird Debugger Kit, as shown in Figure 4-8, select “Tools - > Terminal Emulator --> Terminal Emulator” in the menu bar to open the Serial Port tool.
- As shown in Figure 4-9, select "Tools - > Terminal emulator - > Properties" in the menu bar to open the Serial Port setting pop-up window.
- The connected COM port can be found (after the driver is installed correctly). As shown in Figure 4-10, set “Baud Rate” to 115200, then double-click “Port” option, select Serial Port number (as recognized by PC), and complete Serial Port selection.
- As shown in Figure 4-11, click the “old type phone” icon to connect the Serial Port.

After the above settings, take demo_eclic as an example, when the program is downloaded to the FPGA evaluation board, if the printf function is used in the running program, you can see the printf output on the SES Serial Port as shown in Figure 4-12.

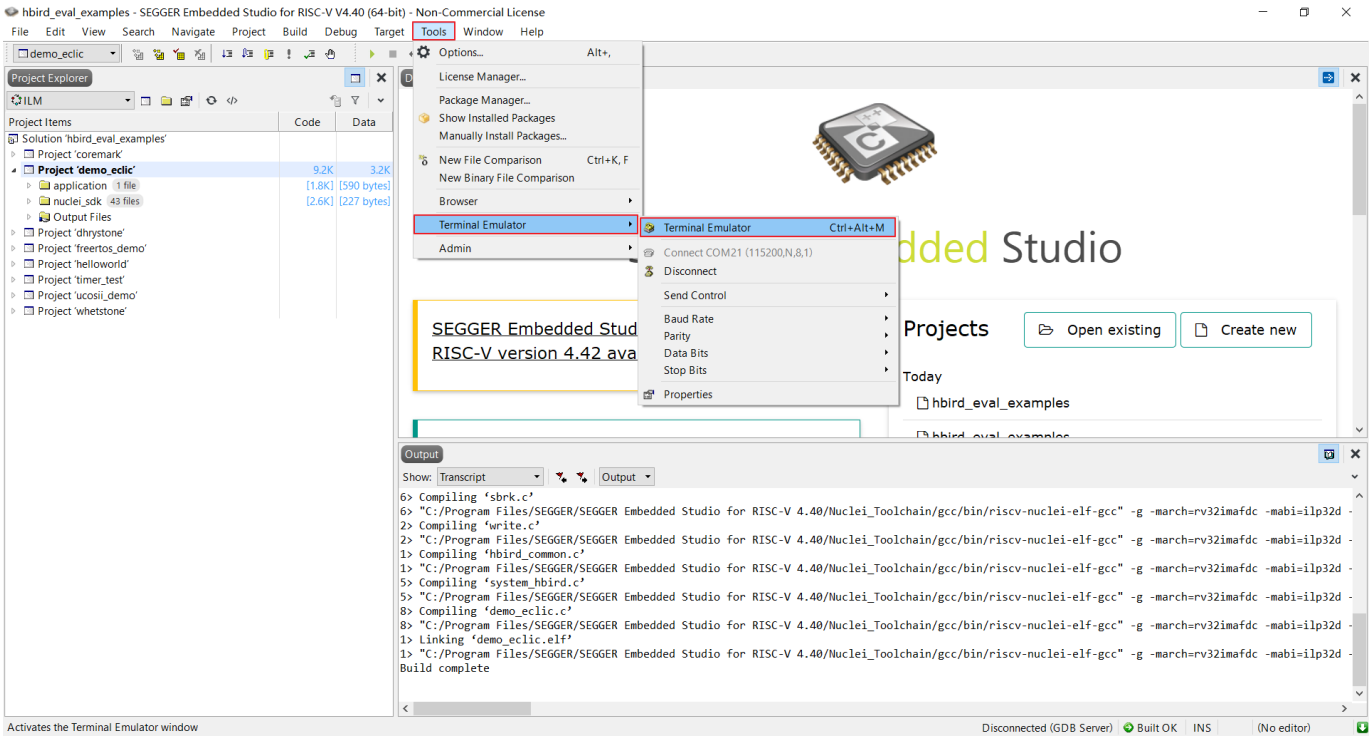

Figure 4-8 Open SES Serial Port Tool

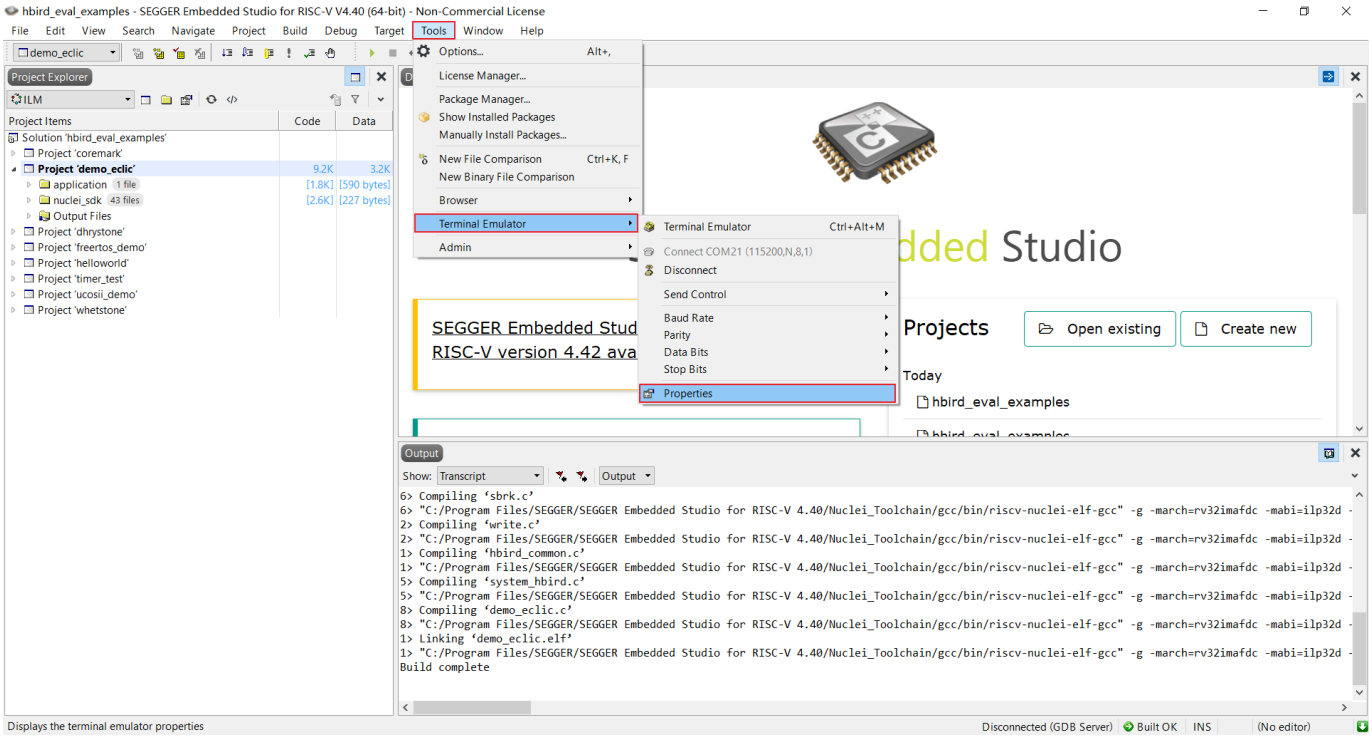

Figure 4-9 Open the Serial Port Setting Pop-up Window

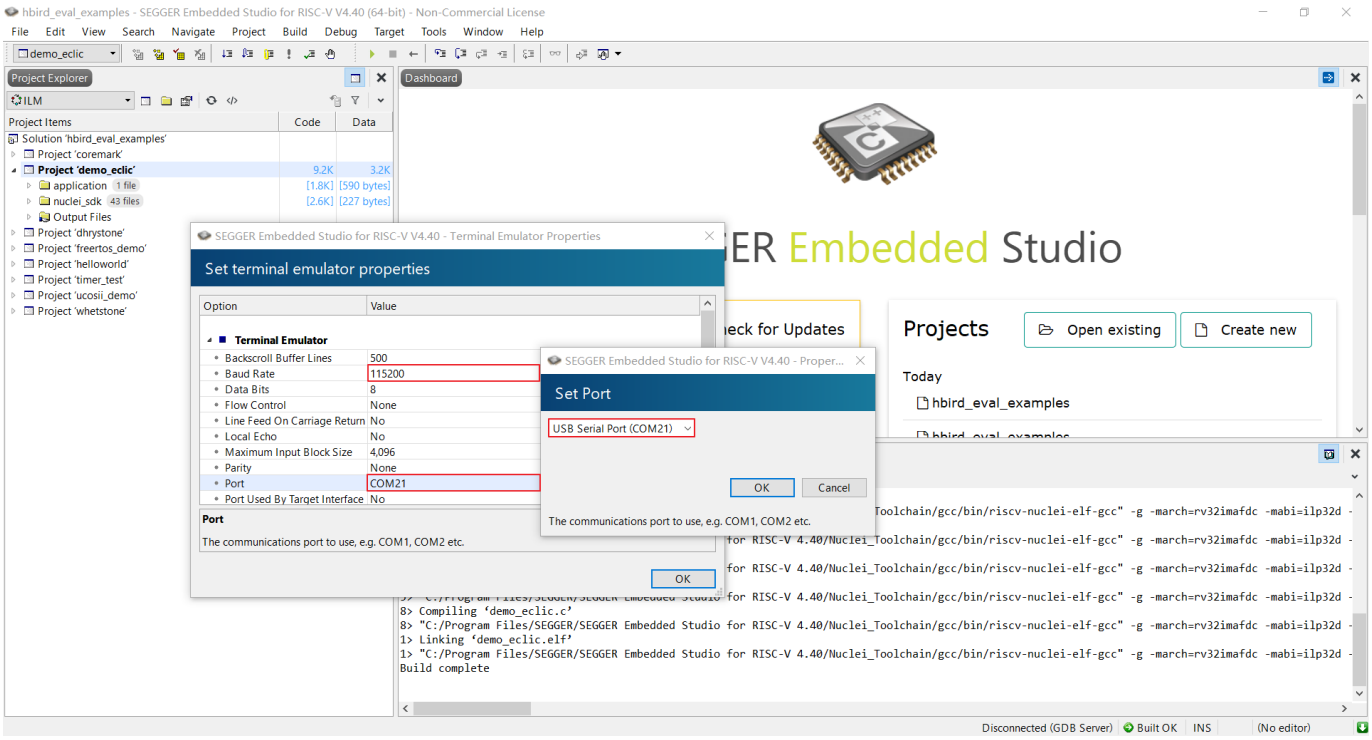

Figure 4-10 Set Baud Rate and COM Number

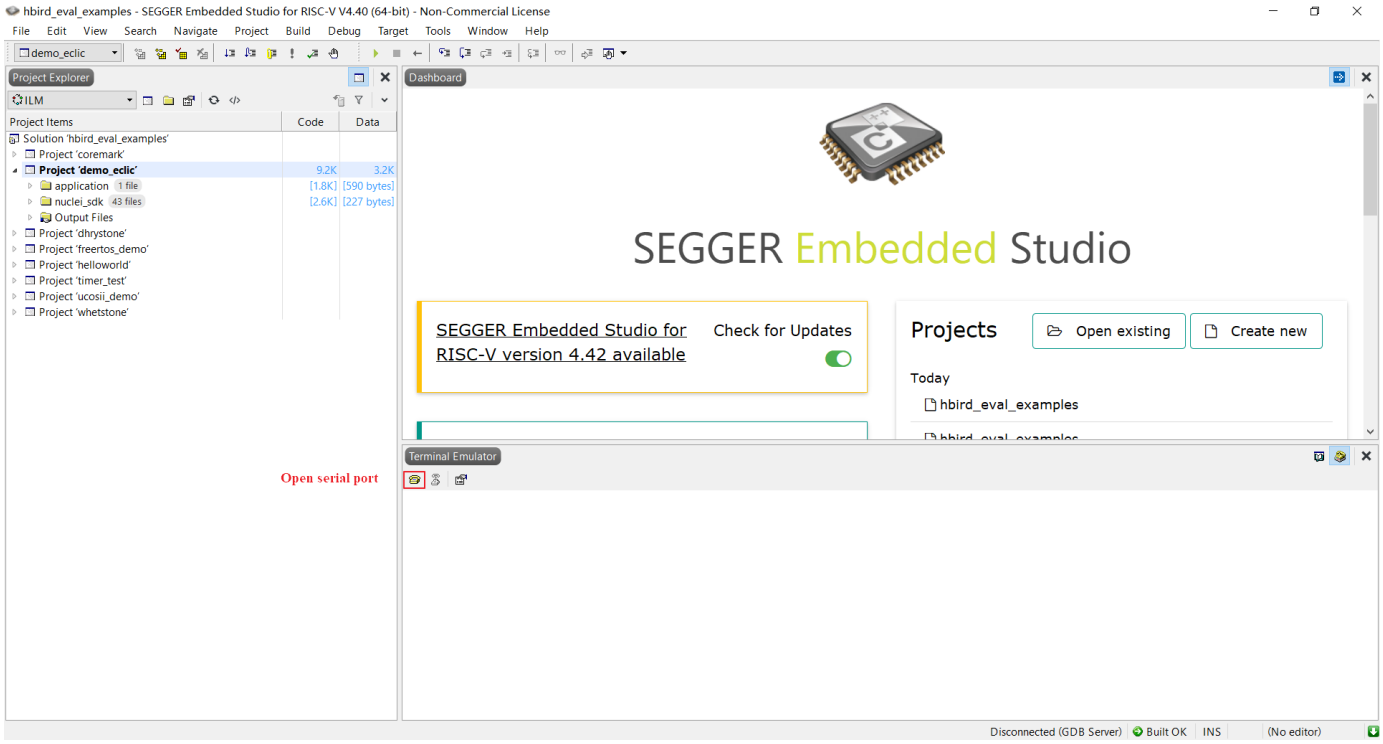

Figure 4-11 Open Serial Port

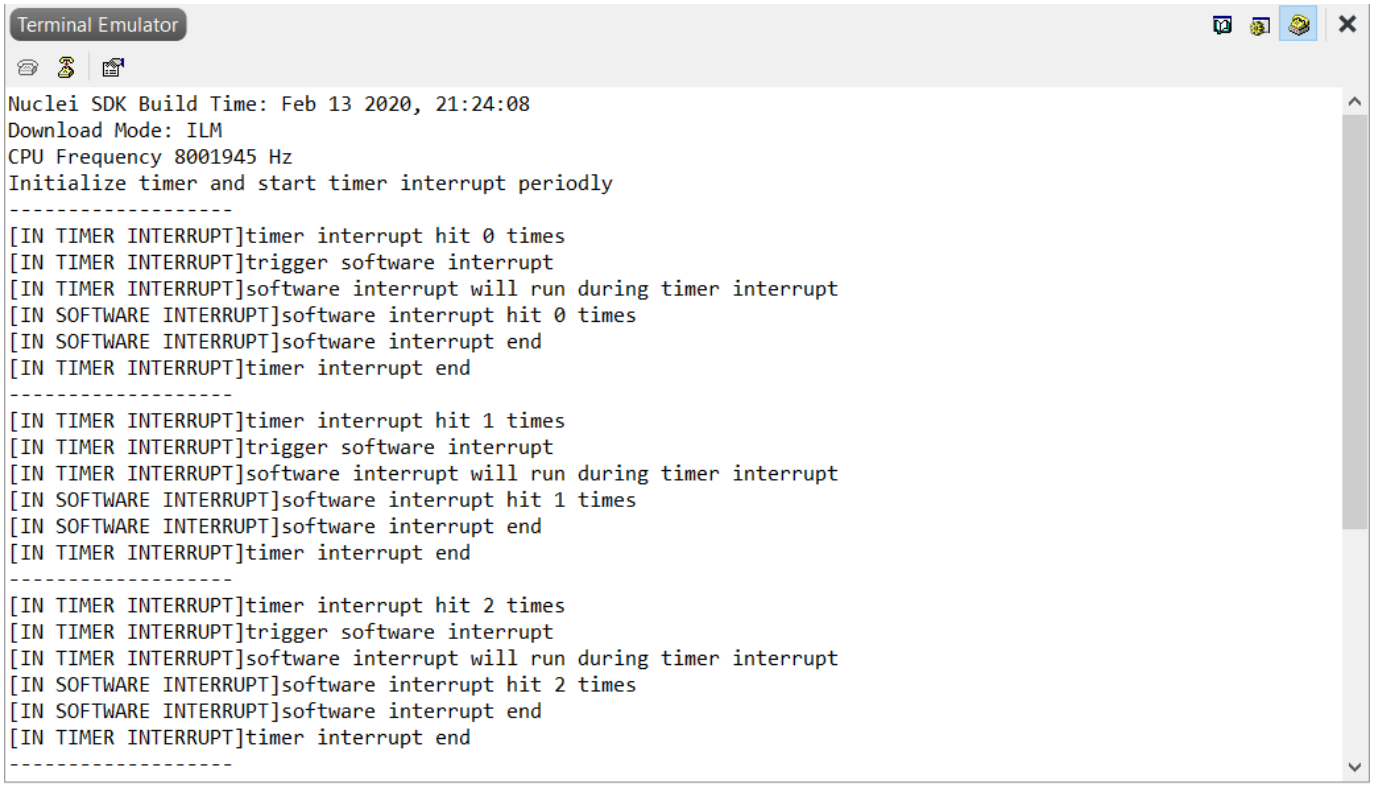

Figure 4-12 The Output of Project demo_eclic through Serial Port

4.2.2. Printout through RTT

If J-Link is used, the RTT (Real Time Terminal) of J-Link (does not need to occupy UART interface of SoC at all) can also be utilized to redirect printf output. The setting steps are as follows:

- Create an “external GNU using project” as shown in Figure 4-13. And when adding a default folder, select the SEGGER folder, as shown in Figure 4-14.
- Drag the SEGGER folder to the current project, right-click the red-box marked file in Figure 4-15, open the right-click menu, select “Excluded From Build” option.
- Recompile the project.

After that, user can redirect the output to RTT, and the debug terminal will be opened automatically at runtime. Take demo-eclic project output as an example, use J-Link RTT to output is as shown in Figure 4-16.

