**RISC-V Class Project Phase 1 – Quick Start Tutorial**

In the Class Project we will create a full implementation of a RISC-V processor and simulate it, using the Codal Simulation environment from Codasip. Codasip is a full-featured commercial product which allows developers to quickly define processor architectures in a relatively high level language called Codal, create test software in the assembly language of the processor and easily simulate and debug the test software and the processor implementation. This document describes a series of steps to create a Project in Codasip containing a processor which is a subset of the RISC-V architecture. To complete Project Phase 1, you will need to:

1. Import the initial hardware implementation
2. Build the hardware
3. Import a software test for it
4. Build the software test
5. Run a simulation of the test software on the hardware to verify the implementation
6. Export the Project for evaluation
7. **Overview of Codasip**

The Codasip tool is designed to accept the definition of a processor in a specific language called Codal. Once the processor is described, Codasip creates a number of critical and compatible pieces necessary to implement the processor. These include:

1. A simulator (referred to as the Debugger in Codasip) of the defined hardware implementation which executes software in the defined machine language format. The simulator is either for an Instruction Accurate (IA) hardware model which executes each instruction fully in one clock cycle, or a Cycle Accurate (CA) hardware model which includes a fully specified hardware pipeline.
2. An assembler which converts the defined assembly language format to the defined machine language format.
3. A disassembler which converts the defined machine language format to the defined assembly language format.
4. A profiler which allows measurement of the performance of a software program on the defined hardware.
5. A C Compiler which converts standard C language into the defined assemble language format.
6. A Verilog description of the processor which may then be used by further tools to implement the processor in a physical form such as an integrated circuit or an FPGA.
7. **Prepare the Server Environment**

The Codasip software is run on a server farm supported by the ECEN Department. The following VPN software from CU’s OIT department must be downloaded and installed in order to access the server. The VPN must be connected to enable any server access.

* 1. <https://oit.colorado.edu/services/network-internet-services/vpn>
  2. Please use "vpn.colorado.edu" as the Domain Name
  3. Log in with your Identikey and the password you use with it

We will use the multi-user aspects of Windows Server 2016 to allow access to a group of student servers and a central machine for transmitting results. There are a number of steps required to initialize a connection to the servers. Initially the Windows implementation will be described, as that has been the most thoroughly tested. MAC and Linux implementations are similar.

* 1. **Windows**

1. Copy the certificate ecee-ca.cer from the Canvas Class Project/Phase 1 folder to your Desktop.
2. Double click the downloaded file ecee.ca.cer.
3. Click Open.
4. In the Certificate window, click on “Install Certificate…”.
5. Install for “Local Machine”.
6. Allow changes for your machine.
7. Select “Place all certificates in the following store”.
8. Select “Browse”.
9. Select “Trusted Root Certification Authorities” and click OK.
10. Click “Next” and “Finish”. You should get a message “The import was successful”.
11. Copy the Remote Desktop file cpub-ECEE-CA-ECEE-CA-CmsRdsh.rdp from the Canvas Class Project/Phase 1 folder to your Desktop.
12. Execute the file by clicking (or double-clicking) it.
13. You may see an error dialog about the publisher. Check “Don’t ask me again for connections to this computer”, and then click Connect. This will generally occur only the first time.
14. This opens a Remote Desktop session. The first time you do this it will ask for your username, which must be “ad\Identikey”. Enter your password and click OK. You are now logged into the server system.
15. You will see the message “Welcome to ECEE”. Click OK. You will now see the server desktop.
    1. **Apple MAC**

For a Mac, download a Remote Desktop application and follow the directions in Section 2.1 starting with item 12). Let the Professor know if you run into any roadblocks.

* 1. **Ubuntu Linux**

In a Linux environment (we believe all environments should work), download a Remote Desktop application and follow the directions in Section 2.1 starting with item 12). Let the Professor know if you run into any roadblocks.

