

**The CCSM  
Climatological Data Ice Model  
Version 5.0**

Combined  
**User's Guide,  
Source Code Reference,  
and Scientific Description**

Brian G. Kauffman

May 2002

**Community Climate System Model**  
National Center for Atmospheric Research, Boulder, CO  
<http://www.cesm.ucar.edu/models>

---

CVS tag \$Name: dice5\_0 \$ Build date: May 16, 2002

# Contents

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Introduction</b>                                     | <b>2</b>  |
| <b>2</b> | <b>Input Datasets</b>                                   | <b>2</b>  |
| 2.1      | Ice Data . . . . .                                      | 2         |
| 2.2      | Domain Data . . . . .                                   | 2         |
| 2.3      | Namelist . . . . .                                      | 2         |
| <b>3</b> | <b>Namelist</b>   | <b>3</b>  |
| <b>4</b> | <b>Output Datasets</b>                                  | <b>5</b>  |
| 4.1      | History Files . . . . .                                 | 5         |
| 4.2      | Restart Files . . . . .                                 | 5         |
| 4.3      | Runtime Diagnostics . . . . .                           | 6         |
| <b>5</b> | <b>Data Exchanged with the Coupler</b>                  | <b>6</b>  |
| 5.1      | Units Convention . . . . .                              | 6         |
| 5.2      | Time Invariant Data . . . . .                           | 6         |
| 5.2.1    | Data Sent to Coupler . . . . .                          | 7         |
| 5.2.2    | Data Received from Coupler . . . . .                    | 7         |
| 5.3      | Time Variant Data . . . . .                             | 7         |
| 5.3.1    | Data Sent to Coupler . . . . .                          | 7         |
| 5.3.2    | Data Received from Coupler . . . . .                    | 8         |
| 5.3.3    | How Output Data is Derived . . . . .                    | 9         |
| <b>6</b> | <b>Running the Data Model</b>                           | <b>9</b>  |
| 6.1      | Overview . . . . .                                      | 9         |
| 6.2      | Master Run Script and Component Setup Scripts . . . . . | 9         |
| <b>7</b> | <b>Source Code Maintenance</b>                          | <b>10</b> |
| 7.1      | Obtaining Source Code . . . . .                         | 10        |
| 7.2      | Data Model Source Code . . . . .                        | 10        |
| 7.3      | Shared Source Code . . . . .                            | 11        |
| 7.4      | Shared Build Environment . . . . .                      | 11        |

## 1 Introduction

The Climatological Data Ice Model (dice5) functions as the sea ice component in a CCSM configuration. Recall that a *configuration* consists of various independent *component models* (e.g. atmosphere, land, ocean, sea-ice), each connected to a *coupler*. This sea ice model component interacts with the CCSM Coupler just like any ice model would, but it is not an active model, rather, it takes SST data from an input data file, infers an ice extent, and sends this data to the Coupler, ignoring any forcing data received from the Coupler. Typically the input data file contains climatological SST data (e.g. from Shea, Trenberth, & Reynolds). Such a "dummy" ice model is useful for seeing how an active atmosphere component behaves when coupled to climatological ice extent.

**Important note:** When assembling a CCSM configuration, the user must carefully consider the limitations and requirements of all components and make sure that the complete set of component models will interact in a meaningful way. In particular, the user must verify that the data provided by this model is adequate for their specific application.

## 2 Input Datasets

The data ice model requires two input netCDF datasets: a domain dataset specifying the model grid domain and an SST dataset. The SST dataset contains 12 months of climatological SST data.

### 2.1 Ice Data

The ice model state variable *ice fraction* (equivalently, ice extent) is derived from input SST data. On startup, this model reads in SST data from a netCDF file. This file contains 12 months of SST data (presumably monthly mean fields) on a global 2x2 degree grid. An SST data file is provided with the distribution. This file contains Shea, Trenberth, & Reynolds (STR) SST climatology data. The file provided can be replaced by any other SST data file that is in the same format. Because netCDF files are self describing, one can query the file itself for specifics about the file format.

Note: if the CCSM configuration consists of a dummy ocean and a dummy ice model (e.g. coupled to active atmosphere and land models), normally one would coordinate the ice model's SST data with the ocean model's SST data so that the SST's and ice extent were consistent.

