Center for Simulation of Advanced Rockets

University of Illinois at Urbana-Champaign

Rocprep User's Guide

Center for Simulation of Advanced Rockets
University of Illinois at Urbana-Champaign
2270 Digital Computer Laboratory
Urbana, IL

Title:	Rocprep User’s Guide
Author:	Mark Brandyberry and Courtlan McLay
Subject:	
Revision:	1
Revision History	Revision 1: Update new features, correct NDA structure description to new format Revision 0: Initial Release
Effective Date:	8/20/2008

1.0 Introduction

Rocprep is a utility for assembling and preparing datasets for *Rocstar* multiphysics simulations. *Rocprep* was written with three goals in mind:

- Automate the tedious and error-prone process of running the physics module preptool(s) on native mesh files to produce fully-partitioned initial solutions, ready for a *Rocstar* simulation.

- Allow both new and experienced users easy access to a centralized archive of well-tested *Rocstar* datasets, rockets and other verification and validation problems. This archive is called a *Native Data Archive*, or *NDA.*

- A portable implementation which is available on all of *Rocstar's* target platforms.

1.1 Rocprep Overview

Rocprep is a sophisticated, object-oriented Perl program designed to ease the process of setting up *Rocstar* datasets. It takes as inputs native mesh files produced by Gridgen, Patran, Truegrid, etc; and pre-prepared *Rocstar* text input files specifying physics parameters, boundary conditions, and other simulation criteria. It produces as output a complete *Rocstar* dataset hierarchy of directories and files, with the problem domain partitioned and initialized for a parallel simulation. Each physics module in the *Rocstar* suite has a unique and specific process for transforming native mesh files into *Rocin/Rocout* compatible solution files. This process is done offline, before a simulation can be run with *Rocstar*, and involves one or more utility programs called preptools.

In essence, *Rocprep* can be considered a rudimentary expert-system that has two basic stages of operation: extraction from an archive and preprocessing. The first stage, extraction, copies required text input files and native mesh formats into place in a user-prescribed new *Rocstar* dataset directory. For the second stage, preprocessing, *Rocprep* presents a convenient unified interface for running the necessary *Rocstar* physics module preptools on a newly created dataset.

There are two additional "sanity check" stages in *Rocprep* that bracket the extract-preprocess steps. These two stages perform a simple checkup to ensure that the necessary files exist in the archive before extraction, and lastly, ensure that all the necessary files for the *Rocstar* simulation have been successfully created after the preprocessing stage. Any errors encountered will be flagged and written to standard output and a logfile for later inspection.

1.2 Related Documents

- "Rocstar 3 User's Guide"

- "RocfloMP User's Guide"

- "The RocfluMP Book"

•"Rocfrac User's Guide"

•"Rocsolid User's Guide"

2.0 Purpose and Methods

Rocprep's purpose is to simplify the assembly and preparation of *Rocstar* simulation datasets. To accomplish this, *Rocprep* has two essential functions:

- Allow users to access a centralized archive of Quality-assured *Rocstar* problems. The archive, called a *Native Data Archive (NDA)*, will contain the raw components of datasets for simulation that may be performed with *Rocstar*. Each problem may be subdivided further into geometry variations, meshes of different coarseness, and simulation parameter settings. The NDA format is described in more detail in section 3.

- Provide a convenient interface for running all the necessary *Rocstar* physics module preptools to produce an initialized and partitioned dataset, ready to run in parallel on a fixed number of processors.

Rocprep is an object-oriented Perl script, and it is designed to run on all platforms supported by the *Rocstar* simulation suite. Presently, *Rocstar* target platforms include variants of AIX, Linux, Solaris, Irix and OS X. Because *Rocprep* is written in Perl, it is independent of the *Rocstar* simulation build. It has no library dependencies on any of the *Rocstar* dynamic libraries. This means that the *Rocprep* script can be directed to use preptools compiled with different makefile switches, for example: with and without CHARM=1.

There are four basic stages of operation in the *Rocprep* pipeline:

- Check that the archive files exist and are accessible by the user.

- Extract (copy) the archive files to create a new dataset directory hierarchy.

- Run the necessary physics preptools, preprocessing the dataset.

