Lattice Propel 2023.2 SDK

User Guide

FPGA-UG-02195-1.0

November 2023

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults, and all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Inclusive Language

This document was created consistent with Lattice Semiconductor’s inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice’s inclusive language FAQ 6878 for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Contents

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Figures

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Glossary

A glossary of terms used in this document.

Terms	Definition
ASCII	American Standard Code for Information Interchange.
BSP	Board Support Package, the layer of software containing hardware-specific drivers and libraries to function in a particular hardware environment.
DUT	Design Under Test.
CDT	C/C++ Development Tools.
CPU	Central Processing Unit.
GUI	Graphical User Interface.
FPGA	Field Programmable Gate Array.
HDL	Hardware description language.
IBIS	Input Output Buffer Information System.
IDE	Integrated Development Environment.
IP	Intellectual Property.
JEDEC	Joint Electron Device Engineering Council.
OCD	On-Chip-Debugging.
OEM	Original Equipment Manufacturer.
OpenOCD	Open On-Chip Debugger.
Perspective	A group of views and editors in the Workbench window.
Programmer	A tool can program Lattice FPGA SRAM and external SPI Flash through various interfaces, such as JTAG, SPI, and I^2C.
RISC-V	Reduced Instruction Set Computer-V. A free and open instruction set architecture (ISA) enabling a new era of processor innovation through open standard collaboration.
RISC-V MC	Lattice RISC-V for Micro-Controller Soft IP.
RISC-V SM	Lattice RISC-V for State-Machine Soft IP.
RISC-V RX	Lattice RISC-V for RTOS Soft IP.
SDK	Software Development Kit. A set of software development tools that allows the creation of applications for software package on the Lattice embedded platform.
SoC	System-on-Chip. An integrated circuit that integrates all components of a computer or other electronic systems.
SRAM	Static Random Access Memory.
UART	Universal Asynchronous Receiver/Transmitter.
UI	User Interface.
UFM	User Flash Memory.
VHDL	Very-High-Speed Integrated Circuit Hardware Description Language.
Workspace	The directory where stores your work, it is used as the default content area for your projects as well as for holding any required metadata.
Workbench	The desktop development environment in Eclipse IDE platform.

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

1. Introduction

Lattice Propel™ is a complete set of graphical and command-line tools to create, analyze, compile, and debug both FPGA-based hardware and software processor systems.

1.1. Purpose

Embedded system solutions play an important role in FPGA system design, allowing you to develop software for a processor in an FPGA device. It provides the flexibility for you to control various peripherals from a system bus.

To develop an embedded system on an FPGA, you need to design the System-on-Chip (SoC) with an embedded processor and develop system software on the processor. Lattice Propel helps you develop your system with a RISC-V processor, peripheral IP, and a set of tools.

The purpose of this document is to introduce Lattice Propel SDK tool and flow to help you quickly get started to build a small demo system. You can find recommended flows of using Lattice Propel SDK in this document as well.

1.2. Audience

The intended audience for this document includes embedded system designers and embedded software developers using Lattice FPGA devices. The complete list of supported devices can be found in Lattice Propel Release Notes. The technical guidelines assume readers have expertise in the embedded system area and FPGA technologies.

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

2. Lattice Propel Development Suite

Lattice Propel development suite includes:

- an integrated development environment (IDE), which is the framework of Propel;
- Lattice Propel Builder, which is for SoC design;
- Lattice Propel SDK, which is for system software development.

2.1. Eclipse IDE

Eclipse IDE provides the Propel development suite a platform to manage the SoC project and the Embedded C/C++ Project in the same workspace.

The SoC project, which extends from the Propel Builder project, provides easy interaction with other Lattice design tools, such as Lattice Diamond™ within Propel.

The Embedded C/C++ Project provides a platform for developing or debugging application code within Eclipse IDE. The project can be created directly from the SoC project with a pre-set Board Support Package (BSP) and applications by using Propel development suite.

2.2. Lattice Propel Builder

Lattice Propel Builder allows you to assemble the larger functional blocks of the design hierarchy. Propel Builder enables you to instantiate modules and IP from the IP Catalog in a schematic view, and can easily connect the modules. Propel Builder also helps you customize address spaces within modules, such as a processor. In Propel development suite, Propel Builder is used to create a microprocessor integrated platform for both hardware and software development.

Refer to Lattice Propel 2023.2 Builder User Guide (FPGA-UG-02196) for more detailed information.

2.3. Lattice Propel SDK

Lattice Propel SDK is based on Eclipse Embedded C/C++ Development Tools (CDT). It allows you to create, build, and debug software application projects that drive the platform within the Eclipse framework.

The main features are:

- Create, build, debug, or manage embedded applications for Lattice RISC-V CPU/SoC solution.
- Provide extra build steps to generate the binary and memory files required for deployment.
- Build using the latest industry standard open source components and tools for RISC-V firmware development and debugging.
- Support Picolibc for RISC-V, and provide lightweight standard library implementation.
- Provide fully-configurable toolchain definitions.

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

3. Lattice Propel Tool Flows

The Propel tool flows including SoC project design flow, C/C++ project design flow, system simulation flow, and programming and On-Chip-Debugging (OCD) debugging flow, are discussed in detail in the following sections.

3.1. Propel Environment

3.1.1. Running Lattice Propel

After installing Lattice Propel, you can launch Propel from the desktop shortcut icon or from the Windows Start menu. When Lattice Propel is invoked, a dialog (Figure 3.1) pops up. You can browse to select where to locate the workspace. For normal needs, just click **Launch** to pick the default location and continue running Lattice Propel.

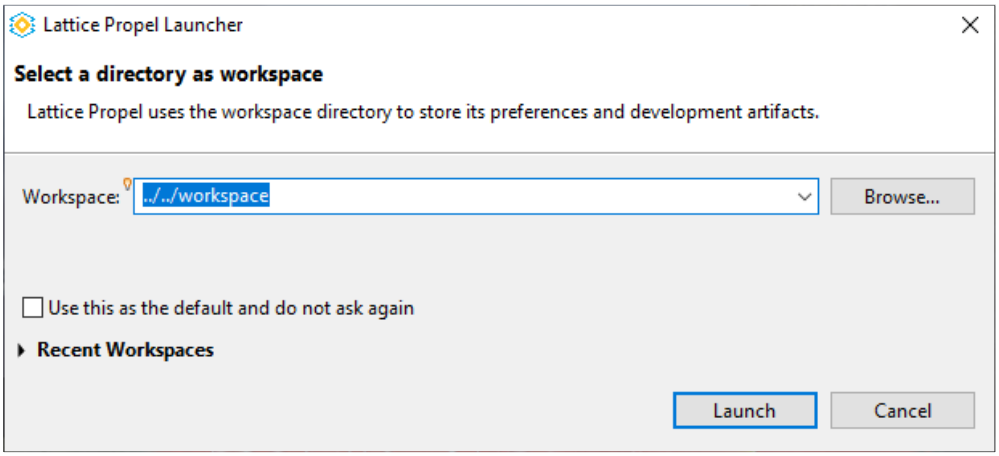


Figure 3.1. Select Workspace Dialog

After the workspace location is chosen, a single workbench window is displayed using default Propel SDK perspective. The default Propel SDK perspective contains the following five functional areas (Figure 3.2).

1. Menu bar and Toolbar, including: File menu, Edit menu, Source menu, Refactor menu, Navigate menu, Search menu, Project menu, Run menu, LatticeTools menu, Window menu, and Help menu.
2. Project Explorer view: displays projects in the workspace.
3. Editor view: provides capability of editing source files.
4. Outline view: displays an outline of a file that is currently open in the editor area.
5. Log area includes these views: Problem view, Tasks view, Console view, Properties view, and Terminal view.

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

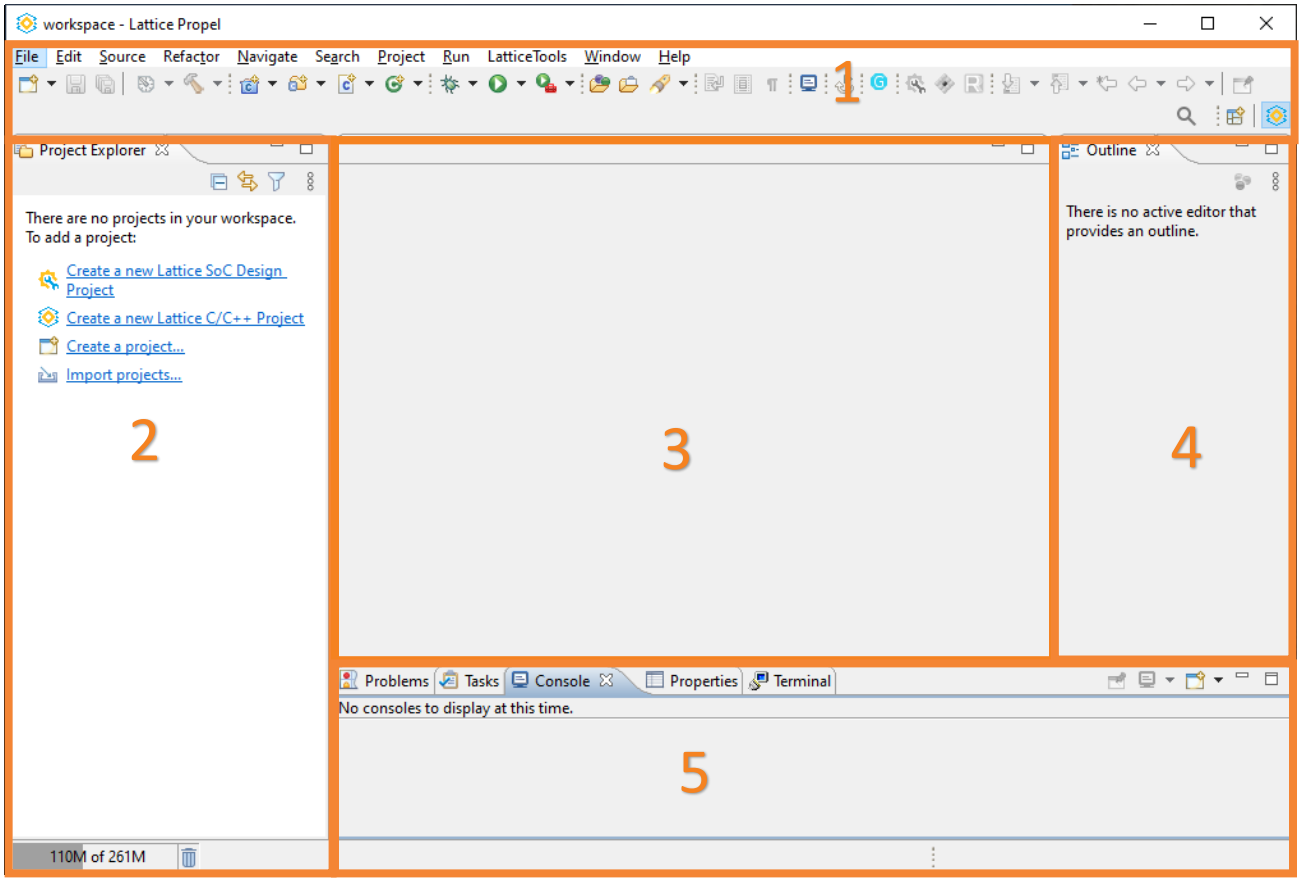


Figure 3.2. Propel Workbench Window

3.1.2. Importing General Projects

In Propel SDK, you can use the importing project wizard to import existing projects into workspace except for Lattice SoC design created by Lattice Propel Builder.

Note: You can only import projects which are created or managed by Propel SDK by choosing **General > Existing Project into Workspace** in Propel SDK.

1. In Lattice Propel (SDK), choose **File** > **Import…**.

   The **Select** wizard opens (Figure 3.3).

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

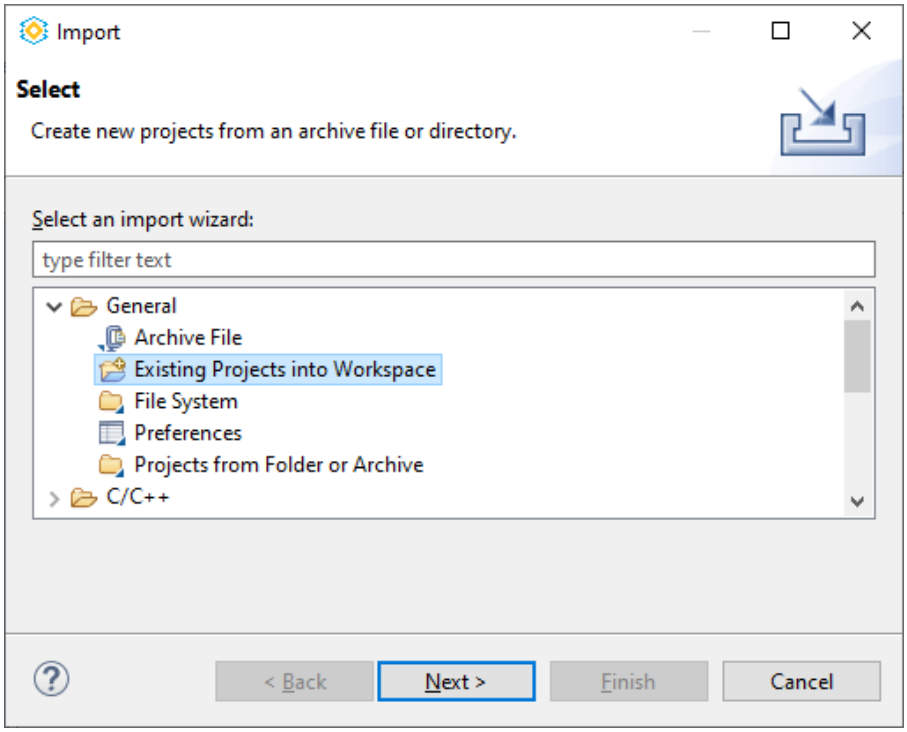


Figure 3.3. Import Projects Wizard — General

2. Select **General** > **Existing Project into Workspace**. Click **Next**.
   The Import wizard switches to **Import Projects** wizard (Figure 3.4).

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

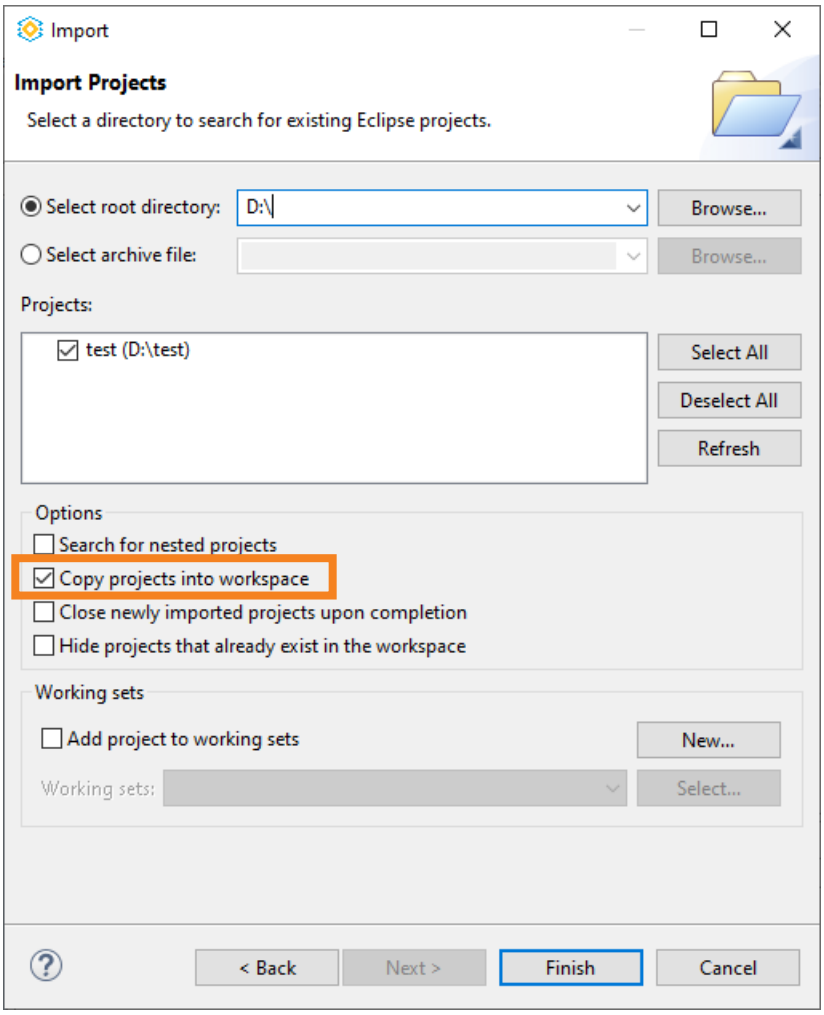


Figure 3.4. Import Existing Projects Wizard

3. Choose either **Select root directory** or **Select archive file**, clicking the associated **Browse** button to locate the directory or file containing the project.
4. **Select** the project or projects you want to import in the **Projects** area.
5. Be sure to check the **Copy projects into workspace** option so that your original projects do not get updated.
6. Click **Finish** to start the importing process.

3.1.3. Importing Lattice SoC Design Projects

In Propel SDK, you can use the Import Wizard to import Lattice SoC design projects into workspace. Existing SoC design projects created by either Propel SDK or Propel Builder can also be imported into Workspace by choosing **Lattice Propel > Lattice SoC Design Projects**.

1. From Lattice Propel SDK, choose **File** > **Import…**.

   The **Select** wizard opens (Figure 3.5).

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

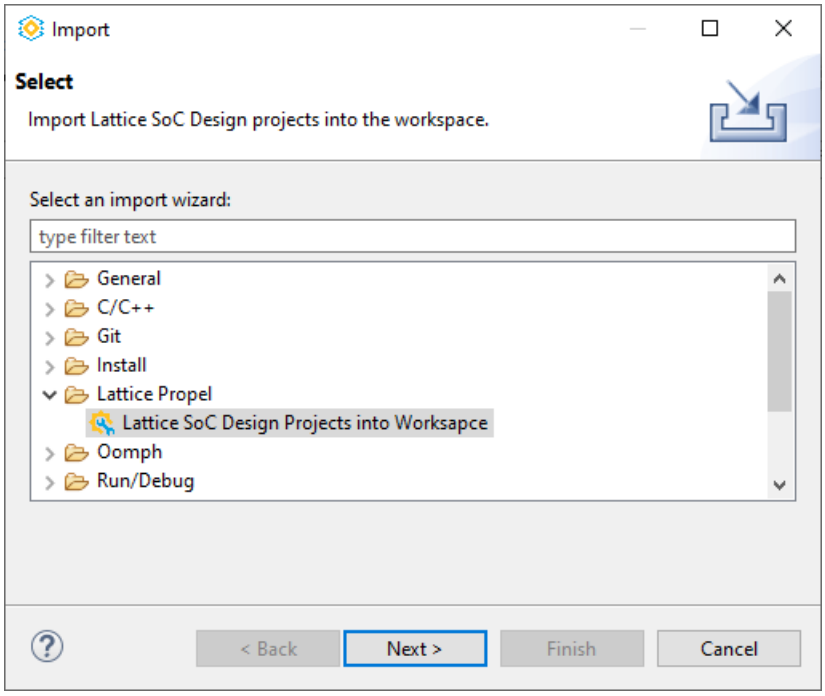


Figure 3.5. Import Projects Wizard – Lattice Propel

2. Select **Lattice Propel > Lattice SoC Design Projects into Workspace**. Click **Next**.
   The **Select** wizard switches to **Import Lattice SoC Design Projects** wizard page (Figure 3.6).

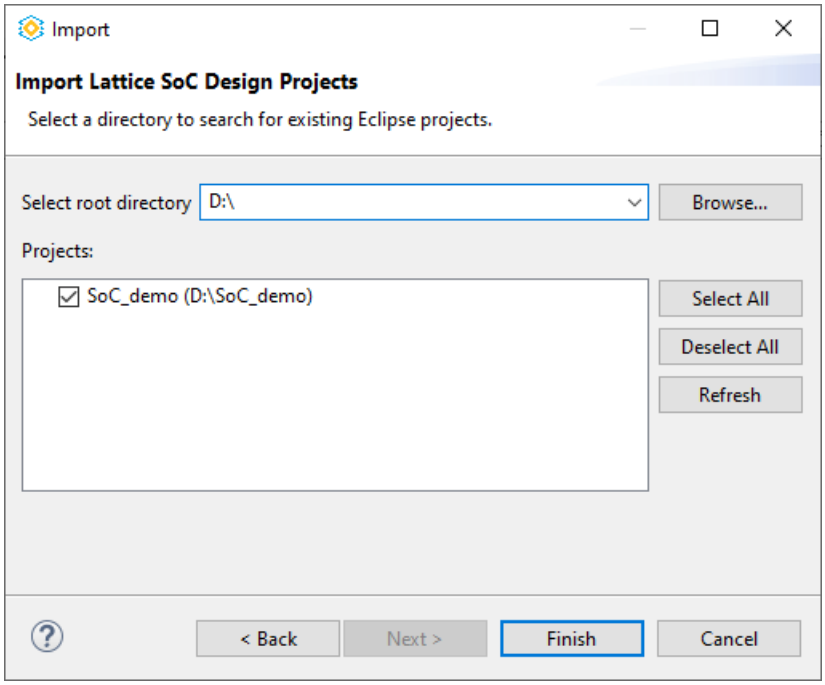


Figure 3.6. Import Lattice SoC Design Projects Wizard

3. Locate the directory containing the projects by clicking the **Browse** button.
4. In **Projects** area, select the SoC design project or projects you want to import.
5. Click **Finish** to start the importing process.

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

3.2. SoC Project Design Flow

A new SoC design project including a Propel Builder design can be started from the Lattice Propel sets. Follow the steps below to create a new SoC design project.

3.2.1. Creating an SoC Design Project

To start a Lattice SoC Design Project from Propel:

1. In Lattice Propel, choose **File** > **New** > **Lattice SoC Design Project**.

   The **Create SoC Project** wizard opens (Figure 3.7). In the **Create SoC Project** wizard, you can specify a device or a board for a Template SoC project.

   - To specify a device for your new Template SoC project, use the drop-down menu to select desired device information, including **Processor**, **Family**, **Device**, **Package**, **Speed**, and **Condition**. Also, select RISC-V SoC Project or Empty Project in the **Template Design** field (Figure 3.7).

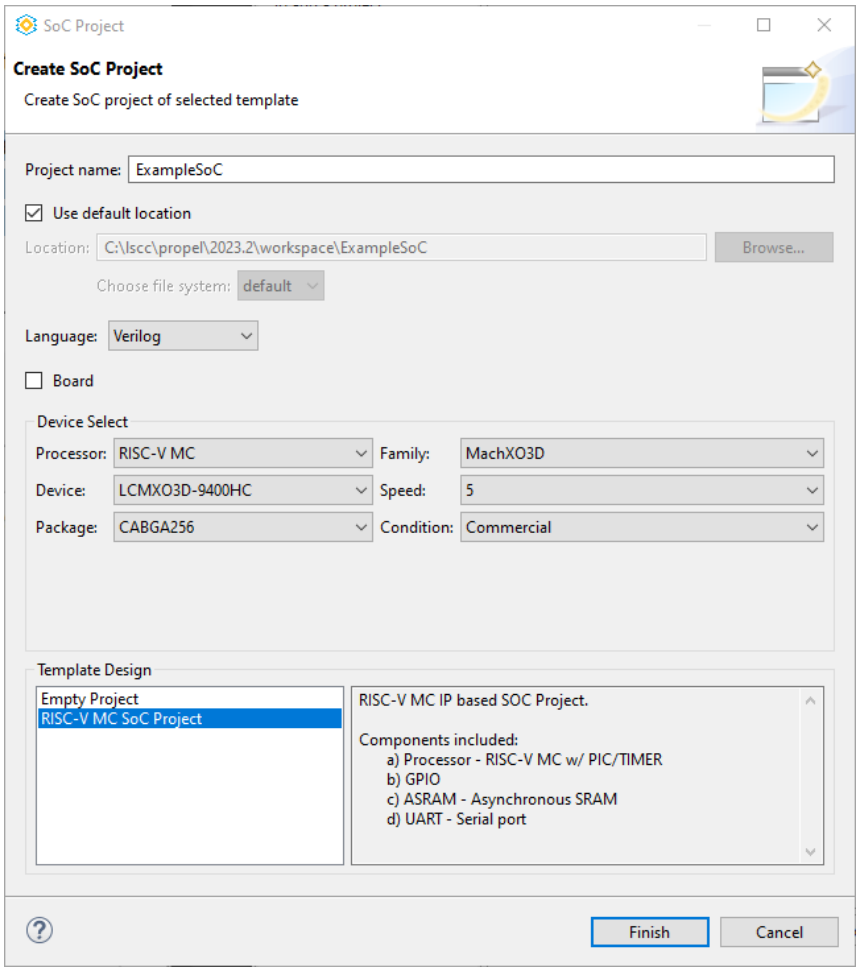


Figure 3.7. Specify a Device for Template SoC Project

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

- Or, to specify a board for a new Template SoC project, check the **Board** checkbox (Figure 3.8).

  Note: You can choose VHDL/Verilog in the **Language** field.

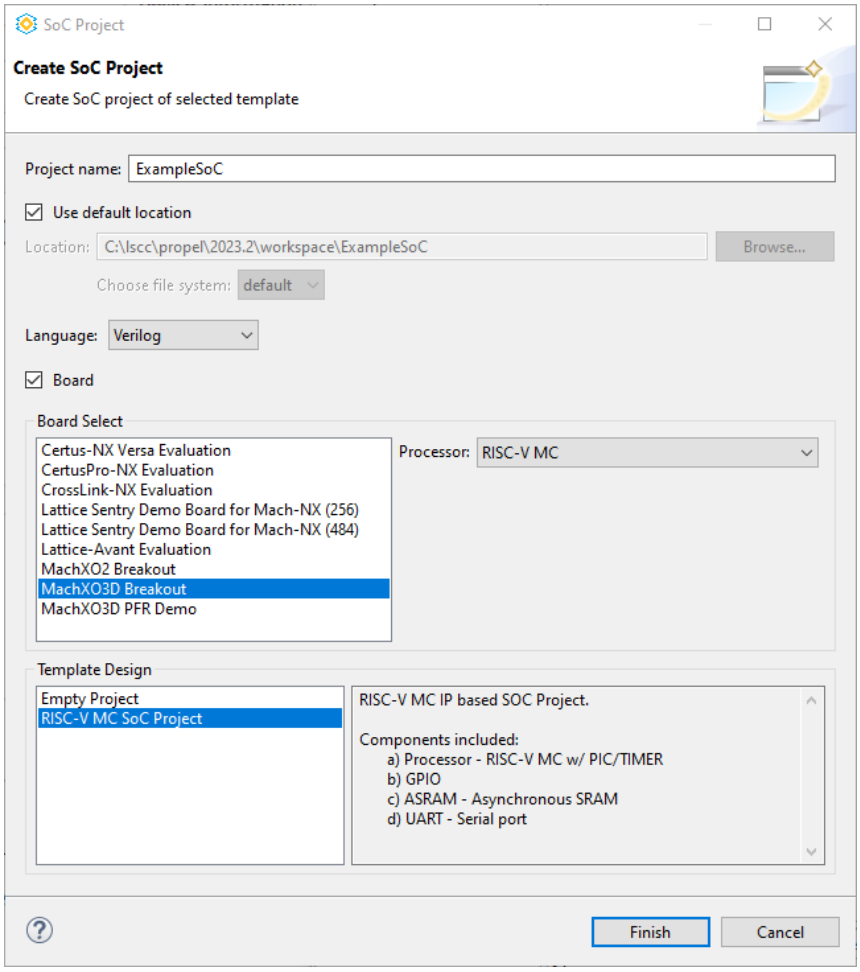


Figure 3.8. Specify a Board for Template SoC Project

2. From the **Board Select** area, select the desired board, such as MachXO3D Breakout.
3. Enter a project name.

   Note: Do not include periods, colons, or spaces in the project name.
4. (Optional) To change the default location, clear the **Use default location** option, then browse for another location. Choose a file system.
5. Select a desired platform template design. In particular, select empty project for building system from scratch.
6. Click **Finish**.