1. **Access the Codasip Software**

Codasip is initiated by double clicking the “Codasip Studio 7.1.2 EVALUATION” desktop icon. When Codasip starts, a Workspace must be provided. The workspace can be of any name but must be in the folder C:\Users\Identikey. The first time a workspace is selected, it will be created. On subsequent initiations, the workspace can be selected from the pulldown menu.

Codasip uses a standard Eclipse Integrated Development Environment (IDE) to provide the user interface. See <http://help.eclipse.org/oxygen/index.jsp?nav=%2F0> for the complete Eclipse documentation. A basic understanding of the Eclipse environment is assumed, and the documentation includes basic Tutorials for those with no Eclipse experience.

Eclipse includes several levels of elements in an environment called a Workbench. The top level of the Workbench is a Perspective, which defines the display layout for different operations. Each Perspective includes a number of Views or Windows, which may be moved and/or resized as desired. Each View may contain multiple tabs.

1. **Import the Initial Project**

Import this project starting with the top level menu:

File -> Import -> General -> Existing Projects into Workspace -> Click Next

Select Archive file -> Browse -> G:/Information/Phase 1/Phase1\_orig.zip, then click on Finish.

Note that this process will be used whenever a Project is to be Imported. This will include test programs and any other generally useful material. All current Projects will be shown in the Project Explorer View and hardware projects will also appear in the Codasip Tasks View. All Projects will be made available in the Information folder.

Figure 1 below shows the initial Codasip Eclipse Workbench after some projects have been Imported. The initial Perspective is the Codasip Perspective. It consists of four Views:

1. The Project Explorer View on the left is where all projects are listed. These can be expanded by folder.
2. The Editor View in the center is where files are edited. Any file may be opened in the Editor View by double-clicking it in the Project Explorer View. The initial element in the Editor View is the Codasip Dashboard, which allows access to some special Codasip tools.
3. The Codasip Tasks View on the right is where most build tasks are executed. Open the Project by left clicking on the arrow to the left of the Project name. A dropdown menu of tasks appears. A task is executed by double-clicking the button to its left.
4. The Console View on the bottom is where messages from builds and runs are displayed.

A screenshot of a cell phone

Description generated with very high confidence

Figure 1

1. **Copy and Rename the Project**

You must create a unique project name for every Project in order to submit your work for evaluation. This project name **MUST** be in the form “standardname”, where standardname is your first initial followed by your last name (all lower case with no spaces) followed by the number of the project phase with no space. For the first phase the project phase **MUST** be “1”. Submissions will not be accepted with any project name which does not follow this format.

Select the Phase1\_orig Project by left clicking on it, then enter CTRL-c to copy it and CTRL-v to paste it (you can also right click and select Copy and Paste). Type the correct project name, standardname format, into the dialog box that appears. This is the actual Project file that you will use, but having Phase1\_orig for a reference is often useful. Figure 2 below shows the Workbench after the project has been copied (to “clind1” in this case).

**A screenshot of a cell phone

Description generated with very high confidence**

Figure 2

1. **Build the Hardware Project**

For Phase 1, start by building the hardware model, the assembler and the simulator for use in this Project. In the Task window, perform the following tasks:

1. Expand the desired Project standardname1 by double clicking on it or by clicking the > symbol. Figure 3 shows the expanded project.

A screenshot of a social media post

Description generated with very high confidence

Figure 3

1. Double click the button to the left of Model Compilation (ia), where the “ia” indicates the Instruction Accurate model we are building. A flashing arrow appears indicating that it is building. Any errors or warnings will be indicated in the Console window. Once the task is complete with no errors, a check mark will be placed over the button in the Task window. Note that there may be orange Warnings in the Console window, which is OK.
2. Double click the button to the left of Assembler (ia) to build the assembler. The warning about the Linker shown in Figure 4 can be ignored.