- Check that the expected *Rocstar* input files are now in place in the new dataset.

2.1 Check the Archive Files

Rocprep begins by checking that all necessary "raw" simulation files exist in the Native Data Archive directory. These files include mesh format files such as Plot3D, Cobalt, Patran Neutral File and Nike3D. They also include text input files such as RocstarControl.txt, module-specific Control.txt files, ".inp" input parameter files, and ".bc" boundary condition files. Any missing file will cause an error, aborting the *Rocprep* execution. A more detailed description of the files and directories required for an archive is given in section 3.1.

2.2 Extraction

Next, *Rocprep* will copy the selected groups of meshes and text input files for each physics module from the NDA location to a user-defined location, creating and populating the directory structure for a new *Rocstar* dataset. *Rocprep* uses Unix local copy commands, so the NDA and the new dataset extraction directory must both be accessible on the same platform.

2.3 Preprocessing

After extracting and creating a directory hierarchy for a *Rocstar* dataset, *Rocprep* will invoke system calls to the necessary physics preptools. Each of the fluids modules require a multi step process: *Rocflo* uses *makeflo* and *rfloprep*, while *Rocflu* uses *rflumap* (twice) and *rfluprep*. The solid mechanics modules are simpler, *Rocfrac* uses *rfracprep*, and similarly *Rocsolid* uses *rsolidprep*. If the user chooses a coupled simulation (both fluids and solids), the *surfdiver* utility is run to create the interface meshes for solution transfer by *Rocface*. *Rocprep* will store each preptool output in a logfile at the root of the problem directory, for inspection should any errors occur during this stage.

2.4 Check the Rocstar Dataset

Finally, *Rocprep* will check to ensure that the expected simulation files have been created successfully by the preprocessing tools. This stage involves parsing the *Rocin* control files, which are text files that tell *Rocstar* how to map HDF/CGNS solution files to processors.

3.0 Building and Running

To obtain *Rocprep*, first sign up for a CSAR CVS account. Follow the instructions given on the web page:

```
http://musgrave.csar.uiuc.edu/CSARCVS_access.htm
```

Rocprep is checked into the CSAR CVS repository in the /*Rocstar/Rocprep*/Codes directory. Check out the code by issuing the command:

```
>$ cvs co Rocstar/Rocprep/Codes
```

The Rocprep code consists of the following 11 Perl modules:

```
Rocprep.pm
RunLogControl.pm
Properties.pm
RocprepProperties.pm
ModuleProcessor.pm
RocfaceProcessor.pm
RocstarProcessor.pm
RocfloProcessor.pm
RocfluProcessor.pm
RocfracProcessor.pm
RocsolidProcessor.pm
```

