RPM Installation Guide

Version 1.4

Revision History

Revision	Description of Change	Date
v1.0	Initial Release	2/2016
v1.1	Updated for OpenCPI Release 1.1	3/2017
v1.2	Updated for OpenCPI Release 1.2	8/2017
v1.3	Updated for OpenCPI Release 1.3	2/2018
v1.3.1	Updated for OpenCPI Release 1.3.1	4/2018
v1.4	Updated for OpenCPI Release 1.4	9/2018

Table of Contents

List of Tables

1 References

This document assumes a basic understanding of the Linux command line environment. It does not require a working knowledge of OpenCPI. However, it is recommended that the user read the *Getting Started* document (up to the "Installation of OpenCPI" section) or reference the *Acronyms and Definitions* document for various terms used within.

Table 1: References

Title	Published By	Link
Getting Started	ANGRYVIPER Team	`Getting_Started.pdf`
Acronyms and Definitions	ANGRYVIPER Team	`Acronyms_and_Definitions.pdf`
Overview	ANGRYVIPER Team	`http://opencpi.github.io/Overview.pdf`
Installation Guide[1]	OpenCPI	`https://opencpi.github.io/OpenCPI_Installation.pdf`
Component Development Guide	OpenCPI	`https://opencpi.github.io/OpenCPI_Component_Development.pdf`
RCC Development Guide	OpenCPI	`https://opencpi.github.io/OpenCPI_RCC_Development.pdf`
HDL Development Guide	OpenCPI	`https://opencpi.github.io/OpenCPI_HDL_Development.pdf`
FPGA Vendor Tools Installation Guide	ANGRYVIPER	`https://opencpi.github.io/FPGA_Vendor_Tools_Installation_Guide.pdf`
Managing Software with `yum`	CentOS Project	`https://www.centos.org/docs/5/html/yum/`
CentOS Deployment Guide: Useful `yum` commands (*e.g.* `yum localinstall`)	CentOS Project	`https://www.centos.org/docs/5/html/5.2/Deployment_Guide/s1-yum-useful-commands.html`

[1]The RPM installation process is quite different from the process explained in the OpenCPI Installation Guide, but the OpenCPI Installation guide has applicable post-installation information for PCI-based boards, etc.

2 Document Overview

This document describes how to **install OpenCPI at a system level** on a development host for multiple users via RPMs. The host installation allows for local software-based execution of OpenCPI applications and components, cross-building for non-x86 platforms, simulation of HDL, and, when available, hardware testing. **Upon completion of this Guide, the steps described in the *Getting Started Guide* must be followed by *each* OpenCPI user.**

The default host installation platform for OpenCPI development is CentOS 6 or CentOS 7 Linux x86_64 (64-bit). Other Linux variants and 32-bit systems have been used successfully, but this document expects the OS to be CentOS 7. Development hosts can either be actual physical systems or virtual machine installations.

This document assumes that CentOS is already installed and proper administrative privileges have been established.

Additional installation options exist for other target processors and technologies such as the Xilinx Zynq SoC (with ARM processor cores and FPGA resources). Preference when targeting non-x86 architectures is given to *cross-building*, rather than self-hosting development. This limits the complexity of installing tools on different development hosts.

Installation of OpenCPI is completed in the following steps:

1. **Section 3**: Acquiring the OpenCPI framework
2. **Section 4**: Installing the OpenCPI framework
3. **Section 5**: Setting up the OpenCPI environment
4. **Section 6**: Testing the OpenCPI installation

These steps result in a development system with tooling and runtime software ready to support development and native execution of OpenCPI components and applications.

3 Acquiring the OpenCPI framework

Currently, the ANGRYVIPER Team releases DVD-Rs containing the RPMs and PDF documentation of the OpenCPI framework. These are also available on `https://opencpi.github.io/`.

4 Installing OpenCPI framework

The ANGRYVIPER Team's recommended installation method for development is through the use of RPMs. The framework can be built from source for a development host, but is not recommended.

The ANGRYVIPER Team provides RPMs to their direct customers and other users can find them on `github.io`.

Understanding OpenCPI RPM naming convention

OpenCPI's RPM naming follows that of the Red Hat Package Manager recommendations of `<name>-<version>-<release>.<dist>.<architecture>.rpm` where:

1. *name* is the name describing the packaged software
2. *version* is the version of the packaged software
   (a) version following the Major.Minor.Sub-minor naming schema
3. *release* is the number of times this version of software has been packaged
   (a) this number is independent of the version

4. *dist* is the OS distribution that the package is built for (*e.g.* `.el7.centos`)
5. *architecture* is shorthand name describing the type of hardware the packaged software is to be installed on
6. "*devel*" is sometimes appended to the package's name to indicate development RPMs which are required for building from source

When to Install

It is recommended that the user install these packages *before* additional tools described in Section 5.1 because the RPMs force the installation of some otherwise-hidden dependencies, *e.g.* 32-bit X11 libraries for ModelSim.