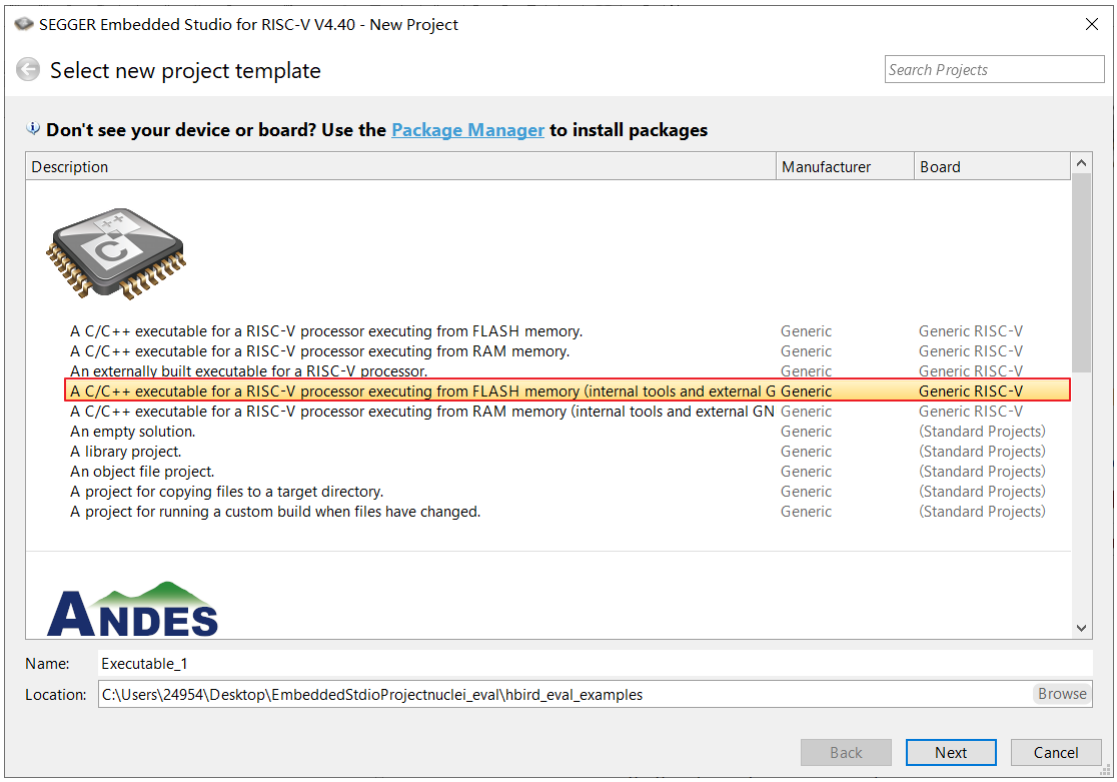

Figure 4-13 Create an External GNU Using Project

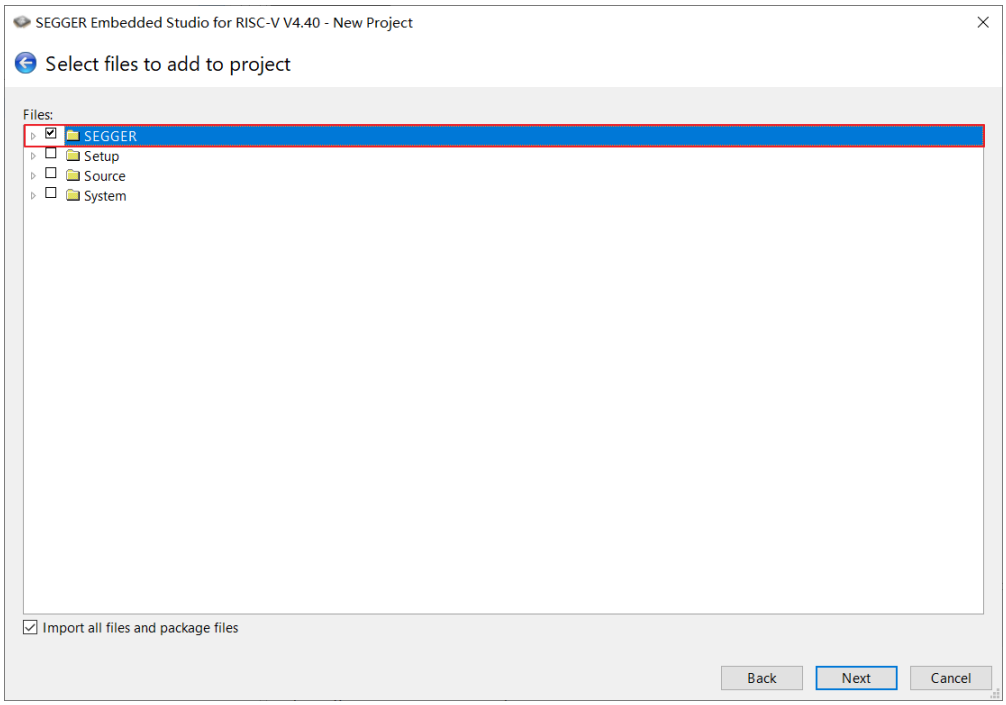

Figure 4-14 Adding SEGGER Folder

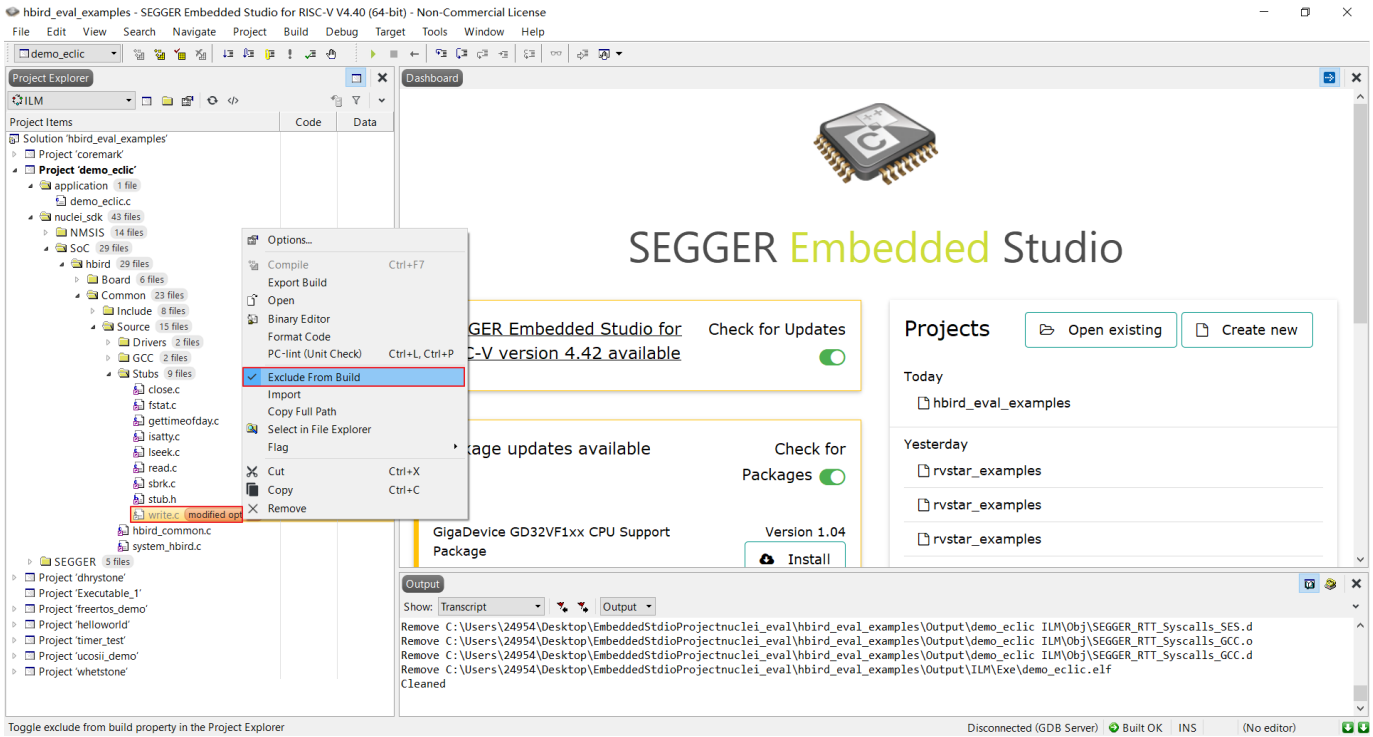

Figure 4-15 Redirect Output through RTT

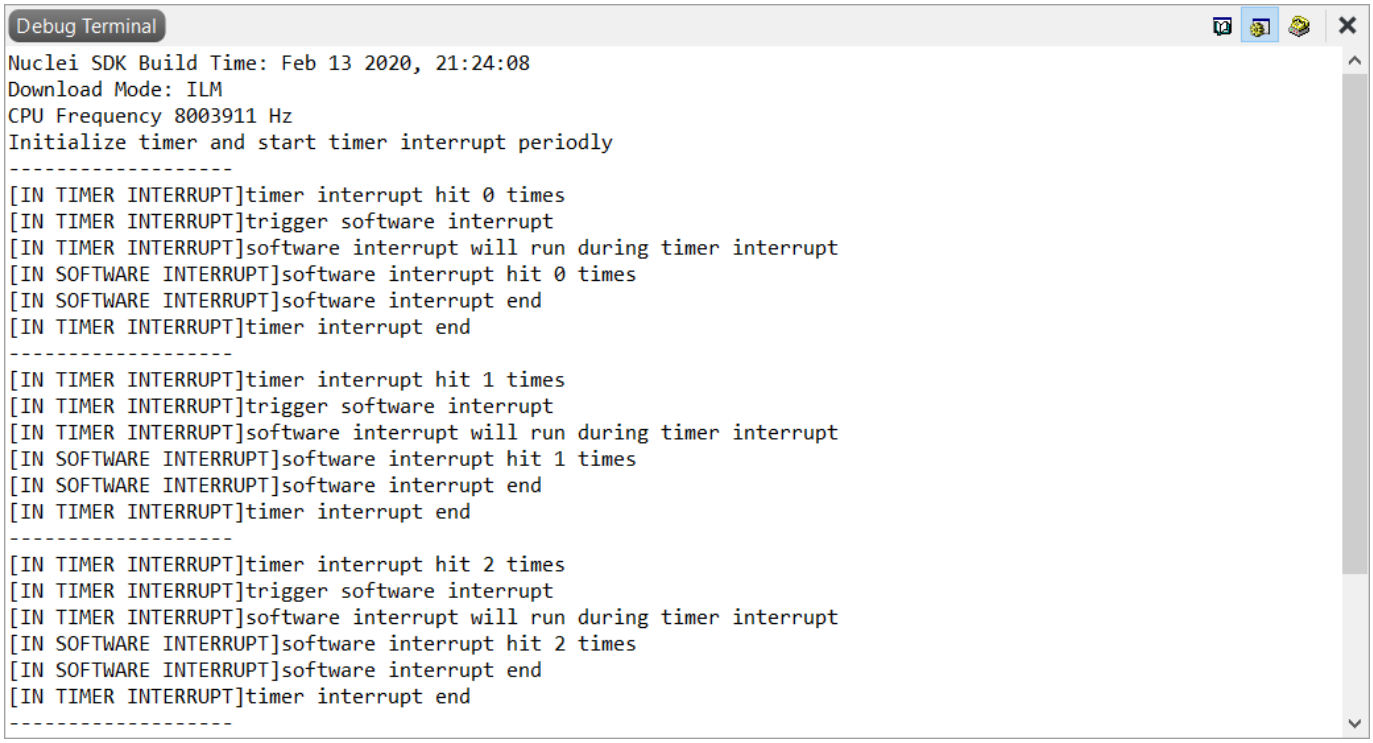

Figure 4-16 The Output of Project demo_eclic through J-Link

4.3. Download Program to Board and Run

After the above settings are completed, you can download the program to the board and run. The steps are as follows:

- Select the “Connect GDB Server” option under “Target” option in the menu bar of SES, as shown in Figure 4-17.
- Select “Download File” option under “Target” option in the menu bar, then select “Download Elf File”, as shown in Figure 4-18, find the Elf file generated by compilation (by default the Elf file is in the “Output Files” folder), and double-click to download it to the board.
  - After downloading the program to the board, you can enter the debugging mode to debug or run it. See Chapter 0 for more details of how to debug.
  - When the program running, if printf function is used, the output can be redirected to PC. Take demo-eclic as an example, you can see the output as shown in Figure 4-12 (Printout through Serial Port) or Figure 4-16 (Printout through RTT).
- If user don’t want to debug or re-download it again, then user can just disconnect GDB Server after downloading, select "Target --> Disconnect GDB Server" in the menu bar, as shown in Figure 4-19.
  - Note: after disconnecting the GDB Server, press the MCU reset button on Hummingbird Evaluation Kit, and the Processor Core will start to execute again from the flash. This is because in the Nuclei Evaluation SoC, the reset address of Procesor Core is the starting address of QSPI0 Flash. Please refer to document <Nuclei_Eval_SoC_Intro.pdf> for more information of the Nuclei Evaluation SoC.

