OpenCPI
GUI User Guide

Revision History

Revision	Description of Change	Date
1.0	Creation	2021-04-19
1.1	Beta release updates	2021-07-06
1.1.1	Update Unit Test section	2021-07-23
1.1.2	Multi-select is now available; default target is host OS	2021-07-29

Table of Contents

1 Introduction

The OpenCPI GUI is a PyQt5 application that allows access to most of the **`ocpidev`** commands. This allows users to supplement command-line interface (CLI) use of **`ocpidev`** by using the features of the GUI. This reduces errors and can streamline OpenCPI project development.

While feature-parity with CLI tools has not been completed, a large number of CLI commands have been implemented within the GUI. For those commands that are not available within the GUI, the CLI can still be used.

Where **`ocpidev`** commands are shown, please refer to the OpenCPI Component Development Guide at the latest OpenCPI Documentation page for a more thorough discussion, as more advanced features are available via the command line.

2 Installation

Source installation is the primary way of using OpenCPI GUI. Prior to running, the user will have to install PyQt5 by running the command **`sudo apt install python3-pyqt5`** for Debian-based distros and **`sudo yum install python36-qt5.x86_64`** for Red Hat-based distros.

To obtain the GUI source code, it needs to be cloned from the Gitlab repository using **`git`**. If **`git`** is not already installed, use the command **`sudo apt install git`** (Debian-based) or **`sudo yum install git`** (Red Hat-based).

Change to the directory where you want the GUI code to reside. The command

`git clone https://gitlab.com/opencpi/ie-gui.git` will clone the GUI source code to the local system.

To launch OpenCPI GUI, use **`cd <path_to>/ie-gui`** directory and execute the command **`python3 ./OpenCPI_GUI.py`**.

Upon first launch, the GUI will have two dialogs appear. One will ask for the installation path of **`/opencpi`**, the other will ask for the project path to display. If opening a pre-existing project, it is recommended to use the path **`~/User_OpenCPI_Projects`** for new projects.

The user guide for OpenCPI GUI can be found within the cloned source code, at

`<path_to>/ie-gui/doc/odt/OpenCPI_GUI_User_Guide.fodt`.

3 Interface Details

The Project Explorer panel shows a view of the currently set project. By default, this is set to show the contents of **`~/User_OpenCPI_Projects`** directory, which is assumed to be the location for all OpenCPI project assets. However, the user is able to change the displayed project via the *OpenCPI Configuration* menu.

The following figure shows the main window. Note that the **`ocpidev`** commands for creating OpenCPI components is in the menu bar. More commands are available via the Project Explorer's right-click context menu.

The main window is divided into six main areas, which are discussed in detail in the following sections:

- Menu bar
- Project Explorer
- Targets
- Unit Tests
- Job Manager
- Output Console

Note that the commands that are executed within the GUI are displayed in the Output Console for clarity. It should also be noted that a lot of the GUI is generated dynamically, resulting in instances where the GUI, or individual windows, will be black for a brief period while the necessary information is generated.

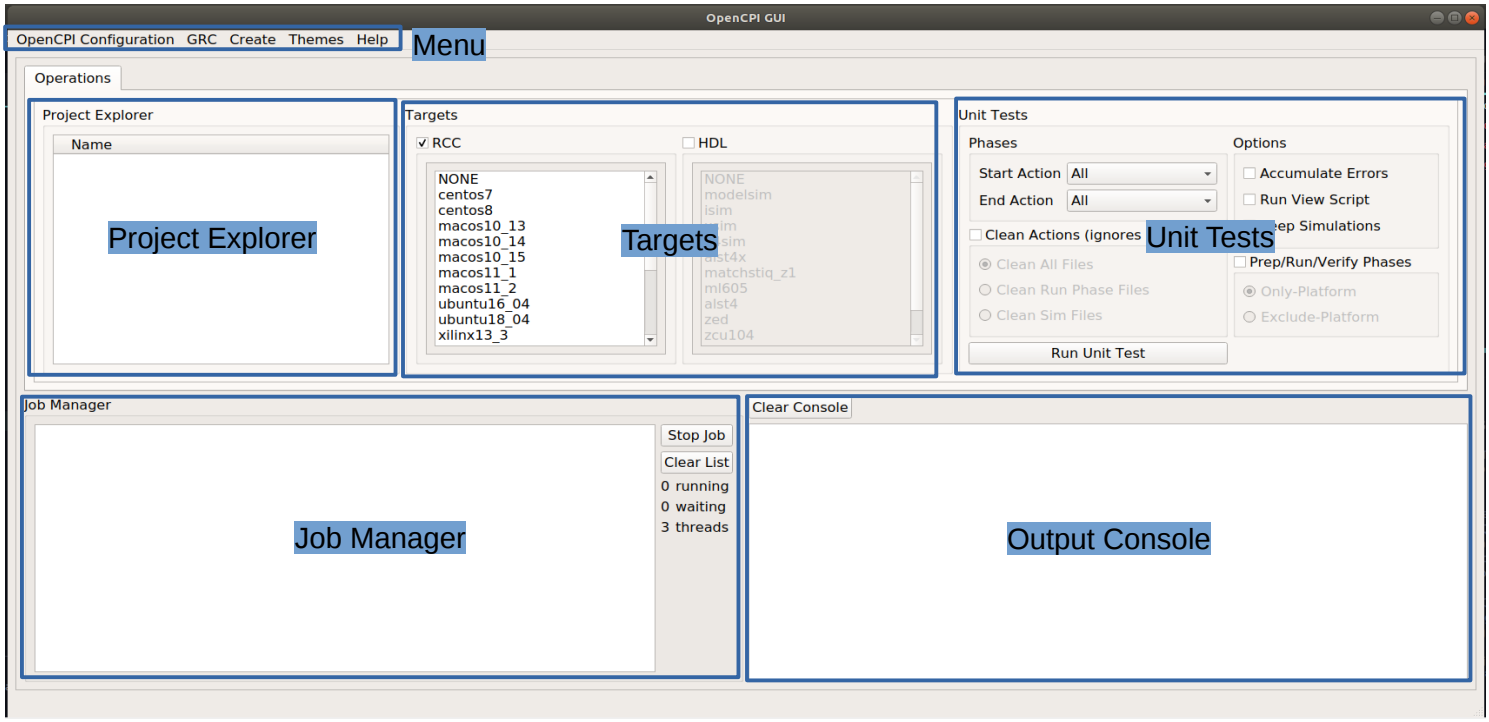


Figure 1: Main Window

4 Menu Bar

While a lot of functionality for OpenCPI projects and related components is provided via Project Explorer's context menu, some features are available through the menu bar.

The Menu Bar contains the following menu options:

- OpenCPI Configuration
- GRC
- Create
- Themes
- Help

These categories will be discussed in the following subsections.

4.1 OpenCPI Configuration Menu

The OpenCPI Configuration menu option handles both setting and viewing certain aspects of the OpenCPI GUI environment.

The following figure shows the OpenCPI Configuration menu options.

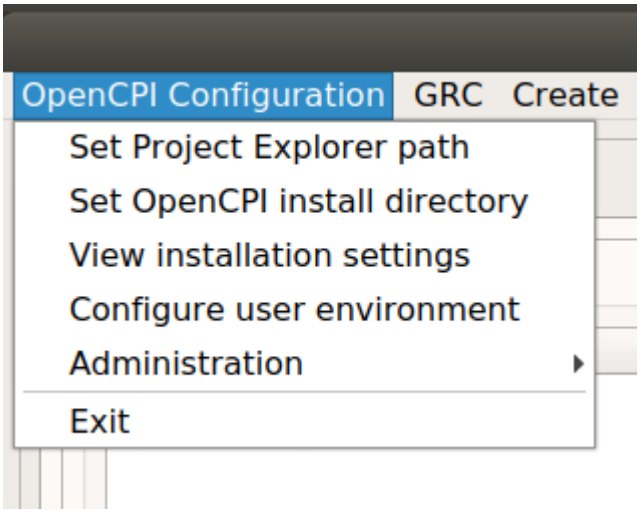


Figure 2: OpenCPI Directory Location Dialog

The bullets below describe the individual options for the menu.

- Set Project Explorer Path

  This option allows the user to set the path that is displayed within the Project Explorer panel. Normally the displayed path will be set to **`~/User_OpenCPI_Projects`** but projects created in different locations can be accessed.

  On first use of the GUI, this dialog will be called to set the correct path prior to setting up the GUI environment.

  The dialog associated with this menu option will show the Project Explorer's currently displayed path, as shown below.

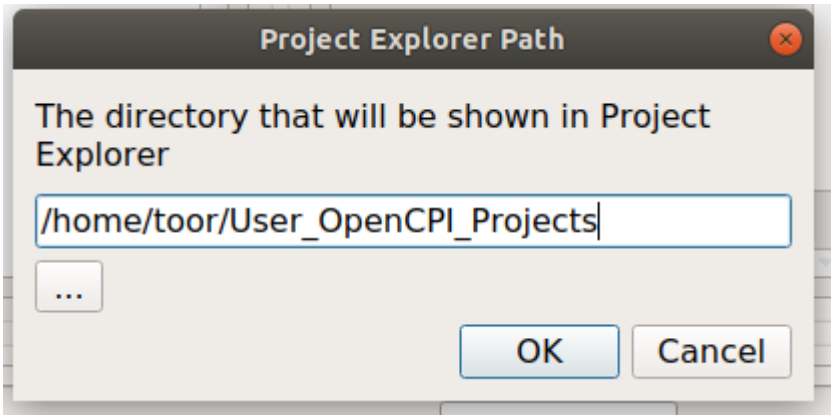


Figure 3: Current path displayed in Project Explorer

- Set OpenCPI Install Directory

  This option allows the user to set the location where **`/opencpi`** directory is located. This location is necessary to ensure that the GUI will correctly execute OpenCPI commands.

  On first use of the GUI, this dialog will be called to set the correct path prior to setting up the GUI environment.

  The dialog associated with this menu option will show the path currently set to the **`/opencpi`** directory, as shown below.

Figure 4: OpenCPI installation path

- View Installation Settings

  This option displays the current environment settings for OpenCPI, including the CDK path and user-set environment settings.

  As shown in the figure below, if the user hasn't set any environment settings, the window will indicate that default settings are used. However, if specific settings are made, such as for XSIM, those settings will be displayed.

  As this window is read-only, environment settings cannot be modified using the GUI. An external application is required to make changes.

