User’s Guide to NCAR CCM2

LINDA BATH
JAMES ROSINSKI
JERRY OLSON

CLIMATE AND GLOBAL DYNAMICS DIVISION
NATIONAL CENTER FOR ATMOSPHERIC RESEARCH
BOULDER, COLORADO
Table of Contents

Acknowledgements vii
Introduction ix
References xi
How To Use This Guide xiii
List of Figures xv
List of Tables xvii
CCM2 Calling Tree xix

I. Using CCM2 1

A. Running the Model 1
   1. Model Run Script 1
   2. Source Code Maintenance 9

B. Model Input Parameters 13
   1. Table of Input Parameters 13
   2. Input Parameters Examples 18
      (a) Example 1 — Initial Run, One Day 18
      (b) Example 2 — Restart Run, 30 Days 23
      (c) Example 3 — Regeneration Run, Retrieve Lost Volume 25
      (d) Example 4 — Branch Run 26
      (e) Example 5 — Initial Run, Add History Tape Fields 28
      (f) Example 6 — Run with Multiyear Ozone Dataset 30

C. Model Datasets 32
   1. Input Datasets 32
      (a) Initial Dataset 32
      (b) Boundary Datasets 35
         (1) Time-Invariant Boundary Dataset 35
         (2) Time-Variant Sea-Surface Temperature (SST) Dataset 36
         (3) Time-Variant Ozone Dataset 37
   2. Model Output Datasets 37
      (a) Model History Tape 38
      (b) Restart/Regeneration Datasets 53

D. Model Printed Output 57
II. CCM2 Internals  73
A. Design Philosophy of CCM2  73
B. Model Data Structures  77
1. Vertical Coordinate  77
2. Gridpoint Data Structures  78
   (a) Model Buffer  80
   (b) History Buffer  83
   (c) Absorptivity/Emissivity Arrays  83
   (d) Local Workspace  83
   (e) In-Core Gridpoint Arrays  84
3. Spectral Data Structures  86
4. Other Common Blocks and Parameter Decks:
   The Parameterization Interface  87
5. Out-of-Core Data Storage: The SSD Work Units  89
C. Disk and Mass Store File Management  90
1. Disk File Management in CCM2  91
2. Use of the NCAR Mass Storage System  92
3. Issuing Shell Commands from the Model  94
D. Continuation Run Logic  96
1. Regeneration Datasets  96
2. How to Make Continuation Runs  98
E. Model Code Flow  100
1. Initialization  101
2. Computational Loops: Code and Data Flow  102
   (a) Time Integration  103
   (b) Latitude Scans  104
   (c) Latitude Pair Driving Routines  105
   (d) Spectral Space Computations Driving Routine  106
   (e) Computational Driving Routines at the Latitude Loop Level  106
F. Multitasking Strategy  109
1. Multitasking Implementation  109
2. Memory Management  112
G. Changing the Model  115
1. Tools for Modifying the Code  115
2. Adding New Variables 115
3. Adding Constituents to the Model:
The Semi-Lagrangian Transport 116
   (a) Initializing Fields 117
   (b) Surface Fluxes, Sources, and Sinks 117
   (c) Adjustment Processes 118
   (d) Global Minima 118
4. Changing Resolution 118
5. Adding Parameterizations 118
6. Trouble-Shooting Model Changes 119
   (a) Resource Allocation Errors 119
   (b) Coding Errors 120
   (c) Formulation Errors 122
7. CCM2 Coding Standard 122
   (a) Code Appearance 123
   (b) Other Fortran Considerations 125
   (c) Program Module Names and Variable Names 125
   (d) Code Structure 126

Glossary A-1

CCM2 Printed Output B-1

CCM2 Parameter Definitions C-1

Master Field List D-1
The authors would like to express appreciation to the members of the Climate Modeling Section, Climate and Global Dynamics Division who read, studied, and suggested improvements to this User’s Guide. We especially want to thank Mariana Vertenstein and John Truesdale, whose vigilance in perusing and testing both the CCM2 code and its documentation throughout the development cycle has led to many error corrections. We would also like to acknowledge Jim Hack’s contributions to this report, and to thank Tom Mayer, Ann Modahl and Francois Thibaud for reviewing the User’s Guide.

We would like to express our thanks to Gloria Williamson of the CCM Core Group for carefully testing all the experimental versions of the Model which led to CCM2, as well as the examples documented here. Her impartial approach to testing and close attention to detail have proven invaluable in developing the final product.
Introduction

This User's Guide describes the overall design of the most recent version of the NCAR Community Climate Model, CCM2. In addition to in-depth information about running the CCM2 model code, the User's Guide gives a detailed description of the code organization and data structures so that CCM2 can be modified by users as mandated by their particular research requirements.

The NCAR CCM2 represents an entirely new atmospheric general circulation model in both function and implementation. The original versions of the NCAR Community Climate Model were based on atmospheric general circulation models developed at two major modeling centers. Version CCM0A had its origins in an early version of the Australian spectral model from the Australian Numerical Meteorology Research Center. CCM0B evolved from an early adiabatic, inviscid version of the spectral model developed at the European Centre for Medium Range Weather Forecasts (ECMWF). The physical parameterizations incorporated into these models included radiation and cloud routines developed at NCAR, as well as convective adjustment, stable condensation, vertical diffusion, surface flux, and surface-energy-balance prescriptions developed at the Geophysical Fluid Dynamics Laboratory (GFDL).

The second generation model, CCM1, evolved from the CCM0B version, and included significant improvements to the radiation and vertical finite-difference schemes. There were more modest improvements to the horizontal and vertical diffusion processes, surface energy exchanges, and the introduction of several new modeling capabilities, such as a seasonal mode in which the specified ocean surface conditions vary with time.

In CCM2, the CCM Development Group has incorporated major changes to all aspects of the parameterized physics, changes in resolved dynamics, and the introduction of new modeling capabilities. In addition to these changes in formulation, the CCM Core Group of programmers has entirely reimplemented the code, with three major objectives: greater ease of use and modification; conformation to a plug-compatible physics interface standard; and the incorporation of single-job multitasking capabilities.

Section I of this report, titled "Using CCM2," presents sample scripts for executing the Model, describes the input parameters, and provides information about input datasets, output file formats, and samples of output from a Model integration.

Section II, "CCM2 Internals," gives a detailed description of the Model control and data flow. The various data structures are described, along with detailed control flow and file management information, and multitasking
capabilities. This section concludes with tips on modifying the basic code, adding and/or changing physical parameterizations, basic trouble-shooting techniques and a description of the “plug-compatible” coding standard adopted for CCM2.

In previous versions of the CCM, the functional details of each model routine were documented in a separate technical report. This information is now internally contained in the newly implemented code, obviating the need for a separate document on the CCM2 program modules. The details of the Model formulation, including the governing equations, numerical algorithms, and physical parameterizations, are described in a companion report entitled Description of the NCAR Community Climate Model (CCM2) (Hack et al., 1992). We expect that the combination of this technical report, the description of the model formulation, and the model code itself, will provide sufficient information to allow users to understand, modify and effectively use the Model.
References


Cray Research, Inc.: I/O Technical Note, SN-3075.

How To Use This Guide

In preparing this User's Guide, the authors have adopted certain conventions to make repeated explanation of CCM2 variables, program modules, etc., unnecessary. The word "Model" with an initial capital letter refers to the standard CCM2 version of the NCAR Community Climate Model.

All CCM2 program module names, such as subroutine or function names, appear in upper case, using a typewriter font (example, LINEMS). CCM2 variable names appear in lower case, also in typewriter font. Cray UNICOS system commands and options to those commands are in bold typewriter font (examples, mswrite and ja -h), and Fortran keywords in italics (example, namelist).

Certain common English terms are used in unique ways when describing the CCM. These terms, plus commonly-used acronyms, appear with definitions in Appendix A, Glossary of Terms.
# List of Figures

| Figure I.1. | CCM2 Initialization Calling Tree. | xx |
| Figure I.2. | CCM2 Time Integration Calling Tree. | xxii |
| Figure I.3.a. | Physics Packages Calling Tree, Part 1. | xxiii |
| Figure I.3.b. | Physics Packages Calling Tree, Part 2. | xxiv |
| Figure I.4. | Semi-Lagrangian Transport Calling Tree. | xxv |
| Figure I.5. | History Tape File Format. | 39 |
| Figure I.6. | History Tape Latitude Record Format. | 49 |
| Figure II.1. | CCM2 Hybrid Vertical Coordinate. | 78 |
| Figure II.2. | Layout of Main CCM2 buffer. | 81 |
| Figure II.3. | Pictorial Representation of /com3d/ Storage. | 85 |
| Figure II.4. | Spectral Storage Arrangement. | 87 |
| Figure II.5. | Use of SSD Work Units in Initialization Phase. | 100 |
| Figure II.6. | Use of SSD Work Units in Time-Stepping Phase. | 101 |
# List of Tables

<table>
<thead>
<tr>
<th>Table I.1</th>
<th>Model Input Parameters</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table I.2</td>
<td>Initial Dataset Fields</td>
<td>33</td>
</tr>
<tr>
<td>Table I.3</td>
<td>Required Fields for Initial Dataset Header</td>
<td>34</td>
</tr>
<tr>
<td>Table I.4</td>
<td>Time-Invariant Boundary Dataset Fields</td>
<td>36</td>
</tr>
<tr>
<td>Table I.5.a</td>
<td>History Tape File Header</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>Record 1 — Integer Variables</td>
<td></td>
</tr>
<tr>
<td>Table I.5.b</td>
<td>History Tape File Header</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>Record 2 — Character Variables</td>
<td></td>
</tr>
<tr>
<td>Table I.5.c</td>
<td>History Tape File Header</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>Record 3 — Real Variables</td>
<td></td>
</tr>
<tr>
<td>Table I.6</td>
<td>Description of Field Information List Arrays, MFLDS and MCFLDS</td>
<td>46</td>
</tr>
<tr>
<td>Table I.7</td>
<td>SI Units</td>
<td>47</td>
</tr>
<tr>
<td>Table I.8</td>
<td>History Tape Latitude Record</td>
<td>50</td>
</tr>
<tr>
<td>Table I.9</td>
<td>Master Regeneration Dataset</td>
<td>54</td>
</tr>
<tr>
<td>Table I.10</td>
<td>Primary Regeneration Dataset</td>
<td>55</td>
</tr>
<tr>
<td>Table I.11</td>
<td>Secondary Regeneration Dataset</td>
<td>56</td>
</tr>
<tr>
<td>Table II.1</td>
<td>System Routine Calls in CCM2</td>
<td>95</td>
</tr>
<tr>
<td>Table C.1</td>
<td>CCM2 Parameter Definitions</td>
<td>C-2</td>
</tr>
<tr>
<td>Table D.1</td>
<td>Master Field List</td>
<td>D-2</td>
</tr>
</tbody>
</table>
CCM2 Calling Tree

The figures in this section depict the CCM2 calling tree, showing all program modules except single-purpose utility routines. These charts read in order of subroutine calls from left to right.

Note the decision block in Figure I.1, Initialization Calling Tree, where processing differs for initial runs vs. continuation runs.

The main time stepping loop is explicitly shown in Figure I.2, Time Integration Calling Tree, where time progresses from left to right.
Figure 1.1. CCM2 Initialization Calling Tree.
Figure 1.2. CCM2 Time Integration Calling Tree.
Figure 1.3.a. Physics Packages Calling Tree, Part 1.
Figure 1.3.b. Physics Packages Calling Tree, Part 2.
Figure 1.4. Semi-Lagrangian Transport Calling Tree.
I. Using CCM2

This section describes the use of CCM2, including instructions for running on the CRAY Y-MP, discussions of the input data files and the printed output, and explanations of error messages. This is meant to be a “cookbook” approach to use of the Model. For more details on the Model design, code, and data flow, see Section II.

A. Running the Model

1. Model Run Script

A UNICOS script for running CCM2 on the NCAR CRAY Y-MP computer may be prepared on the SCD I/O Satellite Computer (meeker), or on any available front-end computer with a connection to the NCAR Cray.

The CCM2 c-shell script is designed to be run in batch mode on the NCAR CRAY Y-MP (shavano) under the UNICOS version 6 operating system. It uses the Network Queueing System (NQS) to specify required system resources via QSUB directives. UNICOS commands then set up an environment for running the Model. The source code is managed using gpp, a preprocessor that prepares the code for submission to the Fortran compiler cft77. Examples on the following pages explain the run script and show the user how to run the Model in its standard form and with user changes.

A separate file system, /ccm on the Cray disk, contains all the necessary files and information for users to access CCM2. A text file, /ccm/READ-ME, contains an explanation of the /ccm file system. A news directory, accessible both from shavano and from the /crestone file system, keeps users advised of CCM2 status. Dated “update” files in /ccm/news/ccm2 (or /crestone/u2/ccm/news/ccm2) give news related to CCM2, and “case” files in /ccm/news/ccm2/controls explain the CCM2 control runs available for analysis. Information concerning this report and other CCM2 documentation will appear in directory /ccm/news/ccm2/docs.

CCM2 is copyrighted by the University Corporation for Atmospheric Research. The code in its entirety may not be moved to and executed on another machine without the express written approval of the NCAR Climate and Global Dynamics Division CCM Coordinating Committee.

The CCM2 source code is available to authorized NCAR Cray users on permanent shavano disk. In directory /ccm/ccm2/T42 is an example Model run script, runmodel.job, which is documented below. In subdirectory /ccm/ccm2/src are all the program modules that go together to make up the CCM2 source code.
Also in /ccm/ccm2/T42 are relocatable binary files that may be linked directly by segldr, the system loader, and executed. File model.o is a relocatable image for running the CCM2 on a single processor ("single-threaded"). File multisubs.o is a relocatable image of the program modules that contain directives to the system automatic multitasking software. If you load this file before model.o, you may run the Model on more than one processor.

An example Model run script is shown in its entirety on the next two pages. This script invokes segldr to load the standard CCM2 relocatable model.o. The Model is executed for a one-half-day simulation. Model input parameters, communicated via the Fortran namelist statement, are explained in detail in “Model Input Parameters” on page 13. Following the script is a detailed explanation of how each command works, along with instructions for using the run script to make changes in the CCM2 code.
Running the Model

```
#QSUB -lT 600
#QSUB -q prem
#QSUB -lM 8Mw
#QSUB -lQ 12Mw
#QSUB -eo
#QSUB
#!/bin/csh -vx
#
# Script to run CCM2 The next two lines cause
# the Model to run multitasked.Remove these lines
# to run single-threaded
#
setenv NCPUS 4
setenv TARGET cray-ymp,numcpus=$NCPUS
#
# The next line ("set mods") indicates that
# modifications to standard CCM2 code are included.
# To run the standard Model, (i.e. there are no code
# modifications) this line should be commented out
# ("#" in column 1).
#
#set mods
#
# Set model case id
#
setenv CASE test1
mkdir $TMPDIR/$CASE
cd $TMPDIR/$CASE
set newvar = 'echo $LOGNAME | tr '[a-z]' '[A-Z]'`
setenv NEWLOG $newvar
(umask 000; mkdir -p /usr/tmp/ccm/$NEWLOG/ccm2/$CASE/hist)
(umask 000; mkdir -p /usr/tmp/$LOGNAME/ccm2/$CASE/rest)
setenv FILEENV ex2.$$
#
# Modifications to standard Model code. If shell
# variable "$mods" is set, the source code is gathered
# and compiled.
#
if ($?mods) then
cat >! mods.$$.F << 'END1'
#include <srclist.F>
'END1'
#
# Define search path for gpp.
#
set gppargs = "-I/crestone/u0/username/ccm2/newstuff\"-
-I/ccm/ccm2/src"
#
# Always include these options to gpp. "-F" allows
# expansion of gpp directive lines. "-U ..." removes
# gpp predefinitions of the given symbols.
```
```
set gppargs = "$gppargs -F -U cray -U CRAY -U CRAY1 -U unix"
#
# Invoke only the gpp phase of cf77. Output will be
# in mods.$$i
#
cf77 -P -wp"$gppargs" mods.$$F
#
# Run code through fmp and cf77 if multitasked, only
# cf77 if not.
if ($?NCPUS) then
    fmp mods.$$i | cf77 -b mods.$$o
else
    cf77 -b mods.$$o mods.$$i
endif
if ($status != 0) then
    echo "Compiler error: Abort"
    exit 1
endif
#
# End of code modification section.
# Now define the set of relocatables to pass
# to segldr.
#
set binaries = /ccm/ccm2/T42/model.o
if ($?NCPUS) then
    set binaries = "/ccm/ccm2/T42/multisubs.o $binarys"
endif
if ($?mods) then
    set binaries = "mods.$$o $binarys"
endif
#
# Start job accounting, then create executable
#
ja
segldr -f indef -o ccm.xx.$$ $binarys \n-L/usr/local/lib,/lib/usr/lib/ccm/ccm2/T42 \n-1 ncaro,ecmfft,net,hpf,mss,ccm \n-M,s
if ($status != 0) then
    echo "Loader error: Abort"
    exit 1
endif
#
# Create namelist file for input to CCM2
#
cat >! n1.dat.$$ << 'END1'
```

Running the Model
RUNNING THE MODEL

The "QSUB" directives at the top of the script specify maximum CPU time (-IT), job class (-q), maximum main memory (-1M), and maximum SSD storage (-1Q) for the job.

The section labelled "Create namelist file for input to CCM2" contains input parameters for defining the Model run. These are explained in detail in "Model Input Parameters" on page 13. You should not submit the Model run script without studying this section carefully.
The following discussion gives detailed information about key sections in the CCM2 run script. A basic knowledge of the UNICOS operating system and the NCAR Cray environment is assumed. Refer to Cray document, *UNICOS User Commands Reference Manual*, SR-2011 for more details.

### Issue QSUB directives

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td># QSUB -t 600</td>
<td>CPU time limit for the job, 600 seconds</td>
</tr>
<tr>
<td># QSUB -q prem</td>
<td>Job class, premium</td>
</tr>
<tr>
<td># QSUB -lM 8Mw</td>
<td>Main memory requirement, 8 megawords</td>
</tr>
<tr>
<td># QSUB -lQ 12Mw</td>
<td>SSD requirement, 12 megawords</td>
</tr>
<tr>
<td># QSUB eo</td>
<td>Echo commands to output as they are executed</td>
</tr>
<tr>
<td># QSUB</td>
<td>End QSUB directives</td>
</tr>
</tbody>
</table>

These memory and SSD limits are sufficient for running the standard Model (T42 horizontal resolution, 18 vertical levels with one constituent) multitasked on four processors. Refer to “Memory Management” on page 112 for more information on memory requirements.

```bash
#!/bin/csh -vx
```

This identifies the script as a c-shell script.

### Set environment variables

- `setenv NCPUS 4`
- `setenv TARGET cray-ymp,numcpus=$NCPUS`
- `setenv FILENV ex2.$$
- `setenv CASE ccm2test`
- `set newvar = 'echo $LOGNAME | tr '[a-z]' '[A-Z]'`
- `setenv NEWLOG $newvar`

- `$NCPUS` A system environment variable specifying the maximum number of CPUs that the autotasker will use in running the Model.
- `$TARGET` A system environment variable providing information to *segldr* about initial stack and heap requirements for the requested number of CPUs.

**NOTE:** Removing these two `setenv` commands from the run script will cause the Model to run on a single processor.

- `$FILENV` A system environment variable, specifying the local file into which the system writes file management information for communication between the shell and the running Model.
- `$CASE` A user environment variable, of up to 8 alphanumeric characters, denoting the identifier for this experiment. The
Model will not run if the case name contains more than 8 characters.

$NEWLOG A user environment variable set to the upper-case version of system variable $LOGNAME, the login name.

```
mkdir $TMPDIR/$CASE
cd $TMPDIR/$CASE
```

These commands make a directory beneath the system temporary directory for this job, and make this directory the current directory. Temporary disk files created during the Model run will reside in this directory, which is unique for each invocation of QSUB. Thus, concurrent Model runs will not overwrite each other’s temporary files. $TMPDIR and any directories beneath it disappear when the Model script completes (this is only true if running in batch mode).

```
{umask 000; mkdir -p /usr/tmp/cfm/$NEWLOG/cas2/$CASE/hist}
{umask 000; mkdir -p /usr/tmp/$LOGNAME/cfm2/$CASE/rest}
```

These commands form the UNICOS (disk) subdirectories where the Model will write its output files. These directories are used by the Model to write history tapes (Model output data) and regeneration data (for continuing an interrupted run), respectively.

The .../$CASE/hist directory is the default directory searched by the Modular Processor for input history tapes. For this reason the umask command makes these directories world-writable. Continuation of the Model is discussed in detail in “Continuation Run Logic” on page 96.

Mass Store pathnames for archiving Model output files are generated by the Model at run time. These are described under “Model Output Datasets” on page 37.
This section defines the proper relocatable binaries to load, based on whether or not the user has requested multitasking by setting $NCPUS as shown above. The subroutines in /u0/ccm2/multisubs.o have been run through the Cray autotasker. The `segldr` command invokes the linking system, presetting core to "indefinite," a value which will result in a system floating point error if used in floating point calculations, and linking in required libraries for the Model execution.

This `cat` command directs the Model namelist input records, up to the `END` tag, to file n1.dat.$$, which is then redirected as input to the Model on execution.

These `assign` commands place information in the file pointed to by $FILENAME, set above, to communicate to the Model that these Fortran units are associated with SSD scratch files (Secondary Data Segments). These

```bash
# Now define the set of relocatables to pass to segldr.
#
set binaries = /u0/ccm2/ccm2.o
if ($?NCPUS) then
  set binaries = "/u0/ccm2/multisubs.o $binary"
endif
if ($?mods) then
  set binaries = "mods.$$ o $binary"
endif
segldr -f indef -o ccm.xx.$$ $binaries \
  -L /usr/local/lib, /lib, /usr/lib, /ccm/ccm2/T42 \
  -L ncaro, ecmff, net, hp, mss, ccm \n  -M, s
if ($status != 0) then
  echo "Loader error: Abort"
  exit 1
endif
```

```
cat >! n1.dat.$$ << 'END1'
ESCCMEXP
CTITLE = 'ccm2t42',
.
  (all namelist input here)
  
$ 'END1'
```

```
assign -F sds.scr fort.11
assign -F sds.scr fort.21
assign -F sds.scr fort.22
assign -F sds.scr fort.60
```
are used as work units by the Model as part of the out-of-core implementation.

Run executable

```bash
ja
pshell ./ccm.xx.$$ < nl.dat.$$ if($status != 0) goto err
ja -sclhft
exit 0
```

CCM2 must be run under `pshell` in UNICOS version 6 because of the calls to system routine `ishell` used to invoke shell commands from the Model. Running under `pshell` prevents UNICOS from replicating the entire program field length every time `ishell` is called.

Error processing

```bash
err
deb u g - B -cl -s ./ccm.xx.$$ exit 1
```

Error processing in this script consists of running the Fortran symbolic debugger, `debug`. This requires that `cft77` be run with the `"-ez"` option. This procedure is recommended only for debugging since it adds substantially to execution time. Reference the Cray Manual, *CF77 Compiling System, Vol.1: Fortran Reference Manual*, for more information on the `debug` command.

2. Source Code Maintenance

The CCM2 source code is stored on shavano disk in `/ccm/ccm2/T42/src` as separate files. There is a separate file for each program module (subroutine, function or block data), named `xxx.F`, and one for each common block, parameter block, or statement function, named `xxx.com`. The "`xxx.com" files may be required by more than one program module, and so are included in each calling module via the `gpp` preprocessor "`#include" directive.

The user may wish to make changes to the CCM2 code before running the Model. This section describes the portion of the Model run script that can be activated when changes are desired. The user must first obtain, perhaps via `ftp`, the Model modules or `common` blocks to change. The following example shows how to inform the run script where to find these changed modules.
Gather source code

```bash
# Modifications to standard Model code. If shell
# variable "$mods" is set, the source code is
# gathered and compiled.
#
# if ($?mods) then
#    cat >! mods.$$.F << 'END1'
#    #include <data.F>
#    #include <preset.F>
#    #include <scan1.F>
#    END1
```

The `cat` command will place the included program modules, data (in file `data.F`), preset, and `scan1`, in a file called "mods.$$.F," where the "$$" will be replaced by a unique process identifier. Each program module potentially contains include directives for `common` statements, etc. Thus, the entire source required by the changes is gathered and directed to `mods.$$.F`, ready for the Fortran compiler.

Run preprocessor

```bash
# Define search path for gpp.
#
# set gppargs = "-I/crestone/u0/username/ccm2/newstuff \
#  -I/ccm/ccm2/src"
#
# Always include these options to gpp. "-F" allows
# expansion of gpp directive lines. "-U ..." removes
# gpp predefined of the given symbols.

set gppargs = "$gppargs -F -U cray -U CRAY -U CRAY1 \
              -U unix"
#
# Invoke only the gpp phase of cf77. Output will be
# in mods.$$.i
#
# cf77 -P -Wp"$gppargs" mods.$$.F
```

Here we set a user environment variable, `$gppargs`, to the directory search path for the changed code. Note that the standard CCM2 code directory, `/ccm/ccm2/src`, must be the final entry in the search path, since the changed routines may require include files that do not exist in user directories.

The directory denoted as “`/crestone/u0/username/ccm2/newstuff`” is to be replaced by one or more of the user’s own directories containing the altered code. For this access method to work, these directories must exist either on the Cray or on a file system that is remote-mounted to the Cray, such as the `/crestone` file system. The `gpp` preprocessor...
will search for files in this path, choosing the first occurrence of each requested file.

The additional setting of environment variable $gppargs adds other required arguments to the preprocessor. Then cf77 is invoked to run gpp only. Its output file will be called “mods.$$ .i.”

```
# Run code through fmp and cf77 if multtasked, # only cf77 if not.
if ($?NCPUS) then
   fmp mods.$$ .i | cf77 -b mods.$$ .o
else
   cf77 -b mods.$$ .o mods.$$ .i
endif
if ($status != 0) then
   echo "Compiler error: Abort"
   exit 1
endif
```

This if-else-endif block instructs the script to run the autotasking preprocessor on the Model code only if a number of CPUs have been requested earlier in the script. Even if $NCPUS = 1, this preprocessor will be run. To be truly single-threaded, remove the setting of $NCPUS and $TARGET shown in “Model Run Script” on page 1 and any autotasking directives will be ignored by the Fortran compiler. This compilation will be for your changes only. It takes approximately 2 minutes of CPU time to compile the entire Model.

If the autotasking and/or compiling steps were not successful, the script will abort with an error message. The final endif ends the main block of commands that are activated if environment variable $mods is set.

```
if ($?mods) then
   set binaries = "mods.$$ .o $binarys"
endif
segldr -f indef -o ccm.xx.$$ $binarys \
   -L /usr/local/lib,/lib,/usr/lib,ccm/ccm2/T42 \
   -I ncaro,ecmfft,net,hpf,mss,ccm \
   -M ,s
```

The if-endif block shown redefines the user environment variable $binarys to include the newly compiled changes. Thus, segldr will load the changed routines ahead of the Model relocatable.
CAUTION: Care must be taken in changing CCM2 common blocks or other program elements that are contained in "include" files (those named "xxx.com"). All routines which include a changed include file must also be recompiled.
B. Model Input Parameters

1. Table of Input Parameters

The Model uses Fortran namelist, an input parameter processing capability in Cray’s extended Fortran compiler, to process all input parameters. Table I.1 on page 14 shows all the available namelist input parameters to CCM2. Parameters are subdivided by function to make the table easier to use. This table includes the following:

- The Preset Value column shows the setting from subroutine PRESET.
- The See Example column lists the number(s) of the relevant example (examples follow the table).
- The Required column (labelled “Req’d”) indicates whether parameters are required or optional. Even though each parameter has a “preset” value, that value may be present only for consistency checks across parameter values, i.e., it may not suffice for the Model run.
- The Description column briefly lists the purpose of the parameter. Some of these descriptions include “default” settings — those which the Model provides for the run if the user did not specify the parameter. Certain parameters under “History tape options” are described as arrays. These are parameters which take a value for each of the declared history tapes, primary and auxiliary. For more detailed descriptions, see the examples following Table I.1.

As noted, many parameters take default values. The input parameters that must be specified on an initial run include NCDATA, the initial dataset pathname; BNDTT, the time-invariant boundary dataset pathname; BNDTVS, the SST dataset pathname; and BNDTVO, the ozone dataset pathname. However, you may wish to specify values other than the defaults. Indeed, it is useful to specify certain input parameters in the script, even if they take the default values, to keep track in the future of what input values were used for the run.
Table 1.1  
Model Input Parameters

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Preset Value</th>
<th>See Example</th>
<th>Req'd</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Define the run</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTITLE</td>
<td>blanks</td>
<td>1</td>
<td>yes</td>
<td>Case title, up to 80 characters</td>
</tr>
<tr>
<td>NSREST</td>
<td>0</td>
<td>1, 2, 3, 4</td>
<td>no</td>
<td>Run type: 0=initial 1=restart 2=regeneration 3=branch run</td>
</tr>
<tr>
<td>NESTEP</td>
<td>-999999999</td>
<td>1</td>
<td>no</td>
<td>Ending timestep (positive) or ending day (negative) of run</td>
</tr>
<tr>
<td>NELAPSE</td>
<td>-999999999</td>
<td>2</td>
<td>no</td>
<td>Elapsed time in iterations (positive) or days (negative) to run. May be input instead of NESTEP. Either NESTEP or NELAPSE must be input</td>
</tr>
<tr>
<td><strong>Input datasets</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NCDATA</td>
<td>blanks</td>
<td>1, 6</td>
<td>yes</td>
<td>Initial dataset MSS pathname. Required for initial run</td>
</tr>
<tr>
<td>BNMTI</td>
<td>blanks</td>
<td>1</td>
<td>yes</td>
<td>Boundary dataset MSS pathname, time-invariant</td>
</tr>
<tr>
<td>BNMTVS</td>
<td>blanks</td>
<td>1</td>
<td>yes</td>
<td>Boundary dataset MSS pathname, time-variant sea-surface temperatures</td>
</tr>
<tr>
<td>SSTCYC</td>
<td>.TRUE.</td>
<td>1</td>
<td>no</td>
<td>Flag for cycling SST dataset. If .false., assume multiyear dataset</td>
</tr>
<tr>
<td>BNMTVO</td>
<td>blanks</td>
<td>1, 6</td>
<td>yes</td>
<td>Boundary dataset MSS pathname, time variant ozone mixing ratios</td>
</tr>
<tr>
<td>OZNCYC</td>
<td>.TRUE.</td>
<td>1, 6</td>
<td>no</td>
<td>Flag for cycling ozone dataset. If .false., assume multiyear dataset</td>
</tr>
<tr>
<td>Variable Name</td>
<td>Preset Value</td>
<td>See Example</td>
<td>Req’d</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
<td>-------------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>Mass Store information</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NSVSN</td>
<td>blanks</td>
<td>1</td>
<td>no</td>
<td>Restart dataset name. If not input, no restart data will be written</td>
</tr>
<tr>
<td>NREVSN</td>
<td>blanks</td>
<td>3, 4</td>
<td>no</td>
<td>Regeneration dataset name (local to subroutine DATA)</td>
</tr>
<tr>
<td>NSWRPS</td>
<td>blanks</td>
<td>1</td>
<td>no</td>
<td>Mass Store write password for all output datasets</td>
</tr>
<tr>
<td>IRT</td>
<td>99999</td>
<td>1, 5</td>
<td>no</td>
<td>Retention period for Mass Store output volumes. (Defaults to 365 for ‘CTPUBLIC’, or 4095 if a virtual volume is specified)</td>
</tr>
<tr>
<td>RIRT</td>
<td>99999</td>
<td>1, 5</td>
<td>no</td>
<td>Regeneration dataset retention period</td>
</tr>
<tr>
<td>NSMVN</td>
<td>‘CTPUBLIC’</td>
<td>1, 5</td>
<td>no</td>
<td>Virtual volume for history tape output</td>
</tr>
<tr>
<td>NRMVN</td>
<td>‘CTPUBLIC’</td>
<td>1</td>
<td>no</td>
<td>Virtual volume for regeneration data output</td>
</tr>
<tr>
<td>ASYNC</td>
<td>.TRUE.</td>
<td>1</td>
<td>no</td>
<td>Flag for asynchronous dispose to Mass Store</td>
</tr>
<tr>
<td>STFNUM</td>
<td>1</td>
<td>1</td>
<td>no</td>
<td>Start history tape naming with this number</td>
</tr>
<tr>
<td>History tape options</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NDENS</td>
<td>1, 1, 1, …</td>
<td>1, 4</td>
<td>no</td>
<td>Array specifying packing densities (1, 2, 3 or 4) for primary and auxiliary history tapes</td>
</tr>
<tr>
<td>NHTFRQ</td>
<td>-24, 0, 0, …</td>
<td>1, 4</td>
<td>yes</td>
<td>Array of history tape write frequencies for primary and auxiliary tapes</td>
</tr>
<tr>
<td>Variable Name</td>
<td>Preset Value</td>
<td>See Example</td>
<td>Req'd</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
<td>-------------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>MFILT</td>
<td>5,5,5,...</td>
<td>1,4</td>
<td>no</td>
<td>Array of number of logical files to write to primary and auxiliary history tapes</td>
</tr>
<tr>
<td>NLFILT</td>
<td>.FALSE.</td>
<td>1</td>
<td>no</td>
<td>Flag to put one extra file on first history tape of case</td>
</tr>
<tr>
<td>NINAVG</td>
<td>'A','A',...</td>
<td>1,5</td>
<td>no</td>
<td>Array of field averaging flags for primary and auxiliary history tapes: 'A' = averaged 'I' = instantaneous 'M' = point minimum 'X' = point maximum. ORO and PHIS are always instantaneous</td>
</tr>
<tr>
<td>EXCLUDE</td>
<td>blanks</td>
<td>4</td>
<td>no</td>
<td>List of fields to exclude from the primary history tape</td>
</tr>
<tr>
<td>PRIMARY</td>
<td>blanks</td>
<td>5</td>
<td>no</td>
<td>List of fields to include in primary history tape (must be in master field list)</td>
</tr>
<tr>
<td>AUXF</td>
<td>blanks</td>
<td>4</td>
<td>no</td>
<td>Array to define up to 5 auxiliary history tapes</td>
</tr>
<tr>
<td>LDEBUG</td>
<td>.FALSE.</td>
<td>1</td>
<td>no</td>
<td>If true, link final output volumes into /usr/tmp/$LOGNAME</td>
</tr>
<tr>
<td><strong>Regeneration options</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NREFRQ</td>
<td>1</td>
<td>1,2</td>
<td>no</td>
<td>Frequency in terms of full history tapes for restart data</td>
</tr>
<tr>
<td>MXSZRG</td>
<td>150</td>
<td>1</td>
<td>no</td>
<td>Maximum size allowed for regeneration dataset. If MXSZRG &lt; 10^6 units are megabytes, otherwise in bytes</td>
</tr>
</tbody>
</table>
### Table 1.1
Model Input Parameters

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Preset Value</th>
<th>See Example</th>
<th>Req'd</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model time</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NNDBAS</td>
<td>-9999</td>
<td>1</td>
<td>no</td>
<td>Base Model day for run. If not input, set to NDCUR from initial dataset header</td>
</tr>
<tr>
<td>NNSBAS</td>
<td>-9999</td>
<td>1</td>
<td>no</td>
<td>Seconds of base Model day for run. If not input, set to NSCUR from initial dataset header</td>
</tr>
<tr>
<td>NNBDAT</td>
<td>-9999</td>
<td>1,6</td>
<td>no</td>
<td>Base date for run as yymmdd. If not input, set to NCDATE from initial dataset header</td>
</tr>
<tr>
<td>NNBSEC</td>
<td>-9999</td>
<td>1,6</td>
<td>no</td>
<td>Seconds of base date. If not input, set to NCSEC from initial dataset header</td>
</tr>
<tr>
<td>DTIME</td>
<td>1200.</td>
<td>1</td>
<td>no</td>
<td>Length of Model timestep in seconds</td>
</tr>
<tr>
<td><strong>Physics controls</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EPS</td>
<td>0.06</td>
<td>1</td>
<td>no</td>
<td>Time filter coefficient</td>
</tr>
<tr>
<td>DIF2</td>
<td>2.5E5</td>
<td>1</td>
<td>no</td>
<td>$V^2$ horizontal diffusion coefficient</td>
</tr>
<tr>
<td>DIF4</td>
<td>1.E16</td>
<td>1</td>
<td>no</td>
<td>$V^4$ horizontal diffusion coefficient</td>
</tr>
<tr>
<td>KMXHDC</td>
<td>1</td>
<td>1</td>
<td>no</td>
<td>Number of levels over which to apply Courant limiter, starting at top of Model</td>
</tr>
<tr>
<td>NLVDRY</td>
<td>3</td>
<td>1</td>
<td>no</td>
<td>Number of layers from the top of the Model over which to do dry convective adjustment. Must be less than plev</td>
</tr>
<tr>
<td>IRAD</td>
<td>-1</td>
<td>1</td>
<td>no</td>
<td>Frequency of radiation in iterations (if positive) or model hours (if negative)</td>
</tr>
</tbody>
</table>
### Table I.1

**Model Input Parameters**

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Preset Value</th>
<th>See Example</th>
<th>Req'd</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRADAE</td>
<td>-12</td>
<td>1</td>
<td>no</td>
<td>Frequency of absorptivity/emissivity calculations in iterations (if positive) or model hours (if negative)</td>
</tr>
<tr>
<td>ITSST</td>
<td>1</td>
<td>1</td>
<td>no</td>
<td>Frequency of SST update in timesteps</td>
</tr>
</tbody>
</table>

2. **Input Parameters Examples**

The example input parameters in this section illustrate the use of each parameter in Table I.1 on page 14. *Namelist* input may appear in any order between `E$CCMEXP` and the final `$` (shown on page 19). All *namelist* variables begin in column 2.