### 2.2 Domain Data

On startup, this model reads in domain data from a netCDF file. Data exchanged with the coupler will be on this model domain. This file, called data.domain.nc, contains x & y coordinate arrays as well as a domain mask. This model uses a latitude/longitude grid only, with 2d coordinate arrays  $x(i,j)$  &  $y(i,j)$ , and a 2d domain mask  $mask(i,j)$ . The domain must be a rectilinear grid, but needn't be a latitude/longitude grid. A mask value of 0 indicates land points (i.e. not in the model's domain). The ice data described above is interpolated onto the domain specified by the domain data file – all data exchanged with the coupler is on the domain specified by the domain data file.

### 2.3 Namelist

On startup, the model reads an input namelist parameter file from stdin. The model has very few input namelist parameters. Typically the only input parameters that are required are those that specify how often the model will communicate with the coupler, and those that specify the input data files. Some can also alter the fluxes sent to the coupler. See the section on input namelist variables for a complete list and description of namelist parameters.

### 3 Namelist

The model reads an input namelist from stdin at runtime.

Following is a list and description of available namelist input parameters.

#### case\_name

Type: character(len=16)

Default: " "

Required: no, but highly recommended

Description: This is the case name text string that appears in output files to help identify the model run.

#### case\_desc

Type: character(len=64)

Default: " "

Required: no, but highly recommended

Description: This is a short text string which is included in output files to help identify the model run.

#### rest\_type

Type: character(len=16)

Default: "continue"

Required: no (but default is of limited usefulness)

Description: This selects the run type.

Valid choices are: "startup", "hybrid", "continue", or "branch".

Selecting "branch" makes rest\_bfile a required input.

#### rest\_pfile

Type: character(len=256)

Default: "./rpointer"

Required: no

Description: This is the complete path and name of the restart "pointer file." This must include an existing, NFS mounted directory. All run types will update this file (and create it, if necessary), but only a continuation runs requires that this file exists prior to the start of the run.

#### rest\_bfile

Type: character(len=256)

Default: "null"

Required: required if rest\_type = "branch", ignored otherwise.

Description: This is the file name of the "branch file" (the IC data file).

Note that a prefix like "mss:" is used to indicate a file archival device, Valid prefix options are:

\* "cp:" or no-prefix indicates a normal unix file copy from an NFS mounted file system.

\* "mss:" indicates a file on NCAR's MSS

\* "null:" indicates no archival -- the file name, stripped of any directory information, indicates a file in the current working directory.

rest\_date

Type: integer

Default: 00010101 (year 1, month 1, day 1, encoded yyyyymmdd)

Required: no (ignored for "continuation" runs, optional for others).

Description: This is the restart date.

\* On startup and hybrid runs, rest\_date is the initial date of the simulation.

\* On branch runs, if rest\_date > 0, it will over-ride the date contained in the rest\_bfile IC restart file. Otherwise the date from the rest\_bfile data file is used.

\* On continuation runs rest\_date is not used (the date contained in the IC file is used).

ncpl

Type: integer

Default: 24

Required: no

Description: This specifies how many times per day the model communicates (exchanges data) with the coupler.

data\_dir

Type: character(len=256)

Default: "local"

Required: no

Description: This specifies where input SST data can be found.

Currently there is no option here, the data must be found in the local directory, ie. the SST data must be prepositioned into the model's execution directory prior to execution.

flux\_acc

Type: integer boolean

Default: 0 (false/off)

Required: no

Description: If activated, the dice5 model accumulates ice formed in the ocean component -- this a flux field sent from the ocean to the ice. This accumulated ice can subsequently be melted back into the ocean component if and when there is sufficient melting potential. Melting potential is also a field sent from the ocean to the ice component.

flux\_swpf

Type: real

Default: 0.0

Required: no

Description: This is the "short-wave penetrating factor". This is the fraction of net solar absorbed by the ice model the penetrates through the ice and ends up being absorbed into the ocean below.

mss\_dir

Type: character(len=256)

Default: "null"

Required: yes

Description: This is a file archival directory name including the specification of an archival device. Restart and history files go into this directory. For continuation runs, the initial condition restart file must also be in this directory. A "prefix" is all the characters up to and including the first occurrence of ":". The prefix of the file indicates a storage device. Valid prefix options are:

\* "cp:" or no-prefix indicates a normal unix file copy to an NFS mounted file system.

\* "mss:" indicates archival onto NCAR's MSS

\* "null:" indicates no archival of any sort.

Some of the following mss\_xxx options are only meaningful when the archival device is NCAR's MSS. With code modification, more archival devices could be implemented (eg. "hpss:").

mss\_rtpd

Type: integer

Default: 365

Required: no

Description: Retention period, in days, for data files sent to the MSS.

See the mswrite man pages at NCAR for more details.

mss\_pass

Type: character(len=16)

Default: " " (no password)

Required: no

Description: Mswrite write password string for files sent to the MSS.

