Community Climate System Model
National Center for Atmospheric Research, Boulder, CO

Community Land Model Version 3.0 (CLM3.0) User's Guide

Mariana Vertenstein, Keith Oleson, Sam Levis and Forrest Hoffman

June 21, 2004

Contents

List of Tables

1 Introduction

CLM3.0 is radically different from either CLM2.1 or CLM2.0, particularly from a software engineering perspective. The major difference is that CLM3.0 is now both scalar cache friendly as well as vector friendly and can perform well on both types of architectures. The subgrid hierarchy introduced in CLM2.1 has been maintained in CLM3.0. However, its implementation within the code has been completely modified in order to permit the code to behave acceptably on vector architectures such as the Earth Simulator or the Cray X1. The CLM3.0 code is in fact easier to read as well as being more accessible to the introduction of new parameterizations than either CLM2.1 or CLM2.0.

Several other software-related changes have also accompanied the new CLM3.0 data structures. A brief discussion of the new data structures as well as the resulting new decomposition algorithm is given in section 5. In addition, a completely new decomposition algorithm has been implemented for multi-tasked and/or multi-threaded model runs that result in a significant improvement in both the load balancing as well as scaling nature of the model. New interfaces have been implemented for the specification of history file fields and initial dataset fields. Furthermore, history output for RTM river output is now on the RTM grid rather than being interpolated to the model grid. We refer the reader to the CLM3.0 Developer's Guide for more comprehensive details of the coding implementations.

CLM3.0 contains several improvements to biogeophysical parameterizations to correct deficiencies in the coupled model climate using CLM2.1. In CLM2.1, the 2-m temperature frequently dropped below the atmospheric potential temperature during daytime heating in certain regions. Stability terms were added to the formulation for 2-m air temperature to correct this. In CLM2.1, there is a discontinuity in the equation that relates the bulk density of newly fallen snow to atmospheric temperature. The equation was modified to correct this problem. Aerodynamic resistance for heat/moisture transfer from ground to canopy does not vary with the density of the canopy in CLM2.1. This leads to high surface soil temperatures in regions with sparse canopies. A new formulation was implemented in CLM3.0 that provides for variable aerodynamic resistance with canopy density. The vertical distribution of lake layers was modified to allow for more accurate computation of ground heat flux. A fix was implemented for negative round-off level soil ice caused by sublimation. Competition between plant functional types (PFTs) for water, in which all PFTs share a single soil column, is the default mode of operation in CLM3.0. CLM2.1 accepts either rain or snow from the atmospheric model. If the precipitation is snow, a formulation based on atmospheric temperature determines the fraction of precipitation that is in liquid form. In CLM3.0, the atmospheric model (in cam and ccsm mode) delivers precipitation explicitly in liquid and/or solid form. In offline mode (uncoupled from an atmospheric model), the formulation based on atmospheric temperature is still used. A fix was implemented to correct roughness lengths for non-vegetated areas. A full scientific description of CLM3.0 is given in

```
Technical Description of the Community Land Model (CLM)
Oleson, K. W., Dai Y., Bonan G., Bosilovich M., Dickinson R., Dirmeyer P.,
Hoffman F., Houser P., Levis S., Niu G.-Y., Thornton, P., Vertenstein M.,
Yang Z.-L., Zeng X.
NCAR Technical Note, NCAR/TN-461+STR, 2004.
```

In the CLM3.0 release we are also distributing the dynamic global vegetation model (CLM-DGVM) and a module for simulating volatile organic compound (VOC) emissions. The VOC module is documented in the CLM3.0 technical description. A full scientific description of the CLM-DGVM as well as an accompanying user's guide is given in

```
The Community Land Model's Dynamic Global Vegetation Model (CLM-DGVM):
Technical Description and User's Guide
Levis, S., Bonan, G. B., Vertenstein, M., Oleson, K. W.
NCAR Technical Note, NCAR/TN-459+IA, 2004.
```

Finally, CLM3.0 supports a new ability, via the input namelist specification (see section 4), to create and use a slightly different surface-data file. This new format surface dataset ("allpfts") is discussed at length in section 6.

2 Obtaining the Source Code and Datasets

The source code and datasets required to run the Community Land Model version 3.0 (CLM3.0) in offline mode (uncoupled from other components of the Community Climate System Model version 3 (CCSM3.0)) can be obtained via the web from:

http://www.cgd.ucar.edu/tss/clm.

The user should refer to the CAM3.0 User's Guide or the CCSM3.0 User's Guide for instructions on obtaining code and datasets to run CLM3.0 coupled to other CCSM3.0 components.

It is assumed that the user has access to the utilities **tar**, Free Software Foundation **gunzip** and **gmake (GNU gmake)**.

The CLM3.0 distribution consists of two tar files:

CLM3.0_code.tar.gz

and

CLM3.0_inputdata.tar.gz.

The file CLM3.0_code.tar.gz contains code, documentation, and scripts. This file must first be uncompressed with the **gunzip** utility and then "untarred" as follows:

gunzip -c CLM3.0_code.tar.gz — tar xvf -

The above command both uncompresses and "untars" the code into the following subdirectory hierarchy containing **clm3/** as the root:

Table 1: **Source Code Directory Structure**

Directory Name	Description
src/	Directory of FORTRAN and "C" source code
src/biogeophys/	Biogeophysics routines (e.g., surface fluxes)
src/biogeochem/	Ecosystem and biogeochemistry routines (e.g., DGVM and VOCs)
src/csm_share/	Code shared by all the geophysical model components of the Community Climate System Model (CCSM). Contains code for CCSM message passing, orbital calculations, and system utilities.
src/main/	Control (driver) routines
src/mksrfdata/	Routines for generating surface datasets
src/riverroute/	River routing (RTM) routines
src/utils/	Independent utility routines
src/utils/esmf/	Earth System Modeling Framework utilities
src/utils/timing/	General purpose timing library
bld/	Directory of build, test and run scripts
bld/offline/	Script to build and execute the model on various platforms
tools/	Directory of tools for input dataset manipulation (these tools are used independent of running the model)
tools/convert_ascii/	Routines for converting user-generated ascii surface dataset files to netCDF format suitable for use by the model
tools/interpinic	Tool for creating new initial CLM3.0 datasets from existing CLM3.0 datasets at another model resolution and/or landmask
tools/newcprnc/	Tool for comparing model netCDF history files

The file CLM3.0_inputdata.tar.gz contains surface and offline atmospheric forcing datasets. This file must first be uncompressed with the **gunzip** utility and then "untarred" as follows:

gunzip -c CLM3.0_inputdata.tar.gz — tar xvf -

The above command results in a directory hierarchy containing **inputdata/lnd/clm2/** as the root. This directory hierarchy is outlined below.

Table 2: **Input Data Directory Structure**

Directory Name	Synopsis
NCEPDATA/	One year's worth of atmospheric forcing variables in monthly netCDF format suitable for running the model in offline mode (uncoupled from the atmospheric model)
pftdata/	Plant functional type (PFT) physiological constants dataset (ascii format)
rawdata/	"Raw" (highest provided resolution) datasets (netCDF format) (used by CLM3.0 to generate surface datasets at model resolution)
rtmdata/	River direction map for RTM in ascii format
srfdata/	Directory containing netCDF CLM3.0 surface and "fgrid" datasets The surface datasets include new-format ("allpfts") and old-format datasets for running CLM3.0 in offline mode NOTE that surface datasets used in any CAM or CCSM run can also be used to run CLM3.0 in offline mode For "fgrid" datasets see section 4.2 and Example 6
output/	Model netcdf history files provided for the user to validate their port of CLM3.0 This data was generated using the file, jobscript.csh (see section 3.1) which was run on the NCAR IBM SP for 1 year using the data in directory NCEPDATA/.

Note that we are currently providing eight surface datasets for running offline CLM3.0. These datasets are located in **inputdata/lnd/clm2/srfdata**. A brief synopsis of each of these datasets is given below. It is important to keep in mind that any surface dataset used for a CAM or CCSM run can also be used for running offline CLM. We are providing these datasets as a way for the user to quickly begin using offline CLM3.0.

- **clms_64x128_USGS_c030605.nc** :
  CAM old-format T42 surface dataset
- **clms_64x128_allpfts_c040426.nc**:
  CAM new-format T42 surface dataset
- **clms_128x256_c031031.nc**:
  CAM old-format T85 surface dataset
- **clms_128x256_allpfts_c040426.nc**:
  CAM new-format T85 surface dataset
- **surface-data.128x064_atm.gx1v3_ocn.080101.nc**:
  CCSM old-format T42 surface dataset
- **surface-data.128x064_atm.gx1v3_ocn.allpfts_c040426.nc**:
  CCSM new-format T42 surface dataset
- **surface-data.256x128_atm.gx1v3_ocn.070903.nc**:
  CCSM old-format T85 surface dataset
- **surface-data.256x128_atm.gx1v3_ocn.allpfts_c040426.nc**:
  CCSM new-format T85 surface dataset

+

3 Creating and Running the Executable

The CLM3.0 model can be built to run in one of three modes. It can run as a stand alone executable where atmospheric forcing data is periodically read in (e.g., using the data in **NCEPDATA**). This will be referred to as offline mode. It can also be run as part of the Community Atmosphere Model (CAM) where communication between the atmospheric and land models occurs via subroutine calls. This will be referred to as cam mode. Finally, it can be run as a component in a system of geophysical models (CCSM). In this mode, the atmosphere, land, ocean and sea-ice models are run as separate executables that communicate with each other via the CCSM flux coupler. This will be referred to as ccsm mode.

CLM3.0 may be run serially (i.e., on a single processor), in parallel using the Message Passing Interface (MPI) for distributed memory tasks, in parallel using the Open Multi-Processing (OpenMP) directives for shared memory tasks, or finally in parallel using both MPI and OpenMP (hybrid parallelism). As an example, the IBM SP consists of distributed memory nodes interconnected by a high performance network connection, and each node contains multiple shared memory processors. When run on the IBM SP, CLM3.0 uses OpenMP directives for parallelism on processors within a shared memory node and MPI routines for parallelism across distributed memory nodes to take full advantage of the capabilities of the hardware. The supported architectures and associated compilers for running CLM3.0 in offline mode are shown in Table 3. Note that when DGVM is defined (see Table 6), bit-for-bit restarts are not supported on the CRAY X1 when processor configuration is changed mid-run.