(a) **Example 1 — Initial Run, One Day**

The input parameters shown in this example request an “initial” run of CCM2 for a one-day simulation. The simulation will start from an initial dataset of 1 September and use the standard boundary condition datasets from the Cray disk.
assign -F sd.s.scr fort.22
assign -F sd.s.scr fort.60
cat > nl.dat.$$ << 'END1'

ESCMMEXP
CTITLE = 'CCM2 standard run',
NCDATA = '/CCM2/T42/%data%/SEP1',
BNDDT = '/CCM2/T42/%data%/tibds',
BNDDTS = '/CCM2/T42/%data%/tvbds',
BNDTVO = '/CCM2/T42/%data%/ozn',
IRT = 300,
RIRT = 20,
NSMVN = 'CTPUBLIC',
NRMVN = 'CTPUBLIC',
ASYNC = .TRUE.,
STFNUM = 100,
NSREST = 0,
NSVSN = 'xstrt',
NSWRPS = 'mypass',
NDENS = 2,
NNBDAT = 000901,
NNBSEC = 0,
NNDBAS = 0,
NNBAS = 0,
MFILT = 10,
NLFILT = .FALSE.,
DTIME = 1200.,
NESTEP = 72,
NHTRQ = 72,
NFRE = 1,
NINAVG = 'A',
LDEBUG = .FALSE.,
MXSZR = 100,
IRAD = -1,
IRADA = -12,
ITSS = 1,
SSTC = .T.,
OZNCYC = .T.,
EPS = .06,
DIF2 = 2.5E5,
DIF4 = 1.E16,
KMXD = 1,
NLVDRY = 3,
$'
'END1'
ja
pshell ./ccm.xx.$$ < nl.dat.$$
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCDATA</td>
<td>This dataset defines the initial conditions for the Model run. The Model can start from a CCM2 history tape, packed or unpacked. See “Initial Dataset” on page 32.</td>
</tr>
<tr>
<td>BNDTI</td>
<td>These standard boundary condition datasets are stored on permanent disk on shavano. BNDTI contains all time-invariant fields, including surface albedo data and orography standard deviations for the gravity wave drag scheme. BNDTVS contains only the time-variant sea-surface temperature (SST) data. The time-variant ozone data are on the separate file BNDTVO because it contains an arbitrary number of levels and may not be longitudinally dependent.</td>
</tr>
<tr>
<td>BNDTVS</td>
<td></td>
</tr>
<tr>
<td>BNDTVO</td>
<td></td>
</tr>
<tr>
<td>IRT</td>
<td>The history tapes from this run will be written to the Mass Store with a retention period of 300 days. According to the NSMVN input parameter in this deck, these datasets are written to the public volume CTPUBLIC, and the Model-imposed maximum retention time is 5 years (1825 days).</td>
</tr>
<tr>
<td>RIRT</td>
<td>This retention time is specifically for restart and regeneration datasets. If this parameter is not input and if NSMVN = NRMVN (see Table I.1 on page 14), it defaults to the value of IRT. Since regeneration datasets are not useful after a compiler change or major system upgrade, it is suggested that they not be retained beyond two years.</td>
</tr>
<tr>
<td>NSMVN</td>
<td>These parameters, set here to their default values, may be used to specify private virtual volumes for history tapes and restart/regeneration datasets, respectively. Virtual volumes must be assigned by NCAR Scientific Computing Division personnel.</td>
</tr>
<tr>
<td>NRMVN</td>
<td></td>
</tr>
<tr>
<td>ASYNC</td>
<td>This parameter, set here to the default, allows the user to choose to wait for all disposes to the Mass Store before continuing the run (synchronous operation). The default value of .TRUE. specifies asynchronous transfers. See “Model Output Datasets” on page 37 for more details about this parameter.</td>
</tr>
<tr>
<td>STFNUM</td>
<td>The first history tape written by this job will be named ‘h0100’ instead of the default ‘h0001’, and the first master regeneration dataset will be called ‘r0100’. The maximum number for history tape and regeneration dataset naming is 9999.</td>
</tr>
<tr>
<td>NSREST</td>
<td>Setting NSREST = 0 requests an initial run, that is, a start from the initial dataset, NCDATA. Continuation (restart, regeneration, and branch) runs may be specified via this</td>
</tr>
</tbody>
</table>
NSVSN

NSVSN specifies the filename of the restart dataset, which will take the Mass Store pathname of the regeneration datasets from this run. This file is a text file containing the latest full Mass Store pathname of the master regeneration dataset from this run. If this parameter is not input, no restart or regeneration data will be written.

NSWRPS

This is the Mass Store write password for all output datasets from this run. This password must remain the same throughout a case, or a continuation run cannot update original volumes.

NDENS

In order to get 10 (MFILT) time samples on a single history volume running the standard T42 CCM2, the data must be packed 2 to 1. If unpacked, the most MFILT should be 5. Packing of data results in a loss of significant digits in the data values.

NNBDAT

These time variables are provided to override the times read in from the initial dataset header. In this example, NNBDAT sets the base date for this run to “000901”, or September 1, 1900. Only the month and day of this date are used by the Model unless one or both of the ozone or SST boundary datasets are multiyear. For more information about times in the Model, see “Model Code Flow” on page 100.

MFILT

Setting MFILT to 10 means that each output history tape volume will contain at most 10 time samples.

NLFILT

This parameter, set here to the default value of .FALSE., allows the user to force one extra sample (MFILT+1) to the first history volume for a case. If doing so, be careful not to set MFILT so large that MFILT+1 samples will not fit on a volume.

This parameter aids in bookkeeping efforts for long Model cases, since timestep 0 is always written to the history tape.

DTIME

This specifies the Model timestep for the run in (floating point) seconds. CAUTION: Changing this variable directly impacts the physical parameterizations in the Model and may impact the Model climate. Changes in resolution will probably require a change in DTIME.
NESTEP  This run will end at timestep 72. The same ending time could have been specified as \texttt{NESTEP=-1} (one day for a run with \texttt{DTIME=1200}).

NHTFRQ  \texttt{NHTFRQ} requests the number of timesteps (if a positive value) or the number of Model hours (if negative) between history tape writes. A value of 72 (72 timesteps of 1200 seconds each) indicates one write per day. Timestep 0 data will always be written to the history tape.

NREFRQ  This parameter, set here to the default of one, allows the user to write and dispose regeneration datasets only after \texttt{NREFRQ} history tapes have been filled and disposed.

NINAVG  Setting \texttt{NINAVG='I'} will cause all fields on the history tape to contain instantaneous values.

LDEBUG  Setting this parameter to \texttt{.TRUE.} will cause the Model to link the last output files at the end of the run so that they are accessible from \texttt{/usr/tmp/$LOGNAME}, possibly saving unnecessary staging from the Mass Store to disk when post-processing the data. See “Use of the NCAR Mass Storage System” on page 92.

MXSZRG  The maximum allowable size for a regeneration dataset for this run will be 100 MB. Depending on the resolution of the Model run, this may result in a splitting of the regeneration datasets to meet this requirement.

IRAD  Full radiation calculations will be done once per hour, or every 3 timesteps, assuming a \texttt{DTIME} of 1200. (alternative form, \texttt{IRAD=3}). This is the default value.

IRADAE  Absorptivity/emissivity calculations will be once every 12 hours, for a \texttt{DTIME} of 1200. (alternative form, \texttt{IRADAE=36}).

ITSST  This parameter, set here to the default value of 1 timestep, determines the frequency of SST updates, i.e., interpolations of SST boundary data to the current Model time.

SSTCYC  This parameter, which here is set to the default, assures that the SST dataset (\texttt{BNDTVS}) will be rewound and reread at the end of each Model year. If \texttt{SSTCYC} is set to \texttt{.FALSE.}, the Model times throughout the run must fall within the times (including the year) covered by the SST dataset.

OZNCYC  This parameter, again taking the default value, tells the Model that the ozone dataset is to be cycled for each Model
year. If ZNCYC is set to .FALSE., the Model times throughout the run must fall within the times (including the year) covered by the ozone dataset.

**EPS**

These parameters, all set to their default values in this deck, control various aspects of the Model physics. These values represent some (not all) of the parameters that are tuned for the standard T42 version of CCM2. These parameters should be changed only with great caution!

**(b) Example 2 — Restart Run, 30 Days**

The following example restarts the run from Example 1 and simulates another 10 Model days. The user has requested that restart datasets be written only for every two history tapes, to save on I/O requests and storage charges. The standard CCM2 history tape will be written once per day.

```plaintext
assign -P sds.scr fort.22
assign -P sds.scr fort.60
cat >! nl.dat.$$ << 'END1'
E$CMESSAGE
 CTITLE = 'CCM2 standard run',
 NCDATA = '/CCM2/T42/data/SEP1',
 ENDTI = '/CCM2/T42/data/tibd',
 ENDTV = '/CCM2/T42/data/tvbd',
 ENDTV0 = '/CCM2/T42/data/czn',
 IRT=300,
 RIRT=20,
 NSREST=1,
 NSVSN='rst1',
 NSWPS='mypass',
 NDENS = 2,
 NNBAT = 000901,
 NNBSEC = 0,
 NNBAS = 0,
 NNSBAS = 0,
 MFILT = 10,
 STIME = 1200.,
 NELAPSE = -30,
 NHTFQ = 72,
 NREFQ = 2,
 IRAD = -1,
 IRADAE = -12,
 SSTCYC = .T.,
 $'
 'END1'
as
psnsh ./ccm.xx.$$ < nl.dat.$$
if($status != 0) goto err
ja = sclht
```
NSREST  NSREST=1 indicates a “restart” run, which starts from the most recent set of regeneration datasets and continues the case.

NELAPSE  This run, as a continuation of the previous one-day simulation, will start at the beginning of day 2 and run for 30 days, ending after day 31, i.e., NELAPSE=-30 means “run 30 days beyond the start point.” This could also be specified by setting NESTEP=2232 (timesteps) or NESTEP=-31 (days), assuming a DTIME of 1200 seconds.

NREFRQ  Regeneration data for this run will be written only after each two history tapes have been filled and disposed.
(c) Example 3 — Regeneration Run, Retrieve Lost Volume

If history volume 3 from a previously run case were to somehow be corrupted, we could regenerate it using the following parameter deck. Here we use NSREST=2 to indicate a regeneration run, and the appropriate set of regeneration datasets — that associated with volume 2 — to continue the run for enough time to build the replacement volume. This procedure differs from the restart in example 2, because we return to the regeneration data stored throughout the case to start the Model. See “Continuation Run Logic” on page 96 for details about regeneration data.

If the lost volume had been volume 2, we would have had to rerun from the beginning, since due to the setting of NREFRQ, there are no regeneration data associated with volume 1.

```
cat >! nl.dat.$$ << 'END1'
E$CMEXP
CTITLE = 'CCM2 standard run',
NCDATA = '/CCM2/T42/data%/SEP1',
BNDT1 = '/CCM2/T42/data%/tibds',
BNDTVS = '/CCM2/T42/data%/tvbds',
BNDTVO = '/CCM2/T42/data%/ozn',
IRT=300,
RIRT=20,
NSREST=2,
NSVSN='rstrt',
NREVSN='r0002',
NSWRFS='mypass',
NDENS = 2,
NNBDAT = 000901,
NNBSEC = 0,
NNDBAS = 0,
NNSBAS = 0,
MFILT = 10,
DTIME = 1200.,
NELAPSE = -10,
NHTFRQ = 72,
NREFRQ = 2,
IRAD = -1,
IRADA = -12,
SSTCYC = .T.,
$'
'END1'
ja
```

NSREST

NSREST=2 indicates a “regeneration” run, which starts from a specific set of regeneration datasets and continues the case.
NREVSN This parameter specifies the regeneration data needed to build the required volume(s). Because we want to regenerate volume 3, we start with the regeneration data labelled 2, associated with the end of the second volume. Only the name of the “master regeneration dataset” is required — the Model will build the additional dataset names itself (see “Restart/Regeneration Datasets” on page 53).

(d) Example 4 — Branch Run

In the following example, the user begins a new case by “branching” from a CCM2 control run. The branch run capability allows the user to change history tape specifications (but not physics parameters), while continuing from a chosen point in the simulation of a previous run. See “Continuation Run Logic” on page 96 for more details about branch runs.

In this example, certain fields are excluded from the primary history tape, and placed on a newly declared auxiliary tape, to be written every timestep.
Note the use of NSREST=3 for a branch run, and the additional values assigned to NDENS, MFILT, and NHTFRQ for the auxiliary tape.

```
assign -F sds.scr fort.22
assign -F nds.scr fort.60
cat >! nl.dat.$$ << 'END1'
ESCCMEXP
CTITLE = 'CCM2 branch run',
NCDATA = '/CCM2/T42/%data%/SEP1',
BNTTI = '/CCM2/T42/%data%/tibds',
BNDTVS = '/CCM2/T42/%data%/tvbs',
BNDTVO = '/CCM2/T42/%data%/ozn',
IRT=4000,
RIRT=600,
NSREST=3,
NSVSN='rstrt',
NREVSN= '/CSM/ccm2/control/rest/r0345',
NSWRPS= 'newpass',
NDENS = 2,2,
NNSDAT = 000901,
NNSSEC = 0,
NNDBAS = 0,
NNSBAS = 0,
MFILT = 10,30,
DTIME = 1200.,
MELAPSE = -5,
NHTFRQ = 72,1,
ASYNC = .T.,
IRAD = 1,
IRADAE = -12,
NSMVN = 'CTMINE',
NRMVN = 'CTMINE',
AUXF = '1', 'PRECL', 'PRECC', 'CLOUD',
EXCLUDE = 'PRECL', 'PRECC', 'CLOUD',
$'
'END1'
ja
pshell ./ccm.xx.$$ < nl.dat.$$ if($status != 0) goto err
```

NREVSN This run uses volume r0345 from a hypothetical "control" run to provide the initial Model data for starting this case.

NHTFRQ NHTFRQ, the history tape write frequency, will be every 72 timesteps for the primary tape, and every timestep for the auxiliary tape. A radiation calculation is also done every timestep (IRAD=1), or the fields on the auxiliary tape would not be updated for each write time.

AUXF This list specifies that a single auxiliary tape will be written (number '1'), containing the fields shown. To request a second auxiliary tape, follow this list by
'2', 'field1', 'field2', etc.
Up to five such tapes may be declared.

EXCLUDE
To save file storage, the fields that will be written to the auxiliary tape above are hereby excluded from the primary history tape.

(e) Example 5 — Initial Run, Add History Tape Fields

The following input parameter deck specifies an initial run with changes to the default primary history tape. Changes include adding certain fields that are in the master field list (see Table D.1 in "Appendix D: Master Field List") to the history tape, and making the values instantaneous. Other optional changes are enumerated below.

```bash
assign -F sds.scr fort.22
assign -F sds.scr fort.60
cat >! nl.dat.$$ << 'END1'
E$CCMEXP
CTITLE = 'CCM2 run',
NCMDATA = '/CCM2/T42/%data%/SEP1',
BNDTI = '/CCM2/T42/%data%/tibds',
BNDVS = '/CCM2/T42/%data%/tvbs',
BNDTVQ = '/CCM2/T42/%data%/ozn',
IRT=4000,
RIRT=600,
NSREST=0,
NSVSN='rstrt',
NSWRPS='pass',
NDENS = 1,
NNBDAT = 000901,
NNBSEC = 0,
NNDBAS = 0,
NNSBAS = 0,
STFNUM = 100,
MFILT = 5,
DTIME = 1200.,
NELAPSE = -10,
NHTFRQ = 72,
IRAD = -1,
IRADA = -12,
NSMVN = 'CTMINE',
NNAV = 'I',
PRIMARY = 'PRECSL', 'PRECS', 'RUNOFF',
MXSZRG = 100,
$ 'END1'
ja
pshell ./ccm.xx.$$ < nl.dat.$$```
NSMVN  NSMVN specifies a (fictitious) private virtual Mass Store volume.

IRT   Because NSMVN specifies a private volume, the history tapes from this run may be written to the Mass Store with a retention period of 4000 days.

RIRT  Regeneration datasets are written to 'CTPUBLIC', so a separate retention time less than the Model limit for public volume files is input.

NINAVG Because NINAVG = 'I', all fields on the history tape will be instantaneous values.

PRIMARY This list adds fields to the primary history tape. Note that these fields must be in the master field list, and the OUTFLD calls activated. See “Adding New Variables” on page 115 for more information.
Example 6 — Run with Multiyear Ozone Dataset

In this deck, the user is trying out a new ozone dataset which spans two years, 1989 and 1990. For this run, the base date and time information must be set via parameters NNBDAT and NNBSEC to fall within the time frame covered by this dataset. Because OZNCYC=.F., the Model will check the year number in searching for the appropriate ozone data.

The user has created an initial dataset from December data to use in testing the new ozone dataset.

```plaintext
assign -F sds.scr fort.22
assign -F sds.scr fort.60
cat >! nl.dat.$$ << 'END1'
E$CMEXP
CTITLE = 'Try multiyear ozone dataset',
NCDATA = '/MYNAME/ccm2/T42/DEC897',
BNDT1 = '/CCM2/T42/%data%/tibds',
BNDTVS = '/CCM2/T42/%data%/tvbds',
BNDTVO = '/MYNAME/ccm2/oz8990',
IRT=10,
NSREST=0,
NSVSN='rstrt',
NSWRPS='pass',
NDENS = 1,
NNBDAT = 891230,
NNBSEC = 0,
NNDAS = 0,
NNSSAS = 0,
MFILT = 5,
DTIME = 1200.,
NELAPSE = -5,
NHTFRQ = 72,
IRAD = -1,
IRADAE = -12,
LDEBUG = .T.,
OZNCYC = .F.,
DIF2 = 2.5E5,
DIF4 = 1.E16,
KMXHDC = 1,
ITSST = 1,
NLVDRY = 3,
EPS = 0.06,