   The SoC design project is created in workbench, and its design is opened and displayed in Lattice Propel Builder (Figure 3.11).

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

3.2.2. Open an SoC Design in Propel Builder

Within an SoC project, there is a Propel Builder design.

To open Propel Builder for an SoC project:

1. In the **Project Explorer** view, select an SoC project.
2. Open the SoC project in one of the following ways from Lattice Propel:
   - Choose **LatticeTools >** **Open Design in Propel Builder** (Figure 3.9).

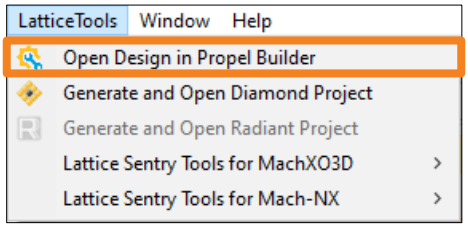


Figure 3.9. LatticeTools Menu

   - Click the **Propel Builder** icon on the toolbar.
   - Right-click the SoC project from **Project Explorer**. Choose **Open Design In** > **Propel Builder** from the popup menu (Figure 3.10).

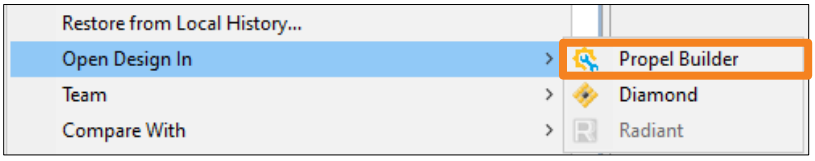


Figure 3.10. Project Explorer Popup Menu

3. The SoC Design is opened and displayed in Propel Builder (Figure 3.11).

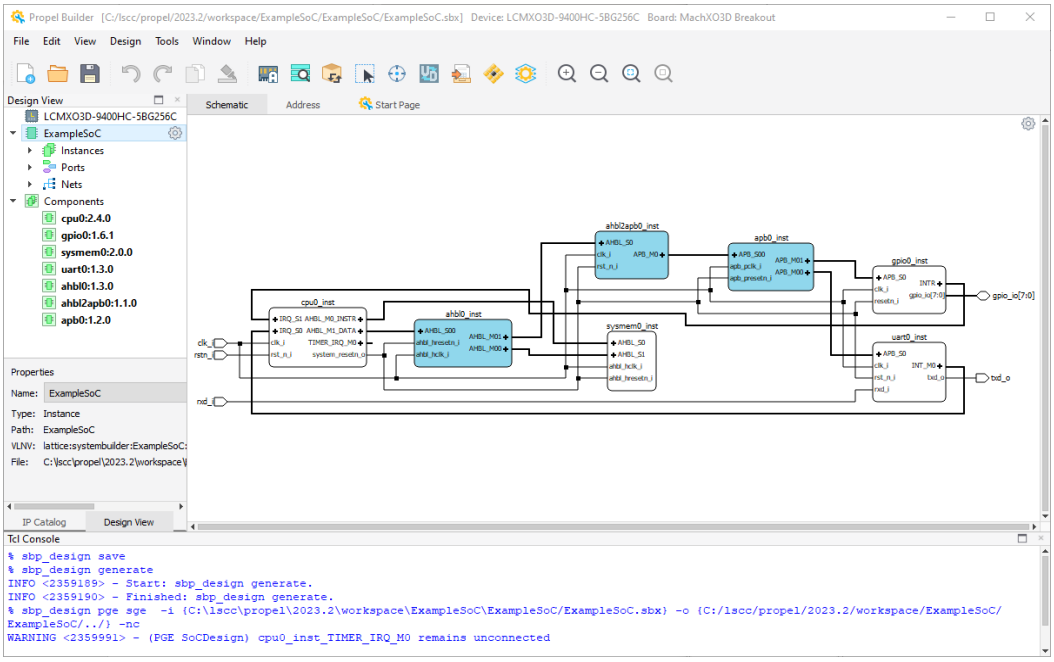


Figure 3.11. Propel Builder Window

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

4. (Optional) Modify the design in Propel Builder as desired. Most of the templates include a functional-ready SoC design.

   Note: You can only create an SoC design using **Empty Project** template inside the Propel Builder. Refer to Lattice Propel Builder 2023.2 User Guide (FPGA-UG-02196) for more details on how to create an SoC design using **Empty Project** template.

3.2.3. Open Design in Lattice FPGA Design Software

Within an SoC project, you can create a Lattice FPGA design project including a Propel Builder design, and then open the FPGA design project in appropriate software. There are two FPGA Design Software available, Lattice Diamond and Lattice Radiant™. Depending on the device family used in the SoC project, only one of the FPGA Design software can be selected from the User Interface (UI), the other is grayed out. If the MachXO3D or LFMNX device family is used, the Lattice Diamond related menu items are active from Lattice Propel UI. If the LIFCL or LFD2NX device family is used, the Lattice Radiant related menu items are active from Lattice Propel UI.

To open FPGA Design Software for an SoC project from Lattice Propel:

1. (Optional) Set Lattice FPGA design software installation location from Lattice Propel. By default, Lattice Propel can find the proper Lattice FPGA design software installation location, usually the latest version installed on the PC. You can overwrite it following steps below.

   Choose **Window** > **Preferences**. The **Preferences** dialog opens (Figure 3.12).

   Select **Propel Setting** from the left pane. Click the **Browse** button to pick up the installation location of Diamond or Radiant. Or, leave the **Radiant Location** and **Diamond Location** fields blank, as default. Lattice Propel can find the location automatically.

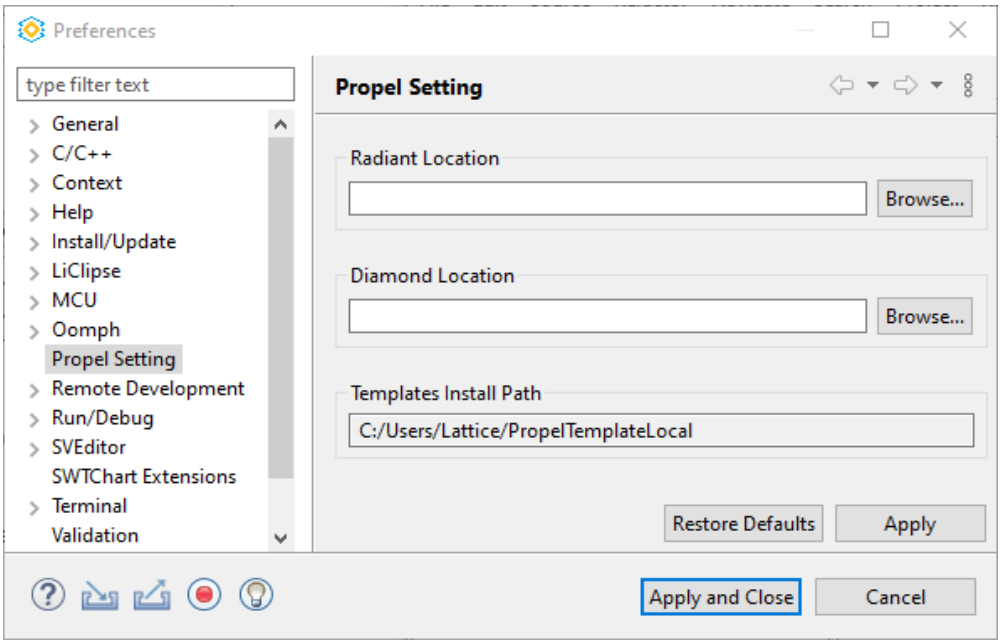


Figure 3.12. Propel Preferences Dialog

2. In the **Project Explorer** view from the Propel main Graphical User Interface (GUI), select an SoC project.
3. Open the SoC project in one of the following ways:
   - Choose **LatticeTools >** **Generate and Open Diamond Project**; Or, choose **LatticeTools >** **Generate and Open Radiant Project**.
   - Click the **Diamond** icon or the **Radiant** icon from the toolbar.

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

- Right-click an SoC project from the **Project Explorer**. Choose **Open Design In** > **Diamond**. Or, choose **Open Design In** > **Radiant** from the right-click menu.

4. The Diamond/Radiant project for SoC is generated at background and is launched (Figure 3.13/Figure 3.14).

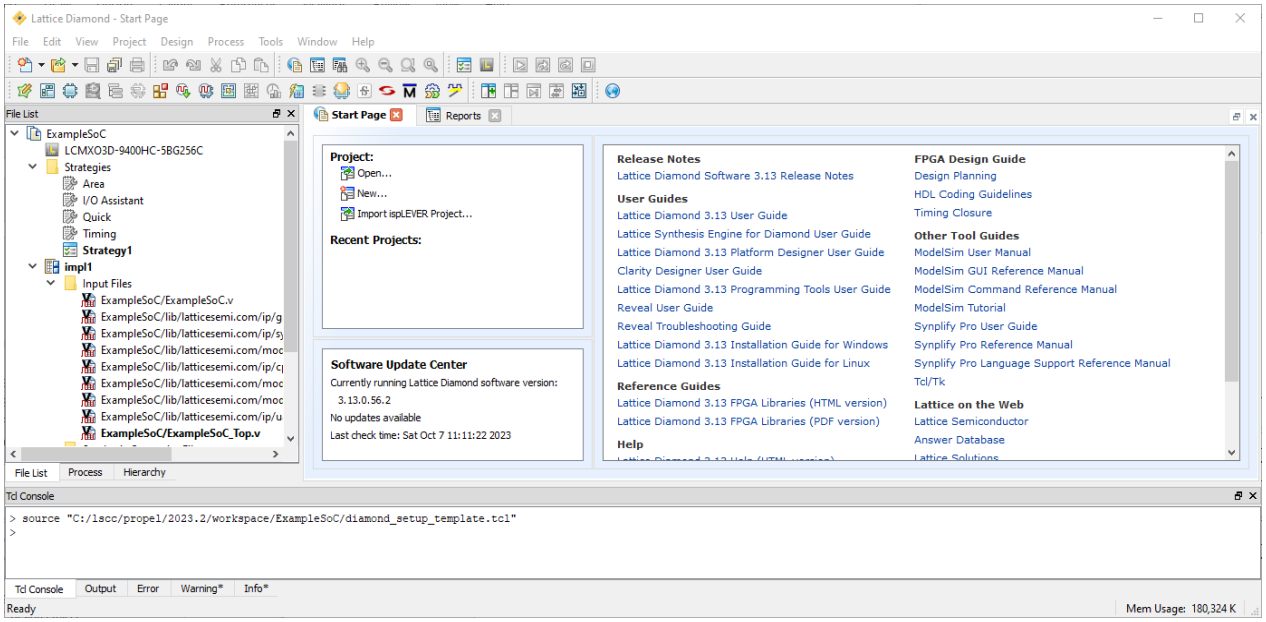


Figure 3.13. Diamond Project

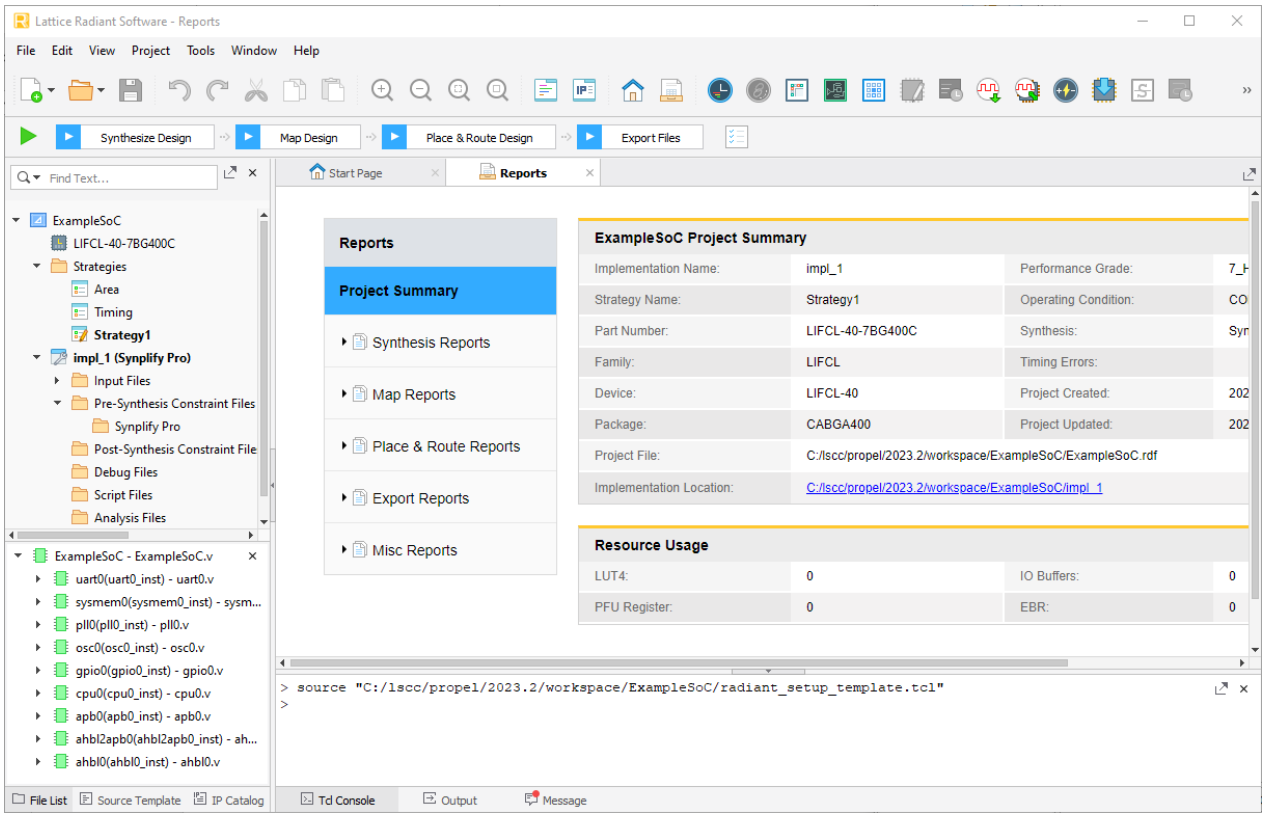


Figure 3.14. Radiant Project

5. (Optional) From the **File List** view of the Diamond/Radiant software:

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

- modify the top-level RTL file (<*proj_name*>_Top.v) to match the SoC design, presupposition of which is that there is a top-level RTL file in your SoC design; or
- create a top-level RTL file (<*proj_name*>_Top.v) to match the SoC design, if the SoC design is created from an Empty Project template and there is no top-level RTL file in your SoC design.

6. (Optional) Modify constraint file (<*proj_name*>.lpf/<*proj_name*>.pdc) to match the SoC design, if you have modified the SoC design.

   Note: This step is a must for the SoC design created from the Empty Project template.

7. Process the design in Lattice Diamond/Radiant.

   In Diamond software, switch to **Process** view of the project (Figure 3.15). Make sure at least one file, IBIS Model, Verilog Simulation File, VHDL Simulation File, Bitstream File, or JEDEC file, is checked in the **Export Files** section for programming. Choose **Process >** **Run**.

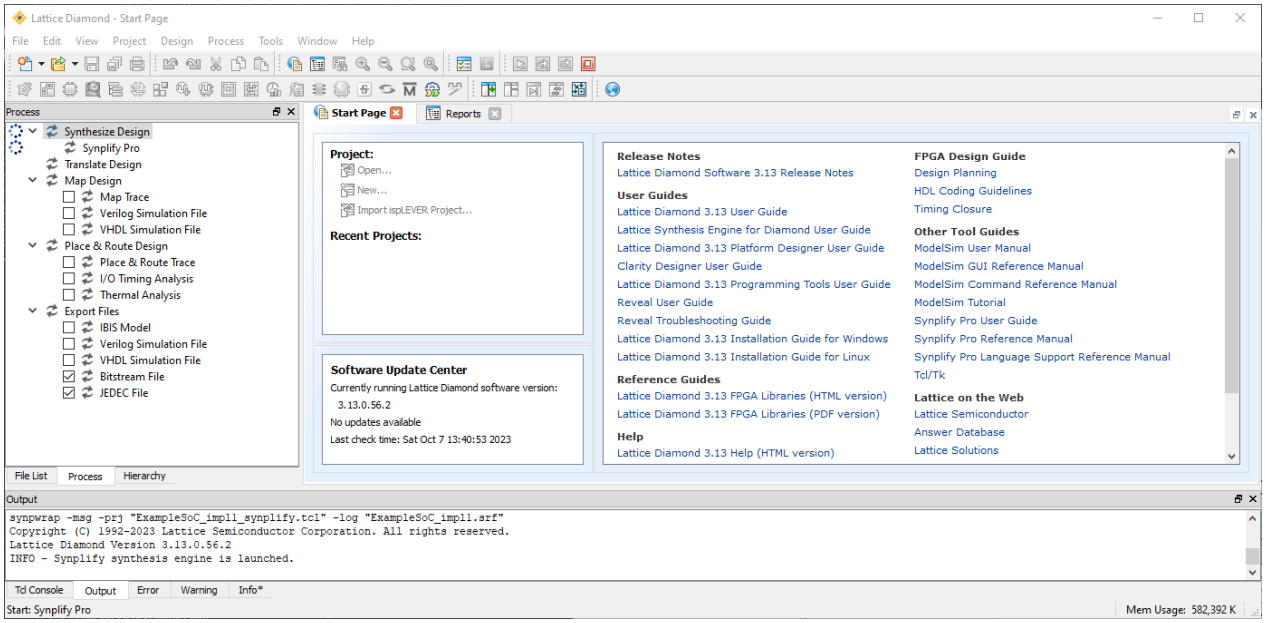


Figure 3.15. Generate Programming File in Lattice Diamond

In Lattice Radiant software, from the **Process** Toolbar, click **Export Files** (Figure 3.16).

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

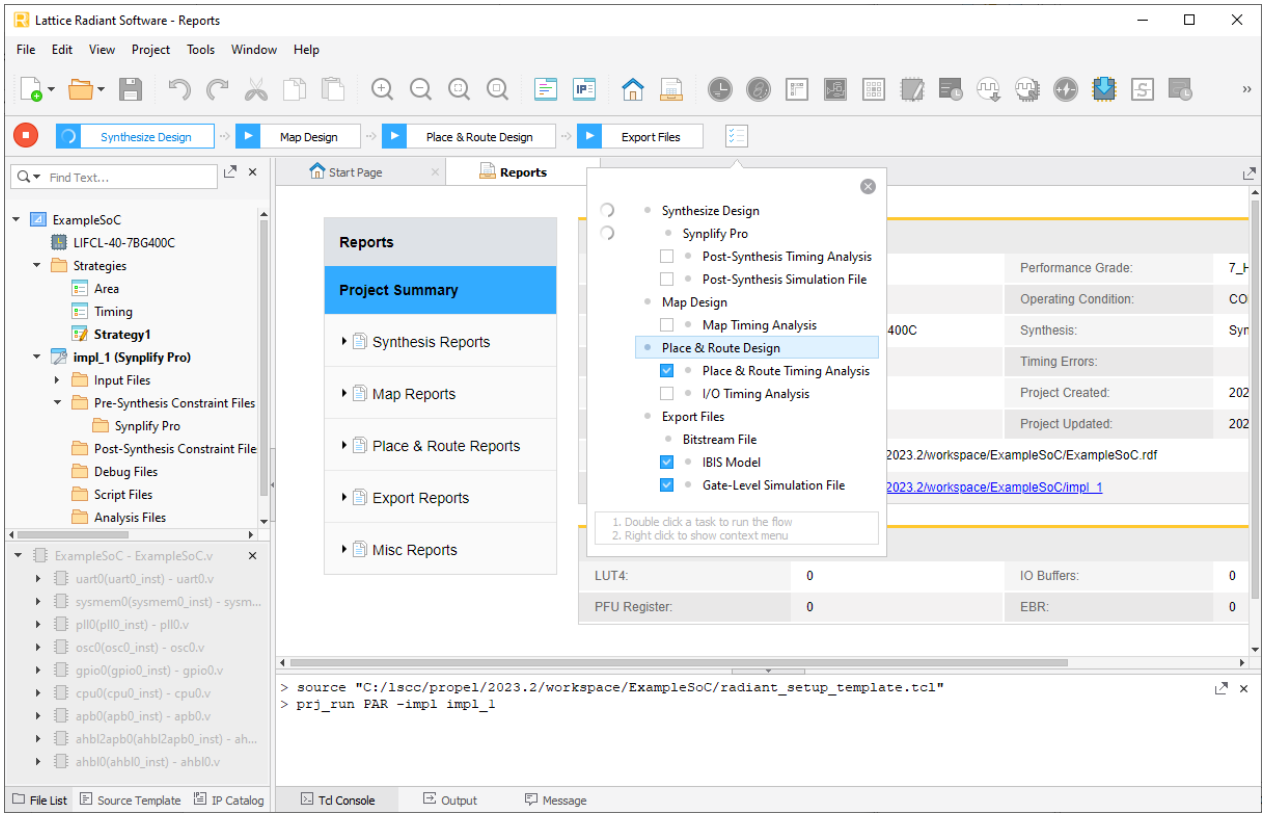


Figure 3.16. Generate Programming File in Lattice Radiant

The Programming file is generated. The generated programming file can be used in the Programmer.

3.2.4. Generating System Environment by Building Project

System environment package including the system environment file and the BSP package is required for the embedded C/C++ project.

To generate system environment package from Lattice Propel:

1. In the **Project Explorer** view, select an SoC project.
2. Choose **Project** > **Build Project**.
3. Check the building result in the **Console** view (Figure 3.17).

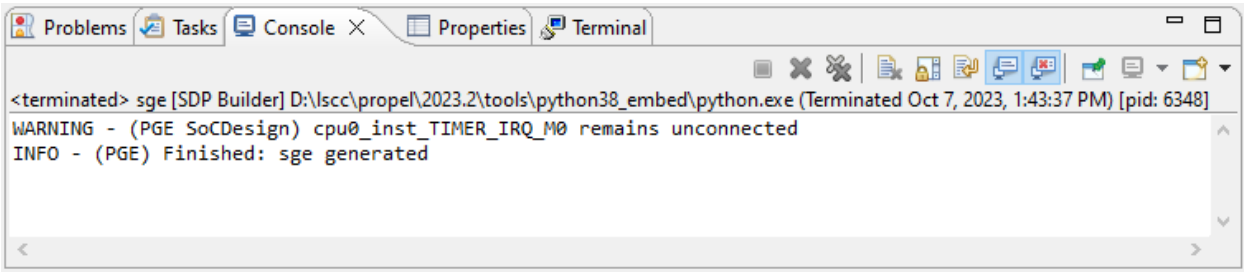


Figure 3.17. Build Result of SoC Project

3.2.5. About SoC Design Project

The SoC project creating starts with a functional-ready SoC design and a default simulation environment. In the **Project Explorer** view, open an SoC project folder and all its sub-folders. The project contains (but is not limited to) the

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

following files (Figure 3.18), some of which may vary upon opening the SoC design project in Diamond or Radiant software:

- <*proj_name*>: folder containing a Propel Builder design including the .sbx file.
- <*proj_name*>/application: folder containing functional-ready embedded application source codes.
- impl1: folder containing the implementation of Diamond/Radiant project.
- sge: folder containing generated package necessary for creating C/C++ project.
- verification: folder containing the SoC verification project.
- verification/sim: folder containing the simulation environment.
- <*proj_name*>.ldf: Diamond project file.
- <*proj_name*>.lpf: Diamond project logical preference file.
- <*proj_name*>.rdf: Radiant project file.
- <*proj_name*>.pdc: Radiant project post-synthesis constraints.
- <*proj_name*>.txt: description file from the template.

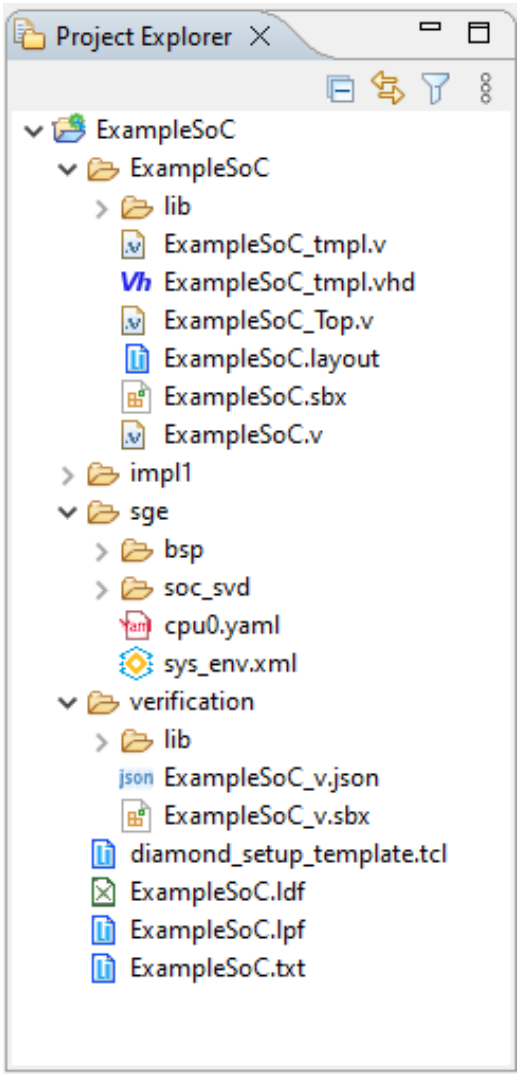


Figure 3.18. Contents of SoC Project

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

3.3. C/C++ Project Design Flow

3.3.1. Creating a Lattice C/C++ Project

To start a Lattice C/C++ Project from Lattice Propel:

1. Choose **File** > **New** > **Lattice C/C++ Project**.
   The C/C++ Project wizard opens with the **Load System and BSP** page (Figure 3.19).

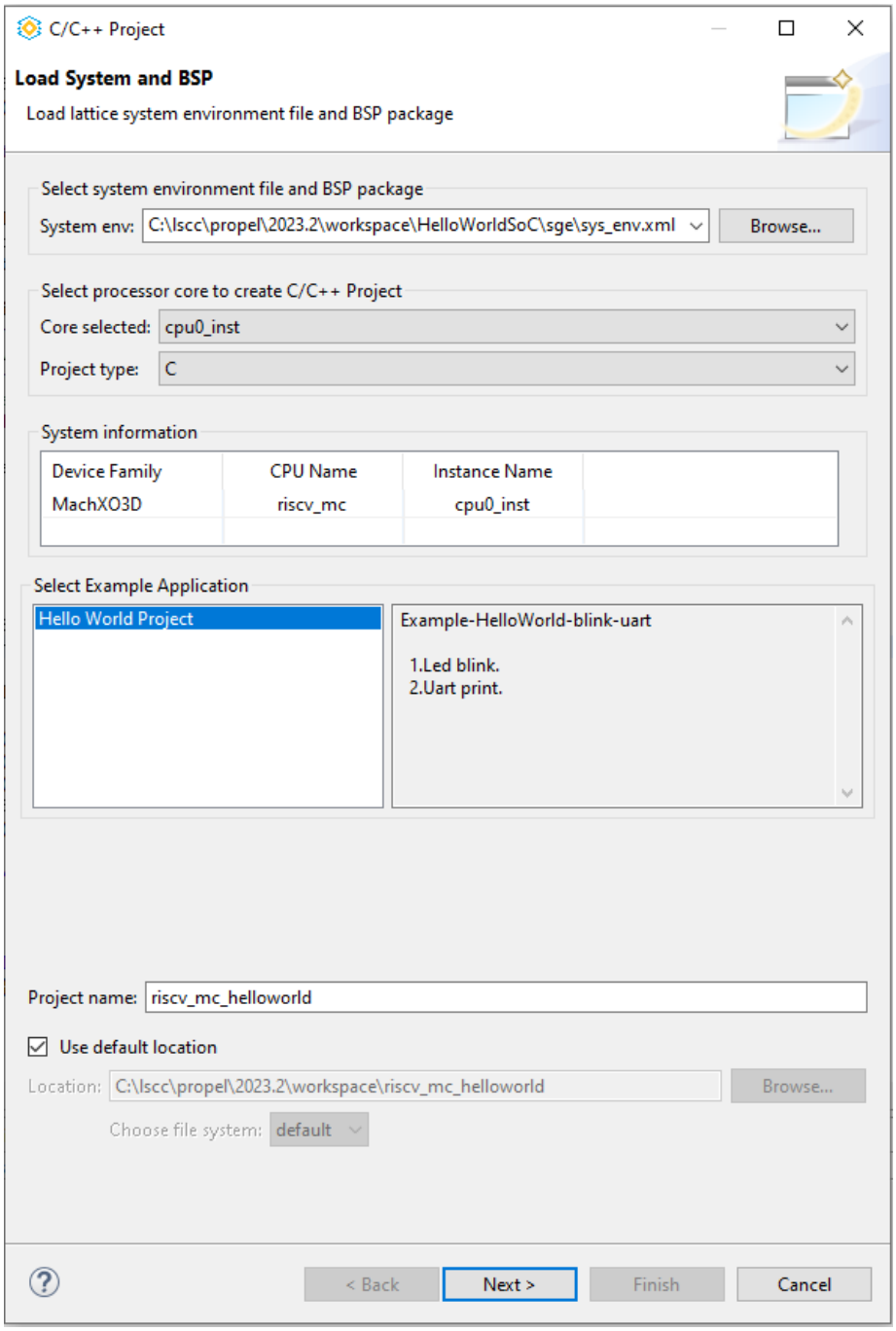


Figure 3.19. Load System and BSP Dialog

2. Browse to the SoC project folder and select the system environment file sys_env.xml.

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

All system environment files available in the current workspace can be selected from the **System env:** drop-down menu.