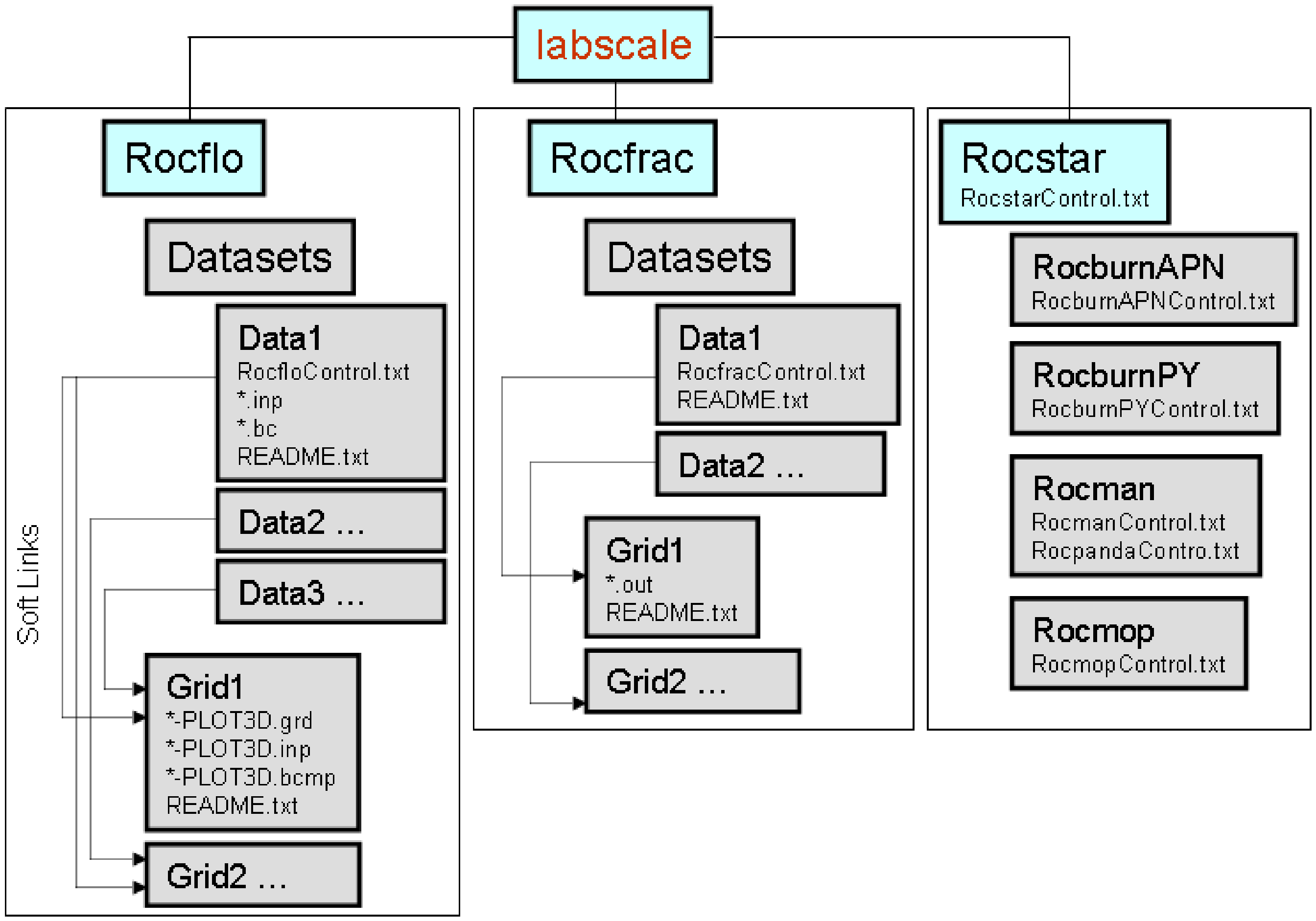
Because *Rocprep* is written in Perl, it is interpreted, not compiled. *Rocprep.pm* is the only executable module, which is run by issuing `./Rocprep.pm` at the command line. At present, *Rocprep* requires that Perl version 5.6 or newer be installed on the target platform.

Before describing the general use of *Rocprep*, it is necessary to understand the structure of the Native Data Archive from which it creates new *Rocstar* datasets.

3.1 Native Data Archive Format

The *Native Data Archive*, NDA, provides centralized storage of well-tested *Rocstar* datasets. Currently, the primary set of NDAs are maintained on `/turing/projects/csar/NDAs`, for broad access by all CSAR-group *Rocstar* users.

Each NDA has a well-defined directory hierarchy. An NDA is organized at the root by a problem name, for example a rocket like the labscale might have a root directory named labscale. The problem name is arbitrary, but generally descriptive of the family of datasets contained therein. Under the problem directory, there are directories for each physics module for which problem data is available: *Rocflo, Rocflu, Rocfrac, Rocsolid*. Each module may be subdivided further into geometry variations, meshes of different coarseness, and simulation parameter settings. The different simulation text input files are grouped in directories under a main Datasets directory named *Data1, Data2*, etc; while the mesh variations are stored under *Grid1, Grid2,* etc. as shown below in the NDA Organization figure. The data directories are soft-linked to the grid directories as appropriate. For example, if the Data files in Data1 and Data4 will work with Grid1, and the Data files in Data1, Data2, and Data3 will work with Grid2, then softlinks are formed in the NDA between Data1 and Data4 in Grid1, and Data1, 2, and 3 in Grid2. In this fashion, the Data directories and Grid Directories are kept separate, without duplication, and the soft links allow maintaining information on which Dataset(s) work with which Grids. Each Data and Grid subdirectory contains a `README.txt` file, with a brief description of the contents of the directory.