'END1'
ja
pshell ./ccm.xx.$$ < nl.dat.$$ 
if($status ! = 0) goto err
```

NCDATA  The user has created an initial dataset from December data to use in testing the new ozone dataset.
BNDTVO  This dataset must contain ozone data prior to 30 December 1989 to past 2 January 1990 so that the Model may interpolate to the requested times. If the Model reads an end-of-file on the ozone dataset, it will abort rather than rewinding to recycle the data as in a single-year dataset.

NNBDAT  NNBSEC  These variables set the base date and seconds of the date to be commensurate with dates on the ozone dataset. If not input, the base date from the initial dataset will be used.
C. Model Datasets

1. Input Datasets

CCM2 requires a number of datasets to start. These include an initial dataset containing initial values of Model-predicted variables and several boundary condition datasets for initializing various aspects of the Model parameterizations. These datasets are all in the form of CCM2 history tapes. To allow the use of the CCM Modular Processor in plotting, merging, or otherwise processing the data, these datasets are all in the form of CCM2 history tapes. For more information on the Processor, see the NCAR Technical note *Introduction to the UNICOS CCM Processor* (Buja, 1992).

(a) Initial Dataset

The initial dataset, specified by namelist input parameter NCDATA, is attached to the model job on the initial run of a case, by calling ATTACH in subroutine INITIAL. On this dataset are initial values of Model-predicted variables, the surface geopotential (PHIS) field and the orography flag (ORO). These ORO flag values should be consistent with the PHIS field, and also with the surface type mask which was used in deriving the sea-surface temperatures and time-invariant surface fields on the other boundary datasets, described on page 35. During the Model run, ORO will change only in response to changes in sea ice, which, for the standard CCM2, is specified in the time-variant SST boundary dataset. All specified boundary data are input to the Model on datasets distinct from the initial dataset and are documented in “Boundary Datasets” on page 35.

The initial dataset is a binary file in the form of a Model history tape, with a three-record header followed by plat latitude data records. See “Model Output Datasets” on page 37 for more detail on the format of the Model history tape.
The initial dataset must contain values for the fields shown in Table 1.2 below.

### Table 1.2
Initial Dataset Fields

<table>
<thead>
<tr>
<th>History Tape Field Name</th>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PLEV Level Fields</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>$T$</td>
<td>temperature (K)</td>
</tr>
<tr>
<td>U</td>
<td>$u$</td>
<td>zonal wind component ($m \cdot s^{-1}$)</td>
</tr>
<tr>
<td>V</td>
<td>$v$</td>
<td>meridional wind component ($m \cdot s^{-1}$)</td>
</tr>
<tr>
<td>Q</td>
<td>$q$</td>
<td>water vapor specific humidity ($Kg \cdot (Kg)^{-1}$)</td>
</tr>
<tr>
<td><strong>1 Level Fields</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHIS</td>
<td>$\phi_s$</td>
<td>surface geopotential ($m^2 \cdot s^{-2}$)</td>
</tr>
<tr>
<td>FS</td>
<td>$p_s$</td>
<td>surface pressure ($Pa$)</td>
</tr>
<tr>
<td>ORO</td>
<td>ORO</td>
<td>surface type flag:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>= 0 over ocean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>= 1 over land</td>
</tr>
<tr>
<td></td>
<td></td>
<td>= 2 over sea ice</td>
</tr>
<tr>
<td>TS1, TS2, TS3, TS4</td>
<td>$T_s$</td>
<td>Four subsurface temperature levels (K)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(TS1 is surface temperature)</td>
</tr>
</tbody>
</table>
The header words shown in Table 1.3 below must be given appropriate values in the initial dataset in order for the Model to run successfully. Sigma vertical coordinate, latitude, and Gaussian weight values will be computed by the Model.

Table 1.3
Required Fields for Initial Dataset Header

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Record Number</th>
<th>Word Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LENHDI</td>
<td>1</td>
<td>1</td>
<td>Length of header record 1 (integer values)*</td>
</tr>
<tr>
<td>NRBD</td>
<td>1</td>
<td>5</td>
<td>Number of records before data records</td>
</tr>
<tr>
<td>NLON</td>
<td>1</td>
<td>9</td>
<td>Number of longitude points in one latitude line*</td>
</tr>
<tr>
<td>NLONW</td>
<td>1</td>
<td>10</td>
<td>Number of longitude points written on this volume (must be equal to or greater than NLON)</td>
</tr>
<tr>
<td>NOREC</td>
<td>1</td>
<td>11</td>
<td>Number of latitude lines*</td>
</tr>
<tr>
<td>NLEV</td>
<td>1</td>
<td>12</td>
<td>Number of vertical levels*</td>
</tr>
<tr>
<td>NTRM</td>
<td>1</td>
<td>13</td>
<td>M spectral truncation parameter*</td>
</tr>
<tr>
<td>NTRN</td>
<td>1</td>
<td>14</td>
<td>N spectral truncation parameter*</td>
</tr>
<tr>
<td>NTRK</td>
<td>1</td>
<td>15</td>
<td>K spectral truncation parameter*</td>
</tr>
<tr>
<td>NFLDH</td>
<td>1</td>
<td>16</td>
<td>Number of fields on this volume</td>
</tr>
<tr>
<td>NDCUR</td>
<td>1</td>
<td>22</td>
<td>Current day number**</td>
</tr>
<tr>
<td>NSCUR</td>
<td>1</td>
<td>23</td>
<td>Seconds of the current day**</td>
</tr>
<tr>
<td>NCDATE</td>
<td>1</td>
<td>26</td>
<td>Current date**</td>
</tr>
<tr>
<td>NCSEC</td>
<td>1</td>
<td>27</td>
<td>Seconds of the current date**</td>
</tr>
<tr>
<td>LENHDC</td>
<td>1</td>
<td>31</td>
<td>Length of header record 2 (character values)</td>
</tr>
<tr>
<td>LENHDR</td>
<td>1</td>
<td>32</td>
<td>Length of header record 3 (real values)</td>
</tr>
<tr>
<td>MPFLDS</td>
<td>1</td>
<td>36</td>
<td>Pointer to integer header field list information</td>
</tr>
<tr>
<td>MCFLD</td>
<td>1</td>
<td>37</td>
<td>Pointer to character header field list information</td>
</tr>
<tr>
<td>MFLDS</td>
<td>1</td>
<td>38 - end of record</td>
<td>Integer field list information</td>
</tr>
<tr>
<td>MCFLDS</td>
<td>2</td>
<td>77 - end of record</td>
<td>Character field list information</td>
</tr>
</tbody>
</table>

* These values are used only to check against Model parameter settings. They are later overwritten in the history tape header before writing.

** These four values provide the starting time information for the model run. They may alternatively be input using namelist parameters NNDBAS, NNSBAS, NNBDAT, and NNBSEC.
All other header words are reset by the Model on an initial run. Some of these values are used by the CCM Modular Processor, so they should not be left “indefinite” if the user intends to run the Modular Processor on the initial dataset itself.

The CCM Modular Processor can create initial datasets by merging fields from one or more history tapes and building a “save history tape.” Processor input control parameters (ICPs) for making an initial dataset appear in Example 2, initial.sh, in the Processor document, *Introduction to the UNICOS CCM Processor* (Buja, 1992).

The CCM Core Group will develop a set of standard initial data files for various resolutions. The T42 datasets reside in permanent disk space on the NCAR CRAY Y-MP8/64 (shavano).

(b) **Boundary Datasets**

All CCM2 boundary datasets are in history tape format. The history tape is documented in “Model History Tape” on page 38. All time-invariant fields required as boundary conditions by the standard CCM2 are on a single dataset containing one time sample. All time-variant fields (monthly values) could theoretically be on a single dataset with multiple time samples, except that the ozone data are unique in two ways: they are defined on vertical levels that differ from those of CCM2, and the data are not longitudinally varying. Therefore, the SSTs and the ozone mixing ratios reside on two separate datasets. Because these fields are not stored on the regeneration dataset, boundary datasets must be read on a continuation run.

By formatting boundary datasets as CCM2 history tapes, we provide a standard for these and future datasets and allow the Model to use standard “building blocks,” such as subroutines RDHDR and MKSLIC, to read data from these files. In addition, the user may use the CCM Modular Processor to aid in building new boundary datasets to support his/her own model research, and analyze these datasets using the Modular Processor. Codes are maintained by the Core Group to convert CCM1 version boundary datasets to CCM2 form.

(1) **Time-Invariant Boundary Dataset**

The time-invariant surface albedo and orography standard deviation fields are on this dataset. The Mass Store pathname is specified as *namelist* input parameter BNDTI. The dataset is read by subroutine TIREAD, called from subroutine INTBND, called from the main program CCM2.
Table I.4 below lists the fields on the time-invariant boundary dataset. All fields are single-level.

### Table I.4
**Time-Invariant Boundary Dataset Fields**

<table>
<thead>
<tr>
<th>History Tape Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALBVSS</td>
<td>Albedo of visible spectrum (0.2 - 0.7 μm) over strong zenith-angle dependent surfaces</td>
</tr>
<tr>
<td>ALBVSW</td>
<td>Albedo of visible spectrum (0.2-0.7 μm) over weak zenith-angle dependent surfaces</td>
</tr>
<tr>
<td>ALBNIS</td>
<td>Albedo of near-infrared spectrum (0.7-4.0 μm) over strong zenith-angle dependent surfaces</td>
</tr>
<tr>
<td>ALBNIW</td>
<td>Albedo of near-infrared spectrum (0.7-4.0 μm) over weak zenith-angle dependent surfaces</td>
</tr>
<tr>
<td>FRCTST</td>
<td>Fraction of strong zenith-angle dependent surfaces within grid box</td>
</tr>
<tr>
<td>RGHNSS</td>
<td>Surface aerodynamic roughness over land (meters)</td>
</tr>
<tr>
<td>EVAPF</td>
<td>Surface evaporation factor (wetness factor), dimensionless</td>
</tr>
<tr>
<td>VEGTYP</td>
<td>Vegetation type, index from 1 to 7</td>
</tr>
<tr>
<td>SNWJAN</td>
<td>January snow cover, liquid water equivalent in meters</td>
</tr>
<tr>
<td>SNWJLY</td>
<td>July snow cover, liquid water equivalent in meters</td>
</tr>
<tr>
<td>SGH</td>
<td>Standard deviation of topography (meters)</td>
</tr>
</tbody>
</table>

(2) **Time-Variant Sea-Surface Temperature (SST) Dataset**

The standard SST boundary dataset contains 12 months of data, as 12 time samples, each with a three-record header and plat latitude records of the SST data.

The Model can read SST datasets containing any number of months, even multiyear datasets, as long as the times on the dataset bracket those required by the current time in the Model. If any two consecutive time samples are read which are more than 45 days apart, the Model assumes the dataset is not properly written and stops.

The Mass Store pathname of the SST dataset is specified as namelist input parameter BNDTVS. The dataset is initially positioned by subroutine SSTINI, called from subroutine INTBND. This routine reads through the data records until it has in memory the two records which bracket the Model start time. When, from within the time-stepping loop, a new time sample of SST data is required, subroutine SSTINT, called from ADVNCE, reads the next record.
(3) Time-Variant Ozone Dataset

This dataset contains ozone volume mixing ratios for an arbitrary number of vertical levels. These levels are defined by the sigma arrays contained in the real portion of the dataset header. The Model reads these values and then interpolates to the Model vertical levels at each longitude point on the grid. The standard T42 ozone dataset, on shavano permanent disk as /ccm/ccm2/T42/ozn, contains 23 levels.

The ozone dataset is in the form of a history tape containing the field OZONE. A second field, PS, is a constant, 1000 millibar (100,000 pascal) pressure field. This is provided so the user can request a pressure interpolation to produce pressure plots using the CCM Modular Processor. Because the ozone data do not vary in longitude, each of the 12 monthly time samples contains flat records of plev levels by one longitude.

The Mass Store pathname of the ozone dataset is specified as namelist input parameter BNDTVO. The dataset is read initially by subroutine OZNINI, called from subroutine INTBND, and then within the timestepping loop by subroutine OZNINT, called from ADVNCE. Ozone path lengths for use by the radiation code are computed by subroutine RADO2.

2. Model Output Datasets

The Model script runs in a subdirectory named for the case identifier, $CASE beneath the directory pointed to by system environment variable $TMPDIR (see “Model Run Script” on page 1). All subdirectories and files created by the Model under this directory are protected from the disk scrubber while the Model script is running, but they will be deleted at the time the run script terminates. For more detail about the use of the Cray disk, see “Disk File Management in CCM2” on page 91.

The output files which the user wishes to archive (history tapes, regeneration datasets, and restart dataset) are transferred to the Mass Store system by the Model as they are completed, if namelist parameters IRT and RIRT are non-zero. These transfers are done asynchronously by system routine mswrite if namelist parameter ASYNC = .TRUE. (the default value).

The Mass Store pathnames for these transfers are generated by the Model. For history tapes, the pathname is as follows:

/NEWLOG/ccm2/CASE/hist/hxxxx

Here, $NEWLOG is the upper case equivalent of the user’s Cray login name, i.e., the user’s root directory on the Mass Store System. $CASE is the case identifier as set in the Model run script (see “Running the Model” on page 1). The history tape file name, hxxxx, is assigned by the Model code,
and described in “Use of the NCAR Mass Storage System” on page 92. For regeneration and restart datasets, the following pathname is used:

$/NEWLOG/ccm2/$CASE/rest/rxxxx

The user may also specify namelist parameter LDEBUG = .TRUE. and the last set of output files will be linked to /usr/tmp/$LOGNAME, available for immediate access by, for instance, the Modular Processor. Files on /usr/tmp/$LOGNAME, however, are subject to the disk scrubber and have no guaranteed retention time.

The Model imposes a maximum on Mass Store retention times for volumes written to CTPUBLIC of 5 years, or 1825 days, and for those written to private virtual volumes of 4095 days. The NCAR Mass Storage System currently limits the size of a volume to approximately 190 megabytes. The user must choose values of namelist parameter MFILT such that individual history tapes do not grow beyond this limit. A namelist parameter, MXSZRG, is provided to limit automatically the size of regeneration datasets. The Model will split regeneration data over several volumes if necessary to satisfy a maximum file size of MXSZRG.

It is recommended that the user specify a non-blank write password, NSWRPS, as file passwords are the only form of security available on the Mass Store System.

As mentioned above, namelist parameter ASYNC allows the user to control whether output files are to be disposed to the Mass Store as a background process or not. If ASYNC = .FALSE., the Model run is suspended until the output file is actually written to the Mass Store. This was used in the early days of UNICOS and remains useful for certain other systems where background transfers are not possible for some reason. It is not a recommended standard operating procedure.

(a) Model History Tape

The Model history tape contains descriptive information and data values written at specified times during the Model run. The local dataset containing the history tape is transferred to the Mass Store Subsystem at times specified by the user. The history tape archives the Model run and may be used as input to various Model postprocessors.

A history tape is written as a series of “logical files,” each file containing gridpoint data for a given time, and consisting of three header records followed by Model field values stored in data records, one record for each latitude line. The record structure of the history tape is depicted in Figure I.5 below. The time frequency at which the files are written may be specified by namelist variable NHTFRQ, described in Table I.1 on page 14.
"File" 1

Header record 1
Header record 2
Header record 3
Latitude index
Longitude count
Data for
Field 1
...
Field nfidh
Latitude index
Longitude count
Data for
Field 1
...
Field nfidh
Remainder of plat latitude records

"Files" 2 thru (mfilt-1)

"File" mfilt

Header record 1
Header record 2
Header record 3
Latitude index
Longitude count
Data for
Field 1
...
Field nfidh

Figure 1.5. History Tape File Format.
The first three records in a logical file comprise the history tape header of Format Code 43 (header variable mftyp). These records are written from common blocks /comhdil/, /comhdc/, and /comhdr/. Using the three-record header structure, the Model splits the integer, character, and real header values into separate records, making the history tape header more easily portable to other computers. The header has three functions:

- On the initial dataset and boundary datasets, it provides some of the information required by the Model to start a case.
- On the output history tapes, it provides the information required by postprocessors to read and locate specific data fields.
- On all datasets of history tape format, it serves an archival function, storing information such as date/time and job sequence numbers concerning the Mass Store volume on which it is written.

The CCM2 history tape is a hybrid coordinate tape, with three arrays of hybrid coefficients in the real header (see Table I.5.a on page 41) along with the Gaussian latitudes and weights.

The contents of the three header records are shown in Table I.5.a through Table I.5.c. Each variable is listed by name and word number within the header record, along with a short description and a typical value based on the T42, 18-layer case. More detailed descriptions of some of the time and volume naming variables follow Table 1.5.
Table 1.5.a
History Tape File Header
Record 1—Integer Variables

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Word Number</th>
<th>Variable Type</th>
<th>Description</th>
<th>T42 Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LENHDI</td>
<td>1</td>
<td>I</td>
<td>Length of header record 1</td>
<td>235</td>
</tr>
<tr>
<td>MFTYP</td>
<td>2</td>
<td>I</td>
<td>Format code for this history tape</td>
<td>43</td>
</tr>
<tr>
<td>MFILH</td>
<td>3</td>
<td>I</td>
<td>&quot;Logical&quot; file number relative to first file on this volume</td>
<td>1</td>
</tr>
<tr>
<td>MFILTH</td>
<td>4</td>
<td>I</td>
<td>Maximum number of files on a history tape volume</td>
<td>5</td>
</tr>
<tr>
<td>NRBD</td>
<td>5</td>
<td>I</td>
<td>Number of records before data records</td>
<td>3</td>
</tr>
<tr>
<td>MAXSIZ</td>
<td>6</td>
<td>I</td>
<td>Length of data record for this volume</td>
<td>57730</td>
</tr>
<tr>
<td>NDAVU</td>
<td>7</td>
<td>I</td>
<td>Length of the data record after unpacking</td>
<td>57730</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>I</td>
<td>Unused</td>
<td>0</td>
</tr>
<tr>
<td>NLOON</td>
<td>9</td>
<td>I</td>
<td>Number of longitude points in one latitude line</td>
<td>128</td>
</tr>
<tr>
<td>NLOONW</td>
<td>10</td>
<td>I</td>
<td>Number of longitude data values written on the history tape</td>
<td>128</td>
</tr>
<tr>
<td>NOREC</td>
<td>11</td>
<td>I</td>
<td>Number of latitude lines and number of data records</td>
<td>64</td>
</tr>
<tr>
<td>NLEV</td>
<td>12</td>
<td>I</td>
<td>Number of vertical levels written on the history tape</td>
<td>18</td>
</tr>
<tr>
<td>NTRM</td>
<td>13</td>
<td>I</td>
<td>M spectral truncation parameter</td>
<td>42</td>
</tr>
<tr>
<td>NTRN</td>
<td>14</td>
<td>I</td>
<td>N spectral truncation parameter</td>
<td>42</td>
</tr>
<tr>
<td>NTRK</td>
<td>15</td>
<td>I</td>
<td>K spectral truncation parameter</td>
<td>42</td>
</tr>
<tr>
<td>NFLDH</td>
<td>16</td>
<td>I</td>
<td>Number of fields on this volume</td>
<td>60</td>
</tr>
<tr>
<td>NSTEPH</td>
<td>17</td>
<td>I</td>
<td>Iteration number at which data for this record are valid</td>
<td>0</td>
</tr>
<tr>
<td>NSTFRH</td>
<td>18</td>
<td>I</td>
<td>Iteration number for the start of this run</td>
<td>0</td>
</tr>
<tr>
<td>NITSRLF</td>
<td>19</td>
<td>I</td>
<td>Number of iterations since last file was written</td>
<td>0</td>
</tr>
<tr>
<td>NDBASE</td>
<td>20</td>
<td>I</td>
<td>Base day number for this case</td>
<td>0</td>
</tr>
<tr>
<td>NSBASE</td>
<td>21</td>
<td>I</td>
<td>Base number of seconds for this case</td>
<td>0</td>
</tr>
<tr>
<td>NDCUR</td>
<td>22</td>
<td>I</td>
<td>Current day number corresponding to NSTEPH</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 1.5.a
History Tape File Header
Record 1—Integer Variables

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Word Number</th>
<th>Variable Type</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSCUR</td>
<td>23</td>
<td>I</td>
<td>Seconds of the current day NDCUR</td>
<td>0</td>
</tr>
<tr>
<td>NBDATE</td>
<td>24</td>
<td>I</td>
<td>Base date (yr mo day) as 6-digit integer</td>
<td>000901</td>
</tr>
<tr>
<td>NBSEC</td>
<td>25</td>
<td>I</td>
<td>Seconds to complete NBDATE date. (Note: May be different from NSBASE.)</td>
<td>0</td>
</tr>
<tr>
<td>NCDATE</td>
<td>26</td>
<td>I</td>
<td>Current date (yr mo day) as 6-digit integer corresponding to NSTEPH</td>
<td>000901</td>
</tr>
<tr>
<td>NCSEC</td>
<td>27</td>
<td>I</td>
<td>Current seconds for date NCDATE</td>
<td>0</td>
</tr>
<tr>
<td>MDT</td>
<td>28</td>
<td>I</td>
<td>Model timestep in seconds</td>
<td>1200</td>
</tr>
<tr>
<td>MHISF</td>
<td>29</td>
<td>I</td>
<td>Frequency (in timesteps) history files are written</td>
<td>72</td>
</tr>
<tr>
<td>MFSTRT</td>
<td>30</td>
<td>I</td>
<td>Flag to indicate type of run:</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 0 for initial run</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 1 for restart</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 2 for regeneration</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 3 for branch run</td>
<td></td>
</tr>
<tr>
<td>LENHDC</td>
<td>31</td>
<td>I</td>
<td>Length of header record 2</td>
<td>209</td>
</tr>
<tr>
<td>LENHDR</td>
<td>32</td>
<td>I</td>
<td>Length of header record 3</td>
<td>239</td>
</tr>
<tr>
<td>MPSIG</td>
<td>33</td>
<td>I</td>
<td>Position in header record 3 of first word of sigma value list.</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>sigapb(1) = REALHD(MPSIG)</td>
<td></td>
</tr>
<tr>
<td>MPLAT</td>
<td>34</td>
<td>I</td>
<td>Position in header record 3 of list of latitude lines</td>
<td>112</td>
</tr>
<tr>
<td>MPWTS</td>
<td>35</td>
<td>I</td>
<td>Position in header record 3 of list of Gaussian weights</td>
<td>176</td>
</tr>
<tr>
<td>MPFLDS</td>
<td>36</td>
<td>I</td>
<td>Position of header field information list in header record 1 (integer values)</td>
<td>38</td>
</tr>
<tr>
<td>MPCFLD</td>
<td>37</td>
<td>I</td>
<td>Position of field information list in header record 2 (character values),</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>given as a number of 8-character items from the start of record 2</td>
<td></td>
</tr>
<tr>
<td>MFLDS</td>
<td>38</td>
<td>I</td>
<td>Integer field information, dimensioned (3, NFLDTH). See description of</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MFLDS array in Table 1.6 on page 46.</td>
<td></td>
</tr>
</tbody>
</table>
Table 1.5.b
History Tape File Header
Record 2 — Character Variables*

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Word Number</th>
<th>Format</th>
<th>Description</th>
<th>T42 Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCASE</td>
<td>1</td>
<td>C*8</td>
<td>Case identifier</td>
<td>388</td>
</tr>
<tr>
<td>MCSTIT</td>
<td>2-11</td>
<td>C*80</td>
<td>Case title</td>
<td>CASE TITLE</td>
</tr>
</tbody>
</table>

Current History Tape Volume

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Word Number</th>
<th>Format</th>
<th>Description</th>
<th>T42 Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LNHSTC</td>
<td>12-21</td>
<td>C*80</td>
<td>MSS pathname of this history volume</td>
<td>/user/ccm2/388/ hist/h0003</td>
</tr>
<tr>
<td>LDHSTC</td>
<td>22</td>
<td>C*8</td>
<td>Date this header record was written (MM/DD/YY)</td>
<td>09/09/92</td>
</tr>
<tr>
<td>LTHSTC</td>
<td>23</td>
<td>C*8</td>
<td>Time this header record was written (HH:MM:SS)</td>
<td>10:14:33</td>
</tr>
<tr>
<td>LSHSTC</td>
<td>24</td>
<td>C*8</td>
<td>Sequence number of run producing this file</td>
<td>NE3526</td>
</tr>
</tbody>
</table>

First History Tape Volume of Case

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Word Number</th>
<th>Format</th>
<th>Description</th>
<th>T42 Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LNHSTF</td>
<td>25-34</td>
<td>C*80</td>
<td>MSS pathname of first history volume for this case</td>
<td>/user/ccm2/388/ hist/h0001</td>
</tr>
<tr>
<td>LDHSTF</td>
<td>35</td>
<td>C*8</td>
<td>Date this case started (MM/DD/YY)</td>
<td>09/09/92</td>
</tr>
<tr>
<td>LTHSTF</td>
<td>36</td>
<td>C*8</td>
<td>Time this case started (HH:MM:SS)</td>
<td>10:34:05</td>
</tr>
<tr>
<td>LSHSTF</td>
<td>37</td>
<td>C*8</td>
<td>Sequence number of run that started this case</td>
<td>NE3526</td>
</tr>
</tbody>
</table>

Initial Dataset Volume

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Word Number</th>
<th>Format</th>
<th>Description</th>
<th>T42 Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LNHSTI</td>
<td>38-47</td>
<td>C*80</td>
<td>MSS pathname of initial data volume</td>
<td>/CSM/ccm2/367/ hist/h0105</td>
</tr>
<tr>
<td>LDHSTI</td>
<td>48</td>
<td>C*8</td>
<td>Date initial data volume was created (MM/DD/YY)</td>
<td>10/27/91</td>
</tr>
<tr>
<td>LTHSTI</td>
<td>49</td>
<td>C*8</td>
<td>Time initial data volume created (HH:MM:SS)</td>
<td>14:26:43</td>
</tr>
<tr>
<td>LSHSTI</td>
<td>50</td>
<td>C*8</td>
<td>Sequence number of run creating initial data</td>
<td>DUMSEQ</td>
</tr>
</tbody>
</table>

*All character strings are left-justified, blank-filled, and multiples of 8 characters.
### Table 1.5.b
History Tape File Header
Record 2 — Character Variables

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Word Number</th>
<th>Format</th>
<th>Description</th>
<th>T42 Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time-Invariant Boundary Dataset Volume</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LNHSTT</td>
<td>51-60</td>
<td>C*80</td>
<td>MSS pathname of time-invariant boundary dataset</td>
<td>/CCM2/T42/tibds</td>
</tr>
<tr>
<td>LDHSTT</td>
<td>61</td>
<td>C*8</td>
<td>Date last header record was written on time-invariant boundary dataset, as MM/DD/YY</td>
<td>12/27/91</td>
</tr>
<tr>
<td>LTHSTT</td>
<td>62</td>
<td>C*8</td>
<td>Time last header record was written on time-invariant boundary volume, as HH:MM:SS</td>
<td>09:12:22</td>
</tr>
<tr>
<td>LSHSTT</td>
<td>63</td>
<td>C*8</td>
<td>Sequence number of run writing time-invariant boundary volume</td>
<td>-1</td>
</tr>
<tr>
<td><strong>Time-variant SST Dataset Volume</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LNHSTVS</td>
<td>64-73</td>
<td>C*80</td>
<td>MSS pathname of SST boundary dataset</td>
<td>/CCM2/T42/tbds</td>
</tr>
<tr>
<td>LDHSTVS</td>
<td>74</td>
<td>C*8</td>
<td>Date SST boundary data volume was created (MM/DD/YY)</td>
<td>12/23/91</td>
</tr>
<tr>
<td>LTHSTVS</td>
<td>75</td>
<td>C*8</td>
<td>Time SST boundary data volume was created (HH:MM:SS)</td>
<td>15:01:43</td>
</tr>
<tr>
<td>LSHSTVS</td>
<td>76</td>
<td>C*8</td>
<td>Sequence number of run creating SST boundary data</td>
<td>-1</td>
</tr>
<tr>
<td><strong>Ozone Dataset Volume</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LNHSTVO</td>
<td>77-86</td>
<td>C*80</td>
<td>MSS pathname of ozone boundary dataset</td>
<td>/CCM2/T42/ozn</td>
</tr>
<tr>
<td>LDHSTVO</td>
<td>87</td>
<td>C*8</td>
<td>Date ozone boundary dataset was created (MM/DD/YY)</td>
<td>12/23/91</td>
</tr>
<tr>
<td>LTHSTVO</td>
<td>88</td>
<td>C*8</td>
<td>Time ozone boundary dataset was created (HH:MM:SS)</td>
<td>15:05:42</td>
</tr>
<tr>
<td>LSHSTVO</td>
<td>89</td>
<td>C*8</td>
<td>Sequence number of run creating ozone boundary data</td>
<td>-1</td>
</tr>
</tbody>
</table>
### Table I.5.b

**History Tape File Header**

Record 2 — Character Variables

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Word Number</th>
<th>Format</th>
<th>Description</th>
<th>T42 Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCFLDS</td>
<td>90-195</td>
<td>C*8</td>
<td>Character field information, dimensioned (2, NFLDH). See description of MCFLDS array in Table I.6 on page 46.</td>
<td>--</td>
</tr>
</tbody>
</table>

---

### Table I.5.c

**History Tape File Header**

Record 3 — Real Variables

<table>
<thead>
<tr>
<th>Variable Name*</th>
<th>Word Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALHD (MPSIG)</td>
<td></td>
<td>Contains 3 arrays of (2 * NLEV +1) hybrid coefficients as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sigapb = hybrid A+B coefficients</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sigma = hybrid A (pressure) coefficients</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sigb = hybrid B (sigma) coefficients</td>
</tr>
<tr>
<td>REALHD (MPLAT)</td>
<td></td>
<td>Points to the first word, relative to the start of the header, of a list of NOREC Gaussian latitudes in degrees (south to north)</td>
</tr>
<tr>
<td>REALHD (MPWTS)</td>
<td></td>
<td>Points to the first word of a list of NOREC Gaussian weights corresponding to Gaussian latitudes</td>
</tr>
</tbody>
</table>

*MPSIG, MPLAT, and MPWTS are from header record 1, words 33, 34, and 35, respectively.

LENHDI (MPFLDS) is the first word of the field information list described by the two-dimensional array MFLDS (3, NFLDH), detailed in Table I.6 on page 46. Character data in the field information list are contained in the two-dimensional array MCFLDS (2, NFLDH), also detailed in Table I.6.
Table 1.6
Description of Field Information List Arrays, MFLDS and MCFLDS

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFLDS (1, IF)</td>
<td>Flag for number of levels and field flag, packed in one integer word. The ones digit is for the level number flag; the tens digit states whether the field is an instantaneous value, a value averaged between history tape writes, a time minimum between writes, or a time maximum between writes.</td>
</tr>
</tbody>
</table>
|               | If the ones digit of MFLDS (1, IF)  
|               | = 0 it is a single-level field,  
|               | = 1 it is a multilevel field at interfaces,  
|               | = 2 it is a multilevel field at levels.  
|               | If the tens digit  
|               | = 0 it is an instantaneous field,  
|               | = 1 it is an averaged field.  
|               | = 2 it is the minimum of the values for each gridpoint since the last history tape write.  
|               | = 3 it is the maximum of the values for each gridpoint since the last history tape write. |
| MFLDS (2, IF) | Pointer to the first field value in packed data record. |
| MFLDS (3, IF) | Data-packing flag (set to 1 if not packed). May differ for different fields. |
| MCFLDS (1, IF) | Field names for field number IF |
| MCFLDS (2, IF) | Units of this field (SI units). See Table 1.7 on page 47. |
### Table 1.7

#### SI Units

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fundamental Units</strong></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>meter</td>
</tr>
<tr>
<td>KG</td>
<td>kilogram</td>
</tr>
<tr>
<td>S</td>
<td>second</td>
</tr>
<tr>
<td>K</td>
<td>degree Kelvin</td>
</tr>
<tr>
<td>MOL</td>
<td>Mole</td>
</tr>
<tr>
<td><strong>Combination Units</strong></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>newton</td>
</tr>
<tr>
<td>PA</td>
<td>pascal</td>
</tr>
<tr>
<td>J</td>
<td>joule</td>
</tr>
<tr>
<td>W</td>
<td>watt</td>
</tr>
</tbody>
</table>

Combination units come first. Within the fundamental units, the ordering is M, KG, S, K, MOL. All units in the numerator come first, followed by a slash, followed by all units in the denominator, e.g., M/S is meters per second. Exponents are indicated by an unsigned integer following the abbreviation — for example, M2 for meter squared and S2 for second squared.

Dimensionless fields are 'FRACTION' or 'FLAG'.

Here is a further explanation of some header variables:

- **LENHDI**: Exact length of the first header record in Cray words, which varies depending on the length of the list pointed to by MPFLDS. This variable is declared as a one-dimensional array to facilitate array I/O of the first header record.

- **LENHDC**: Exact length of the second header record in Cray words, which varies depending on the length of the list pointed to by MPCFLD.

- **LENHDR**: Exact length of the third header record in Cray words, which varies depending on the length of the lists pointed to by MPSIG, MPLAT, and MPWTS.

- **MFTYP**: History tape format identifier. When new formats are generated, this flag is changed to provide format information for postprocessors.
The number of longitudinal gridpoints in the Model.

The number of longitude values written on the history tape for each latitude line. Currently, this must equal NLON.

Frequency in timesteps of history tape write, set from namelist variable NHTFRQ.

If the history tape is packed (NDENS>1), this pointer will reflect the position of field IF in the record before it is unpacked. This allows postprocessing programs to process only the desired fields from the data record.

All words in the latitude record are floating point. If the Model is run multitasked, the order of the latitude records on the CCM2 history tape is indeterminate. If it is run single-threaded, the order is by latitude pairs, starting with the northern-most point, then the southern-most, then the next northern-most, the next southern-most, etc.

The first word of the record is a latitude index, counting from south to north.

The second word of the record is a count of the number of longitudes contained in that record. Currently, this number is always float (plon) (the latitude record contains only floating point values) but allows for the possible future implementation of a variable grid.

Field information starts in word 3 of the record. The pointers contained in the header array MFLDS point to the beginning word of data for each field, packed or unpacked. Thus, it is possible to read in a packed record and unpack only the desired fields.

In Figure I.6 below, we illustrate history tape latitude record number 5 (the fifth latitude from the southern-most) for T42 horizontal resolution. This figure shows how the pointers in MFLDS relate to the field data contained in the record.
In Table I.8 below, we describe the fields that are written to the default primary history tape. The column labelled “Buffer pointer or location” contains information about where the field is stored. Pointers refer to locations in the main Model buffer, documented in Figure II.2 on page 81. Local arrays are accompanied by the subroutine in which they are declared, e.g., “ta (LINEMS).”

In the column labelled “NL,” a 1 indicates a single-level field, and an N indicates a multilevel field. The column labelled “A/I” indicates if the field is averaged (A) in time over the timesteps since the last history tape write, or contains an instantaneous (I) value.

Field symbols shown are those used in the report *Description of the NCAR Community Climate Model (CCM2)* (Hack *et al.*, 1992). Reference this report for more information about CCM2 history tape fields.
<table>
<thead>
<tr>
<th>Field No.</th>
<th>Field Name</th>
<th>Symbol</th>
<th>Location</th>
<th>Field Description</th>
<th>NL</th>
<th>A/I</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PHIS</td>
<td>NPHIS</td>
<td></td>
<td>surface geopotential</td>
<td>1</td>
<td>I</td>
<td>$m^2 \cdot s^{-2}$</td>
</tr>
<tr>
<td>2</td>
<td>PS</td>
<td>P</td>
<td>NPSM1</td>
<td>surface pressure</td>
<td>1</td>
<td>A</td>
<td>Pa</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>T</td>
<td>/com3d/t3</td>
<td>temperature</td>
<td>N</td>
<td>A</td>
<td>K</td>
</tr>
<tr>
<td>4</td>
<td>U</td>
<td>u</td>
<td>/com3d/u3</td>
<td>zonal wind component</td>
<td>N</td>
<td>A</td>
<td>$m \cdot s^{-1}$</td>
</tr>
<tr>
<td>5</td>
<td>V</td>
<td>v</td>
<td>/com3d/v3</td>
<td>meridional wind component</td>
<td>N</td>
<td>A</td>
<td>$m \cdot s^{-1}$</td>
</tr>
<tr>
<td>6</td>
<td>q (or \ TRxx)</td>
<td>q</td>
<td>/com3d/q3</td>
<td>specific humidity</td>
<td>N</td>
<td>A</td>
<td>$Kg \cdot Kg^{-1} \cdot s^{-1}$</td>
</tr>
<tr>
<td>7</td>
<td>TA01</td>
<td>$dq \over dt$</td>
<td>ta</td>
<td>total advection of q</td>
<td>N</td>
<td>A</td>
<td>$Kg \cdot Kg^{-1} \cdot s^{-1}$</td>
</tr>
<tr>
<td>8</td>
<td>VD01</td>
<td>dqv</td>
<td>(VDINTR)</td>
<td>q vertical diffusion tendency</td>
<td>N</td>
<td>A</td>
<td>$Kg \cdot Kg^{-1} \cdot s^{-1}$</td>
</tr>
<tr>
<td>9</td>
<td>DC01</td>
<td>$\delta q \over \delta t$</td>
<td>dqcond</td>
<td>q tendency from adjustment physics</td>
<td>N</td>
<td>A</td>
<td>$Kg \cdot Kg^{-1} \cdot s^{-1}$</td>
</tr>
<tr>
<td>10</td>
<td>DTH</td>
<td>$F_{TS}(T^*)$</td>
<td>NDTHP1</td>
<td>T horizontal diffusion tendency</td>
<td>N</td>
<td>A</td>
<td>$K \cdot s^{-1}$</td>
</tr>
</tbody>
</table>
| 11       | ORO        | NORO |           | surface type flag:  
= 0. for ocean  
= 1. for land  
= 2. for sea ice | 1  | I   | flag |
| 12       | WET        | W  | NWS      | soil moisture | 1  | A   | m     |
| 13       | SNOWNH     | S_n | NSN      | water equivalent snow depth | 1  | A   | m     |
| 14       | PRECL      | P_{ls} | precl | large-scale, stable precipitation | 1  | A | $m \cdot s^{-1}$ |
| 15       | PRECC      | P_{cs} | precc | convective precipitation | 1  | A | $m \cdot s^{-1}$ |
| 16       | SHFLX      | $c_p \rho_l (w'\theta)'_o$ | shflx | surface sensible heat flux | 1  | A | $W \cdot m^{-2}$ |
| 17       | LHFLX      | $L \rho_l (w'q)'_o$ | lhflx | surface latent heat flux | 1  | A | $W \cdot m^{-2}$ |
| 18       | QFLX       | $\rho_l (w'q)'_o$ | cflx | surface water flux | 1  | A | $Kg \cdot m^{-2} \cdot s$ |
| 19       | PBLH       | h  | NPBLHT   | planetary or atmosphere boundary layer (PBL) height | 1  | A | m |
| 20       | USTAR      | u* | ustar | surface friction velocity | 1  | A | $m \cdot s^{-1}$ |

1 Written to history tape from LINEMS local array pstarm.
<table>
<thead>
<tr>
<th>Field No.</th>
<th>Field Name</th>
<th>Symbol</th>
<th>Buffer Pointer or Location</th>
<th>Field Description</th>
<th>NL</th>
<th>A/I</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>TPERT</td>
<td>$\theta_o$</td>
<td>NTPERT</td>
<td>PBL plume temperature perturbation</td>
<td>1</td>
<td>A</td>
<td>$K$</td>
</tr>
<tr>
<td>22</td>
<td>QPERT</td>
<td>$q'_o$</td>
<td>NQPERT</td>
<td>PBL plume moisture perturbation</td>
<td>1</td>
<td>A</td>
<td>$Kg \cdot Kg^{-1}$</td>
</tr>
<tr>
<td>23</td>
<td>DTV</td>
<td>dtv</td>
<td>PHYS</td>
<td>T vertical diffusion tendency</td>
<td>N</td>
<td>A</td>
<td>$K \cdot s^{-1}$</td>
</tr>
<tr>
<td>24</td>
<td>FSNS</td>
<td>$F_N^S(P_T)$</td>
<td>fsns (PHYS)</td>
<td>net downward solar flux at surface</td>
<td>1</td>
<td>A</td>
<td>$W \cdot m^{-2}$</td>
</tr>
<tr>
<td>25</td>
<td>FLNS</td>
<td>$F_N^L(P_T)$</td>
<td>flns (PHYS)</td>
<td>net upward longwave flux at surface</td>
<td>1</td>
<td>A</td>
<td>$W \cdot m^{-2}$</td>
</tr>
<tr>
<td>26</td>
<td>FLNT</td>
<td>$F_N^S(P_T)$</td>
<td>flnt (PHYS)</td>
<td>net upward longwave flux at top</td>
<td>1</td>
<td>A</td>
<td>$W \cdot m^{-2}$</td>
</tr>
<tr>
<td>27</td>
<td>FSNT</td>
<td>$F_N^S(P_T)$</td>
<td>fsnt (PHYS)</td>
<td>net downward solar flux at top</td>
<td>1</td>
<td>A</td>
<td>$W \cdot m^{-2}$</td>
</tr>
<tr>
<td>28</td>
<td>CLOUD</td>
<td>$A_c$</td>
<td>PHYS</td>
<td>cloud fraction</td>
<td>N</td>
<td>A</td>
<td>fraction</td>
</tr>
<tr>
<td>29</td>
<td>EFFCLD</td>
<td>$\varepsilon A_c$</td>
<td>effclld (PHYS)</td>
<td>effective cloud fraction</td>
<td>N</td>
<td>A</td>
<td>fraction</td>
</tr>
<tr>
<td>30</td>
<td>FLNTC</td>
<td>$F_N^L(P_T)_{clr}$</td>
<td>flntc (PHYS)</td>
<td>net clear-sky upward longwave flux at top</td>
<td>1</td>
<td>A</td>
<td>$W \cdot m^{-2}$</td>
</tr>
<tr>
<td>31</td>
<td>FSNTC</td>
<td>$F_N^S(P_T)_{clr}$</td>
<td>fsntc (PHYS)</td>
<td>net clear-sky downward solar flux at top</td>
<td>1</td>
<td>A</td>
<td>$W \cdot m^{-2}$</td>
</tr>
<tr>
<td>32</td>
<td>FLNSC</td>
<td>$F_N^L(P_S)_{clr}$</td>
<td>flns (PHYS)</td>
<td>net clear-sky upward longwave flux at surface</td>
<td>1</td>
<td>A</td>
<td>$W \cdot m^{-2}$</td>
</tr>
<tr>
<td>33</td>
<td>FSNSC</td>
<td>$F_N^S(P_S)_{clr}$</td>
<td>fsnsac (PHYS)</td>
<td>net clear-sky downward solar flux at surface</td>
<td>1</td>
<td>A</td>
<td>$W \cdot m^{-2}$</td>
</tr>
<tr>
<td>34</td>
<td>OMEGA</td>
<td>$\omega$</td>
<td>omga (LINEMS)</td>
<td>vertical pressure velocity</td>
<td>N</td>
<td>A</td>
<td>$Pa \cdot s^{-1}$</td>
</tr>
<tr>
<td>35</td>
<td>DQP</td>
<td>$R_{es} + R_{ls}$</td>
<td>qc (LINEMS)</td>
<td>Q tendency from rainout</td>
<td>N</td>
<td>A</td>
<td>$Kg \cdot Kg^{-1} \cdot s^{-1}$</td>
</tr>
<tr>
<td>36</td>
<td>TAUX</td>
<td>$\tau_\lambda$</td>
<td>taux (PHYS)</td>
<td>zonal surface stress</td>
<td>1</td>
<td>A</td>
<td>$N \cdot m^{-2}$</td>
</tr>
<tr>
<td>37</td>
<td>TAUY</td>
<td>$\tau_\mu$</td>
<td>tauy (PHYS)</td>
<td>meridional surface stress</td>
<td>1</td>
<td>A</td>
<td>$N \cdot m^{-2}$</td>
</tr>
<tr>
<td>38</td>
<td>SRFRAD</td>
<td>$F_N^S + F_{\downarrow}(P_S)$</td>
<td>NDRP1</td>
<td>radiative flux absorbed at the surface</td>
<td>1</td>
<td>A</td>
<td>$W \cdot m^{-2}$</td>
</tr>
<tr>
<td>39</td>
<td>QRS</td>
<td>$Q_s$</td>
<td>NQRS</td>
<td>solar heating rate</td>
<td>N</td>
<td>I</td>
<td>$K \cdot s^{-1}$</td>
</tr>
<tr>
<td>40</td>
<td>QRL</td>
<td>$Q_{iw}$</td>
<td>NQRL</td>
<td>longwave heating rate</td>
<td>N</td>
<td>I</td>
<td>$K \cdot s^{-1}$</td>
</tr>
<tr>
<td>Field No.</td>
<td>Field Name</td>
<td>Symbol</td>
<td>Buffer Pointer or Location</td>
<td>Field Description</td>
<td>NL</td>
<td>A/I</td>
<td>Units</td>
</tr>
<tr>
<td>----------</td>
<td>--------------</td>
<td>--------</td>
<td>----------------------------</td>
<td>------------------------------------------------</td>
<td>----</td>
<td>-----</td>
<td>-------------</td>
</tr>
<tr>
<td>41</td>
<td>CLDTOT</td>
<td>$A_c^T$</td>
<td>cltot (PHYS)</td>
<td>total cloud cover (random overlap)</td>
<td>1</td>
<td>A</td>
<td>fraction</td>
</tr>
<tr>
<td>42</td>
<td>CLDLOW</td>
<td>$A_c^L$</td>
<td>cllow (PHYS)</td>
<td>low cloud cover (random overlap)</td>
<td>1</td>
<td>A</td>
<td>fraction</td>
</tr>
<tr>
<td>43</td>
<td>CLDMED</td>
<td>$A_c^M$</td>
<td>clmed (PHYS)</td>
<td>medium cloud cover (random overlap)</td>
<td>1</td>
<td>A</td>
<td>fraction</td>
</tr>
<tr>
<td>44</td>
<td>CLDHGH</td>
<td>$A_c^H$</td>
<td>clgh (PHYS)</td>
<td>high cloud cover (random overlap)</td>
<td>1</td>
<td>A</td>
<td>fraction</td>
</tr>
<tr>
<td>45</td>
<td>TS1</td>
<td>$T_s$</td>
<td>NTSSUB</td>
<td>surface temperature (level 1)</td>
<td>1</td>
<td>A</td>
<td>K</td>
</tr>
<tr>
<td>46</td>
<td>TS2</td>
<td>$T_2$</td>
<td>NTSSUB</td>
<td>subsurface temperature (level 2)</td>
<td>1</td>
<td>A</td>
<td>K</td>
</tr>
<tr>
<td>47</td>
<td>TS3</td>
<td>$T_3$</td>
<td>NTSSUB</td>
<td>subsurface temperature (level 3)</td>
<td>1</td>
<td>A</td>
<td>K</td>
</tr>
<tr>
<td>48</td>
<td>TS4</td>
<td>$T_4$</td>
<td>NTSSUB</td>
<td>subsurface temperature (level 4)</td>
<td>1</td>
<td>A</td>
<td>K</td>
</tr>
<tr>
<td>49</td>
<td>SOLIN</td>
<td>$S_I$</td>
<td>solin (PHYS)</td>
<td>solar insolation</td>
<td>1</td>
<td>A</td>
<td>W·m⁻²</td>
</tr>
<tr>
<td>50</td>
<td>UTGW</td>
<td>$\frac{\delta u}{\delta t}_{GW}$</td>
<td>utgw (GWINTR)</td>
<td>gravity wave drag u tendency</td>
<td>N</td>
<td>A</td>
<td>m·s⁻²</td>
</tr>
<tr>
<td>51</td>
<td>VTGW</td>
<td>$\frac{\delta v}{\delta t}_{GW}$</td>
<td>vtw (GWINTR)</td>
<td>gravity wave drag v tendency</td>
<td>N</td>
<td>A</td>
<td>m·s⁻²</td>
</tr>
<tr>
<td>52</td>
<td>TAUGWX</td>
<td>$(\tau_x)_{GW}$</td>
<td>taugx (GWINTR)</td>
<td>gravity wave drag zonal surface stress</td>
<td>1</td>
<td>A</td>
<td>N·m⁻²</td>
</tr>
<tr>
<td>53</td>
<td>TAUGWY</td>
<td>$(\tau_y)_{GW}$</td>
<td>taugy (GWINTR)</td>
<td>gravity wave drag meridional surface stress</td>
<td>1</td>
<td>A</td>
<td>N·m⁻²</td>
</tr>
<tr>
<td>54</td>
<td>DTCOND</td>
<td>$\frac{\delta T}{\delta t}_A$</td>
<td>dtcond (LINEMS)</td>
<td>T tendency from adjustment physics</td>
<td>N</td>
<td>A</td>
<td>K·s⁻¹</td>
</tr>
<tr>
<td>55</td>
<td>CMFDT</td>
<td>$\frac{\delta T}{\delta t}_{cc}$</td>
<td>cmfdt (LINEMS)</td>
<td>T tendency from moist convection</td>
<td>N</td>
<td>A</td>
<td>K·s⁻¹</td>
</tr>
<tr>
<td>56</td>
<td>CMFDQ</td>
<td>$\frac{\delta q}{\delta t}_{cc}$</td>
<td>cmfdq (LINEMS)</td>
<td>Q tendency from moist convection</td>
<td>N</td>
<td>A</td>
<td>Kg·Kg⁻¹·s⁻¹</td>
</tr>
<tr>
<td>57</td>
<td>CMFMC</td>
<td>$M_c$</td>
<td>cmfmc (LINEMS)</td>
<td>total convective mass flux</td>
<td>N</td>
<td>A</td>
<td>Kg$m^{-2}$·s⁻¹</td>
</tr>
<tr>
<td>58</td>
<td>CMFSL</td>
<td>$F_{s-Ll}$</td>
<td>cmfsl (LINEMS)</td>
<td>convective liquid water static energy flux</td>
<td>N</td>
<td>A</td>
<td>W·m⁻²</td>
</tr>
<tr>
<td>59</td>
<td>CMFLQ</td>
<td>$F_{q+I}$</td>
<td>cmflq (LINEMS)</td>
<td>convective total water energy flux</td>
<td>N</td>
<td>A</td>
<td>W·m⁻²</td>
</tr>
<tr>
<td>60</td>
<td>CNVCLD</td>
<td>$A_{cc}$</td>
<td>clc (PHYS)</td>
<td>convective cloud fraction</td>
<td>1</td>
<td>A</td>
<td>fraction</td>
</tr>
</tbody>
</table>
(b) Restart/Regeneration Datasets

The Model periodically saves information required to restart the case and writes it to separate files known as regeneration datasets. Variables from common blocks /com3d/, /comslt/, /comtim/, /combst/, /comqfl/, and /crdsrf/ are written to the "master" regeneration dataset. The "primary" regeneration dataset includes the data on the work units nrbl and nral, plus the three-dimensional arrays. If needed, the absorptivities and emissivities in /crdalb/ are written to a "secondary" regeneration dataset. (See the discussion of IRADA in "Table of Input Parameters" on page 13.)

Each set of regeneration files is associated with a history file, reflected by the naming conventions. Thus, regeneration datasets 'r0003', 'r0003.A', and 'r0003.a' were written by the Model at the time that history tape 'h0003' was disposed to the Mass Store Subsystem. These datasets provide a checkpoint for the Model at the point in the simulation when the last file on h0003 was written. If the system fails or the Model stops for some other reason before another history tape is disposed, the regeneration data allows the user to start the Model up from the end of h0003, with a minimum of computer resources lost in continuing the case.

The "restart dataset" (namelist parameter NSVSN) is merely a text file containing the Mass Store pathname of the most recently written master regeneration dataset. Thus, for a "restart run" (namelist parameter NSREST=1) the Model can read the restart dataset, acquire the appropriate regeneration files, and continue from the end of the previous run.

The master regeneration dataset, dataset 'r0003' in the example above, contains the following information:
The primary regeneration dataset ('r0003.A' in the above example) contains data from the Model buffer and the 3-dimensional arrays, allowing the continuation processing to set all variables as they were at the time the restart datasets were written. For the standard Model, all of these data will fit on a single volume (approximately 45 megabytes). However, for higher-resolution runs, this may not be true. The Model will calculate the expected length of the dataset and split it, writing latitudes across separate files if necessary. The second primary dataset would be called r0003.B, and the third r0003.C.

<table>
<thead>
<tr>
<th>Record Number</th>
<th>Common Block</th>
<th>Variable Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>/comhst/</td>
<td>mtapes, nwtime, nlfilt, mcfldsi, nupnt, npnt</td>
</tr>
<tr>
<td>2</td>
<td>/comhst/</td>
<td>nhfrq, ndens, nflds, nfils, hbufpt</td>
</tr>
<tr>
<td>3</td>
<td>/comhst/</td>
<td>nrlen, nplen, mfilt</td>
</tr>
<tr>
<td>4</td>
<td>/comhed/</td>
<td>caseid</td>
</tr>
<tr>
<td>5</td>
<td>/comhst/</td>
<td>nfpath, cpath, ppath, nhfil</td>
</tr>
<tr>
<td>6</td>
<td>/comtim/</td>
<td>ndcurf, nsurf, ndatf, ncsecf</td>
</tr>
<tr>
<td>7</td>
<td>/comhed/</td>
<td>lnhsti, ldhsti, lthsti, lshsti</td>
</tr>
<tr>
<td>8</td>
<td>/comtim/</td>
<td>nbdate, nbsec, ndbase, nsbase</td>
</tr>
<tr>
<td>9</td>
<td>/comhed/</td>
<td>mpsig, siga, sigb</td>
</tr>
<tr>
<td>10</td>
<td>/comss/</td>
<td>nrfil</td>
</tr>
<tr>
<td>11</td>
<td>/comqfl/</td>
<td>tqf, dqf, fixmas</td>
</tr>
<tr>
<td>12</td>
<td>/crdalb/</td>
<td>rghnss, snwjan, snwjly, evapf, vevapf, vegtyp</td>
</tr>
<tr>
<td>13</td>
<td>/comlun/</td>
<td>rg1lat, rg1siz, rg1buf, nmr1, rg2lat, rg2siz, rg2buf, nmr2, mxxszrg, rg1ext, rg2ext</td>
</tr>
</tbody>
</table>
### Table I.10
**Primary Regeneration Dataset**

<table>
<thead>
<tr>
<th>Record Number</th>
<th>Common Block</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>b1 in LINDRV</td>
<td>the “b” buffer (work unit nrbl)</td>
</tr>
<tr>
<td>2</td>
<td>b2 in LINDRV</td>
<td>the “a” buffer (work unit nra1)</td>
</tr>
<tr>
<td>3</td>
<td>/com3d/</td>
<td>time n 3-D arrays u3, v3, t3, q3</td>
</tr>
<tr>
<td>4</td>
<td>/com3d/</td>
<td>time n-1 3-D arrays u3, v3, t3, q3</td>
</tr>
<tr>
<td>5</td>
<td>/comslt/</td>
<td>slt array lammp</td>
</tr>
<tr>
<td>6</td>
<td>/comslt/</td>
<td>slt array phimp</td>
</tr>
<tr>
<td>7</td>
<td>/comslt/</td>
<td>slt array sigmp</td>
</tr>
<tr>
<td>8</td>
<td>/comslt/</td>
<td>slt array hqfcst for the first constituent</td>
</tr>
<tr>
<td>9</td>
<td>/comslt/</td>
<td>slt array qfcst for the first constituent. These records are followed by hqfcst and qfcst for the remaining constituents, if specified (pcnst&gt;1).</td>
</tr>
</tbody>
</table>

Records 1 through 9 are repeated for the number of latitudes on the dataset.

| 9*plat+1      | /com3d/       | time n array ps                                                          |
| 9*plat+2      | /com3d/       | time n-1 array ps                                                        |
| final record of primary dataset | /comslt/ | slt variables hw1, hw2, hw3, alpha                                       |

The secondary regeneration dataset is required only if the history tape write frequency does not evenly divide the absorptivity/emissivity interval IRADAE. In this case, absorptivities/emissivities would not necessarily be calculated on the first timestep after restart and so must be saved on a regeneration dataset. It is advisable to avoid this situation if possible, since this dataset, even for the standard T42 Model, is relatively large. On high-resolution runs, the Model will split the secondary regeneration dataset over latitudes if necessary.
The secondary regeneration dataset consists of plat records containing
the following three arrays:

Table I.11
Secondary Regeneration Dataset

<table>
<thead>
<tr>
<th>Array</th>
<th>Location</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>abstot (plond, plevp, plevp)</td>
<td>local to RADCLW</td>
<td>total absorptivity</td>
</tr>
<tr>
<td>absnxt (plond, plev, 4)</td>
<td>local to RADCLW</td>
<td>total nearest layer absorptivity (each layer is subdivided by 2 and absorptivities are computed in 2 directions)</td>
</tr>
<tr>
<td>emstot (plond, plevp)</td>
<td>local to RADCLW</td>
<td>total emissivity</td>
</tr>
</tbody>
</table>
D. Model Printed Output

This section documents the printed output that may result from running CCM2. The standard printout from a successful CCM2 run is documented in “Printout from a CCM2 Run” below. Here, selected portions of the printout are reproduced and explained in detail. “Model Error Messages” on page 61 lists all CCM2 error messages that may originate from the CCM2 code exactly as they appear in the printed output, followed by an explanation of the message.

1. Printout from a CCM2 Run

The standard printed output from the Model is largely self-explanatory. However, some aspects are not self-explanatory, but are important for the user to understand. We will use the output generated by a successful one-day simulation as run on the NCAR Cray to illustrate the Model printout. The entire printout appears in “Appendix B: CCM2 Printed Output,” while selected portions are explained in this section.

Your printout will echo certain commands from the run script. A `segldr` statistical summary is requested by this script and will appear at the beginning of the printout. The first output from the Model itself declares it as NCAR CCM2 and documents the starting date and time of the run:

```
NCAR Community Climate Model, Version 2.0
Copyright (C) 1992
University Corporation for Atmospheric Research
All Rights Reserved

DATE 08/13/92   TIME 14:08:17
```

Note that CCM2 is copyrighted by the University Corporation for Atmospheric Research, as mentioned in “Running the Model” on page 1.

Because of the “E” in column one of the `namelist` group designator `CCMEXP`, all `namelist` records are echoed to the output file. This is followed by a summary of `namelist` variables, with explanations.

Next you will see several `assign` commands as the Model is setting up output files and attaching the input datasets for reading. Then the hybrid coordinate layer locations and reference pressures are printed from subroutine `HYCOEF`.

Information from the boundary dataset headers is printed as these files are attached and positioned to the appropriate date.
Printouts from subroutine PREALC indicate how space is preallocated for output disk files. SSD files are opened at two places in the Model, but following the last of these, the SSD requirement is printed:

```
Opened save file unit as direct access unit 22 size: 3702784 words
Total size of files normally assigned to SDS: 11206656 words 12 Megawords.
```

followed by a table, “Summary of Logical Unit Assignments,” that lists the logical units used by the Model. This unit assignment is always the same for the standard Model. However, it can change if changes to the Model require additional units, for instance (see “Disk File Management in CCM2” on page 91). Therefore, it is important for the user to refer to this table in identifying unit numbers from system error messages.

When the Model enters the time loop, a printed heading defines the standard print for each time step, as follows:

```
<table>
<thead>
<tr>
<th>NSTEP</th>
<th>RMSZ</th>
<th>RMSD</th>
<th>RMST</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8.833474687402251E-05</td>
<td>7.338332180666193E-06</td>
<td>252.716</td>
</tr>
<tr>
<td>STPS</td>
<td>STQ</td>
<td>COURANT</td>
<td>HOR</td>
</tr>
<tr>
<td>9.84934E+04</td>
<td>2.767275591143027E+01</td>
<td>0.81</td>
<td>0.24</td>
</tr>
</tbody>
</table>
```

where:

- RMSZ is the Global RMS Vorticity
- RMSD is the Global RMS Divergence
- RMST is the Global RMS Temperature
- STPS is the Global Mass Integral
- STQ is the Global Moisture Integral
- COURANT HOR is the Maximum Courant Number for Horizontal Velocity Field
- COURANT VERT is the Maximum Courant Number for Vertical Velocity Field

The first time a history tape header is written for a particular tape (primary or auxiliary), the entire header is printed as follows:
### **HEADER FOR CCM2 HISTORY TAPE**

#### **Primary History Tape**

**CASE:** merg202  
**TITLE:** plx20 merge

<table>
<thead>
<tr>
<th>LENHDI</th>
<th>MFTYP</th>
<th>MFILH</th>
<th>MFILTH</th>
<th>NRD</th>
<th>MAXSIZ</th>
</tr>
</thead>
<tbody>
<tr>
<td>217</td>
<td>43</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>57730</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PLAT</th>
<th>PLEV</th>
<th>PTRM</th>
<th>PTRN</th>
<th>PTRK</th>
<th>NFLDH</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>18</td>
<td>42</td>
<td>42</td>
<td>42</td>
<td>60</td>
</tr>
</tbody>
</table>

#### FIELD LIST

<table>
<thead>
<tr>
<th>FLD NO.</th>
<th>NAME</th>
<th>FLD PT.</th>
<th>PACK.</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PHIS</td>
<td>0</td>
<td>3</td>
<td>M2/S2</td>
</tr>
<tr>
<td>2</td>
<td>PS</td>
<td>10</td>
<td>131</td>
<td>PA</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>12</td>
<td>259</td>
<td>K</td>
</tr>
</tbody>
</table>

#### SIGMA VALUES AT FULL LEVELS

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4.80930E-03</td>
<td>1.30731E00</td>
<td>0.00000E+01</td>
<td>3.2559E-02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.38712E+00</td>
<td>1.89190E00</td>
<td>0.00000E+01</td>
<td>2.5123E+00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.01275E+00</td>
<td>5.98248E00</td>
<td>0.00000E+01</td>
<td>6.9516E+00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.29275E+00</td>
<td>9.70446E00</td>
<td>0.00000E+01</td>
<td>9.9252E+00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### SIGMA VALUES AT HALF LEVELS

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2.91700E-03</td>
<td>7.9292E00</td>
<td>0.0000E+00</td>
<td>2.1553E+00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.17984E+00</td>
<td>1.6308E00</td>
<td>0.0000E+00</td>
<td>2.1948E+00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.55809E+00</td>
<td>5.5128E00</td>
<td>0.0000E+00</td>
<td>6.4920E+00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.03302E+00</td>
<td>9.5599E00</td>
<td>0.0000E+00</td>
<td>9.8511E+00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### LATITUDES

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-87.86380</td>
<td>-85.09653</td>
<td>-82.31291</td>
<td>-79.52561</td>
<td>-76.73690</td>
<td></td>
</tr>
<tr>
<td>-59.99702</td>
<td>-57.20663</td>
<td>-54.41620</td>
<td>-51.62573</td>
<td>-48.83524</td>
<td></td>
</tr>
<tr>
<td>-32.09194</td>
<td>-29.30136</td>
<td>-26.51077</td>
<td>-23.72017</td>
<td>-20.92957</td>
<td></td>
</tr>
</tbody>
</table>

#### GAUSSIAN WEIGHTS

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.78328070742408E-03</td>
<td>4.14703326043206E-03</td>
<td>6.50445796886448E-03</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.34630478966701E-02</td>
<td>1.57260304760191E-02</td>
<td>1.79517157756741E-02</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.43527025686925E-02</td>
<td>2.63774697150316E-02</td>
<td>2.83396726142516E-02</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This header is described in detail in "Model History Tape" on page 38.

As time stepping continues, more "NSTEP" printouts occur, then abbreviated header printouts as additional history tape files are written.

When it is time to transfer output files to the Mass Store, information concerning this operation is printed. First, subroutine SAVDIS calls system routine mswrite, and the actual command is printed for the user's information. The following printout then explains the parameters to mswrite and, in the case of the history tape, gives other information about the output data.

SAVDIS: Disposing Mass Store Volume/JQUSER/ccm2/test/hist/h0001
Write password = passwd
Retention Time = 35 DAYS
Cartridge = CTPUBLIC
Comment Field =
DAYS: 0.000-0.500 DATES: 0.000Z 901 - 12.000Z 901
Primary history tape
Output at NSTEP = 37
Number of time samples on this tape = 2
Model Day = 0.50

At the end of the run the following printout indicates the length of the Model simulation:

Number of completed timesteps: 36
Time step 37 partially done to provide convectively adjusted and
time filtered values for history tape.

******** END OF MODEL RUN ********

Job accounting information is summarized by the ja command. In the "Job Accounting — Command Report" table below, the entry starting with "ccm.xx." shows the elapsed, user, and system CPU seconds used in running the Model, and the memory "hiwater" mark, which is designated in "clicks," or 512-word blocks.

Thus, this run began at approximately 14:19, and ran 161 (user) CPU seconds in 2944 elapsed (wall clock) seconds. At the point of greatest memory usage, it used 12908 "clicks," or 6.6 Megawords of main memory.
A summary of each CPU will follow, giving CPU time for each processor. A detailed breakdown of actual GAU (general accounting units) charges will follow this summary.

2. Model Error Messages

All errors trapped by the Model result in a printed message starting with the name of the subroutine of origin. The following is an alphabetical list of these messages, along with an explanation for each. All errors are fatal (cause a Model abort) unless otherwise stated. Where possible, we include a suggested course of action for the user.

**ATCHBND:** Failed to open file xxxxxxx ierr=nn

The requested boundary dataset file was found on the Cray disk as xxxxxxx, but the open statement failed.

**ATCHBND:** Dataset not found for unit nn

The requested boundary dataset was not found on the Cray disk or in the specified Mass Store pathname. The user's specification of the boundary dataset name in the namelist may be syntactically incorrect.

**ATTACH:** Failed to open file xxxxxxx ierr=nn

The requested file was found on the Cray disk, but the open statement failed.

**ATTACH:** Error from msread -- Message from merror

**ATTACH:** Local file = /xxx/xxxxxxx
Remote pathname = /xxx/xxx/xxxxxxx

The msread (Mass Store read) command in ATTACH returned the indicated error code.

**ATTACH:** Error opening unit uu Message number = nn

The attempt in subroutine ATTACH to open the file failed.
BLDFLD: Error in field name translation,
field name = nnnnnn

Statement function TRATOI encountered a non-space, non-alphanumeric character when translating the field name to a base 37 integer for faster compares. Field names should be left-justified with no more than 8 alphanumeric characters (embedded spaces are OK).

BLDFLD: Too many fields -- pflds, if=nnn mmm

Only pflds words of space are allocated for arrays carrying field information. If you have added fields, you must increase pflds accordingly.

BLDFLN: maximum output file number exceeded

A history tape number was requested which exceeded 9999. Formatting of history tape names is limited to "h" plus 4 digits (see “Use of the NCAR Mass Storage System” on page 92).

CALDYI: Invalid base month input: mm

The input month number (from the initial dataset or from NNBDAT in the namelist) was not between 1 and 12.

CALDYI: Invalid base day of base date input: ddd

The input day number (from the initial dataset or from NNBDAT in the namelist) was not valid for the input month.

CALDYI: Invalid base seconds (msbase): ssss

The input number of seconds (from the initial dataset or from NNSBAS or NNBSEC in the namelist) was not valid.

CALDYI: Invalid base seconds (mbsec): ssss

The input number of seconds (from the initial dataset or from NNSBAS or NNBSEC in the namelist) was not valid.

DATA: Cannot find case environment variable

The setenv command in the run script for the environment variable $CASE was either missing or erroneous.

DATA: Case environment variable .gt. 8 characters.

The environment variable $CASE contained more than 8 characters. The case name must fit into one Cray word, mcase in the history tape header.
DATA: Cannot find logname environment variable

This error would indicate a system problem, since $LOGNAME is a system environment variable.

DATA: Must input either nestep or nelapse

The user must specify an ending time for the Model run.

DATA: Too many auxiliary tapes declared.

The user has declared more than ptapes auxiliary history tapes via the input parameter AUXF. Users who want more than 5 auxiliary tapes must increase ptapes in pagrid.com.

DATA: Must specify non-zero nhtfrq

The history tape write frequency for one of the declared history tapes is zero.

DATA: iradae must be an even multiple of irad.

iradae = ii, irad = jj

The absorptivity/emissivity calculation must fall on a radiation timestep.

DATA: Auxiliary write times must evenly divide primary write times. Primary frequency=nnn, Auxiliary frequency=nnn.

The history tape write frequency (NHTFRQ) for an auxiliary tape must be an even divisor of the value for the primary tape, or restart will not work properly.

DATA: ***NOTE: Extra overhead invoked putting a/e numbers on restart dataset***

To avoid, make mod(nhtfrq, iradae) = 0

Not a fatal error. There will be a restart file generated for absorptivity/emissivity values since the history tape write does not necessarily occur on an absorptivity/emissivity timestep, as defined by IRADAЕ.

DATA: KMXHDC must be between 0 and plev-1

KMXHDC, an input parameter, is the number of levels over which to apply the Courant limiter.

DATA: Starting tape number (stfnum) too large.

Max = 9999. Stfnum = ssss

Output file numbering starts at STFNUM, which must not be larger than the largest allowable number (4 digits).
DATA:Invalid packing density \( n \)

Packing density NDENS must be between 1 and 4.

DATA:For BRANCH run, NREVSN must be a full pathname.

Because the case name changes on a branch run, the user must specify the full pathname for NREVSN.

DATA:Pathname for regeneration file too long.

NREVSN was input with more than 80 characters.

DATA:Invalid averaging flag input for tape \( t \)
Flag = \( F \)

NINAVG must be one of the following:
A  averaged
I  instantaneous
M  minimum
X  maximum

See Table I.1 on page 14.

FLDLST:cant find auxiliary field aaaaaa

Input parameter AUXF requested a field not in the master field list.

GAUAW:Error exit, no convergence in 10 iterations

The iterative technique to find abscissas for the Gaussian integration failed to converge in 10 tries. Check that epsilon (GAUAW local variable eps) is consistent with machine precision.

GESTBL:FATAL ERROR

TMAX AND TMIN REQUIRE A LARGER DIMENSION ON THE LENGTH OF THE SATURATION VAPOR PRESSURE TABLE

ESTBL(PLENEST) TMAX, TMIN, AND PLENEST \( \Rightarrow fff.f \)
ggg.gg iii

The GESTBL procedure requires that the parameter plenest (in eslookup.com) be \( \geq \) tmax - tmin + 2 (where tmin and tmax are set in ESINTI). Reset parameter plenest to satisfy this condition.

GFFGCH:FATAL ERROR

TRANSITION RANGE FOR WATER TO ICE SATURATION VAPOR PRESSURE, TR, EXCEEDS MAXIMUM ALLOWABLE VALUE OF 40.0 DEGREES C \( \text{TR} = fff.f \)

This routine assumes that the maximum size of the transition region for calculating saturation vapor pressure over water and saturation...
vapor pressure over ice is 40 degrees C. The variable \texttt{trice} in \texttt{ESINTI} must be set to be \texttt{.le. 40} to avoid this error.

\textbf{HERXIN:} Fatal error: \texttt{ppdy} must be set to 4

Parameter \texttt{ppdy} in \texttt{parslt.com} must be set to 4 for the semi-Lagrange code to run properly.

\textbf{INIDAT:} Local space not large enough for initial dataset record. Local parameter \texttt{phtbf} must be at least \texttt{nnnn}

The initial dataset record size is too large for the locally declared buffer in \texttt{INIDAT}. \texttt{phtbf}, a local parameter to \texttt{INIDAT} that defines the size of space allocated for reading a data record from the initial dataset, must be increased.

\textbf{INITIAL:} Failed to open unit \texttt{uu} \texttt{ierr=nn}

The Fortran \texttt{open} statement returned a status of \texttt{nn} when opening file \texttt{NSVSN} (the restart dataset).

\textbf{INITIAL:} The regeneration file size is too small:
Maximum file size (\texttt{mxszrg}): \texttt{mmmmmm}
Size of a latitude record: \texttt{llllll}

A single latitude record is too large to fit in the allowed regeneration file size, as defined by \textit{namelist} parameter \texttt{MXSZRG}.

\textbf{INITIAL:} Too many primary regeneration files required: Maximum number of files (\texttt{pnrg1}) = \texttt{nnn}
Maximum size of files (\texttt{mxszrg}) = \texttt{mmmm}
Actual number of files required = \texttt{nn}
Actual size of latitude record = \texttt{llll}
Number of latitudes per file = \texttt{nn}

Parameter \texttt{pnrg1} in \texttt{/comlun/} must be increased.

\textbf{INITIAL:} Too many secondary regeneration files required: Maximum number of files (\texttt{pnrg2}) = \texttt{nnn}
Maximum size of files (\texttt{mxszrg}) = \texttt{mmmm}
Actual number of files required = \texttt{nn}
Actual size of latitude record = \texttt{llll}
Number of latitudes per file = \texttt{nn}

Parameter \texttt{pnrg2} in \texttt{/comlun/} must be increased.

\textbf{INITIAL:} Error returned from \texttt{RDHDR} for record \texttt{n}

An error was returned when reading header record \texttt{n} from the initial dataset.
INITIAL: MISMATCH BETWEEN INITIAL DATASET HEADER AND PARAMETERS
MLON: nnnn PLON: mmmm
MOREC: nnnn PLAT: mmmm
MLEV: nnnn PLEV: mmmm
MTRM: nnnn PTRM: mmmm
MTRN: nnnn PTRN: mmmm
MTRK: nnnn PTRK: mmmm

The header from the initial dataset has different resolution parameter(s) than the Model parameters indicate.

INITIAL: Illegal format type on initial dataset.
The rightmost two digits of the initial dataset format type (MFTYP) must be equal to 43 (CCM2).

INITCOM: mmax=ptrm+1 .gt. plon/2
The Model resolution parameters are inconsistent.

INTHT: Too many history tapes declared, max= n
To increase, change parameter ptapes.
ptapes defaults to 6, including the primary history tape.

INTHT: Failed to open file uu, ierr= ee
The attempt to open the history tape unit failed.

INTHT: Unable to open file sunit: ios= eee
An error status was returned from trying to open unit nhist for direct access on the SSD.

IOSTOP: ****************** I/O ERROR ******************
Message
IOSTOP uses system software to interpret I/O errors. Errors handled by IOSTOP will be identified by subroutine name.

LAGYIN: Error: ppdy .ne. 4
Parameter ppdy in parslt.com must be set to 4 for the semi-Lagrange code to run properly.

LIMDX: Local work array DELI not dimensioned large enough. Increase local parameter pbpts to iiii
The parameter pbpts, local to LIMDX, must be increased to the value of pidim.
LUNITS: Unable to open file uu: ios = eee

Error status eee was returned when opening unit uu for direct access.

MKSLIC: Level mismatch for field: ffffffff The array is single level but the data is multilevel
MKSLIC: Level mismatch for field: ffffffff The array is multilevel but the data is single level

MKSLIC was called incorrectly for this field, or the data being transferred are incorrect.

MKSLIC: The required input field ffffffff was not on the initial data file The available fields are: (list of fields)

The initial dataset is missing a required field.

NAVU: Ran out of Fortran unit numbers

Routine NAVU can't find an unused logical unit number between 1 and 99.

OUTFLD: Error in translation of field name.

kfname = kkkkkkk

Statement function TRATOI could not translate field name. Field names should left-justified with no more than 8 alphanumeric characters (embedded spaces are OK).

OZNINI: Consecutive rewinds requested on ozone dataset.

Rewinds should be requested on this dataset only to cycle for another year. This error may indicate a corrupted or empty ozone dataset.

OZNINI (or OZNINT): Error>>>>>Two consecutive time slices on ozone dataset are more than 45 days apart.

Although the Model can run with irregularly spaced ozone data, these data are interpolated in time, so large gaps may cause inaccuracy. The most recently read ozone header will be printed for additional information.

OZNINI: All data on ozone dataset is beyond current date

Data on ozone dataset do not bracket the current Model date. The most recently read ozone header will be printed for additional information.
OZNINI (or OZNINT): Multiyear file assumed. EOF read.

An unexpected end of file was encountered on a noncycling ozone dataset.

POINTS: Mismatch between calculated length and parameter NLNBUF1 (or NLNBUF2) = nnnnnn,
         PFLENB (or PFLENA) = mmmmmm

The allocated space as indicated by parameter value PFLENB or PFLENA is insufficient for the corresponding calculated length of data.

PREALC: Cannot allocate nnnnnn words for file fffff, status word = m

This is not a fatal error. The attempt to preallocate disk space for the indicated file was unsuccessful. The file may still be written.

QNEG3 from NNNNNN: m=mnm lat=lll Min. mixing ratio violated at iiiii points. Reset to r.rrrrrEee
Worst = w.wwwwwwEee at i,k=iiii kkk

This nonfatal message advises the user that some point values of the constituent field have fallen below their respective minimum acceptable threshold (see qmin (pcnst) in comqmin.com), and have been reset to the minimum threshold value. This should be a relatively rare occurrence, i.e., frequent occurrences of this message indicate a fundamental problem with the treatment of the particular constituent field.

RDHARR: end of file on unit nn

An end of file was encountered in reading a history tape data record. This error may occur when reading the initial dataset or any boundary dataset. See the accompanying traceback for the calling routine.

RDHDR: integer (or character or real) header record too long. Allocated space = nnn, Actual length= mmm

Not enough space was allocated for one of the header records. If the actual read of the record produces an error, control returns to the calling routine for error processing.

READARR: I/O error
READARR: End of file

READARR calls IOSTOP to document these errors.
READRG: I/O error in record number nn
READRG: End of file
READRG calls IOSTOP to process these errors.

READRG: Must change case name on a branch run. Previous name = cccc, current case name = cccc

To avoid overwriting existing output files, the case name ($CASE) on a branch run cannot be the same as the case from which the branch is made.

READRIC: bad read of record nn from unit uu

Direct access read returned an error on unit uu. IOSTOP is called to further explain the error message.

RESTRT: No restart data available

A restart dataset was not specified in the namelist of the previous run. Therefore, no regeneration datasets were written.

RESTRT: Error returned from RDHDR for record n

Restart and regeneration runs use information from the last history tape header. An error was encountered in reading a header from this history tape.

RESUME: nstep (starting time step) .ge. nstop nstep = nnn, nstop = mmm

namelist parameter NESTEP should be greater than the restart timestep.

SAVDIS: Error from mswrite -- errmsg

System routine merror is called to interpret a nonzero (error) status from mswrite.

SAVDIS: Error in ishell call, ier= nnn

The indicated status was returned from system routine ishell. The call which generated this error is printed just prior to this message.

SCAN1: *** Courant limit exceeded at k, j= kk jj (estimate = eee.eee), solution has been truncated to wavenumber nnn ***

The CFL limit has been exceeded. Therefore the high-frequency end of the spectrum has been truncated at the level specified by namelist parameter KMXHDC. Not a fatal error.
SETTAU: UNSTABLE MEAN TEMPERATURE. STOP.

The mean temperatures set for the semi-implicit scheme result in an unstable scheme which would blow up if used.

SRFTSB: Error returned from mtdlss

An error has been detected in the multiple tridiagonal linear system solver MTDLSS. The error can be one of two types: a declared storage error or a numerical error (e.g., a singular matrix). Print the error status flag, ier, and see MTDLSS for additional detail.

SSTINI: Consecutive rewinds requested on sst dataset.

Rewinds should be requested on this dataset only to cycle for another year. This error may indicate a corrupted or empty SST dataset.

SSTINI (or SSTINT): Error >>>>> Two consecutive time slices on sst dataset are more than 45 days apart.

Although the Model can run with irregularly spaced SST data, these data are interpolated in time, so large gaps may cause inaccuracy. The most recently read SST header will be printed for additional information.

SSTINI: All data on sst dataset is beyond current date

Data on SST dataset does not bracket the current Model date. The most recently read SST header will be printed for additional information.

SSTINI (or SSTINT): Multiyear file assumed. EOF read.

An unexpected end of file was encountered on a noncycling SST dataset.

TIREAD: End-of-file reading header for time-invariant dataset, record number n

Probably the time-invariant dataset is empty or corrupted.

TRJGL: ***** MODEL IS BLOWING UP ***** ***
Error in subroutine trjgl *** Parcel associated with longitude ll, level kk and latitude jj is outside the model domain.

The semi-Lagrangian transport package has detected that the Model is becoming unstable.
TRUNC: Error in truncation parameters
ntrm.lt.(ptrk.ptrn)
TRUNC: Error in truncation parameters
ptrk.lt.ptrn
The horizontal truncation parameters in pmgrid.com are not consistent.

VRTMAP:**** Error: Not enough artificial grid intervals. Currently, "pmap" is set to ppppp
Reset parameter "pmap" to at least nnnnn
In the semi-Lagrangian transport scheme, the artificial vertical grid declared to map departure points into the Model grid is not large enough to handle the vertical resolution of this run. The user must increase the value of STEPON local parameter pmap.

WRAPUP: Failed to open file hhhhhh ierr=nnn
Auxiliary tapes are opened to append by WRAPUP. This open failed.

WRITERIC: bad write of record rr to unit uu
The Fortran direct access write to unit uu returned an error. IOSTOP is called to further explain the error message.

WRTARR: I/O error
IOSTOP is called to explain an error returned from an array write.
In this section we describe the CCM2 code in detail, for those users who seek to modify the Model or add their own parameterizations to it. First, all data structures are outlined, including gridpoint and spectral arrays. Then the I/O internal to the Model is documented, along with disk and Mass Store file management.

We present the main time-marching procedure as code flow and describe the strategy used to multitask CCM2. In “Changing the Model” on page 115, we show how to make changes to the code and how to troubleshoot those changes. Finally, we document the coding standard used in the standard Model.

This section is meant for those who wish to make substantive changes to the CCM2 code. Familiarity with advanced aspects of the UNICOS operating system is assumed. We will reference Cray manuals where helpful. Also note the glossary of terms in “Appendix A: Glossary.”

A. Design Philosophy of CCM2

Large computer models are notorious for being difficult to read, understand, and modify. In 1989, an international committee of geophysical modelers wrote some coding rules designed to improve these aspects of large models, especially when adding new parameterizations. Their paper, “Rules for Interchange of Physical Parameterizations” (Kalnay et al., 1989), provided valuable guidance in the design of CCM2.

In creating CCM2, the CCM Core Group tried to retain the simplicity of use that characterized the previous version of the Community Climate Model, while adding features to make it easier for the more sophisticated user to change or enhance the Model to facilitate his/her own research. Such a user may want to add variables and parameterizations to the Model, or change information on the history tape. Making these tasks simpler and more modular in nature benefits even the casual user of CCM2.

In the previous version of the Community Climate Model, CCM1, a large blank common buffer provided storage for all gridpoint fields. It served both as “permanent” cycling storage for the out-of-core\(^1\) implementation and as temporary storage, managed piecemeal by the various physics routines. This method of memory management had few safeguards and was difficult to understand and modify. In CCM2, memory is managed differently, with the following goals in mind:

1. This term refers to model data structures which reside in secondary storage and are cycled in and out of main memory as required.

\(^1\) Out-of-core implementation refers to the use of secondary storage (such as disk) to store model data that cannot fit into the primary memory of the computer.
• Using memory more efficiently, mainly to run economically at higher resolutions. Meeting this goal required that we retain the out-of-core characteristics of CCM1.

• Making data access convenient and understandable at the physical parameterization level.

• Accommodating parallel processing at the latitude loop level.

• Taking advantage of contiguous storage in doing I/O for the out-of-memory implementation.

In CCM2, the main Model buffer consists of only those variables that must be carried at more than one time level, more than one latitude scan or for contiguity purposes.

Use of synchronous I/O, and the Cray SSD

Double buffering of the Model buffer, a feature in early releases of CCM1, was eliminated in CCM2. Double buffering allowed some overlap of I/O with computations. One latitude band’s worth of data could be flowing into one buffer while computations were proceeding on another. Since the solid-state storage device (SSD) on the CRAY Y-MP is so fast (data transfer rate can actually exceed memory-to-memory speed), it was decided that the minimal benefits of double buffering were outweighed by the complications of its implementation and the penalty of a larger memory requirement.

The Model’s use of the SSD is optimized by synchronous unblocked I/O to Secondary Data Segments (SDS), as implemented in UNICOS Version 6.

3-dimensional arrays

To accommodate the semi-Lagrangian transport code, certain variables are needed in main memory as three-dimensional (longitude by vertical level by latitude) arrays, in two time levels. These arrays are in named common /com3d/ and, therefore, reside in main memory throughout the Model run.

Named common is also used in the Model to carry variables that do not vary latitudinally, as a means of avoiding enormous argument lists. All data statements which set common variables are contained in a block data routine, blkdat.

Local variables on stack

Temporary storage, such as that for Fourier coefficients during the spectral transform, is maintained as locally declared arrays on the “stack,” the automatic dynamic memory management mechanism of Fortran. All these data structures are described in detail in “Model Data Structures” on page 77.

The control structure of the Model with respect to timestepping and grid-point-to-spectral transformations, as described in the following sections, is similar to that of CCM1. The semi-Lagrangian transport comprises a third multitasked latitude loop.
Only the control routines of the Model address directly into the Model buffer. Gridpoint physical parameterizations are called via subroutine LINEMS, the main latitude line processor for the first latitude scan. Here an interface routine, LINDRV, between STEPON and LINEMS, uses argument-list equivalencing to map the buffer to multidimensional, mnemonic field arrays. This data handling method, by removing the clumsy indexing of the buffer/pointer system from the parameterizations, allows physical parameterization packages to be easily replaced by other packages, i.e., “plug-compatible.”

The CCM2 history tape interface is designed to provide users with easily used modular utility routines for recording data on the model output history file. Fields that appear in a master field list, generated at initialization time by subroutine BLDFLD, may be included on the history tape either by default or by namelist request of the user. A field is recorded in the history tape buffer by calling subroutine OUTFLD.

The user is responsible for keeping track of the time at which the recorded field is valid and how it relates to the time recorded on the header. Provision is made for recording both instantaneous and accumulated fields, as well as maxima and minima. Subroutine WRITUP, called from LINEMS, writes the latitude record from the history tape buffer to the disk and at the appropriate time subroutine WRAPUP transfers the disk file to the Mass Store.

CCM2 contains a multiple history tape capability. In the standard Model, all fields that are declared “active” in the master field list appear on the primary history tape. The user may declare additional tapes, known as auxiliary history tapes, via namelist input parameters. Auxiliary tapes may contain the same fields as the primary tape or different fields and may be written at different frequencies, with different packing densities, and with a different number of history tape writes per volume if desired.

The major parallel processing in the Model is at the latitude loop level, with an arbitrary number of processors sharing the work of each of the three latitude loops. Data structures not dimensioned by latitude are allocated on the Fortran stack, so that each processor has its own copy of writable memory for these arrays. I/O to the “work files” is implemented as Fortran direct access synchronous I/O to accommodate the random order of latitude loop execution. In the multitasked Model, all data accumulations, e.g., Gaussian quadrature, occur in the same order as in the single-threaded Model to guarantee reproducible results.

Due to the multitasking of the Model and the resulting random order of execution of the first latitude loop, the history tape, which is written sequentially, may be randomly ordered by latitude. Each latitude record contains as its first word a latitude index, numbered from south to north.
The CCM2 code follows a written syntactical standard, described in "Trouble-Shooting Model Changes" on page 119 of this document. The code has been edited to appear homogeneous as far as comments, indenting, etc., to make it more readable to the user. The intent of the commenting standard is to internally document the code so as to make a separate module document unnecessary.
B. Model Data Structures

CCM2 is a partially in-core, partially out-of-core model. This section is devoted to a discussion of the design philosophy and implementation techniques of both in-core and out-of-core gridpoint and spectral space global data structures. Some special array-indexing constructs are also addressed. Specific details of local data structures in individual routines are discussed in “Model Code Flow” on page 100.

1. Vertical Coordinate

The vertical coordinate used in CCM2 is a hybrid sigma-pressure system (Description of the NCAR Community Climate Model (CCM2) (Hack et al., 1992)). In this system, the upper regions of the atmosphere are discretized by pressure only. Lower vertical levels have the terrain-following sigma (p/ps) vertical coordinate smoothly merged in, with the lowest level(s) being pure sigma. A schematic representation of the hybrid vertical coordinate and vertical indexing used in CCM2 is presented in Figure II.1 below.
2. Gridpoint Data Structures

Gridpoint space prognostic variables and many arrays associated with the semi-Lagrangian transport scheme (SLT) are maintained in core. Other gridpoint fields exist either in the main out-of-core Model buffer, the history buffer, or as local workspace in individual routines. All gridpoint data are stored with longitude as the fastest varying subscript, followed by (if applicable) level, constituent, latitude, and finally time. Longitude indices start at Greenwich and proceed from west to east around the globe. Level indices run from the top of the atmosphere to the bottom. Latitudinal ordering is from south to north.

The term "constituent" refers to water vapor plus an arbitrary number of user-defined advected species. The number of constituents, defined by the Fortran parameter "pcnst," will always be at least 1 since water vapor is...
always transported in the default Model configuration. The variable \( \text{lat} \) is used consistently throughout CCM2 to represent the current latitude index, counting from south to north. This is the order of storage for all latitudinally dependent arrays. Two time-level indices are necessary on the prognostic variables due to the leapfrog timestepping scheme.

Model gridpoint space arrays are almost universally declared with longitudinal dimension \( \text{plond} \), where \( \text{plond} = \text{plon} + 1 + 2*\text{nxpt} \). \( \text{plond} \), \( \text{plon} \), and \( \text{nxpt} \) are all defined in Fortran \texttt{parameter} declarations. These and other parameter values used by the Model are tabulated in “Appendix C: CCM2 Parameter Definitions.”

\( \text{plon} \) is the number of actual data points in the longitudinal direction, and \( \text{nxpt} \) is a wrapping number required by the SLT package. Arrays that go through the Fourier transform require two additional longitude points at the end to accommodate storage of the wave 0 (mean) information by the \texttt{fft} package. Arrays used by the SLT code require at least 1 additional longitude point at the beginning and at least 2 additional points at the end of most arrays. To avoid proliferation of a dizzying assortment of longitudinal dimensions, it was decided to make the longitudinal dimension of Model arrays \( \text{plond} \) wherever possible, whether or not it was required.

Actual data start at longitude location \( 1 + \text{nxpt} \) for all of the prognostics except surface pressure, and location 1 for all other non SLT-specific arrays with longitude dimension \( \text{plond} \). Surface pressure data (in /com3d/) start at location 1 because this array is not used in the SLT package. This difference in starting location poses an indexing problem for non-SLT gridpoint space Model routines. It is undesirable to have to keep track of where the real data start for every array. The solution was to pass the address of where the data start to non-SLT Model routines at a very high level in the calling tree. For example, consider the call to \texttt{LINEMS} from \texttt{LINDRV} (this is very high in the calling tree: refer to Fig. 1.2). Some of the calling and declaration sequence looks like:

```
LINDRV:
$ call linems(..., ps(1, lat, n3), q3(i1, l, l, j, n3),
      bl(nzml+1), ...)