4.1 Installing OpenCPI from RPMs

It is recommended that the user installs all available packages whenever possible. If limited by available disk space, Table 2 can be used to help determine which of the packages should be installed based upon the intended use of the target machine.

Within OpenCPI, there are two types of implementations, called *Workers*, that are used in this framework: Resource-Constrained C Language (RCC) Workers and Hardware Description Language (HDL) Workers. RCC Workers are written using either C or C++ and are designed for either x86 or ARM architecture, while HDL Workers are written in VHDL and are designed for Field Programmable Gate Arrays (FPGAs) or HDL Simulators. For further details regarding RCC and HDL Workers see the *OpenCPI RCC Development Guide* and the *OpenCPI HDL Development Guide* (cf. Table 1).

Table 2: RPM Decision Guide

	Runtime RCC Host	Runtime HDL Host	RCC-Only Development (x86 RCC exclusive)	RCC/HDL Development (x86 RCC, non-SoC[1] FPGA HDL)	RCC/HDL Development (Targeting non-x86 HW/SW platform)
`angryviper-ide...rpm`			✓	✓	✓
`opencpi-...rpm`	✓	✓	✓	✓	✓
`opencpi-debuginfo...rpm`			✓	✓	✓
`opencpi-devel...rpm`			✓	✓	✓
`opencpi-doc...rpm`			✓	✓	✓
`opencpi-driver...rpm`		✓		✓	✓
`opencpi-project-bsp...rpm`[2]			✓	✓	✓
`opencpi-*-platform...rpm`					✓

[1] "Non-SoC" meaning a standalone FPGA *without* an integrated processor, *e.g.* Xilinx ML605.
[2] BSP RPMs may not be provided with the standard/basic RPMs, but represent a placeholder for RPMs providing Board Support Package Projects.

The RPMs each have specific usage. Table 3 outlines what each of the RPMs are used for.

Table 3: RPM Descriptions

RPMs	Description
`angryviper-ide-*.x86_64.rpm`	The ANGRYVIPER IDE (Eclipse with plugins). See Appendix B for an alternative method to set up the IDE using an existing Eclipse installation.
`opencpi-*.x86_64.rpm`	Base installation RPM includes the runtime portion of the Component Development Kit (CDK) and the source for the `ocpi.core` and `ocpi.assets` Projects containing framework essential components, workers, platforms, etc.
`opencpi-debuginfo-*.x86_64.rpm`	Debug symbols needed to debug the framework.
`opencpi-devel-*.x86_64.rpm`	Additional header files and scripts for developing new assets as HDL and/or RCC.
`opencpi-doc-*.x86_64.rpm`	Includes most of the documentation found at github.io. A symlink can be found at `/opt/opencpi/documentation.html`. If you receive the RPM directly from the AV team, it may include BSP documentation that is not available on GitHub.
`opencpi-driver-*.noarch.rpm`	OpenCPI driver. Once installed, any subsequent kernel updates will cause the driver to be built automatically on restart.
`opencpi-hw-platform-X-Y-*.noarch.rpm`	Additional files necessary to build the framework targeting specific hardware platform "X" when running RCC platform "Y" ("Y" *can* be "`no_sw`"). This RPM also includes hardware-specific SD Card images when applicable.
`opencpi-project-bsp-*.noarch.rpm`	A `*.bsp.*` Project (*e.g.* `ocpi.bsp.e3xx`) contains a Board Support Package for a particular physical radio, e.g. RCC/HDL Platform Support, Device Workers, etc. There are certain BSPs which are located in the `ocpi.assets` Project and therefore do not require their own separate BSP RPMs. As noted in Table 2, these RPMs are *only* needed for development; the `hw-platform` RPMs contain all required runtime files to deploy to an SD Card.
`opencpi-sw-platform-*.noarch.rpm`	Additional files necessary to build the framework targeting specific RCC/software platforms, independent of the final deployed hardware.

4.1.1 Installing from `github.io`

Installation may be completed[1] using the publicly available RPMs on github.io with the following reference commands:

Configure Yum Repository

```
$ sudo yum install yum-utils
$ sudo yum-config-manager --add-repo=https://opencpi.github.io/repo/opencpi-v1.4.0.repo
```

Install External Dependencies

```
$ sudo yum install epel-release
$ sudo yum install libXft.i686 libXext.i686
```

List Available Packages

To see packages available in the repository (to cross-reference with Tables 2 and 3):

```
$ yum list 'opencpi*'
```

[1]This will not include the ANGRYVIPER IDE; see Appendix B for installation

Install

You can choose individual packages to install (cf. Tables 2 and 3), or install *every* OpenCPI package available:
`$ sudo yum install 'opencpi-*'`

4.1.2 Installing from DVD-R

Installation may be completed using yum with the following command:

```
$ sudo yum localinstall --nogpgcheck <location of RPMs>/*rpm
```

5 Setting up the OpenCPI Environment

5.1 HDL Simulator(s) and/or Compiler(s)

For FPGA development and/or HDL simulation, OpenCPI requires vendor-provided tools (*e.g.* Xilinx Vivado, Mentor Graphics ModelSim). Refer to the *FPGA Vendor Tools Installation Guide* from Table 1 for instruction in installing and configuring these tools for use with OpenCPI.