NDA Organization

An example showing an NDA for the labscale rocket is given above, with subdirectories for only two physics modules (*Rocflo* and *Rocfrac*) for brevity. The *Rocstar* subdirectory contains the simulation control file, `RocstarControl.txt`. Under the *Rocstar* directory are the *Rocburn* module control files, each in a corresponding subdirectory. Also, *Rocman* and *Rocmop* subdirectories contains control files for *Rocman*, *Rocpanda* and *Rocmop* modules. See the *Rocstar 3 User's Guide* for what these files should contain.

3.2 Single Step Example – Processing Everything at Once

The simplest way to use *Rocprep* is to extract from an NDA and preprocess in one step. First, the user must choose which physics modules are needed for the problem at hand. Examine the available data in the NDA by reading each `README.txt` description, and choosing the appropriate Data and Grid subdirectories for each module; fluid and/or solid.

The *Rocprep* example given below is for the labscale rocket dataset, a coupled fluid-solid interaction using the *Rocflo* and *Rocfrac* modules in Rocstar. Assume that the user has determined the mesh geometry variant desired for both *Rocflo* and *Rocfrac*, and selected problem parameter sets that correspond. Next, build *Rocstar* in a location accessible on the target platform, in this case we use the *Rocstar* default directory *genx/Codes/bin* under the user's home. For more information on *Rocstar* build options and instructions, see the *Rocstar 3 User's Guide*. Finally the user should decide on the number of parallel partitions desired for the problem, for the labscale a typical partitioning is 16 processors. This number depends on the size of the meshes and the resources available on the simulation target platform. It is a tradeoff between communication speed and computational speed.

```
>$ Rocprep.pm –A –d /csar/NDAs/labscale –t my_new_labscale
     –o 1 1 –f 2 4 –n 16 –p genx/Codes/bin
```

Switch	***Description***
`-A`	major mode is **All**, do all four stages
`-d`	choose the labscale rocket problem archive located at */csar/NDAs/labscale*
`–t`	create a new Rocstar dataset directory in the current working directory, named *my_new_labscale*
`–o 1 1`	choose the Data1 and Grid1 subdirectories under *labscale/Rocflo*
`–f 2 4`	choose the Data2 and Grid4 subdirectories under *labscale/Rocfrac*
`–n 16`	partition the problem for 16 processors
`–p`	use the *Rocstar* preptools compiled in the directory *genx/Codes/bin*

The result of this example will be a fully partitioned and initialized *Rocstar* dataset located in the user's current working directory, under a new directory named *my_new_labscale*.

3.3 Two Step Example – Extract, move, Preprocess

If the platform hosting the NDA does not have sufficient resources to run the preptools, it may be desirable to split the dataset creation process into two steps: extraction and preprocessing. In the example below, the same coupled labscale rocket dataset (shown in section 3.2) is extracted from the NDA, but the preprocessing step is delayed:

```
>$ Rocprep.pm –E –d /csar/NDAs/labscale –t my_new_labscale
     –o 1 1 –f 2 4
```

Switch	***Description***
`-E`	major mode is **Extract,** *Rocprep* will extract the files from the NDA and stop
`-d`	choose the labscale rocket problem archive located at */csar/NDAs/labscale*
`–t`	create a new *Rocstar* dataset directory in the current working directory, named *my_new_labscale*
`–o 1 1`	choose the Data1 and Grid1 subdirectories under *labscale/Rocflo*
`–f 2 4`	choose the Data2 and Grid4 subdirectories under *labscale/Rocfrac*

In this mode, a set of raw input files and native mesh formats are extracted and structured in a *Rocstar* input directory hierarchy under *my_new_labscale*. At this point, the user can optionally tar/gzip the files and copy them to a remote platform with more resources before continuing.