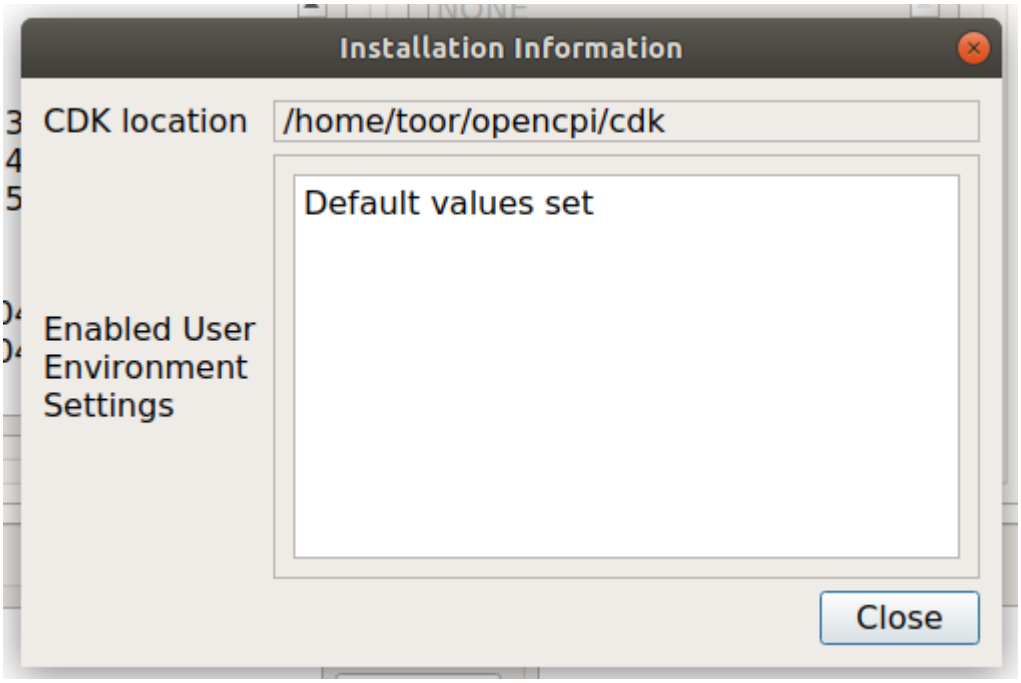


Figure 5: Installation Environment Settings

- Configure User Environment

  This option will launch an external application, allowing the user to update the **`user-env.sh`** file. The application used will be the default application, set by the user or OS, for text files, as shown below.

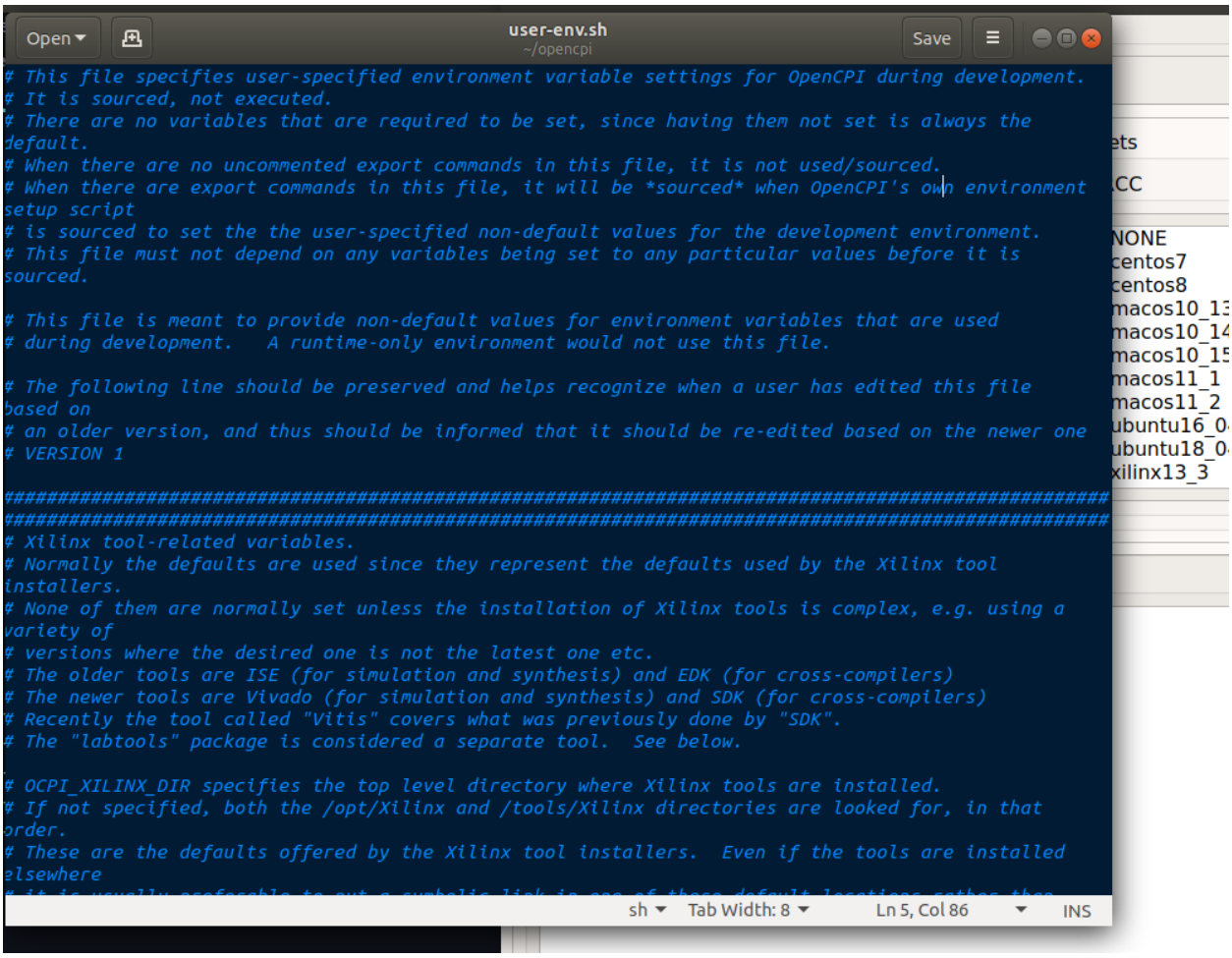


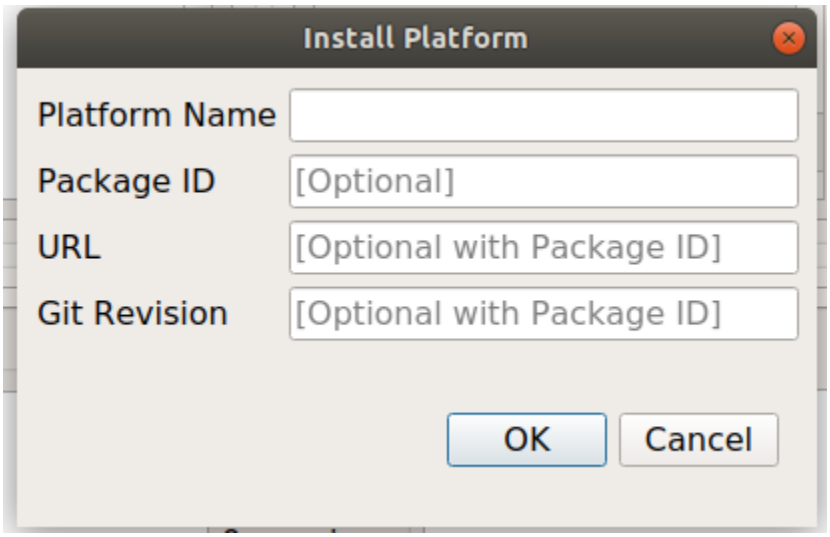
*Figure 6: Modifying **`user-env.sh`** file*

- Administration

  This menu option actually has to sub-options: Install Platform and Deploy Platform. These choices are used with the **`ocpiadmin`** command.

  - *Install Platform* will install the specified platform, using the dialog below. The only required information is the platform name.

    Optional information includes the package ID, platform URL, and Git revision, though the last two items required the package ID to be provided first. This does mean that the desired platform does not have to be on the local file system. It can be downloaded directly from a remote location, such as a web site or Git repository.



- Deploy Platform will deploy the platform using the specified RCC and HDL targets, as shown below.

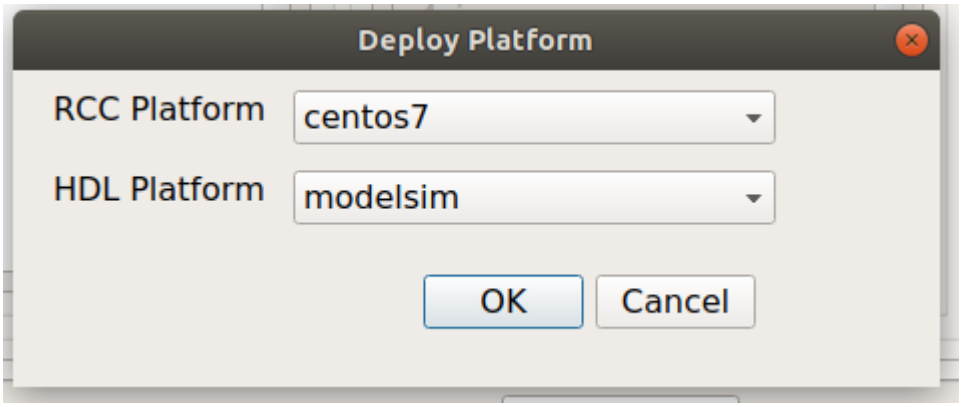


Figure 7: Deploy Platform Dialog

5 GRC Menu

For users who desire to work with GNU Radio Companion concurrently with OpenCPI, the GRC menu is available. Two menu options are available: *Configure GRC* and *Launch GRC*. GNU Radio must be installed to utilize GRC.

The configuration dialog allows setting of the **PYTHONPATH**, which is necessary to launch GRC. The default setting is the normal path for Python packages.

The following figure shows the GRC configuration dialog.

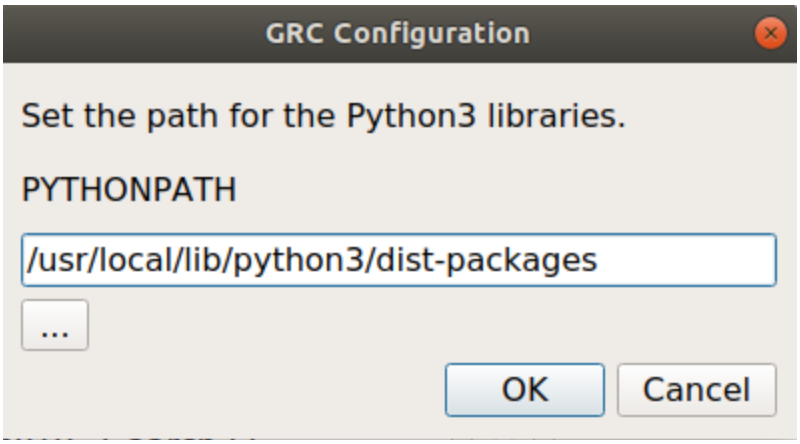


Figure 8: GRC Configuration Dialog

Executing GRC via the Launch GRC menu will open GRC in a separate window (example below), assuming GNU Radio is installed. This allows the user to have both GRC and the OpenCPI GUI open at the same time, if desired.