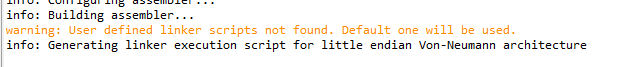


Figure 4

1. Double click the button to the left of Disassembler (ia) to build the disassembler.
2. Double click the button to the left of Simulator (ia) to build the simulator.
3. **Import and Build the Test Program**

The next step is to Import the test program into the Project window, which is the phase1\_test project in the Information/Phase 1 folder.

1. Import the test program with File -> Import -> General -> Existing Projects into Workspace -> Select archive file -> Browse -> G:/Information/Phase 1/phase\_test1.zip. Click on Finish.
2. The phase1\_test project will now appear in the Project Explorer window, so expand it as shown in Figure 5 below.
3. Right-click the first grayed line next to the gear and select “Change SDK”. Select the standardname1.ia SDK, which will be used to build the software test. Click OK.
4. Expand the src folder by clicking on the arrow to the left of it, as shown in Figure 5.
5. The assembler source code for the test is in the phase1\_test.s file. Double click to open it in the Editor View so that you can review the code if necessary.
6. Build the test software by selecting the project (the phase1\_test line) and either executing Ctrl-b or right-clicking and selecting Build Project. This will build the software and produce an object file in the Binaries folder.

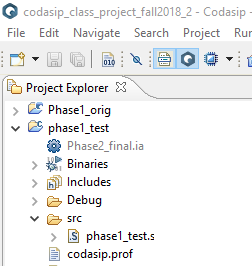


Figure 5

1. If the build of the test program produces the error shown in Figure 6, the build was attempted when a simulation was still running. Go to the Debug Perspective and stop the simulation with the red square button.

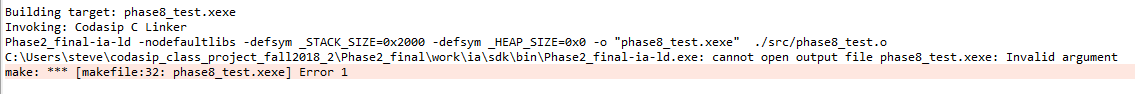


Figure 6

1. **Search Function**

A very valuable Eclipse function is searching, executed by clicking Search in the menu bar and selecting “File”. This will bring up a dialog box where a text string can be entered in the Containing text: area. Clicking Search will open a new tab in the Console window with an entry for every occurrence of the string in any project, displayed in the project hierarchy. Only the last instance found is displayed, but the hierarchy can be expanded to show all occurrences.

1. **Simulate the Test Program**

The next step is to simulate the test program.

1. Select from the ribbon at the top of the Codasip Workbench Run -> Debug Configurations
2. Double click “Codasip C/C++ Applications” in the left column which will produce the dialog box shown in Figure 7. You can rename the configuration in the Name: box but that is not required.
3. Click Select in the C/C++ Project: area, which has the small “1” in a blue box.
4. Double click “phase1\_test” project. “phase1\_test” will appear as the project.
5. Click Select in the Application: area, which has the small “2” in a blue box.
6. Select “Select from project…”
7. Double click “phase1\_test.xexe”, which chooses the object file of the software to run. The executable file will appear as the Application.
8. Click Select in the Debugger area, which has the small “3” in a blue box. Make sure the button to the left of “Managed” is selected.
9. Double click “standardname1.ia.standardname1-is-simulator” in the Debugger area, which chooses the hardware build on which to run the software. The simulator will appear as the Debugger.
10. When the phase1\_test.s file was opened in the Editor View, note that the initial label is “\_start”. This is where the test program should begin execution.
11. In the Debug Configurations dialog box, go to the “Debugger” tab at the top of the window as shown in Figure 8.
12. In the “…at function or address:” box, change the entry to “\_start”. This tells the simulator where to initiate the run.
13. Click the Debug button at the bottom of the dialog box to start the simulation. This will produce a dialog box asking you if you want Eclipse to switch to the Debug Perspective. There is also a box which can be selected to always make this choice. It is recommended to select the always switch option, but not required.
14. Click Yes to initiate the simulation. Eclipse will switch to the Debug Perspective and begin the simulation.