3. If the platform has more than one processor, choose one core.
4. Choose the project type, C or C++.
5. Select the application in the Example application list.
6. There is a default project name but you need to check it. Suggest not using periods, colons, or spaces in your project name. Though spaces are allowed, they may cause certain issue with some tools.
7. By default, the **Use default location** option is checked. The default file system is selected automatically. Suggest using the default location unless you have special need to a special location.
8. Click **Next**.

   The **Lattice Toolchain Setting** dialog opens (Figure 3.20).

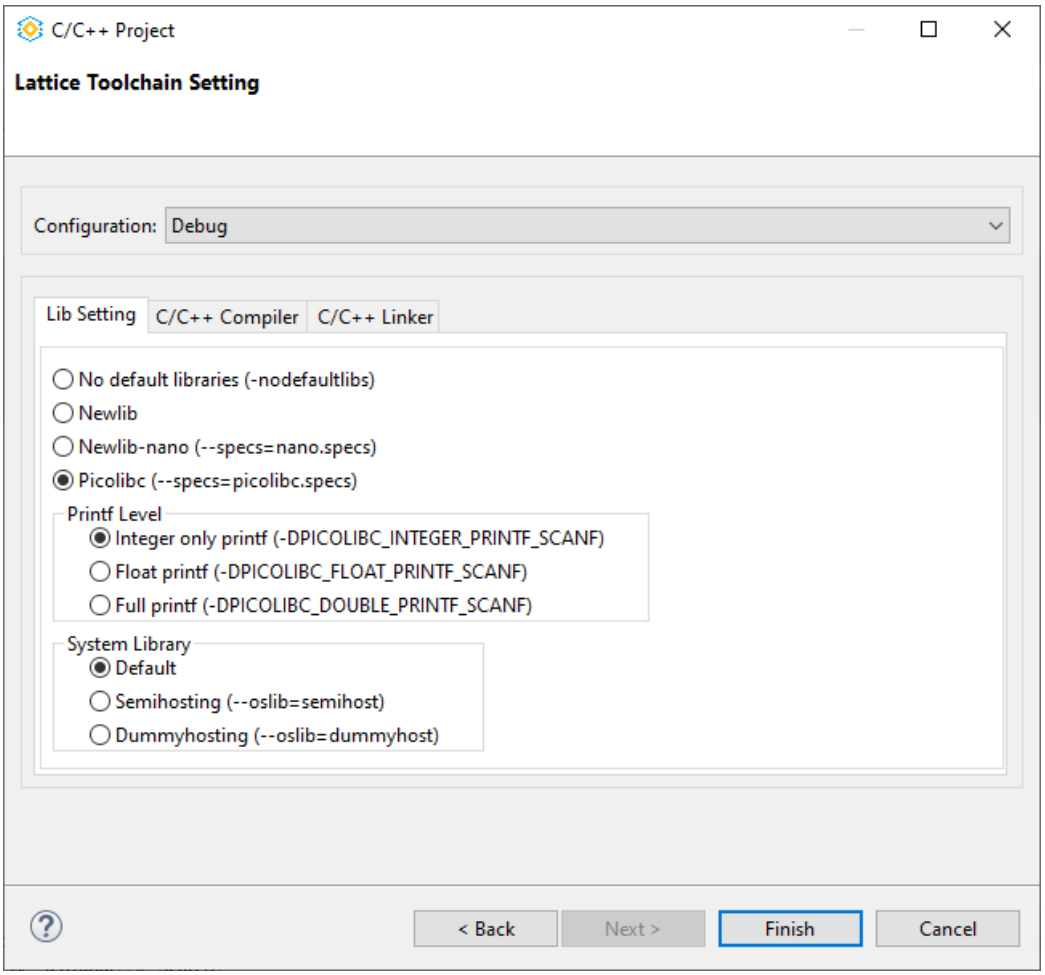


Figure 3.20. Lattice Toolchain Setting Dialog

9. By default, two toolchain configuration modes, **Debug** and **Release**, can be chosen from the Configuration drop-down menu.
   - **Debug** configuration creates executables containing additional debug information that lets the debugger make direct associations between the source code and the binary files generated from the original source.
   - **Release** configuration provides the tools with options setting to create an application with the best performance.

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

You can modify frequently-used library, compiler, and linker options for each configuration. For a complete toolchain setting, go to project properties after creating the project. Refer to the Advanced Tool Chain Setting section.

- In **Lib Setting** tab, standard C library can be reconfigured. Picolibc (C Libraries for Smaller Embedded Systems) is selected by default and it supports different printf levels.
- In **C/C++ Compiler** tab, optimization level and debug level can be reconfigured for each toolchain configuration.
- In **C/C++ Linker** tab, Remove unused code (--gc-sections) is checked by default for garbage collection of unused code.

10. Click **Finish**.

    The Lattice C/C++ project is created and is displayed using the Propel SDK perspective. A perspective is a collection of tool views for a particular purpose. The Propel SDK perspective is for creating Lattice C/C++ programs.

3.3.2. Updating a Lattice C/C++ Project

When you make changes to an SoC project, sometimes you want to synchronize the changes to an existing Lattice C/C++ project instead of creating a new Lattice C/C++ project. In this case, you can use the update C/C++ project feature.

Note: This feature overwrites the corresponding files or settings of your existing C/C++ project. Be sure to back up your C/C++ project before using this feature.

To update a Lattice C/C++ Project from Lattice Propel:

1. Generate the latest system environment package according to the Generating System Environment by Building Project section.
2. In the **Project Explorer** view, select a C/C++ project.
3. Choose **Project** > **Update Lattice C/C++ Project…**.

   The C/C++ Project wizard opens for updating system and BSP (Figure 3.21).

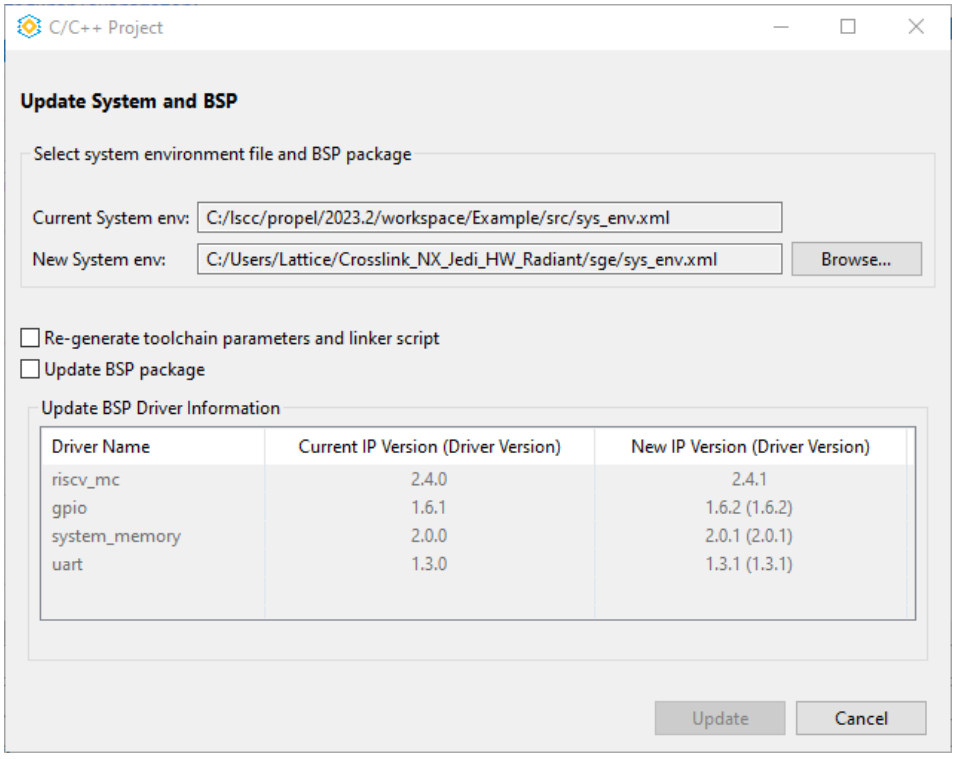


Figure 3.21. Update System and BSP Dialog

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

4. Browse to the SoC project folder and select the system environment file sys_env.xml.
5. Select the checkbox for what you can update:
   - Re-generate toolchain parameters and linker script: check this option if you want to modify CPU or memory in the system.
   - Update BSP package: check this option if you want to add additional IP components into the system.
6. Click **Update** to make changes for the selected C/C++ project.

3.3.3. Building a Lattice C/C++ Project

To build a Lattice C/C++ project in Propel:

1. In the **Project Explorer** view, select a C/C++ project.
2. Follow steps below if you want to change the active build configuration:
   a. Choose **Project > Build Configurations > Manage…**. Or, click the **Configuration** icon on the toolbar.
   b. The **Manage Configurations** dialog opens (Figure 3.22) for choosing active configuration. By default, a **Debug** configuration creates executables containing additional debug information that lets the debugger make direct associations between the source code and the binary files generated from the original source. A **Release** configuration provides the tools with options setting to create an application with the best performance.

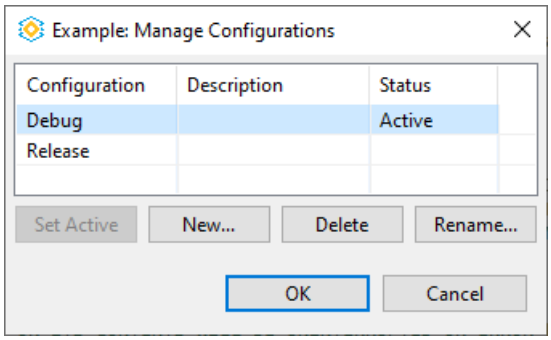


Figure 3.22. Manage Configurations Dialog

3. Choose **Project** > **Build Project**. Or, click the **Build** icon on the toolbar.
4. The results of the build command are displayed in the **Console** view (Figure 3.23).

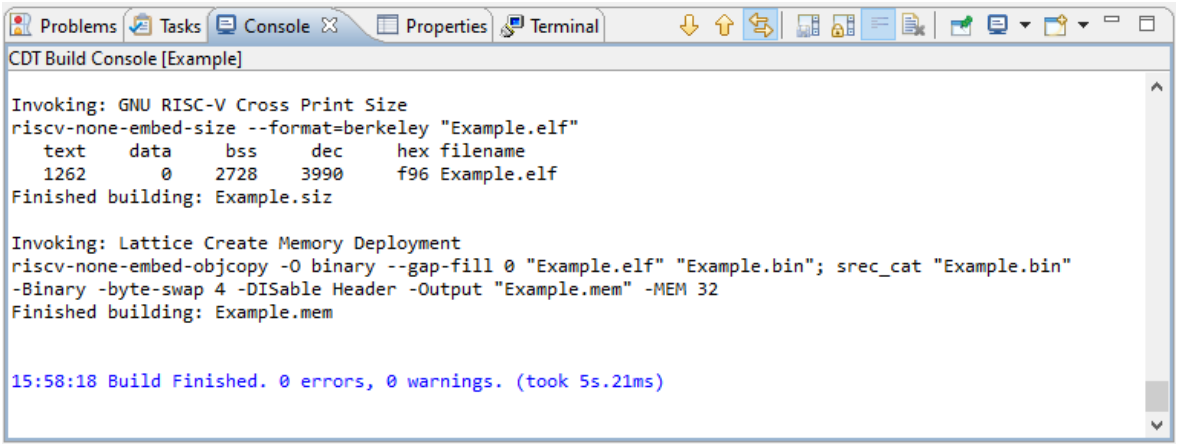


Figure 3.23. Build Result of C/C++ Project

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

3.3.4. About Lattice C/C++ Project

The Lattice C/C++ project starts with source code. In the Project Explorer view, open a C/C++ project folder and all its sub-folders. The project contains:

- src/bsp/driver: folder containing driver codes from the IP in the platform.
- src/bsp/sys_platform.h: header file that defines DEVICE_FAMILY (the Lattice FPGA), address mapping, and any IP parameters that can be used by the drivers.
- src/main.c: source file containing the main routine, which is the entry-point of a C/C++ program.
- src/cpu0.svd: system view description file used for peripherals registers view at debug perspective.
- src/cpu0.yaml: processor description file used at debugging time.
- src/linker.ld: linker script file.
- src/sys_env.xml: system environment file describing aspects of the platform, such as memory spaces.

After building the project, the build output can be found in each build configuration folder, the **Debug** folder or the **Release** folder (Figure 3.24). The Debug or Release folder contains:

- <*proj_name*>.elf: executable file used in on-chip debugging.
- <*proj_name*>.bin: binary file used in deploying the application to flash memory.
- <*proj_name*>.lst: extended listing file generated by tool objdump.
- <*proj_name*>.map: linker map file.
- <*proj_name*>.mem: Lattice system memory initialization file used in System_Memory IP.

Note: Some of the files listed in Figure 3.24 are intermediate files that you do not need to take care of.

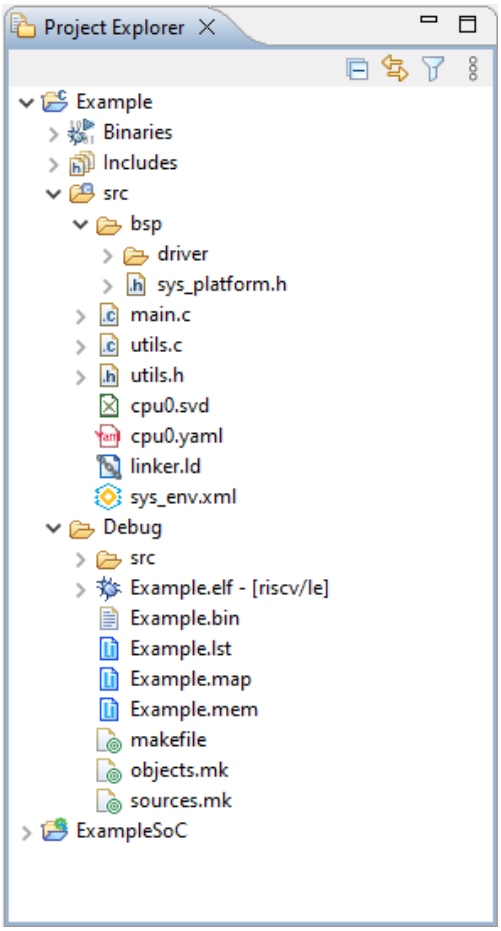


Figure 3.24. Contents of C/C++ Project

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

3.3.5. Writing Code

Lattice Propel is based on Eclipse IDE. You can write application code following the process and usage of the same tools as any in Eclipse IDE. You can get more detailed information regarding Eclipse IDE from the Propel online help.

For writing code, Lattice Propel SDK provides two extra aids:

- Lattice System Platform: An overview of the processor platform can be displayed (Figure 3.25).
- Linker Editor: An overview of the memory regions of linker script can be displayed. You can modify key linker parameters via the graphical interface (Figure 3.26).

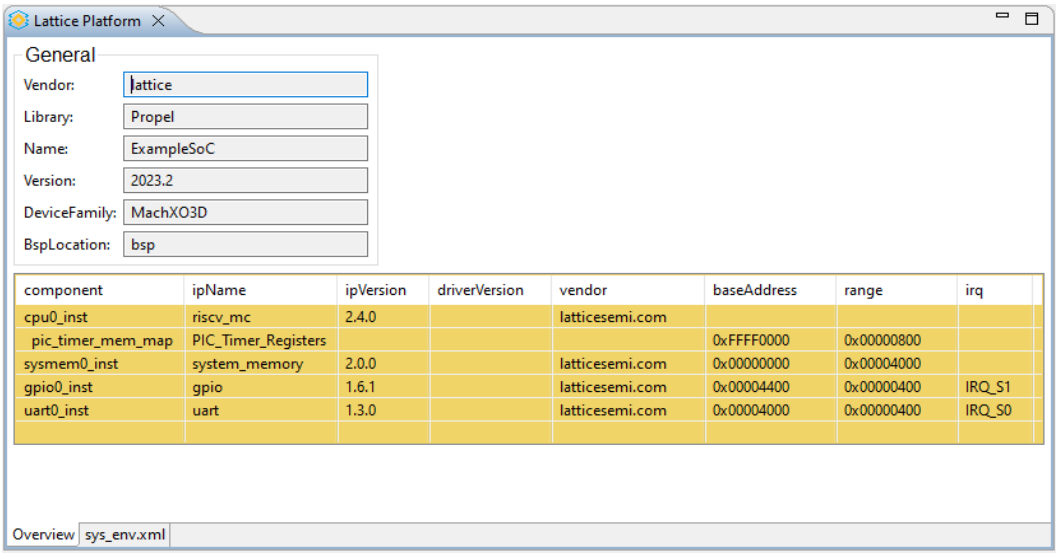


Figure 3.25. Lattice System Platform

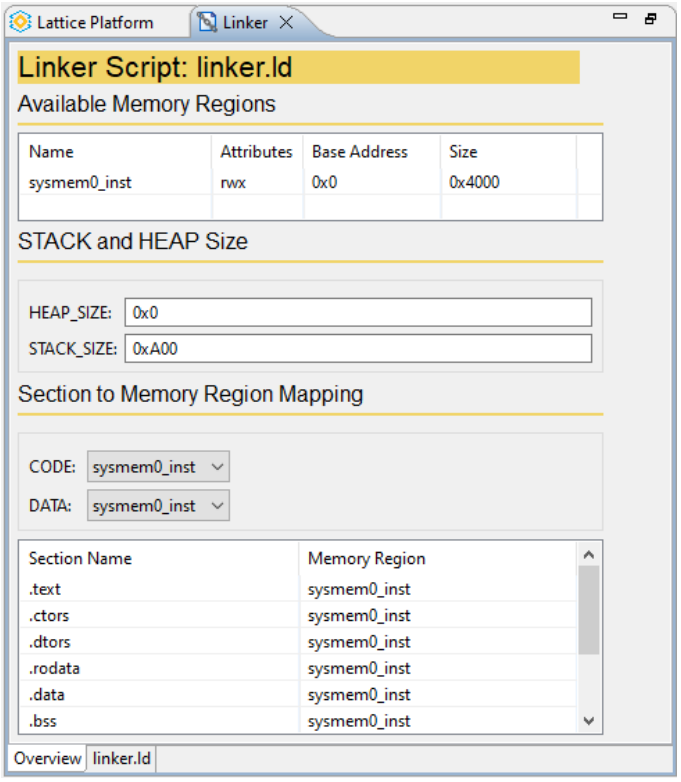


Figure 3.26. Linker Editor

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

3.3.6. Advanced Tool Chain Setting

Follow the process below to modify the tool chain settings of a C/C++ project.

To change tool chain setting in Project Properties in Lattice Propel:

1. In the **Project Explorer** view of Propel, select a C/C++ project.
2. Choose **Project** > **Properties.** The Properties for current project opens (Figure 3.27).
3. Select **Settings** of **C/C++ Build** category from the left pane. Select the **Tool Settings** tab.

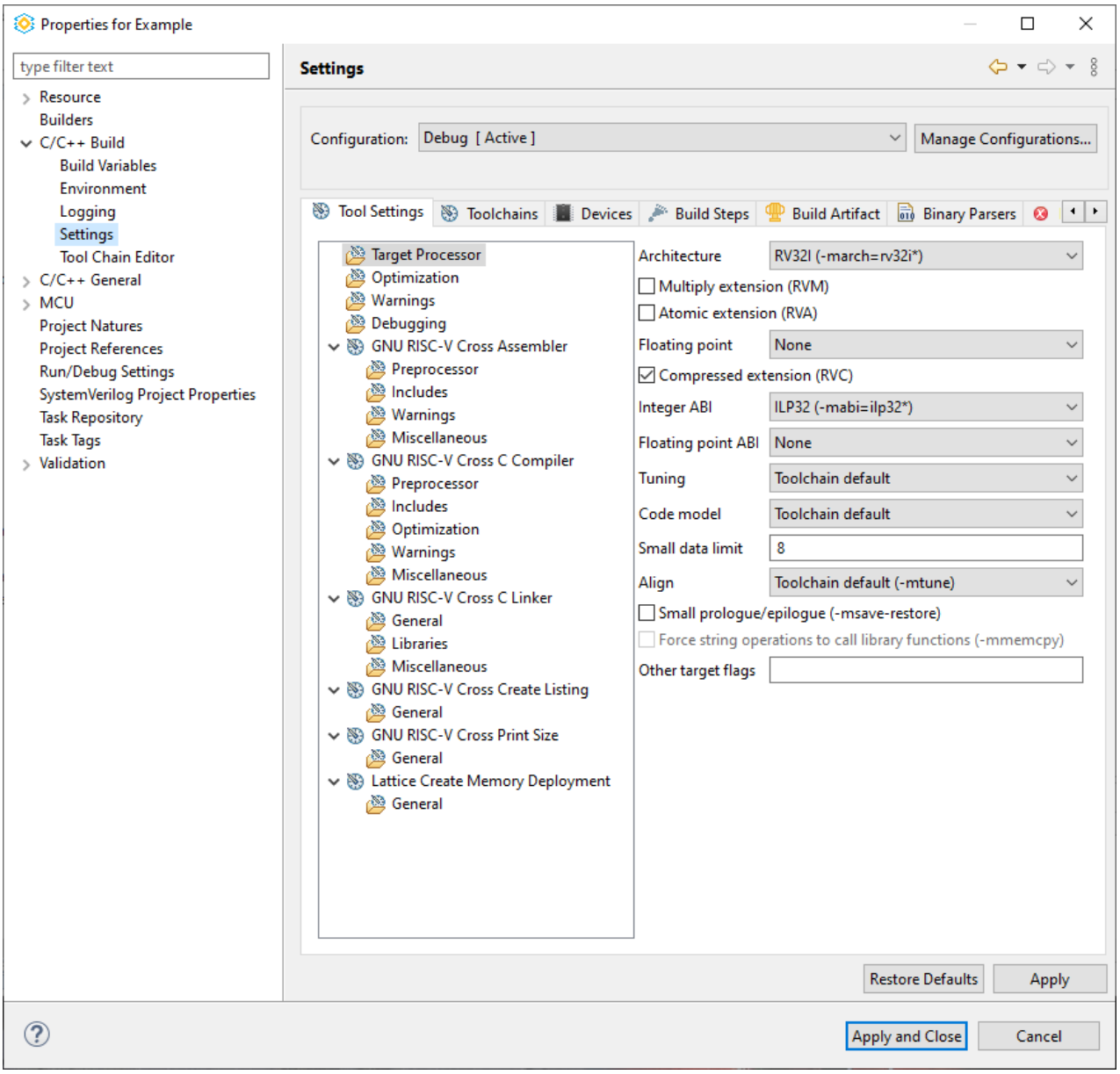


Figure 3.27. Properties of C/C++ Project

4. Customize the tools and tool options. All your customization can be resulted in the build configuration in the **Tool Settings** properties tab. The build configuration is used during your C/C++ project building.

   Note: The setting for each configuration, **Debug** or **Release**, is independent.

5. Click **Apply and Close** to save the change.

   Note: You may need to clean the project to make the new setting take effect for the whole project.

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

3.4. System Simulation Flow

The SoC Project created from template has a default simulation environment for you to setup and start functional simulation. It is generated automatically along with the SoC project creation. You can use it as a start point and customize accordingly.

The default simulation environment is with the following features:

- Provides similar user experience as real board-level debugging, such as for Hello World SoC, key components including RISC-V MC, System memory, and UART.
- Simulates user-modified template SoC with extended HDL designs.
- Simulates the whole system using real C/C++ projects as stimulus with the necessary modification and with all the details for debugging.
- Supports user extension with a friendly and flexible approach.

3.4.1. Launch Simulation

To launch simulation:

1. In Propel Builder, update the SoC design to enable simulation features.
   - Enable the checkbox for **Simulation Mode**, and disable the checkbox for **Debug Enable** for RISC-V MC IP module in the **General** area (Figure 3.28).

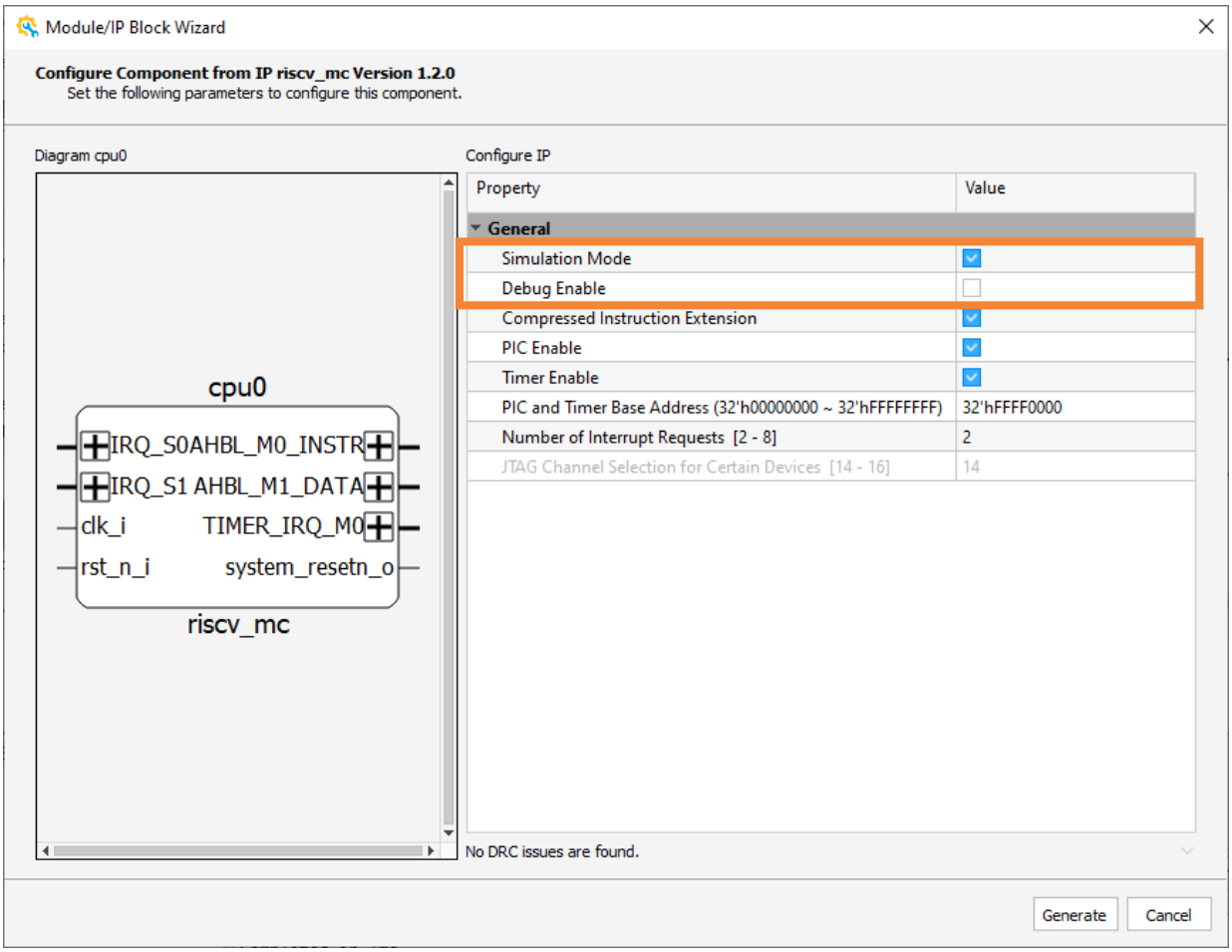


Figure 3.28. Configure Module RISC-V MC

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

- Enable the checkbox for **Initialize Memory** for system_memory IP module from the **Initialization** area of the **General** tab, and set **Initialization File** generated from the corresponding C/C++ project (Figure 3.29).