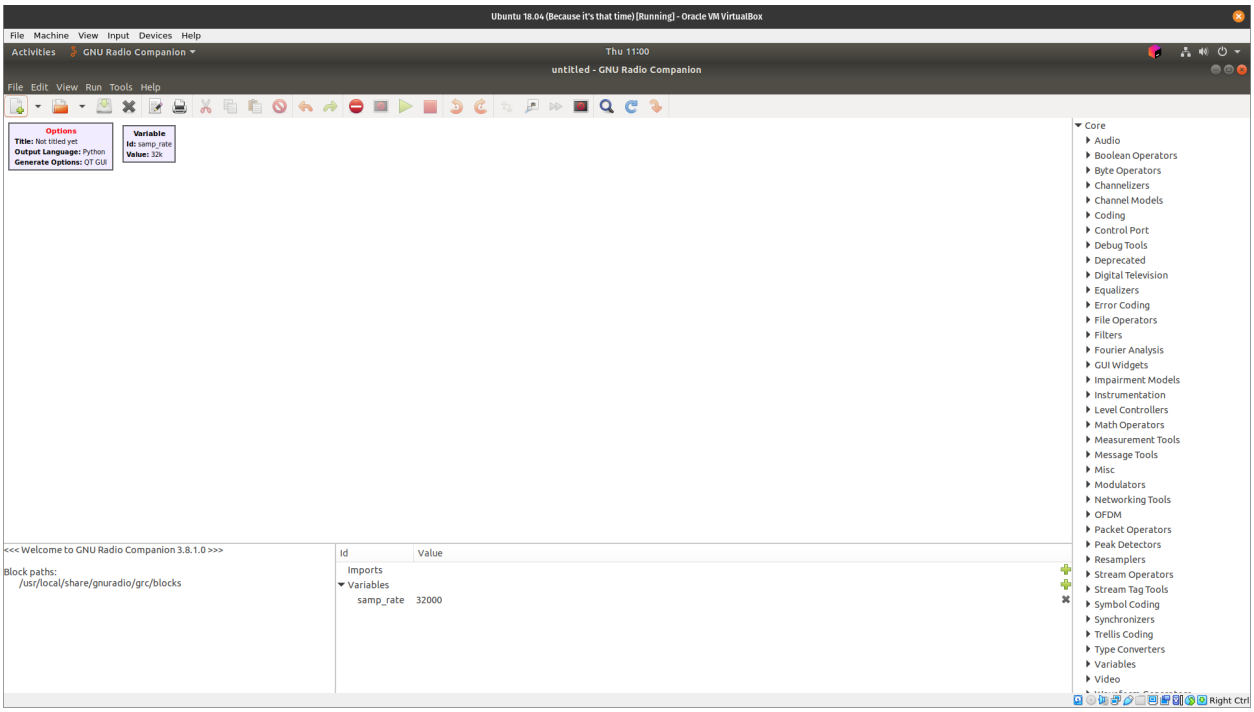


Figure 9: GRC Window

6 Create Menu

In this section, we will work through the Create menu options, shown in the figure below. It is important to note that only the basic functionality of **`ocpidev`** create options are provided; the specific commands that are implemented are provided in their respective sections.

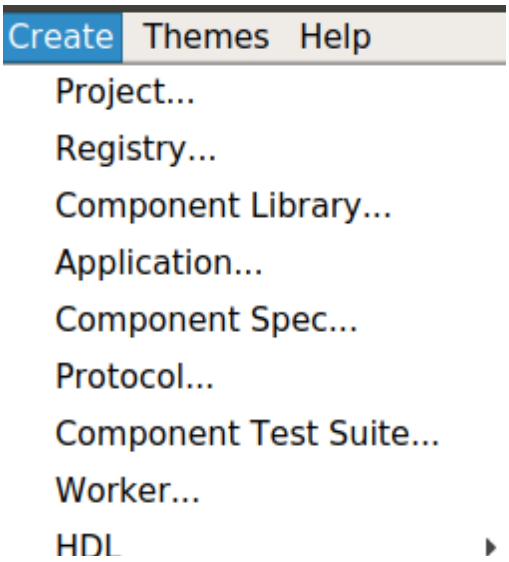


Figure 10: Creation Menu

Any command output will be shown in the output console at the bottom of the GUI. However, not every **`ocpidev`** command has output, nor do they have the same format.

6.1 Create Project

The dialog for creating a new project is shown in the figure below. Only Project Name is required; the other fields are optional.

New Project Location allows the user to specify where the new project will be located, assuming they do not want to use the default path.

Package Prefix will default to local if not provided, while Package Name will have the same name as the project.

Project Dependency allows the user to specify another project the new project will be dependent on.

Register Project allows the user to register the project upon creation, rather than as a later step.

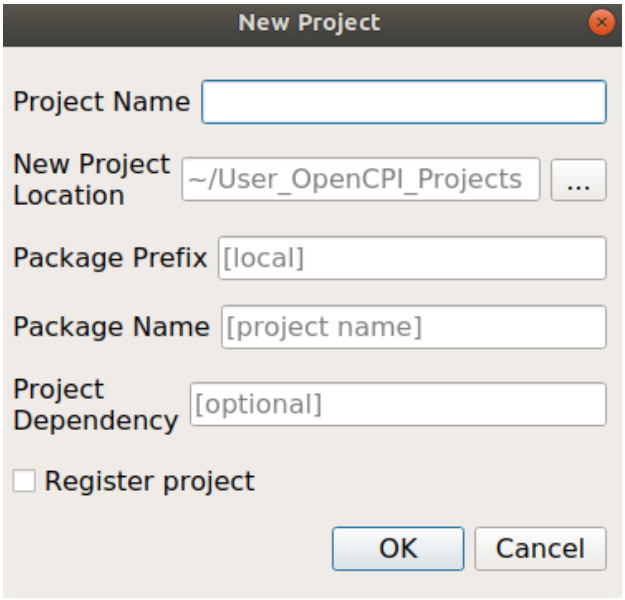


Figure 11: New Project Dialog

If the Project Explorer display path is set to the same location as the new project, the new project appear in the Project Explorer panel, shown below. Otherwise the Project Explorer display path will have to be changed to see the new project.

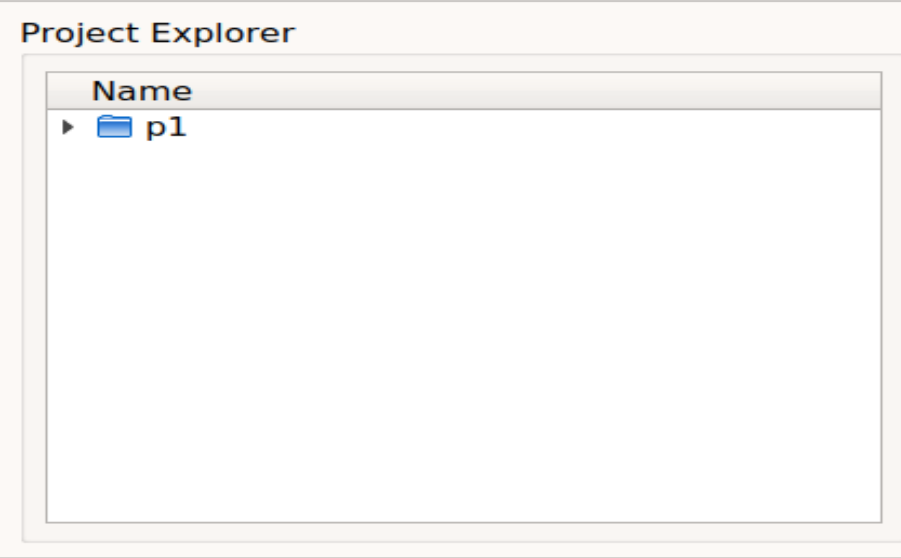


Figure 12: New Project in Project Explorer

6.2 Create Registry

To create new registries, select “Create->Registry...” from the menu bar. A new dialog box will appear, shown below.

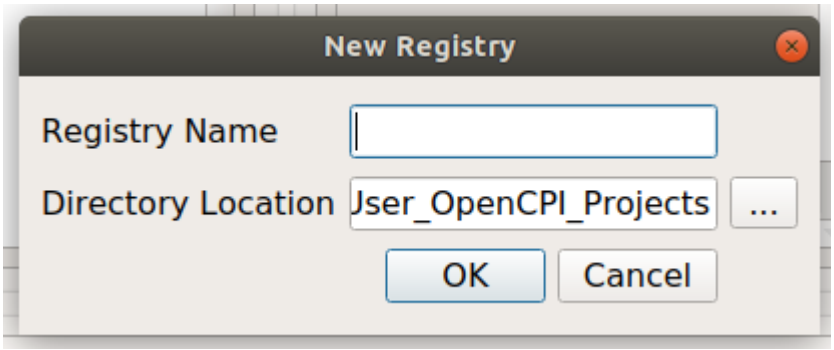


Figure 13: New Registry Dialog

Provide the name of the registry and the desired location. The default location is the set to ***~/User_OpenCPI_Projects***. When the new registry is created, it will be added to the designated location.

Just because the registry has been added to the directory doesn’t mean it’s available for use with your OpenCPI projects. You will receive a notice in the output console indicating that there is another step to complete, as shown in the following figure.

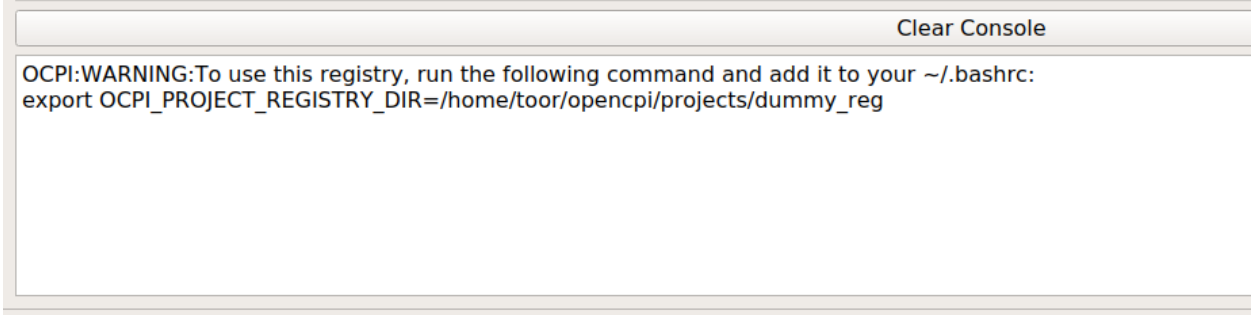


Figure 14: New Registry Warning

6.3 Create Library

Creating a new library is similar to creating a new project. The primary difference is that you have to associate the new library with an existing project, as shown below. The Associated Project combobox automatically lists all the projects listed in Project Explorer' display path. The new library is located at **`/<project>/components/<library>`**.