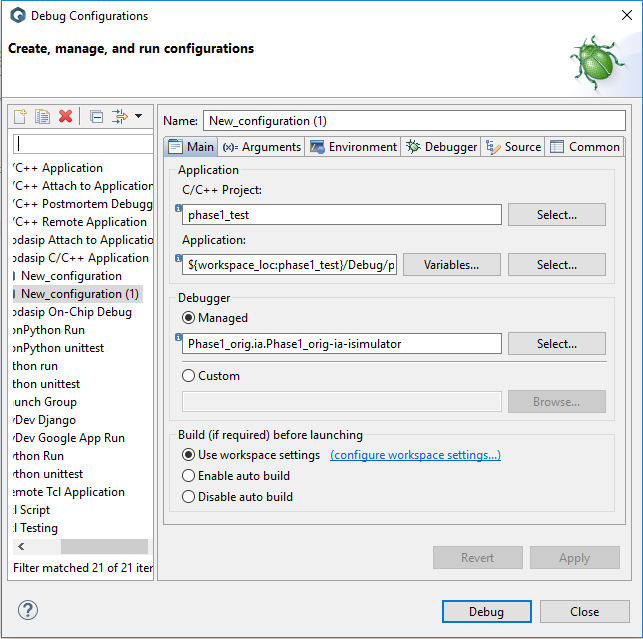


Figure 7

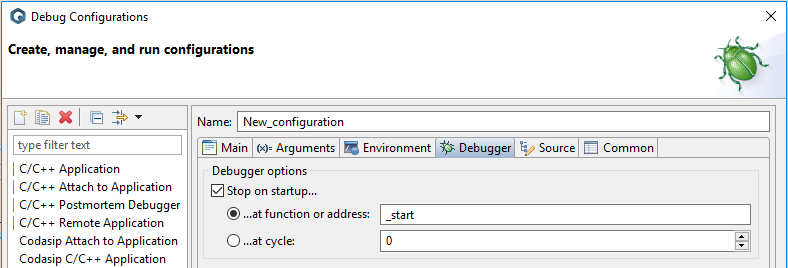


Figure 8

The Debug Perspective is shown in Figure 9 below. As in the Codasip Perspective there are several Views in the Debug Perspective.

1. The Debug View in the upper left shows which simulations are running. In this case there is only a single simulation, and in general that is the desired environment.