Table 3: **Offline Mode Supported Architectures**

Hardware	**Architecture**	**OS**	**Compiler**
IBM SP	RS600	AIX	IBM XL
SGI	MIPS	IRIX64	MIPS
Intel	Intel X86	Linux	pgf90, lf95
CRAY X1	Cray X1	Unicos/mp	ftn
NEC SX6	NEC SX6	Super-UX	f90
NEC SX6	NEC SX6	Super-UX	sxf90 (cross compiler)

The method of building and running CLM3.0 depends on both the target architecture and whether the model will be run serially, in pure MPI mode (distributed memory), in pure OpenMP mode (shared memory) or in hybrid mode. A general discussion of the various aspects of building and running CLM3.0 follows.

Several cautionary notes have to be made concerning Linux systems. For Linux operating systems, CLM3.0 is supported for both lf95 and pgf90 compilers. We have noted problems in running CLM3.0 in hybrid mode using pgf90 compilers. Consequently, OpenMP threading has been turned off by default for Linux systems. (The user may turn it back on by setting **SMP** to TRUE in jobscript.csh (see below). We have also noted problems in bounds checking with pgf90 when running in debug mode (**DEBUG** = TRUE). As a result, the bounds checking in the Makefile has been temporarily disabled.

3.1 offline mode: using jobscript.csh

In order to build and run CLM3.0 in offline mode, a sample script, jobscript.csh, and a corresponding Makefile are provided in the **bld/offline** directory. In addition, two perl scripts **mkDepends** (used to generate dependencies in a form suitable for inclusion into a Makefile), and **mkSrcfiles** (used to make a list of files containing source code) are also included. The user need not worry about these perl scripts.

The script, jobscript.csh, creates a model executable at T42 model resolution with RTM (river transport model) activated and CLM-DGVM and VOCs deactivated, determines the necessary input datasets, constructs the input model namelist and runs the model for one day. Users must edit this script appropriately in order to build and run the executable for their particular requirements and in their particular environment.

This script is provided only as an example to aid the novice user in getting CLM3.0 up and running as quickly as possible.

The script can be run with minimal user modification, assuming the user resets several environment variables at the top of the script. In particular, the user must set the following:

1. Set batch queue directives for the required machine if will be running in batch mode as opposed to interactive mode. Batch mode means submitting jobscript.csh through a queueing system. Interactive mode means typing jobscript.csh at the prompt on your screen.
2. Set model resolution: number of model longitudes (LSMLON) and latitudes (LSMLAT).
3. Set **ROOTDIR** to point to the full disk pathname of the root directory containing the untarred source code.
4. Set **MODEL_EXEDIR** to point to the directory where the user wants the executable to be built and run.
5. Set **CSMDATA** to point to the full disk pathname of the root directory containing the untarred input dataset subdirectories.
6. Determine if MPI and/or OpenMP will be used, and if so determine the number of MPI tasks and/or the number of OpenMP threads (note that for linux systems the number of mpi tasks is determined from the batch directives if running in batch mode).
7. Set required library paths and include paths.
8. Modify the namelist.
9. Determine which model cpp tokens will be enabled.

The script can be divided into five functional sections: 1) specification of script environment variables; 2) creation of the model input namelist; 3) creation of two header files (misc.h and preproc.h) and a directory search path file (Filepath) needed to build the model executable; 4) creation of the model executable; and 5) execution of the model. Jobscript.csh is set up so that the user will normally only have to modify sections 1) and 2) in order to obtain a model executable and associated namelist. Each of these functional sections is discussed in what follows.

3.1.1 Specification of script environment variables

Table 4 lists the user modifiable script environment variables. Some of these variables are used by the Makefile to build the model executable. Although the script provides tentative settings for all these variables, the provided values will generally have to be modified by the user.

Table 4: **User Modifiable Script Variables**

Script Variable	Description
LSMLON	Number of model grid longitudes.
LSMLAT	Number of model grid latitudes.
ROOTDIR	Full pathname for the root source code directory.
MODEL_EXEDIR	Full pathname for the directory where the model executable will reside. Object files will be built in the directory $MODEL_EXEDIR/obj.
CSMDATA	Full pathname of root input datasets directory.
SMP	Flag that turns on OpenMP (valid values are TRUE or FALSE) If SMP is TRUE, OpenMp is enabled and NTHREADS is the number of OpenMP threads. If SMP is FALSE, OpenMp is disabled and NTHREADS is ignored. Currently, SMP will be set to FALSE for Linux platforms.
NTHREADS	Number of OpenMP multitasking threads. NTHREADS should not exceed the number of physical CPUs (ie, processors) on a shared memory machine and should not exceed the number of CPUs in a node on a distributed memory machine. This setting is ignored if SMP is set to FALSE.
SPMD	Flag that turns on MPI (valid values are TRUE or FALSE). If SPMD is TRUE, MPI is enabled and NTASKS is the number of MPI tasks. If SPMD is FALSE, MPI is disabled and NTASKS is ignored.
NTASKS	Number of MPI tasks. Setting is ignored if SPMD is set to FALSE. NTASKS should not exceed the number of physical CPUs (ie, processors) on a machine.
LIB_NETCDF	Full pathname of directory containing the netCDF library. The setting depends on user's target machine.
INC_NETCDF	Full pathname of directory containing netCDF include files. The setting depends on user's target machine.
LIB_MPI	Full pathname of directory containing the MPI library. The setting depends on user's target machine. Only needed if SPMD is set to TRUE. Not needed on IBM SP, Cray X1, NEC SX6 machines, where the MPI library and include files are obtained directly from the compiler command.
INC_MPI	Full pathname for directory containing the MPI include files. The setting depends on user's target machine. Only needed if SPMD is set to TRUE. Not needed on IBM SP, Cray X1, NEC SX6 machines.
linux_fort_compiler	Determines which linux fortran compiler is used. Valid values are pgf90 or lf95. Only used on linux platforms.
linux_machine	Name of linux machine. Currently NCAR-CGD linux machine "anchorage" is supported in the scripts. The user will need to tailor this for their own platform. Only used on linux platforms.
linux_mpirun_cmnd	The full pathname of mpirun to use if $SPMD is set to TRUE. Multiple fortran compilers often exist on a single linux system. Each compiler can have a unique path where the mpich mpirun binary is installed for that compiler. Only used on linux platforms.
linux_batch	Determines if run will be in batch or interactive mode. Only used on linux platforms.
DEBUG	Flag that turns on debugging in Makefile. Valid values are TRUE or FALSE.

It is assumed that the user has set up the environment for their particular compiler choice (e.g. **LD_LIBRARY_PATH** is set correctly in the user's environment).

3.1.2 Setting the Namelist

Before building and running the model with jobscript.csh, the user must specify model namelist variables via the CLM3 namelist, **clmexp**, found in jobscript.csh. Without modification, jobscript.csh generates a default namelist file, lnd.stdin, which results in a one day model run using the provided datasets. Namelist variables can be divided into several categories: run definitions, datasets, history and restart file settings and land model physics settings. A full discussion of model namelist variables is given in section 4.

3.1.3 Creation of header and directory search path files

The user will generally not need to modify the section of jobscript.csh that creates the header and directory search path files. The script creates three files in the directory **$MODEL_EXEDIR/obj**: the header files misc.h and preproc.h and the directory search path file, Filepath. To modify these files, the user should edit the file contents from within the script rather than attempt to edit the files directly, since the script will overwrite the files upon execution. The use of these files by **gnumake** is discussed below. Each of these files is summarized below.

The file, **misc.h**, contains resolution- and model-independent C-language pre-processor (cpp) tokens.

Table 5: **misc.h CPP tokens**

misc.h cpp token	**Description**
SPMD	If defined, enables distributed memory, SPMD (single program multiple data), implementation. Automatically defined by script if environment variable **SPMD** is set to TRUE. See section 3.1.1.
PERGRO	If defined, enables modifications that test reasonable perturbation error growth. Not supported at this time.

The file **preproc.h** contains resolution-dependent and model-dependent C-language cpp tokens.

Table 6: **preproc.h CPP tokens**

preproc.h cpp token	**Description**
OFFLINE	If defined, offline mode is invoked
COUP_CSM	If defined, ccsm mode is invoked
COUP_CAM	If defined, cam mode is invoked
SCAM	If defined, cam single column mode is invoked
	This is only applicable if COUP_CAM is ALSO defined
LSMLON	Number of model longitudes
LSMLAT	Number of model latitudes
RTM	If defined, RTM river routing is invoked
VOC	If defined, voc emission is computed
DGVM	If defined, dynamic vegetation model is activated
NOCOMPETE	If defined, competition for water is turned off (each pft has its own column) This mode is no longer officially supported

C-preprocessor directives of the form #include, #if defined, etc., are used in the model source code to enhance code portability and allow for the implementation of distinct blocks of functionality (such as incorporation of different modes) within a single file. Header files, such as misc.h and preproc.h, are included with #include statements within the source code. When **gnumake** is invoked, the C preprocessor includes or excludes blocks of code depending on which cpp tokens have been defined. C-preprocessor directives are also used to perform textual substitution for resolution-specific parameters in the code. The format of these tokens follows standard cpp protocol in that they are all uppercase versions of the Fortran variables, which they define. Thus, a code statement like

parameter(lsmlon = LSMLON); parameter(lsmlat = LSMLAT)

will result in the following processed line (for T42 model resolution):

parameter(lsmlon=128) ; parameter(lsmlat=64)

where LSMLON and LSMLAT are set in preproc.h via the jobscript.

Filepath contains a list of directories used by Makefile to resolve the location of source files and to determine dependencies. The search begins in the current directory and proceeds to each directory appearing in Filepath, in the order in which they are specified. All files appearing in these directories will be used unless duplicate files are found. For the case of duplicate files, the first file found will be used by gnumake to create the object file. If user-modified code is introduced, Filepath should contain, as the first entry, the directory containing such modified code.