Keep note of where the *license files* are, the *version number* of the tools, and *where the tools are installed*, as this information will be needed to configure the required environment variables.

5.2 The `opencpi` Group

At this point, certain users should be added to the `opencpi` group. When a user creates a Project, it is likely that the Project should be `registered`. Registering a Project allows other users and Projects to access its assets. The default Registry on an RPM-configured system is located at `/opt/opencpi/project-registry`. In order for a user to register Projects in this default location, the user will need to be a member of the `opencpi` group. To add a user to the `opencpi` group, run the following command:

```
% sudo usermod -aG opencpi <username>
```

If this command is run as user `<username>`, the user will need to log out and back in to apply this change.

Note that users can use a personal non-default Project Registry. For more information on this, please visit the *OpenCPI Component Development* document or the *Getting Started Guide* (cf. Table 1).

5.3 Setup Environment

> The Framework tries very hard to accept vendor default installation and configuration without additional settings. This section is only required if Section 5.1 and/or the *FPGA Vendor Tools Installation Guide* required a non-standard configuration.

Setting up the environment when installing from RPM requires root privileges. Navigate to `$(OCPI_CDK_DIR)/env.d` and notice the following example scripts:

- `altera.sh.example`
- `modelsim.sh.example`
- `site.sh.example`
- `xilinx.sh.example`

Every time a new `bash`[2] *login* shell is opened, all `*.sh` files in `/opt/opencpi/cdk/env.d` are executed, and all `*.sh.example` files in `/opt/opencpi/cdk/env.d` are *ignored*. To enable a script for execution, the name of the script

[2]Some problems have been reported when the user's shell is set to `/bin/sh` and not `/bin/bash`.

must be changed so that the `.example` suffix is removed. A simple demonstration is below:

```
% sudo cp altera.sh.example altera.sh
```

Now `altera.sh` will execute every time a new shell is opened.

If using the Altera tools, the `altera.sh` will need to be created and the variables `OCPI_ALTERA_DIR`, `OCPI_ALTERA_VERSION`, and `OCPI_ALTERA_LICENSE_FILE` must be defined in `altera.sh`. The `altera.sh` script also calls another script to set up the rest of the variables needed for the Altera tools.

If using the ModelSim tools, the `modelsim.sh` will need to be created and the variables `OCPI_MODELSIM_DIR` and `OCPI_MODELSIM_LICENSE_FILE` must be defined in `modelsim.sh`.

If using the Xilinx tools, the `xilinx.sh` will need to be created and the variable `OCPI_XILINX_LICENSE_FILE` must be defined in `xilinx.sh`. If using an installation of Xilinx Vivado that was *not* installed in the default `/opt` directory then the variable `OCPI_XILINX_VIVADO_DIR` must be defined in `xilinx.sh`. If using a version other than the most recent one installed in that location, then the variable `OCPI_XILINX_VIVADO_VERSION` must be defined in `xilinx.sh`. If using an installation of Xilinx ISE that was *not* installed in the default `/opt` directory then the variable `OCPI_XILINX_DIR` must be defined in `xilinx.sh`. If not using the 14.7 version of ISE, then the variable `OCPI_XILINX_VERSION` must be defined in `xilinx.sh`. The `xilinx.sh` script also calls another script to set up the rest of the variables needed for the Xilinx tools. See the *FPGA Vendor Tools Installation Guide* (cf. Table 1) for more information on Xilinx license setup.

The script `site.sh.example` has been provided as an example central location where any other variables can be defined globally. *Remember that the names of the scripts do not matter; only the `*.sh` extension.* More configuration variables can be found in the *Getting Started Guide*.

Once all the desired scripts have been created and edited, open a new shell and check to see that the environment is now set up.

5.4 Removing OpenCPI RPMs

In the event that the OpenCPI RPM needs to be uninstalled, or reinstalled, the best way to remove the OpenCPI RPM is to use `yum` to erase the RPMs from Table 3 as seen below:

```
% sudo yum erase <RPM name>
```

6 Testing the OpenCPI installation

To verify the OpenCPI installation, there is a command `ocpitest` that presents various test options. `ocpitest --showtests` will list all available. Some require additional files to be present or Projects to be built, but for a fresh RPM install, you can use:

```
% ocpitest driver os datatype load-drivers container
```

The first test, `driver`, will require `sudo` access. A successful install will output "All tests passed." at the end of the test.