Read passwords are not an option with the Coupler.

See the mswrite man pages at NCAR for more details.

info\_debug

Type: integer

Default: 1

Required: no

Description: Debugging information level: 0, 1, 2, or 3.

\* 0 => write the least amount of debugging information to stdout

\* 1 => write a small amount of debugging information to stdout

\* 2 => write a medium amount of debugging information to stdout

\* 3 => write a large amount of debugging information to stdout

## 4 Output Datasets

### 4.1 History Files

The data ice model does not create history files. The only data associated with this model is the data that is already contained in the input datasets.

### 4.2 Restart Files

The dice5 model creates restart files. The only data on the restart files is *accumulated water*, which is ice formed by the ocean component that is stored in dice5 so that it can be subsequently melted back into the

ocean component (thus conserving salinity). By default the dice5 model does not accumulate ice formed by the ocean, and thus in default mode the dice5 model does not actually need the data on its restart file, although it always creates restart files anyway. The ability to accumulate water is activated by the namelist parameter *flux\_Qacc*. See the Namelist section of this document.

### 4.3 Runtime Diagnostics

The data ice model generates diagnostic messages which are written to stdout. This output consists mostly of brief messages that indicate how the simulation is progressing and whether any error conditions have been detected. Stdout also contains a record of the values of all model input parameters.

## 5 Data Exchanged with the Coupler

Each component model exchanges data with the coupler only. Component models have no direct connection with each other – all data is routed through the coupler. Most data is in the form of 2D fields. This data is accompanied by certain timing and control information (arrays of scalar real or integer values), such as the current simulation data and time.

### 5.1 Units Convention

All data exchanged conforms to this units convention:

Sign convention:

positive value  $\Leftrightarrow$  downward flux

Unit convention:

|               |   |
|---------------|---|
| temperature   | ~ Kelvin                                    |
| salt          | ~ g/kg                                      |
| velocity      | ~ m/s                                       |
| pressure      | ~ N/m <sup>2</sup> = Pa                     |
| humidity      | ~ kg/kg                                     |
| air density   | ~ kg/m <sup>3</sup>                         |
| momentum flux | ~ N/m <sup>2</sup>                          |
| heat flux     | ~ W/m <sup>2</sup>                          |
| water flux    | ~ (kg/s)/m <sup>2</sup>                     |
| salt flux     | ~ (kg/s)/m <sup>2</sup>                     |
| coordinates   | ~ degrees north or east                     |
| area          | ~ radians <sup>2</sup>                      |
| domain mask   | ~ 0 $\Leftrightarrow$ an inactive grid cell |

### 5.2 Time Invariant Data

This section provides a list of the time invariant data exchanged between the coupler and each component model. Generally this data is the "domain" data: coordinate arrays, domain mask, cell areas, etc. It is assumed that the domain of all models is represented by a 2D array (although not necessarily a latitude/longitude grid).

### 5.2.1 Data Sent to Coupler

```
domain data
* grid cell's center coordinates, zonal      (degrees north)
* grid cell's center coordinates, meridional (degrees east)
* grid cell's four vertex coordinates, zonal      (degrees north)
* grid cell's four vertex coordinates, meridional (degrees east)
* grid cell area (radians squared)
* grid cell domain mask ( 0 <=> not in active domain)
* ni,nj: the dimensions of the underlying 2D array data structure

time coordination data
* ncpl: number of times per day the component will communicate (exchange
data) with the coupler.

other information
* IC flag: indicates whether the coupler should use model IC's contained
on the coupler's restart file or IC's in the initial message sent from
the component model.
```

### 5.2.2 Data Received from Coupler

```
time coordination data
* date, seconds: the exact time the coupler will start the simulation from.
```

## 5.3 Time Variant Data

This section provides a list of the time-evolving data sent exchanged between the coupler and the data model. Generally this is state, flux, and diagnostic quantities.

Each data model provides the coupler with a set of output fields. Output fields from a model include output states (which can be used by another component to compute fluxes) and output fluxes (fluxes that were computed within the model and which need to be exchanged with another component model).

The coupler provides each component model with input fields. Input fields sent to a model include input states (the state variables of other models, which are needed to do a flux calculation) and input fluxes (a forcing fields computed by some other component).

Flux fields sent to or from the coupler are understood to apply over the communication interval beginning when the data was received and ending when the next message is received. The data models must insure that fluxes sent to the coupler are appropriate in this context.