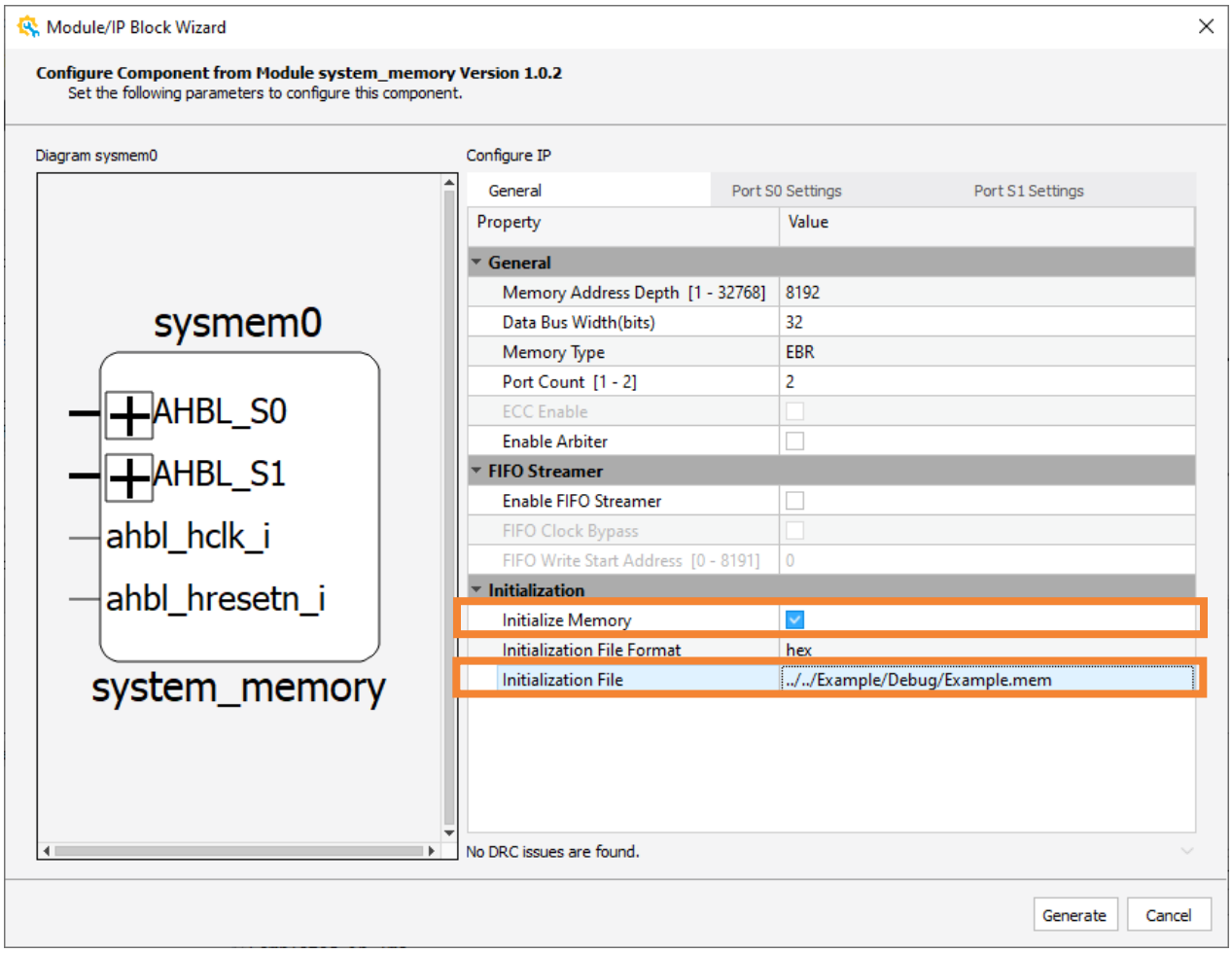


Figure 3.29. Configure Module System Memory

2. Click the **Switch** icon on the toolbar to switch between SoC design and SoC verification project (Figure 3.30).
3. After the SoC design is switched to an SoC verification project, click the **Generate** icon to generate the simulation environment for OEM ModelSim. Click the **Launch Simulation** icon (Figure 3.30).

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

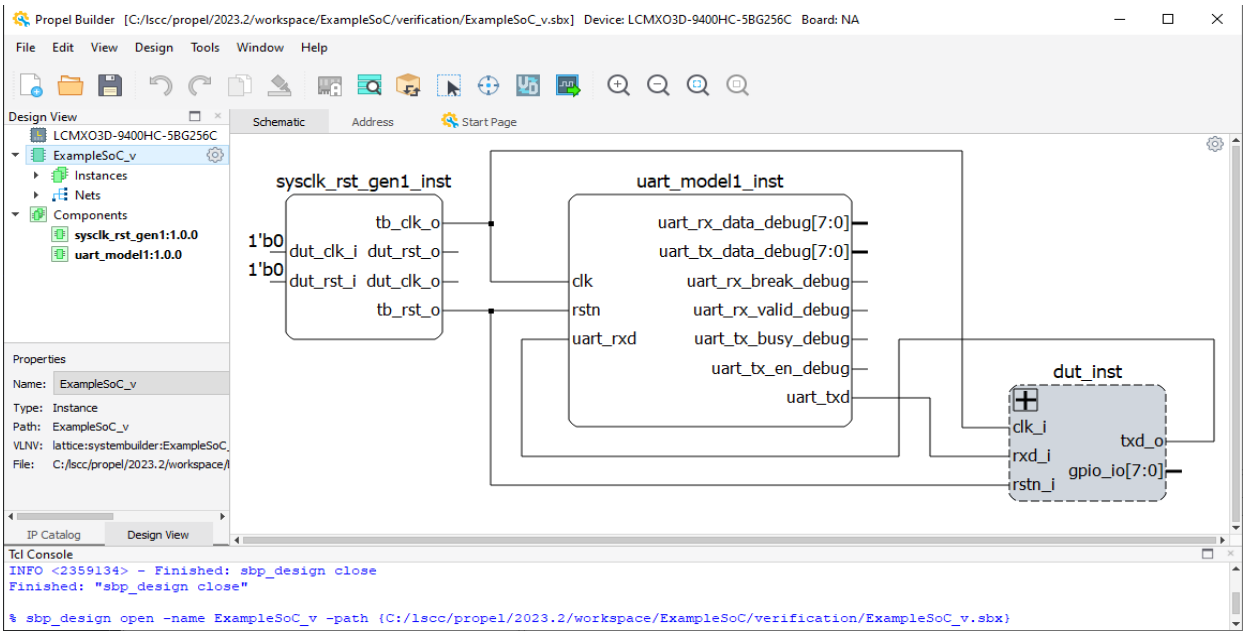


Figure 3.30. SoC Verification Project

4. ModelSim is launched running simulation for the SoC verification project. The corresponding waveform of the SoC verification project for the Hello World project is shown (Figure 3.31). Check the waveform.

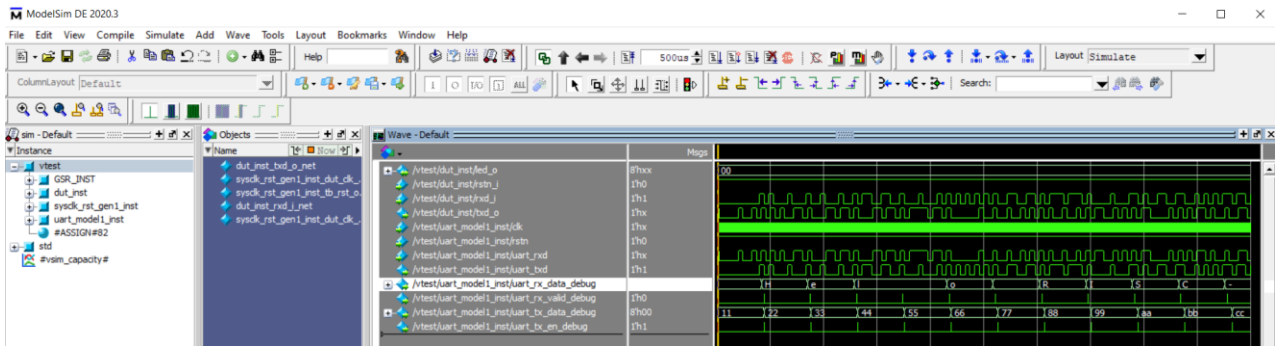


Figure 3.31. Waveform of Hello World Project

3.4.2. Simulation Details

The default simulation environment is located at the generated sim folder inside the SoC verification project in Lattice Propel. It contains:

```
[sim]                          -- generated simulation environment
    [hdl_header]
        soc_regs.v             -- register definitions of all the components in DUT/SOC
        sys_platform.v         -- base address, user settings of all the components in DUT/SOC
    [misc]
        *.*                    -- all the mem, hex, txt files will be copied here
    flist.f                    -- file list for HDLs
    msim.do                    -- do script for simulator, it can be qsim for Questasim
    wave.do                    -- do script for adding signals in waveform window
    <project_name>.sv          -- top testbench, SystemVerilog based
```

You can extend more verification features in the top testbench.

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

About Hello World Project

The Hello World Project is designed to print some strings using a top-level UART port. The testbench instantiates a UART model to receive these data. Each byte (refer to uart_rx_data shown in Figure 3.31) can be checked using ASCII format. Meanwhile, this UART model also supports sending the data to Design Under Test (DUT). The data is stored in a text file and is enabled by the STIMULUS_GEN parameter.

3.5. Programming and On-Chip-Debugging Flow

This section describes the process of testing and debugging application code on the actual hardware including the Lattice FPGA with the hardware design installed. Debugging with Propel SDK follows the same process and uses of the same tools in Eclipse IDE.

Before debugging, download the hardware design created from Diamond/Radiant Programmer. Refer to the User Guide of the specific evaluation board for more details on the evaluation board.

3.5.1. Creating a Debug Launch Configuration

To debug a program, a debug launch configuration must be created. Most of the settings for a debug launch configuration can be automatically entered. Only a few settings need to be manually configured.

To create a debug launch configuration:

1. In the **Project Explorer** view of Lattice Propel, select a C/C++ project.
2. Build the project and ensure the executable file is available. Refer to the Building a Lattice C/C++ Project section for details on the process.
3. Choose **Run** > **Debug Configurations…**.

   The **Debug Configurations** dialog opens (Figure 3.32).

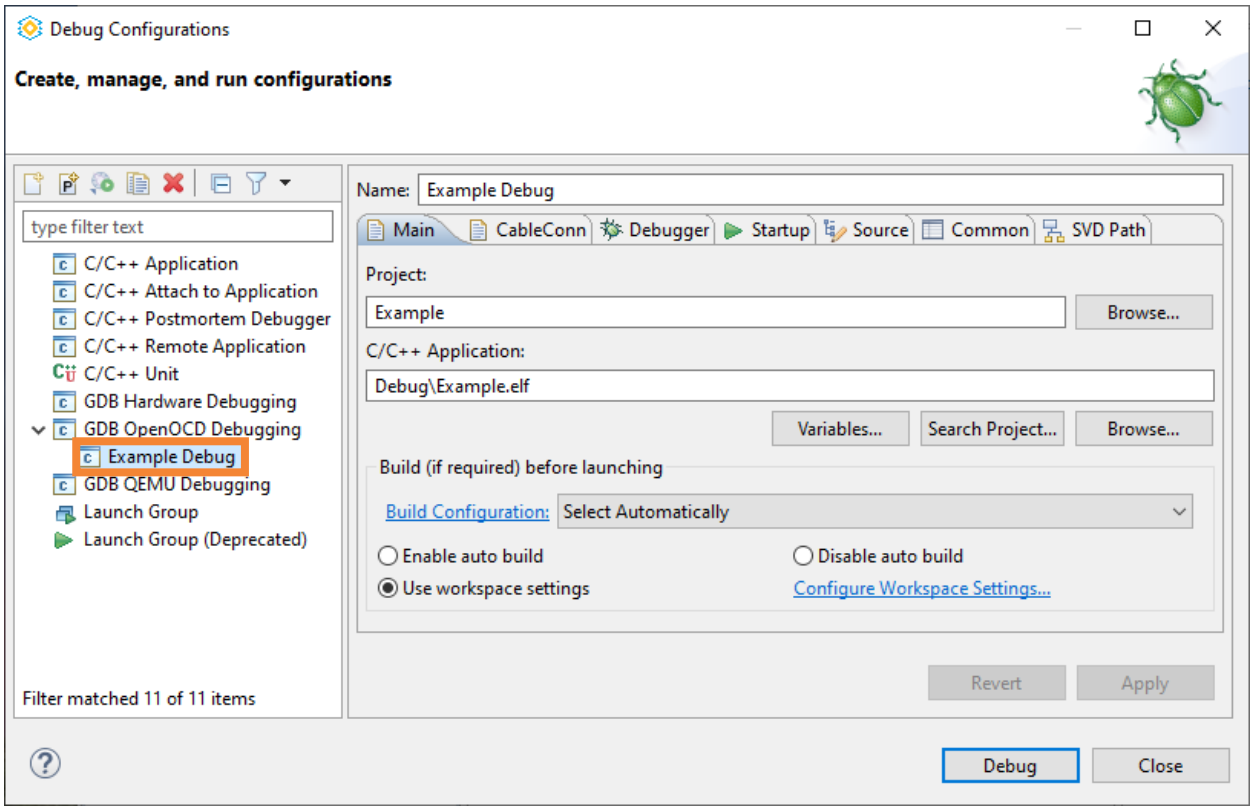


Figure 3.32. Debug Configurations Page

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

4. Double-click **GDB OpenOCD Debugging** to create a new launch configuration.

   A multi-tab page is displayed. The Main tab should already be filled in with the project name, application file name, and location.

5. Select the **CableConn** tab (Figure 3.33). This tab enables you to select a specific device on a specific cable port.

   Click the **Detect Cable** button. Select the specific cable port from the **Port** drop-down list. By default, the first available cable port: FTUSB-0 is selected.

   Click the **Scan Device** button. Select the specific device from **Device** drop-down list. By default, the first available device on selected cable port is selected.

   Select the JTAG channel number from **Channel** drop-down list. By default, channel 14 is selected with the same value as the processor preset.

   Keep the cable speed so that you can use the default clock divider.

   Note: You need to repeat the **Detect Cable** and **Scan Device** steps if you have plugged or unplugged the cable.

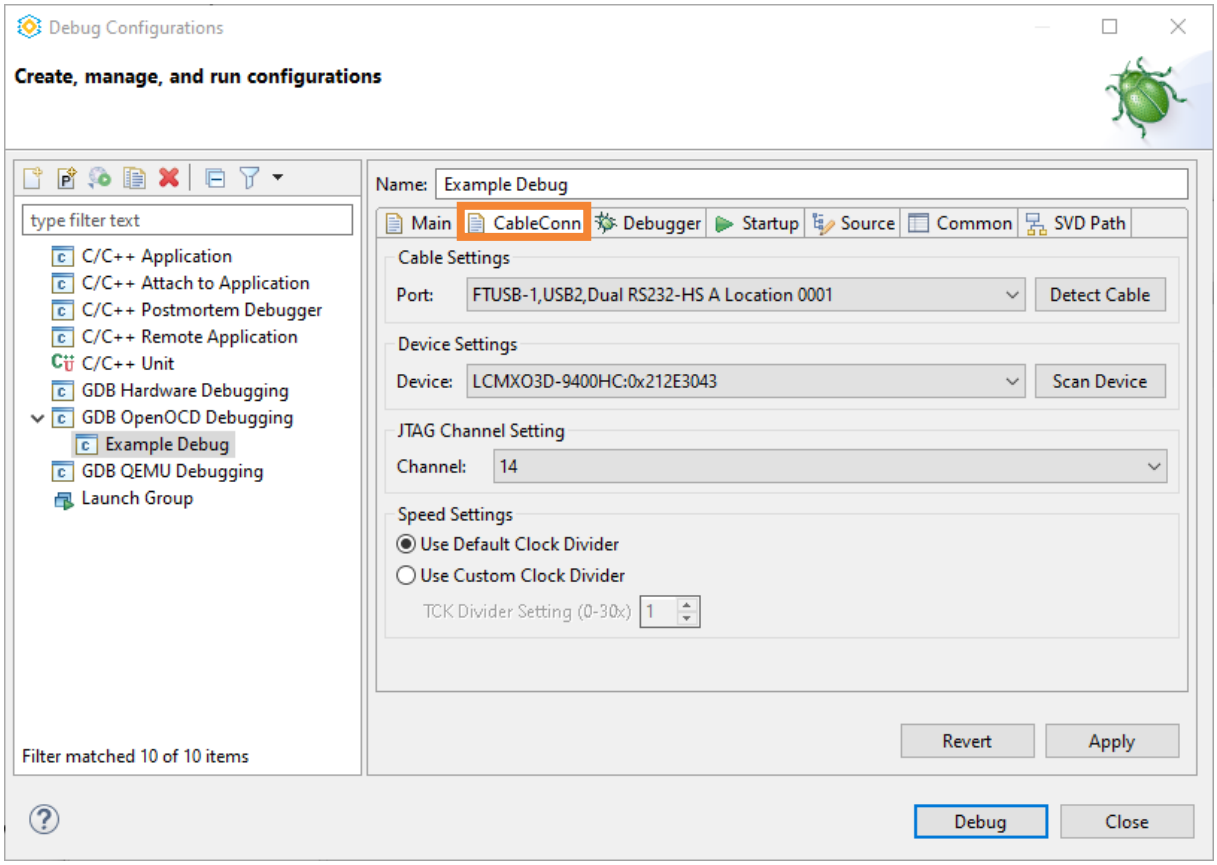


Figure 3.33. CableConn Tab of Debug Configurations

6. Select the **Debugger** tab (Figure 3.34). It is critical that the **Config options** field contains the correct command line options to be passed to OpenOCD.

   -c "set port ${PORT}"* is required for the selection connected to the Lattice cable. The value of variable *"${PORT}" comes from the cable settings of the **CableConn** tab.

   -c "set target ${DEVICE}"* is required for the selection of a specific device on the Lattice cable. The value of variable *"${DEVICE}" comes from the device settings of **CableConn** tab.

   -c "set channel ${CHANNEL}"* is required for setting the JTAG channel. The value of variable *"${CHANNEL}" comes from the jtag channel setting of the **CableConn** tab.

   -c "set tck ${TCKDIV}"* is required for setting the clock divider of the Lattice cable. The value of variable *"${TCKDIV}" comes from the speed setting of the **CableConn** tab.

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

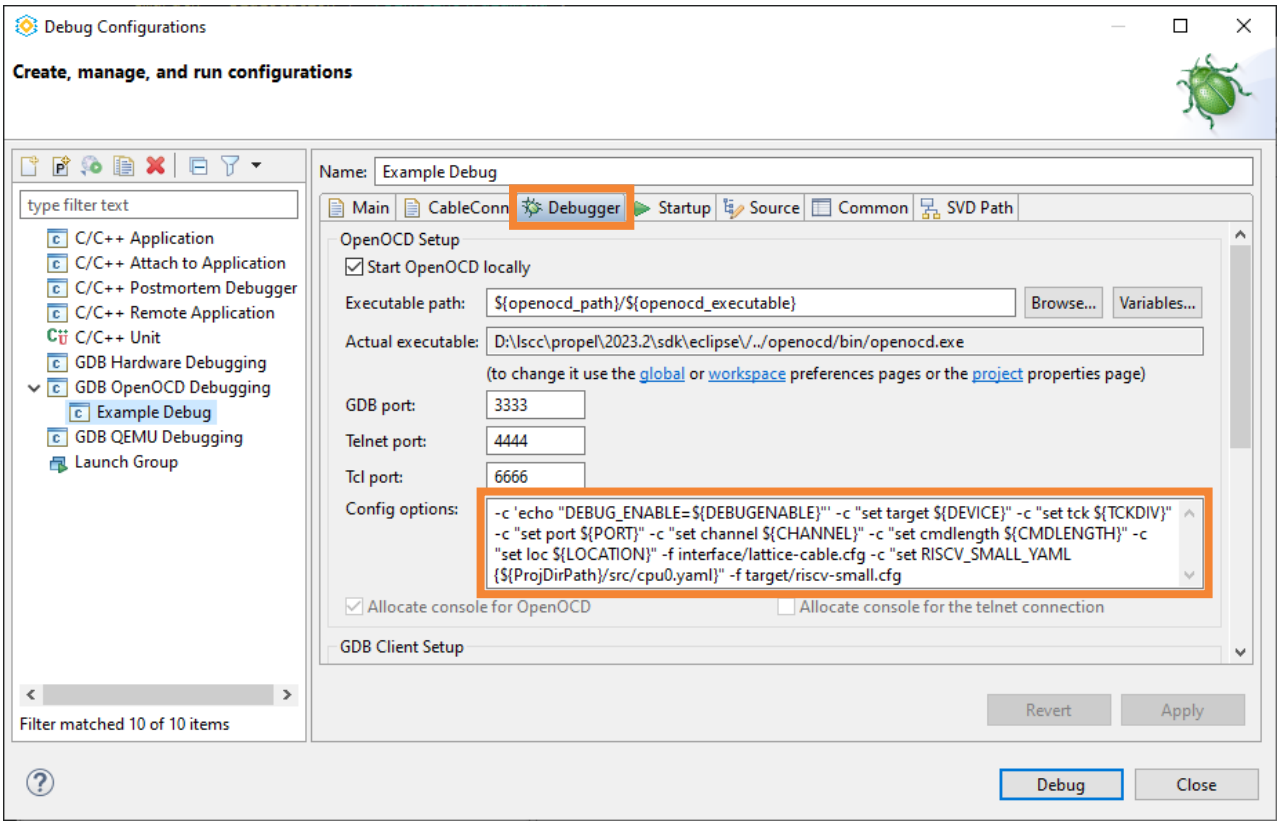


Figure 3.34. Debugger Tab of Debug Configurations

7. (Optional) Select the **Common** tab (Figure 3.35). The **Save as** > **Local file** option is selected by default. This causes the debug launch configuration to be saved into the workspace.

   You can change the setting of the **Save as** field to **Shared file**. In this way, the debug launch configuration is saved into the project and this aids the project portability.

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

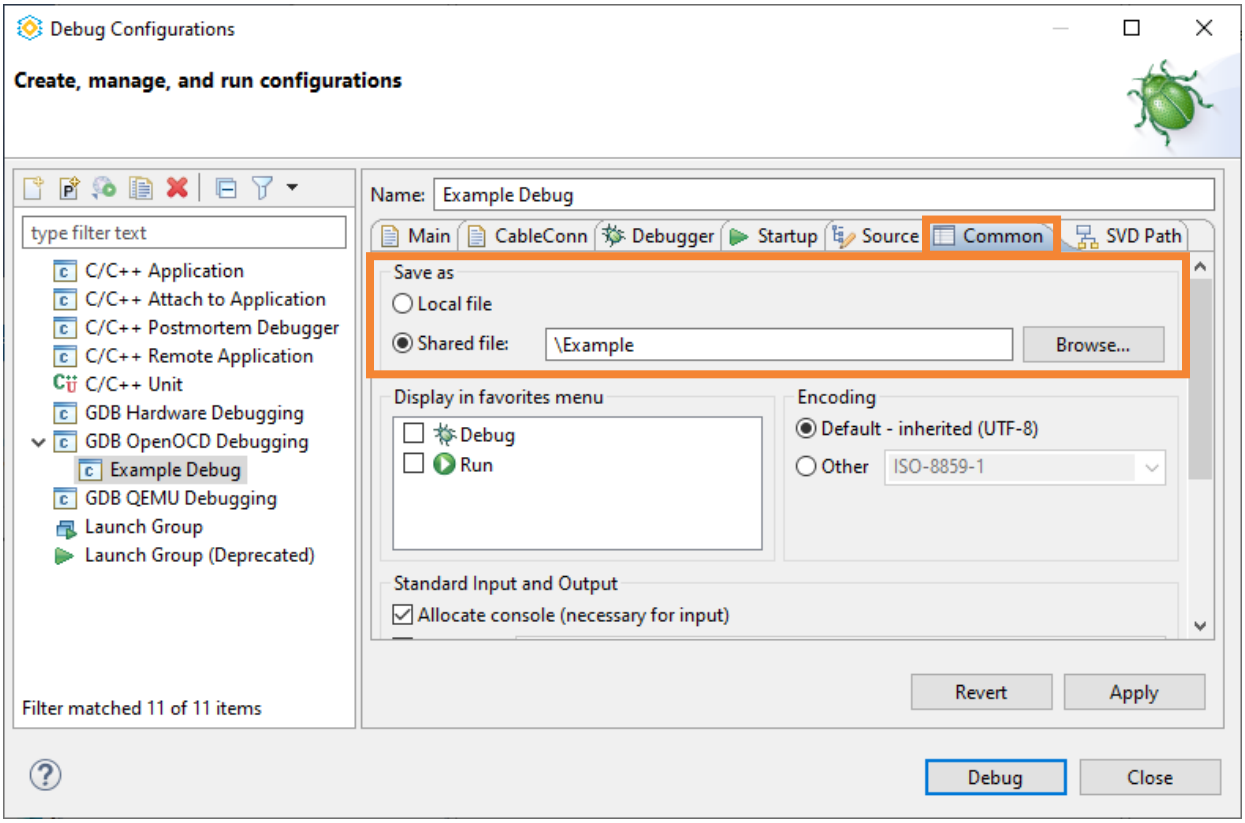


Figure 3.35. Common Tab of Debug Configurations

8. Remain settings as default. Do not change the settings unless necessary, or unless you understand what effect these changes may bring.
9. Click **Apply** to keep the current settings.
10. Click **Close**.

3.5.2. Starting a Debug Session

Before starting a debug session, be sure that:

- The Lattice cable is connected to the computer.
- The target device is power ON.
- The hardware design has a debug enabled processor module and already programming into the target device.

With the above steps completed properly, follow the steps below to start the debug session from Lattice Propel.

1. Choose **Run** > **Debug Configurations…**.
2. If necessary, expand the **GDB OpenOCD Debugging** group.
3. Select the newly-defined configuration.
4. Click the **Debug** button (Figure 3.35).

   Alternatively, for later sessions, use the **Debug** icon on the toolbar. Do not click the Debug icon directly. Instead, click the down arrow beside the **Debug** icon. Select the desired debug configuration from the drop-down menu (Figure 3.36).

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

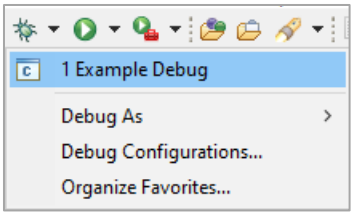


Figure 3.36. Debug Icon on Toolbar

5. Wait for a few seconds for switching to debug perspective, starting the server, connecting to the target device, starting the gdb client, downloading the application, and starting the debugging session.
6. The Lattice Propel Window displays, as shown in Figure 3.37. The execution stops right at the beginning of the main() function.