LINEMS:
$ subroutine linems(... ps, q3, vort)
   real ps(plond),
   q3(plond, plev, pcnst),
   vort(plond, plev)
```
In LINDRV, each argument passed to LINEMS represents the starting location of the real data for that field. This way all references to fields in LINEMS and below in the calling tree can begin at array element 1. The user need not be concerned with offsets unless he/she is dealing with code at the LINDRV level or farther up where the full data structures are available.

Analogous to the longitudinal case, additional nonphysical gridpoints are needed by the SLT package in the latitudinal direction. Some arrays are therefore dimensioned “platd,” where platd = plat + 2*(nxpt + jintmx). Fortran parameter “plat” is the number of actual Model latitudes. “nxpt” and “jintmx” are extensions beyond the southernmost and northernmost physical locations. The starting latitude location for physical data in these arrays is 1 + nxpt + jintmx. LINDRV passes to lower-level routines the array address corresponding to the latitude index at which the real data start.

In the above example at the level of LINEMS, latitude indices are no longer present because all computations in LINEMS and below in the calling tree are independent of latitude. A similar approach is used with the array that contains moisture plus other advected constituents. In the radiation code, for example, water vapor is the only advected specie. Therefore, array h2ommr in the radiation package has no constituent index.

(a) Model Buffer

The main Model buffer contains longitude-level slices of a number of fields stored contiguous to each other. These arrays are cycled to and from an I/O device, latitude band by latitude band. A schematic representation of the main Model buffer is presented in Figure II.2 below. The cycling of the buffer to and from the work units is depicted in Figure II.5 on page 100 and in Figure II.6 on page 101.
Figure II.2. Layout of Main CCM2 buffer.

The buffer holds only those out-of-memory fields that must be carried across discrete Model time levels, that are needed by more than one Gaussian latitude scan, or that need to be contiguous to other fields. One example of fields that must be carried from one time level to the next is soil temperatures, determined from the solution of a set of time-dependent equations. Another example is the radiative heating rates that must be passed forward in time between calls to the radiation routines.

Surface pressure, vorticity, and divergence are in the buffer for contiguity purposes. Contiguity is required so that the fft package is invoked a minimal number of times. This is very important since the fft used in CCM2 vectorizes over the number of transforms being done. By putting surface pressure next to vorticity and divergence, an 18-level model produces a respectable vector length of 37 (i.e. $2 \times 18 + 1$) for the Fourier transform of these fields. If surface pressure were transformed separately, the vector length for this field would be 1, which would significantly degrade performance. Surface pressure is actually memory-contained.
Use of buffer fields

Integer pointers to the start of buffer fields are maintained for use in the high-level control routines of the Model, but computationally the fields are accessed as individual, appropriately dimensioned arrays. For instance, a high-level routine in CCM2 might contain a subroutine call using the buffer pointers to reference a particular field, thus:

```fortran
  call xxx(buf(nqrs+1),...)
  ...
  return
  end
```

and the called subroutine references this field mnemonically, thus:

```fortran
  subroutine xxx(qrs,...)
  ...
  real qrs(plond,plev)
  ...
  do x=1,plev
    do i=1,plon
      ...qrs(i,k)
    end do
  end do
  end subroutine
```

This approach yields readable code and actually runs a bit faster than using direct buffer references throughout. It also enables usage of the array bounds checker (cft77 -Rb) for debugging. Equivalent functionality may be obtained through Fortran pointer variables. Subroutine SPEGRD uses the pointer implementation. SPEGRD contains the statements:

```fortran
  real buf(pflen) ! model buffer
  pointer (pz,z) ! declaration of pointer variable
  real z(plond,plev) ! part of model buffer
  ...
  pz = loc(buf(nzpl+1)) ! array z may now be accessed
```

(common/com3d/), and is copied to and from the Model buffer for the sole purpose of avoiding the fft performance penalty.
Array $z$ is referenced as a separate two-dimensional array as was $qrs$ in the example above, but the need for an interface routine to pass in the address $buf(nzp1+1)$ is obviated.

(b) History Buffer

In CCM2, data written to the history file are maintained in a separate buffer distinct from the main Model buffer. This buffer must also be cycled to the (SSD (as unit nhist) because few fields written to the history file are instantaneous. Therefore, these data must be carried across time levels between OUTFLD calls to the history buffer. The history buffer is defined by what is known in Cray nomenclature as an “automatic array.” This is local workspace, the size of which is determined upon invocation of the routine in which the array is declared.

The declaration in LINDRV looks like:

```fortran
subroutine lindrv(lhhbuf)
real hbuf(lhbuf)
```

Space for $hbuf$ is allocated dynamically from the system dynamic memory repository (called the “heap”) when LINDRV is called. Thus, the user is not required to count single- and multi-level fields on the history tape and set Fortran parameters accordingly. These computations are all handled internally by the Model.

(c) Absorptivity/Emissivity Arrays

Longwave radiation calculations of total absorptivities and emissivities for all gases are also maintained on an out-of-memory file unit (nabem) and are referenced latitude band by latitude band in an unblocked fashion. In the default Model configuration, the absorptivities and emissivities are computed and written out every 12 simulated hours and read in as input every hour. Size and relative infrequency of use are the main reasons for maintaining these fields in secondary storage. Nonadjacent layer absorptivities, nearest-layer absorptivities, and total emissivity arrays are stacked in array absems in order to minimize the number of I/O calls.

We recommend that the user choose a history tape write frequency (specified by parameter NHTFRQ) that divides evenly into the frequency of the longwave radiation calculation (specified by IRADAE). Otherwise, the large absorptivity/emissivity arrays must be written to the regeneration datasets.

(d) Local Workspace

Nearly all local workspace declared in individual routines is stack-based. That is to say, variables declared locally to a given subroutine exist only during the lifetime of that routine, disappearing upon execution of the.
return statement. This approach has the advantage of allowing the user to declare workspace wherever it is required without having to worry about clobbering space that another routine needs. Also, more efficient use of memory is made and debugging tasks are simplified.

A stack-based memory management scheme is also vital for multitasking considerations. This topic is discussed in more detail in “Multitasking Strategy” on page 109. Of locally declared variables, only those initialized with a data statement, or explicitly referred to in a save statement, are allocated statically. When a variable is allocated statically, its value is preserved between calls to the routine in which it is declared.

(e) In-Core Gridpoint Arrays

Prognostic variables surface pressure, zonal wind, meridional wind, temperature, and water vapor (hereafter referred to as ps, u, v, t, and q) are kept in memory at all times (common/com3d/). This is primarily for the benefit of the SLT code. The SLT algorithm requires data simultaneously at a number of longitudes and latitudes, which is conceptually easier to understand (and implement) if all the data are memory-resident. Also, since the prognostic variables are required in all three latitude scans in the Model (SCAN1, SCANSLT, and SCAN2), they are the prime candidates for keeping in memory in to minimize I/O requests to the out-of-core file units.

A schematic view of the data structure defining u, v, t, and q is shown in Figure II.3 below. This perplexing storage arrangement accomplishes two important goals. The contiguity of fields u, v, and t results in a more efficient Fourier transform for the same reasons mentioned earlier for surface pressure, vorticity, and divergence in the Model buffer. Additionally, the SLT algorithm takes advantage of the contiguity of u and v to produce more efficient code. The prognostic variables are stored in a common block, i.e., static storage, because they are required throughout the entire Model. In CCM2 actual temperature, not perturbation temperature, is the quantity stored in the three-dimensional data structure. Perturbation temperature is computed only as it is needed. This was not the case in CCM1.
If \( \text{pcnst} \) is greater than 1 (i.e., water vapor plus other user-defined advected species are being transported), longitude-height cross sections of the additional constituents are stored adjacent to water vapor. Access to the constituent array is of the form \( q_3(i\text{lm}+i,k,m,j\text{lm}+\text{lat},n3) \) where \( m \) is the constituent index, \( \text{lat} \) is the south-to-north latitude index, and \( n3 \) is the time level index. \( m=1 \) refers to the moisture field, with additional constituents stored in locations \( m=2, \text{pcnst} \).

Since \( u, v, \) and \( t \) are a part of this same data structure, a dimensioning construct was employed to avoid having to include a constituent index on these fields when accessing the full data structure. Fortran parameter \( \text{plevd} \), defined as \( \text{plev} \times \text{pcnst} \), allows dimensioning of \( u, v, \) and \( t \) as \( (\text{plond}, \text{plevd}, \text{platd}, 2) \).
The *common* block in which the prognostics reside is declared as follows:

```
common/com3d/n3m1,n3
common/com3d/ps (plond, plat, 2), x(plond, plev, 3+pcnst, platd, 2)
```

C

```
real u3(plond, plevd, platd, 2), ! u-wind component
$v$ v3(plond, plevd, platd, 2), ! v-wind component
$t$ t3(plond, plevd, platd, 2), ! temperature
$q$ q3(plond, plev, 3+pcnst, platd, 2) ! specific humidity and constituents
equivalence (u3, x(1, 1, 1, 1, 1)),
(v3, x(1, 1, 2, 1, 1)),
t3, x(1, 1, 3, 1, 1)),
(q3, x(1, 1, 4, 1, 1))
```

C

```
integer n3m1,n3 ! time index pointers
real ps, ! surface pressure
$x$ ! contiguous u,v,t,q
```

The 3 in the 3+pcnst dimension accounts for prognostic variables u, v, and t. "n3" and "n3m1" are time level pointers used as the last subscript in the prognostic arrays. Each holds the value 1 or 2, with the values toggled after each timestep to avoid a memory transfer of time n–1 data. Equivalence array "x" allows reference to all four three-dimensional prognostic arrays as a single entity.

**SLT in-core arrays**

In addition to the prognostics, SLT three-dimensional fields lammp, phimp, sigmp, qfcst, and hqfcst are held in core and given global scope by keeping them in a *common* block (/comsIt/). These arrays are in *common*, rather than local to SCANSLT, because they need to be written to the regeneration dataset and because they are required in other scan loops.

Other three-dimensional arrays held in memory, but which are not written to the regeneration dataset are local SCANSLT arrays uxl, uxr, qx1, and qxr. wfld is a three-dimensional array computed in SCAN1 but used in SCANSLT. It is, therefore, local to STEPON, which calls both SCAN1 and SCANSLT. The SLT scheme may transport an arbitrary number of constituents. It is important to realize that in-core arrays q3, qfcst, hqfcst, qx1, and qxr have a constituent dimension, and transporting a large number of constituents can lead to rapid growth in memory use.

### 3. Spectral Data Structures

Spectral coefficient arrays and Legendre polynomials are kept in memory in *common* block /comspe/. The spectral coefficients for vorticity, divergence, temperature, moisture, and log surface pressure are maintained as individual one-dimensional arrays (vz, d, t, q, and alps, respectively).
Ordering of the complex coefficients within the arrays is along diagonals of $m, n$ (i.e. zonal wavenumber, total wavenumber) space. Level index "k" (not to be confused with spectral truncation parameter "K") is the slowest varying index of these arrays. Spectral space computations on a vector computer are done most efficiently with this ordering scheme because it results in longer vector lengths than any other ordering scheme. Arrays ncoefi, nm, and nco2 in /comspe/ provide starting locations and lengths so that various $m, n,$ and $k$ can be easily accessed. A pictorial representation of the structure of CCM2 spectral space arrays is presented in Figure II.4 below.

![Diagram](image-url)

**Figure II.4.** Spectral Storage Arrangement. Notice the linear storage arrangement for spherical harmonic coefficients and associated Legendre polynomials.

4. Other Common Blocks and Parameter Decks: The Parameterization Interface

As mentioned in "Design Philosophy of CCM2" on page 73, the manner in which parameterizations interface with the Model is designed to easily allow replacement of a parameterization. The major consideration for such replacement is interfacing the data structures of the new code with those of the main Model. This interface is an effort to accommodate so-called "plug-compatible" physics packages, such that existing parameterizations may be removed and replaced easily and parameterizations developed elsewhere can with a minimum of programming effort be incorporated into the Model.
Toward this goal, each physics package has its own common block which contains only those physical constants which are required for its own use. For example, /comvd/ contains eight variables, namely, just those constants that are required by the vertical diffusion package. The various physical parameterizations that make up the physics of CCM2 are hooked to the Model control code using a standard form of initialization and interface routines.

The Model must perform initialization tasks at, among other times, the beginning of a run and the beginning of a timestep. Each parameterization contains an initialization routine called at the beginning of the Model run. In most cases, these routines are named “XXINTI,” where “XX” is the standard prefix for the particular parameterization. Most of these routines are called from subroutine INTI, called from the main program, CCM2, before the timestepping procedure begins. These routines set constants in the parameterization-specific common blocks and do other initialization tasks that are not time or latitude-dependent. Some data initialization is done within the block data subprogram.

Certain constants may appear in more than one common block, as needed by various parameterizations. For instance, constant cpair, the specific heat of dry air, appears in /comadj/, /comgw/, /comrf/, /comts/, /comvd/ and /crdcon/, as well as in /comcon/. Variables in /comcon/ are set in subroutine INITCOM. Then INTI passes the required /comcon/ constants into each “XXINTI” routine, which sets the individual common block constants from the passed values. It is at this point that unit conversions or other manipulations as required by the individual parameterization may be performed. This convention allows another parameterization to interface with the Model constant values without changes to its own internal common storage.

Other common blocks (i.e., those not directly related to physics code) are defined by functional purpose and exist either to give their variables global scope or to avoid the necessity of passing certain variables through many argument lists.

Each parameterization includes an interface routine that passes data between the Model control code (i.e., the Model data structures) and the parameterization via the Fortran calling sequence. This allows the parameterization to access each required Model field using a mnemonically named multidimensional array, as shown in “Model Buffer” on page 80.

Like the Model common blocks, Fortran parameter statements are split by functional purpose into separate “include” files. One include file contains the basic grid resolution parameters (pmgrid.com). Other parameter include files are as follows: pagrid.com contains some auxiliary parameters used primarily by the history tape handler; pspect.com contains the spectral resolution parameters; pdataloc.com contains pointers to
the starting data locations in the extended (SLT) grid. See "Appendix C: CCM2 Parameter Definitions." There are a few parameter include files that are parameterization-specific. parpbl.com is used by the planetary boundary layer scheme, and parsit.com contains constants used by the SLT scheme.

5. Out-of-Core Data Storage: The SSD Work Units

All I/O in CCM2 is synchronous. In addition to being simpler and easier to understand than double buffering, single buffered, synchronous I/O allows the use of standard Fortran read and write which makes porting the code to different machine architectures an easier task. CCM2 uses the UNICOS Version 6 "Secondary Data Segment" (SDS) file construct on the Cray SSD for the work files required by the out-of-core implementation. These files are written as Fortran direct access files, where each record represents a latitude line. This facilitates multitasking over the latitude loop, since the Model can randomly access any latitude record.

The units 11, 21, 22, and 60 (Model variables nral, nrbl, nhist and nabem) are assigned as scratch units on the (SSD, i.e., when the Model job terminates, these files automatically disappear from the SSD. Subroutine LUNITS opens units nral, nrbl, and nabem for Fortran direct access. The data on nral are the portion of the buffer reserved for time n+1 data (called b2 in Subroutine LNDRV), between pointers nprg and nprgnd (see Figure II.2 on page 81). The data on nrbl are the buffer b in Subroutine LNDRV, containing the half-time-filtered data at nprgt1 and the n−1 data at nprgml.

Both the main Model buffer and the history buffer are written to and read from the out-of-core I/O device (SSD) as unblocked files. Record size is computed by rounding up to the nearest multiple of 512 words. This saves an enormous amount of system CPU time which would otherwise have to be spent blocking and unblocking the data records for the user.

---

1. I/O Technical Note, Cray System Documentation, SN-3075 6.0.
C. Disk and Mass Store File Management

File management in the Model includes acquiring files needed as input from the NCAR Mass Storage System (MSS), accessing data from these files as they reside on temporary Cray disk using standard Fortran I/O of various kinds, creating Model output files on disk, and disposing them to the MSS using a standardized naming scheme. In this discussion, we will describe how and where in the code these operations take place.

A Model initial run starts from an input dataset of atmospheric conditions, called the “initial dataset”, and from several boundary condition datasets. All these are described in detail in “Input Datasets” on page 32. On a continuation run (see “Continuation Run Logic” on page 96), the Model state is recreated from regeneration datasets and possibly the last history tape written by the previous run.

Subroutine INITIAL issues a call to Model utility routine ATCHBND to acquire the initial dataset from the MSS via the system routine msread. ATCHBND first checks to see if the file exists in the current directory on the Cray disk. If not, it next checks to see if the file resides in the Cray permanent disk directory defined by /ccm/ccm2/T42 for T42 resolution, where standard boundary datasets are stored. If not, msread reads the dataset from the MSS to the current directory. In any event, the file is then assigned to unit ninit.

INITIAL then calls Model routine RDHDR to read the header records of the initial dataset. Subroutine INIDAT reads and transforms the data. Processing of the initial dataset is then complete.

As described in “Restart/Regeneration Datasets” on page 53, the restart dataset contains only a Mass Store System pathname for the set of regeneration datasets last disposed to the MSS. In the case of a restart run, subroutine RESTRT uses this pathname to acquire all relevant regeneration datasets, calling Model utility routine ATACH. For restart and regeneration runs, RESTRT also acquires the last history tape from the case, reads through the time samples to get the latest header data, and positions the tape to write the next sample. If there are auxiliary tapes declared, these are also positioned.

The main program calls subroutine INTBND to initialize boundary condition datasets. This routine calls in turn subroutine TIREAD to process the time-invariant boundary dataset, subroutine SSTINI for the SST dataset and subroutine OZNNINI for the ozone dataset, first calling ATCHBND to acquire each dataset as described for the initial dataset. The Model reads these datasets on both initial and continuation runs.

On an initial run, the history tape unit hunit is assigned by subroutine INTHT. For a continuation run, this is done by RESTRT, both for continu-
ing an existing history tape, as described above, and for starting a new tape, as in the branch run. Subroutine RGNFLS is called by INITIAL for initial runs and by RESTRT for continuation runs to assign the complete set of regeneration datasets.

Routines assign, setf, msread and mswrite are NCAR Cray-specific system routines. assign connects a UNIX file with a Fortran logical unit number. setf preallocates disk space for a UNIX file. Routines msread and mswrite read and write files on the Mass Storage System. See Table II.1 on page 95 for the locations of these system routine calls.

I/O in the Model is discussed in detail in “Out-of-Core Data Storage: The SSD Work Units” on page 89. All I/O is implemented using standard Fortran I/O statements. Although the work files used in the out-of-core implementation are unblocked to enhance I/O performance, the history tape is written as a sequential, COS blocked dataset, due to the existence of external programs, such as the CCM Modular Processor, which read the history tape. Regeneration datasets are also COS blocked.

1. Disk File Management in CCM2

As on many supercomputer systems, disk space on the CRAY Y-MP is a precious commodity. To enhance usability of the temporary file system, the Cray system staff has made several logical divisions in the temporary disk space. The Model interacts with two of these—the job temporary space known as $TMPDIR and the user temporary space allocated under the directory /usr/tmp.

The directory pointed to by system environment variable $TMPDIR actually resides in /usr/tmp. It is created as a unique directory at login time. For batch jobs, this is the time at which the queue management subsystem NQS starts the job on the Cray. This directory exists only until the batch run script exits, at which time it and all its file links disappear.

The user temporary disk space, /usr/tmp/$LOGNAME, remain after job completion, but is subject to being “scrubbed” of individual files, based presumably on size and age of the file. At times, this translates into a retention time of only a few minutes!

By making a $TMPDIR directory the “current directory” for the Model run, we assure that all local files are protected from scrubbing. The user may choose, via namelist parameter LDEBUG, to link the final set of output files from a run into /usr/tmp/$LOGNAME, (see Table I.1 on page 14) to possibly avoid an additional stage-up from the Mass Store to the Cray disk when performing post-analysis of the history tape, for instance.

Most of the logical unit numbers in CCM2 are determined at execution time. Only the SSD work units are assigned in the run script. Model utility routine NAVU keeps a catalog of logical unit numbers in use and when
called returns an unused number. After all units are assigned, a table of unit numbers and their usage is printed in the Model output.

A heavily used file system such as the temporary disk system on the Cray can become very fragmented. Large output files such as the history tape are slow to read when this happens. For this reason, the Model preallocates space equal to the expected length of the output file at assign time. It turns out that the -n option on the assign command does not do a good job of finding contiguous space for preallocation—the only command which will do this is setf -c. However, this method returns a fatal error if there is not enough contiguous space available for the entire file. Therefore, the Model uses an iterative procedure, in subroutine PREALC, to find as much contiguous space as possible for an output file. This method is used in assigning all large output datasets, including the history files, and the primary and secondary regeneration files.

The Model is able to call system routine mswrite with the nowait parameter, even though the local files reside on $TMPDIR, due to the way in which asynchronous mswrite works. This routine will establish a link (ln) for the output file into a protected directory owned by the system before it returns control to the calling program. Thus, even if the Model script exits before all output files reach the Mass Store and $TMPDIR and its files are no longer accessible, the links to the protected directory preserve the files until they are actually written to the Mass Store.

2. Use of the NCAR Mass Storage System

The Model uses the NCAR Mass Store Subsystem (MSS) for archival storage of all input and output datasets. Input datasets are staged to the Cray disk from the MSS and accessed by the Model via the system Fortran-callable msread. The output history tapes and restart and regeneration datasets are disposed to the MSS via Fortran-callable mswrite. Routines that perform mass store operations are isolated in the Model to facilitate replacement when the Model is moved to another system.

The Model issues msread requests as synchronous operations, that is, the Model will wait until the read is complete before resuming execution. By default, the Model issues asynchronous mswrite commands, although the user may request synchronous writes via the namelist parameter ASYNC. Because output volumes are written to the Mass Store System only after they are complete and the next volume will have a different name, there is no danger of overwriting a volume before it actually gets to the Mass Store. No provision exists for read passwords on Model output volumes.

Output volume naming in CCM2 is entirely automatic and may not be overridden by input parameter settings. Output files will be written to disk in the directory pointed to by $TMPDIR/$CASE and then linked to the...
Linking into /usr/tmp

The history file will be linked into the directory

```
/usr/tmp/ccm/$NEWLOG/ccm2/$CASE/hist
```

and the restart and regeneration datasets into directory