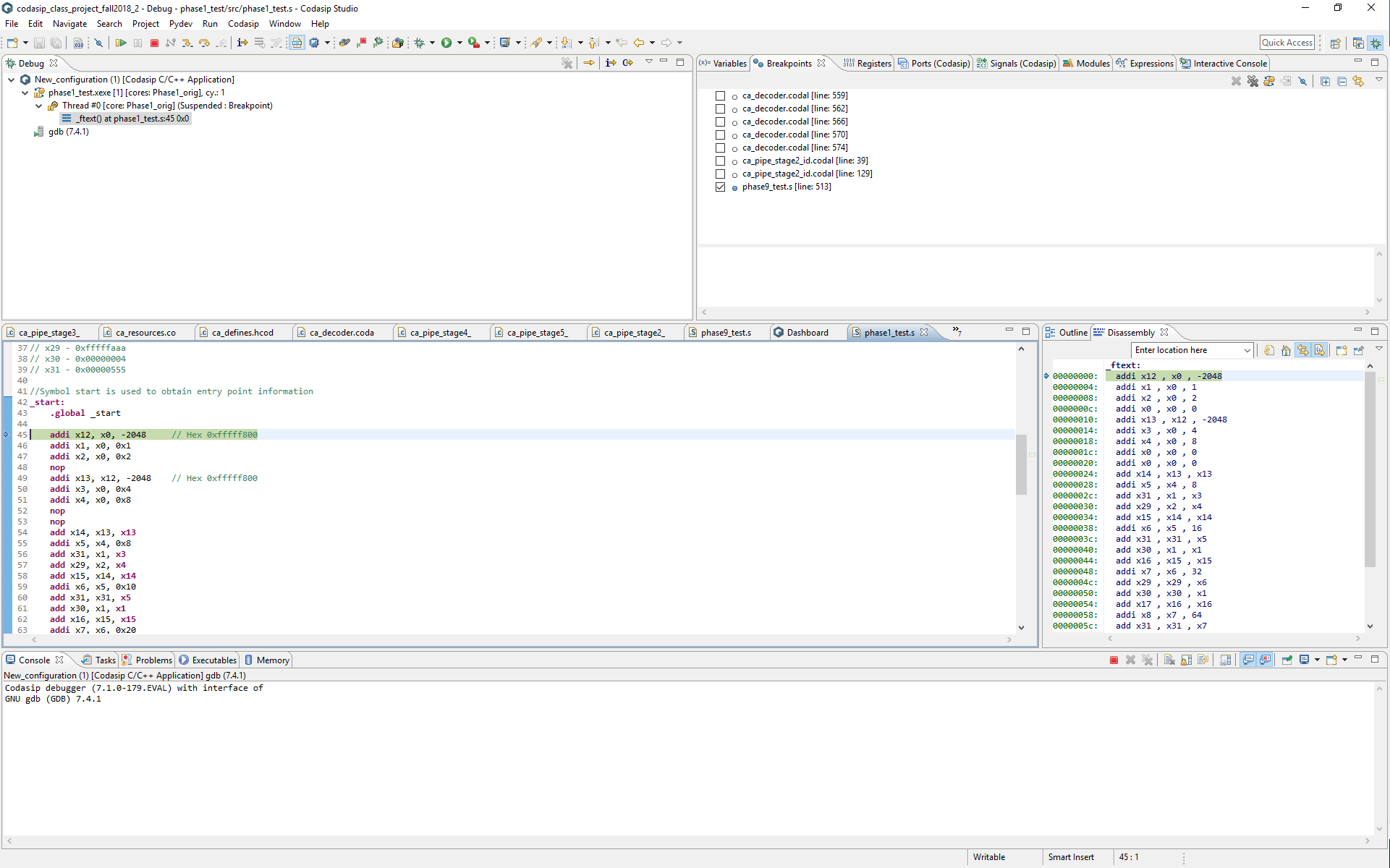


Figure 9

1. The Display View in the upper right includes several tabs which allow the observation of the internal state of the simulation.
2. For this test we only need to observe the state of the RISC-V Register File, so select the Registers tab.
3. Expand the Architectural Registers section by clicking on the arrow to the left of it.
4. Expand the codasip\_top\_level line.
5. Expand the standardname1 line.
6. Expand the rf\_xpr section to show all 32 registers in the Register File.
7. Select rf\_xpr[0], scroll down until rf\_xpr[31] is visible, hold down the Shift key and select rf\_xpr[31], which will select all of the registers.
8. Right click in the Name column of one of the selected registers, select Number Format and then select Hex. You can switch between multiple formats, but Hex is necessary to determine the correct behavior of the add\_test\_nop2 test.
9. The Editor View on the center left is the same as the one in the Codasip Perspective. In debug mode it initializes to the source code of the test program, and the green highlight shows the next instruction to be executed.
10. The Outline View on the center right lists all labels of the program and will be used for other functions in the future.
11. The Console View at the bottom is the same as in the Codasip Perspective, but there are several different Consoles which may be displayed, and the tool will switch automatically between them depending on the state of the simulation.

The simulation is executed using the control buttons in the Debug toolbar, as shown in Figure 10. There are several key functions here.