To complete the dataset, the second step, preprocessing, proceeds as the following command is issued:

```
>$ Rocprep.pm –P –d my_new_labscale –p genx/Codes/bin –n 16
```

Switch	***Description***
`-P`	major mode is Preprocess
`-n 16`	partition the problem for 16 processors
`-p`	use the Rocstar preptools compiled in the directory *genx/Codes/bin*

The preprocess option assumes that *my_new_labscale* is a directory that contains all the raw files necessary to create a new *Rocstar* dataset and pre-organized in the correct input hierarchy. Note that this command could also be used to perform all preprocessing on a *Rocstar* dataset carefully assembled by hand.

4.0 Input and Output (User Interface)

Rocprep uses a combination of command line arguments and a key-value based text control file, `RocprepControl.txt`, as inputs. *Rocprep* produces a *Rocstar* dataset directory and a collection of text logfiles as output. The following sections describe the input and output formats in more detail.

4.1 Command line Arguments

Rocprep has a variety of commandline arguments to control the flow of execution, specify *Rocstar* modules to extract and process, and manage the creation of *Rocstar* datasets. These command line arguments override any values set in the `RocprepControl.txt` file. The following table summarizes the current set of command line switches available:

Switch	*Description*
	Major Modes of Operation
`-A, --all`	Extract and preprocess
`-C, --check`	Check an existing dataset specified by -d <path>
`-E, --extract`	Copy NDA files to target directory specified by -t <path>
`-P, --preprocess`	Run *Rocstar* module preptools on raw data specified by -d <path>
	Physics Module Options
	[m] and [n] are optional Data and Grid subdirectory numbers used for extraction only. When running -P or -C, simply specify the switch itself.
`-o [m] [n]`	Select *Rocflo* extraction and/or preptools.
`-u [m] [n]`	Select *Rocflu* extraction and/or preptools.
`-f [m] [n]`	Select *Rocfrac* extraction and/or preptools.
`-s [m] [n]`	Select *Rocsolid* extraction and/or preptools.
`-b`	Select *Rocburn*. This switch is presently always on by default.
	Module-specific Flags
`-r <m>`	specify <m> regions (*Rocflu* only), default is -n value
`-splitaxis <n>`	force split along n=0,1, or 2 axis (*Rocflo* only – actually a *makeflo* option)
`-un <units>`	Convert model units to meters (Rocfrac only)
	General Options
`-i <o|u|f|s>`	Surfdive the interface meshes. The default action will infer surfdiver combinations from the physics switches given.
`-d <path>`	Path to source data. Default is current working directory.
`-h, --help`	Print a short usage message describing all switches and terminate
`-n <m>`	Specify a number of partitions/processors
`-r <m>`	Specify the number of regions. Rocflu only, see Rocflu User's Guide
`-t <path>`	Target path for a new Rocstar dataset. This is required if major mode is -E extract or -A all.
`-p <path>`	Specify the path to the Rocstar preptool binaries. The default will use the user's path.
`-x, --ignore`	Ignore the RocprepControl.txt file if it exists in the problem directory

The Major Modes switches are illustrated in section 3. –A, -E, and –P will generally be the three that will be used. The physics module options, when used with the –A flag will all be used as shown in section 3.2, giving the Data and Grids from each Physics solver to be used in the dataset. The general options, when used with the –A mode flag, will generally all be used except for –i and –x. It is generally required to give the path to the source data, the path to the target dataset to be produced, the number of partitions to preprocess, and the path to the preprocessing codes to use (which all must be stored in one directory). The –i switch will only be used in combination with the –P mode. The –x switch is only used if there is a need to preprocess the dataset (using the –P mode) in a way that is different than it was originally extracted in.

Finally, the Module-specific flags are specific preprocessing pass-through options for specific physics modules. –splitaxis is the only one that will be used often with *Rocflo*, and that is to restrict the direction of “slicing” the block model to partition it with the *makeflo* preprocessing utility. The 0,1, 2 input to the splitaxis flag correspond to the computational block i, j, and k directions. Further description of the –splitaxis parameter will be found in the *makeflo* user’s guide.