Users can add new search directories by editing jobscript.csh under "build Filepath". The default Filepath directory hierarchy for CLM3.0 is as follows:

Table 7: **Filepath**

Source Directories	**Functionality**
$MODEL_SRCDIR/main	control routines (history, restart, etc)
$MODEL_SRCDIR/biogeophys	biogeophysics routines
$MODEL_SRCDIR/biogeochem	ecosystem and biogeochemistry routines
$MODEL_SRCDIR/riverroute	river routing routines
$MODEL_SRCDIR/csm_share/shr	code shared by all CCSM geophysical model components
$MODEL_SRCDIR/utils/timing	timing routines
$MODEL_SRCDIR/mksrfdata	generation of surface dataset routines

3.1.4 Building the model

The user will generally not need to modify the section of jobscript.csh that builds the model executable. Jobscript.csh invokes **gnumake** to generate the model executable. The file, Makefile, located in the **bld/offline** directory, contains commands used by **gnumake** on each of the supported target architectures. Successful invocation of **gnumake** results in an executable, "clm", along with a log file, "compile_log.clm", documenting the build procedure. Any problems encountered during the build procedure will be documented in this log file. A parallel **gnumake** is achieved in the script by invoking **gnumake** with the -j option, which specifies the number of job commands to run in parallel.

3.1.5 Running the executable

The user will generally not need to modify the section of jobscript.csh that runs the model executable. Jobscript.csh will execute the commands required to run the model under the supported target architectures. Most MPI implementations provide a startup script which accepts the MPI executable as a command line argument. Additional command line arguments allow the user to specify details such as the various machine architectures or number of processes to use for the run. Once MPI has created the specified number of processes, model execution will begin. The collection of active tasks will then compute locally and exchange messages with each other to integrate the model.

Upon successful completion of the model run, several files will be generated in **MODEL_EXEDIR**. These include history, restart, and initialization files (see section 4.3) as well as log files documenting the model execution. These log files will have names of clm.log.YYMMDD-HHMMSS, where YY is the last two digits of the current model year, MM is the month, DD is the day of the month, HH is the hour, MM is the minutes, and SS is the seconds of the start of the model run. Timing files, (e.g. "timing.0"), containing model performance statistics are also generated in the executable directory.

3.2 cam mode

When running the model as part of the CAM executable, CAM build and run scripts must be utilized. The user should refer to User's Guide (to the NCAR Community Atmosphere Model 3.0 (CAM 3.0) at

www.ccsm.ucar.edu/models/ccsm3.0/cam/

for specific details on building and running the CAM executable. We will only discuss some essential points of the CAM build and run scripts here.

The header files, preproc.h and misc.h, as well as the directory search path file, Filepath, are needed for the CAM build procedure in an analogous manner to the CLM3.0 build procedure. The user should keep in mind that the CLM3.0 directory hierarchy **MUST appear after** the CAM directory hierarchy in Filepath. CLM3.0 contains several files that have the same name as the corresponding CAM files (e.g. time_manager.F90). When running in CAM mode, the corresponding CAM file must be used. The CAM build and run scripts ensure this occurs.

The CLM3.0 namelist, **clmexp**, must also be specified. By default, RTM river routing is not enabled in cam mode (i.e. the cpp variable, RTM, is not defined). Furthermore, CLM3.0 does not permit the user to independently set several namelist variables (in particular, those dealing with history file logic and run control logic) when running in cam mode. CLM3.0 will override any user input for these variables with the corresponding values used by the CAM model. This is discussed in more detail in section 4.6.

3.3 ccsm mode

CCSM3.0 will contain CLM3.0. In ccsm mode RTM is defined by default, because this provides the fresh water flux from the land to the ocean model. We refer the reader to the CCSM3.0 User's Guide for further details on running in ccsm mode.

4 Namelist Parameters

CLM3.0 namelist inputs are presented in sections 4.1 - 4.7 below. In what follows, "mode" has values of "offline", "ccsm", "cam" or "all", corresponding to offline mode, ccsm mode, cam mode, or all the modes. If a namelist variable setting is listed as **required**, the value must be set in the namelist in order for the model to execute successfully. If a setting is specified as **required** and the mode is only given as offline, then that variable must only be specified when running in offline mode. For namelist variable settings not listed as **required**, the code will provide default settings at initialization. In the following variable descriptions, we refer to examples presented in section 9. See Example 1 for a description of the minimum required namelist for a successful run.

4.1 Run definitions

The following list specifies namelist variables associated with the definition of run case names, run types (restart, initial or branch), model time step, and initial run date.

An initial run starts the model from either initial conditions that are set internally in the code (referred to as arbitrary initial conditions) or from an initial conditions dataset (see namelist variable **FINIDAT**) that enables the model to start from a spun-up state.

A restart run is an exact continuation of a previous simulation from its point of termination. Output from a restart run should be bit-for-bit the same as if the previous simulation had not stopped. Run control variables set in the namelist must be the same as in the run that is being restarted.

A branch run is a new case that uses a restart dataset from a previous simulation to begin the integration. For a branch run, the length of the history interval and the output history fields do not have to be the same as in the control simulation. For example, the branching option can be used to output selected fields more frequently than was the case in the original run or to add new auxiliary history files to the model run.

name:	**CASEID**
description:	Case name (short identifier for run) (see ex. 1,2,3).
type:	char*256
mode:	offline, ccsm (obtained from atm in cam mode)
default:	**required** (must be changed for branch run unless **BRNCH_RETAIN_CASENAME** is set to .true.)

name:	**CTITLE**
description:	Case title for use within history files (long identifier).
type:	char*256
mode:	offline, ccsm (obtained from atm in cam mode)
default:	blank

name:	**NSREST**
description:	Run type (0 for initial run, 1 for restart, 3 for branch) (see ex. 1,2,3).
type:	integer
mode:	offline, ccsm (obtained from atm in cam mode)
default:	**required**

name:	**DTIME**
description:	Model time step (seconds) (see ex. 1).

type: integer
mode: offline, must agree with CAM model in ccsm mode, obtained from CAM model directly in cam mode
default: **required** (suggested range: 1200-3600 s)

name: **STOP_YMD**
description: Stop date (YYYYMMDD)
type: integer
mode: offline (obtained from CAM in cam mode and CCSM coupler in ccsm mode)
default: **required** (if **NESTEP** or **NELAPSE** not set)

name: **STOP_TOD**
description: Stop time of day (seconds). Ignored if **STOP_YMD** not defined
type: integer
mode: offline (obtained from CAM in cam mode and CCSM coupler in ccsm mode)
default: 0

name: **NESTEP**
description: Ending run time in model time steps (positive) or days (negative) (see ex. 1). Value is ignored if **STOP_YMD** is specified.
type: integer
mode: offline (ending time obtained from CAM in cam mode and CCSM coupler in ccsm mode)
default: **required** (if neither **NELAPSE** nor **STOP_YMD** are set)

name: **NELAPSE**
description: Elapsed run time in model time steps (positive) or days (negative) (see ex. 2). Value is ignored if either **STOP_YMD** or **NELAPSE** are specified.
type: integer
mode: offline (ending time is obtained from CAM in cam mode and CCSM coupler in ccsm mode)
default: **required** (if neither **NESTEP** nor **STOP_YMD** are set)

name: **START_YMD**
description: Start date of run (yyyymmdd format) (see ex. 1).
type: integer
mode: offline, ccsm (obtained from CAM model in cam mode)
default: **required**

name: **START_TOD**
description: Start time of day of run (seconds).
type: integer
mode: offline, ccsm (obtained from CAM in cam mode)
default: 0

name: **REF_YMD**
description: Reference date for time coordinate (yyyymmdd format).
type: integer
mode: offline, ccsm (obtain from CAM in cam mode)
default: no default value but if not set, then the model reference date will automatically be set to **START_YMD**

name: **REF_TOD**
description: Reference time of day for time coordinate (secs).
type: integer
mode: offline, ccsm (obtain from CAM in cam mode)
default: 0

name: **CALENDAR**
description: Calendar to use in date calculations ("no-leap" or "gregorian").
type: char*32
mode: offline, ccsm (obtained from CAM in cam mode)
default: no-leap

name: **CLUMP_PPROC**
description: Number of "clumps" per process (see section 5).
type: integer
mode: offline, ccsm (obtained from CAM in cam mode)
default: 1 if OpenMP is disabled
number of OpenMP threads (specified in the environment) if OpenMP is enabled.

name: **BRNCH_RETAIN_CASENAME**
description: Flag to retain case name on a branch run. If true, then allow case name to remain the same for branch run. In cam mode, this flag must be set for both the cam and clm namelists.
type: logical
mode: offline, cam, ccsm
default: .false.

4.2 Specification of model input datasets

The following list specifies namelist variables associated with model input datasets.

name: **FSURDAT**
description: Full pathname of surface dataset (see ex. 1,5,6).
type: char*256
mode: all
default: blank (details at the end of this section)

notes: surface datasets provided with the distribution are in $CSMDATA/srfdata, raw datasets to generate surface datasets are in $CSMDATA/rawdata

name: **FINIDAT**
description: Full pathname of initial conditions dataset (see ex. 1,2,3).
type: char*256
mode: all
default: blank (details at the end of this section)
notes: no initial datasets are provided, the user will need to generate these

name: **FPFTCON**
description: Full pathname of plant functional type (PFT) physiological constants dataset (see ex. 1).
type: char*256
mode: all
default: **required**
notes: dataset provided is $CSMDATA/pftdata/pft-physiology

name: **FRIVINP_RTM**
description: full pathname of RTM input dataset (see ex. 4).
type: char*256
mode: offline, ccsm
default: **required** if cpp token **RTM** is defined in preproc.h
notes: dataset provided is $CSMDATA/rtmdata/rdirc.05

name: **NREVSN**
description: Full pathname of restart file name (only for branch runs) (see ex. 3).
type: char*256
mode: all
default: **required** (only if branch run, NSREST=3)

name: **MKSRF_ALL_PFTS**
description: Flag to turn on new surface dataset format. If true, new surface dataset format (see section 6) is used.
type: logical
mode: all
default: .false.