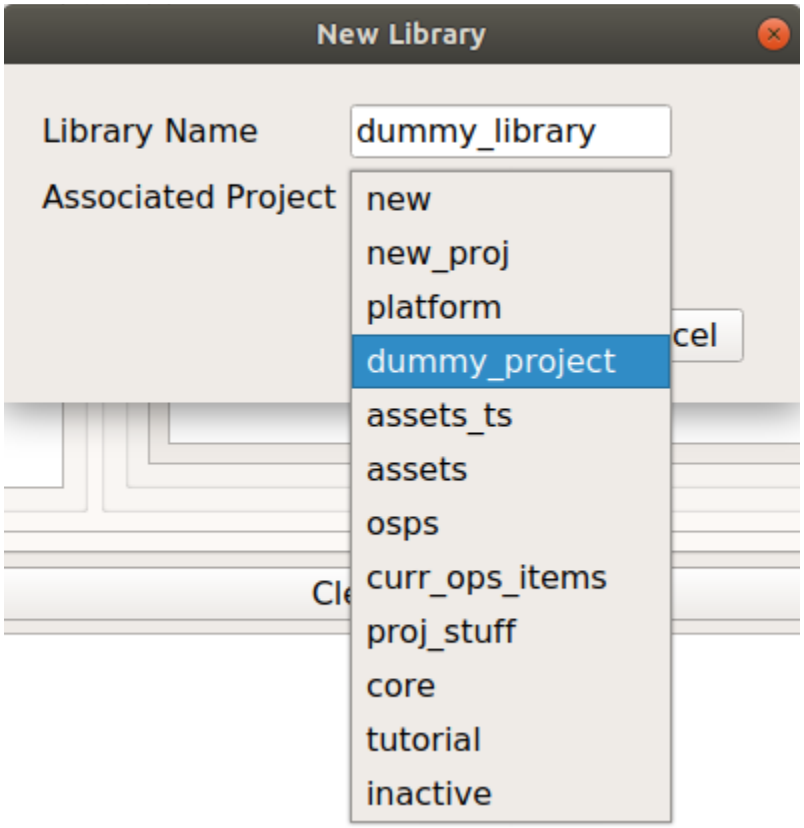


Figure 15: New Library Dialog

6.4 Create Application

Adding applications to a project is much like adding a library: provide the application name and associated project, shown in the figure below. The new application will be found in ***/<project>/*applications**.

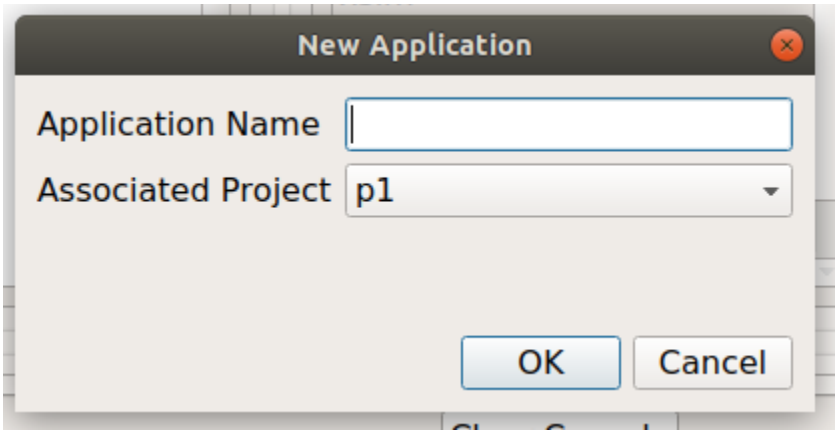


Figure 16: New Application Dialog

6.5 Create Component Spec

Unlike the previous dialogs, the new component spec dialog has an additional selection: Associated Library, shown below. As a particular project can have multiple libraries, you not only have to select the desired project, but also the library the spec should be associated with. The new spec will be found in **`/<project>/components/<library>`**.

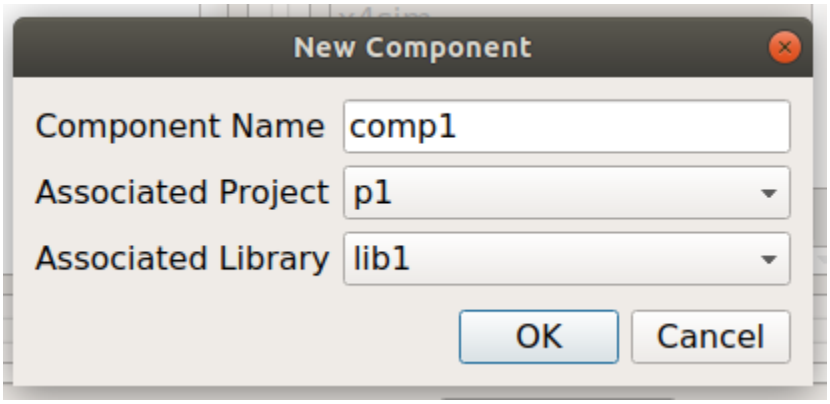


Figure 17: New Component Dialog

6.6 Create Protocol

Like component specs, the New Protocol dialog asks the user for the project and library associations for the protocol, shown below. The new protocol can be found in the ***<project>/<library>*/specs** directory.

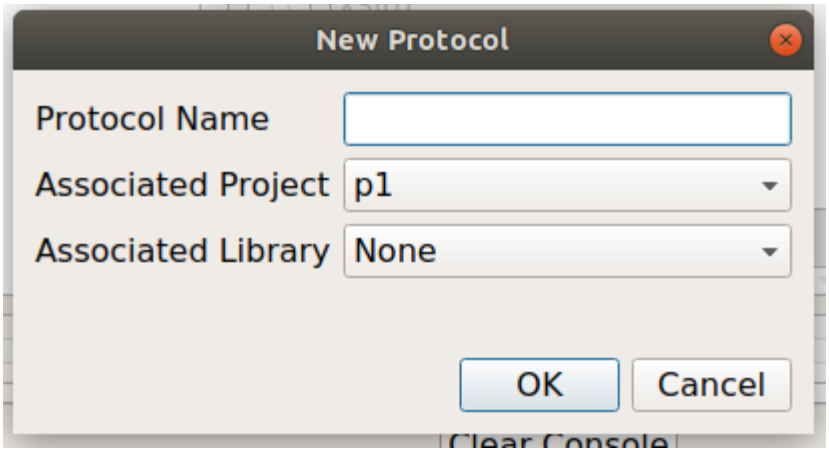


Figure 18: New Protocol Dialog

6.7 Create Unit Test Suite

When creating a new test suite, no name is required, but the project, library, and component spec the test will be associated with are required.

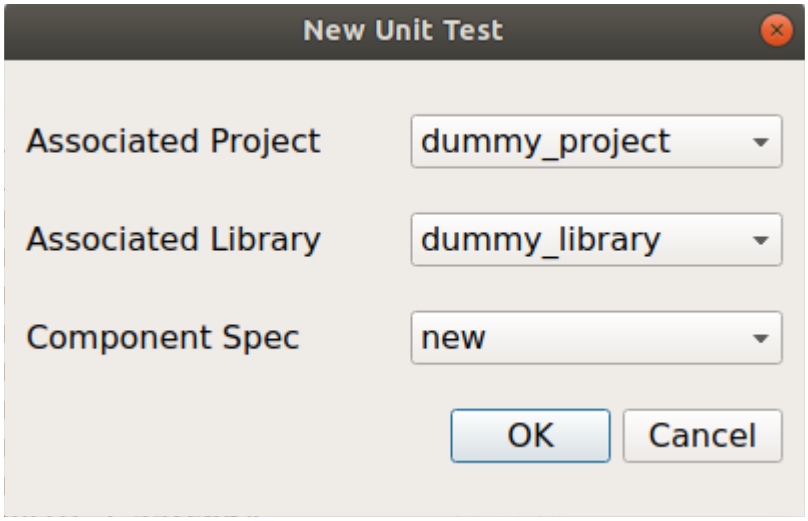


Figure 19: New Test Suite Dialog

The new test is added as
/<project>/components/<library>/<component_spec>.test.

6.8 Create Worker

New workers have quite a few options compared to the other creation dialogs, as shown in the following image. After entering the worker's name, select the project it is associated to.

Next, pick the model it will be using: RCC or HDL. Based on the model, the languages available will change. RCC allows for either C or C++ while HDL only allows for VHDL.

Finally, select the project's library that will be used by the worker and any component spec; this is the only item in the dialog that is optional to change.

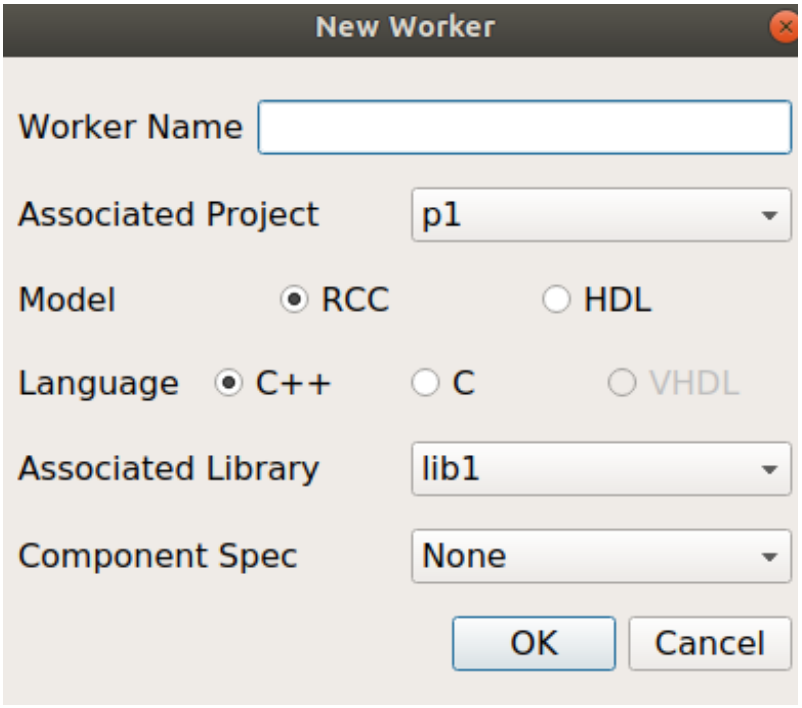


Figure 20: New Worker Dialog

The new worker is found under its associated library at **`/<project>/components/<library>/<worker_name>`**.

6.9 Create HDL Assembly

Creating an HDL assembly only requires the assembly name and associated project, displayed below. The new assembly will be found in the project directory at **`/<project>/hdl/assemblies/<assembly_name>`**.

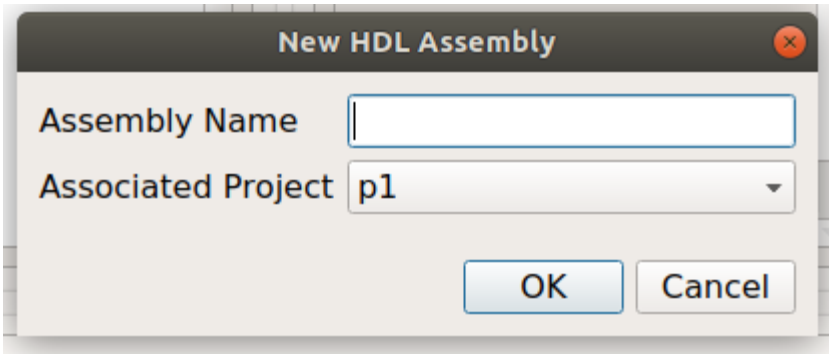


Figure 21: New HDL Assembly Dialog

6.10 Create HDL Platform

Making a new HDL platform has some additional fields (below) beyond the normal name and project: part number and time server frequency. Once the new platform information is input, the HDL platform is located under the project's HDL directory at **`/<project>/hdl/platforms/<platform_name>`**.