```
/usr/tmp/$LOGNAME/ccm2/$CASE/rest
```

where:

- `$LOGNAME` login name UNICOS environment variable,
- `$NEWLOG` upper case form of `$LOGNAME`,
- `$CASE` a case name of up to 8 alphanumeric characters, specified by setting environment variable `$CASE` in the run script.

History tape files are named as follows:

For the primary history tape, “h” plus 4 digits, starting by default with 0001, 0002, .... For example, h0004 is the fourth history tape written for a particular case, assuming that input parameter STFNUM, described below, takes the default value of one.

For auxiliary history tapes, “hx” plus the same 4 digits,

where,

- “x” is a lower-case alphabetic character, starting with “a,” e.g., ha0004. If namelist parameter STFNUM is entered, the 4-digit numbers will start with the value of STFNUM.

Regeneration dataset names are:

For the master regeneration dataset, “r” plus 4 digits starting 0001, 0002, etc., by default, or starting with STFNUM as above.
For primary regeneration files, r0001A, r0002A, etc.

For secondary regeneration files (absorptivity/emissivity data) r0001a, r0002a, etc.

If a single regeneration dataset would exceed namelist parameter MXSZRG in length, the Model will split the data across two or more datasets, as needed. For the primary regeneration files, these datasets will take the names r0001B, r0002B, and r0001C, r0002C, etc., as required. For secondary datasets, if needed, the names would be r0001b, r0002b and so on.

The restart dataset name is input as namelist parameter NSVSN. This file is overwritten at each history tape dispose time and contains the Mass Store pathname of the current master regeneration dataset.

The Cray disk path /usr/tmp/ccm is used as a working directory by the CCM Modular Processor, PROC02. History tapes are linked there to facilitate subsequent analysis by the Processor without unnecessary Mass Store staging. Please note, however, that there is no guaranteed retention time on the /usr/tmp file system.

Mass store directories for Model output files are:

$/NEWLOG/ccm2/case/hist for history tapes, and

$/NEWLOG/ccm2/case/rest for restart and regeneration datasets,

where

$NEWLOG is the upper-case form of $LOGNAME, i.e., the user’s root name on the Mass Store system.

3. Issuing Shell Commands from the Model

File operations that cannot be handled with Fortran library calls are accomplished by issuing shell commands via system routine ishell. Calling ishell would replicate the job field length, except that the Model is run under another system “shell” called pshell. The number of ishell calls in CCM2 is minimized by making use of the Fortran interface to commands such as mswrite. However, as mentioned above, it is advantageous to preallocate disk space for large files such as history and regeneration volumes. The only way to do this reliably is with the system setf command, which has no Fortran interface.

System-specific calls in CCM2 are isolated to a few areas of the code, making the Model more easily portable to other computers. Table II.1 below shows the system routines required by CCM2 and the Model routines from which they are called.
<table>
<thead>
<tr>
<th>System Routine</th>
<th>Description</th>
<th>Model Routines</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Local File Interface</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>assign</td>
<td>Attach local file to job</td>
<td>atchbnd, attach, restrt, rgnfls, wrapup</td>
</tr>
<tr>
<td>setf</td>
<td>Preallocate space on disk for local file</td>
<td>prealc</td>
</tr>
<tr>
<td><strong>Mass Store Interface</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>msread</td>
<td>Read from Mass Store</td>
<td>attach</td>
</tr>
<tr>
<td>merror</td>
<td>Print Mass Store errors</td>
<td>attach, savdis</td>
</tr>
<tr>
<td>mswrite</td>
<td>Write to Mass Store</td>
<td>savdis</td>
</tr>
<tr>
<td><strong>Interface with Run Script</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>getenv</td>
<td>Get value of environment variable</td>
<td>data, igtseq</td>
</tr>
<tr>
<td><strong>Miscellaneous System Functions</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>abort</td>
<td>Fatal error stop, issue traceback</td>
<td>endrun</td>
</tr>
<tr>
<td>date</td>
<td>Return current date as character string</td>
<td>ccm2, inthed, wrthdr</td>
</tr>
<tr>
<td>clock</td>
<td>Return current time as character string</td>
<td>ccm2, inthed, wrthdr</td>
</tr>
<tr>
<td>ishell</td>
<td>Issue system (shell) command</td>
<td>iostop (for error processing) prealc (for disk space preallocation) savdis (to remove and link local files)</td>
</tr>
</tbody>
</table>
D. Continuation Run Logic

CCM2 is often run for very long simulations, requiring CPU time limits that may exceed the mean time between failures, due to environmental problems, system maintenance, etc., on a computer. Therefore, control logic within the Model allows it to restart when a case terminates, whether normally or abnormally, and continue running with little or no loss of resources.

In addition, when starting a long Model run, it is a very good idea to run your script as a 1-day initial run, then restart the case for the remainder of the experiment. Then, if the system does an automatic rerun of your job, it will start from the latest regeneration files, rather than rerunning from the beginning of the case.

There are three different kinds of continuation runs:

- **Restart run**: A “restart” run (*namelist* parameter NSREST = 1) starts from the last point at which output was disposed, and is exact, that is, the model results for a restart run will be identical to the results of the case if it had not been interrupted.

- **Regeneration run**: A “regeneration” run (*namelist* parameter NSREST = 2) may start from any point in the case at which output was disposed and is also exact. It is usually used to regenerate one or more history volumes that may have been corrupted since the case was run and requires that appropriate regeneration datasets from the same case exist. Regeneration is also useful for “backing up” when the most recently written regeneration datasets are corrupted, making a restart run impossible.

- **Branch run**: A “branch” run (*namelist* parameter NSREST = 3) uses the contents of a specified regeneration dataset to start the Model on a new case. The history tape for this case is built “from scratch,” so it may differ in content, packing density, or any other way from history files of the original case. For example, the user may use a branch run to change the history tape in order to resample subperiods of the original case.

The Model simulation from a branch run will be the same as that from the original run. Because the branch run starts from a “snapshot” (in the regeneration dataset) of the work files, changes affecting the length of the buffer, the three-dimensional arrays, etc.—such as adding constituents—are not allowed.

1. **Regeneration Datasets**

In order to provide this “check-pointing” capability (not to be confused with UNICOS system checkpointing), the Model periodically saves the information required to restart the case and writes it to separate files known as regeneration datasets. High-resolution runs of the Model may generate
work units which, when combined, would exceed the maximum Mass Store file size of 190 MB. Therefore, based on resolution parameters, the Model automatically determines the size of these datasets and divides them into separate units as required. The default maximum file size allowed is 150 MB, but the user may override the default by specifying namelist parameter MXSZRG.

Clearly, there is a trade-off to be considered when creating regeneration datasets. The amount of data disposed to the Mass Store can become very large, the default being to write a full set of regeneration datasets every time a history tape is disposed. In high-resolution runs or runs that write the history tape very frequently, this can become untenable. Therefore, a namelist variable, NREFRQ, is provided to specify that regeneration data be written only for each n history tapes, where n=NREFRQ. Thus, the resources lost on restart may be greater, but the amount of data stored is much less.

As described in “Restart/Regeneration Datasets” on page 53, the restart dataset contains a full Mass Store pathname, pointing to a “master” regeneration dataset. This dataset, on unit nrg, contains variables from common blocks /com3d/, /coms1t/, /comtim/, /comhst/, /comhed/, /comqfl/ and /crdsrf/. The data on work units nrbl and nral are written to as many “primary” regeneration datasets as are necessary such that the datasets do not exceed MXSZRG in size, using units specified by /comlun/ array nrg1.

If the specified history tape write frequency is not an even multiple of the absorptivity/emissivity calculation frequency, as determined by namelist parameter IRADAE, it is necessary to write the absorptivity/emissivity arrays to a regeneration dataset. In this case, these data are written to as many “secondary” regeneration datasets as necessary, using logical unit numbers from /comlun/ array nrg2.

If NREFRQ=1, one set of regeneration datasets is created for each history tape written, so that for a run which generated history tapes h0001, h0002, and h0003, there will also exist master regeneration datasets r0001, r0002, and r0003 and primary regeneration datasets r0001.A, r0002.A, and r0003.A, and, if all data did not fit on files of size MXSZRG, also r0001.B, etc. If absorptivity/emissivity data are saved, they will be on files r0001.a, r0002.a, etc., and possibly r0001.b, etc.

These units contain copies of the work files and selected common variables as they existed at the time of the Mass Store write of the corresponding history tape. They are disposed to the Mass Store Subsystem at the same time as is the history tape, i.e., when the history tape is “full,” as determined by the namelist input variable MFILT. (See “Model Input Parameters” on page 13.) At timesteps for which restart information is to be saved, subrou...
tine SCAN1 calls WRTRS1 to copy the work units. At history tape write
time, subroutine WRITUP calls WSDS to write the common blocks to unit
nrg. Then the history tape and all regeneration datasets are transferred to
the Mass Store.

2. How to Make Continuation Runs

A Model restart is accomplished by resubmitting the same Model job deck
that produced the initial run with the namelist parameter NSREST set to 1
and parameter NESTEP changed if the ending time of the run is to be
changed. Internal flag nlres in /comctl/ is set to .true. The Model
then reads the “restart dataset,” which contains the Mass Store pathname of
the latest regeneration datasets to start the Model.

The continuation procedure initializes the Model from these datasets,
positions the history tape(s) properly, and continues the run. A restart must take
place from the most recently saved data sets, as the restart dataset itself is
continually overwritten by the Model. If the case terminates normally (runs
to the specified ending timestep NESTEP), it can be restarted with essen-
tially no loss of resources. If the Model terminates abnormally (e.g., hard-
ware failure, time limit exceeded, etc.), the computer time since the last
Mass Store transfer of the regeneration datasets will be lost. Note that if
you expect to continue a case, you should specify the ending time step of
the run to be coincidental with a history tape write.

A regeneration run requires that NSREST be set to 2, and an additional
parameter, NREVSN, be set to the name of the regeneration file from which
to start. Thus, to regenerate history tape “h0002,” NREVSN should be set
to “r0001.” Internal flag nlhst in /comctl/ will be set to .true. The Model
will position and read all regeneration datasets associated with
the first history tape. It also reads through the first history tape, retaining
the most recent header information. It then assigns new history and regen-
eration datasets, and starts as in the restart run.

A branch run is specified by setting nsrest to 3, and nrevsn to the full
Mass Store pathname of the regeneration dataset from which to start. If the
user wishes to branch from regeneration files r0024, r0024.A, and
r0024.B from case “ccmtest”, he/she should specify nrevsn as fol-

\[
NREVSN = '/$NEWLOG/ccm2/ccmtest/rest/r0024'
\]

The case name (environment variable $CASE) in the CCM2 run script
must be changed; the Model will stop if it is not.

Also note that NESTEP is an absolute timestep, starting from the original
initial run. In the preceding example, if r0024 contained data from model
timestep 485, the user must set input parameter NESTEP to be greater than 485. As noted in “Running the Model” on page 1, namelist parameter NELAPSE may be used instead of NESTEP, specifying an elapsed time to run. Input parameters from the columns labeled “Mass Store Information,” “History Tape Options,” and “Regeneration Options” in Table I.1 on page 14, may be changed as desired. Auxiliary history tapes may be added or removed. The branch run was implemented to enable the user to run the Model in the standard configuration up to the time of interest, then change parameters such as write frequency, fields on the tape, etc., for a continuation of the experiment.

In general, the frequency of Mass Store transfers for restart and history files, as specified by MFILT, should be small enough so that the amount of time lost by a continuation run is not too great (perhaps 30-60 minutes of CPU time) and large enough that the overhead incurred in writing to the Mass Store is not unreasonable.

Attempts to regenerate volumes that were initially generated prior to major operating system or compiler changes on the Cray are not guaranteed to be successful. This fact limits the usefulness of regeneration datasets to a maximum of two years (the usual time between major compiler releases). For this reason, a separate namelist input parameter, RIRT, is provided to specify retention time for regeneration data on the Mass Store. It is recommended that RIRT be set to no more than 730 days.
E. Model Code Flow

This section provides a narrative of the Model code flow through the initialization phase and the computational loops. The graphical calling trees (illustrated in Figure I.1 through Figure I.4 starting on page xviii) and the schematic diagrams (in Figure II.5 and Figure II.6 below) provide helpful reference for reading this section. Details of the physical parameterizations in CCM2 are not presented here—these are documented in the NCAR Technical Note *Description of the NCAR Community Climate Model (CCM2)* (Hack et al., 1992).

![Diagram of Model Code Flow](image)

*Figure II.5. Use of SSD Work Units in Initialization Phase.*
1. Initialization

The first tasks performed by the Model are defining the logical units used for I/O (subroutine LUNITS), presetting namelist variables (subroutine PRESET) and reading in the Fortran namelist data (subroutine DATA), and setting pointers and lengths associated with the main Model buffer (subroutine POINTS).

From this point on, initialization takes one of two paths: initial runs are set up by subroutine INITIAL and the routines beneath INITIAL in the calling tree; all continuation runs are initialized by subroutine RESUME. These two paths are roughly parallel in nature and call many of the same routines. The key difference is that the initial branch must read the initial dataset, transform the data, and do the first I/O to the SSD work files (see subroutine INIDAT, below). The continuation branch must initialize Model data structures from the appropriate regeneration files, again doing first I/O to the work files.
INIDAT first reads input variables ps, u, v, t, and the surface geopotential (phis) from the initial dataset. It then invokes the Temperton Fast Fourier Transform package (fft991) to take these fields into Fourier space. INIDAT calls SPETRU to spectrally truncate these data. Surface pressure derivative fields dps, dpsm, and dpsl, as well as vorticity and divergence, are also initialized to their spectrally truncated values in SPETRU. After return from SPETRU, INIDAT transforms the spectrally truncated fields back into gridpoint space by a second call to fft991.

The moisture field q and any additional constituents being transported are then read in from the initial dataset and copied into common block /com3d/. Moisture and constituents are not spectrally truncated; instead, subroutine QNEG3 checks to make sure all values exceed some minimum threshold.

INIDAT also reads in the surface type flag (oro) and the subsurface temperature fields. The initial values of the prognostic variables are copied to both time levels of the in-core arrays in /com3d/ in order to facilitate a forward timestep at Model start. All subsequent timesteps are "leapfrog," a centered time-differencing scheme. Temporarily halving the value of 2*(delta-t) at Model start (nstep=0) allows the same code to be exercised for both the initial forward and the successive leapfrog timesteps. This temporary halving is done in subroutine STEPON.

Since some fields initialized in INIDAT are held in the main Model buffer, INIDAT must also do the initial I/O to the requisite work file units. It then computes global integrals of dry mass and moisture which are needed later in the code.

INIDAT is not multitasked. It is only executed at start-up time and does not represent a sufficient portion of the entire Model computation time to warrant the effort of multitasking.

The local (stack-based) database of INIDAT is quite large. Three-dimensional arrays exist for vorticity, divergence, and the work array required by the fft package. This is partly due to the fact that the Model can start from a randomly ordered (in latitude) initial dataset. Since local memory is allocated on the stack (see "Memory Management" on page 112), the large size of the local database does not increase total memory utilization by CCM2 since the local workspace in INIDAT will disappear before the time integration begins.

2. Computational Loops: Code and Data Flow

This section describes data flow and code flow through the timestepping phase of CCM2. The order of routines discussed moves across, rather than down, the calling tree. Traversal of the CCM2 calling tree stops at the level of individual parameterizations, e.g., the bottom row of routines in Figure I.2 on page xx. Discussion of these parameterizations is contained
in the Description of the NCAR Community Climate Model (CCM2) (Hack et al., 1992) and in the main comment blocks in the individual routines themselves. Where applicable, local data structures are also discussed. Refer to Figure I.1 through Figure I.4 (starting on page xviii) and Figure II.5 on page 100.

(a) Time Integration

Subroutine STEPON controls the timestepping in CCM2. After the initialization phase is complete for either an initial run or a continuation run, STEPON loops through the time integration to completion of the simulation, incrementing /comtim/ variable nstep after completing each Model iteration. STEPON first calls the SLT time-invariant initialization routine GRD INIT and then calls the routines that perform the three Gaussian latitude scans. These are SCAN1, SCANSLT, and SCAN2.

DYNDRV is the other computationally intensive routine called from STEPON. This spectral space routine drives multitasked loops that are parallelized over total wavenumber index “n” and vertical level index “k.”

The sequence SCAN1, SCANSLT, DYNDRV, SCAN2 comprises one Model timestep. If the Model terminates normally, SCAN1 will be the last routine called in this sequence, as well as the first to be executed upon restarting the Model. The only reason to call SCAN1 at the end of a run is to write the fully time filtered data to the history tape buffer, completing a time accumulation period for the history tape. Thus, the history tape buffer need not be written to the regeneration dataset.

A three-dimensional array (longitude, level, latitude) called wfld is defined locally in STEPON and contains the vertical motion field used in the SLT package. wfld must be declared this high in the calling tree because its computation is most efficiently done in the SCAN1 branch, though it is only needed in SCANSLT (wfld is computed as “etadot” in subroutine GRMULT). Numerous other SLT-specific arrays are defined local to STEPON. They must be defined here because they are computed in the grid initialization routine GRD INIT (called from STEPON) as well as in the SLT package itself.

Fourier coefficient arrays are also defined locally in STEPON. The values are computed in LINEMS (below SCAN1) and used in the Gaussian quadrature routine (QUAD) which is below DYNDRV. It is not possible to both define and use these arrays only in the SCAN1 branch of the calling tree because of multitasking considerations. Gaussian quadrature requires a summation over latitude. Therefore, the multitasking for this part must be done over a different index. A separate multitasked loop is required where the parallelization is over a different array index. In CCM2, the quadrature is parallelized over total wavenumber “n,” and invoked from DYNDRV.
Subroutine ADVNCE is called from SCAN1 for each timestep, outside of the multitasked latitude loop. ADVNCE drives the updating of current time information (CALDYI) and time-variant boundary dataset information (SSTINT and OZINT).

Current model time information is computed in CALDYI. From the standpoint of model computations, the output variable calday (current Julian day plus fraction) is all that is required for the time integration. However, other variables having to do with current date and current day are also output from CALDYI because they are required by the history tape handler. Note that there are no leap years in the Model; the definition of a “year” is 365 days.

(b) Latitude Scans

The first Gaussian latitude loop is done in SCAN1. Global integrals are initialized, the history tape header and restart files written (if applicable for the current timestep), and current model time information is computed prior to execution of the multitasked loop over latitude pairs. The loop over latitude pairs in SCAN1 is where all the model physics, some of the dynamics, and history tape computations are done. After this loop the Courant limiter is applied. Root mean square vorticity, divergence, and temperature, along with global integrals of surface pressure and moisture are then computed (subroutine STATS) and printed along with the current timestep index and maximum wind.

SCANSLT is called after SCAN1. This is the only latitude scan in CCM2 in which the full dimensionality of the “extended grid” (plond and platd) is required. In addition to the physical data locations (plon and plat), the extended grid includes additional array storage before the start and after the end of the physical data in both the longitudinal and latitudinal dimensions. The extended grid must be initialized at each timestep. This is done in SLTINI which is called as the first executable statement in SCANSLT.

Next comes the main multitasked latitude loop. Here the un-time-filtered wind field at time level “n” is used to predict the evolution of the time-filtered moisture and (if applicable) constituent fields, from time n−1 to time n+1 using the semi-Lagrangian transport scheme of Williamson and Rasch (see Description of the NCAR Community Climate Model (CCM2) (Hack et al., 1992)). Forecast expense in terms of computer time for additional constituents is minimal. Each iteration of the latitude loop produces a forecast of constituent concentration which is stored in array qfcst. No I/O is required in SCANSLT, since the prognostic arrays u3, v3, wfld and q3 are entirely in core (see “Time Integration” on page 103).

The local database in SCANSLT contains numerous multi-dimensional arrays. These arrays (ux1, uxr, qx1, qxr) contain spatial derivatives of the wind and constituent fields. In the default Model configuration (18-level T42, with water vapor the only constituent) this large local database
does not have a great effect on the memory hi-water mark. However, this situation changes rapidly as additional constituents are added and the size of the arrays qxl and qxr overwhelms the size of the local database in the biggest branch in the SCAN1 calling tree.

When additional constituents are transported, the user will find that varying the value of the environment variable $NCPUS will have little or no effect on the high-water mark for the run. Thus, the maximum possible value of $NCPUS should be chosen when running CCM2 with multiple constituents. For more details about memory management in the multitasked environment, refer to “Memory Management” on page 112.

SLTINI fills the extensions of certain arrays used by the SLT package in the longitudinal and latitudinal dimensions. Only a small amount of computational work is performed in SLTINI, but the routine is multitasked over latitude bands since it is called every timestep.

The primary function of SCAN2 is as a driving routine for the conversion of spectral space prognostic variables back to gridpoint space and final computation of global integrals of mass and moisture integrals for the timestep. As in SCAN1, the multitasked latitude loop within SCAN2 is over latitude pairs rather than individual latitudes in order to account for the symmetric properties of spectral coefficients for north-south Gaussian latitude pairs. The last step in SCAN2 is to toggle the time indices n3 and n3ml after the conversion back to gridpoint space is complete. No copying of the actual data is necessary. The prognostic data that were pointed to by the current time index (n3) will become the previous time index (n3ml), and the data just computed in time index n3ml will become the “next” (n3) time index at the start of the next iteration in STEPON.

(c) Latitude Pair Driving Routines

In addition to driving latitude-independent computations, routines included under this heading perform the requisite I/O calls to the main Model buffer out-of-core work file units and the history buffer work file unit.

LINDRV exists solely for the purpose of synchronizing computations in LINEMS for individual latitude pairs. LINDRV first reads in main Model buffer data from the work file units nral, nrbl, and the history buffer work file unit sunit within the latitude pair loop. The SLT constituent forecast (contained in array “qfcst”) for the current latitude is then copied into the proper location in the model q array. The copy is done here rather than in SCAN2 of the previous timestep because routines below SCAN2 require the old data in that time level of q. After calling routine LINEMS (discussed below), LINDRV writes data from the Model buffer to work file unit nrbl and from the history tape buffer to unit sunit.

The history buffer is defined locally in LINDRV. It is the one and only array in CCM2 for which the space is truly dynamically allocated. See the dis-
discussion on the history buffer under the heading “History Buffer” on page 83 for details. Other locally defined workspace in this routine includes the main Model buffers b1 and b2 and the spectral arrays grut1, grut2, gruq1, and gruq2, which are needed in LINEMS but not in the dynamics part of the code driven by DYNDRV.

SPEGRD drives the inverse Legendre transform from spectral to Fourier space (GRCALC) and the inverse Fourier transform from Fourier space back to gridpoint space for a given north-south Gaussian latitude pair. Like LINDRV, it owes its existence to the multitasking of north-south latitude pairs. SPEGRD writes the transformed data to the work file unit nral for subsequent input to LINDRV at the next time iteration.

The Fortran pointer statement is utilized in SPEGRD (but nowhere else in CCM2) in order that individual arrays contained in Model buffer “buf” can be addressed as appropriately dimensioned individual arrays (see “Model Buffer” on page 80 for details). In addition to the Model buffer, local workspace is defined in SPEGRD for the fft package as well as symmetric and antisymmetric Fourier coefficients.

(d) Spectral Space Computations Driving Routine

Gaussian quadrature, completion of the semi-implicit timestep, and horizontal diffusion calculations are all accomplished within multitasked loops in DYNDRV. The quadrature and semi-implicit timestep computations are parallelized over total wavenumber “n,” and are done sequentially in the same loop. Horizontal diffusion calculations are parallelized over the vertical index “k,” and, therefore, require a separate loop. No I/O takes place in the DYNDRV branch of CCM2.

(e) Computational Driving Routines at the Latitude Loop Level

LINEMS performs many of the same functions it did in CCM1, such as calling the adjustment physics, APHYS, time filtering, and calling the tendency physics driver TPHYS. Additional functions include computation of some SLT quantities for output to the history tape, application of mass and constituent fixers, and checking for too small constituent concentrations after various model phases. Much of the code in LINEMS is devoted to pre-computing certain arrays for input to various physics packages. Much effort was spent eliminating needless duplicate computations in separate areas of the code in CCM2. The “precomputation” of many arrays in LINEMS which are then passed to lower-level routines is a manifestation of this effort.

There are two “physics” drivers called from LINEMS. The first of these is APHYS, which controls physics applied in an adjustment fashion. Adjustment is the first of the physical parameterizations applied during a timestep. Initial data written to the history tape go through adjustment.
A time filter is applied to the prognostic variables just as in CCM1, but in CCM2 it is applied as in-line code instead of as a separate subroutine. All but vorticity and divergence are now time filtered in one step as opposed to two. Vorticity and divergence are most efficiently time filtered in a two-step procedure because these fields are still maintained on an out-of-core file unit.

Gridpoint-to-Fourier space calculations are performed in LINEMS just as they were in CCM1. The remainder of the spectral space computations are left to the dynamics driving routine DYNDRV due to multitasking considerations (see the discussion of this topic under STEPON, above).

LINEMS also converts the wind field between straight meters/second and meters/second multiplied by cosine(latitude). These conversions are necessary because the SLT code uses the direct meters/second form while most of the rest of the model uses the cosine(latitude) form.

**TPHYS**

Most of the CCM2 physics packages are called from TPHYS. This routine controls cloud computations, radiation, surface temperature and flux calculations, the planetary boundary-layer scheme, vertical diffusion, rayleigh friction, and gravity wave drag. Cloud calculations are needed internally in the model by only the radiation code. Since radiative computations are normally not performed every timestep (IRAD > 1), clouds are calculated only on timesteps when the radiative transfer code is exercised.

**RADCTL**

RADCTL controls the radiative transfer code. Units are cgs while the rest of the model uses mks units. RADOUT does the conversion from cgs to mks for requisite radiation fields written to the history tape. Frequency of longwave absorptivity and emissivity calculations is controlled by onamelist variable IRADAE. This variable must be an even multiple of IRAD, that is, the absorptivities and emissivities can be calculated only on a radiation timestep. In addition, IRADAE should be an even multiple of the history tape write frequency, NHTFRQ, in order that the absorptivities and emissivities need not be written to the restart dataset.

The absorptivities and emissivities are maintained on an out-of-core file unit since IRADAE is likely to be greater than IRAD. If we are on an absorptivity/emissivity (a/e) timestep, these data are computed and written to work unit nabem. If we are on a non-a/e radiation timestep, they are
read in from out-of-core unit nabem. Control of this computation and I/O is done in RADCLW, which is called from RADCTL.

SLTB1 drives the SLT constituent forecast calculations. Each call to this routine fills one latitude line of three-dimensional array qfcst. For more detail on the algorithms of the SLT, reference Description of the NCAR Community Climate Model (CCM2) (Hack et al., 1992).
F. Multitasking Strategy

CCM2 was designed to run most efficiently on machines built by Cray Research, Inc. (CRI). Since about 1985, technological advances in CRI hardware have been focused on the addition of more CPUs and memory to their shared memory multiprocessing computers. To take advantage of this multiprocessing capability, it was decided early on to make CCM2 a multitaskable model, with the details of implementation transparent to the user. To run multitasked, the user need only specify the number of processors to use by setting the environment variable $NCPUS to that number. If $NCPUS is not set, the Model will run "single-threaded," i.e., on a single processor.

On a dedicated machine, turnaround time for a multitasked CCM2 should be faster than its single-threaded equivalent by almost a factor of $NCPUS. This is because the Model is over 99% parallel, not counting initialization. In practice, even in a nondedicated environment, turnaround should improve for a multitasked job. Actual speedup will most often be substantially less than a factor of $NCPUS, however, because of competition for memory and processor access from other users. In a dedicated environment, $NCPUS should always be set equal to the number of physical processors on the machine. A general rule of thumb when running CCM2 multitasked in a multiprogrammed, i.e., not dedicated, environment is to set $NCPUS to half this number or less.

The above discussion assumes availability of a solid-state storage device (SSD) for all of the scratch file I/O. If the out-of-core work files are written to disk instead, performance enhancement for a multitasked job will not be nearly as dramatic because disk I/O is approximately three orders of magnitude slower than SSD I/O, and I/O to a given unit is always single-threaded.

One prerequisite in the design of the Model was that an identical simulation be guaranteed whether running in multitasked or single-threaded mode. To accomplish this goal, certain summations of critical quantities are always single-threaded, e.g., the moisture summation performed by QMASSD in SPEGRD. A CCM2 regeneration run produced under the same compiler and operating system release as the original run produces an identical simulation, i.e., the history tapes from the two runs compare bit for bit. When adding new code to the Model, the user should always ensure that deterministic results are obtained when running multitasked. Side-by-side multitasked vs. single-threaded one-day simulations should be sufficient to answer this question.

1. Multitasking Implementation

CCM2 is multitasked using Cray "Autotasking." An advantage of autotasking over other multitasking products is that it introduces no portability
problems. All requisite information to autotask CCM2 is contained in directives which take the form of Fortran comments.

Although the autotasking preprocessor is designed as an “automatic multitasker,” CCM2 employs neither its automatic data scoping nor its automatic multitasking features. Data-scoping directives produced by the autotasker often proved to be incorrect, and the fine granularity of the automatic multitasking resulted in unacceptable system overhead. Also, identical answers could not be obtained on single-threaded vs. multitasked runs. As a result, the only features of the autotasker which are utilized in CCM2 are those which 1) indicate where iterations of “do loops” may be run in parallel, and 2) automatically generate system calls to parcel out work to available processors and synchronize at the end of the loop.

Autotasking directives are placed very high in the calling tree telling a pre-compiler (called fmp) to autotask specific loops. Data scoping is done explicitly in these directives, which are of the form:

\[
\text{CMIC}$\ DO\ \text {ALL SHARED ( . . ) PRIVATE ( . . ).}
\]

The DO ALL portion of the directive tells fmp that each iteration of the next loop can be done independently.

Names within the parentheses after SHARED indicate those variables that are global to the loop and that can be shared across multiple processors. Conversely, variables declared as PRIVATE must have separate storage for each processor. Examples of variables that may be shared include those which are read-only within the loop, or which have separate storage already allocated for each iteration of the loop. Variables taking different values during separate iterations of the loop must be private. The loop index itself is a good example of such a variable. Autotasking compiler directives exist only in routines DYNDRV, SCAN1, SCAN2, SCANSLT, and SLTINI. These routines drive all the physics and dynamics in CCM2.

In any multitasked code, work spun off to one processor must be independent of the work being done (potentially) simultaneously on all other processors. In CCM2, iterations of the Gaussian latitude loops in SCAN1 and SCAN2 are independent of north-south latitude pair. Therefore, each of these routines contains a multitasked loop of the form:

\[
\text{CMIC}$\ DO\ \text {ALL SHARED ( . . ) PRIVATE ( . . )}
\text {do irow=1, plat/2}
\text {...}
\text {end do}
\]
Computations within the loop for each value of \texttt{irow} are on separate logical processors. If the number of logical processors exceeds the number of physical processors, (this is certainly true for T42 CCM2 run on an 8 processor CRAY Y-MP), iterations not assigned a processor must wait until a processor becomes available before they will execute.

In the SLT routines which drive multitasked loops (SCANSLT and SLTINI), each latitudinal iteration may be done in parallel. In the spectral dynamics driven by DYNDRV, the Gaussian quadrature and semi-implicit timestep computations are parallelized over diagonals of the spectral truncation (remember that diagonal elements of the spectral arrays are stored contiguous to one another; see “Spectral Data Structures” on page 86). DYNDRV also drives the horizontal diffusion calculations, which are parallelized over model level.

It is important to realize that there is no guarantee of the order in which multitasked iterations of a loop will be either initiated or completed. For this reason, coding constructs of the form,

\begin{verbatim}
subroutine xxx(lat)
  if (lat.eq.1) then
    ... code to initialize static variables ...
  end if
\end{verbatim}

will not work inside multitasked portions of the Model. If \texttt{lat} = 2 happens to be the first process to reach routine \texttt{xxx}, the static workspace of the routine will not be properly initialized. To guarantee that routine \texttt{xxx} will work properly when multitasked, the static variables it uses must be set in a single-threaded part of the code before it is invoked.

Another result of the unpredictability of calculation order under multitasking was that direct access I/O to the work file units became necessary. If sequential access were utilized, there would be no guarantee that a given processor’s request for a latitude band of data would fall in the correct order.

Sequential I/O is still used for the history tape, even though the output order of latitude bands is totally unpredictable in a multitasked run. Sequential I/O is necessary because the history file contains variable-length records, and standard Fortran direct access I/O requires fixed-length records. To identify the latitude bands, we include the latitude index of each band as the first value in each data record in the history file. This is the primary reason a new format type became necessary for CCM2 history tapes.

There is one place in CCM2 where a particular routine owes its existence to multitasking considerations: the only purpose for subroutine LINDRV is...
to ensure that iterations of individual latitude pairs are executed sequentially in routine LINEMS. To guarantee deterministic results, sequential execution is necessary in accumulating the symmetric and antisymmetric spectral coefficients. LINDRV has a "iter=1, 2" (the 2 halves of a latitude pair) loop which is guaranteed to run sequentially because it is itself contained within a multitasked region of the code, i.e., both trips through the loop for any latitude pair will be done by the same processor.

In most multitasked codes, it is necessary to ensure that certain regions within that code are exercised single-threaded (i.e., that only one processor is executing that part of the code at any time). In CCM2 this is true when writing the history tape. The history-file data would certainly be unreadable if more than one processor were trying to write their sequential records simultaneously! Under the current operating system release, standard Fortran I/O is automatically guarded against multiple-processor access to a single unit. Therefore, no explicit "CMICS$ GUARD" statements are necessary.

2. Memory Management

A stack-based memory management scheme was a crucial element in the design of CCM2. When data are allocated on the stack, their lifetime is limited to the time during which the module (subroutine or function) where the data structure is declared is active. In other words, storage for locally defined workspace in a given routine disappears upon execution of the return statement. Stack-based local memory allocation makes for more efficient memory usage since memory is allocated to each module only when required. Also, it makes for more readable code because one needn’t be concerned about local variables outside of the declaring routine.

The second reason why stack-based memory allocation is used in CCM2 has to do specifically with multitasking. Unlike static memory, stack memory declared within multitasked regions of code is automatically replicated across processors. Thus, use of the stack is the only reasonable way to ensure that multitasked routines are "reentrant," i.e., they may be entered by more than one processor at once without violating previously defined data space.

For example, consider array pmid declared locally in subroutine LINEMS. LINEMS exists within the main multitasked loop in SCAN1. pmid is dimensioned (plond, plev) with no latitude index because each latitude, by virtue of its execution on a separate processor, has its own copy of pmid.

There are a few places in CCM2 where locally defined memory is statically allocated within a multitasked region of the code. This is done for some read-only memory mainly to avoid having to reset certain variables every time a routine is invoked. These variables may be declared static via a For-
tran save statement, or by virtue of appearing in a data statement. Users who change multitasked routines should beware of resetting these statically allocated memory locations. We recommend that the user read relevant sections of the CF77 Compiling System, Volume 4: Parallel Processing Guide, SG-3074 before attempting to alter the multitasked code.

One negative ramification of running CCM2 on multiple processors is that memory usage is increased as compared to a single-threaded run. A small amount of overhead is created by the autotasking libraries themselves. The primary culprit, however, is the stack memory replication across processors as just discussed. To determine quite precisely how much extra memory will be required per processor, one need only look at the output from segldr -M, s for a multitasked run. Printout will be of the form:

Program Statistics
Non-segmented object module written to- ccm.xx.21307
Allocation order- text, data, bss
Program origin- 0 octal 0 decimal
Program length- 30600521 octal 6488401 decimal
Start entry point is "$START" at address 0a
Transfer is to entry point 'CCM2' at address 51730d

Program is multitasked, targetted for 4 CPUs
Task head entry points-
DYNDRV0196
DYNDRV0211
SCAN10570
SCAN20216
SCANS1T8293
S1TINI0134

Managed memory statistics
Initial stack size- 10256607 octal 2186631 decimal
Stack increment size- 400 octal 256 decimal
Initial task stack size- 1404050 octal 395304 decimal
Task stack increment size- 400 octal 256 decimal
Initial managed memory size- 16310666 octal 3772854 decimal
Managed memory increment size- 1000 octal 512 decimal
Available managed memory- 7761 octal 4081 decimal
Base address of managed memory/stack- 12267633 octal
Base address of pad area- 12267373 octal

The key piece of information here is the “Initial task stack size.” In this example, segldr is telling us that each additional processor will require 395,304 words of storage. Since the program length is roughly 6.5 Mw (see “Program length” above) when running on 4 CPUs, running on 8 CPUs will require 4 x 395,304 additional words of core memory, or a total of roughly 8.1 Mw. Using this technique, the user can predict how much
memory his job will use and what is an appropriate value of $NCPUS without having to run the Model.

To find out after model execution how much memory a job actually used, run the `ja -h` command after model execution. Output will be of the form:

<table>
<thead>
<tr>
<th>Command Name</th>
<th>Started At</th>
<th>Memory HiWater</th>
</tr>
</thead>
<tbody>
<tr>
<td>ccm.xx.2</td>
<td>17:16:52</td>
<td>13052</td>
</tr>
</tbody>
</table>

The 13052 number says that the Model used 13052 blocks x 512 words/block = 6,682,624 words of memory at its high-water point, i.e. the most memory that was required at any one time. In practice the number should be quite close to what the user can predict from `seg1dr` statistics using the above technique.
G. Changing the Model

The most common changes to the Model are adding new variables, adding transported constituents, changing the content of the history tape, changing resolution, and adding a parameterization. This section provides some guidelines for making these kinds of changes in CCM2.

1. Tools for Modifying the Code

The section, “Source Code Maintenance” on page 9 explains how to change CCM2 code and incorporate the changes into the standard Model. Additional tools for validating these changes are discussed in “Trouble-Shooting Model Changes” on page 119.

2. Adding New Variables

The procedure for adding variables to CCM2 is dependent on several issues. Maybe the variable is already in the Model but not on the history tape. Or you need an entirely new variable, perhaps a new constituent for transport. Several commonly requested modifications will be discussed here.

Case 1: The variable is in the Model, but the user wants it on a history tape. If the field is in the Master Field List (see Table D.1 in “Appendix D: Master Field List”), the user may modify the Model code to uncomment the OUTFLD call for that field (remove the “C” from column 1) and include the field in the specification of the namelist parameters PRIMARY or AUXF. PRIMARY will place the field on the primary history tape. AUXF will define an auxiliary tape, as described in Table I.1 on page 14.

If the field is not in the master list, the user must first add it to the list by modifying BLDFLD, as shown below. Next, space for the field must be allocated in the history tape buffer by increasing parameter pfflds in parameter file pagrid.com.

The user must then add an OUTFLD call for the field at the proper location in the code. This means that OUTFLD must be called below LINDRV in the calling tree. Once these steps are taken, the field may be added to the history tape using namelist parameters PRIMARY or AUXF, described above.

The master field list consists of two arrays, fieldn(2,pfflds) and iflds(3,pfflds), residing in common/comhst/.fieldn contains the following character information:

| word 1 | 8-character field name, left-justified, alphanumeric or spaces only |
| word 2 | 8-character units description |

Array iflds contains the following integer information:
word 1  Level indicator,
        0=single-level,
        1=multi-level at interfaces,
        2=multi-level at levels

word 2  TRATOI translation of field name. This translator, a statement function in file tratoi. com, builds a unique integer from the field name, allowing the Model to use an integer compare in searching the field list, which is much more efficient than a character compare. This translation imposes the limitations noted above for defining field names and will translate all input to upper case.

word 3  Active/inactive flag. A value of 1=active indicates that this field will appear on the default primary history tape. This flag can be overridden by input parameters PRIMARY and EXCLUDE.

A complete master field list appears in "Appendix D: Master Field List."

Case 2: The variable is new to the Model and diagnostic in nature. The field should be in a user-defined local array, or in the Model buffer if the information needs to be carried across time levels. To add a field to the buffer requires adding buffer field pointers to common /comgrd/ and setting those pointers in subroutine POINTS. Parameters in pagrid.com which determine buffer lengths must also be changed. These parameters are documented in Figure II.2 on page 81. Follow directions under Case 1 to include the new variable on the history tape.

Case 3: The variable is new to the Model and is to be transported by the SLT scheme. The next section describes the steps required to make this change.

3. Adding Constituents to the Model:
   The Semi-Lagrangian Transport

The parameter pcnst represents the number of constituents carried in the Model. CCM2 carries one constituent (water vapor) by default. To add one or more constituents, in addition to changing pcnst to a value greater than 1, the user must either create an initial dataset containing values for all new constituents or introduce code into subroutine INIDAT to set constituent values before doing the initial write to the work files.

The Model will then:

* Create array space for the new constituents and associated diagnostics.

* Advect all constituents using SLT.
• Apply vertical diffusion and convective transport to the new constituents.

• Write constituent forecasts and associated diagnostics to the history tape.

**NOTE:** These new constituents will act strictly as passive tracers. There are no feedback mechanisms built into the Model. As additional physical processes on new constituents, the user must also do the following steps:

• Define surface fluxes and atmospheric source/sinks.

• Add other “adjustment” processes for constituents.

• Set global minima.

• Build in constituent feedbacks to the Model.


These user changes are described in detail below:

(a) **Initializing Fields**

The user may generate an initial dataset, using the CCM Modular Processor, which contains all of the desired constituent fields. Once this dataset is created, the Model is capable of automatically reading the new constituent initial conditions into the proper array space, provided the field names on the initial dataset and those in the Model (“tracnam”) are the same (defined in Table D.1 in “Appendix D: Master Field List”). Element tracnam(1) is reserved for water vapor, q.

Alternatively, the user may write new code and insert it into subroutine INIDAT, which sets the constituent values as desired. This code should replace the call to MKSLIC which reads constituent fields from the initial dataset, commented as “C Initialize non-h2o tracers.”

(b) **Surface Fluxes, Sources, and Sinks**

Code to generate surface fluxes and atmospheric sources and sinks should be added to subroutine TPHYS immediately after the call to SRFINT. Surface fluxes (Kg/m²/sec) are to be computed and stored in the array cflx (local to TPHYS) beginning at the array position (1,2). (The first column of the cflx array is reserved for the surface flux of water vapor.) Sources and sinks are also to be computed as tendencies (Kg/Kg/sec) and stored in the array srcsnk beginning at the array position (1,1,2). The user should not add these fluxes, sources, and sinks to the constituent fields themselves; the Model will do so. The user must, however, set up code to write these tendencies to the history tape for post-processing if so desired.
(c) Adjustment Processes

Code to perform adjustment physics (other than convective transport) to the time level \( n \) constituents should be inserted within APHYS. Unlike the source/sink and flux calculations described above, there is no code that does the adjustment. Users must supply this code.

(d) Global Minima

During the evolution of a Model run, constituent values may fall below an “expected” minimum (e.g., /comqmin/ variable \( q_{\text{min}} \) for water vapor).\(^1\) Currently, the Model checks for this possibility for all constituents at all points in the atmosphere. All points whose values fall below a given minimum for each constituent are reset to that minimum.

NOTE: When this procedure is invoked, global mass is not conserved. Also, the default minima may not be appropriate for constituents specified by the user. These values reside in the array \( q_{\text{min}} \) and can be reset by the user in the routine INITCOM. The array element \( q_{\text{min}}(1) \) is reserved for water vapor.

4. Changing Resolution

Any change in resolution from the standard 18-level T42 configuration requires changing all initial and boundary datasets. Changes in horizontal resolution may also require changing physical parameterizations and/or changes in namelist parameters, such as timestep, diffusion coefficients, etc. Standard initial and boundary datasets reside on permanent Cray disk for the T42 Model (see “Model Run Script” on page 1, for these file names).

5. Adding Parameterizations

Before attempting to add your own parameterization to CCM2, it would be helpful to read “Other Common Blocks and Parameter Decks: The Parameterization Interface” on page 87, concerning the design of the parameterization interface. Adherence to the rules outlined in that section means that a parameterization need have no knowledge of overall Model data structures—all that is needed are longitude and level dimensions of input arguments on the Model grid. Latitude, time, and extended grid addressing are all handled by the Model control code.

Implementing parameterizations in this way requires that some restrictions be placed on the coding of the parameterizations. A parameterization should need no knowledge of the order in which Model fields are stored in memory. This implies that if a parameterization requires two time levels of a particular field, say temperature, these two levels are passed as two separate arguments.

\(^1\) CCM2 uses 0. as the minimum for water vapor in subroutine VDIFF.
rate arrays by the interface routine. Parameterizations should provide separate output arguments to return “answers” to the main Model; that is, they should not modify input arguments which point into the main Model buffer or other Model data structures. Time tendencies may be calculated from the input and output arguments at the level of the parameterization interface routine. Computed (or recomputed) fields should also be stored in the Model buffer by the interface routine.

The user also needs to be aware of restart requirements. If there are variables in your parameterization that need to be present at restart, they must be initialized at restart time. For new variables this means that they must appear on the regeneration dataset. If you store your new field in the Model buffer, it will automatically be written to the regeneration dataset. Otherwise, you must add code to subroutine WRTRS1 to write your data and subroutine SPLITF to read it for a continuation run.

6. Trouble-Shooting Model Changes

When a model run goes awry, where does one turn? First, if there is any doubt as to the reason for abnormal termination, ensure that the Model is run single-threaded. This is because abnormal termination in a multitasked job can result in confusing ancillary error messages that are generated as a result of the initial error with which the user is concerned.

If the Model terminates with an internally generated error message, the first place to look is “Model Error Messages” on page 61 for guidance as to the nature of the abnormal end. However, this error message may be insufficient to determine what must be done to fix the simulation (e.g., “OK, fine, the dry adjustment procedure failed to converge, but why?”). Sometimes there is no internally generated error message, as for example when a floating point error occurs. The purpose of this section is to suggest courses of action to determine the nature of the problem when the model terminates abnormally.

Resource allocation errors are addressed first. Next, we discuss remedies if a coding error is suspected. Finally, analysis tools are described for physics formulation errors (i.e., where there is an error in the way the user is changing certain prognostic variable calculations).

(a) Resource Allocation Errors

If insufficient storage is allocated for scratch files located on the SSD, the Model will terminate and a somewhat informative error message will be printed if the file assign was done with novfl specified. novfl instructs the operating system to abort if file I/O to the SSD exceeds the request as specified on the #QSUB -1Q line in the job script. To determine how much SSD storage is required, refer to the line of model printout
which says:

**Total size of files normally assigned to SDS:**

The user’s `#QSUB -LQ` request must at least match this number if `novfl` was specified. If `novfl` is not specified, scratch file I/O requests which exceed the `#QSUB -LQ` specification will spill to disk. Spilling to disk can outrageously degrade turnaround time for a long simulation.

An insufficient memory request (`#QSUB -LM`) may result in a reasonable error message of the form:

**Not enough space**

or it may not. Check the line of `segldr` output which reads:

**Program length= 22363070 octal 4843064 decimal**

You must ensure that the number of megawords requested on the `#QSUB -LM` line is larger than the program length in decimal. If the initial program length is less than the `#QSUB` request, but only by a few hundred thousand words, a memory allocation problem may still exist. The reason is that during a run the operating system often needs to allocate additional memory for its own purposes. If sufficient memory is not available, this may not be evident from the error message received. The bottom line is: if the Model is using close to the amount of memory declared on the `#QSUB -LM` line and it appears to be dying due to some system error (e.g.: `SAVDIS:Error in ishell call,ier= 256`), try increasing the memory request by approximately 1 million words and rerun the job.

### (b) Coding Errors

One of the rules imposed by the CCM2 coding standard (see “CCM2 Coding Standard” on page 122) is the use of the Fortran *implicit none* statement in all Model program modules. This is done expressly to catch errors in variable usage, and is recommended for all user-written code for the same reason.

The Cray utility `flint` is most useful for determining if there is an argument list mismatch between routines, when a variable is referenced before it is set, or for other serious coding errors such as *common* block length mismatches between routines. It also gives helpful hints about unused variables, variables that are set but never referenced, and the like.
flint looks at source code, so the user can check his code for errors before even running it through the compiler. We recommend gathering all the source in one file, then invoking flint with the -g and -X132 command line arguments. This utility can provide other information, such as a calling tree, or a symbol table with cross-reference list. The user is referred to the UNICOS man page for further details.

Given reasonable output from flint and continued suspicion of a coding error, there are some compiler options that may help if run-time debugging becomes necessary. We recommend that the Model always be run single-threaded during run-time debugging. It is important to determine early on whether a user change to Model code affects only multitasked execution. Also, output generated as a result of these options is much easier to interpret if generated during single-threaded execution.

The bounds checker

The -Rb command line argument to cft77 turns on array bounds checking. During model execution of code compiled with this option, a message will be printed whenever an array is referenced outside its declared dimensions. This option disables all vectorization, resulting in an enormous increase in CPU time (up to a factor of 100!) used for a given simulation. The -Rb option is therefore only practical if the Model is blowing up very early in the run, or if the user wants to turn it on in a very limited section of the code. In standard CCM2 code, arrays are never referenced outside of their declared bounds.

The symbolic debugger

The -ez option to the compiler can also be helpful for run-time debugging of CCM2. This option causes debug symbol table information to be written to the object file produced by the compiler, which is then used by utilities to analyze the post-mortem core file after the Model crashes. Its usage results in a relatively small performance penalty (less than 10%) during Model execution, so it can feasibly be used when the Model doesn’t die until well into the simulation.

There are many utilities that use this symbol table information, (prof, cdbx, etc.), but the only one which will be discussed here is debug. This utility is most useful for reporting values of common variables and local variables at the time of Model termination. The -B option to debug requests that all common variables be printed. The -cl option says to print local variables only one level deep into the branch of the calling tree which was active at the time of Model termination. The -d option can be used to determine how many values from each array dimension are actually printed when debug examines the user’s “core” file. Other options to debug are available, and again the user is referred to the man page for further details.

Presetting to "indefinite"

Statically allocated memory locations should always be initialized to “indefinite” during a Model run (the -f indef option to segldr). It costs virtually nothing to do this and will cause an error to be reported if
uninitialized static memory is used in any floating-point computation. A more costly but equally useful option for debugging is the -ei compiler option. This option initializes stack memory to indefinite and will likewise result in an error if memory so initialized is used in a floating-point computation. It is much more expensive than static initialization because memory values must be set each time stack memory is made available to a routine. In CCM2, this means a very large amount of memory must be initialized every timestep. Runs using this option will incur a performance penalty of approximately 10%.

(c) Formulation Errors

Adding a new physics package to the Model or changing the settings for various namelist parameters can sometimes result in a Model abort. Hopefully, an informative message was printed to provide a starting point for debugging (see “Model Error Messages” on page 61). Where the Model died is not always the same place where things first started going wrong, however.

Some of the tools mentioned in the previous section (“Coding Errors” on page 120) can also be helpful if a formulation error is suspected. In particular, use of the -ez compiler option followed by the debug command is a very useful diagnostic tool if the Model is actually crashing.

If the Model is running but producing incorrect or suspicious history files, a quick and easy-to-use diagnostic program, called cprtps, is available from the Core Group. This program provides a statistical analysis of differences in history file data. It requires two input history tapes with data valid for identical Model times. No user input is required.

CPRTPS compares fields of the same name on each tape, printing out statistics about the number of differences found, location and magnitude of worst absolute difference, location and magnitude of worst relative difference, rms difference, maximum and minimum field values, and average field values.

7. CCM2 Coding Standard

This section describes the coding standard for the NCAR Community Climate Model Version 2 (CCM2). The released CCM2 code adheres as much as possible to this standard. Any code that subsequently becomes a part of the Model should also follow the standard. The CCM Core Group reserves the right to change any such code to conform to this standard.

One reason for imposing a standard on CCM2 coding is to enhance the usefulness of certain system debugging tools. For instance, CCM2 may be run through the Fortran Lint (flint) global cross-reference and code checking tool on the Cray and produce intelligible output of a reasonable length. That is, there are few instances of nonstandard Fortran usage in the Model.
and, therefore, if you use this tool to debug your own Model changes, the output should be relevant to your code.

Another debugging tool to be used on the CCM is the Cray bounds checker. There are no constructs in the standard CCM2 code that generate large numbers of “warnings” or errors in using these tools. We strongly recommend that users adhere to this rule in their own code.

(a) Code Appearance

All Fortran statements except comments and quoted strings will be coded using lower case letters. Comments should be in mixed case to help set them apart from the code. All character literals and strings, including those in format statements, will be delimited by the single quote (apostrophe) character.

Executable code will be commented to explain the function of the statements following the comments. These comments (one or more lines) will be set off by blank comment lines (“C” in column 1 only). On those comment lines containing text, the text will start in column 3. It is suggested that comments denoting logical groupings of code, declarations, etc., will be offset with comment lines consisting of a “C” in column 1 and “-” characters in columns 2 through 72.

In cases where brief comments on the actual line of code clarifies the code, the “!” construct may be used, where there must be white space between the end of the statement and the “!.” There will be no completely blank lines in the code.

Special forms of comments for declaration statements are described below.

For each program module, following the subroutine or function statement, a block of comments delineated by “C----...” comment lines will briefly describe the function of the module. Following this, a block of comments labelled “Code history” indicates the originator of the current version of the module and the Core Group member responsible for standardizing the code, along with relevant dates. This section may include any notes pertinent to revisions, usage, etc.

Each routine will contain a Fortran implicit none statement, i.e., all variables must be explicitly typed and all called routines other than Fortran intrinsics declared external. Array dimensions will be specified via type statements, so there will be no dimension statements.

The main Model parameter statements and common statements will be stored as separate files to be “included” wherever appropriate in the code (see “Model Run Script” on page 1). Include files will have one blank comment line at the top and one at the bottom. Between “include” directives subroutines will contain “C----...” comment lines to provide a visual aid to
Delineating declarative statements

separating common blocks, etc. An "included" file must not itself contain an "include." The common variables and parameters should be vertically registered in fields of 8 characters, up to 5 to a line. Each block should start with a comment describing the functionality of the statement.

Subroutine arguments should be typed and described in a section titled "Arguments." Each argument will be typed on a separate line, with a comment following a "!" character to briefly describe its function. Within this list, input arguments will appear first, following three comment lines, as shown below. Input/output arguments will be treated similarly, followed by output arguments:

```
C---------------------------------- Arguments ----------------------------------
C
C Input arguments
C
integer nnn ! First input argument

C
C Input/output arguments
C
real fff ! Another argument

C
C Output arguments
C
character(*) ccc ! Output argument

C
C Passed through
C
character(*) sss ! This argument is passed through to routine x
```

See "Trouble-Shooting Model Changes" on page 119 for details on how argument lists should appear in subroutine and call statements.

Any parameter statements that are local to a particular program module will not be in an "include" file, but will appear in a section titled "Local Parameters".

The use of equivalence statements is discouraged, but if an equivalence should become necessary, it should be carefully commented.

data statements are meant to assign a starting value to local variables only—those setting common variables should appear in the block data module.
Following the "Code History" section, declarative statements should appear in the order *implicit none*, main Model *parameter* statements, *common* statements, type declarations of arguments, local parameters preceded by their type statements, type declarations of local variables, *equivalence* statements, *data* statements, *save* statements, *external* statements and statement functions.

The continuation character used throughout the code is the dollar sign ($).

**(b) Other Fortran Considerations**

Printed output may be generated using *format* statements or as list-directed output (*write (6, *)*). For efficiency reasons, list-directed output should not be used for printout which is repeated throughout the run. It may be used for diagnostic printout preceding an abnormal termination.

Certain diagnostic printout may be included which is not intended to be a part of standard Model output. This printout should be under the control of a local logical variable rather than being commented out.

*format* statements containing scaling factors (e.g., 1pe12.3) should have a "," between the scaling character and the formatting character, as in "1p,e12.3."

Obscure "magic numbers" should not appear totally undocumented in calculations. These values should be assigned to local variables and commented as to their use.

**(c) Program Module Names and Variable Names**

Up to eight characters will be allowed for all Fortran names.

There are no strict conventions for naming modules in the control portion of the Model. Where the basic function of a module has not changed from that in CCM1, the name may remain the same, since users are familiar with the existing names. Modules within parameterizations, however, should all begin with the same two- or three-character designator, as in the CCM2 radiation parameterization ("RAD"). The remaining 5 or 6 characters are used as a mnemonic to indicate the general function of the module. Model *common* blocks will be named beginning with "com."

Fortran *parameter*’s in the main Model *parameter* statements will begin with the letter "p." This convention is also suggested for so-called "Local Parameters."

No specific naming conventions will be imposed for local variables.
Model code should use “structured programming” constructs where possible, avoiding the use of the `go to` statement and excessive use of statement labels. The `do end do` construct may be used for short (up to 30 lines of code) `do` loops where it is easy to see the extent of the loop without paging through the code. `do` loops and `if-then-else` blocks should be consecutively indented three spaces at a time. Where part of a calculation appears on a continuation statement, the code should be indented past the equal sign if possible. A continuation line should never begin in a column preceding where the initial line of the statement begins. A single blank should appear on either side of the equal sign, and on either side of all `+` and `-` signs not enclosed in parentheses.

Statement numbers should monotonically increase within a program module, by an increment of at least 10. Labels of 9000 and above are reserved for `format` statements. Statement labels should be right-justified to column 5. `format` statements should appear at the end of the program module, following the return statement.

The argument lists in Fortran `subroutine` and `call` statements will be formatted as follows:

Multiple arguments in a list should appear in the order input, input/output and output arguments. Arguments should be vertically registered, left-justified in an eight-character field. Argument lists for a particular subroutine and all its calls should line up and register correspondingly, if possible. If not, each additional line required should be indented consecutively, as shown below, to find a corresponding argument without counting all arguments up to that point.

```
subroutine x(a, b, c, d, e, f, g)
    call x(pdela(index1, index2), pdelb(index1, index2),
           carray(index3, index4), dummy(index5, index6),
           ework(index7),
           f(i1, i2), g(i1, i2))
```

Arrays passed through the argument list should be dimensioned according to the actual number of elements referenced within the calling routine, i.e., do not dimension dummy arrays by 1. If the dimension is not known (say, in a general-purpose utility routine) it may be specified by a `{*}`. This allows...
Portability issues

the code to run using the system bounds checker without generating many bogus errors.

Where possible, Cray “vector merge” functions should be replaced by standard Fortran IF constructs. These functions may be called only when required for vectorization.
Appendix A: Glossary

case — A term used to denote a CCM2 experiment, including one initial run and as many continuation runs as required to conclude the experiment. A Model case is cataloged using the $CASE environment variable, which appears in word 1 of the character (second) history tape header record.

CCM1 — The previous version of the NCAR Community Climate Model, the basic code from which CCM2 was built.

CCM2 — The current version of the NCAR Community Climate Model, as described in the introduction to this User’s Guide.

CCM Modular Processor — A program available from the CCM Core Group which post-processes CCM history tapes on the NCAR computing system.

dispose — Transfer of output datasets to the NCAR Mass Store System.

Fortran-callable — Describes a UNICOS shell command for which there exists a Fortran interface, i.e., a system function or subroutine call-able from Fortran.

frozen — Describes the Community Climate Model in its standard form, as documented in this User’s Guide and presented to the community.

gpp — “Generic Preprocessor,” a UNICOS system tool to preprocess Fortran code prior to compiling. This preprocessor is used by the CCM2 run script to gather source code for subsequent compilation.

heap — A repository of main memory dynamically managed by Fortran, from which the Fortran stack is allocated.

history tape — A binary dataset, the primary output medium for field values generated by CCM2 in the course of a time integration.

include file — In the case of CCM2, a file, called xxx.com, containing code, such as a common statement, which may need to be included in more than one location in the source file.

MSS — The NCAR Mass Store System, consisting of the IBM 3090 Mass Storage Control Processor (MSCP), a large IBM disk farm for intermediate storage, an IBM 3480 Cartridge Tape System for
archival storage and a Storage Tek Automatic Cartridge System (robot).

**multitasked** — Refers to a program configured to execute on more than one central processor simultaneously.

**namelist** — A Fortran extension that processes input parameters in a free-form fashion.

**out-of-core** — Refers to the use of secondary storage for cycling to and from main memory during a Model run.

**packing** — A process optionally applied to output history tape data, which compresses the values according to a specified density, either 2, 3 or 4 to 1, via system routine PACKAF.

**plug-compatible** — Refers to a parameterization coding standard to promote ease of replacement and/or exchange of parameterizations in the Model.

**pointer** — An integer variable used as an offset in indexing into the main Model buffer. Not to be confused with a Fortran pointer statement.

**preprocessor** — A system program which takes input directives and source code and builds a file ready for a language compiler.

**single-threaded** — Refers to code that executes on only one processor at a time.

**SDS** — Secondary Data Segment, a file management method available via UNICOS software for use on the SSD.

**SSD** — Solid-State Storage Device, a very high-performance secondary storage device on the Cray Y-MP.

**stack** — Main memory dynamically managed by Fortran, used for storing local variables in program modules.

**timestep** — Refers to a single time integration of the Model from one value of nstep to the next.

**word** — The Cray 64-bit word.
Appendix B: CCM2 Printed Output

This appendix contains the complete printed output from a four-timestep CCM2 run. Selected portions are explained in “Printout from a CCM2 Run” on page 57.
NCAR Community Climate Model, Version 2.0
Copyright (C) 1992
University Corporation for Atmospheric Research
All Rights Reserved

DATE 08/13/92 TIME 14:08:17

Opened save file A as direct access unit, size: 950272 words.
Opened save file B as direct access unit, size: 2752512 words.

E$CCMEXP
CTITLE = 'ccm2t42',
NCDATA = '/CCM2/T42/%data%/SEPl',
BNDTI = '/CCM2/T42/%data%/tibds',
BNDTVS = '/CCM2/T42/%data%/tvbs',
BNDTVO = '/CCM2/T42/%data%/ozn',
IRT=35,
NSVSN = 'rstrt',
NSREST=0,
NSWRPS='passwd',
NDENS = 1,
NNBDAT = 000901,
NNBSEC = 0,
NNDDBAS = 0,
NNSBAS = 0,
MFILT = 5,
DTIME = 1200.,
NESTEP = 4,
NHTFRQ = 4,
IRAD = 4,
IRADAED = 4,
SSTCYC = .T.,
OZNCYCYC = .T.,
DIF4 = 1.E16,
HYDRO = .F.,
$

*** INPUT PARAMETERS (CCMEXP) ***

Initial run
********** THIS IS CASE test**********
ccm2t42
User name to build pathnames = jquser
Initial dataset is: /CCM2/T42/%data%/SEP1
Time-invariant boundary dataset is: /CCM2/T42/%data%/tibds
Time-variant boundary dataset (sst) is: /CCM2/T42/%data%/tvbds
Time-variant boundary dataset (ozone) is: /CCM2/T42/%data%/ozn
Restart dataset is: rstrt
Write password for output files (NSWRPS) is passwd
Restart flag (NSREST) 0=no,1=yes,2=regen 0
Retention time for History Volumes = 35 Days
Virtual MS volume for History Tape (NSMVN) CTPUBLIC
Virtual MS volume for Regeneration data (NRMVN) CTPUBLIC
History tape 1 will not be packed
History Tape 1 write frequency (NHTFRQ) 4
Number of files per tape (MFILT) 5
Fields on history tape 1 will be averaged.
Regeneration data will be written for every history tape
Base day,seconds of day = 0 0
Base date,seconds of date = 901 0
Time step to end run (NESTEP) 4
Time step in seconds (DTIME) 1200.
Time filter coefficient (EPS) 0.060
DEL2 Horizontal diffusion coefficient (DIF2) 0.250E+06
DEL4 Horizontal diffusion coefficient (DIF4) 0.100E+17
Number of levels Courant limiter applied 1
Lowest level for dry adiabatic adjust (NLVDRY) 3
Frequency of Radiation (IRAD) 4
Frequency of Absorptivity/Emissivity (IRADAE) 4
Frequency of SST Initialization (ITSST) 1

Transport will be by SEMI-LAGRANGIAN Method
SST boundary dataset will be reused for each model year
OZONE boundary dataset will be reused for each model year
Output files will be disposed ASYNCHRONOUSLY
Running in PRODUCTION mode -- output files will not be linked to /usr/tmp
------------------------------------------------------------------------
ATCHBND: Sending the following to assign: assign -a /ccm/ccm2/T42/SEP1 -b 96 fort.4
INITIAL:HEADER READ AND CHECKED CORRECTLY
1 Layer Locations (*1000)
  1  2.9170  0.0000  2.9170
  2  7.9292  0.0000  7.9292
  3 21.5539  0.0000 21.5539
       32.5591  0.0000 32.5591
<table>
<thead>
<tr>
<th>C</th>
<th>49.1834</th>
<th>63.9471</th>
<th>0.0000</th>
<th>0.0000</th>
<th>49.1834</th>
<th>63.9471</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>83.1425</td>
<td>81.6768</td>
<td>17.3664</td>
<td>117.9849</td>
<td>99.0432</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>79.9308</td>
<td>38.0541</td>
<td>60.6928</td>
<td>138.7129</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>75.7738</td>
<td>87.3088</td>
<td>115.8237</td>
<td>189.1908</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>70.5752</td>
<td>148.9037</td>
<td>219.4789</td>
<td>251.2394</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>64.2963</td>
<td>223.2996</td>
<td>287.5959</td>
<td>324.8475</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>56.9838</td>
<td>309.9406</td>
<td>366.9244</td>
<td>408.9554</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>48.7913</td>
<td>407.0096</td>
<td>455.8009</td>
<td>501.2755</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>39.9895</td>
<td>511.2977</td>
<td>551.2872</td>
<td>598.2482</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>30.9631</td>
<td>618.2465</td>
<td>649.2096</td>
<td>695.1694</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>22.1902</td>
<td>722.1927</td>
<td>744.3829</td>
<td>786.5099</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14.2039</td>
<td>816.8173</td>
<td>831.0212</td>
<td>866.4074</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>7.5413</td>
<td>895.7590</td>
<td>903.3003</td>
<td>929.2755</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>2.6838</td>
<td>953.3137</td>
<td>955.9975</td>
<td>970.4457</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>0.0000</td>
<td>985.1122</td>
<td>985.1122</td>
<td>992.5282</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>0.0000</td>
<td>1000.0000</td>
<td>1000.0000</td>
<td>1000.0000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reference pressures (Pa):

1. 291.7000
2. 792.9200
3. 2155.3900
4. 4918.3400
5. 8314.2500
6. 11798.4900

Pressures (Pa):

1. 291.7000
2. 792.9200
3. 2155.3900
4. 4918.3400
5. 8314.2500
6. 11798.4900
7 16308.2600 18919.0800 5639.6300
8 21947.8900 25123.9400 6811.7000
9 28759.5900 32484.7500 7932.8500
10 36692.4400 40895.5400 8887.6500
11 45580.0900 50127.5500 9548.6300
12 55128.7200 59824.8200 9792.2400
13 64920.9600 69516.9400 9517.3300
14 74438.2900 78650.9900 8663.8300
15 83102.1200 86640.7400 7227.9100
16 90330.0300 92927.5500 5269.7200
17 95599.7500 97044.5700 2911.4700
18 98511.2200 99252.8200 1488.7800
19 100000.0000

Truncation Parameters
NTRM = 42
NTRN = 42
NTRK = 42

REFERENCE TEMPERATURES FOR SEMI-IMPLICIT SCHEME =
300.000 300.000 300.000 300.000 300.000 300.000 300.000 300.000 300.000 300.000
GRAVITY WAVE PHASE SPEEDS (M/S) FOR MEAN STATE =
341.904 200.945 123.358 84.715 60.844 46.374 35.929 28.283 22.039 17.019 13.100 10.028
GRAVITY WAVE EQUIVALENT DEPTHS (M) FOR MEAN STATE =
INIDAT: DRY MASS OF INITIAL DATA BEFORE CORRECTION = 9.8494400257E+04
MASS WILL BE HELD = 9.8222000000E+04
MASS OF MOISTURE AFTER REMOVAL OF NEGATIVES = 2.7140282197E+02
ATCHBND: Sending the following to assign:assign -a /ccm/ccm2/T42/tibds -b 96 fort.1
ATCHBND: Sending the following to assign:
assign -a /ccm/ccm2/T42/tvbds -b 96 fort.3

SSTINI: Read sst data for date (yymmdd) 840116
SSTINI: Read sst data for date (yymmdd) 840214
SSTINI: Read sst data for date (yymmdd) 840316
SSTINI: Read sst data for date (yymmdd) 840415
SSTINI: Read sst data for date (yymmdd) 840516
SSTINI: Read sst data for date (yymmdd) 840615
SSTINI: Read sst data for date (yymmdd) 840716
SSTINI: Read sst data for date (yymmdd) 840816
SSTINI: Read sst data for date (yymmdd) 840915

ATCHBND: Sending the following to assign:
assign -a /ccm/ccm2/T42/ozn -b 96 fort.2

OZNINI: Read ozone data for date (yymmdd) 840116
OZNINI: Read ozone data for date (yymmdd) 840214
OZNINI: Read ozone data for date (yymmdd) 840316
OZNINI: Read ozone data for date (yymmdd) 840415
OZNINI: Read ozone data for date (yymmdd) 840516
OZNINI: Read ozone data for date (yymmdd) 840615
OZNINI: Read ozone data for date (yymmdd) 840716
OZNINI: Read ozone data for date (yymmdd) 840816
OZNINI: Read ozone data for date (yymmdd) 840915

RAYLIEGH FRICTION

BOTTOM LEVEL: NBOTRL= 0 NTOPRL= 1

COEFFICIENTS

0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00
0.0000E+00

0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00

***** SATURATION VAPOR PRESSURE TABLE COMPLETED *****

PREALC: Sent following string to ishell:

setf -c -n 36088b:36088b h0001 2> /dev/null

PREALC: 36088 blocks of space for h0001 allocated in contiguous chunks of at least 36088 blocks

INTHT: Sending following string to assign:

assign -a h0001 -b 114 fort.20

Opened save file sunit as direct access unit 22 size: 3702784 words

Total size of files normally assigned to SDS: 11206656 words 12 Megawords.

**** Summary of Logical Unit assignments ****

Initial dataset unit (ninit) = 4
Time-inv boundary dataset (nbndti) = 1
Ozone dataset unit (nozone) = 2
SST dataset unit (nsst) = 3
NSTEP RMSZ RMSD RMST STPS STQ COU RANT
NSTEP = 0 8.833474687402251E-05 7.333832180666193E-06 252.716 9.84934E+04 2.76727591143027E+01 0.81 0.24

*** HEADER FOR CCM2 HISTORY TAPE ***

*** Primary History Tape ***

CASE: f20i
TITLE: plx20

LENHDI MFTYP MFILH MFILTH NRBD MAXSIZ NDAVU MXXX NLON NLOFW
217 43 1 5 3 57730 57730 0 128 128

PLAT PLEV PTRM PTRN PTRK NFLDH NSTEPH NSTPRH NITSLF NDBASE
64 18 42 42 42 60 0 0 0 0

NSBASE NNDCUR NSCUR NBDATE NBSEC NCDATE NCSEC MDT MHISF MFSTRT
0 0 0 000901 0 000901 0 1200 4 0

LENHDC LENHDR MPSIG MPLAT MPWTS MPFLDS MPHFLD
209 239 1 112 176 38 90

MSS PATH NAME
CURRENT /JQUSER/ccm2/test hist/h0001
FIRST /JQUSER/ccm2/test hist/h0001
INITIAL /CSM/ccm2/367 hist/h0105
TI BOUNDARY /CCM2/T42 %data% /tibds
SST BOUNDARY /CCM2/T42 %data% /tvbds
OZONE BOUNDARY /CCM2/T42 %data% /ozn

DATE TIME SEQ NO.
10/02/92 12:25:26 CI9290
10/02/92 12:25:16 CI9290
10/27/91 14:26:43 DUMSEQ
12/27/91 09:12:22 - 1 -
12/23/91 15:01:43 - 1 -
09/21/92 13:34:26
<table>
<thead>
<tr>
<th>FLD NO.</th>
<th>NAME</th>
<th>FLG.</th>
<th>FLD PT.</th>
<th>PACK.</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PHIS</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>M2/S2</td>
</tr>
<tr>
<td>2</td>
<td>PS</td>
<td>10</td>
<td>131</td>
<td>1</td>
<td>PA</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>12</td>
<td>259</td>
<td>1</td>
<td>K</td>
</tr>
<tr>
<td>4</td>
<td>U</td>
<td>12</td>
<td>2563</td>
<td>1</td>
<td>M/S</td>
</tr>
<tr>
<td>5</td>
<td>V</td>
<td>12</td>
<td>4867</td>
<td>1</td>
<td>M/S</td>
</tr>
<tr>
<td>6</td>
<td>Q</td>
<td>12</td>
<td>7171</td>
<td>1</td>
<td>KG/KG</td>
</tr>
<tr>
<td>7</td>
<td>TA01</td>
<td>12</td>
<td>9475</td>
<td>1</td>
<td>KG/KGS</td>
</tr>
<tr>
<td>8</td>
<td>VD01</td>
<td>12</td>
<td>11779</td>
<td>1</td>
<td>KG/KGS</td>
</tr>
<tr>
<td>9</td>
<td>DC01</td>
<td>12</td>
<td>14083</td>
<td>1</td>
<td>KG/KGS</td>
</tr>
<tr>
<td>10</td>
<td>DTH</td>
<td>12</td>
<td>16387</td>
<td>1</td>
<td>K/S</td>
</tr>
<tr>
<td>11</td>
<td>ORO</td>
<td>0</td>
<td>18691</td>
<td>1</td>
<td>FLAG</td>
</tr>
<tr>
<td>12</td>
<td>WET</td>
<td>10</td>
<td>18819</td>
<td>1</td>
<td>M</td>
</tr>
<tr>
<td>13</td>
<td>SNOWH</td>
<td>10</td>
<td>18947</td>
<td>1</td>
<td>M</td>
</tr>
<tr>
<td>14</td>
<td>PRECL</td>
<td>10</td>
<td>19075</td>
<td>1</td>
<td>M/S</td>
</tr>
<tr>
<td>15</td>
<td>PRECC</td>
<td>10</td>
<td>19203</td>
<td>1</td>
<td>M/S</td>
</tr>
<tr>
<td>16</td>
<td>SHFLX</td>
<td>10</td>
<td>19331</td>
<td>1</td>
<td>W/M2</td>
</tr>
<tr>
<td>17</td>
<td>LHFLX</td>
<td>10</td>
<td>19459</td>
<td>1</td>
<td>W/M2</td>
</tr>
<tr>
<td>18</td>
<td>QFLX</td>
<td>10</td>
<td>19587</td>
<td>1</td>
<td>KG/M2/S</td>
</tr>
<tr>
<td>19</td>
<td>PBLH</td>
<td>10</td>
<td>19715</td>
<td>1</td>
<td>M</td>
</tr>
<tr>
<td>20</td>
<td>USTAR</td>
<td>10</td>
<td>19843</td>
<td>1</td>
<td>M/S</td>
</tr>
<tr>
<td>21</td>
<td>TPERT</td>
<td>10</td>
<td>19971</td>
<td>1</td>
<td>K</td>
</tr>
<tr>
<td>22</td>
<td>QPERT</td>
<td>10</td>
<td>20099</td>
<td>1</td>
<td>KG/KG</td>
</tr>
<tr>
<td>23</td>
<td>DTV</td>
<td>12</td>
<td>20227</td>
<td>1</td>
<td>K/S</td>
</tr>
<tr>
<td>24</td>
<td>FSNS</td>
<td>10</td>
<td>22531</td>
<td>1</td>
<td>W/M2</td>
</tr>
<tr>
<td>25</td>
<td>FLNS</td>
<td>10</td>
<td>22659</td>
<td>1</td>
<td>W/M2</td>
</tr>
<tr>
<td>26</td>
<td>FLNT</td>
<td>10</td>
<td>22787</td>
<td>1</td>
<td>W/M2</td>
</tr>
<tr>
<td>27</td>
<td>FSNT</td>
<td>10</td>
<td>22915</td>
<td>1</td>
<td>W/M2</td>
</tr>
<tr>
<td>28</td>
<td>CLOUD</td>
<td>12</td>
<td>23043</td>
<td>1</td>
<td>FRACTION</td>
</tr>
<tr>
<td>29</td>
<td>EFFCLD</td>
<td>12</td>
<td>25347</td>
<td>1</td>
<td>FRACTION</td>
</tr>
<tr>
<td>30</td>
<td>FLNTC</td>
<td>10</td>
<td>27651</td>
<td>1</td>
<td>W/M2</td>
</tr>
<tr>
<td>31</td>
<td>FSNTC</td>
<td>10</td>
<td>27779</td>
<td>1</td>
<td>W/M2</td>
</tr>
<tr>
<td>32</td>
<td>FLNSC</td>
<td>10</td>
<td>27907</td>
<td>1</td>
<td>W/M2</td>
</tr>
<tr>
<td>33</td>
<td>FSNSC</td>
<td>10</td>
<td>28035</td>
<td>1</td>
<td>W/M2</td>
</tr>
<tr>
<td>34</td>
<td>OMEGA</td>
<td>12</td>
<td>28163</td>
<td>1</td>
<td>PA/S</td>
</tr>
<tr>
<td>35</td>
<td>DQP</td>
<td>12</td>
<td>30467</td>
<td>1</td>
<td>KG/KGS</td>
</tr>
<tr>
<td>36</td>
<td>TAUX</td>
<td>10</td>
<td>32771</td>
<td>1</td>
<td>N/M2</td>
</tr>
<tr>
<td>37</td>
<td>TAUY</td>
<td>10</td>
<td>32899</td>
<td>1</td>
<td>N/M2</td>
</tr>
<tr>
<td>38</td>
<td>SRFRAD</td>
<td>10</td>
<td>33027</td>
<td>1</td>
<td>W/M2</td>
</tr>
<tr>
<td>39</td>
<td>QRS</td>
<td>12</td>
<td>33155</td>
<td>1</td>
<td>K/S</td>
</tr>
<tr>
<td>40</td>
<td>QRL</td>
<td>12</td>
<td>35459</td>
<td>1</td>
<td>K/S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>CLDTOT</td>
<td>10</td>
<td>37763</td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>CLDLOW</td>
<td>10</td>
<td>37891</td>
<td></td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>CLDMED</td>
<td>10</td>
<td>38019</td>
<td></td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>CLDHGH</td>
<td>10</td>
<td>38147</td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>TS1</td>
<td>10</td>
<td>38275</td>
<td></td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>TS2</td>
<td>10</td>
<td>38403</td>
<td></td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>TS3</td>
<td>10</td>
<td>38531</td>
<td></td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>TS4</td>
<td>10</td>
<td>38659</td>
<td></td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>SOLIN</td>
<td>10</td>
<td>38787</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>UTGW</td>
<td>12</td>
<td>38915</td>
<td></td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>VTGW</td>
<td>12</td>
<td>41219</td>
<td></td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>TAUGWX</td>
<td>10</td>
<td>43523</td>
<td></td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>TAUGWY</td>
<td>10</td>
<td>43651</td>
<td></td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>DTCOND</td>
<td>12</td>
<td>43779</td>
<td></td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>CMFDT</td>
<td>12</td>
<td>46083</td>
<td></td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>CMFDQ</td>
<td>12</td>
<td>48387</td>
<td></td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>CMFMC</td>
<td>11</td>
<td>50691</td>
<td></td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>CMFSL</td>
<td>11</td>
<td>52995</td>
<td></td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>CMFLQ</td>
<td>11</td>
<td>55299</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>CNVCLD</td>
<td>10</td>
<td>57603</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SIGMA VALUES AT FULL LEVELS**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4.80330000000002E-03</td>
<td>1.30731000000000E-02</td>
<td>3.25591000000001E-02</td>
<td>6.39471000000000E-02</td>
</tr>
<tr>
<td>1.35712900000000E-01</td>
<td>1.89190800000000E-01</td>
<td>5.98248200000000E-01</td>
<td>9.70456999999999E-01</td>
</tr>
<tr>
<td>5.01759499999998E-01</td>
<td>5.98248200000000E-01</td>
<td>6.95169400000000E-01</td>
<td>9.92582000000000E-01</td>
</tr>
<tr>
<td>9.29275499999998E-01</td>
<td>9.70456999999999E-01</td>
<td>9.92582000000000E-01</td>
<td>1.00000000000000E+00</td>
</tr>
</tbody>
</table>

**SIGMA VALUES AT HALF LEVELS**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2.91700000000000E-03</td>
<td>7.92920000000000E-03</td>
<td>2.15539000000000E-02</td>
<td>4.91834000000000E-02</td>
</tr>
<tr>
<td>1.17984900000000E-01</td>
<td>1.63082600000000E-01</td>
<td>2.19478900000000E-01</td>
<td>2.87589999999999E-01</td>
</tr>
<tr>
<td>4.55800900000000E-01</td>
<td>5.98248200000000E-01</td>
<td>6.95169400000000E-01</td>
<td>9.92582000000000E-01</td>
</tr>
<tr>
<td>9.03300000000000E-01</td>
<td>9.70456999999999E-01</td>
<td>9.92582000000000E-01</td>
<td>1.00000000000000E+00</td>
</tr>
</tbody>
</table>

**LATITUDES**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-87.86380</td>
<td>-85.09653</td>
<td>-82.31291</td>
<td>-79.52561</td>
</tr>
<tr>
<td>-59.97020</td>
<td>-57.20663</td>
<td>-54.41620</td>
<td>-51.62573</td>
</tr>
<tr>
<td>-32.09194</td>
<td>-29.30136</td>
<td>-26.51077</td>
<td>-23.72017</td>
</tr>
<tr>
<td>-4.18592</td>
<td>-1.39531</td>
<td>1.39531</td>
<td>4.18592</td>
</tr>
<tr>
<td>23.72017</td>
<td>26.51077</td>
<td>29.30136</td>
<td>32.09194</td>
</tr>
</tbody>
</table>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-4.06473</td>
<td>-4.06473</td>
<td>-4.06473</td>
<td>-4.06473</td>
</tr>
<tr>
<td>-34.88252</td>
<td>-34.88252</td>
<td>-34.88252</td>
<td>-34.88252</td>
</tr>
<tr>
<td>-37.67309</td>
<td>-37.67309</td>
<td>-37.67309</td>
<td>-37.67309</td>
</tr>
<tr>
<td>43.25419</td>
<td>43.25419</td>
<td>43.25419</td>
<td>43.25419</td>
</tr>
<tr>
<td>48.83524</td>
<td>48.83524</td>
<td>48.83524</td>
<td>48.83524</td>
</tr>
<tr>
<td>52.995</td>
<td>52.995</td>
<td>52.995</td>
<td>52.995</td>
</tr>
<tr>
<td>57.603</td>
<td>57.603</td>
<td>57.603</td>
<td>57.603</td>
</tr>
</tbody>
</table>

SIGMA VALUES AT FULL LEVELS:

- 4.80330000000002E-03
- 1.30731000000000E-02
- 3.25591000000001E-02
- 6.39471000000000E-02
- 9.90432000000000E-02

SIGMA VALUES AT HALF LEVELS:

- 2.91700000000000E-03
- 7.92920000000000E-03
- 2.15539000000000E-02
- 4.91834000000000E-02
- 8.31425000000000E-02

LATITUDES:

-87.86380
-59.97020
-32.09194
-4.18592
23.72017
### Gaussian Weights

```
<table>
<thead>
<tr>
<th>Weight</th>
<th>Value</th>
<th>Weight</th>
<th>Value</th>
<th>Weight</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.59785</td>
<td>5</td>
<td>0.68143</td>
<td>7</td>
<td>0.75415</td>
</tr>
<tr>
<td>4</td>
<td>0.63574</td>
<td>6</td>
<td>0.70362</td>
<td>8</td>
<td>0.71186</td>
</tr>
<tr>
<td>2</td>
<td>0.59785</td>
<td>5</td>
<td>0.68143</td>
<td>7</td>
<td>0.75415</td>
</tr>
</tbody>
</table>
```

### Reference Temperatures

```
<table>
<thead>
<tr>
<th>Temperature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>341.904</td>
<td>200.945</td>
</tr>
<tr>
<td>123.358</td>
<td>84.715</td>
</tr>
<tr>
<td>300.000</td>
<td>300.000</td>
</tr>
</tbody>
</table>
```

### Gravity Wave Phase Speeds (M/S) for Mean State

```
<table>
<thead>
<tr>
<th>Speed</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.834762234809E-02</td>
<td></td>
</tr>
<tr>
<td>4.834762234809E-02</td>
<td></td>
</tr>
</tbody>
</table>
```

### Gravity Wave Equivalent Depths (M) for Mean State

```
<table>
<thead>
<tr>
<th>Depth</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.504579686483E-03</td>
<td></td>
</tr>
<tr>
<td>4.1470326043206E-03</td>
<td></td>
</tr>
</tbody>
</table>
```

### Reference Temperatures for Semi-Implicit Scheme

```
<table>
<thead>
<tr>
<th>Temperature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.25</td>
</tr>
<tr>
<td>0.25</td>
<td>0.25</td>
</tr>
</tbody>
</table>
```

### **Header for CCM2 History Tape**