name: **MKSRF_FVEGTYP**
description: Full pathname of raw plant functional type dataset (see ex. 5).
type: char*256
mode: all
default: **required** (if FSURDAT is blank)
notes: dataset provided is $CSMDATA/rawdata/mksrf_pft.nc

name: **MKSRF_FSOITEX**
description: full pathname of raw soil texture dataset (see ex. 5).
type: char*256
mode: all
default: **required** (if FSURDAT is blank)
notes: dataset provided is $CSMDATA/rawdata/mksrf_soitex.10level.nc

name: **MKSRF_FSOICOL**
description: Full pathname of raw soil color dataset (see ex. 5).
type: char*256
mode: all
default: **required** (if FSURDAT is blank)
notes: dataset provided is $CSMDATA/rawdata/mksrf_soicol_clm2.nc

name: **MKSRF_FLANWAT**
description: Full pathname of raw inland water dataset (see ex. 5).
type: char*256
mode: all
default: **required** (if FSURDAT is blank)
notes: dataset provided is $CSMDATA/rawdata/mksrf_lanwat.nc

name: **MKSRF_FURBAN**
description: full pathname of urban dataset (see ex. 5).
type: char*256
mode: all
default: **required** (if FSURDAT is blank)
notes: dataset provided is $CSMDATA/rawdata/mksrf_urban.nc

name: **MKSRF_FGLACIER**
description: Full pathname of glacier dataset (see ex. 5).
type: char*256
mode: all
default: **required** (if FSURDAT is blank)
notes: dataset provided is $CSMDATA/rawdata/mksrf_glacier.nc

name: **MKSRF_FLAI**
description: full pathname of leaf and stem area index, canopy top and bottom height dataset (see ex. 5).
type: char*256
mode: all
default: **required** (if FSURDAT is blank)
notes: dataset provided is $CSMDATA/rawdata/mksrf_lai.nc

name: **MKSRF_OFFLINE_FNAVYORO**
description: 20 min navy orography dataset used to generate land mask (see ex. 5).

type: char*256
mode: offline
default: **required** (if MKSRF_OFFLINE_FGRID not set and FSURDAT is blank)
notes: dataset provided is $CSMDATA/rawdata/mksrf_navyoro_20min.nc

name: **MKSRF_OFFLINE_FGRID**
description: Dataset specifying land grid and mask at desired resolution (see ex. 6).
type: char*256
mode: offline
default: blank, **required** (if MKSRF_OFFLINE_FNAVYORO not set and FSURDAT is blank)
notes: datasets provided in $CSMDATA/srfdata

name: **MKSRF_OFFLINE_EDGEN**
description: Northern edge of land grid (degrees north) (see ex. 5).
type: real
mode: offline
default: 90.

name: **MKSRF_OFFLINE_EDGEE**
description: Eastern edge of land grid (degrees east) (see ex. 5).
type: real
mode: offline
default: 180.

name: **MKSRF_OFFLINE_EDGES**
description: Southern edge of land grid (degrees north) (see ex. 5).
type: real
mode: offline
default: -90.

name: **MKSRF_OFFLINE_EDGEW**
description: Western edge of grid land (degrees east) (see ex. 5).
type: real
mode: offline
default: -180.

name: **OFFLINE_ATMDIR**
description: Directory containing atmospheric forcing datasets (see ex. 1).
type: char*256
mode: offline
default: **required**
notes: datasets provided are in directory $CSMDATA/NCEPDATA

Additional details: **FSURDAT** specifies a surface dataset containing time-invariant land properties such as plant functional types and soil textures and time-variant properties such as leaf area index. If **FSURDAT** is set to the empty string, a new surface dataset is generated at run time for the specified model resolution. The creation of a new surface dataset requires the specification of the full pathname of the following raw datasets: **MKSRF_FVEGTYP**, **MKSRF_FSOITEX**, **MKSRF_FSOICOL**, **MKSRF_FLANWAT**, **MKSRF_FURBAN**, **MKSRF_FGLACIER**, **MKSRF_FLAI**. These datasets are only used for the generation of a model surface dataset.

In addition to raw datasets, a land/ocean mask is also required for the creation of a new surface dataset. If the model is run in ccsm or cam mode, this mask is obtained from either the ccsm flux coupler or from the cam atmosphere model at startup. In offline mode, however, the land/ocean mask can either be calculated from a high resolution orography dataset by setting the namelist variable **MKSRF_OFFLINE_FNAVYORO** or can be read in from an input dataset already at the target resolution via the setting of the namelist variable **MKSRF_OFFLINE_FGRID**.

Finally, if a surface dataset is to be created at run time, the user must specify if a new-format or old-format dataset will be generated. By default, old-format datasets are created. By setting the namelist variable **MKSRF_ALL_PFTS** to .true., a new-format surface dataset will be created (see section 6).

Subroutines involved in creating a surface dataset at run time reside in the source code directory **mksrfdata/**. In most cases the creation of a surface dataset involves a straightforward interpolation from the raw dataset resolution to the desired model resolution. For soil texture, however, averaging would create new soil types. Consequently, the model determines the dominant soil texture profile per gridcell from the raw resolution to the desired resolution.

Once the surface dataset is created, the user should use **FSURDAT** to point to that dataset, in order to avoid creating the same dataset multiple times.

The input file specified by the namelist variable **FINIDAT** contains values for the time-dependent variables needed to initialize the model from a spun-up state. If **FINIDAT** is set to the empty string, the model is internally initialized to non spun-up values. The setting of the namelist variable, **HIST_CRTINIC** (described in section 4.3) can be used to generate initial CLM files during a model run. We also provide a new tool, **interpinic**, which provides users with the ability to use an already existing initial dataset at one model resolution to generate a new initial dataset at another model resolution. Section 7 provides more details of this tool.

When the cpp token **RTM** is defined, the RTM river routing scheme will be invoked in running the model. In this case, **FRIVINP_RTM** must be set to a river routing dataset.

In offline mode, time dependent atmospheric forcing data must be read in periodically. The directory containing these files is given by **OFFLINE_ATMDIR**. This variable is ignored in cam and ccsm mode.

4.3 Specification of history and restart files

The following describes namelist variables associated with history, restart, and initialization files. In what follows, max_tapes denotes the maximum allowable number of different types of history files (tapes) that the model can produce (currently set to 6) and max_flds denotes the maximum number of history fields that may appear on any given history tape (currently set to 1000).

name:	**HIST_CRTINIC**
description:	Frequency with which initial datasets will be generated. Valid values are 'YEARLY','MONTHLY','DAILY','6-HOURLY' or'NONE'.
type:	char*8
mode:	offline, ccsm (obtained from CAM in cam mode)
default:	'YEARLY'

name:	**HIST_NHTFRQ**

description: History tape interval(s)
(+ for model time steps, - for hours, 0 for monthly ave) (see ex. 1,3).
type: integer array (up to **max_tapes** values separated by commas)
mode: offline, ccsm (obtained from CAM in cam mode)
default: 0

name: **HIST_MFILT**
description: Number of time samples per history tape(s) (see ex. 3,4).
type: integer array (up to **max_tapes** values separated by commas)
mode: offline, ccsm (obtained from CAM in cam mode)
default: 1 when HIST_NHTFRQ=0, else 30

name: **HIST_NDENS**
description: Output tape precision(s). Valid values are 1 (double precision) or 2 (single precision).
type: integer array (up to **max_tapes** values separated by commas)
mode: all
default: 2

name: **HIST_DOV2XY**
description: Per tape spatial averaging flag. If set to true, produces grid-average history fields on output tape. If set to false, one-dimensional fields are produced (see ex. 4).
type: logical array (up to **max_tapes** values separated by commas)
mode: all
default: .true.

name: **HIST_AVGFLAG_PERTAPE**
description: Per tape time averaging flag. Valid values are 'A' (average over history period), 'I' (instantaneous), 'X' (maximum over history period) or 'M' (minimum over history period).
type: char*1 array (up to **max_tapes** values separated by commas)
mode: all
default: blank (in this case a hardwired flag per variable is used, found in histFldsMod.F90)

name: **HIST_TYPE1D_PERTAPE**
description: Per tape one dimensional output type. Only used if one dimensional output is selected for the given tape (via the setting of HIST_DOV2XY). Valid values are 'GRID', 'LAND', 'COLS', 'PFTS'. For example, if one dimensional output is selected for tape 3 and HIST_TYPE1D_PERTAPE is set to 'COLS', then all the fields will have 1d column output. If the specified one dimensional output type is not defined for a given field, output values will be set to 1.e36 for that field.
type: char*4 array (up to **max_tapes** values separated by commas)
mode: all
default: blank (in this case a hardwired type per variable is used)

name: **HIST_EMPTY_HTAPES**

description: If set to true, all the history tapes are empty by default. Only variables explicitly listed by the user will be output.

type: logical

mode: all

default: .false.

name: **HIST_FINCL1, ..., HIST_FINCL6**

description: List of fields to include on the respective history tape. See tables 9-18 for the list of default fields on the primary history tape. Namelist specification can take one of two forms. The user may simply specify the name of the field to be included on the history tape (in which case the default time averaging for that field will be used). For example, HIST_FINCL2='TV', will add the field TV to the second history tape with whatever default time averaging was specified for TV. Alternatively, the user may specify the field name, followed by a ":" followed by the time averaging flag desired (valid flags are 'I' for instantaneous, 'A' for average, 'M' for minimum, and 'X' for maximum). For example, HIST_FINCL2='TV:I" will add the field TV with instantaneous output to the second history tape (see ex. 3,4).

type: char*34 array (up to **max_flds** values separated by commas)

mode: all

default: blank

name: **HIST_FEXCL1, ..., HIST_FEXCL6**

description: List of fields to exclude from the respective history tape. The field name must appear in the Master Field List of the default history tape (the primary tape). See tables 9-18 for more details.

type: char*32 array (up to **max_flds** values separated by commas)

mode: all

default: blank

name: **MSS_IRT**

description: Mass Store retention period (days) for output datasets (see ex. 4). User must have Mass Store access. For example NCAR supercomputers recognize Mass Store commands.

type: integer

mode: offline, ccsm (obtained from CAM in cam mode)

default: 0 (i.e., history files will be written to local disk, not the NCAR Mass Store)

name: **MSS_WPASS**

description: Mass Store write password for output datasets.

type: char*8

mode: all

default: blank

name: **RPNTPATH**

description:	Full unix pathname of the local restart pointer file.
type:	char*256
mode:	all
default:	File lnd.CASEID.rpointer placed in user's home directory

name:	**ARCHIVE_DIR**
description:	Mass Store directory to archive output files
type:	char*256
mode:	all
default:	/USERNAME/CSM/CASEID

Additional details: The model writes its own history, restart and initial files. History files are in netCDF file format and contain model data values written at user specified frequencies during a model run. Each field has a default time averaging flag determining how that field will be accumulated in time over a given history interval. The choices are to record averaged, instantaneous, maximum, or minimum values. The user may overwrite this default setting via the namelist variable **HIST_FINCLt** where t can equal 1 to 6. If the user wishes to see a field written at more than one output frequency (e.g. daily and hourly), additional history files must be declared containing that field. By default, CLM3.0 produces a monthly averaged primary history file and allows the user to define up to five auxiliary history files. All files contain grid averaged data unless the namelist variable **HIST_DOV2XY** is set to false for a given file. Primary history files contain the string 'h0', whereas auxiliary history files contain the string 'h1', 'h2', 'h3','h4' and 'h5'.