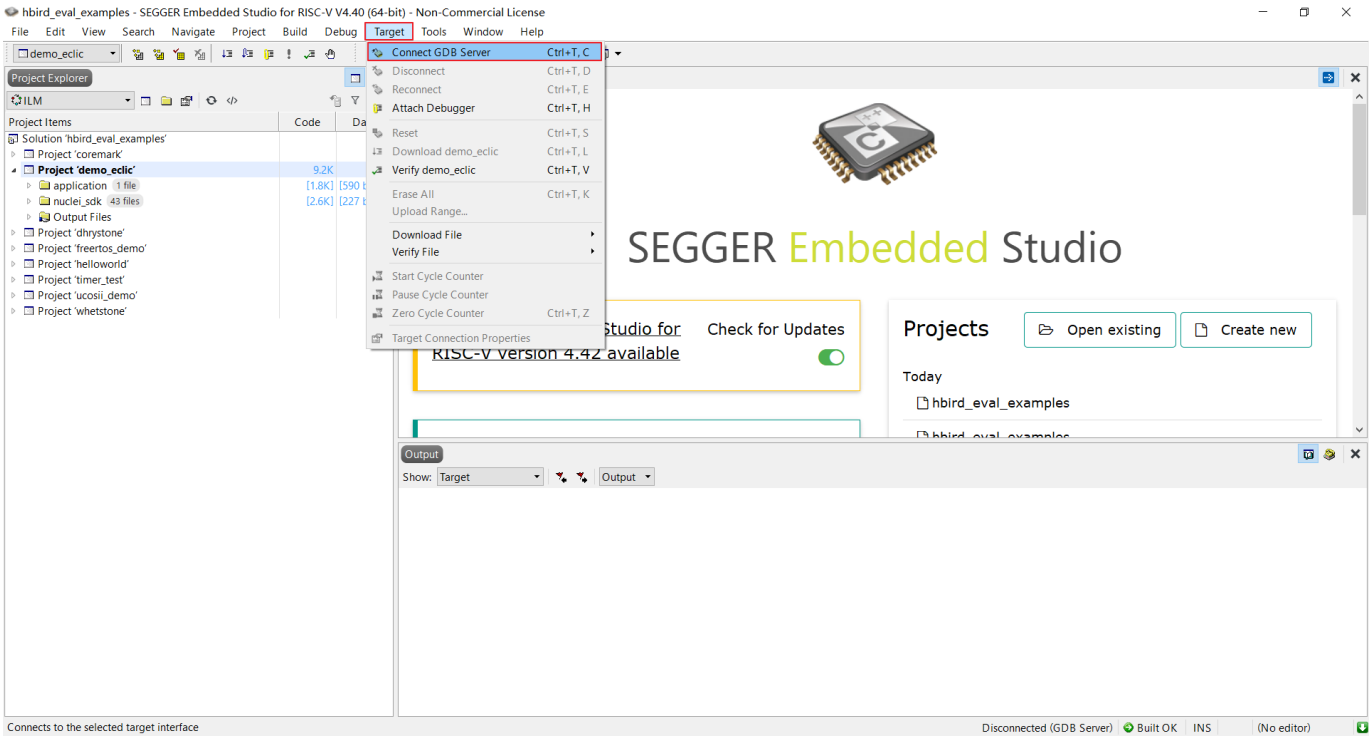

Figure 4-17 Connect GDB

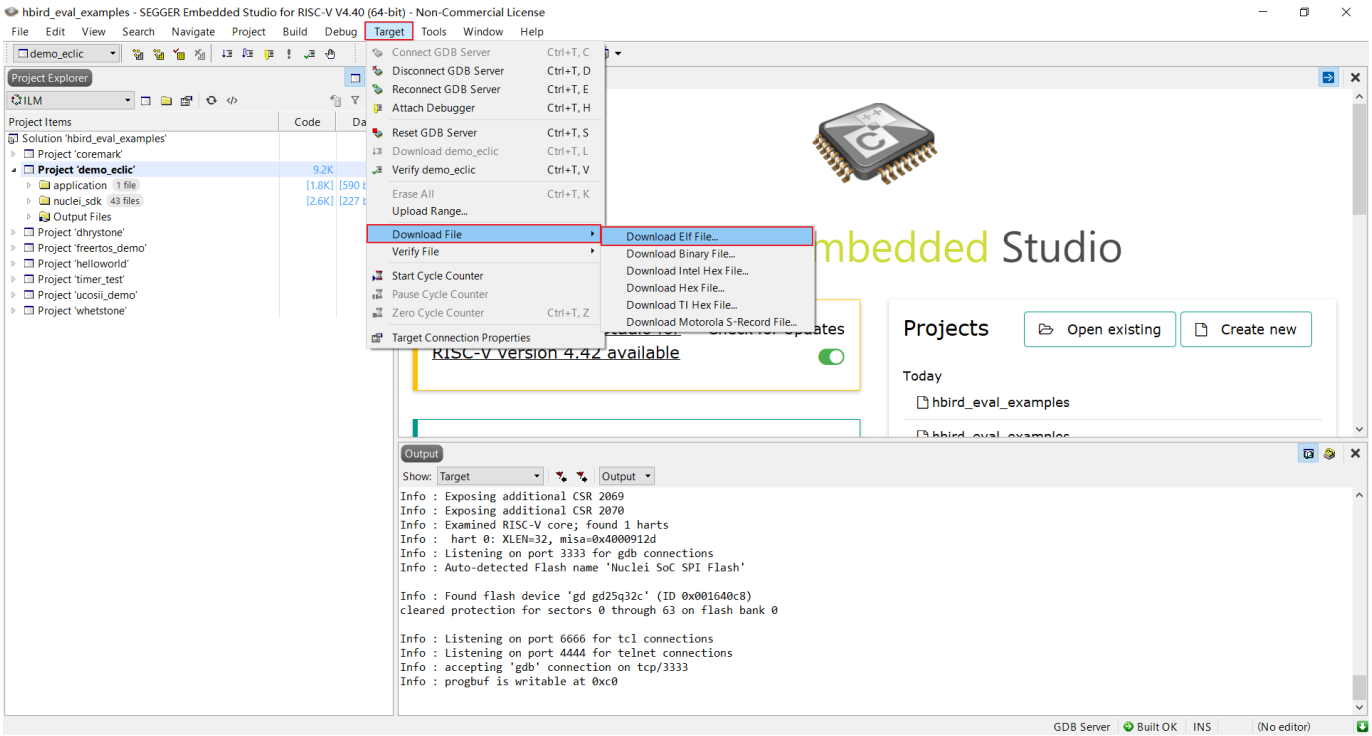

Figure 4-18 Download Elf File

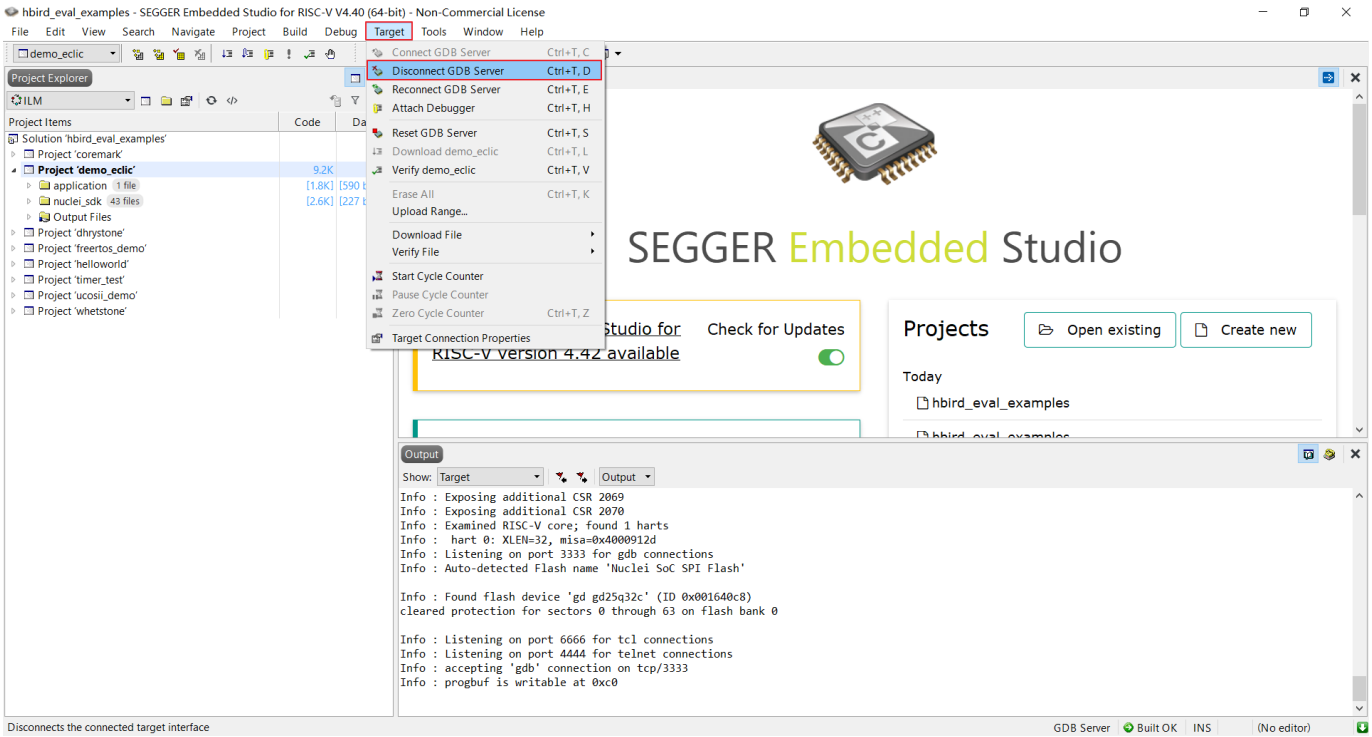

Figure 4-19 Disconnect GDB Server

5. Debug Project