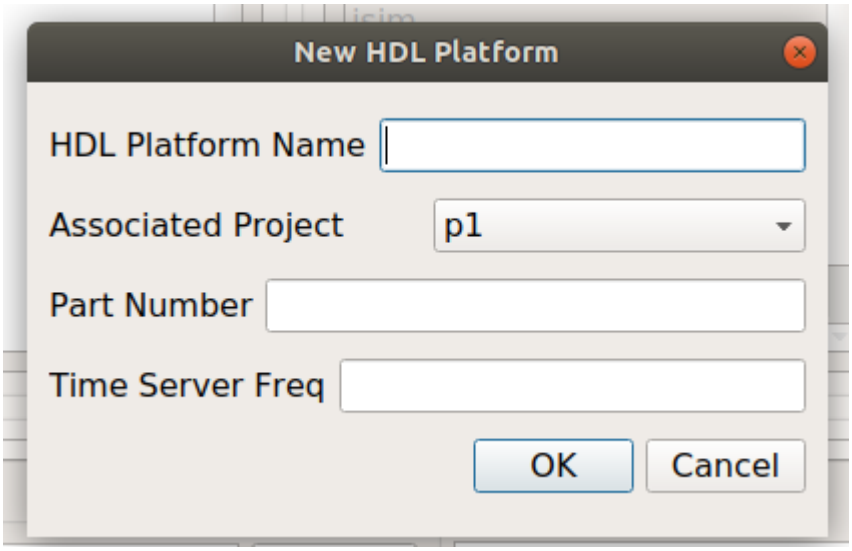


Figure 22: New HDL Platform Dialog

6.11 Create HDL Primitive Library

The dialog for primitive libraries is the same as for HDL assemblies: library name and associated project. When completed, the new primitive library is added to the HDL directory at **`/<project>/hdl/primitives/<lib_name>`**.

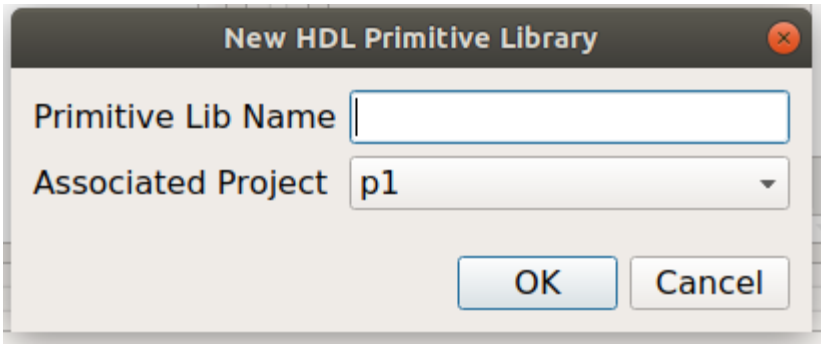


Figure 23: New HDL Primitive Library Dialog

6.12 Create HDL Primitive Core

Same as HDL primitive library, provide the name and associated project. The new core will be added to the **`/<project>/hdl/primitives/<core_name>`**.

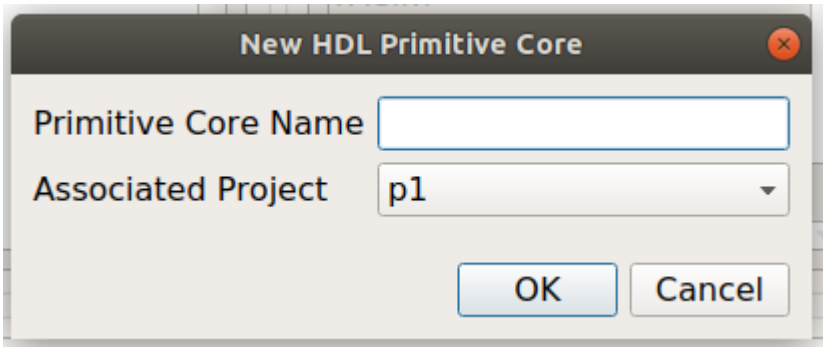


Figure 24: New HDL Primitive Core Dialog

7 Themes Menu

Several color themes have been provided for OpenCPI GUI.

- None: restores the color theme to the default OS theme.
- Ubuntu: provides the default theme for the Ubuntu desktop environment.
- Dark Orange: a dark theme, with splashes of orange highlights.
- Irritable Eel: a blue-green theme for an underwater feel.
- Color Blind: a color palette that should work for most types of color blindness.
- Toxic Sunset: for those with a tolerance for clashing colors.

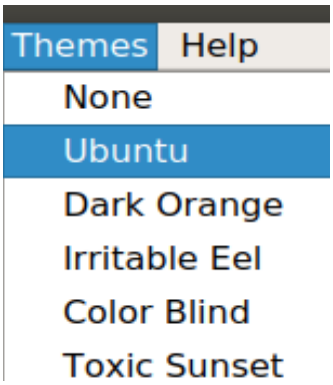


Figure 25: Theme Options

8 Help Menu

The Help menu provides the following options:

- Help Contents: a link to https://opencpi.gitlab.io/releases/latest/ , where the most recent OpenCPI documentation is located.
- Report Bug/Enhancement: a link https://gitlab.com/opencpi, the OpenCPI Gitlab repository, to allow ticket creation of bug reports or feature requests.
- About: the standard license and copyright information about the application.

9 Project Explorer

To access most **`ocpidev`** commands other than those related to create and testing, select the target in the Project Explorer panel and right-click. A context menu will appear with a number of different options (below).

Depending on the desired action, simply select the target of the command and pick the appropriate action. If an action is selected for the wrong asset type, an error message will be displayed in the output console.

To deselect an item, use Ctrl+click on the currently selected item. Items in Project Explorer can be renamed by double-clicking the item and providing the new name.

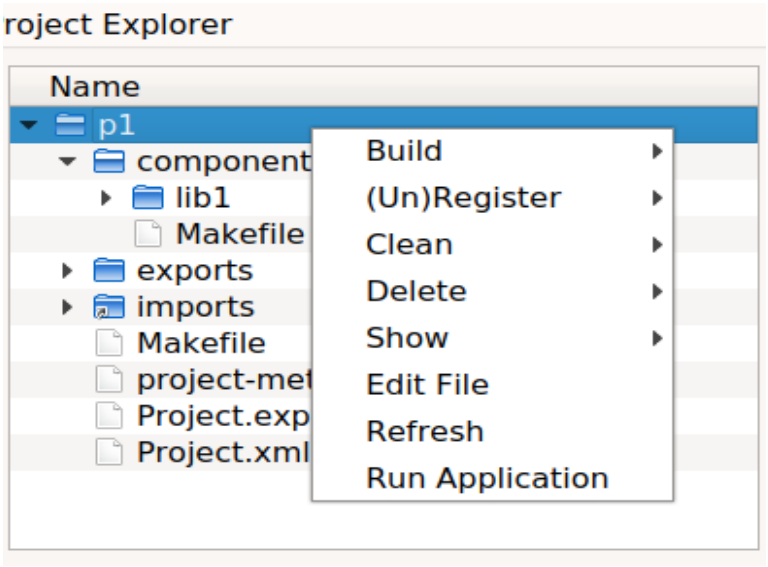


Figure 26: Project Explorer Context Menu

9.1 Build Menu

The Build menu can be used for any asset that can be built, specifically projects, libraries, applications, unit tests, workers, and HDL assets. After clicking on the build action, the appropriate **`ocpidev`** command will be called and the results printed to the output console at the bottom of the screen.

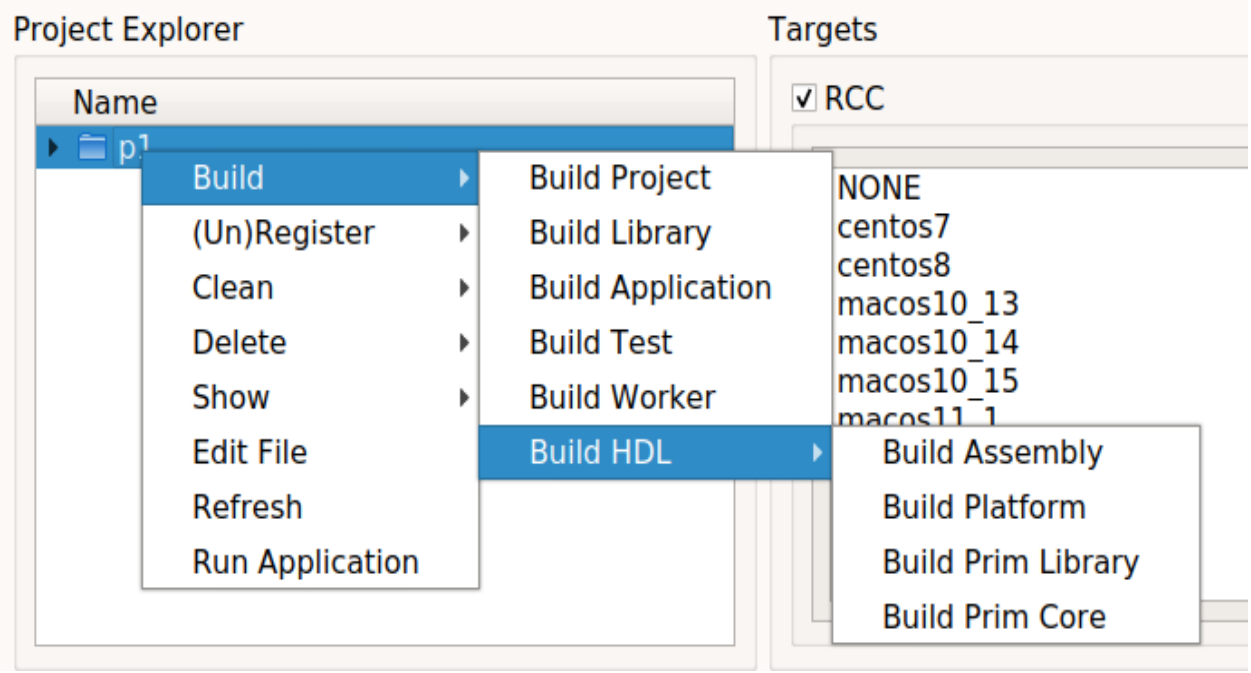


Figure 27: Build Menu Options

By default, clicking on an asset and running the build command will build the asset for the host OS. To use something other than the OS, you can select an RCC or HDL target first (Section 10), then run the build command on the desired Project Explorer asset.

9.2 (Un)Register Menu

The (Un)Register menu option allows the user to register or deregister the selected project. Registering sets the project to the default registry if it wasn't set upon creation while UnRegister removes that assocation.

9.3 Clean Menu

Much like the Build menu, the Clean menu applies to the same OpenCPI assets, but runs the **`ocpidev clean <asset_type> <asset_name>`** command.