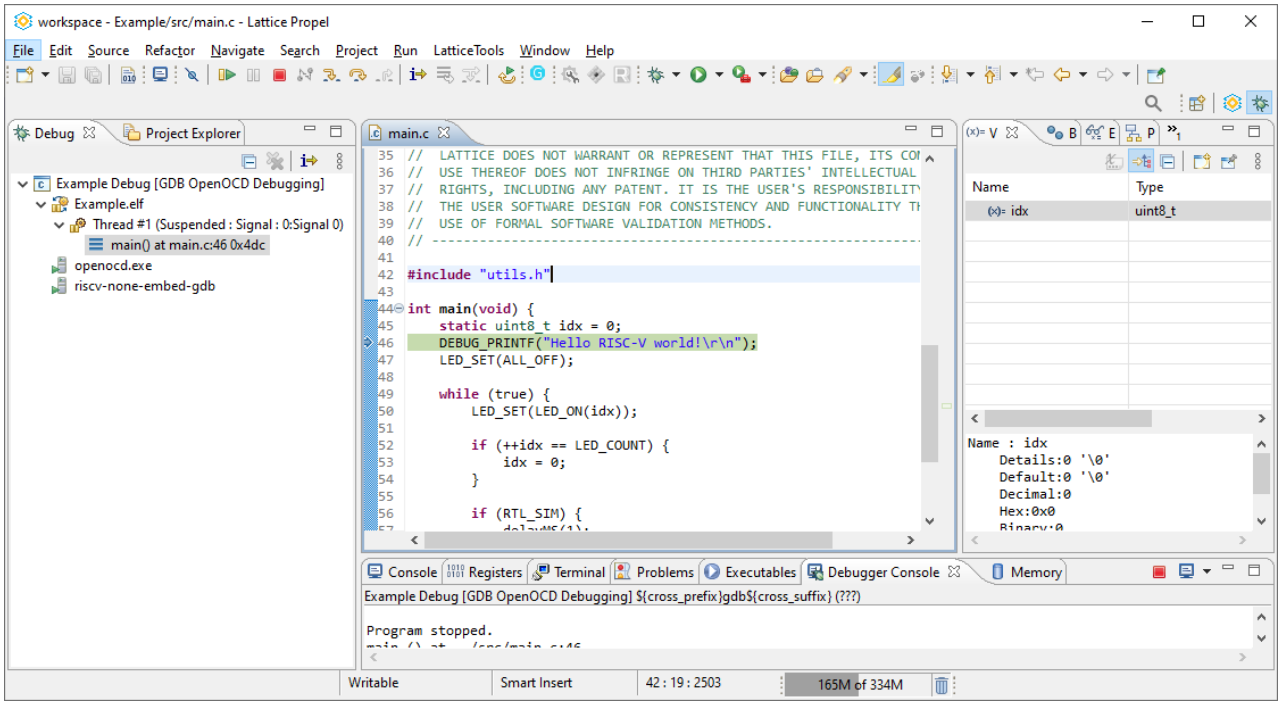


Figure 3.37. Debug Perspective

3.5.3. Peripherals Registers View

Peripherals registers view provides an easy-to-use interface for examining or modifying the values of peripheral registers during a debug session.

To use peripherals registers view in Lattice Propel (Figure 3.38):

1. Make sure an active debug session is run and shown in the debug perspective.
2. Find the **Peripherals** view which is in the same window with the **Variables** and **Breakpoints** views. For any reason, if this view is not found, re-open it from **Window** > **Show View** > **Peripherals**.

   The **Peripherals** view lists all peripherals available in the system view description svd file within the C/C++ Project.
3. Selecting a peripheral in the **Peripherals** view can open a **Memory Monitor** that is mapped to the corresponding peripheral memory area.
4. You can examine and modify the value of the peripherals register in the **Memory** view.

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

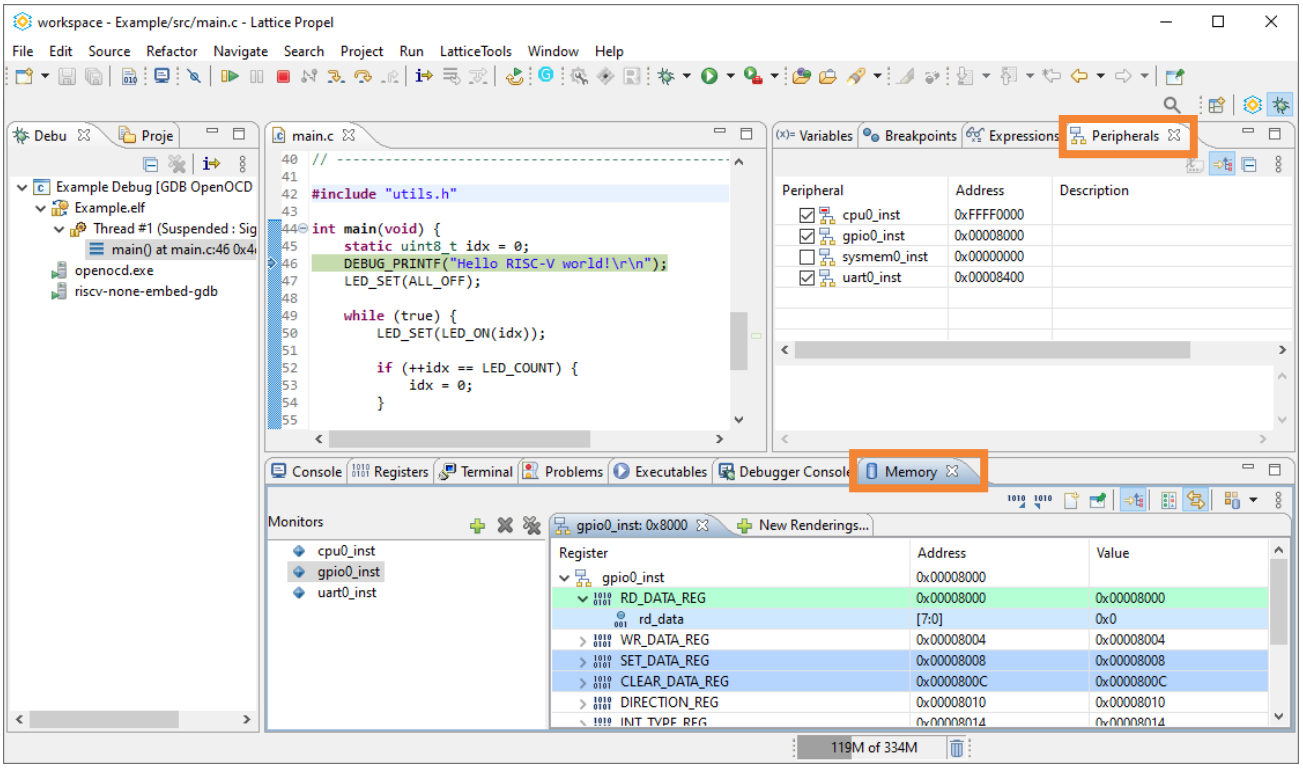


Figure 3.38. Peripherals View in Debug Perspective

3.5.4. Serial Terminal Tool

Serial port communication is frequently used during the microcontroller debugging. Lattice Propel SDK provides a built-in terminal tool including serial support for debugging.

To launch a serial terminal:

1. Find the **Terminal** view nested to the **Console** view. If this view is not found, re-open it from **Window** > **Show View** > **Terminal**.
2. In the **Terminal** view, click the **Open a Terminal** icon . The **Launch Terminal** dialog opens (Figure 3.39).
3. Choose the **Serial Terminal** and configure the **Serial port** with **Baud rate**.

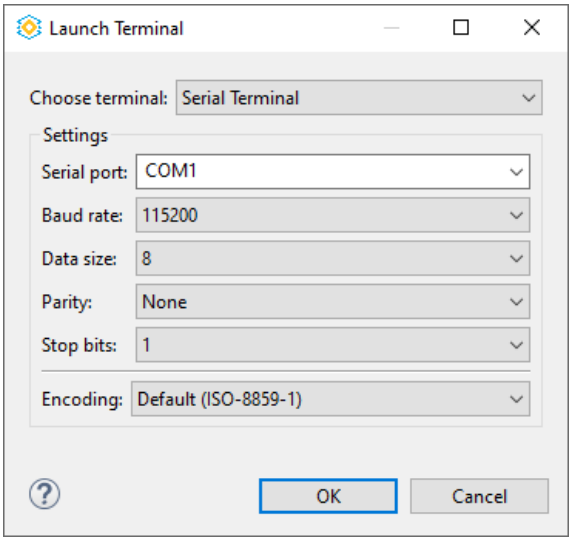


Figure 3.39. Launch Terminal Dialog

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

4. Click **OK**. A connection opens.
5. (Optional) Click the **Toggle Command Input** icon that adds an edit box to enter text (Figure 3.40).

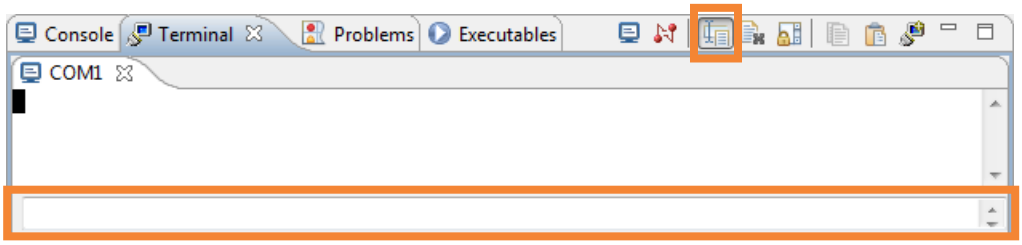


Figure 3.40. Terminal View

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

4. Propel Tutorial – Hello World

This Hello World project can be found from the template. The Hello World project provides a template of using hardware and software design with the minimal resource required.

Following this tutorial, you can easily create a hardware and software project. After that, you can run the project on your evaluation board.

The tutorial uses a MachXO3D breakout board for demonstration. It uses RS232 UART function. To enable RS232 UART function on the MachXO3D breakout board, the following reworks on the board are required.

1. Hardware reworks on MachXO3D breakout board.

   Short R14 and R15 using 0 Ω resistors, as shown in Figure 4.1.

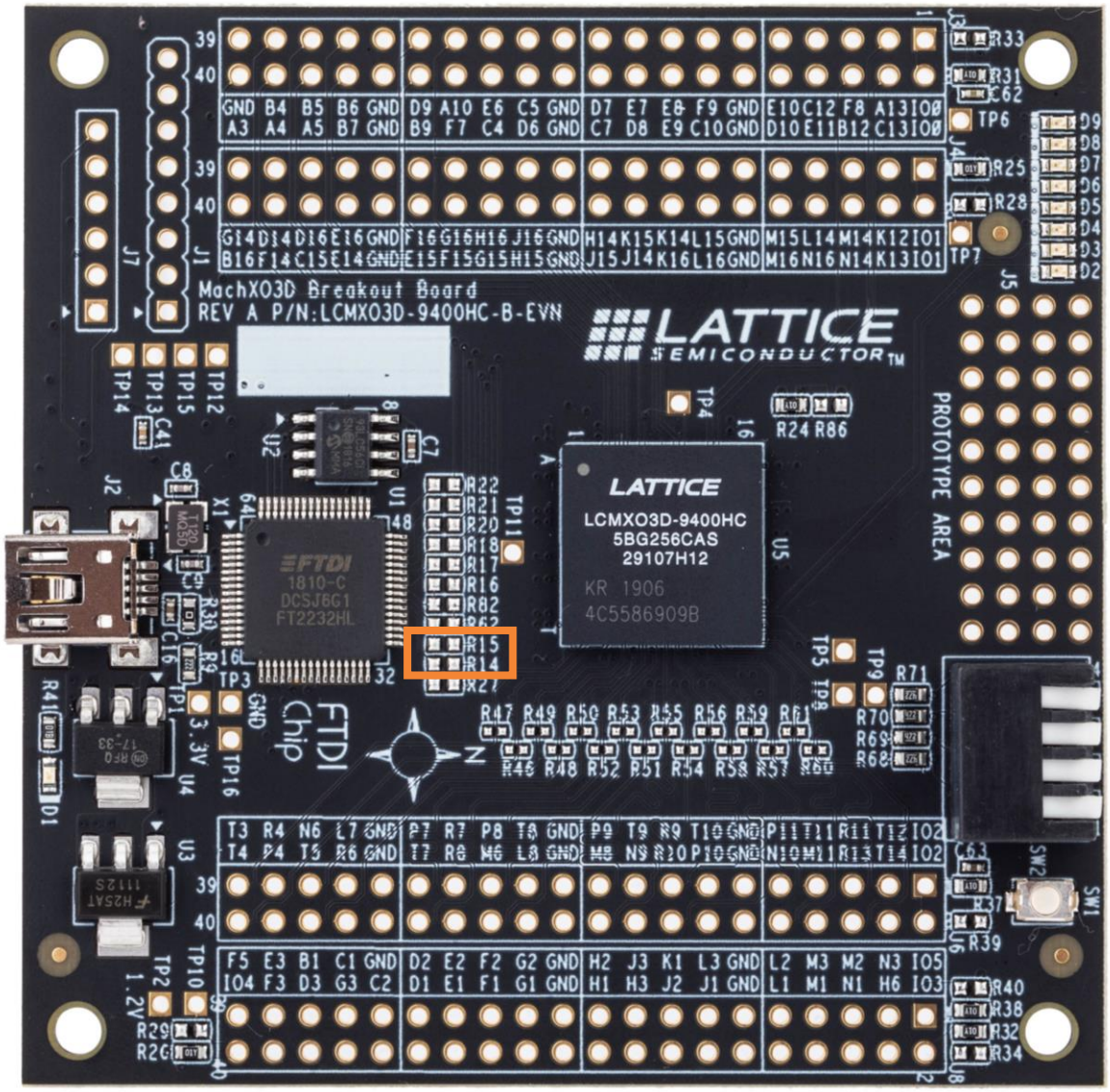


Figure 4.1. MachXO3D Breakout Board

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

2. Configure the FTDI device with FT_Prog software to enable the UART function.

   Connect the MachXO3D breakout board to host PC and power ON.

   Open the FT_Prog software. Select **DEVICES > Program**.

   Make sure Port B is configured as RS232 UART in Hardware and Virtual COM Port in Driver. See Figure 4.2.

   Select **DEVICES > Program** from the FT_Prog software again. From the opened Program Devices dialog, click **Program**.

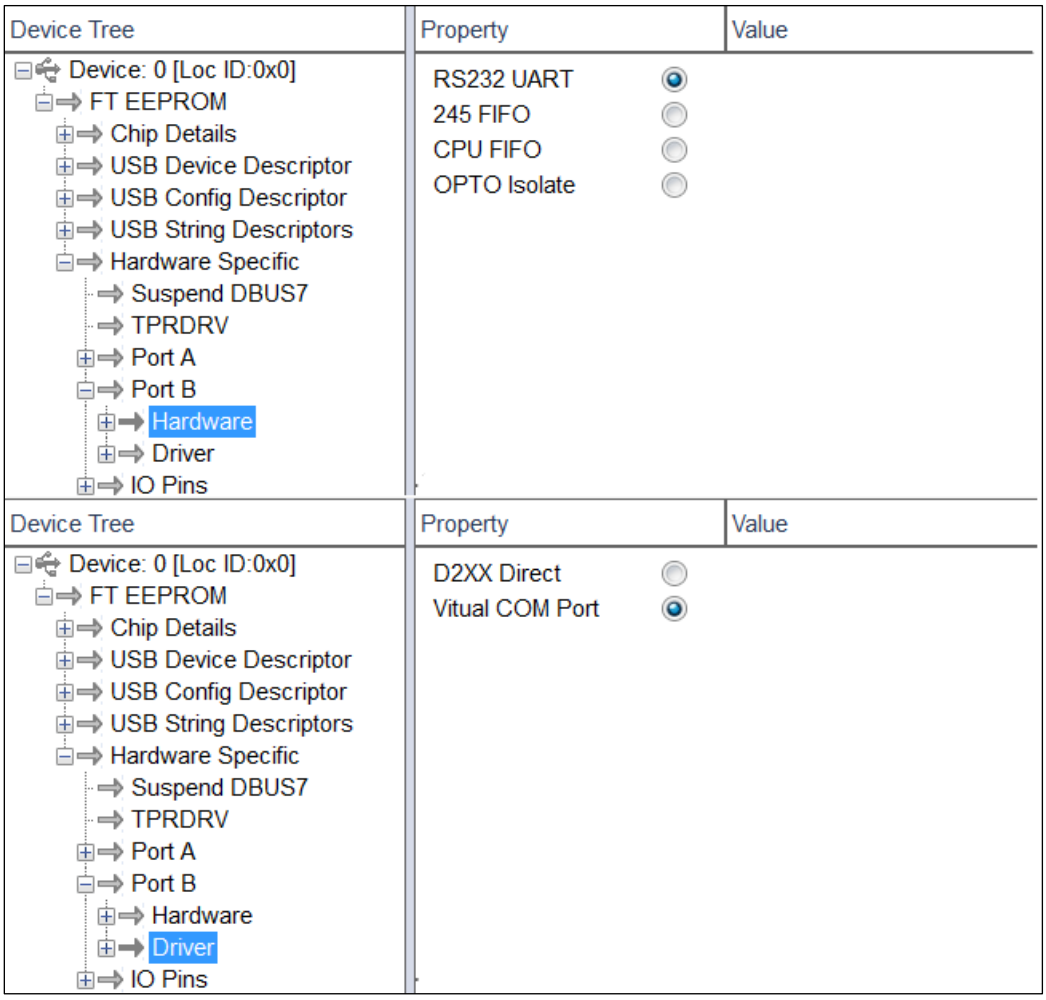


Figure 4.2 Configure the FTDI Device

3. All the reworks for the MachXO3D breakout board are completed.

4.1. Creating SoC Design Project and Preparing Hardware Design

To start an SoC Design Project from Lattice Propel:

1. Choose **File** > **New** > **Lattice SoC Design Project**.
2. The **Create SoC Project** wizard opens (Figure 4.3). Enter a project name, such as **HelloWorldSoC**. Select the **RISC-V MC SoC Project** template.
3. Click **Finish.** An SoC project is created.

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

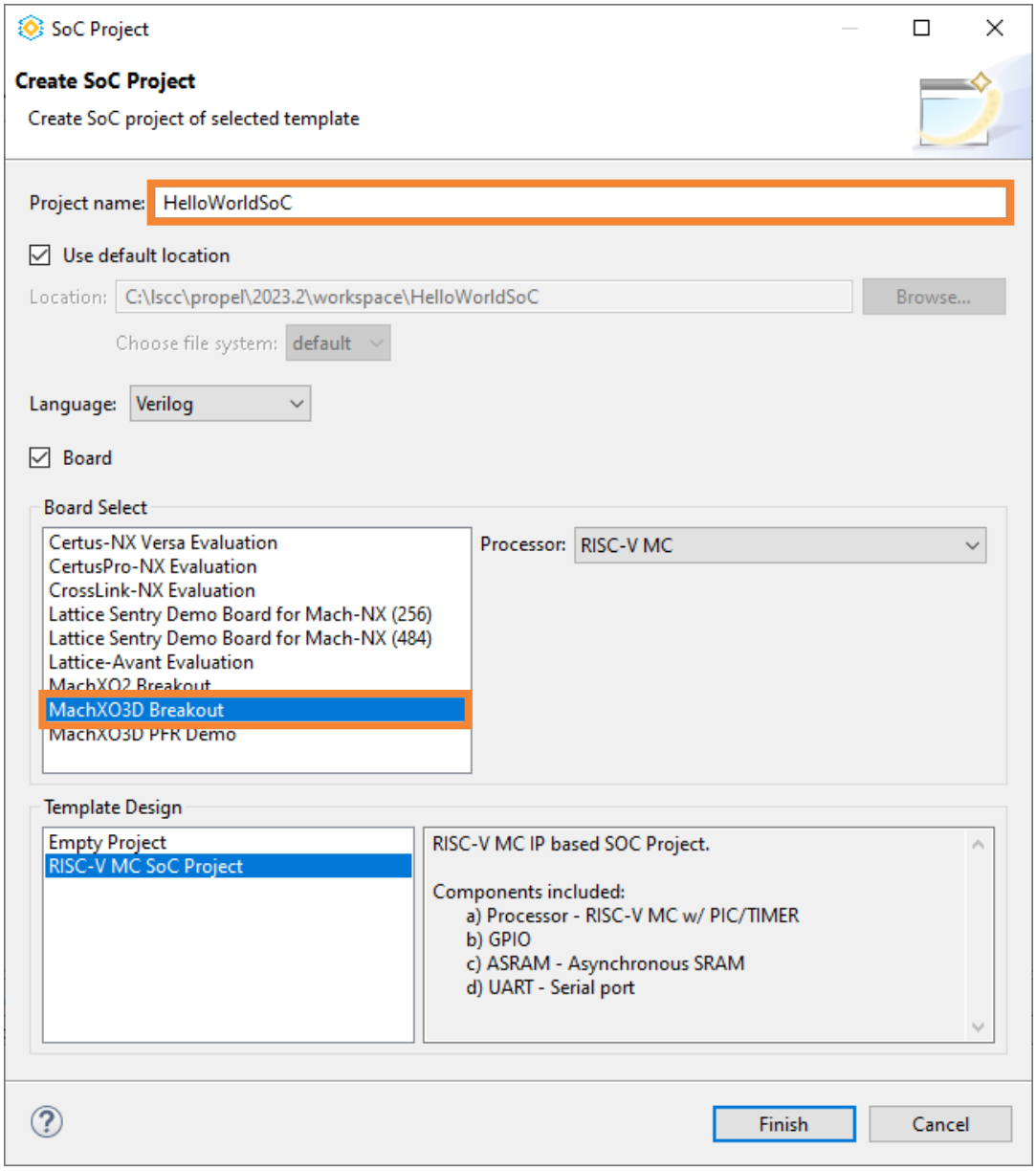


Figure 4.3. Create SoC Project Wizard

4. The created SoC project can be found in the workbench. Its design is opened and displayed in Propel Builder for review (Figure 4.4).

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

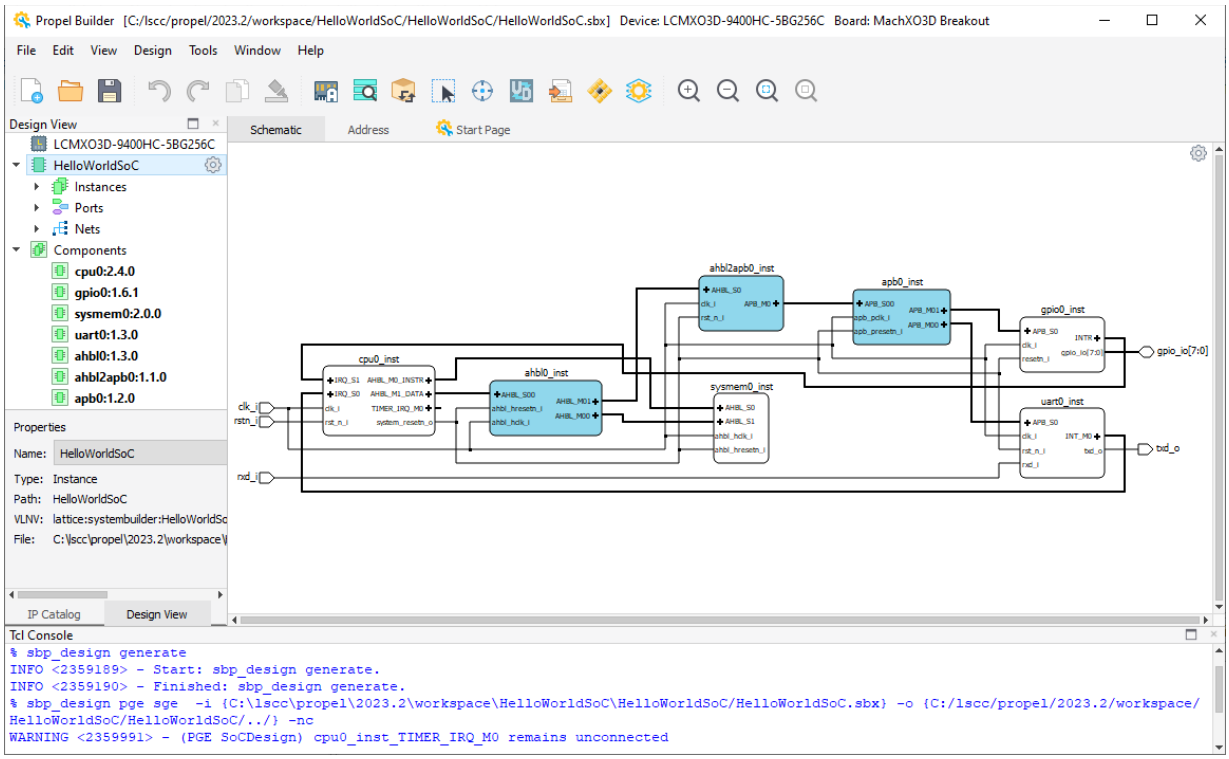


Figure 4.4. Propel Builder Window

5. (Optional) In Propel Builder, update the SoC design to set the preloaded software. Enable the checkbox for **Initialize Memory** for system_memory IP module from the **Initialization** area of the **General** tab, and set **Initialization File** (Figure 4.5) generated from the corresponding Hello World C project in the Creating Hello World C Project section.

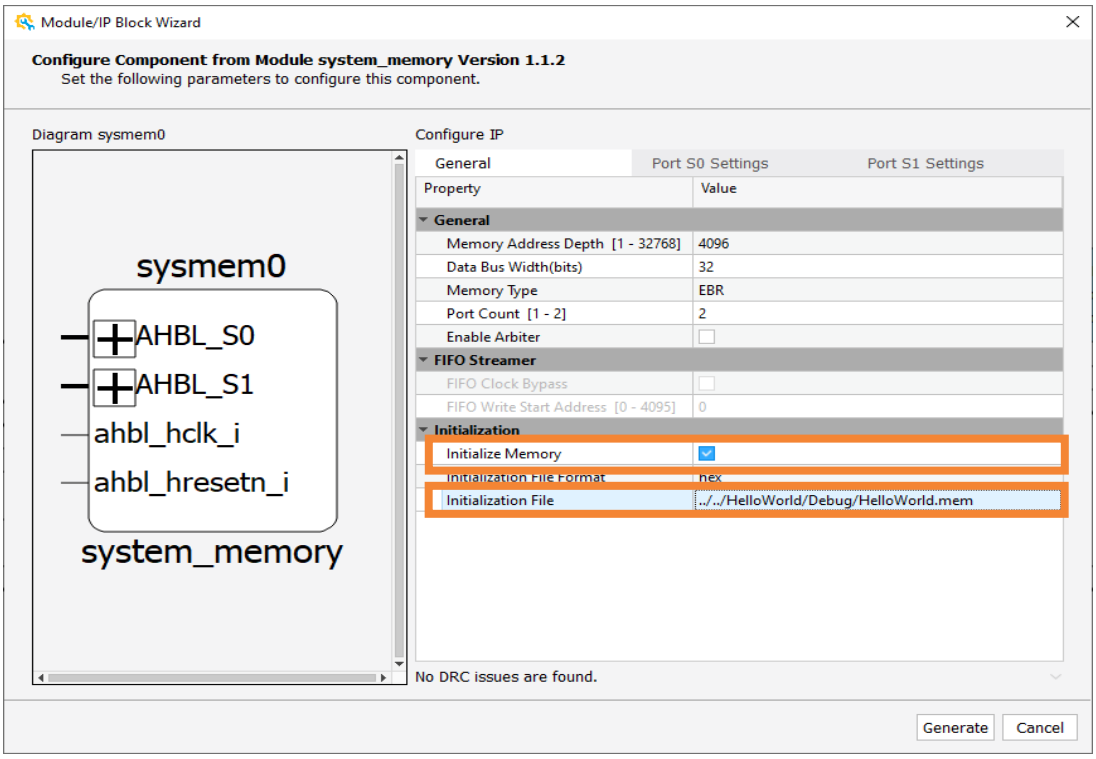


Figure 4.5. Configure Module System Memory

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

4.2. Launching Lattice Diamond

Launch Lattice Diamond from the created SoC project. To do that:

1. In Propel **Project Explorer** view, select the SoC project HelloWorldSoC.
2. Click the Diamond icon on the toolbar. A Diamond project is created and thus opened automatically in **Lattice Diamond**.
3. Switch to **Process** view of the Diamond project and make sure **Bitstream File** or **JEDEC File** is checked in the **Export Files** section (Figure 4.6).
4. Choose **Process** > **Run**. Wait for generating programming file successfully. You can see a green checkmark before each successfully completed process.