There are many debugging features of SES. Please refer to the following website for detailed usage of debugging:

- SEGGER WIKI: https://wiki.segger.com/Embedded_Studio
- SES manual: https://www.segger.com/downloads/embedded-studio

To enter the debugging mode, press F5 directly or select the “GO” option of “Debug” option in the menu bar. The user interface after successfully entering the debugging mode is as shown in Figure 5-1.

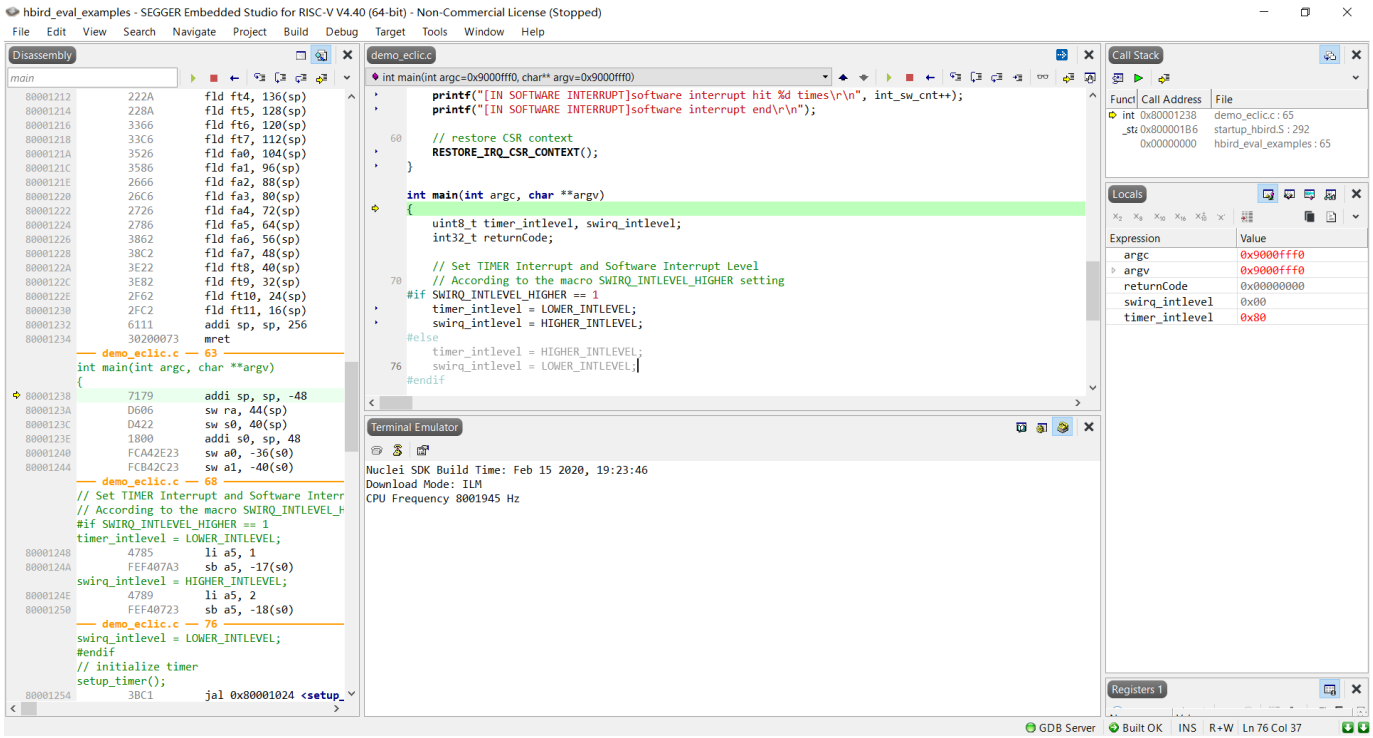

Figure 5-1 Enter Debugging Mode