The model will also create netCDF datasets containing one dimensional instantaneous values of initial data fields at the frequency defined by namelist variable **HIST_CRTINIC**. These datasets can be utilized as "spun-up" initial conditions.

Restart files are in binary format and can be used only for restart or branch runs from previous model simulations. Whenever a restart file is written, a corresponding local disk restart pointer file is overwritten. The restart pointer file contains the name of the latest model restart file. By default, the restart pointer file is placed in the user's home directory under the name, lnd.CASEID.rpointer. The user may modify the full pathname of the restart pointer file via the setting of the namelist variable **RPNTPATH**.

The following table specifies the naming convention used for output files. In this table the string yyyy refers to the model year, mm refers to the model month, dd is the model day and sssss corresponds to seconds into the model day. Initial and restart files always contain one time slice of instantaneous data. However, non-monthly history files may contain multiple time slices of time-averaged data, so yyyy-mm-dd-sssss corresponds to the first timeslice of data on the file. **CASEID** is the case identifier set via the namelist input.

CASEID.clm2.r.yyyy-mm-dd-sssss restart files

CASEID.clm2.i.yyyy-mm-dd-sssss.nc initial files

CASEID.clm2.h[012345 .yyyy-mm.nc] monthly average history files

CASEID.clm2.h[012345 .yyyy-mm-dd-sssss.nc] non-monthly history files

History, restart and initialization files can be archived on the NCAR Mass Storage System (MSS) if the namelist variable **MSS_IRT** is set to a value greater than zero. History, restart and initial files are archived as follows (where **USERNAME** is the upper-case equivalent of the user's login name, i.e., the user's root directory on the MSS):

history files /USERNAME/csm/CASEID/lnd/hist

restart files /USERNAME/csm/CASEID/lnd/rest

initial files /USERNAME/csm/CASEID/lnd/init

4.4 Specification of input physics variables

name:	**IRAD**
description:	Frequency of solar radiation calculations (+ for model time steps, - for hours).
type:	integer
mode:	offline, must be consistent with CAM3.0 in ccsm mode, obtained from CAM in cam mode
default:	-1

name:	**CSM_DOFLXAVE**
description:	If set to true, flux averaging is performed over the duration set in **IRAD**.
type:	logical
mode:	ccsm (must agree with CAM3.0 setting in atm.setup.csh)
default:	.true. (.false. not supported)

name:	**WRTDIA**
description:	If true, global average 2-m temperature written to standard out (ascii log file of the run) (see ex. 4).
type:	logical
mode:	all
default:	.false.

name:	**PERTLIM**
description:	Perturbation limit (not supported at this time)
type:	real
mode:	all
default:	0

4.5 Specification of RTM River routing

name:	**RTM_NSTEPS**
description:	Frequency in number of time steps at which RTM is called.
type:	integer
mode:	all
default:	number of timesteps in 3 hours

4.6 Specification of cam mode namelist

When running in cam mode, certain CLM3.0 namelist variables cannot be set independently. In particular, any user specification for the namelist variables, **CASEID**, **CTITLE**, **IRAD**, **NSREST**, **HIST_CRTINIC**, **MSS_IRT**, **HIST_NHTFRQ**, and **HIST_MFILT** (the last two only for primary history files) will be overwritten by values obtained from CAM3.0 at startup. All other namelist settings may be set independently by the user.

The following table specifies the namelist variables that are overwritten with values obtained from cam and lists the associated CAM3.0 namelist variable and its default value.

Table 8: **Namelist Variables overwritten with CAM settings**

CLM Namelist	CAM namelist	CAM default
CASEID	CASEID	required
CTITLE	CTITLE	blank
NSREST	NSREST	0
IRAD	IRADSW	-1
HIST_CRTINIC	INITHIST	'MONTHLY'
HIST_NHTFRQ(1)	NHTFRQ(1)	0
HIST_MFILT(1)	MFILT(1)	1
MSS_IRT	MSS_IRT	365

4.7 Specification of ccsm mode namelist

When running in ccsm mode, the user must ensure that settings of the namelist variables, **IRAD**, **DTIME** and **CSM_DOFLXAVE**, have identical values to the corresponding CAM namelist variables **IRADSW**, **DTIME** and **FLXAVE** in the script cam.buildnml_prestage.csh (see CCSM3.0 User's Guide).

In ccsm mode the RTM input dataset must be specified in the namelist using variable **FRIVINP_RTM**, because RTM is defined by default.

In ccsm mode, CAM and CLM run on the same grid which depends on the specific ocean domain utilized. Consequently, a different surface dataset is required for each atm/ocn grid combination. Currently all supported ccsm-clm surface datasets will be released with the ccsm distribution. Not supported ones may be generated by the user by setting **FSURDAT** = ' '.

Additionally, a CLM3.0 spun-up initial dataset may be provided containing values for the time-dependent variables needed to initialize the model from a spun-up state by setting the namelist variable **FINIDAT**. This file **MUST** have the same atm/ocn resolution and landmask as the model run for which it will be used. The variables appearing in the **FINIDAT** file will be internally initialized to non spun-up values at run time if **FINIDAT** is not set.

Finally, the namelist variable **CSM_DOFLXAVE** is specific to ccsm mode only. If this variable is set to true (the default setting), flux averaging is performed over the time interval specified by the namelist variable, **IRAD** (**IRAD** must be greater than 1). The false setting is not supported.

5 CLM3.0 Data Structures

In what follows, we provide a brief summary of the CLM3.0 data structures. Understanding of these data structures is essential before the user attempts to modify code and/or add new history output fields to the model.

The subgrid hierarchy in CLM3.0 is composed of gridcells, landunits, columns and plant functional types (pfts). Each gridcell can have a different number of landunits, each landunit can have a different number of columns and each column can have multiple pfts. This results in efficient memory allocation, and allows for the implementation of many different types of subgrid representations.

The first subgrid level, the landunit, is intended to capture the broadest spatial patterns of subgrid heterogeneity. These broad patterns include physically distinct surface types (glaciers, lakes, wetlands, and vegetated areas). In terms of CLM3.0 variables, the central distinguishing characteristic of the landunit is that this is where physical soil properties are defined: texture, color, depth, pressure-volume relationships, and thermal conductivity.

The second subgrid level, the column, is intended to capture potential variability in the soil and snow state variables within a single landunit. The central characteristic of the column is that this is where the state and flux variables for water and energy in the soil and snow are defined. Regardless of the number and type of pfts occupying space on the column, the column physics operates with a single set of upper boundary

fluxes, as well as a single set of transpiration fluxes from multiple soil levels. These boundary fluxes are weighted averages over all pfts.

The third and final subgrid level is referred to as the plant functional type (pft), but it also includes the treatment for bare ground. It is intended to capture the biophysical and biogeochemical differences between broad categories of plants, in terms of their functional characteristics. All fluxes to and from the surface are defined at the pft level, as are the vegetation state variables (e.g. vegetation temperature, canopy water storage, and carbon for the leaf, stem, and roots).

In addition to state and flux variable data structures for conserved quantities (energy, water, carbon, etc.), each subgrid level also has a physical state data structure for handling quantities that are not involved in conservation checks (diagnostic variables). For example, soil texture is defined through physical state variables at the landunit level, the number of snow layers and the roughness lengths are defined as physical state variables at the column level, and the leaf area index and the fraction of canopy that is wet are defined as physical state variables at the pft level.

The hierarchical subgrid data structures are implemented in the code through the modules **clmtype.F90**, **clmtypeInitMod.F90**, **decompMod.F90** and **initGridCellsMod.F90**. These routines are all in the **/src/main/** subdirectory. The new code makes extensive use of the Fortran 90 implementation of the derived data type. This permits the user to define new data types that can consist of multiple standard data types (integers, doubles, strings) as well as other derived data types.

This subgrid hierarchy is implemented in CLM3.0 as a set of nested derived types. The entire definition is contained in module **clmtype.F90**. Extensive use is made of pointers, both for dynamic memory allocation and for simplification of the derived type referencing within subroutines. The use of pointers for dynamic memory allocation ensures that the number of subgrid elements at each level in the hierarchy is flexible and resolved at run time, thereby eliminating the need to statically declare arrays of fixed dimensions that might end up being sparsely populated. The use of pointers for referencing members of the derived data type within the subroutines provides a coherent treatment of the logical relationships between variables (e.g., the user cannot inadvertently change a pft-level variable within a subroutine that is supposed to operate on the column states and fluxes), and a more transparent representation of the core algorithms (it is easy to tell when the code is in a column or pft loop).

The module, **clmtype.F90**, is organized such that derived types which are members of other derived types are defined first (a Fortran 90 compiler requirement). In particular, the energy and mass conservation data types are defined first, followed by data types constituting the pft level, column level, landunit level, gridcell level and the model domain level. Finally, the hierarchical organization of these types is defined, starting with the model domain level, which consists in part of a pointer to an array of gridcells, each of which consists in part of a pointer to an array of landunits, each of which has a pointer to an array of columns, which each have a pointer to an array of pfts.

Model initialization occurs in module **initializeMod.F90**. A brief summary of the CLM3.0 initialization is provided. For a more detailed discussion, the user is referred to the **CLM3.0 Developer's Guide**. The first step in CLM3.0 initialization is to determine processor and thread decomposition (i.e. "clump" layout). This is done via a call to subroutine **initDecomp** in module **decompMod.F90**. Subsequently, memory is allocated for the clm data structures in subroutine **initClmtype** in module **clmtypeInitMod.F90**. Once memory allocation has occurred, the hierarchy of the data structures (e.g. assignment of pfts to columns, etc.) is determined in subroutine **initGridCells** in module **initGridCellsMod.F90**. Use is made of input gridded datasets defining the spatial distribution of pfts and other surface types (glacier, lake, etc.). Finally, the necessary model filters (e.g. isolating soil points, lake points, etc) are determined in routine **initFilters** in module **filterMod.F90**.