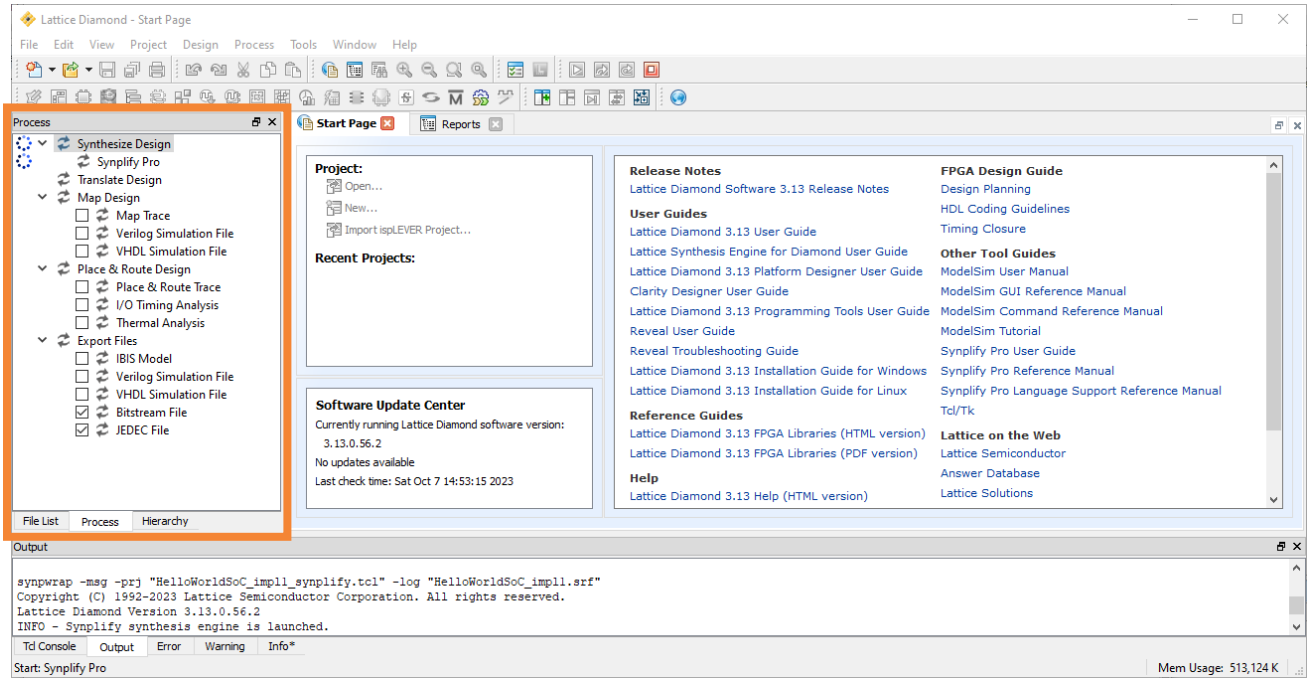


Figure 4.6. Generate Programming File

4.3. Programming the Target Device

Once the programming file is exported successfully in the last section, it is ready to program the target device. Make sure the evaluation board is powered ON and connected correctly to the host PC before performing the following procedure.

1. Click the **Programmer** icon on the toolbar of the Lattice Diamond Project Explorer.
2. The **Programmer: Getting Started** dialog pops up (Figure 4.7). Click **OK**.

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

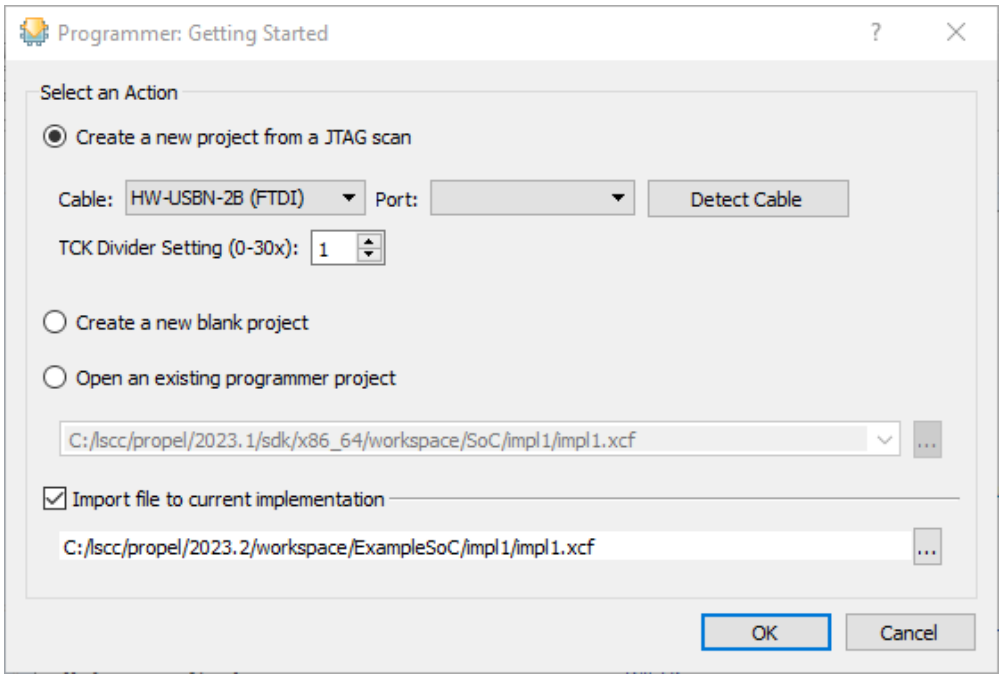
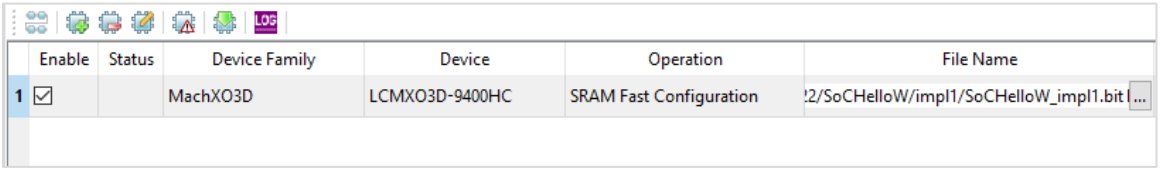


Figure 4.7. Programmer Getting Started Dialog

3. Review the **Device Family**, **Device**, **Operation**, and **File Name** in the Programmer window (as shown below).



4. Click the **Program** icon to download the programming data file to the device.

4.4. Creating Hello World C Project

Creating C project requires a system environment from SoC project as input.

1. In Propel **Project Explorer** view, select the SoC project HelloWorldSoC.
2. Choose **Project** > **Build Project**.

   System environment of the select SoC project is generated under the SoC project folder. Check the result from the **Console** view (Figure 4.8).

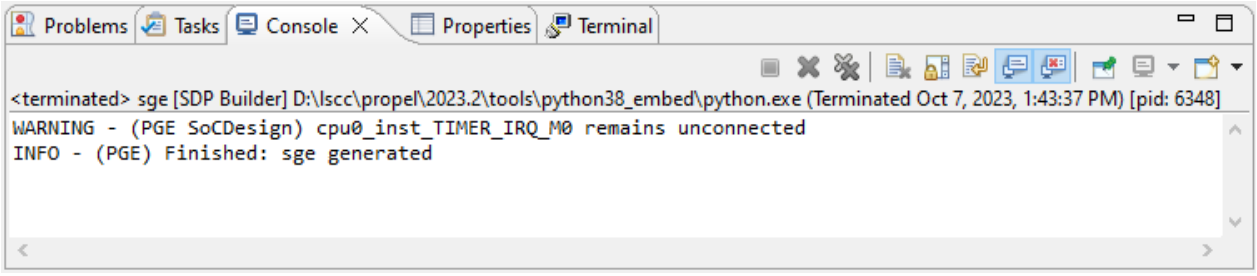


Figure 4.8. Build Result of HelloWorld SoC Project

3. Choose **File** > **New** > **Lattice C/C++ Project**.

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

The C/C++ Project wizard opens with the **Load System and BSP** page (Figure 4.9).

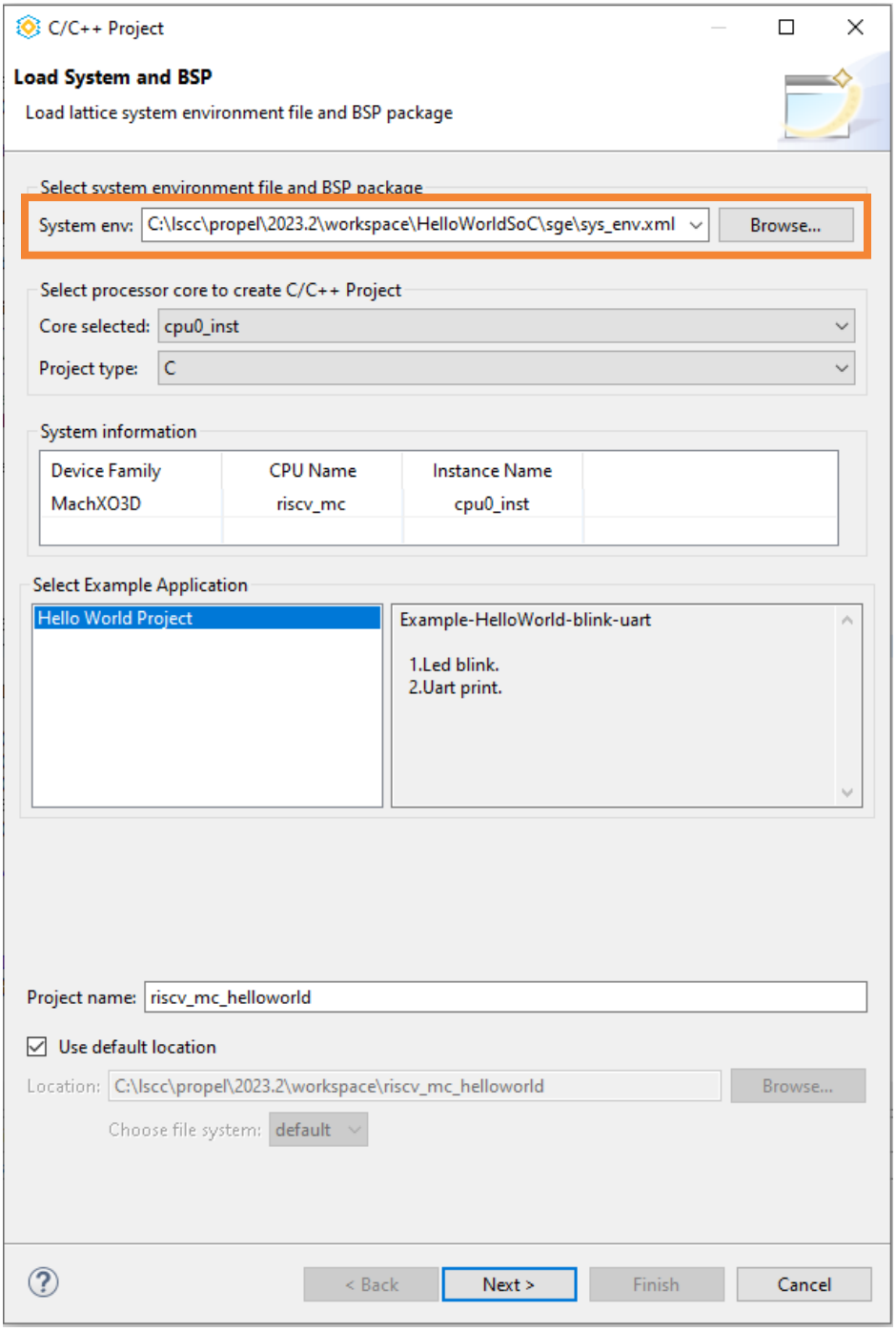


Figure 4.9. Load System and BSP Page

4. Select the system environment file just generated (Figure 4.9).
5. Change project name to HelloWorld (Figure 4.9). Click **Next**. Then click **Finish**.
   The C project is created and displayed in the workbench.
6. Choose **Project** > **Build Project**.
7. Check the build result from the **Console** view (Figure 4.10).

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

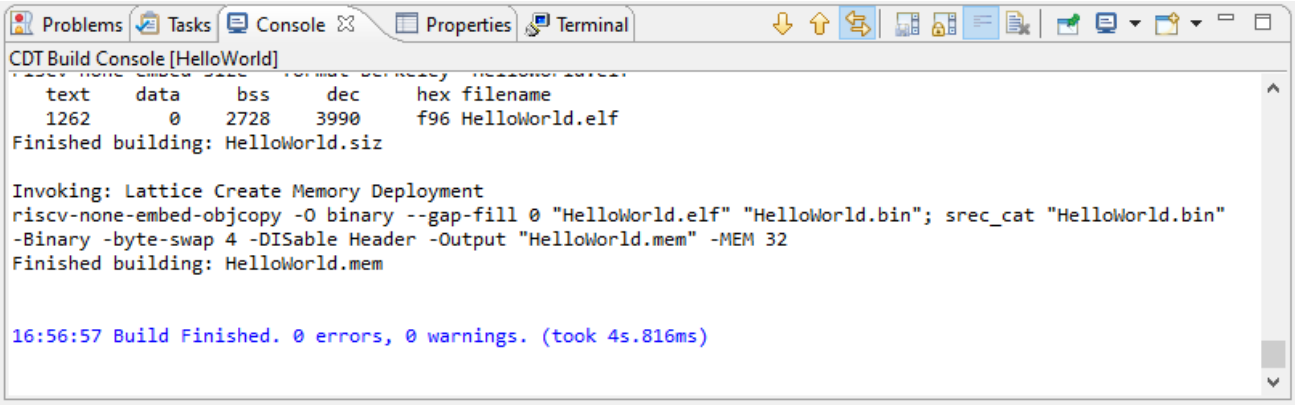


Figure 4.10. Build Result of HelloWorld C Project

4.5. Running Demo on MachXO3D Breakout Board – Hello World

1. Find the **Terminal** view nested to the **Console** view. If this view is not found, re-open it from **Window** > **Show View** > **Terminal**.
2. In the **Terminal** view, click the **Open a Terminal** icon .
3. Choose the **Serial Terminal** and configure the **Serial port** with **Baud rate** 115200 (Figure 4.11).

   Note: The serial port number depends on specific PC.

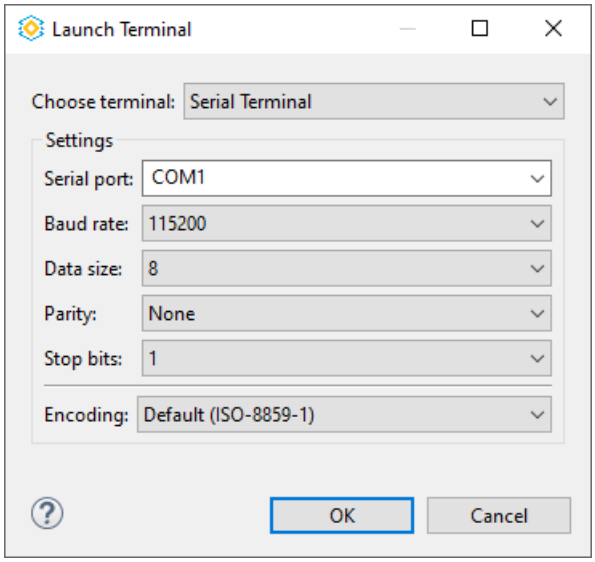


Figure 4.11. Launch Terminal Dialog

4. Click **OK**. A serial connection communicating with UART is ready.
5. In the **Project Explorer** view, select the C project, HelloWorld.
6. Choose **Run** > **Debug Configurations…**.
7. Double-click **GDB OpenOCD Debugging** to create a new launch configuration (Figure 4.12).
8. Click the **Debug** button.

   Wait for a few seconds for switching to the debug perspective, starting the server, allowing it to connect to the target device, starting the gdb client, downloading the application, and then starting the debugging session.

   Note: This demo uses the default debug configuration options.

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

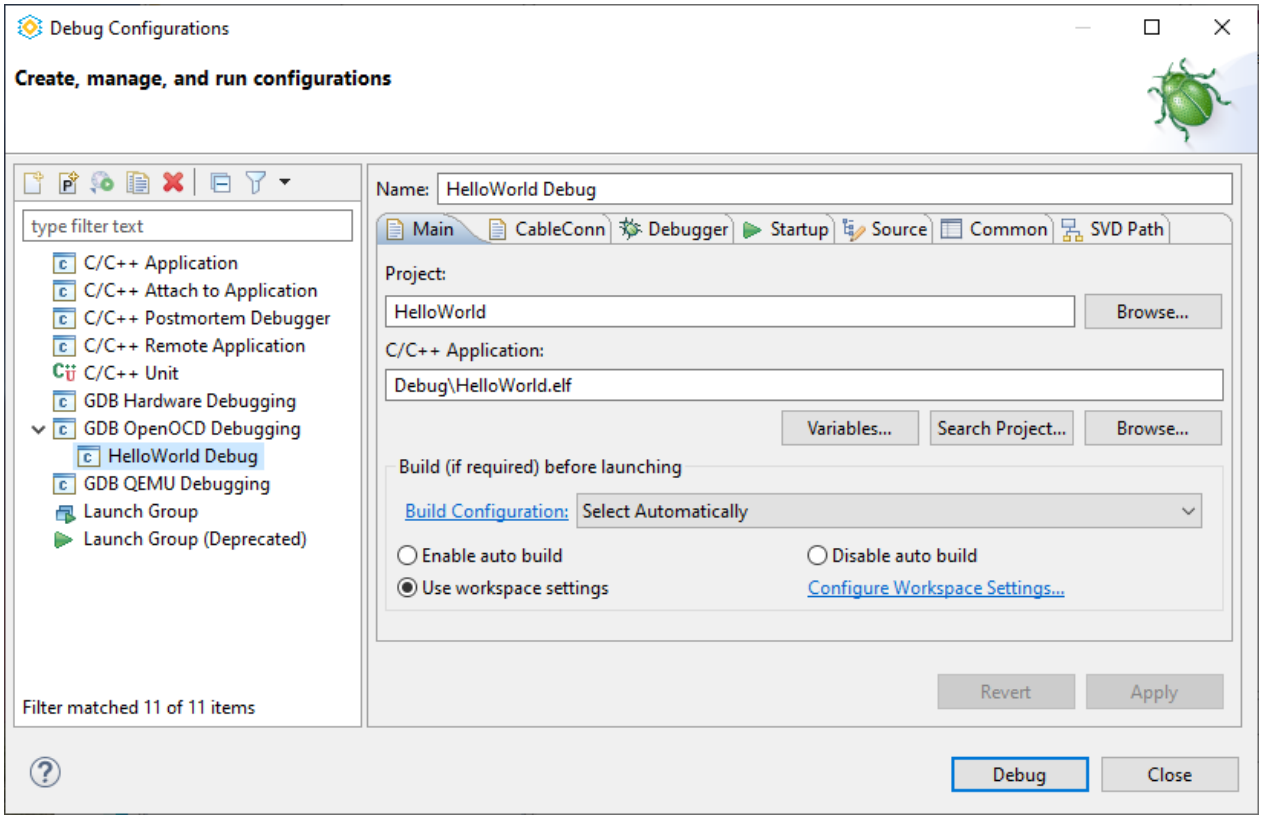


Figure 4.12. Debug Configurations Page

9. Click the **Resume** icon on the toolbar. The serial terminal outputs **Hello RISC-V world!** (Figure 4.13).

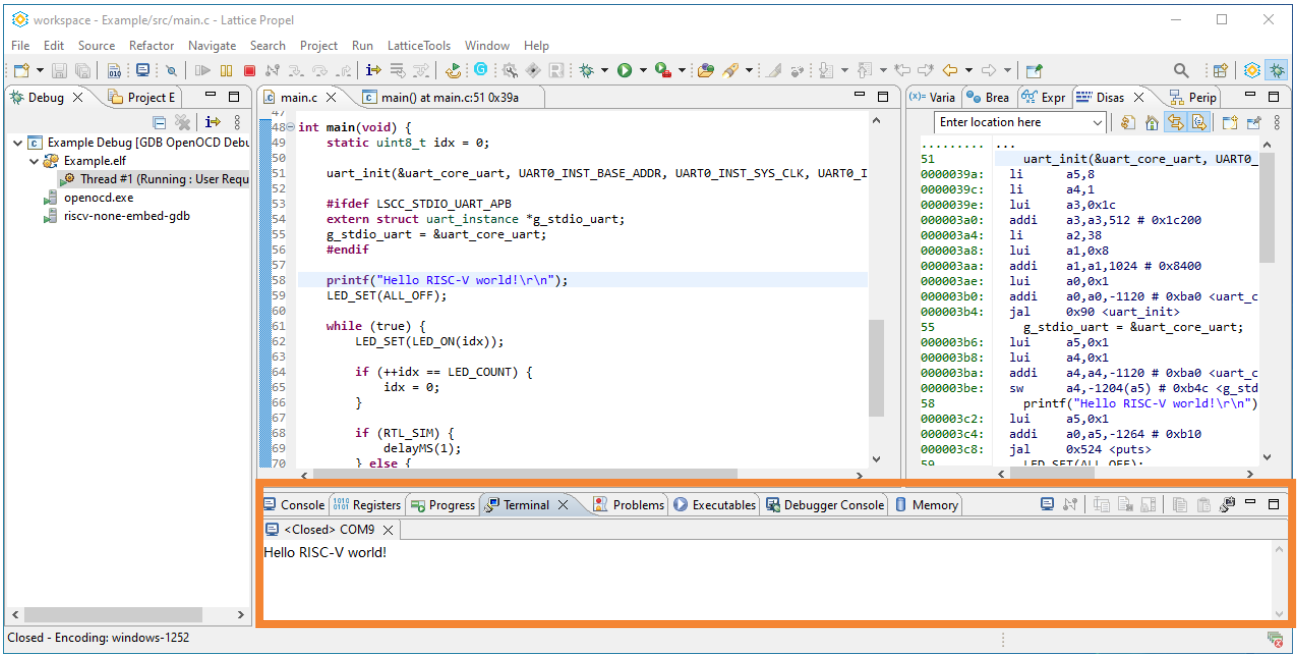


Figure 4.13. Run Result of Hello World Project

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

5. Propel Tutorial – Code Coverage

The code coverage function is supported in RISC-V RX Demo C Project and FreeRTOS C Project.

This tutorial uses a CertusPro-NX Evaluation Board with RISC-V RX SoC Project for demonstration.

The example C project is RISC-V RX Demo.

5.1. Creating RX Demo C project and Opening Code Coverage Functionality

1. The RX Demo C project can be created through the **Load System and BSP** page (Figure 5.1).

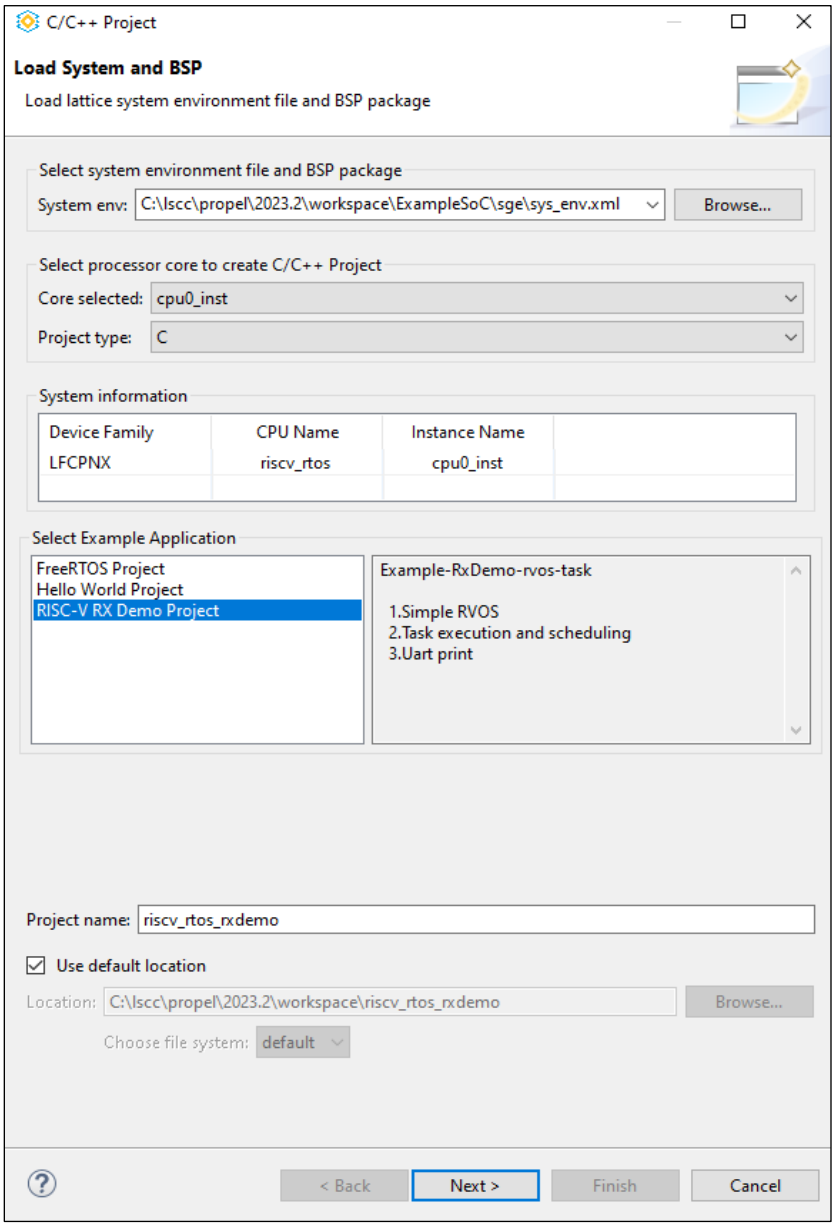


Figure 5.1. Create RX Demo C Project

2. The System env should use a RX core SoC project.
3. Select **RISC-V RX Demo Project** and check the project name.
4. Click **Next**. The Lattice Toolchain Setting dialog opens. Choose the **C/C++ Compiler** tab (Figure 5.2).

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

5. Select the checkbox **Generate gcov information**.
6. Click **Finish**.

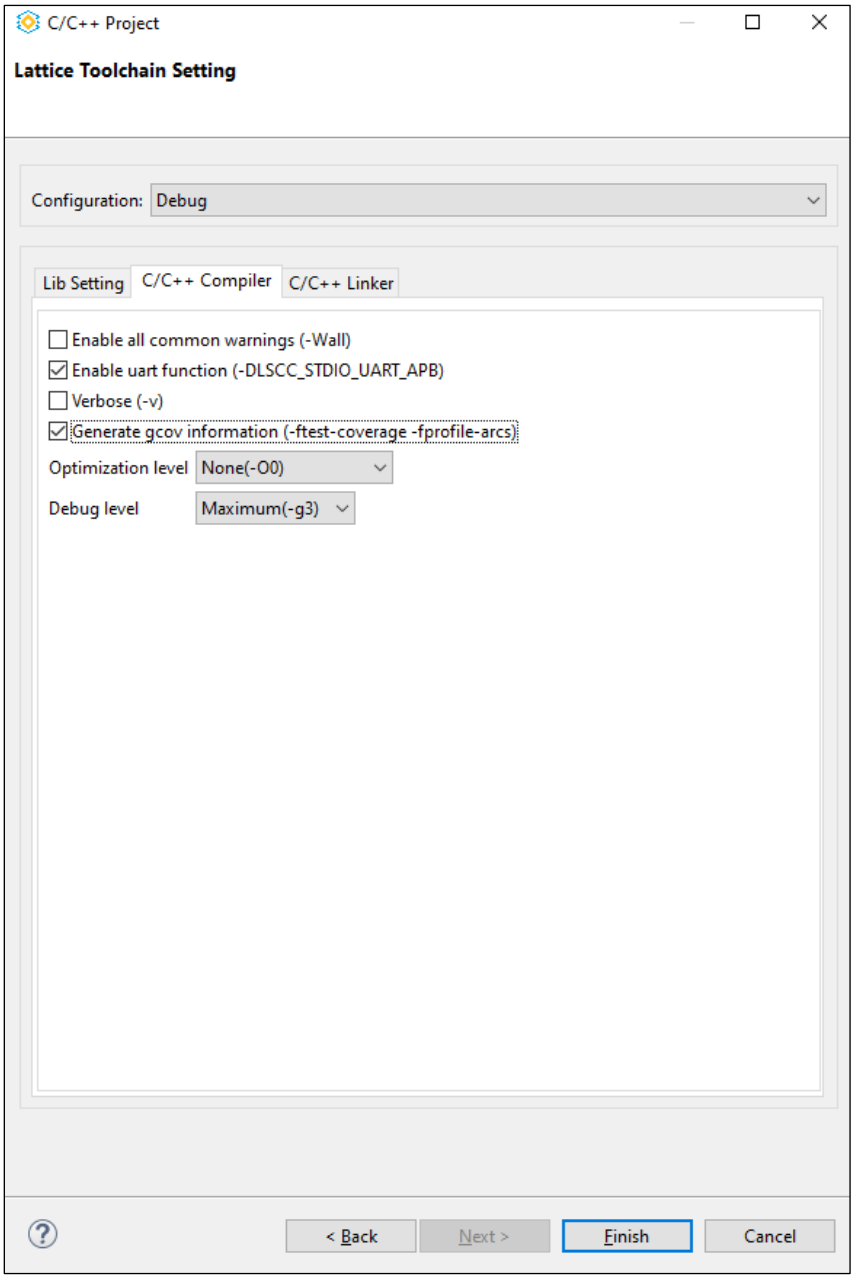


Figure 5.2. C/C++ Compiler

5.2. Prepare the Hardware and Programming the Target Device

1. Create RISC-V RX SoC Project with CertusPro-NX Evaluation Board.
2. Generate programing file by running Lattice Radiant in this SoC Project.
3. Program file to the target device, CertusPro-NX Evaluation Board.