9.4 Delete Menu

To delete OpenCPI assets, select the item and right-click to bring up the Delete menu. A confirmation dialog will appear, allowing the user a chance to decline deletion. This is important because the deletion is irrevocable; the asset is not saved prior to deletion.

9.5 Show Menu

The Show menu item is not context-sensitive. That is, you don't have to select a particular item within Project Explorer for it to function. Two options are available: show registry and show workers.

Show registry will display the projects currently associated with the default registry, shown below.

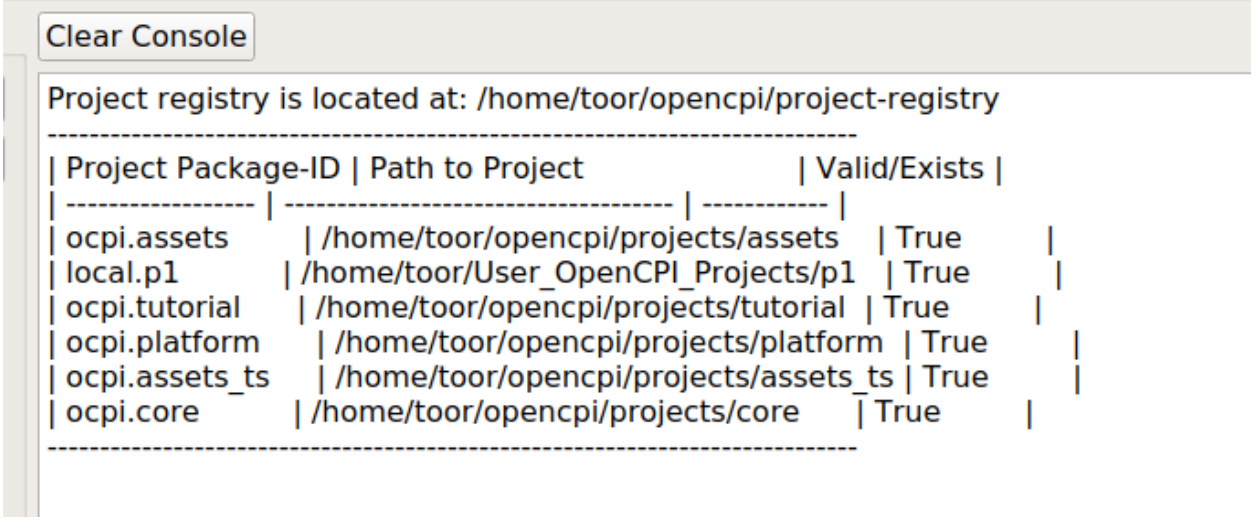


Figure 28: Show Registry output

Show workers shows the workers currently created, shown below.

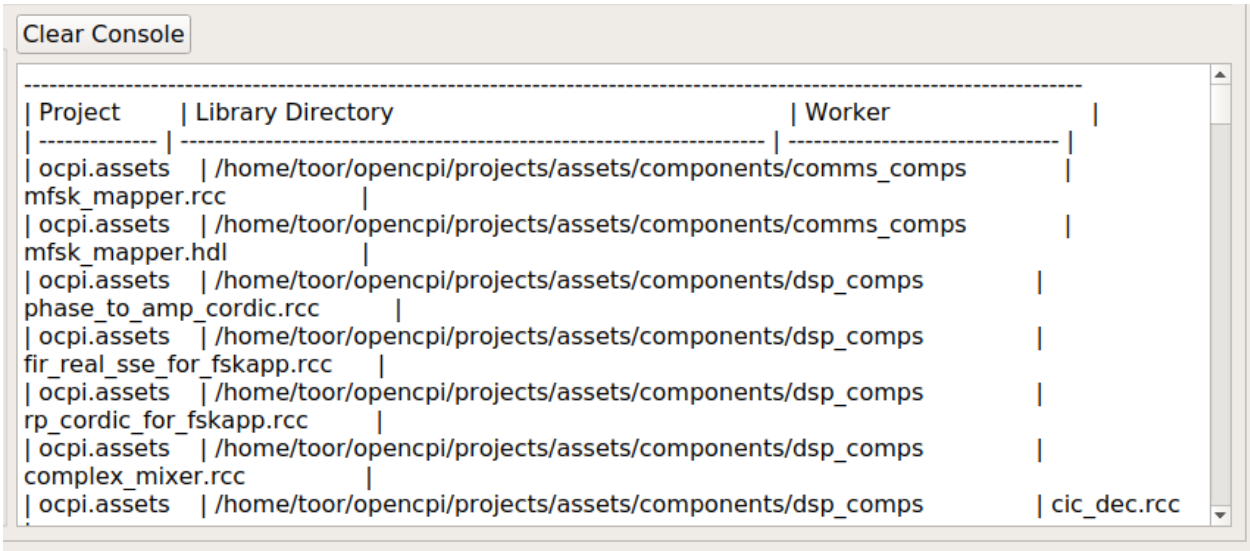


Figure 29: Show Workers output

9.6 Edit File

This command will open the selected file in the user's default text editing program. The following file types are available for editing:

- .xml
- .cpp
- .txt
- .cc
- .c
- .vhd
- .v
- .py
- .sh
- .deps
- .mk
- .h
- .hh
- .rst
- .exports
- Makefile
- package-id
- workers

9.7 Refresh

While the Project Explorer is configured to auto-update when the display path is changed or there are changes within the project directory, a manual refresh option is available for users in case they desire to use it.

9.8 Run Application

This option allows the user to execute **`ocpidev run`** on the selected application. The dialog allows the user to provide optional arguments prior to executing the command, as shown below.

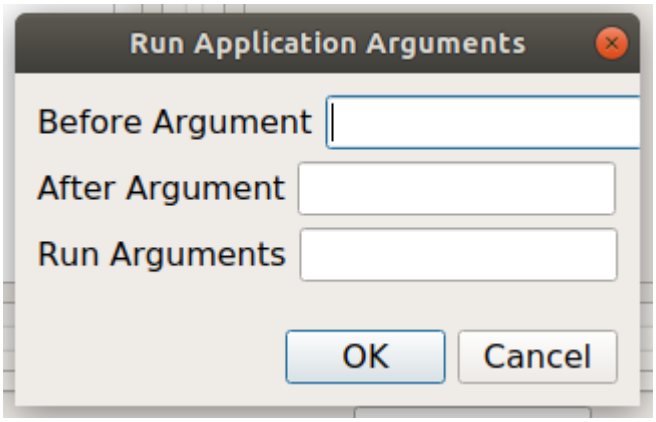


Figure 30: Run Application Dialog

Multiple arguments can be added to a field, separated with a space, as shown below.

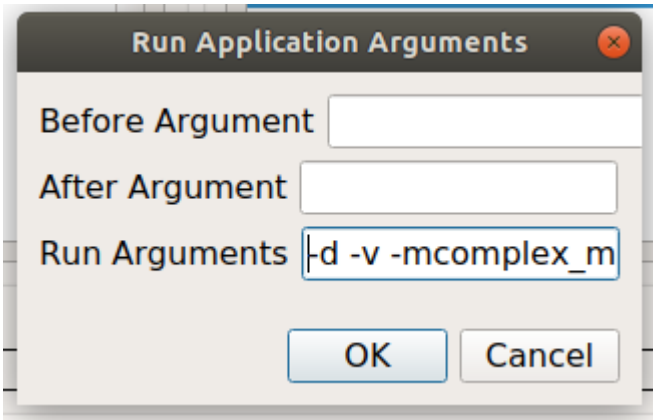


Figure 31: Using multiple arguments in Run Application

10 RCC and HDL Platform Targets

To access RCC or HDL targets, simply click the checkbox next to the group name (below). By activating the HDL panel, any selections in the RCC panel are ignored; in other words, the HDL targets take priority over the RCC targets, if that is an option for the particular command.

Multiple selection of RCC and HDL platforms is now available. Thus, a single OpenCPI command can target multiple platforms at the same time, rather than having to run a separate command for each target.

If NONE is selected as a target, regardless of operation, then the default target is the host OS. This also applies when multi-selecting targets: including NONE in the item selection will nullify the multiple selection and result in the command targeting the host OS.

Items listed in the RCC and HDL panels are read-only. Attempting to rename the item by double-clicking will not work; the item will show the new name but the change is not saved. This is because the list is dynamically created via an **`ocpidev`** command.

Note: If the RCC/HDL panels are not populated correctly or show an error message, the GUI should be restarted. The problem is because of incorrect data about the location of /opencpi. This problem will affect other parts of the GUI, so a reload should ensure that all parts of the GUI work correctly as well.

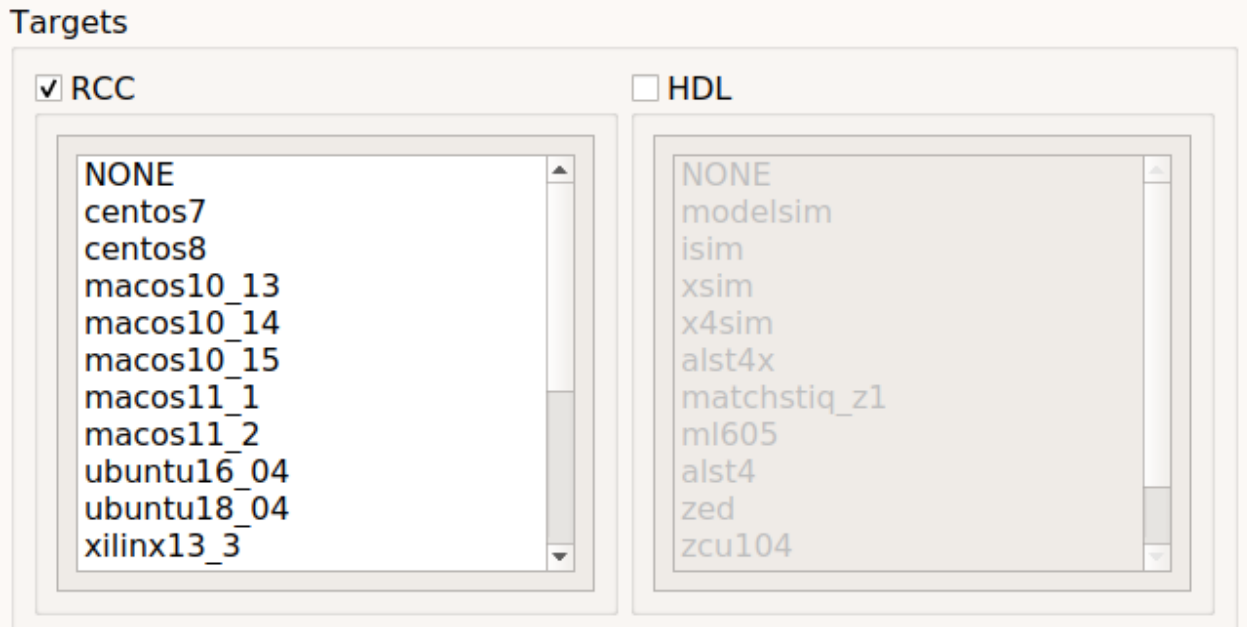


Figure 32: RCC and HDL Platforms

11 Unit Tests

The Unit Tests panel is separated into four sections: Phases, Clean Actions, Options, and Prep/Run/Verify Phases, as shown in the following screenshot.