6 CLM3.0 Surface Dataset Formats

As mentioned in section 4.2, CLM3.0 now supports two surface-data formats. The new format differs from the old format in only two variables:

- PFT - removed from the new format surface-data file.
- PCT_PFT - percent of the land gridcell covered by each pft, including pfts with zero percent cover (not percent of the vegetated portion covered by the four dominant pfts).

The new format surface dataset will be created if the following namelist variable is set:

```
mksrf_all_pfts = .true.
```

The original format surface dataset will be created by default, or if

```
mksrf_all_pfts = .false.
```

The surface data in its new format provides more flexibility than in the original format by allowing the user to decide the number of dominant pfts per gridcell for their simulation without creating a new surface-data file each time this number changes. This is possible because all pfts found in the raw data are included in the new format surface dataset in the order that they are listed in the pft-physiology file **FPFTCON**. Using the new format surface-data file as input, the model selects at run-time the 4 dominant natural pfts and places them on a natural vegetation landunit, while it places crops separately on a crop landunit.

The user may change at compile time the model parameter that specifies the number of dominant pfts used in the model. The parameter, **maxpatch_pft** in module **clm_varpar.F90** in the **src/main** directory determines the maximum number of vegetated pfts in the naturally vegetated landunit. Currently this is set to 4. If the user were to increase this value to 6 and recompile the code, then the model would select the 6 dominant natural pfts and place them on the natural vegetation landunit. In the original format, after recompiling the code, a new surface dataset would need to be created, which would contain the 6 dominant pfts (both natural and crop).

Similar to **maxpatch_pft**, the parameter **maxpatch_cft** specifies the maximum number of crop pfts in the crop landunit in the new format. Currently this number is set to 2 and corresponds to corn and wheat vegetation types. In CLM3.0, these vegetation types for corn and wheat are currently hard-wired to values of 15 and 16, respectively, which are identical in terms of their physiological properties. This hard-wiring of values will be removed in future releases.

The new format surface-data file will facilitate changing the land cover in the middle of a simulation. It is important to note, however, that such a capability will require additional code development which is not currently in the CLM3.0 release. The presence of distinct natural and crop landunits will also allow the separate treatment of these landunits using different modules. For example, the dynamic vegetation model (DGVM) in CLM3.0 may be used for the natural landunit, while a crop model could be used for the crop landunit. Although no crop model is included yet and only natural vegetation is permitted when DGVM is active in the CLM3.0 release, the new surface dataset form will facilitate the introduction of these changes in the code.

The new surface-data format and the separate vegetation landunits lead to small changes in CLM's simulated fluxes to the atmosphere. As a result, one will see small changes in the simulated climate in CAM or CCSM mode. Users should decide to work with one of the surface-data formats and not switch between such formats in the middle of a study. However, if a user desires to do surface dataset development, they are strongly encouraged to use the new surface dataset format.

7 Creating a Spun-up Initial Dataset

Often it may take several simulated years to "spin up" the model. For example, in CCSM mode it may take numerous decades to spin up the deep soil liquid water. As a result, it has become necessary to provide users with the tool to create a new initial dataset using a spun up initial dataset at another resolution or landmask. In CLM3.0 we are providing such a tool, **interpinic**, in the directory **tools/interpinic**. The only constraint is that the user must have already created a "template" initial dataset at the new resolution before this tool is used. This can be done by running the model for one day as in Example 1 but with an additional namelist setting:

```
hist_crtinic = 'DAILY'
```

A new initial dataset will be created as a result of this run. The tool **interpinic** will then overwrite the non spun-up values of CLM3.0 variables in this initial dataset with spun-up values from the spun-up initial dataset.

8 History File Fields

The following sections discuss both the fields that may currently be output to CLM3.0 history tapes as well as code modifications that the user must make to add new fields to the history tapes.

Tables 9-18 list the fields that currently may be output to a CLM3.0 history tape. By default, these fields appear on the primary history tape. The dimensions of each field may include 'time' (days since the beginning of the simulation), 'levsoi' (number of soil layers, levsoi = 10), 'levlak' (number of lake layers, levlak=10), and 'lat' and 'lon' (number of latitude and longitude points, e.g., lat=64, lon=128 for a T42 simulation) for grid averaged two dimensional output, and 'gridcell', 'landunit', 'column' or 'pft' for one dimensional output. The dimensions of the RTM fields, QCHOCNR and QCHANR, include 'lonrof' and 'latrof', the number of latitude and longitude points on the RTM grid. The two RTM fields are always output on the 0.5 degree RTM grid regardless of the resolution of the model run. Note that the 1d dimension type appearing in the dimensions entry specifies only the default 1d output type. For example, 'TSA' will be output by default in pft 1d output. However, that default type may be changed for a given history tape via the setting of the namelist variable **HIST_TYPE1D_PERTAPE**. A history field may appear as single-level (SL) or multi-level (ML). Finally, unless explicitly specified in the description, all fields are time averaged over the requested history interval.

Table 9: Master Field List - Temperature and Humidity

Name	Description	Units	1d Output	Level	Spatial Validity
TSA	2-m air temperature	K	pft	SL	global
TV	vegetation temperature	K	pft	SL	global
TG	ground temperature	K	column	SL	global
TSOI	soil temperature	K	column	ML	lakes excluded
TLAKE	lake temperature	K	column	ML	only lakes included
TSNOW	snow temperature	K	column	SL	lakes excluded
TREFMNAV	daily minimum of hourly-averaged 2-m temperature	K	pft	SL	global
TREFMXAV	daily maximum of hourly-averaged 2-m temperature	K	pft	SL	global
Q2M	2-m specific humidity	kg/kg	pft	SL	global

Table 10: Master Field List - Surface Radiation

Name	Description	Units	1d Output	Level	Spatial Validity
FSA	absorbed solar radiation	W/m2	pft	SL	global
SABG	solar radiation absorbed by ground	W/m2	pft	SL	global
SABV	solar radiation absorbed by vegetation	W/m2	pft	SL	global
FSR	reflected solar radiation	W/m2	pft	SL	global
FSRVD	direct visible reflected solar radiation	W/m2	pft	SL	global

Table 10: Master Field List - Surface Radiation

FSRND	direct near-infrared reflected solar radiation	W/m2	pft	SL	global
FSRVDLN	direct visible reflected solar radiation at local noon	W/m2	pft	SL	global
FSRNDLN	direct near-infrared reflected solar radiation at local noon	W/m2	pft	SL	global
FSRVI	diffuse visible reflected solar radiation	W/m2	pft	SL	global
FSRNI	diffuse near-infrared reflected solar radiation	W/m2	pft	SL	global
FIRA	net infrared (longwave) radiation	W/m2	pft	SL	global
FIRE	emitted infrared (longwave) radiation	W/m2	pft	SL	global

Table 11: Master Field List - Surface Energy Fluxes

Name	Description	Units	1d Output	Level	Spatial Validity
FCTR	canopy transpiration	W/m2	pft	SL	global
FCEV	evaporation of canopy-intercepted water	W/m2	pft	SL	global
FGEV	ground evaporation	W/m2	pft	SL	global
FSH	sensible heat flux	W/m2	pft	SL	global
FSH_G	sensible heat from ground	W/m2	pft	SL	global
FSH_V	sensible heat from vegetation	W/m2	pft	SL	global
FGR	heat flux into snow/soil (includes snow melt)	W/m2	pft	SL	global
FSM	snow melt heat flux	W/m2	column	SL	global
TAUX	zonal surface stress	kg/m/s2	pft	SL	global
TAUY	meridional surface stress	kg/m/s2	pft	SL	global

Table 12: Master Field List - Vegetation Phenology

Name	Description	Units	1d Output	Level	Spatial Validity
ELAI	exposed one-sided leaf area index	m2/m2	pft	SL	global
ESAI	exposed one-sided stem area index	m2/m2	pft	SL	global

Table 13: Master Field List - Canopy Physiology

Name	Description	Units	1d Output	Level	Spatial Validity
RSSUN	sunlit leaf stomatal resistance (minimum over time interval)	s/m	pft	SL	lakes excluded
RSSHA	shaded leaf stomatal resistance (minimum over time interval)	s/m	pft	SL	lakes excluded
BTRAN	transpiration beta factor (soil moisture limitation)	unitless	pft	SL	lakes excluded
FPSN	photosynthesis	unitless	pft	SL	global

Table 14: Master Field List - Hydrology

Name	Description	Units	1d Output	Level	Spatial Validity
H2OSOI	volumetric soil water (ratio of water to total soil volume)	mm3/mm3	column	ML	lakes excluded
H2OSNO	snow depth (liquid water equivalent)	mm	column	SL	global
FSNO	fraction of soil covered by snow	unitless	column	SL	global
H2OCAN	water on the canopy	mm	pft	SL	global
SOILLIQ	soil liquid water	kg/m2	column	ML	lakes excluded
SOILICE	soil ice	kg/m2	column	ML	lakes excluded
SNOWLIQ	snow liquid water	kg/m2	column	SL	lakes excluded
SNOWICE	snow ice	kg/m2	column	SL	lakes excluded
SNOWDP	snow height	m	column	SL	global
SNOWAGE	snow age	unitless	column	SL	global
QINFL	water infiltration in soil	mm/s	column	SL	global
QOVER	surface runoff	mm/s	column	SL	global
QRGWL	surface runoff at glaciers, wetlands, and lakes	mm/s	column	SL	global
QDRAI	sub-surface drainage	mm/s	column	SL	global
QINTR	canopy interception of precipitation	mm/s	pft	SL	global
QDRIP	throughfall	mm/s	pft	SL	global
QMELT	snow melt	mm/s	column	SL	global
QSOIL	ground evaporation	mm/s	pft	SL	global
QVEGE	evaporation of canopy-intercepted water	mm/s	pft	SL	global
QVEGT	canopy transpiration	mm/s	pft	SL	global
QCHOCNR	RTM river discharge into ocean (included if **RTM** defined)	m3/s	2-D only	SL	global
QCHANR	RTM river flow (included if **RTM** defined)	m3/s	2-D only	SL	global