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

5.3. Compiling and Running Demo

5.3.1. Compiling C Project

1. In the **Project Explorer** view, select the C project, riscv_rtos_rxdemo.
2. Choose **Project** > **Build Project**.

5.3.2. Running Demo

1. In the **Project Explorer** view, select the C project, riscv_rtos_rxdemo.
2. Choose **Run** > **Debug Configurations…**.
3. Double-click **GDB OpenOCD Debugging** to create a new launch configuration (Figure 5.3).
4. Click the **Debug** button.

   Wait for a few seconds for switching to the debug perspective, starting the server, allowing it to connect to the target device, starting the gdb client, downloading the application, and then starting the debugging session.

   Note: This demo uses the default debug configuration options.

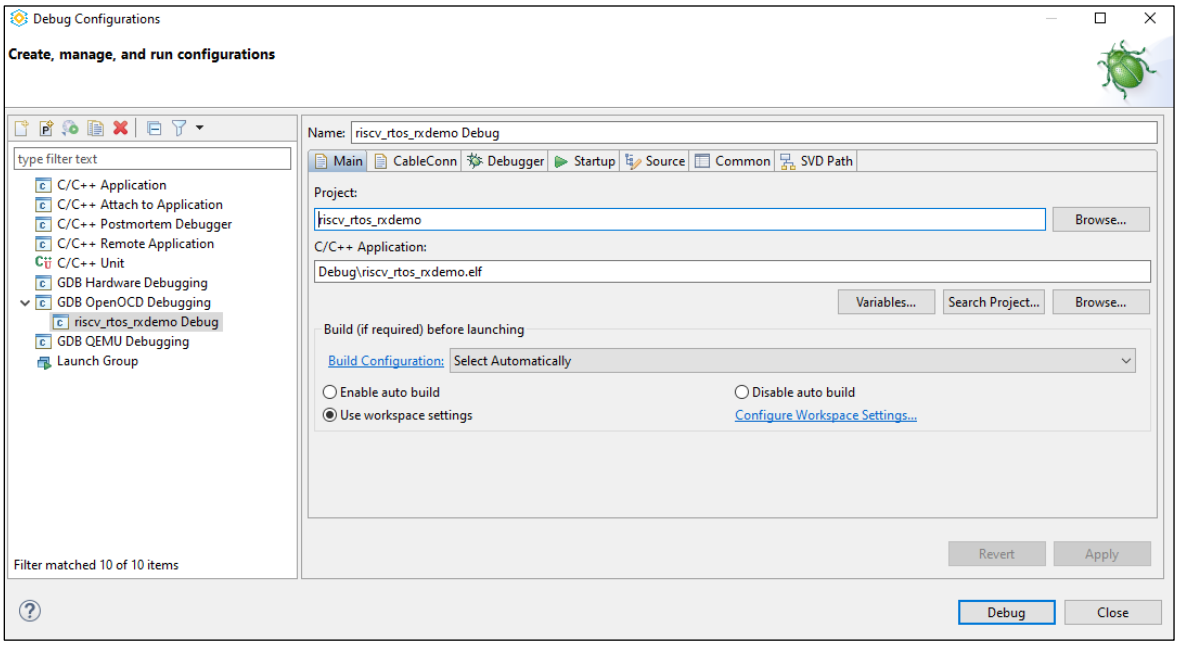


Figure 5.3. Debug Configuration

5. Choose **Run** > **Resume**. Then the console output logs.

   Wait for a few minutes, until the console logs: **do coverage data dump and coverage data dump done** (Figure 5.4).
6. Choose **Run** > **Terminate** to stop running.

5.3.3. Display Coverage Information

1. Check the coverage files (Figure 5.5).
2. Double click one of the coverage files (Figure 5.6) and click **OK**.
3. Wait for a few seconds, the coverage information is displayed (Figure 5.7).

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

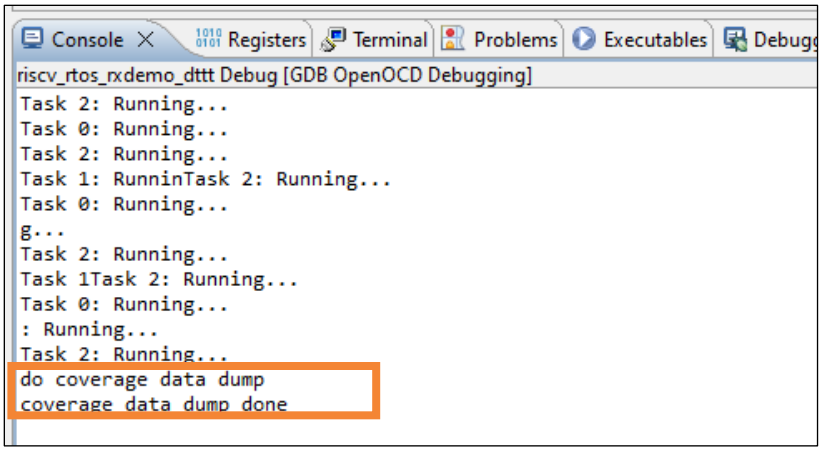


Figure 5.4. Console Logs

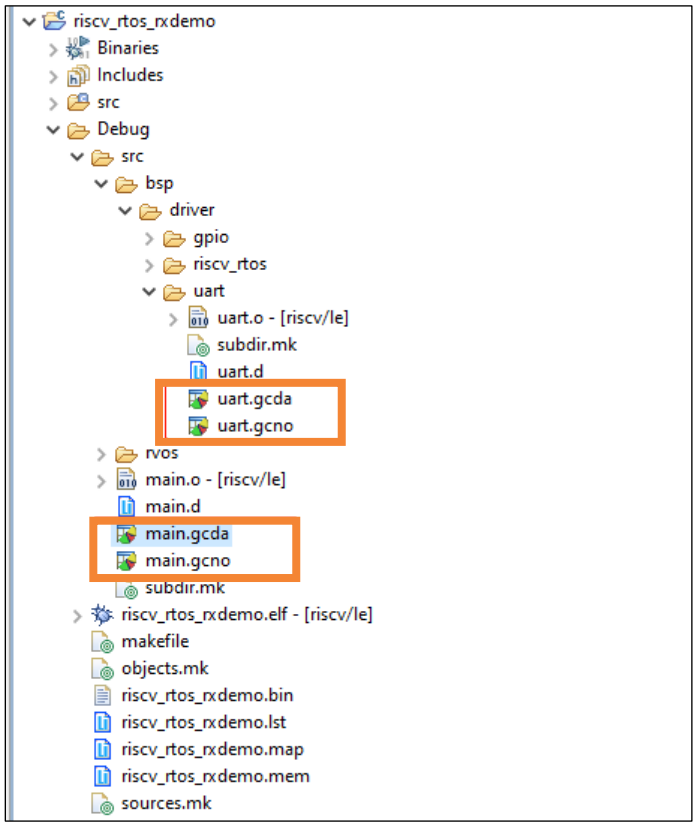


Figure 5.5. Coverage Files

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

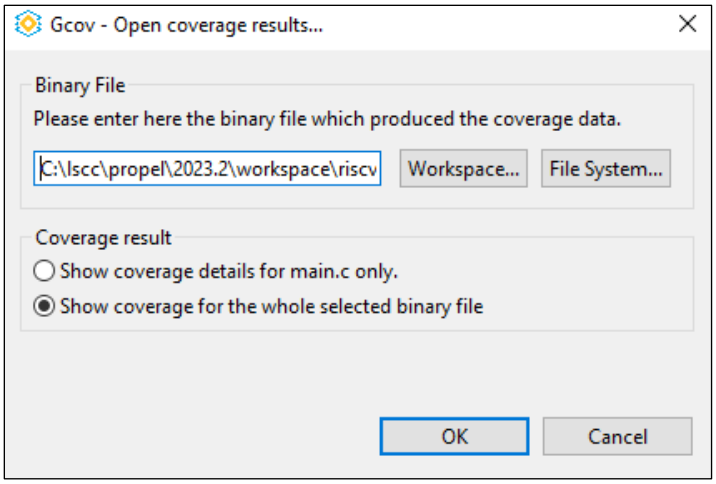


Figure 5.6. Open Coverage Results

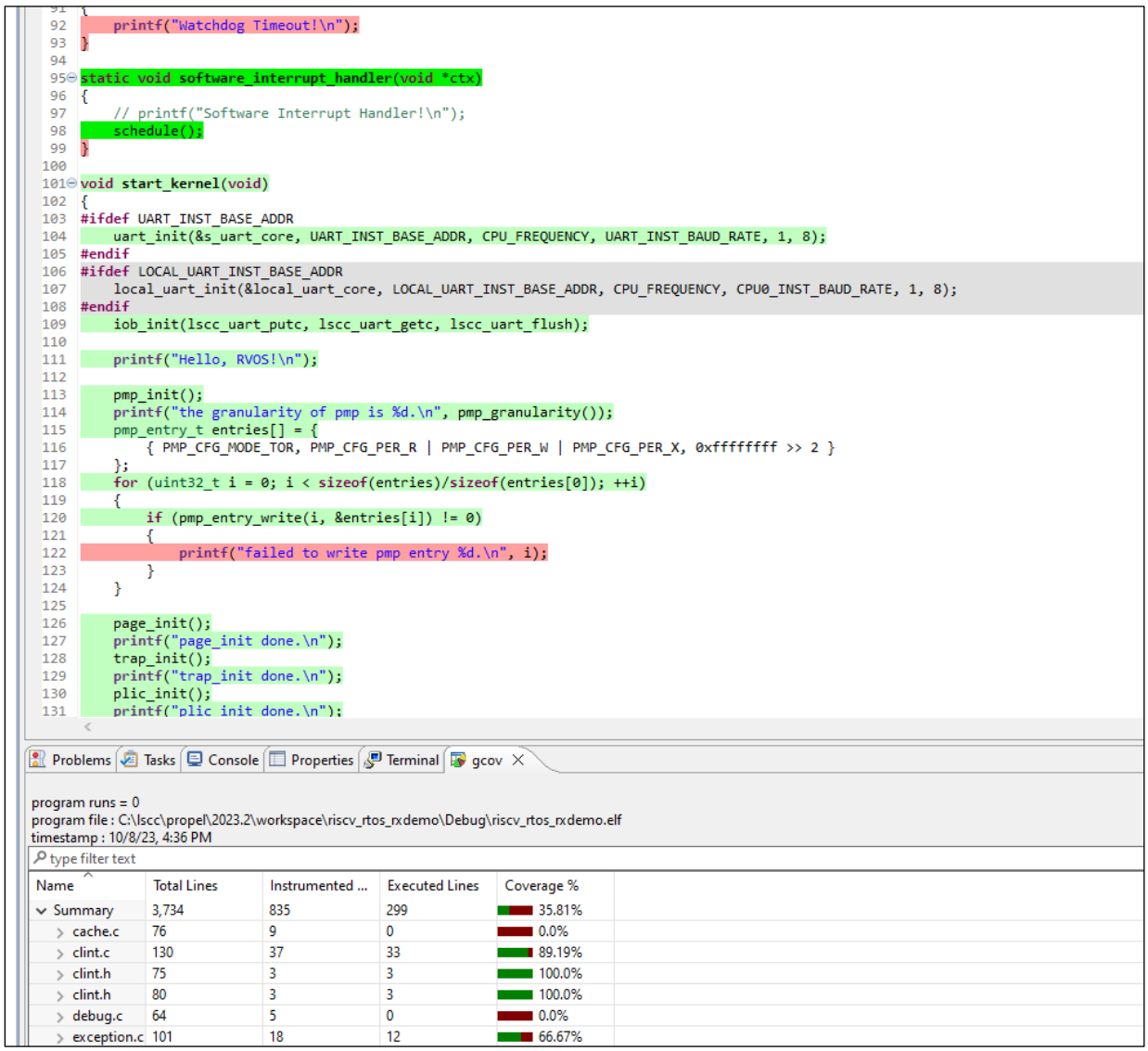


Figure 5.7. Code Coverage Information

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

5.4. How to Enable Code Coverage for Existing C Project

1. In the **Project Explorer** view, select the C project riscv_rtos_rxdemo_default as an example.
2. Choose **Project** > **Properties** > **C/C++ Build** > **Settings.**
3. Select **GNU RISC-V Cross C Compiler** > **Preprocessor**. Add the defined symbol, LSCC_COVERAGE (Figure 5.8).
4. Select **GNU RISC-V Cross C Compiler** > **Miscellaneous**. Add the compiler flag, -fprofile-arcs -ftest-coverage (Figure 5.9).
5. Select **GNU RISC-V Cross C Linker** > **Libraries**. Add the library, smallgcov (Figure 5.10).
6. Select **GNU RISC-V Cross C Linker** > **Miscellaneous**. Add the linker flag, --defsym=_HEAP_SIZE=0x1000 (Figure 5.11).
7. **Note**: The 0x1000 is a suggested value and it can be changed to a fit value if necessary.
8. Select **GNU RISC-V Cross C Linker** > **Miscellaneous**. Check link flags in the label, Other linker flags. Make sure the linker flag, --oslib=semihost, is supported (Figure 5.12).
9. Click **Apply and Close**.
10. Rebuild this C project.

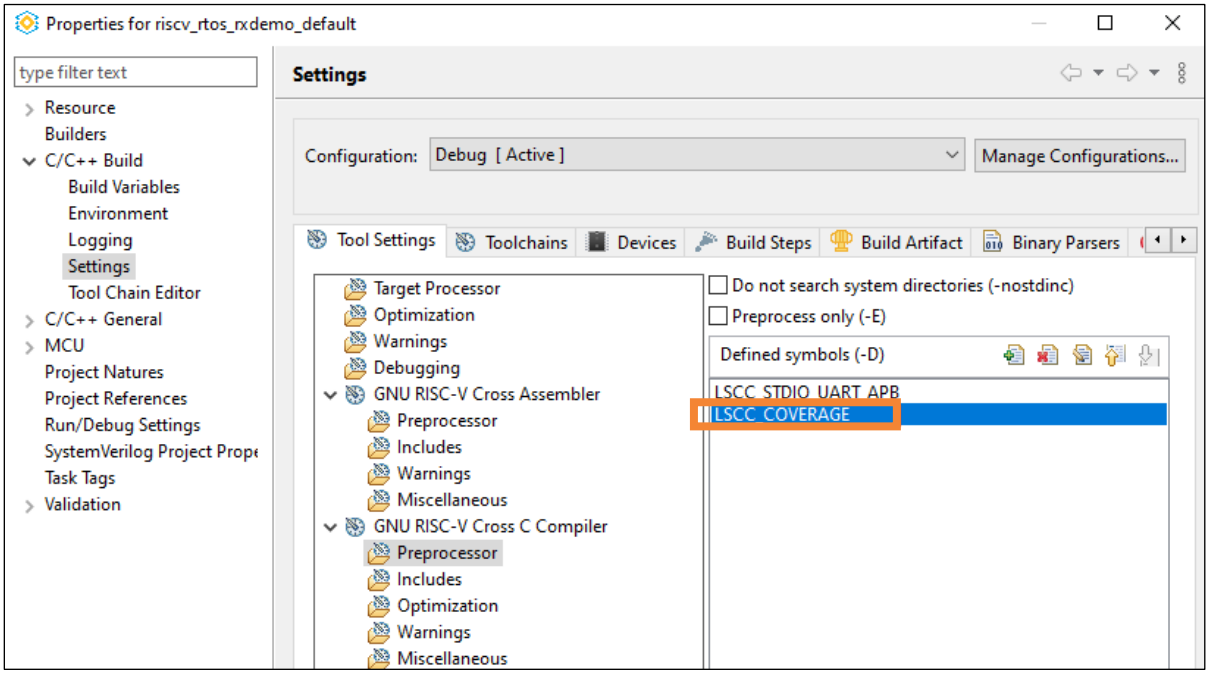


Figure 5.8. Defined Symbols

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

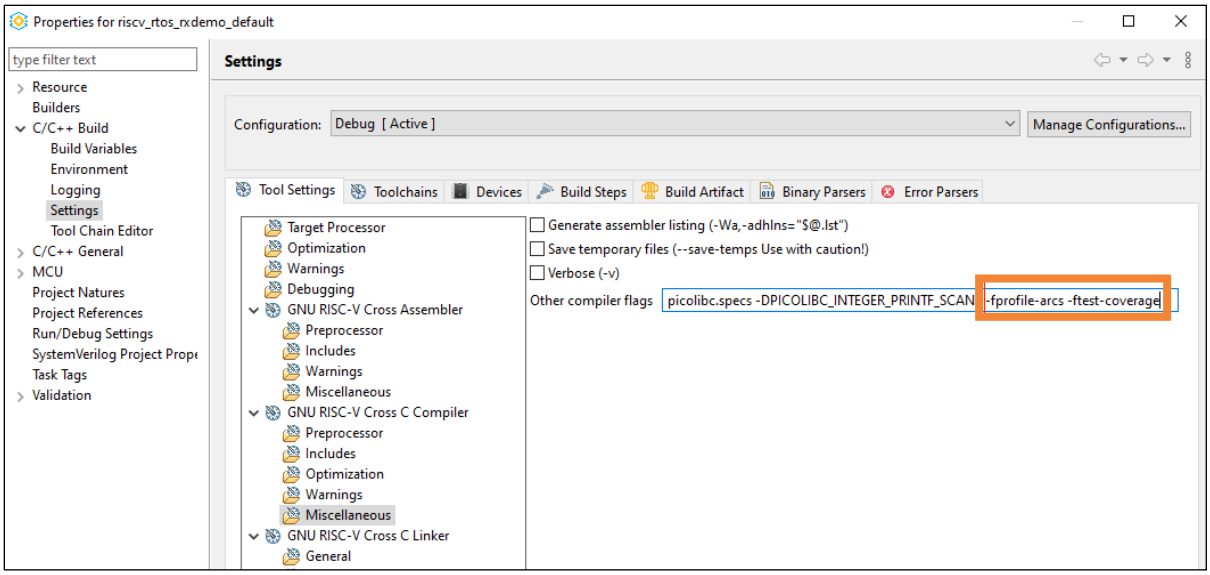


Figure 5.9. Compiler Flags

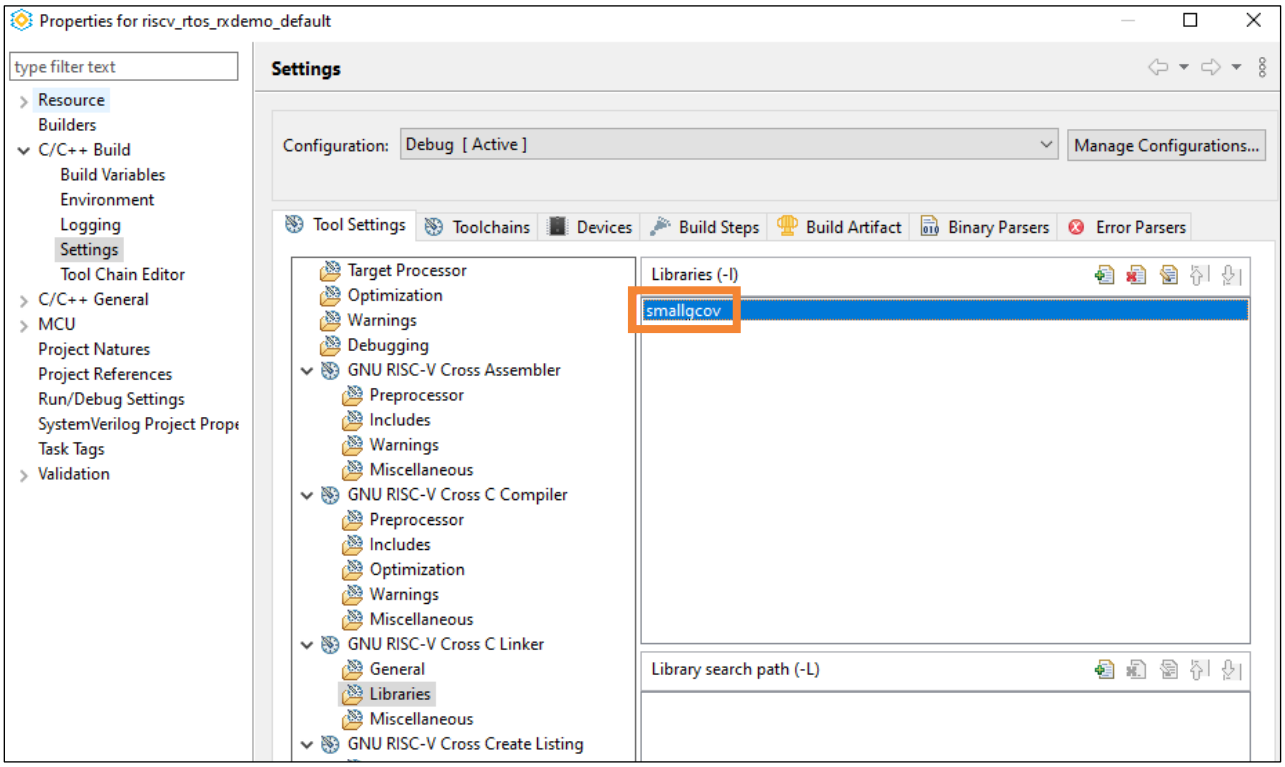


Figure 5.10. Libraries

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

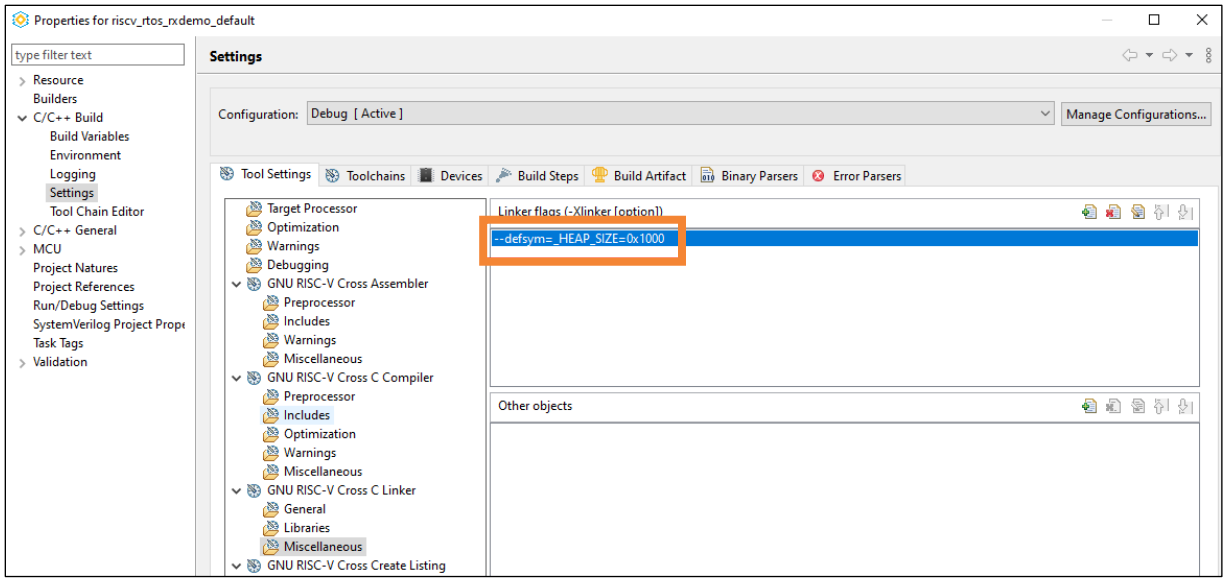


Figure 5.11. Linker Flags

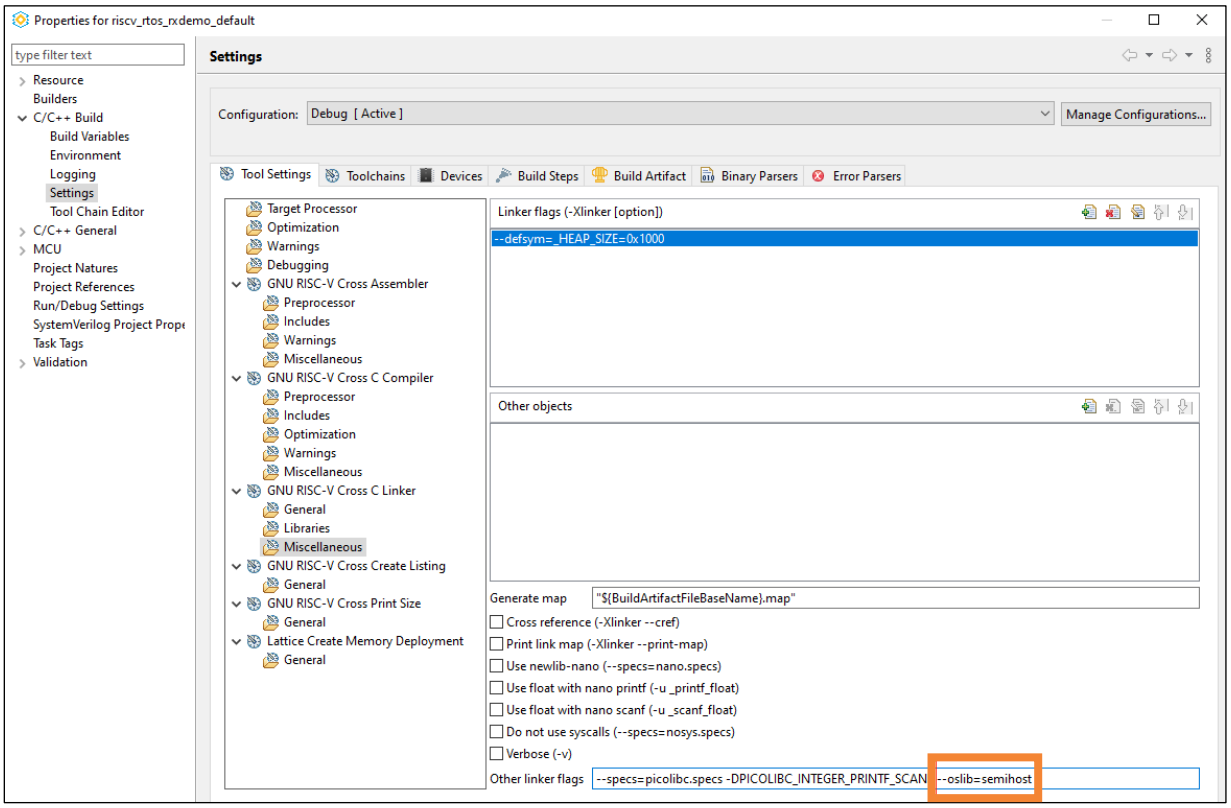


Figure 5.12. Other Linker Flags

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Appendix A. Linker Script and System Memory Deployment

During the Lattice C/C++ Project creation, Lattice Propel SDK generates a linker script file, linker.ld, within the project. This linker script file contains a memory region list parsing from the corresponding SoC design.