4.2 RocprepControl.txt File Structure

Rocprep obeys a key-value based text input file named `RocprepControl.txt` if it is located in the root of the problem directory. Each line in this file contains a keyword, generally all CAPITAL letters, followed by a separator (`->`) and a value. For example:

```
ROCFLO->1
ROCFLU->0
ROCFRAC->1
ROCSOLID->0
ROCFLOROCFRAC->1
SOURCEDIR->./
```

This file contains keywords for each of the four *Rocstar* physics modules, with *Rocflo* and Rocfrac set to “true” (1). The ROCFLOROCFRAC keyword is also set to (1), meaning that surfdiver will create interface meshes between the *Rocflo* and *Rocfrac* problem domains. This is equivalent to using the -o and -f commandline switches.

The control file is considered an advanced feature of *Rocprep*, and users are discouraged from editing or creating their own files. The format and use of this file is volatile, and may be subject to change with the next revision of *Rocprep*.

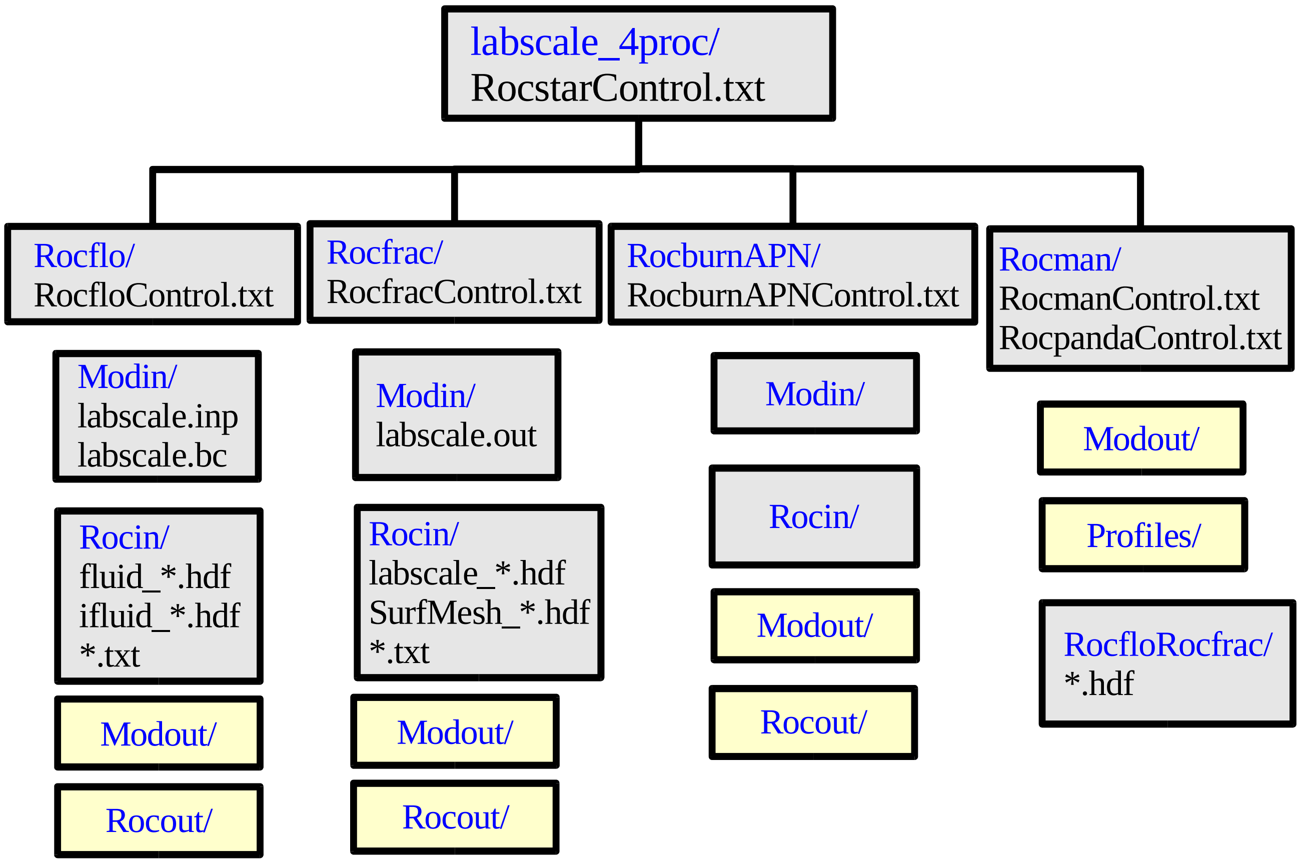
4.3 Rocstar Dataset Structure

A *Rocstar* dataset is a directory hierarchy with a root directory that is descriptive of the problem set. For illustrative purposes, assume we have a labscale rocket with both *Rocflo* and *Rocfrac* inputs, similar to other examples given in this Guide. The top-level directory in this case is named “*labscale_4proc*”, meaning a labscale dataset that has been partitioned for running on 4 processors. Inside the root directory is the simulation control file, `RocstarControl.txt`. This file contains information selecting the physics modules to run, the type of I/O to use, simulation start and stop times, tolerances, and other stopping criteria. See the *Rocstar 3 User's Guide* for more information.

Under the problem directory is a subdirectory for each physics module, in this case: *Rocflo*, *Rocfrac*, and *RocburnAPN*. There is an additional directory for *Rocman*, the module that controls the simulation timestep and coupling algorithm. Each of these modules has its own control file residing in the corresponding directory: `RocfloControl.txt, RocfracControl.txt, RocburnAPNControl.txt` and `RocmanControl.txt`. These files contain information and parameters specific to the module.

Inside each physics module subdirectory is a set of directories for storing additional input files. The Modin and Modout directories are for module-specific native input and output files respectively. The Rocin and Rocout directories are for *Rocstar* formatted I/O in HDF4 or CGNS files. The Rocin/Rocout files are volume and surface mesh information for the simulations, as well as *.txt files which *Rocin* uses to map these files to the correct processor in a parallel simulation. Interface surface meshes for *Rocface* are stored in `Rocman/RocfloRocfrac` in this example.