Table 15: Master Field List - Water and Energy Balance Checks

Name	Description	Units	1d Output	Level	Spatial Validity
ERRSOI	soil/lake energy conservation error	W/m2	column	SL	global
ERRSEB	surface energy conservation error	W/m2	pft	SL	global
ERRSOL	solar radiation conservation error	W/m2	pft	SL	global
ERRH2O	total water conservation error	mm	column	SL	global

Table 16: Master Field List - Atmospheric Forcing

Name	Description	Units	1d Output	Level	Spatial Validity
RAIN	rain	mm/s	gridcell	SL	global
SNOW	snow	mm/s	gridcell	SL	global
TBOT	atmospheric air temperature	K	gridcell	SL	global
WIND	atmospheric wind velocity magnitude	m/s	gridcell	SL	global
THBOT	atmospheric air potential temperature	K	gridcell	SL	global
QBOT	atmospheric specific humidity	kg/kg	gridcell	SL	global
ZBOT	atmospheric reference height	m	gridcell	SL	global
FLDS	incident longwave radiation	W/m2	gridcell	SL	global
FSDS	incident solar radiation	W/m2	gridcell	SL	global
FSDSVD	direct visible incident solar radiation	W/m2	pft	SL	global
FSDSND	direct near-infrared incident solar radiation	W/m2	pft	SL	global
FSDSVDLN	direct visible incident solar radiation at local noon	W/m2	pft	SL	global
FSDSNDLN	direct near-infrared incident solar radiation at local noon	W/m2	pft	SL	global
FSDSVI	diffuse visible incident solar radiation	W/m2	pft	SL	global
FSDSNI	diffuse near-infrared incident solar radiation	W/m2	pft	SL	global

Table 17: Master Field List - Soil

Name	Description	Units	1d Output	Level	Spatial Validity
ZSOI	soil layer node depth	m	column	ML	lakes excluded
DZSOI	soil layer thickness	m	column	ML	lakes excluded
WATSAT	volumetric soil water at saturation (equal to the porosity)	mm3/mm3	column	ML	glaciers, wetlands, lakes excluded
SUCSAT	saturated soil matric potential	mm	column	ML	glaciers, wetlands, lakes excluded
BSW	slope of soil water retention curve	unitless	column	ML	glaciers, wetlands, lakes excluded

Table 18: Master Field List - Volatile Organic Compounds (only included if VOC defined)

Name	Description	Units	1d Output	Level	Spatial Validity
BIOGENCO	biogenic CO flux	μg/m2/h	pft	SL	global
ISOPRENE	isoprene flux	μg/m2/h	pft	SL	global
MONOTERP	monterpene flux	μg/m2/h	pft	SL	global
ORVOC	other reactive VOC flux	μg/m2/h	pft	SL	global
OVOC	other VOC flux	μg/m2/h	pft	SL	global
VOCFLXT	total VOC flux into atmosphere	μg/m2/h	pft	SL	global

Table 19: Master Field List - Dynamic Vegetation (only included if DGVM defined)

Name	Description	Units	1d Output	Level	Spatial Validity
FMICR	microbial respiration	μmol/m2/s	pft	SL	global
FRMS	stem maintenance respiration	μmol/m2/s	pft	SL	global
FRMR	root maintenance respiration	μmol/m2/s	pft	SL	global
FRMF	foliage maintenance respiration	μmol/m2/s	pft	SL	global
FRG	growth respiration	μmol/m2/s	pft	SL	global
FCO2	net CO2 flux	μmol/m2/s	pft	SL	global
DMI	net primary production	μmol/m2/s	pft	SL	global
HTOP	height of top of canopy	m	pft	SL	global
HBOT	height of bottom of canopy	m	pft	SL	global
TLAI	total one-sided leaf area index	m2/m2	pft	SL	global
TSAI	total one-sided stem area index	m2/m2	pft	SL	global
TDA	daily average 2-m temperature	K	pft	SL	global
T10	10-day running mean of 2-m temperature	K	pft	SL	global
AGDD0	growing degree-days base 0 degrees C	degree-days	pft	SL	global
AGDD5	growing degree-days base 5 degrees C	degree-days	pft	SL	global

Note that for snow related fields (e.g. SNOWLIQ), horizontal averaging is done only using columns that have snow. In this horizontal averaging, lake subgrid points are excluded. Furthermore, for snow related fields, vertical averaging is done by summing (e.g., SNOWLIQ) or averaging (e.g., TSNOW) only over valid snow layers.

Also note that additional history fields appear in the primary history file when DGVM is defined. In addition, a separate history file is produced once per year when DGVM is defined. This file is described in the CLM-DGVM user's guide.

9 Offline Mode Namelist Examples

The following examples illustrate different namelist options that can be used to run CLM3.0 in offline mode.

9.1 Example 1: Offline initial run, one day, global

When the model is run in offline mode using a pre-existing surface dataset, the minimum namelist parameters are: **CASEID**, **NSREST**, **NESTEP** or **NELAPSE**, **FSURDAT**, **FPFTCON**, **OFFLINE_ATMDIR**, **START_YMD**, and **DTIME**. If **FSURDAT** is blank, a surface dataset will be generated at run time and additional variables need to be specified (see section 4.2 and Examples 5 and 6). Namelist parameters not specified will be set to default values. The following gives an example of a simple namelist.

```
&clmexp
CASEID          =  'test01'
NSREST          =   0
NESTEP          =  -1
FSURDAT         =  '$CSMDATA/srfdata/clms_64x128_USGS_c030605.nc'
FINIDAT         =  ' '
FPFTCON         =  '$CSMDATA/pftdata/pft-physiology'
FRIVINP_RTM     =  '$CSMDATA/rtmdata/rdirc.05'
OFFLINE_ATMDIR  =  '$CSMDATA/NCEPDATA'
START_YMD       =  19971231
DTIME           =  1800
HIST_NHTFRQ     =  -24
/
```

CASEID = 'test01'
Case identifier which distinguishes this particular simulation from another. The string in **CASEID** shows up in the names of history, restart, and initial files, in the restart pointer file name (see Example 2) and in the Mass Store pathname where history, restart, and initial files are placed if the Mass Store is used.

NSREST = 0
Requests an initial run, as opposed to a restart or a branch run. An initial run does not require the use of an initial input datafile (**FINIDAT**). If none is provided, the model uses non spun-up initialization provided in the code (see **src/main/iniTimeVar.F90**).

NESTEP = -1
Specifies the run's ending time to be at the end of day 1.

FSURDAT = '$CSMDATA/srfdata/clms_64x128_USGS_c030605.nc'
Specifies the name of the surface data input file. This particular T42 surface dataset can be used both in cam and offline mode. The model resolution (i.e. parameters **LSMLON** and **LSMLAT**) must be compatible with the resolution of **FSURDAT**. If the filename appeared without a path specifying its exact location, the file would be expected in the executable directory, defined by the environment variable **$MODEL_EXEDIR**.

FINIDAT = ' '
Specifies the initial file to be used to prescribe initial values for time-dependent variables. Since no file is specified in this case, the model will be internally initialized to non spun-up values (arbitrary initialization).

FPFTCON = '$CSMDATA/pftdata/pft-physiology'
Specifies a file with PFT (Plant Functional Type) parameters.

FRIVINP_RTM = '$CSMDATA/rtmdata/rdirc.05'
Specifies the input file required for the operation of RTM (River Transport Model). By default, RTM will operate at half degree horizontal resolution and will be invoked every 3 hours, where the fluxes input to RTM (i.e., runoff) are averaged over the 3 hour period. If the user wants the RTM scheme to operate at a different frequency than once every 3 hours, **RTM_NSTEPS** should be set to the desired value of timesteps. Use of RTM is activated in the jobscript.csh with the C pre-processor (cpp) directive #define RTM in the header file **preproc.h** (see section 3.1.3).

OFFLINE_ATMDIR = '$CSMDATA/NCEPDATA'
Specifies the location of the atmospheric driver data set. Such a data set is required for the model to run in offline mode.

START_YMD = 19971231
Specifies the base date of the simulation and must be compatible with the atmospheric input data. For example, **START_YMD** = 19971231 will use the atmospheric input file 1997-12.nc. In a restart or branch run, **START_YMD** need not be changed, as long as it refers to a date earlier than the date of restart or branch.

DTIME = 1800
Specifies the simulation's timestep in seconds. In offline mode, the model can handle a timestep of up to 3600 seconds.

HIST_NHTFRQ = -24
Primary history files and restart files will be produced in the executable directory and will be written every 24 hours.

9.2 Example 2: Restart run

The following namelist generates a restart run which continues the run in Example 1.

```
&clmexp
CASEID         = 'test01'
NSREST         =  1
NELAPSE        = -1
FSURDAT        = '$CSMDATA/srfdata/clms_64x128_USGS_c030605.nc'
FINIDAT        = ' '
FPFTCON        = '$CSMDATA/pftdata/pft-physiology'
FRIVINP_RTM    = '$CSMDATA/rtmdata/rdirc.05'
OFFLINE_ATMDIR = '$CSMDATA/NCEPDATA'
START_YMD      =  19971231
DTIME          =  1800
HIST_NHTFRQ    =  -24
/
```

NSREST = 1
Requests a restart run. A restart run finds the name of the appropriate restart file automatically by reading the file, lnd.**CASEID**.rpointer found by default in the user's home directory. In this example, the pointer file will be lnd.test01.rpointer. Restart runs are meant to be 'seamless', producing the same output as runs which continued without a restart.

NELAPSE = -1
Specifies the run's ending time to be one day after the point of restart. This is equivalent to entering **NESTEP** = -2, since the previous run stopped at the end of day 1. If **NESTEP** were used in this namelist, it would override the value given to **NELAPSE**.

All other namelist variables remain the same to ensure a 'seamless' restart (for information, see example 1). Also, for a seamless restart, the user should generally execute the code with the same executable used in the initial run (ie, without compiling the code again). The jobscript will not recompile the code unless the user has made changes to the code or files have been removed from the bld directory.