Note: Illegal memory regions are not imported to linker script. A memory region is considered illegal if it has any of the following conditions:

- No connection to CPU
- Address space conflict

Each memory region has a list of attributes to specify whether or not using a particular memory region for an input section (Figure A.1).

- r: Read-only section.
- w: Read/Write section indicating the memory region is connected to the data port of CPU.
- x: Executable section indicating the memory region is connected to the instruction port of CPU.

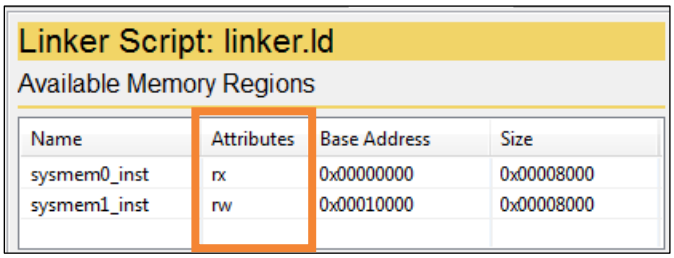


Figure A.1. Memory Regions in Linker Script

The generated linker script contains a mapping table of section pointing to the memory region (Figure A.2). Depending on the attributes of each memory region, code section and data section can point to the same or different memory regions.

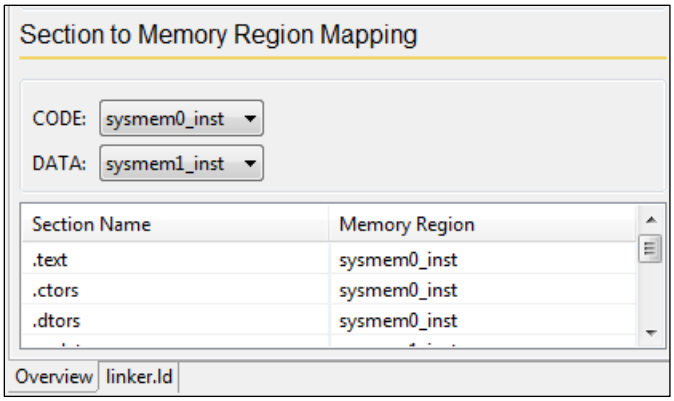


Figure A.2. Section to Memory Region Mapping

During the Lattice C/C++ Project building, Lattice Propel SDK generates Lattice system memory initialization files. Depending on the number of the memory regions used, it generates single memory initialization file or multiple memory initialization files. The following picture (Figure A.3) shows an example of multiple memory files being generated, separated for code and data segments.

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

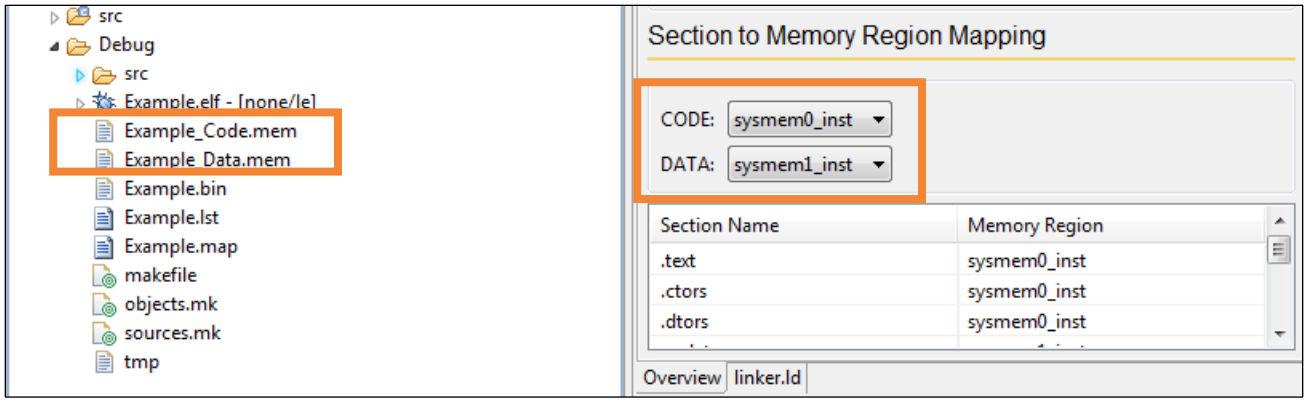


Figure A.3. Linker Script and Generated Memory Files

If you modify the linker script manually after the Lattice C/C++ project creation, especially changing the number of the used memory regions, the number of memory files generated during project building cannot be changed automatically. You can re-configure the generation of memory files in Lattice Propel through the following ways:

1. In the **Project Explorer** view of Lattice Propel, select a C/C++ project.
2. Choose **Project** > **Properties.** The **Properties** dialog opens showing the properties of the current project.
3. Select **Settings** of **C/C++ Build** category from the left pane. Select the **Toolchains** tab (Figure A.4).

   Check **Create memory file**, if you point the code and data segments to the same memory region. Or, check **Create multiple memory files**, if you point the code and data segments to separate memory regions.
4. Click **Apply**.

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

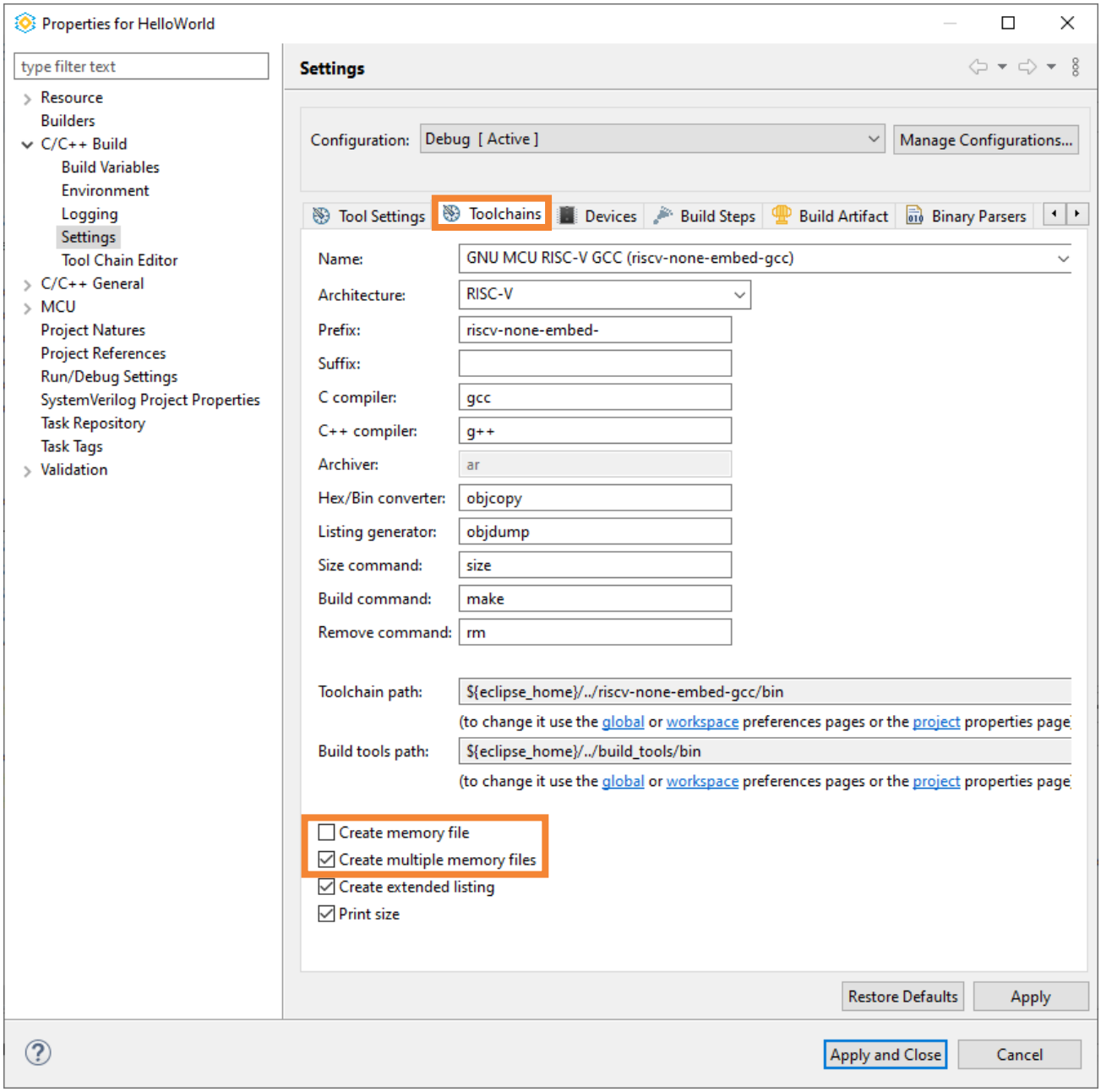


Figure A.4. Toolchains Tab of C/C++ Build Settings

5. Go back to the **Tool Settings** tab. The relevant Lattice memory deployment tools can be found (Figure A.5). Customize the tool options as needed.
6. Click **Apply and Close** to save the change.

   Note: The setting for each configuration, **Debug** or **Release**, is independent.

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

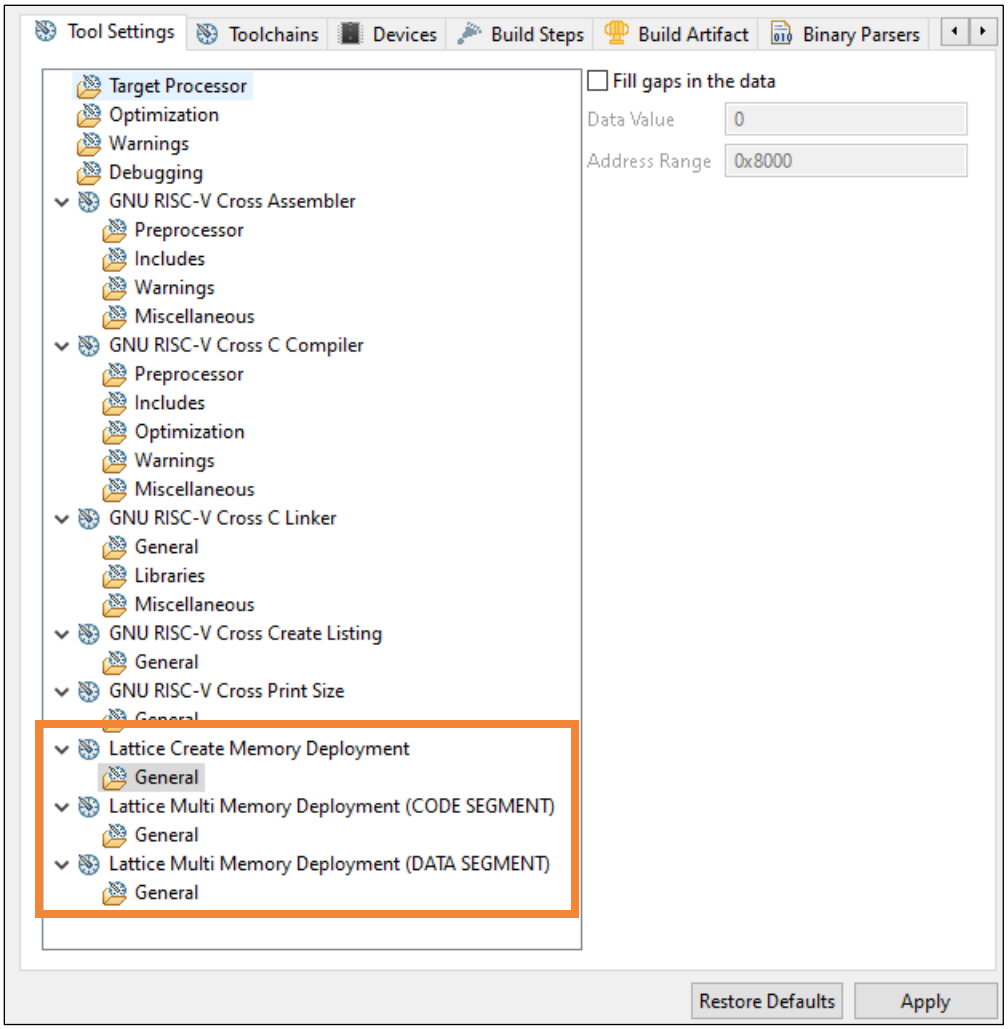


Figure A.5. Tool Settings Tab of C/C++ Build Settings

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Appendix B. Standard C Library Support

Lattice Propel SDK bundles Picolibc as the standard library support (https://keithp.com/picolibc/). Picolibc is a set of standard C libraries, both libc and libm, designed for smaller embedded systems with limited ROM and RAM. Picolibc includes code from Newlib (https://sourceware.org/newlib/) and AVR Libc (https://www.nongnu.org/avr-libc/).

Printf and Scanf Levels in Lattice Propel SDK

Lattice Propel SDK provides three levels of printf support with the support of Picolibc, which can be selected when creating a Lattice C/C++ project (Figure B.1), or be modified in the toolchain settings in the project properties after the project is created, noting that the options in both compiler and linker need to be set (Figure B.2 and Figure B.3).

- Integer only printf (-DPICOLIBC_INTEGER_PRINTF_SCANF). This is the default selection in Lattice Propel SDK, which removes the support for all float and double conversions to save code size.
- Float only printf (-DPICOLIBC_FLOAT_PRINTF_SCANF). It requires a special macro for float values: `printf_float`. To make it easier to switch between that and other two levels, that macro should also be correctly defined for the other two levels.

  Here is a sample program to demonstrate the usage:

```
#include <stdio.h>

void main(void) {
       printf(" 2^61 = %lld, Pi = %.17g\n", 1ll << 61,
printf_float(3.141592653589793));
}
```

- Full printf (-DPICOLIBC_DOUBLE_PRINTF_SCANF). This offers full printf functionality, including both float and double conversions.

System Library Interfaces Used in Lattice Propel SDK

Lattice Propel SDK provides three system library for stdio support with the help of Picolibc, which can be selected when creating a Lattice C/C++ project (Figure B.1), or be modified in the toolchain settings in the project properties after the project is created, noting that only the option in linker needs to be set (Figure B.3).

- Default. Use the default system library interface (UART) in BSP, this requires a UART instance inside the SoC design. The default library is implemented in the processor driver code and no additional linker options are required.
- Semihosting (--oslib=semihost). Semihosting is a mechanism that enables code running on the target to communicate with and use the I/O of the host computer. Lattice Propel SDK provides semihosting support in On-Chip-Debugging flow. It allows printing messages to the debugger console without relying on the UART instance, and it also supports file I/O.
- Dummyhosting (--oslib=dummyhost). Dummy stdio hooks. This allows programs to link without requiring any system-dependent functions. This is only used if the program does not provide its own version of stdin, stdout, and stderr.

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

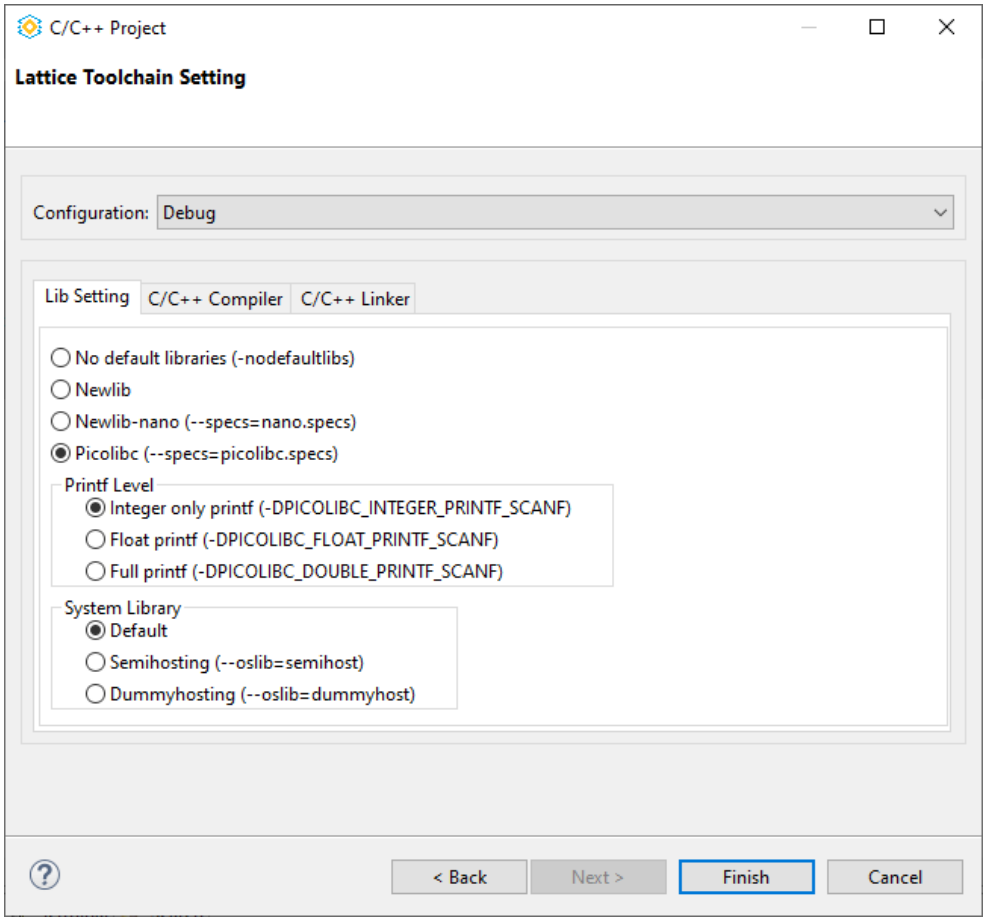


Figure B.1. Lattice Toolchain Setting Dialog

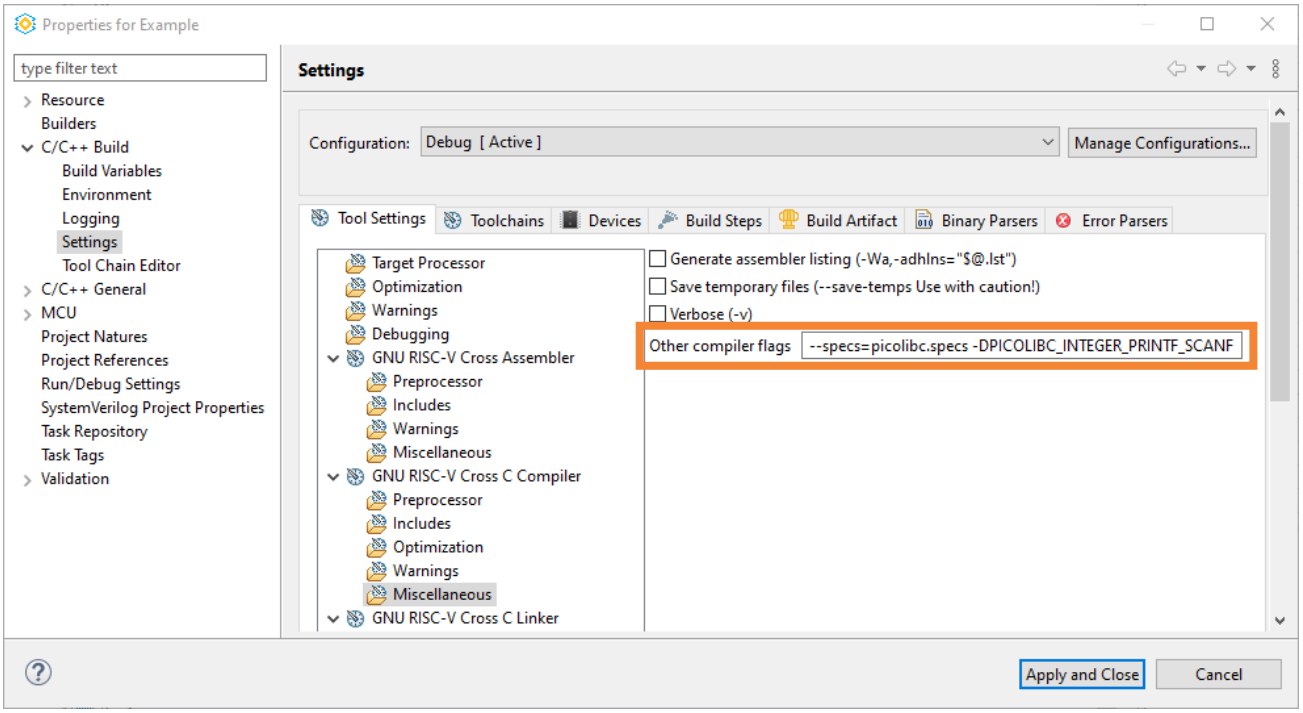


Figure B.2. Properties of C/C++ Project – Compiler Options

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

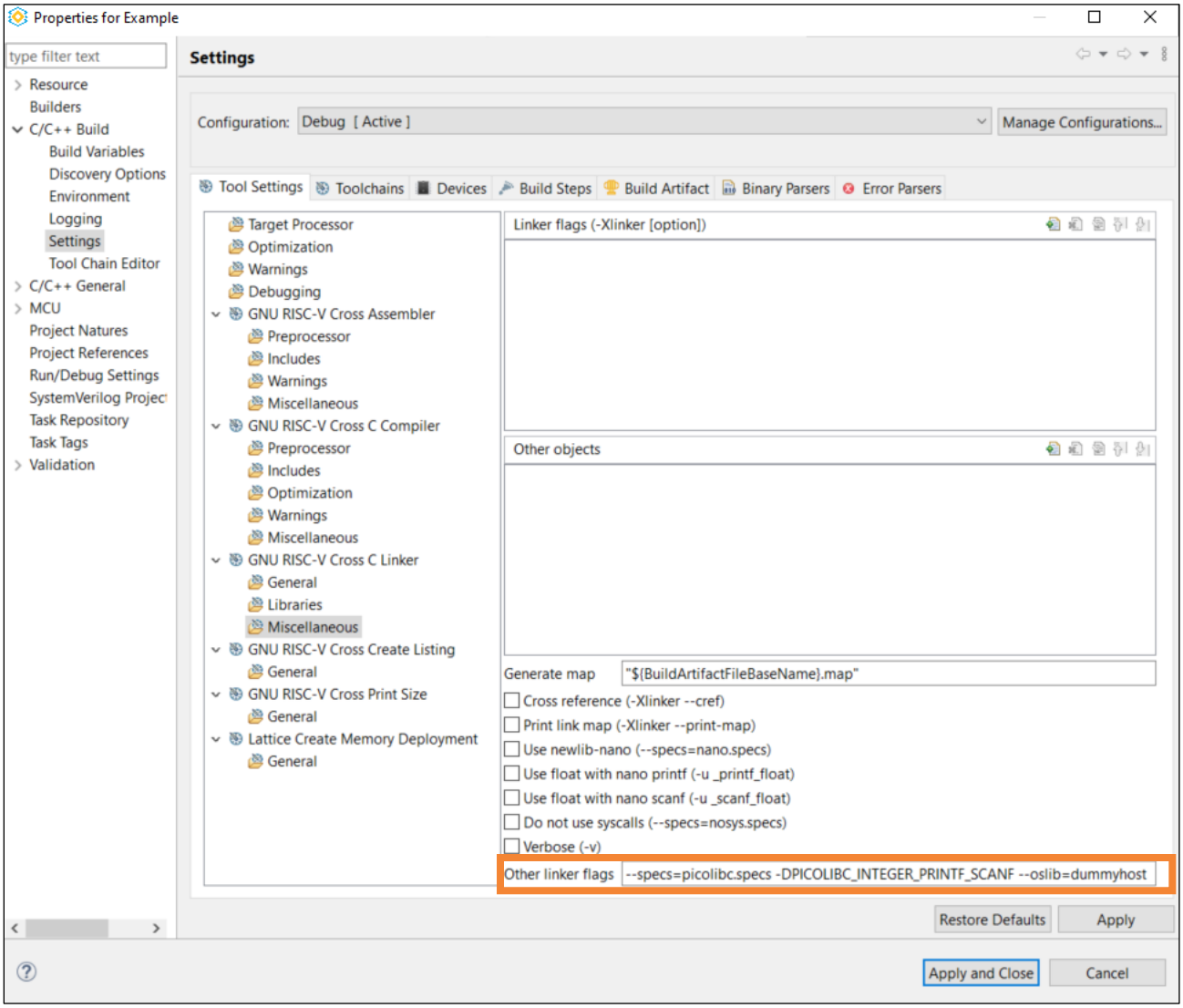


Figure B.3. Properties of C/C++ Project – Linker Options

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Appendix C. Third-party Command-line Tools in Lattice Propel SDK

Lattice Propel SDK integrates with a number of third-party command-line tools, which can be used with Propel User Interface or as stand-alone command-line tools. You can find these command-line tools and their documentation according to the following list.

1. Windows Build tools

   Version: 4.2.1-1

   Binaries Location: <*Propel Installation Location*>\sdk\build_tools\bin

   Documents Location: <*Propel Installation Location*>\sdk\build_tools\share

2. OpenOCD

   Version: 0.10.0

   Binaries Location: <*Propel Installation Location*>\sdk\openocd\bin

   Documents Location: <*Propel Installation Location*>\sdk\openocd\doc

3. GNU RISC-V Embedded GCC

   Version: 10.2.0-1.2

   Binaries Location: <*Propel Installation Location*>\sdk\riscv-none-embed-gcc\bin

   Documents Location: <*Propel Installation Location*>\sdk\riscv-none-embed-gcc\share\doc

4. Picolibc

   Version: 1.7.4

   Binaries Location: <*Propel Installation Location*>\sdk\riscv-none-embed-gcc\riscv-none-embed\picolibc\riscv-none-embed\lib

   Documents Location: <*Propel Installation Location*>\sdk\riscv-none-embed-gcc\doc

5. SRecord

   Version: 1.6.4

   Binaries Location: <*Propel Installation Location*>\sdk\tools\bin

   Documents Location: <*Propel Installation Location*>\sdk\tools\bin

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Appendix D. Command-line Environment Setting Script in Lattice Propel SDK

Lattice Propel provides a script to set necessary environment variables for running all Propel SDK command line tools. It enables you to fast get up to using command line tools, especially for Lattice specific tools.

To use the script:

1. Run the script propel_commandline.cmd (Windows) or propel_commandline.sh (Linux) under <*Propel Installation Location*>.

   The message Lattice Propel Commandline is printed, and it lets you go to a new command line interface.
2. All command line tools in the Propel SDK can be run from this command line interface without any special configuration.

   Here are examples of using the command line tools within this script.

   Example A: Build a C project under command line.

```
$ cd <C project Location>/Debug
$ make
```

   Example B: Launching openocd under command line.

```
$ cd <C project Location>
$ openocd -c "gdb_port 3333" -c "set target 0" -c "set tck 1"  -c "set port
FTUSB-0" -c "set RISCV_SMALL_YAML src/cpu0.yaml" -f interface/lattice-cable.cfg
-f target/riscv-small.cfg
```

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

References

- Lattice Propel 2023.2 Release Notes (FPGA-AN-02068)
- Lattice Propel 2023.2 Installation for Windows User Guide (FPGA-AN-02069)
- Lattice Propel 2023.2 Installation for Linux User Guide (FPGA-AN-02070)
- Lattice Propel Builder 2023.2 Usage Guide (FPGA-UG-02196)
- IP Packager 2023.2 User Guide (FPGA-UG-02197)
- Lattice Propel Revision Control User Guide (FPGA-UG-02198)
- MachXO3D Family Data Sheet (FPGA-DS-02026)
- MachXO3D Programming and Configuration Usage Guide (FPGA-TN-02069)
- MachXO3D Breakout Board User Guide (FPGA-UG-02084)

For more information, refer to:

- Lattice Propel Web Page
- Lattice Insights for Training Series and Learning Plans

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

For frequently asked questions, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Revision History

Revision 1.0, November 2023

Section	Change Summary
All	Production release.

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

www.latticesemi.com