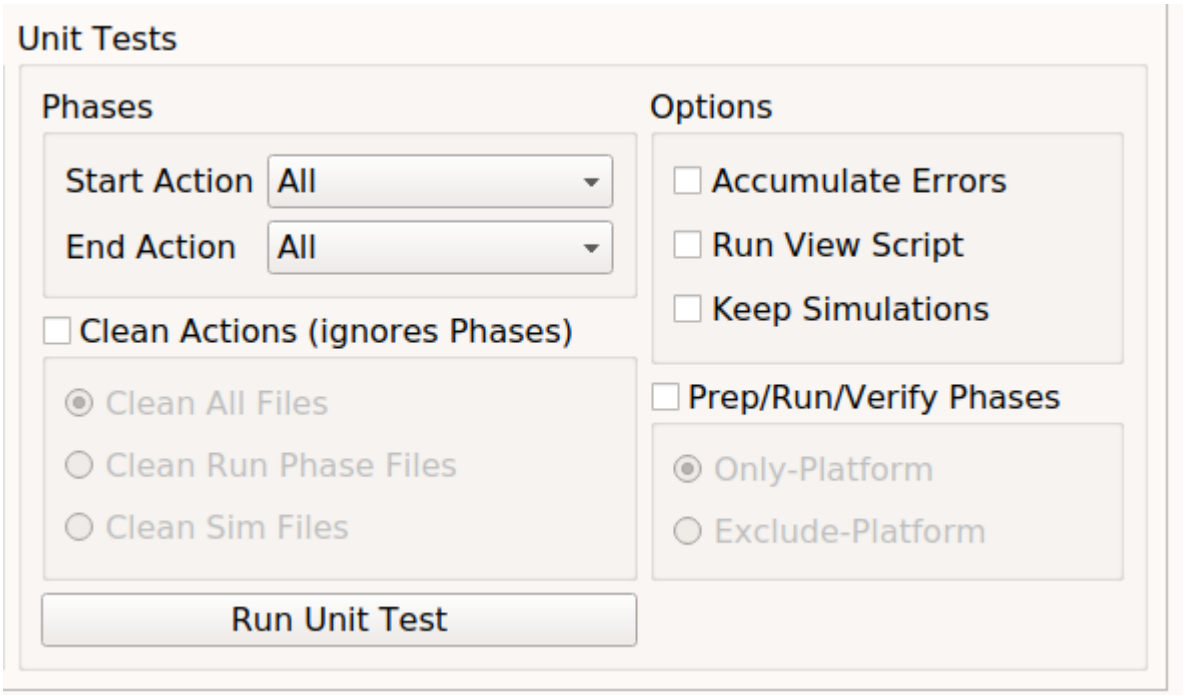


Figure 33: Unit Test Panel

Unit tests can have both RCC and HDL platforms selected as targets. Multiple selection is available for these targets, as shown in the following screenshot.

For normal unit tests, if “NONE” is selected as a target, whether individually or in a multiple selection, the unit test defaults to running tests with no RCC or HDL targets. However, if “Only-Platform” or “Exclude-Platform” options are used, a notification will appear indicating that “NONE” cannot be selected when either of those options are selected.

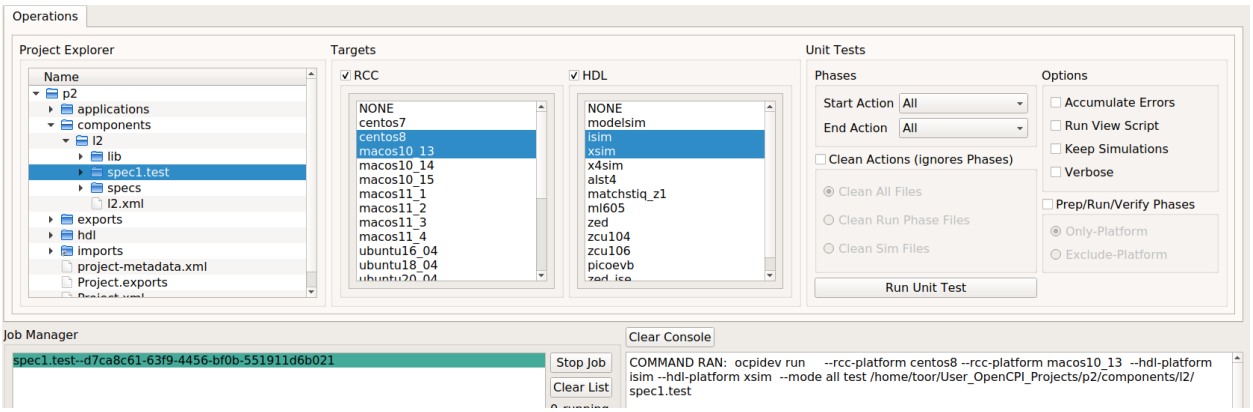


Figure 34: Unit test multi-selection

11.1 Phases

The Phases area consists of the Start and End Actions, which correspond to the five phases available when testing. The Start Action is the initial phase for the unit test while the End Action is the final phase for the test.

While all five phases are available to each action (shown below), only the accepted phase sequences are allowed (shown in a tool tip). Invalid sequences will result in an error. While only two actions are shown, the actual sequences include all additional phases between the start and end actions.

For clarity, the allowed start/end actions are:

- all/all
- generate/generate
- generate/build
- prepare/verify
- prepare/prepare
- prepare/run
- run/run
- verify/verify

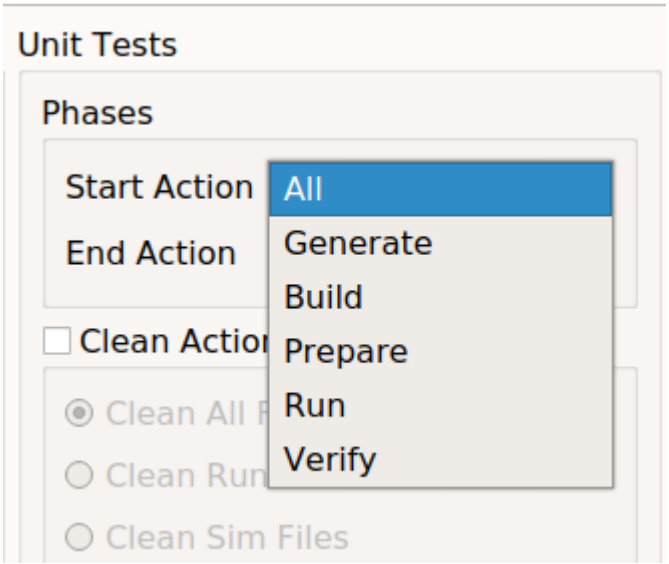


Figure 35: Unit Test Phases

11.2 Clean Actions

Clean actions supersede the phases; activating the clean actions panel will ignore any phases that are selected and only run one of the clean actions. By default, Clean All Files is set to run when the panel is active, but any of the actions can be selected.

The following figure shows the Clean Actions panel.

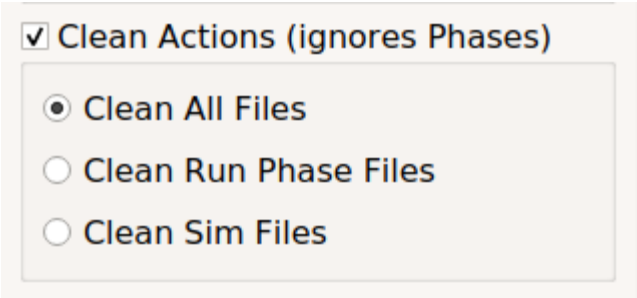


Figure 36: OpenCPI Clean Actions Panel

11.3 Options

The items in the Options panel (below) can be applied to any unit test. Multiple options can be stacked in the same test run.

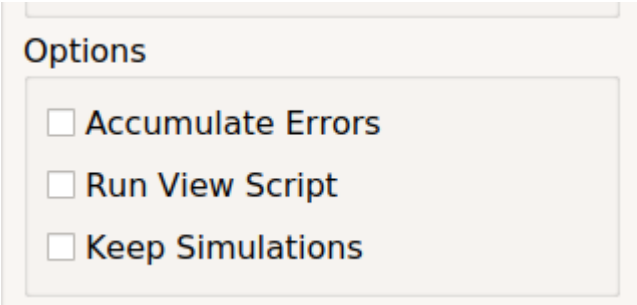


Figure 37: Unit Test Options

11.4 Prepare/Run/Verify Phases

When enabled, this groupbox allows the user to select **`only-platform`** or **`exclude-platform`**, which are used as arguments to the unit test execution command. These options only apply to prepare, run, verify, or any combination of those phases.

Both options allow for selection of an RCC, an HDL, or both to be included in the test.

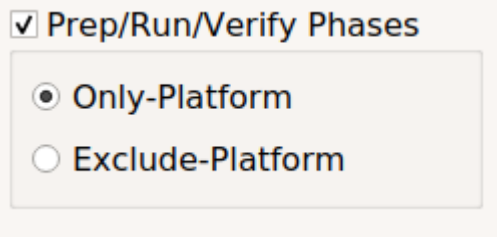


Figure 38: Prepare, Run, Verify-specific Options

12 Job Manager

The job manager is a key part of the GUI. It handles multithreading for long-running commands, such as build, unit tests, and install platform. Each process is shown in the manager window and is color-coded based on it's status (see below). When a job is started, it's displayed name constitutes the name of the OpenCPI asset plus a unique hash value. This ensures that the job manager can identify separate actions on the same asset.

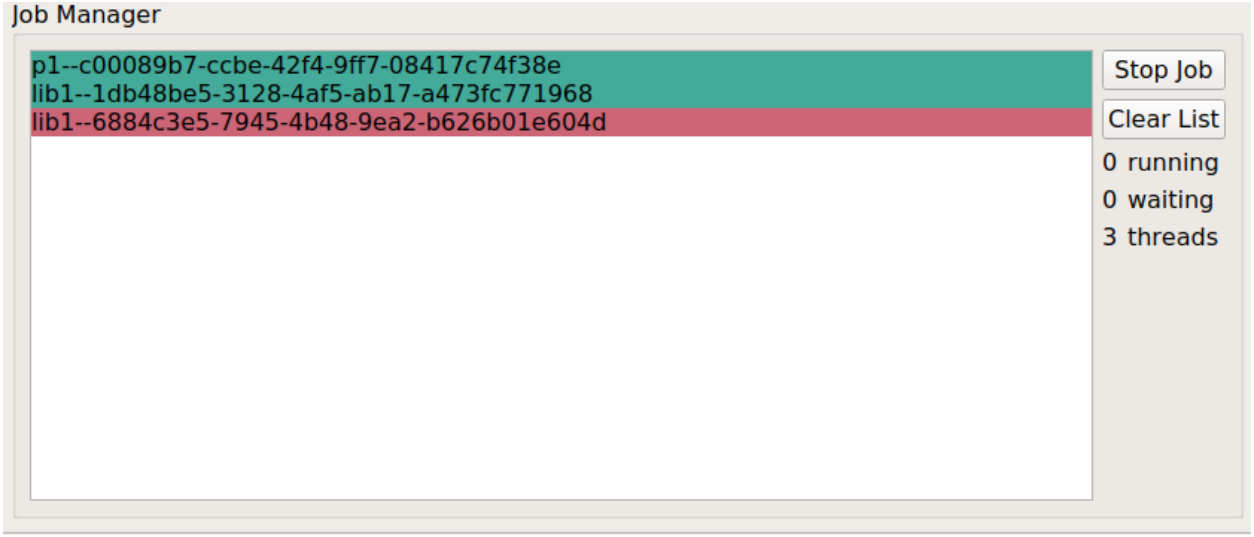


Figure 39: Job Manager Panel

The colors have the following meanings:

- Green: job successfully completed
- Red: job failed with an error message
- Yellow: job is currently running
- Grey: job has been manually stopped

Two buttons are provided to the right of the manager window itself: Stop Job and Clear List. The Stop Job button allows manual stopping of the selected job; the job is completely stopped and not just paused. To re-run the job, it will have to be recreated and launched.