### 5.3.1 Data Sent to Coupler

```
states
* ice fraction
* surface temperature (Kelvin)
* albedo: visible      , direct
* albedo: near-infrared, direct
* albedo: visible      , diffuse
```



\* albedo: near-infrared, diffuse

fluxes

\* atm/ice: zonal surface stress (N/m<sup>2</sup>)  
\* atm/ice: meridional surface stress (N/m<sup>2</sup>)  
\* atm/ice: latent heat (W/m<sup>2</sup>)  
\* atm/ice: sensible heat (W/m<sup>2</sup>)  
\* atm/ice: longwave radiation, upward (W/m<sup>2</sup>)  
\* atm/ice: evaporation ((kg/s)/m<sup>2</sup>)  
\* ice/ocn: penetrating shortwave radiation (W/m<sup>2</sup>)  
\* ice/ocn: ocean heat used for melting (W/m<sup>2</sup>)  
\* ice/ocn: melt water ((kg/s)/m<sup>2</sup>)  
\* ice/ocn: salt flux ((kg/s)/m<sup>2</sup>)  
\* ice/ocn: zonal surface stress (N/m<sup>2</sup>)  
\* ice/ocn: meridional surface stress (N/m<sup>2</sup>)

diagnostic quantities

\* net shortwave radiation (W/m<sup>2</sup>)  
\* 2 meter reference air temperature (Kelvin)

### 5.3.2 Data Received from Coupler

states

\* ocn: temperature (Kelvin)  
\* ocn: salinity (g/kg)  
\* ocn: zonal velocity (m/s)  
\* ocn: meridional velocity (m/s)  
\* atm: layer height (m)  
\* atm: zonal velocity (m/s)  
\* atm: meridional velocity (m/s)  
\* atm: potential temperature (Kelvin)  
\* atm: temperature (Kelvin)  
\* atm: specific humidity (kg/kg)  
\* atm: density (kg/m<sup>3</sup>)

fluxes

\* ocn: dh/dx: zonal surface slope (m/m)  
\* ocn: dh/dy: meridional surface slope (m/m)  
\* ocn: Q>0: heat of fusion (W/m<sup>2</sup>), or  
Q<0: melting potential (W/m<sup>2</sup>)  
\* atm: shortwave radiation: downward, visible, direct (W/m<sup>2</sup>)  
\* atm: shortwave radiation: downward, near-infrared, direct (W/m<sup>2</sup>)  
\* atm: shortwave radiation: downward, visible, diffuse (W/m<sup>2</sup>)  
\* atm: shortwave radiation: downward, near-infrared, diffuse (W/m<sup>2</sup>)  
\* atm: longwave radiation, downward (W/m<sup>2</sup>)  
\* atm: precipitation: liquid ((kg/s)/m<sup>2</sup>)  
\* atm: precipitation: frozen ((kg/s)/m<sup>2</sup>)

### 5.3.3 How Output Data is Derived

The dice5 ice extent (fractional coverage at each grid cell) is calculated based on the input SST data. Wherever the SST is greater than one degree above freezing the ice fraction is zero. Wherever the SST is below freezing the ice fraction is unity. In between the ice fraction varies linearly from zero to unity.

The ice surface temperature is computed according to a formula that roughly approximates the ice temperature found in standalone versions of CCM2 (an older version of the CCSM2.0 atmosphere component). The formula is (for the northern hemisphere):

$$T_{ice} = 260 + 10\cos(2\pi(rmonth - 8)/12)$$

where  $rmonth = 0$  corresponds to January 1st, and  $rmonth = 6$  corresponds to July 1st.

Surface albedos sent to the coupler are somewhat arbitrary but are based on reasonable values for snow and ice albedos combined with an assumption about the fraction of surface snow vs. ice. The albedo values are:

$$\begin{aligned}\alpha_{visible,direct} &= 0.586 \\ \alpha_{near-infrared,direct} &= 0.774 \\ \alpha_{visible,diffuse} &= 0.557 \\ \alpha_{near-infrared,diffuse} &= 0.772\end{aligned}$$

The atmosphere/ice sensible, latent, and upward-longwave heat fluxes, evaporation, surface stress, and 2-meter reference temperatures are calculated according to an involved formula that is documented in the CCSM2.0 Coupler (cpl5) document. See that document for details. The Coupler does a similar calculation for atmosphere/ocean fluxes.

The default value of penetrating short-wave radiation is zero. Optionally, by changing the namelist parameter `flux_swpf` ("short-wave penetrating factor"), will allow an arbitrary fraction of net solar absorbed to pass through the ice component and be absorbed into the ocean below.