Before a simulation is run, the Modout and Rocout directories will be empty.



4.4 *Rocprep Log File*

After each run, *Rocprep* produces a series of logfiles. The main logfile, *rocprep.log*, will be written to the current working directory. This file contains the same information that is written to standard output during the *Rocprep* run. The file has a complete list of keyword-value pairs at the top, followed by a summary of each stage of the execution, with any error messages.

The following is an example `rocprep.log`, with the key-value section abbreviated:

```
***************************************************************************
Tue Mar  1 19:14:44 2005: Rocprep Initialized

ALL                = 1
BINDIR             = /home/cmclay/genx_charm/bin/
NUMPROCS           = 16
...
SOURCEDIR          = /csar/NDAs/labscale/
TARGETDIR          = /home/cmclay/my_new_labscale/

Tue Mar  1 19:14:44 2005: Checking NDA files

Ending phase: Check NDA Files for module RocfloProcessor.
Ending phase: Check NDA Files for module RocfracProcessor.
***************************************************************************

Tue Mar  1 19:14:44 2005: Extracting NDA files to rocstar dataset

Ending phase: Extract NDA Files for module RocfloProcessor.
Ending phase: Extract NDA Files for module RocfracProcessor.
***************************************************************************

Tue Mar  1 19:14:45 2005: Running preprocessor codes to make rocstar dataset

Ending phase: Run Preprocessors for module RocfloProcessor.
Ending phase: Run Preprocessors for module RocfracProcessor.
Ending phase: Run Preprocessors for module RocfaceProcessor.
***************************************************************************

Tue Mar  1 19:14:58 2005: Checking rocstar dataset files for consistency

Ending phase: Check Rocstar Dataset Files for module RocfloProcessor.
Ending phase: Check Rocstar Dataset Files for module RocfracProcessor.
Ending phase: Check Rocstar Dataset Files for module RocfaceProcessor.
***************************************************************************

Run terminated with error: NO ERRORS

***************************************************************************
```

The output from each physics preptool is also saved in a logfile at the root of the problem directory. In the *Rocflo/Rocfrac* labscale example, a total of four logfiles will be produced:

- makeflo.log
- rfloprep.log
- rfracprep.log
- surf_flofrac.log

If *Rocprep* fails during preprocessing, the user can investigate further by looking at the record in each preptool logfile.

5.0 Examples and Test Problems

There are no test cases for *Rocprep* as of this writing. The central Native Data Archive is located on the turing Apple Cluster. For more information, contact the author.