9.3 Example 3: Branch run

The following namelist generates a branch run starting from restart files generated by Example 1. The user may branch a run with the same executable used in the initial run (i.e., without recompiling the code) unless branching is used to test changes in the code (for debugging or sensitivity purposes).

```
&clmexp
CASEID          = 'branch_run'
NSREST          =  3
NREVSN          = 'test01.clm2.r.1998-01-01-00000'
NELAPSE         = -1
FSURDAT         = '$CSMDATA/srfdata/clms_64x128_USGS_c030605.nc'
FINIDAT         = ' '
FPFTCON         = '$CSMDATA/pftdata/pft-physiology'
FRIVINP_RTM     = '$CSMDATA/rtmdata/rdirc.05'
OFFLINE_ATMDIR = '$CSMDATA/NCEPDATA'
START_YMD       = 19971231
DTIME           = 1800
HIST_FINCL2     = 'TV:I'
HIST_NHTFRQ     = -3,5
HIST_MFILT      =  2,3
/
```

See Example 1 for explanations of namelist variables which remain unchanged.

NSREST = 3
Requests a branch run.

NREVSN = 'test01.clm2.r.1998-01-01-00000'
Supplies the name of the restart file which will initialize this run. (This file can be produced by running Example 1 above).

NELAPSE = -1
Specifies the run's ending time to be one day after the point of branching.

HIST_FINCL2 = 'TV:I'
Add an auxiliary history file with the field "TV" that is output instantaneously.

HIST_NHTFRQ = -3,5
Changes the frequency of primary history writes to every 3 hours. The write frequency of the auxiliary file is every 5 time steps. This is an example of a change which a user may wish to test in a branch run.

HIST_MFILT = 2,3
The primary history file will have 2 time samples on every tape. The auxiliary history file will have 3 time samples on every tape.

9.4 Example 4: Auxiliary history files

This example covers the addition of an auxiliary history file, the removal of a field from the primary history file and the change of field type in a history file. A variety of other namelist options are also illustrated.

```
&CLMEXP
CASEID          = 'rtm_run'
NSREST          =  0
NESTEP          = -31
```

```
FSURDAT         = '$CSMDATA/srfdata/clms_64x128_USGS_c030605.nc'
FINIDAT         =  ' '
FPFTCON         = '$CSMDATA/pftdata/pft-physiology'
FRIVINP_RTM     = '$CSMDATA/rtmdata/rdirc.05'
OFFLINE_ATMDIR  = '$CSMDATA/NCEPDATA'
START_YMD       =  19980101
DTIME           =  1800
HIST_FEXCL1     = 'TSNOW'
HIST_FINCL2     = 'TV','TG:I'
HIST_DOV2XY     = .true.,.false.
HIST_NHTFRQ     = -24,-12
HIST_MFILT      =  4,2
MSS_IRT         = 365
WRTDIA          = .true.
/
```

For namelist variables which are repeated, refer to Examples 1, 2, and 3.

HIST_FEXCL1 = 'TSNOW'
The field 'TSNOW' will be excluded from the primary tape.

HIST_FINCL2 = 'TV','TG:I'
Specifies the two fields to be added to the auxiliary history output. The first field, 'TV', will have the default time averaging done, whereas the second field, 'TG', will have instantaneous output.

HIST_DOV2XY = .true.,.false.
History output will appear in gridded two-dimensional format for the primary file and in one-dimensional subgrid format for the auxiliary file.

HIST_NHTFRQ = -24,-12
History output will be directed to the primary history file every 24 model hours and to the auxiliary history file every 12 hours.

HIST_MFILT = 4,2
Each primary history file will contain 4 time slices of output, while each auxiliary history file will contain 2 time slices of output.

MSS_IRT = 365
Output files will be archived on the NCAR Mass Storage System with a retention time of 365 days.

WRTDIA = .true.
A global average of land surface air temperature as diagnostic will appear in the standard output file of the simulation.

9.5 Example 5. Generation of regional grid surface dataset

A regular grid surface dataset can be generated at run time for a single gridcell or for gridcells comprising a regional or global domain. In all cases, the cpp tokens **LSMLON** and **LSMLAT** must be set to the desired resolution (e.g., **LSMLON**=1, **LSMLAT**=1 for a single gridcell simulation or **LSMLON**=120, **LSMLAT**=60 for a 3 degree by 3 degree global simulation). To generate a surface dataset for a regional run, the variables **MKSRF_OFFLINE_EDGES**, **MKSRF_OFFLINE_EDGEN**, **MKSRF_OFFLINE_EDGEE**, and **MKSRF_OFFLINE_EDGEW** and their values need to be added to the namelist. A surface dataset will be created with the name surface-data.LSMLONxLSMLAT.nc (e.g., for a single point simulation the

file name will be surface-data.001x001.nc). The model can then be run by following Example 1 where **FSURDAT** points to the new surface dataset.

In the following example, a regional grid is created over the Amazon basin. **LSMLON** and **LSMLAT** should be set to 15 and 11, respectively, for 3 degree by 3 degree horizontal resolution.

```
&clmexp
CASEID                 = 'create_regional_surfdat'
NSREST                 = 0
NESTEP                 = 2
START_YMD              = 19971231
DTIME                  = 1800
FSURDAT                = ' '
FRIVINP_RTM            = '$CSMDATA/rtmdata/rdirc.05'
FPFTCON                = '$CSMDATA/pftdata/pft-physiology'
OFFLINE_ATMDIR         = '$CSMDATA/NCEPDATA'
MKSRF_OFFLINE_FNAVYORO = '$CSMDATA/rawdata/mksrf_navyoro_20min.nc'
MKSRF_FVEGTYP          = '$CSMDATA/rawdata/mksrf_pft.nc'
MKSRF_FSOITEX          = '$CSMDATA/rawdata/mksrf_soitex.10level.nc'
MKSRF_FSOICOL          = '$CSMDATA/rawdata/mksrf_soicol_clm2.nc'
MKSRF_FLANWAT          = '$CSMDATA/rawdata/mksrf_lanwat.nc'
MKSRF_FGLACIER         = '$CSMDATA/rawdata/mksrf_glacier.nc'
MKSRF_FURBAN           = '$CSMDATA/rawdata/mksrf_urban.nc'
MKSRF_FLAI             = '$CSMDATA/rawdata/mksrf_lai.nc'
MKSRF_OFFLINE_EDGEN    =  12
MKSRF_OFFLINE_EDGES    = -21
MKSRF_OFFLINE_EDGEE    = -36
MKSRF_OFFLINE_EDGEW    = -81
/
```

FSURDAT = ' '
A surface dataset named surface-data.LSMLONxLSMLAT.nc will be created in the model executable directory. LSMLON and LSMLAT are defined in jobscript.csh (see section 3.1.3).

MKSRF_OFFLINE_FNAVYORO = '$CSMDATA/rawdata/mksrf_navyoro_20min.nc'
Points to the orography dataset used to derive the model's land mask in offline mode. The environment variable **$CSMDATA** is explained in 3.1.1.

MKSRF_FVEGTYP, **MKSRF_FSOITEX**, **MKSRF_FSOICOL**, **MKSRF_FLANWAT**, **MKSRF_FGLACIER**, **MKSRF_FURBAN**, and **MKSRF_FLAI**
Specify the raw (usually high resolution) input datasets used to create the model surface dataset.

MKSRF_OFFLINE_EDGES, **MKSRF_OFFLINE_EDGEN**, **MKSRF_OFFLINE_EDGEE**, and **MKSRF_OFFLINE_EDGEW**
Must be defined for the desired model regional domain because the default values assume a global domain. The units are degrees north for EDGES and EDGEN and degrees east for EDGEE and EDGEW.

9.6 Example 6. Generation of global gaussian surface dataset

Only global surface datasets can be created on a non-regular grid, such as a gaussian grid. To generate a surface dataset on a gaussian grid, the cpp tokens **LSMLON** and **LSMLAT** must be set to the desired resolution (e.g., **LSMLON**=128, **LSMLAT**=64 for a T42 grid), and **MKSRF_OFFLINE_FGRID** must be set to the appropriate dataset in **$CSMDATA/srfdata** specifying the model grid, land mask and

land fraction for the model grid. At T42 resolution, a surface dataset, "surface-data.128x064.nc", will be created in the model executable directory. This dataset may be renamed by the user to be more self-explanatory. For example, the surface dataset created using this example was clms_64x128_USGS_c030605.nc in **$CSMDATA/srfdata**. And with the addition of MKSRF_ALL_PFTS=.true. to this namelist, we created clms_64x128_allpfts_c040426.nc.

The following namelist will result in the generation of a surface dataset on a global gaussian grid.

```
&clmexp
CASEID              = 'create_global_surfdat'
NSREST              =  0
NESTEP              =  2
START_YMD           = 19971231
DTIME               = 1800
FSURDAT             = ' '
FRIVINP_RTM         = '$CSMDATA/rtmdata/rdirc.05'
FPFTCON             = '$CSMDATA/pftdata/pft-physiology'
OFFLINE_ATMDIR      = '$CSMDATA/NCEPDATA'
MKSRF_OFFLINE_FGRID = '$CSMDATA/srfdata/fgrid.clms_64x128_USGS_c030605.nc'
MKSRF_FVEGTYP       = '$CSMDATA/rawdata/mksrf_pft.nc'
MKSRF_FSOITEX       = '$CSMDATA/rawdata/mksrf_soitex.10level.nc'
MKSRF_FSOICOL       = '$CSMDATA/rawdata/mksrf_soicol_clm2.nc'
MKSRF_FLANWAT       = '$CSMDATA/rawdata/mksrf_lanwat.nc'
MKSRF_FGLACIER      = '$CSMDATA/rawdata/mksrf_glacier.nc'
MKSRF_FURBAN        = '$CSMDATA/rawdata/mksrf_urban.nc'
MKSRF_FLAI          = '$CSMDATA/rawdata/mksrf_lai.nc'
/
```

FSURDAT = ' '
A surface dataset named surface-data.128x064.nc will be created at run time in the model executable directory.

MKSRF_OFFLINE_FGRID = '$CSMDATA/srfdata/fgrid.clms_64x128_USGS_c030605.nc'
Points to the dataset containing the model grid, land mask and fractional land for the surface dataset.

MKSRF_FVEGTYP, **MKSRF_FSOITEX**, **MKSRF_FSOICOL**, **MKSRF_FLANWAT**, **MKSRF_FGLACIER**, **MKSRF_FURBAN**, and **MKSRF_FLAI**
Same as Example 5.