1. The green arrow (shortcut F8) runs the simulation until it completes or a halt instruction or breakpoint is encountered.
2. The left yellow arrow (shortcut F5) steps into a routine and the right yellow arrow (shortcut F6) steps over a routine. In this environment these two functions are essentially equivalent.
3. The red box (shortcut CTRL-F2) terminates the simulation.

Step the simulation using the step into arrow and observe two things: the executing instruction indicator (the green box) in the Editor window will step though the instructions, and the Register tab in the Display window will show updated values for each Register File register as they are written by the instructions.

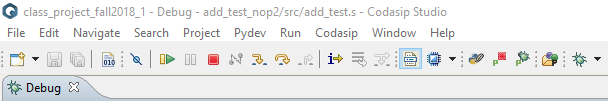


Figure 10

After a few steps, run the simulation with F8 and it should stop just beyond the halt instruction. Note that RISC-V does not have a real HALT instruction, so we have created a special instruction using an unused opcode to allow us to stop simulations. Once the simulation is halted, the values in all of the Register File registers should match the values described in the comment at the beginning of the test code. If all values match, the program has executed successfully. Terminate it with the red box.

The “bug” button at the right edge of Figure 10 re-executes the current Debug Configuration, and is handy for rerunning simulations after bug fixes.

1. **Changing Perspectives**

The Perspective can be easily changed using the buttons in the upper right corner of each Perspective, as shown in Figure 11. The “bug” selects the Debug Perspective, the double box next to it selects the Codasip Perspective, and the “+” box produces a dialog box which allows the selection of any Perspective.



Figure 11

1. **Scoring the Project**

The project score will be a function of when it is correctly submitted, with the correct project name and zip file name. Incorrect submissions will be declined, with the goal being within 1 day, and they must be corrected and resubmitted to receive credit. The score will be reduced by 25% for each day the correct is submitted after the Target Date. There may be deductions for incorrect naming and other procedural errors.

1. **Export the Successful Project**

A Project is submitted for grading by Exporting it. To do this, return to the Codasip Perspective, right click the hardware Project standardname1 in the Project Explorer and select Export. Select General -> Archive File and select Next, which will produce the dialog box shown in Figure 12. Expand the Project, and uncheck the “work” folder. This must be unchecked to produce an acceptable submission, as otherwise the submitted file will be too large.

In the “To archive file:” box, select Browse and go to the G:/Submission folder. In the file box enter standardname1 which must match the Project name. Incorrect naming will produce an unacceptable submission. Once the correct name is entered, select Finish. This will produce a standardname1.zip file in the Submission folder. Send a Slack Direct Message to the Professor when the Project is submitted.

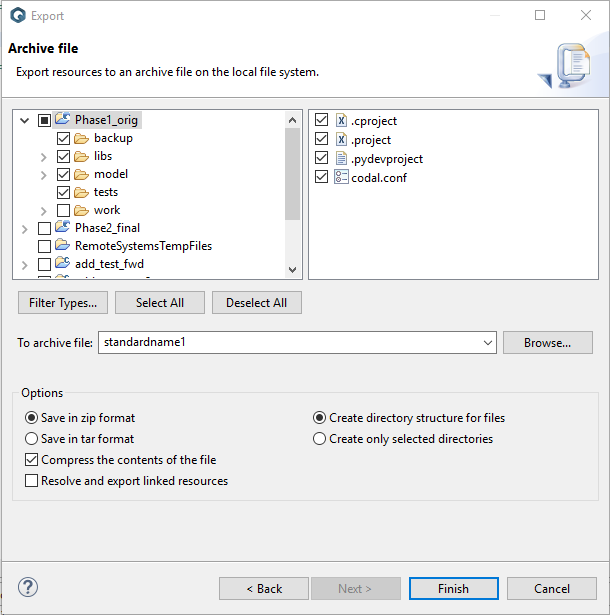


Figure 12