Appendices

A Prerequisites and Their Modifications

This section provides a list of various Free and Open Source software required by the Framework. They are included within the RPMs behind the scenes to allow the Framework to function, as well as provide utility to RCC Workers. OpenCPI's packaging of these ensures they will not conflict with other[3] installed copies by using a non-standard installation location. Listed below are any explicit modifications required, but implied with every item are possible modifications to the build configuration to override the library's final installation location along with cross-compilation targeting various platforms.

A.1 Analog Devices' AD9361 no-OS Library

Source: `https://github.com/analogdevicesinc/no-OS.git` (tied to specific git hash)
Diff: `projects/assets/prerequisites/ad9361/ad9361.patch`

- Patches to allow older compilers to compile (missing `stdint.h` includes)
- Fix memory leaks on de-allocation
- Move some top-level structs from `common.h` into `ad9361.h` to limit scope of items, *e.g.* "`struct clk`"

A.2 GNU Multiple Precision Arithmetic Library (GMP)

Source: `https://mirror.csclub.uwaterloo.ca/gnu/gmp/gmp-6.1.2.tar.xz`
Diff: (N/A)

A.3 Google Test (GTEST)

Source: `https://github.com/google/googletest/archive/release-1.8.0.zip`
Diff: (N/A)
Note: Not available to RCC Workers

A.4 Liquid DSP

Source: `https://github.com/jgaeddert/liquid-dsp/archive/v1.3.1.tar.gz`
Diff: `projects/assets/prerequisites/liquid/malloc.patch`

- Disable `autoconf` macros for `malloc()` and `realloc()` that cause missing `rpl_malloc` symbols for some cross-compilers; assumes cross-compilers have sane/modern `malloc()` implementations

A.5 Lempel-Ziv-Markov chain algorithm (LZMA/XZ)

Source: `https://tukaani.org/xz/xz-5.2.3.tar.gz`
Diff: (N/A)

A.6 PatchELF

Source: `http://nixos.org/releases/patchelf/patchelf-0.9/patchelf-0.9.tar.gz`
Diff: (N/A)
Note: Not available to RCC Workers

[3] OS vendor, EPEL, other third-party-packagers, etc.

B Appendix - Add the ANGRYVIPER IDE to Eclipse using the Plugin

The ANGRYVIPER IDE is constructed using the Eclipse Neon release and a plugin developed by the ANGRYVIPER team. Since the entire IDE is too large to be placed on GitHub, the following instructions may be used to obtain the IDE by downloading Eclipse and installing the plugin as an Eclipse drop-in.

Download Plugin JAR

1. Obtain the latest ANGRYVIPER plugin jar file

   ```
   wget https://opencpi.github.io/ide/av.proj.ide.plugin_1.4.jar
   ```

B.1 Eclipse (Neon Release)

1. Download the Eclipse Neon IDE for C/C++ Developers

   URL: https://www.eclipse.org/neon/

2. Install Eclipse by extracting the archive in the desired location

3. Start Eclipse

   Go into the folder where it was installed and click/run `eclipse`

4. Put the `av.proj.ide.plugin_*.jar` file in the `eclipse/dropins` folder

5. Install Sapphire via the Eclipse Marketplace

   In Eclipse, navigate to "Help → Eclipse Marketplace". Search for "Sapphire". There should be one search result for Sapphire. Click the "Install" button. Sapphire and its dependencies will be installed.

6. Restart Eclipse when prompted.

7. Eclipse now has the ANGRYVIPER IDE functionality.

B.2 Eclipse (Oxygen Release)

The process to construct the IDE is the same as described above using the Oxygen release for C/C++ Developers.

Note: At this time, the ANGRYVIPER Team has not been able to 100% verify using the plugin in Oxygen release. Eclipse Oxygen changed an API that caused problems for Sapphire, and Sapphire 9.1.1 has been released to correct the issue. The unknown part of the process is whether or not the Eclipse Marketplace will have the new version of Sapphire. If it does not, it can be installed manually as follows:

1. In Eclipse, navigate to "Help → Install New Software".

2. Add the Sapphire 9.1.1 repository

   Click the "add" button (to add a new repository site), fill in the popup form:

   ```
   name:  Sapphire9.1.1
   location:  http://download.eclipse.org/sapphire/9.1.1/repository/
   ```

3. Click "OK" to add it

4. Select the down arrow at the end of the "work with:" input. Select the new Sapphire repository.

5. Select Sapphire. If Samples and Tests appear in the list; deselect them.

6. Install