Clear List will clear all finished jobs from the manager window. This includes successful, unsuccessful, and manually stopped jobs but does not affect currently running jobs.

Below the two buttons is a group of thread notifications. The "running" count shows how many multithreaded operations are currently being executed. The "waiting" count shows how many jobs are enqueued for processing. The "threads" count shows how many threads are available for job processing.

The total number of threads is based on the thread pool available for the local system. The pool is equal to the number of CPUs on the system, thus a higher number of processors means more threads are available.

The job manager has a context menu available, shown below.

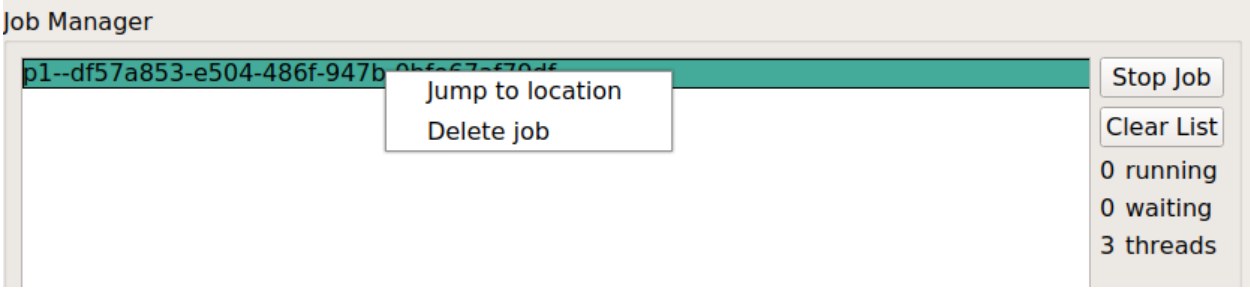


Figure 40: Job Manager Context Menu

This menu provides two capabilities: jump to job output and delete selected job. Jumping to job output scrolls the Output Console to the start of the selected job's output, highlighting the specific job name for the worker, as shown below.

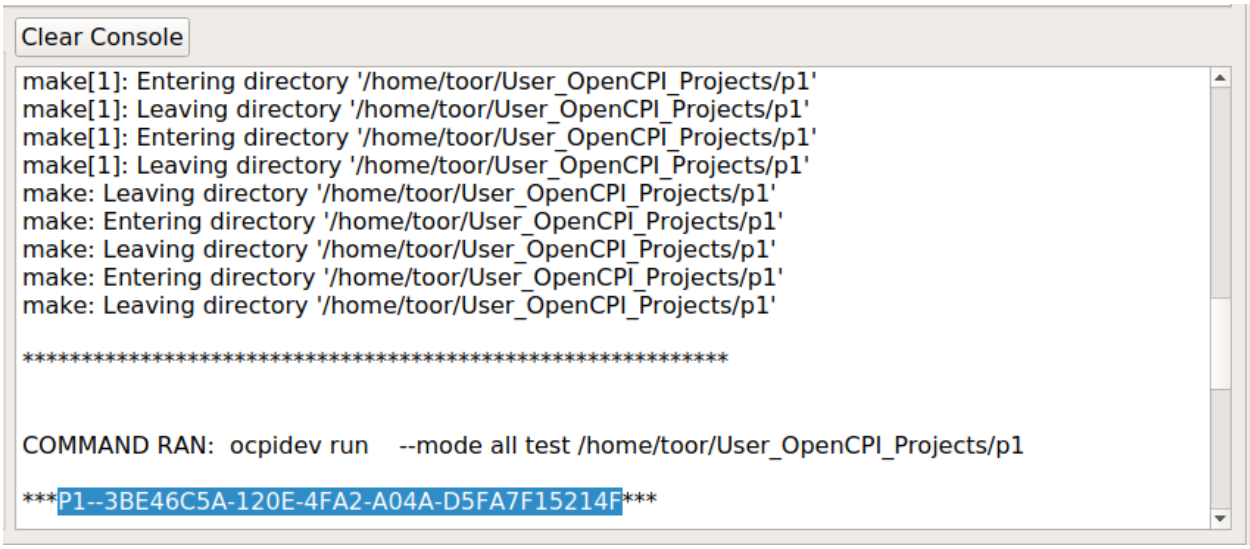


Figure 41: Job Manager "Jump to location" Command

Deleting the selected job removes that particular job from the job manager window. This is different from the Clear List button, which removes all jobs from the manager window.

13 Output Console

The output console (shown below) is a redirection of the OS terminal. In other words, what you would normally see when running command line **`ocpidev`** will be shown in the GUI console. However, it is output only; it does not accept input commands.

The Clear Console button allows the user to clear the information in the console as desired. This is most commonly used between operations so that each command's output is all that is displayed. However, if multiple jobs have been executed, using the "Jump to location" right-click option in the Job Manager will scroll the console's content to the start of the selected job's output.

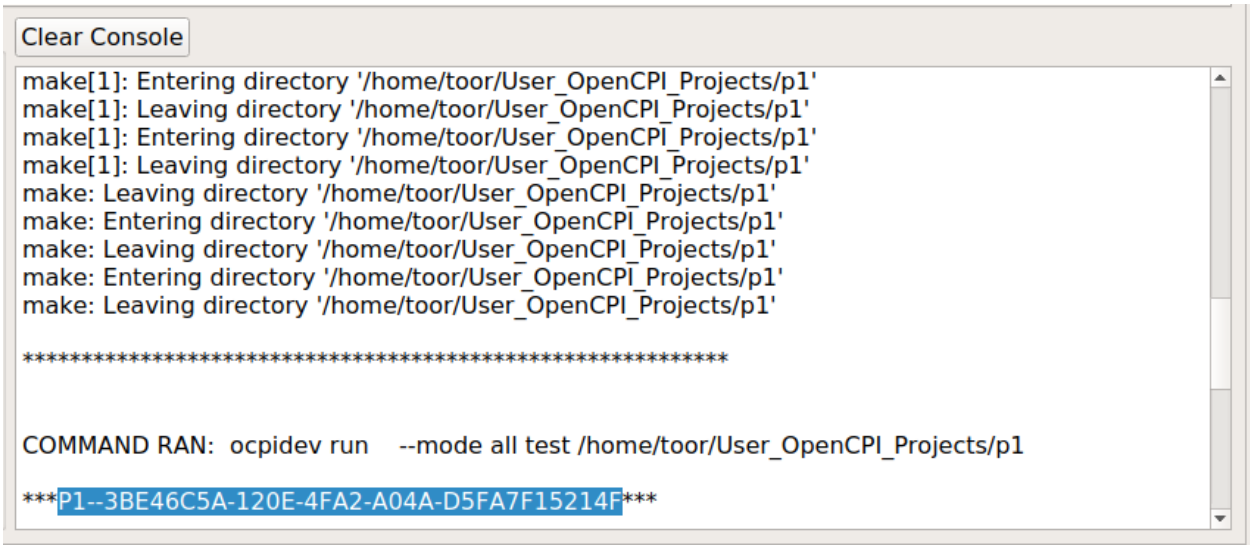


Figure 42: Output Console Example

14 Functionality Notes

The following sections provide information about particular functionality of the OpenCPI GUI.

14.1 Reusing Project Names

If a project is deleted and a new project, with the same name, is created an error will appear in the Output Console, shown below.

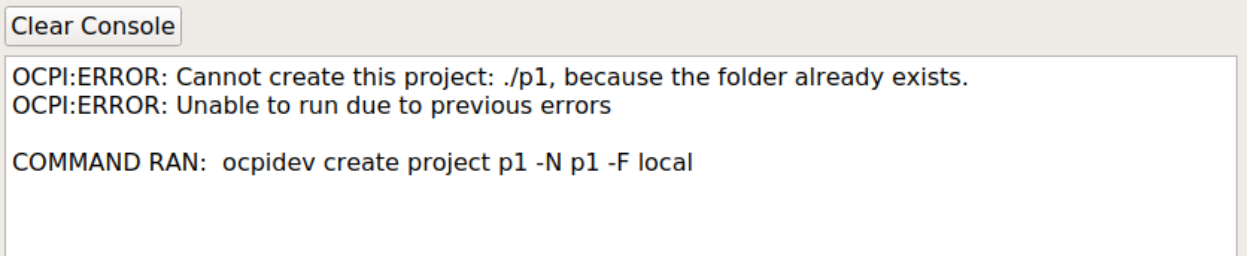


Figure 43: Project Name Reuse Error

This error is erroneous, as the project will still be created. Certain metadata is maintained in the file system when projects are created and deleted, leading to some confusion within the GUI about the current status of a project.

While project name reuse can be performed, to avoid confusion with the error message, it is recommended to avoid reusing project names.

14.2 Editing Files

As noted previously, certain files can be edited via the GUI. Editing is provided via the Linux **xdg-open** command, meaning whichever editing program has been set in the OS for that particular file type will be opened. Most commonly, this will be the default text editor for the OS, but if the user has changed the application associated with a particular file type, it may be different.

14.3 Unit Tests and HDL Targets

Unit tests will fail if the HDL groupbox is checked but HDL target is set to NONE, as this will cause an empty **`–hdl-platform`** argument to be passed to the ocpidev run command. Therefore, if an HDL target needs to be used, it is best to uncheck the HDL Targets groupbox.

14.4 RCC Workers

There is a known issue with OpenCPI regarding RCC workers and C++ files. When using the command

`ocpidev create worker <name>.rcc -L c++ -S <spec>`

the following error may occur

`Error: Invalid language "c++" for model "rcc". Available: "c" "c++"`

Depending on system configuration, this has been found to occur in both the GUI and CLI execution. This will be addressed in a future OpenCPI update.

A workaround for this is to remove the **`-L c++`** portion of the command, as the default functionality is to create C++ files:

`ocpidev create worker <name>.rcc -S <spec>`