Melt water is set to zero by default. Optionally, the `flux_acc` namelist parameter activates the dice5 model's ability to accumulate ice formed in the ocean component. This ice formed is flux field sent from the ocean to the ice. This accumulated ice can subsequently be melted back into the ocean component if and when there is sufficient melting potential. Melting potential is also a field sent from the ocean to the ice component.

Salt flux is always set to zero.

The ice/ocean surface stress is set equal to the atmosphere/ice surface stress (see above).

## 6 Running the Data Model

### 6.1 Overview

The dice5 cannot run by itself, it can only execute in the context of running the complete CCSM system – a framework that requires atmosphere, ice, land, and ocean components, as well as a coupler component. The scripts that build and run the CCSM system are described in detail in the CCSM 2.0 User's Guide, a holistic guide to running the complete system. This Guide includes a line-by-line explanation of the master run script and data model "setup scripts". A brief description of the CCSM run scripts is below. See the CCSM User's Guide for complete information.

### 6.2 Master Run Script and Component Setup Scripts

Two levels of c-shell scripts are used to build and run the CCSM system. A master "run script" coordinates the building and running the complete system while the component model "setup scripts" are responsible for configuring each individual CCSM component (including dice5). Each CCSM component setup script is

run in its own, separate subdirectory, where its executable resides and in which all of its input and output files are kept. The CCSM execution is controlled by the master script, referred to as "the run script".

The run script has the following tasks:

- a. Set batch system options
- b. Define common build and run environment variables
- c. Select multi-processing and resolution specs
- d. Run the setup script for each component
- e. Run the CCSM Integration.
- f. Archive/harvest/resubmit when this run is finished

The common build and run environment variable defined in the run script are automatically propagated to each of the component model setup scripts. These variables define such things as the machine architecture, the number of CPUs to run on, common experiment and file naming conventions.

Once the master run script has defined the common environment, each of the component models (cpl, atm, ice, lnd, and ocn) are configured using individual component setup scripts (e.g. dice.setup.csh) which:

- a. Parse the environment variables sent from the master run script. These are, in effect, input variables that might be required by a setup script or otherwise might alter the behavior of the setup script.
- b. Position or create any input data files, as necessary, including input namelist data files and climatological data files.
- c. Build the component model executable.

Finally, when all of the component model setup scripts have successfully completed, the run script executes all CCSM components simultaneously. The CCSM component models run simultaneously as a multi-program/multi-data (MPMD) message passing system, using MPI to exchange data.

## 7 Source Code Maintenance

### 7.1 Obtaining Source Code

The source code is available as part of the CCSM 2.0 distribution at <http://www.cesm.ucar.edu/models/> . This distribution includes the source code for all CCSM component models. Documentation for other CCSM component models, as well as input data for running the models, is also available at this site.

### 7.2 Data Model Source Code

The source code is written using standard Fortran 90.

The source code was developed using the CVS revision control system, but only one "tagged" version of the code is available within any source code distribution. The code contains CVS information that can be used to identify the code contained in a particular distribution.

### 7.3 Shared Source Code

The dice5 model source code itself (found in `.../models/ice/dice5/` in the CCSM2.0 distribution) is incomplete and cannot be compiled (due to missing subroutines) unless it is compiled along with "CCSM shared code" (found in `.../models/csm_share/`). This shared code is an un-compiled library of support routines.

The source code itself has no machine dependencies, although the CCSM shared code does have some machine dependencies. One function of the shared code is to collect and isolate machine dependent code, and to provide CCSM component models with machine-independent wrappers to such code.

Another function of the shared code is to provide a mechanism for the various component models to be consistent with one another, for example, to use an identical value for pi or for the latent heat of fusion. Similarly, the shared code contains a library routine for calculating a solar angle, so that all component models can be consistent in their solar angle calculations.

### 7.4 Shared Build Environment

The CCSM distribution includes a shared build environment which includes a makefile (a GNU makefile), a variety of machine-dependent makefile macro files and a dependency generator. This common build environment is used to build all CCSM components including dice5. The build environment is found in `.../models/bld/` subdirectory of the CCSM source code distribution.

The makefile, which requires the use of gnu-make, is machine independent, but it "includes" (a standard make functionality) a machine-dependent macros definition file. Several macros files are included in the distribution, but because such macro definitions are typically very machine and site specific, it is expected that end users will need to create a new macros definition file for their site.

Also part of the build environment is a dependency generator. This is written in standard c, and thus is compiled with the standard Unix cc command. The dependency generator is particularly useful when hacking code, either by modifying some files or adding new ones.