```
*** Primary History Tape ***
CASE: f20i
TITLE: plx20
LENHD: 217
MFTYP: 43
MFILH: 2
MFILTH: 5
NRBD: 3
MAXSIZ: 57730
NDAVU: 57730
MXXX: 0
NLON: 128
NLONW: 128
```
<table>
<thead>
<tr>
<th>PLAT</th>
<th>PLEV</th>
<th>PTRM</th>
<th>PTRN</th>
<th>PTRK</th>
<th>NFLDH</th>
<th>NSTEPH</th>
<th>NSTPRH</th>
<th>NITSLF</th>
<th>NDBASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>18</td>
<td>42</td>
<td>42</td>
<td>42</td>
<td>60</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NSBASE</th>
<th>NNDCUR</th>
<th>NSCUR</th>
<th>NBDATE</th>
<th>NBSEC</th>
<th>NCODE</th>
<th>NCSEC</th>
<th>MDT</th>
<th>MHISF</th>
<th>MFSTRT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>4800</td>
<td>000901</td>
<td>0</td>
<td>000901</td>
<td>4800</td>
<td>1200</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LENHDC</th>
<th>LENHDR</th>
<th>MPSIG</th>
<th>MPLAT</th>
<th>MPWTS</th>
<th>MPFLDS</th>
<th>MPHFLD</th>
</tr>
</thead>
<tbody>
<tr>
<td>209</td>
<td>239</td>
<td>1</td>
<td>112</td>
<td>176</td>
<td>38</td>
<td>90</td>
</tr>
</tbody>
</table>

MSS PATH NAME
- DATE TIME SEQ NO.
  - CURRENT /JQUSER/ccm2/test/hist/h0001 08/13/92 14:25:34 CI9290
  - FIRST /JQUSER/ccm2/test/hist/h0001 08/13/92 14:25:16 CI9290
  - INITIAL /CSW/ccm2/367/test/hist/h0105 10/27/91 14:26:43 DUMSEQ
  - TI BOUNDARY /CCM2/T42/%data%/tibds 12/27/91 09:12:22 1
  - SST BOUNDARY /CCM2/T42/%data%/tvbs 12/23/91 15:01:43 1
  - OZONE BOUNDARY /CCM2/T42/%data%/ozn 09/21/92 13:34:26

SAVDIS: calling mswrite as follows:
```
mswrite -f TR -nomail -nowait -t 5 -v CTPUBLIC -c "DAYS: 0.000-0.055 DATES: 0.000Z 901 - 1.333Z 901" -w
```
passwd h0001 /BAT
H/ccm2/f2oi/history/h0001

SAVDIS: Disposing Mass Store Volume /JQUSER/ccm2/test/hist/h0001
Write password = passwd
Retention Time = 35 DAYS
Cartridge = CTPUBLIC
Comment Field =
DAYS: 0.000-0.500 DATES: 0.000Z 901 - 12.000Z 901
Primary history tape
Output at NSTEP = 37
Number of time samples on this tape = 2
Model Day = 0.50

---

Number of completed timesteps: 36
Time step 37 partially done to provide convectively adjusted and time filtered values for history tape.

--------- END OF MODEL RUN ---------

STOP (called by CCM2)
CP: 41.871s, Wallclock: 367.718s, 1.4% of 8-CPU Machine
HWM mem: 6570735, HWM stack: 2158456, Stack overflows: 0
+ if ( 0 != 0 ) goto err  
+ ja -sclhft

---

**Job Accounting - Command Report**

<table>
<thead>
<tr>
<th>Command</th>
<th>Started At</th>
<th>Elapsed Seconds</th>
<th>User CPU Seconds</th>
<th>Sys CPU Seconds</th>
<th>I/O Wait Sec</th>
<th>I/O Unlock Sec</th>
<th>CPU Mem</th>
<th>I/O WMem</th>
<th>Kwords</th>
<th>Log</th>
<th>I/O Memory</th>
<th>Ex</th>
</tr>
</thead>
<tbody>
<tr>
<td>ja</td>
<td>14:19:05</td>
<td>0.0790</td>
<td>0.0005</td>
<td>0.0079</td>
<td>0.0004</td>
<td>0.0689</td>
<td>0.0583</td>
<td>0.0579</td>
<td>0.00</td>
<td>1</td>
<td>124</td>
<td>0</td>
</tr>
<tr>
<td>setf</td>
<td>14:22:08</td>
<td>0.0903</td>
<td>0.0014</td>
<td>0.0151</td>
<td>0.0000</td>
<td>0.0695</td>
<td>0.0310</td>
<td>0.0000</td>
<td>0.00</td>
<td>0</td>
<td>64</td>
<td>0</td>
</tr>
<tr>
<td>sh</td>
<td>14:22:08</td>
<td>0.4021</td>
<td>0.0038</td>
<td>0.0396</td>
<td>0.0000</td>
<td>0.0656</td>
<td>0.0551</td>
<td>0.0000</td>
<td>0.00</td>
<td>0</td>
<td>113</td>
<td>0</td>
</tr>
<tr>
<td>ccm.xx.2</td>
<td>14:19:05</td>
<td>2944.8041</td>
<td>161.4891</td>
<td>11.3307</td>
<td>6.6888</td>
<td>0.8330</td>
<td>2.6393</td>
<td>6.2717</td>
<td>8737.00</td>
<td>247</td>
<td>12908</td>
<td>0</td>
</tr>
<tr>
<td>pshell</td>
<td>14:19:05</td>
<td>722.4056</td>
<td>0.0002</td>
<td>0.0043</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0307</td>
<td>0.0000</td>
<td>0.02</td>
<td>4</td>
<td>64</td>
<td>0</td>
</tr>
</tbody>
</table>

---

**Job Accounting - Command Flow Report**

```
parent ( CPU time) -> child ( CPU time) ...
```

```
ja ( 0.0084)  
ccm.xx.2 ( 43.0170)  
pshell ( 0.0045) -> sh ( 0.0435) -> setf ( 0.0165)
```

---

**Job Accounting - Summary Report**

---

Job Accounting File Name: /usr/tmp/nqs.+++++00YD/.jacct79350
<table>
<thead>
<tr>
<th>Operating System</th>
<th>sn1036 sn1036 6.1 cbh.31 CRAY Y-MP</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Name (ID)</td>
<td>jquser (xxxx)</td>
</tr>
<tr>
<td>Group Name (ID)</td>
<td>ncar (100)</td>
</tr>
<tr>
<td>Account Name (ID)</td>
<td>09013402 (3028)</td>
</tr>
<tr>
<td>Gaus Allocated</td>
<td>924.0000</td>
</tr>
<tr>
<td>Gaus Used, as of 09/30/92</td>
<td>341.0849</td>
</tr>
<tr>
<td>Job Name (ID)</td>
<td>CI9290 (79350)</td>
</tr>
<tr>
<td>Report Starts</td>
<td>08/13/92 14:19:05</td>
</tr>
<tr>
<td>Report Ends</td>
<td>08/13/92 14:31:12</td>
</tr>
<tr>
<td>Elapsed Time</td>
<td>2945 Seconds</td>
</tr>
<tr>
<td>User CPU Time</td>
<td>161.2276 Seconds</td>
</tr>
</tbody>
</table>

### Multitasking Breakdown

<table>
<thead>
<tr>
<th>Concurrent CPUs * Connect seconds = CPU seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 * 5.1104 = 5.1104</td>
</tr>
<tr>
<td>2 * 3.1034 = 6.2069</td>
</tr>
<tr>
<td>3 * 3.4108 = 10.2325</td>
</tr>
<tr>
<td>4 * 5.0808 = 20.3232</td>
</tr>
</tbody>
</table>

### System CPU Time

- 1.8623 Seconds

### I/O Wait Time (Locked)

- 6.6892 Seconds

### I/O Wait Time (Unlocked)

- 1.0371 Seconds

### CPU Time Memory Integral

- 115.2577 Mword-seconds

### SDS Time Memory Integral

- 4385.0667 Mword-seconds

### I/O Wait Time Memory Integral

- 41.9500 Mword-seconds

### Data Transferred

- 8.5322 MWords

### Maximum memory used

- 6.3008 MWords

### Maximum SDS used

- 12.0000 MWords

### Logical I/O Requests

- 252

### Physical I/O Requests

- 274

### Number of Commands

- 5

GAU Components

- Job Charge : 0.00100
- CPU Charge : 0.01197
- Memory Charge : 0.00044
Disk Activity (mwd) :  8.53225  
SDS Memory (mwd)   :  12.00000  

Charge before Queue Factor
(Excluding MSS/NTWK/TAGS) : 0.01484 GAUs
Multiplier for prem Queue : 1.50
Charged against Allocation : 0.02225 GAUs
+ exit 0
logout
Appendix C: CCM2 Parameter Definitions

The table below defines the CCM2 Fortran parameters, contained in parameter include files pagrid.com and pspect.com.
### Table C.1

**CCM2 Parameter Definitions**

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>T42 Value</th>
<th>Deck</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>plon</td>
<td>128</td>
<td>pmgrid</td>
<td>number of longitudes</td>
</tr>
<tr>
<td>plev</td>
<td>18</td>
<td>pmgrid</td>
<td>number of vertical levels</td>
</tr>
<tr>
<td>plat</td>
<td>64</td>
<td>pmgrid</td>
<td>number of latitudes</td>
</tr>
<tr>
<td>pcnst</td>
<td>1</td>
<td>pmgrid</td>
<td>number of constituents (including water vapor)</td>
</tr>
<tr>
<td>plevmx</td>
<td>4</td>
<td>pmgrid</td>
<td>number of subsurface levels</td>
</tr>
<tr>
<td>plevp</td>
<td>19</td>
<td>pmgrid</td>
<td>plev + 1</td>
</tr>
<tr>
<td>nxpt</td>
<td>1</td>
<td>pmgrid</td>
<td>number of points outside the active domain for interpolant (slt)</td>
</tr>
<tr>
<td>jintmx</td>
<td>1</td>
<td>pmgrid</td>
<td>number of extra latitudes in polar region for slt</td>
</tr>
<tr>
<td>plond</td>
<td>131</td>
<td>pmgrid</td>
<td>number of longitudes in the slt extended domain</td>
</tr>
<tr>
<td>platd</td>
<td>68</td>
<td>pmgrid</td>
<td>number of latitudes in the slt extended domain</td>
</tr>
<tr>
<td>plevd</td>
<td>72</td>
<td>pmgrid</td>
<td>fold plev, pcnst indices into one</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(plevd = plev * (3 + pcnst))</td>
</tr>
<tr>
<td>plnlv</td>
<td>2304</td>
<td>pagrid</td>
<td>length of multilevel field slice (plon * plev)</td>
</tr>
<tr>
<td>plndlv</td>
<td>2358</td>
<td>pagrid</td>
<td>length of multilevel 3-d field slice</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(plond * plev)</td>
</tr>
<tr>
<td>pbflnb</td>
<td>28165</td>
<td>pagrid</td>
<td>length of buffer 1</td>
</tr>
<tr>
<td>pbflna</td>
<td>14672</td>
<td>pagrid</td>
<td>length of buffer 2</td>
</tr>
<tr>
<td>pflenb</td>
<td>43008</td>
<td>pagrid</td>
<td>length of buffer 1, padded for unblocked I/O</td>
</tr>
<tr>
<td>pflena</td>
<td>14848</td>
<td>pagrid</td>
<td>length of buffer 2, padded for unblocked I/O</td>
</tr>
<tr>
<td>ptifld</td>
<td>11</td>
<td>pagrid</td>
<td>number of fields on time-invariant boundary dataset</td>
</tr>
<tr>
<td>ptvsfld</td>
<td>1</td>
<td>pagrid</td>
<td>number of fields on time-variant boundary dataset</td>
</tr>
<tr>
<td>ptofld</td>
<td>1</td>
<td>pagrid</td>
<td>number of fields on ozone dataset</td>
</tr>
<tr>
<td>plenhi</td>
<td>304</td>
<td>pagrid</td>
<td>length of integer header record</td>
</tr>
<tr>
<td>plenhc</td>
<td>267</td>
<td>pagrid</td>
<td>length of character header record</td>
</tr>
<tr>
<td>plenhr</td>
<td>239</td>
<td>pagrid</td>
<td>length of real header record</td>
</tr>
<tr>
<td>ptapes</td>
<td>1</td>
<td>pagrid</td>
<td>maximum number of history tapes allowed</td>
</tr>
<tr>
<td>pflds</td>
<td>89</td>
<td>pagrid</td>
<td>number of fields in master field list</td>
</tr>
</tbody>
</table>

*Valid for T42, 18-level, single constituent run.*
## Table C.1

### CCM2 Parameter Definitions

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>T42 Value</th>
<th>Deck</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ptileni</td>
<td>70</td>
<td>pagrid</td>
<td>length of time-invariant integer header</td>
</tr>
<tr>
<td>ptilenc</td>
<td>111</td>
<td>pagrid</td>
<td>length of time-invariant character header</td>
</tr>
<tr>
<td>ptvoleni</td>
<td>43</td>
<td>pagrid</td>
<td>length of time-variant ozone integer header</td>
</tr>
<tr>
<td>ptvolenc</td>
<td>93</td>
<td>pagrid</td>
<td>length of time-variant ozone character header</td>
</tr>
<tr>
<td>ptvseni</td>
<td>40</td>
<td>pagrid</td>
<td>length of time-variant SST integer header</td>
</tr>
<tr>
<td>ptvslenc</td>
<td>93</td>
<td>pagrid</td>
<td>length of time-variant SST character header</td>
</tr>
<tr>
<td>plenhis</td>
<td>37</td>
<td>pagrid</td>
<td>length of integer header scalars</td>
</tr>
<tr>
<td>plenhcs</td>
<td>89</td>
<td>pagrid</td>
<td>length of character header scalars</td>
</tr>
<tr>
<td>ptilenis</td>
<td>37</td>
<td>pagrid</td>
<td>length of time-invariant integer scalars</td>
</tr>
<tr>
<td>ptilencs</td>
<td>89</td>
<td>pagrid</td>
<td>length of time-invariant character scalars</td>
</tr>
<tr>
<td>ptolenis</td>
<td>37</td>
<td>pagrid</td>
<td>length of ozone integer header scalars</td>
</tr>
<tr>
<td>ptolencs</td>
<td>89</td>
<td>pagrid</td>
<td>length of ozone character header scalars</td>
</tr>
<tr>
<td>ptslenis</td>
<td>37</td>
<td>pagrid</td>
<td>length of time-variant integer header scalars</td>
</tr>
<tr>
<td>ptslencs</td>
<td>89</td>
<td>pagrid</td>
<td>length of time-variant character header scalars</td>
</tr>
<tr>
<td>ptrm</td>
<td>42</td>
<td>pspect</td>
<td>m truncation parameter</td>
</tr>
<tr>
<td>ptrn</td>
<td>42</td>
<td>pspect</td>
<td>n truncation parameter</td>
</tr>
<tr>
<td>ptrk</td>
<td>42</td>
<td>pspect</td>
<td>k truncation parameter</td>
</tr>
<tr>
<td>pmax</td>
<td>43</td>
<td>pspect</td>
<td>number of diagonals</td>
</tr>
<tr>
<td>pmaxp</td>
<td>44</td>
<td>pspect</td>
<td>number of diagonals plus 1</td>
</tr>
<tr>
<td>pnmax</td>
<td>43</td>
<td>pspect</td>
<td>number of values of n</td>
</tr>
<tr>
<td>pmmax</td>
<td>43</td>
<td>pspect</td>
<td>number of values of m</td>
</tr>
<tr>
<td>par0</td>
<td>42</td>
<td>pspect</td>
<td>intermediate parameter</td>
</tr>
<tr>
<td>par2</td>
<td>903</td>
<td>pspect</td>
<td>intermediate parameter</td>
</tr>
<tr>
<td>pspt</td>
<td>946</td>
<td>pspect</td>
<td>total number of complex spectral coefficients</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>retained</td>
</tr>
<tr>
<td>psp</td>
<td>1892</td>
<td>pspect</td>
<td>$2 \times pspt$ (real) size of coefficient array per</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>level</td>
</tr>
<tr>
<td>pspl</td>
<td>34056</td>
<td>pspect</td>
<td>total dimension for spectral coefficients</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>($psp \times plev$)</td>
</tr>
</tbody>
</table>

*Valid for T42, 18-level, single constituent run.
Appendix D: Master Field List

The table below shows the Master Field List, as constructed by subroutine BLDFLD. This list contains as fields which appear on the default history tape, indicated by a check mark (✓) in the first column, as well as all "inactive" fields which may be placed on a history tape by specifying input parameters AUXF or PRIMARY.

For more information concerning these Model fields, see Description of the NCAR Community Climate Model (CCM2) (Hack et al., 1992).
Table D.1
Master Field List

<table>
<thead>
<tr>
<th>Default Tape</th>
<th>Field Name</th>
<th>NPRGM1 Pointer or Common Location*</th>
<th>NPRGTL Pointer</th>
<th>NPRG Pointer</th>
<th>Field Description</th>
<th>NL</th>
<th>A/I</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>PHIS</td>
<td>NPHIS</td>
<td></td>
<td></td>
<td>surface geopotential</td>
<td>1</td>
<td>1</td>
<td>m²·s⁻²</td>
</tr>
<tr>
<td>✓</td>
<td>PS</td>
<td>NPSM1</td>
<td>NPSM2</td>
<td>NSP1</td>
<td>surface pressure</td>
<td>1</td>
<td>A</td>
<td>Pa</td>
</tr>
<tr>
<td>✓</td>
<td>T</td>
<td>/com3d/</td>
<td></td>
<td></td>
<td>temperature</td>
<td>N</td>
<td>A</td>
<td>K</td>
</tr>
<tr>
<td>✓</td>
<td>U</td>
<td>/com3d/</td>
<td></td>
<td></td>
<td>zonal wind component</td>
<td>N</td>
<td>A</td>
<td>m·s⁻¹</td>
</tr>
<tr>
<td>✓</td>
<td>V</td>
<td>/com3d/</td>
<td></td>
<td></td>
<td>meridional wind component</td>
<td>N</td>
<td>A</td>
<td>m·s⁻¹</td>
</tr>
<tr>
<td></td>
<td>ETADOT</td>
<td></td>
<td></td>
<td></td>
<td>Etadot on half levels</td>
<td>N</td>
<td>A</td>
<td>1/s</td>
</tr>
<tr>
<td>✓</td>
<td>Q (or TRxx)</td>
<td>/com3d/</td>
<td></td>
<td></td>
<td>tracer (first is always specific humidity)</td>
<td>N</td>
<td>A</td>
<td>Kg·Kg⁻¹</td>
</tr>
<tr>
<td></td>
<td>HAxx</td>
<td>hqfcst (LINEMS)</td>
<td></td>
<td></td>
<td>horizontal advection of tracer</td>
<td>N</td>
<td>A</td>
<td>Kg·Kg⁻¹·s⁻¹</td>
</tr>
<tr>
<td></td>
<td>VAxx</td>
<td>vqfcst (LINEMS)</td>
<td></td>
<td></td>
<td>vertical advection of tracer</td>
<td>N</td>
<td>A</td>
<td>Kg·Kg⁻¹·s⁻¹</td>
</tr>
<tr>
<td></td>
<td>DFxx</td>
<td>dqfx3 (LINEMS)</td>
<td></td>
<td></td>
<td>SLT fixer of tracer</td>
<td>N</td>
<td>A</td>
<td>Kg·Kg⁻¹·s⁻¹</td>
</tr>
<tr>
<td>✓</td>
<td>TAxx</td>
<td>ta (LINEMS)</td>
<td></td>
<td></td>
<td>total advection of tracer</td>
<td>N</td>
<td>A</td>
<td>Kg·Kg⁻¹·s⁻¹</td>
</tr>
<tr>
<td>✓</td>
<td>VCxx</td>
<td>dqv (VDINTR)</td>
<td></td>
<td></td>
<td>diffusion tendency of tracer</td>
<td>N</td>
<td>A</td>
<td>Kg·Kg⁻¹·s⁻¹</td>
</tr>
<tr>
<td>✓</td>
<td>DCxx</td>
<td>dqcond (LINEMS)</td>
<td></td>
<td></td>
<td>tracer tendency from adjustment physics</td>
<td>N</td>
<td>A</td>
<td>Kg·Kg⁻¹·s⁻¹</td>
</tr>
<tr>
<td></td>
<td>TExx</td>
<td></td>
<td></td>
<td></td>
<td>time tendency of tracer</td>
<td>N</td>
<td>A</td>
<td>Kg·Kg⁻¹·s⁻¹</td>
</tr>
<tr>
<td></td>
<td>SSxx</td>
<td></td>
<td></td>
<td></td>
<td>tracer source/sinks (penst-1 values)</td>
<td>N</td>
<td>A</td>
<td>Kg·Kg⁻¹·s⁻¹</td>
</tr>
<tr>
<td></td>
<td>SFxx</td>
<td></td>
<td></td>
<td></td>
<td>tracer surface fluxes (penst-1 values)</td>
<td>N</td>
<td>A</td>
<td>Kg·m²·s</td>
</tr>
<tr>
<td></td>
<td>DUH</td>
<td>NDUHM1</td>
<td>NDUHP1</td>
<td></td>
<td>u horizontal diffusive heating rate</td>
<td>N</td>
<td>I</td>
<td>K·s⁻¹</td>
</tr>
<tr>
<td></td>
<td>DVH</td>
<td>NDVHM1</td>
<td>NDVHP1</td>
<td></td>
<td>v horizontal diffusive heating rate</td>
<td>N</td>
<td>I</td>
<td>K·s⁻¹</td>
</tr>
<tr>
<td>✓</td>
<td>DTH</td>
<td>NDTHM1</td>
<td>NDTHP1</td>
<td></td>
<td>T horizontal diffusion</td>
<td>N</td>
<td>A</td>
<td>K·s⁻¹</td>
</tr>
</tbody>
</table>

* Local arrays are indicated by array name and subroutine name (e.g., ta (LINEMS)). Pointers are to main Model buffer (see "Model Buffer" on page 80).
<table>
<thead>
<tr>
<th>Default Tape</th>
<th>Field Name</th>
<th>NPRGM1 Pointer or Common Location</th>
<th>NPRGTLocTPorteR</th>
<th>Field Description</th>
<th>NL</th>
<th>A/I</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>NDQH</td>
<td>NDQHM1</td>
<td>NDQHP1</td>
<td></td>
<td>q horizontal diffusion</td>
<td>N</td>
<td>A</td>
<td>Kg·Kg⁻¹·s⁻¹</td>
</tr>
<tr>
<td>✓ ORO</td>
<td>NORO</td>
<td></td>
<td></td>
<td>surface type flag: 1</td>
<td>1</td>
<td>I</td>
<td>flag</td>
</tr>
<tr>
<td>✓ WET</td>
<td>NWS</td>
<td></td>
<td></td>
<td>soil moisture</td>
<td>1</td>
<td>A</td>
<td>m</td>
</tr>
<tr>
<td>✓ SNOWH</td>
<td>NSN</td>
<td></td>
<td></td>
<td>water equivalent snow depth</td>
<td>1</td>
<td>A</td>
<td>m</td>
</tr>
<tr>
<td>✓ PRECL</td>
<td>precl</td>
<td>(LINEMS)</td>
<td></td>
<td>large-scale stable precipitation</td>
<td>1</td>
<td>A</td>
<td>m·s⁻¹</td>
</tr>
<tr>
<td>✓ PRECC</td>
<td>precc</td>
<td>(LINEMS)</td>
<td></td>
<td>convective precipitation</td>
<td>1</td>
<td>A</td>
<td>m·s⁻¹</td>
</tr>
<tr>
<td>PRECSL</td>
<td></td>
<td></td>
<td></td>
<td>large-scale stable snowfall</td>
<td>1</td>
<td>A</td>
<td>m·s⁻¹</td>
</tr>
<tr>
<td>PRECSC</td>
<td></td>
<td></td>
<td></td>
<td>convective snowfall</td>
<td>1</td>
<td>A</td>
<td>m·s⁻¹</td>
</tr>
<tr>
<td>RUNOFF</td>
<td>NRNFP1</td>
<td></td>
<td></td>
<td>soil moisture runoff</td>
<td>1</td>
<td>A</td>
<td>m·s⁻¹</td>
</tr>
<tr>
<td>✓ SHFLX</td>
<td>shflx</td>
<td>(PHYS)</td>
<td></td>
<td>surface sensible heat flux</td>
<td>1</td>
<td>A</td>
<td>W·m⁻²</td>
</tr>
<tr>
<td>✓ LHFLX</td>
<td>lhflx</td>
<td>(PHYS)</td>
<td></td>
<td>surface latent heat flux</td>
<td>1</td>
<td>A</td>
<td>W·m⁻²</td>
</tr>
<tr>
<td>✓ QFLX</td>
<td>cflx</td>
<td>(PHYS)</td>
<td></td>
<td>surface water flux</td>
<td>1</td>
<td>A</td>
<td>Kg·m⁻²·s⁻¹</td>
</tr>
<tr>
<td>✓ PBLH</td>
<td>ustar</td>
<td>(VDINTR)</td>
<td></td>
<td>planetary boundary layer height</td>
<td>1</td>
<td>A</td>
<td>m</td>
</tr>
<tr>
<td>✓ USTAR</td>
<td>ustar</td>
<td>(VDINTR)</td>
<td></td>
<td>surface friction velocity</td>
<td>1</td>
<td>A</td>
<td>m·s⁻¹</td>
</tr>
<tr>
<td>T10</td>
<td></td>
<td></td>
<td></td>
<td>10 m potential temperature</td>
<td>1</td>
<td>A</td>
<td>K</td>
</tr>
<tr>
<td>Q10</td>
<td></td>
<td></td>
<td></td>
<td>10 m specific humidity</td>
<td>1</td>
<td>A</td>
<td>Kg·Kg⁻¹</td>
</tr>
<tr>
<td>CGH</td>
<td></td>
<td></td>
<td></td>
<td>pbl nonlocal transport, heat</td>
<td>N</td>
<td>A</td>
<td>K·m⁻¹</td>
</tr>
<tr>
<td>CGQ</td>
<td></td>
<td></td>
<td></td>
<td>pbl nonlocal transport, humidity</td>
<td>N</td>
<td>A</td>
<td>1·m⁻¹</td>
</tr>
<tr>
<td>CGS</td>
<td></td>
<td></td>
<td></td>
<td>pbl nonlocal transport, multiplier</td>
<td>N</td>
<td>A</td>
<td>s·m⁻²</td>
</tr>
<tr>
<td>Default Tape</td>
<td>Field Name</td>
<td>NPRGM1 Pointer or Common Location</td>
<td>NPRGTL Pointer</td>
<td>NPRG Pointer</td>
<td>Field Description</td>
<td>NL</td>
<td>A/I</td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
<td>-----------------------------------</td>
<td>----------------</td>
<td>-------------</td>
<td>-------------------</td>
<td>----</td>
<td>-----</td>
</tr>
<tr>
<td>✓</td>
<td>TPERT</td>
<td>NTPERT</td>
<td></td>
<td></td>
<td>pbl plume tempera-</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>QPERT</td>
<td>NQPERT</td>
<td></td>
<td></td>
<td>ture perturbation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>KVM</td>
<td></td>
<td></td>
<td></td>
<td>diffusivity for mo-</td>
<td>N</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>DUV</td>
<td></td>
<td></td>
<td></td>
<td>U vertical diffu-</td>
<td>N</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>DWVV</td>
<td></td>
<td></td>
<td></td>
<td>sion</td>
<td>N</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>DTV</td>
<td></td>
<td></td>
<td></td>
<td>T vertical diffu-</td>
<td>N</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>FSNS</td>
<td>fsns (PHYS)</td>
<td></td>
<td></td>
<td>net downward sol-</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>FLNS</td>
<td>flns (PHYS)</td>
<td></td>
<td></td>
<td>net upward longwave</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>FLNT</td>
<td>flnt (PHYS)</td>
<td></td>
<td></td>
<td>flux at top of mod-</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>FSNT</td>
<td>fsnt (PHYS)</td>
<td></td>
<td></td>
<td>el model</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>CLOUD</td>
<td>cld (PHYS)</td>
<td></td>
<td></td>
<td>cloud fraction</td>
<td>N</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>SETLWP</td>
<td></td>
<td></td>
<td></td>
<td>prescribed liquid water</td>
<td>N</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>CLDLWP</td>
<td></td>
<td></td>
<td></td>
<td>water path</td>
<td>N</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>EFFCLD</td>
<td>effcld (PHYS)</td>
<td></td>
<td></td>
<td>effective cloud</td>
<td>N</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>FLNTC</td>
<td>flntc (PHYS)</td>
<td></td>
<td></td>
<td>clearsky up-</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>FSNTC</td>
<td>fsntc (PHYS)</td>
<td></td>
<td></td>
<td>clearsky down-</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>FLNSC</td>
<td>flnsc (PHYS)</td>
<td></td>
<td></td>
<td>clearsky up-</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>FSNSC</td>
<td>fnsnc (PHYS)</td>
<td></td>
<td></td>
<td>clearsky down-</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>Default Tape</td>
<td>Field Name</td>
<td>NPRGM1 Pointer or Common Location*</td>
<td>NPRGTL Pointer</td>
<td>NPRG Pointer</td>
<td>Field Description</td>
<td>NL</td>
<td>A/I</td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
<td>-----------------------------------</td>
<td>----------------</td>
<td>--------------</td>
<td>----------------</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>✓</td>
<td>OMEGA</td>
<td>omga (LINEMS)</td>
<td></td>
<td></td>
<td>vertical pressure velocity</td>
<td>N</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>DQP</td>
<td>qc (LINEMS)</td>
<td></td>
<td></td>
<td>Q tendency from rainout</td>
<td>N</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>TAUX</td>
<td>taux (PHYS)</td>
<td></td>
<td></td>
<td>zonal surface stress</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>TAUY</td>
<td>tauy (PHYS)</td>
<td></td>
<td></td>
<td>meridional surface stress</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>SFRFRAD</td>
<td>NDRPI</td>
<td></td>
<td></td>
<td>radiative flux absorbed at the surface</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>QRS</td>
<td>NQRS</td>
<td></td>
<td></td>
<td>solar heating rate</td>
<td>N</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>QRL</td>
<td>NQRL</td>
<td></td>
<td></td>
<td>longwave heating rate</td>
<td>N</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>CLDTOT</td>
<td>cltot (PHYS)</td>
<td></td>
<td></td>
<td>random overlap total cloud cover</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>CLDLow</td>
<td>cllow (PHYS)</td>
<td></td>
<td></td>
<td>random overlap low cloud cover</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>CLDMED</td>
<td>clmed (PHYS)</td>
<td></td>
<td></td>
<td>random overlap medium cloud cover</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>CLDHGH</td>
<td>clhgh (PHYS)</td>
<td></td>
<td></td>
<td>random overlap high cloud cover</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>TOTLWP</td>
<td></td>
<td></td>
<td></td>
<td>total liquid water path</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>TS1</td>
<td>NTSSUB</td>
<td></td>
<td></td>
<td>surface temperature (level 1)</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>TS2</td>
<td>NTSSUB</td>
<td></td>
<td></td>
<td>subsurface temperature, level 2</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>TS3</td>
<td>NTSSUB</td>
<td></td>
<td></td>
<td>subsurface temperature, level 3</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>TS4</td>
<td>NTSSUB</td>
<td></td>
<td></td>
<td>subsurface temperature, level 4</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>✓</td>
<td>SOLIN</td>
<td>solin (PHYS)</td>
<td></td>
<td></td>
<td>solar insolation</td>
<td>1</td>
<td>A</td>
</tr>
</tbody>
</table>

Master Field List
### Table D.1
Master Field List

<table>
<thead>
<tr>
<th>Default Tape</th>
<th>Field Name</th>
<th>NPRGM1 Pointer or Common Location*</th>
<th>NPRGTL Pointer</th>
<th>NPRG Pointer</th>
<th>Field Description</th>
<th>NL</th>
<th>A/I</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>UTGW</td>
<td>utgw (GWINTR)</td>
<td></td>
<td></td>
<td>gravity wave dragU tendency</td>
<td>N</td>
<td>A</td>
<td>(m \cdot s^{-2})</td>
</tr>
<tr>
<td>✓</td>
<td>VTGW</td>
<td>vtgw (GWINTR)</td>
<td></td>
<td></td>
<td>gravity wave dragV tendency</td>
<td>N</td>
<td>A</td>
<td>(m \cdot s^{-2})</td>
</tr>
<tr>
<td>✓</td>
<td>TAUGWX</td>
<td>taugx (GWINTR)</td>
<td></td>
<td></td>
<td>gravity wave drag zonal surface stress</td>
<td>1</td>
<td>A</td>
<td>(N \cdot m^{-2})</td>
</tr>
<tr>
<td>✓</td>
<td>TAUGWY</td>
<td>taugy (GWINTR)</td>
<td></td>
<td></td>
<td>gravity wave drag meridional surface stress</td>
<td>1</td>
<td>A</td>
<td>(N \cdot m^{-2})</td>
</tr>
<tr>
<td>✓</td>
<td>DTCOND</td>
<td>dtcond (LINEMS)</td>
<td></td>
<td></td>
<td>T tendency from adjustment physics</td>
<td>N</td>
<td>A</td>
<td>(K \cdot s^{-1})</td>
</tr>
<tr>
<td>✓</td>
<td>CMFDT</td>
<td>cmfdt (LINEMS)</td>
<td></td>
<td></td>
<td>T tendency from moist convection</td>
<td>N</td>
<td>A</td>
<td>(K \cdot s^{-1})</td>
</tr>
<tr>
<td>✓</td>
<td>CMFDQ</td>
<td>cmfdq (LINEMS)</td>
<td></td>
<td></td>
<td>Q tendency from moist convection</td>
<td>N</td>
<td>A</td>
<td>(Kg \cdot Kg^{-1} \cdot s^{-1})</td>
</tr>
<tr>
<td></td>
<td>CMFDQR</td>
<td></td>
<td></td>
<td></td>
<td>rainout (condensation)</td>
<td>N</td>
<td>A</td>
<td>(Kg \cdot Kg^{-1} \cdot s^{-1})</td>
</tr>
<tr>
<td>✓</td>
<td>CMFMC</td>
<td>cmfmc (LINEMS)</td>
<td></td>
<td></td>
<td>total convective mass flux</td>
<td>N</td>
<td>A</td>
<td>(Kg \cdot m^{-2} \cdot s^{-1})</td>
</tr>
<tr>
<td>✓</td>
<td>CMFSL</td>
<td>cmfsl (LINEMS)</td>
<td></td>
<td></td>
<td>convective liquid water static energy flux</td>
<td>N</td>
<td>A</td>
<td>(W \cdot m^{-2})</td>
</tr>
<tr>
<td>✓</td>
<td>CMFLQ</td>
<td>cmflq (LINEMS)</td>
<td></td>
<td></td>
<td>convective total water flux</td>
<td>N</td>
<td>A</td>
<td>(W \cdot m^{-2})</td>
</tr>
<tr>
<td>✓</td>
<td>CNVCLD</td>
<td>clc (PHYS)</td>
<td></td>
<td></td>
<td>convective cloud fraction</td>
<td>1</td>
<td>A</td>
<td>fraction</td>
</tr>
</tbody>
</table>
Index

A
absorptivity/emissivity calculation 18, 22, 53, 55–56, 63, 94, 97, 107
array bounds checker 82, 121, 123, 127
assign
   see file assign
automatic array,Fortran extension 83
autotasking preprocessor
   see preprocessors,autotasking
auxiliary history tape 15, 16, 26–28, 63, 64, 71, 75, 90, 93, 99, 115
averaged fields
   see history tape,averaged fields

B
base date 17, 21, 30, 31, 42, 62
base day 17, 41
batch 1, 7, 91
boundary dataset 13, 14, 20, 35–37, 40, 44, 57, 61, 68, 90, 118
ozone mixing ratios 13, 14, 20, 21, 22, 30, 31, 37, 44, 67, 68
time-invariant 13, 14, 20, 35, 36, 44, 70
time-variant 13, 14, 18, 20, 21, 22, 32, 35, 36, 44, 70, 90, 104
branch run
   see continuation run,branch
buffer
   history tape 75, 83, 103, 105
   main Model 49, 50, 54, 65, 80–84, 89, 96, 101, 102, 105, 106, 116, 119
continuation run 35, 54, 90
   branch 14, 20, 26, 27, 42, 64, 91, 96, 98, 99
   regeneration 14, 16, 20, 25, 42, 69, 96, 98, 109
   restart 14, 23, 24, 53, 69, 90, 96–98, 103, 119
control run 1, 26, 27
CPU time 5, 11, 60, 61, 89, 96, 99, 121
Cray disk scrubber 91
current directory 7, 90, 91

d
data structures
   contiguity 74, 80, 81, 84, 111
   gridpoint 38, 78–86, 101, 112, 118
   extended grid 104, 118
   main Model buffer
      see buffer,main Model
   three-dimensional arrays(/com3d/) 50, 53, 74, 84–86, 96, 97, 102, 103
   spectral 73, 77, 86, 106, 111
   stack-based 74, 75, 83
   static 84, 121
debug,Fortran symbolic debugger 9, 121, 122
debugging Model changes 9, 119
   comparing history tapes 122
directory search path 10, 11
double buffering 74, 89

E
end-of-file 31, 68, 69, 70
environment variable 6, 7, 10, 11, 37, 62, 91, 93, 95, 98, 105, 109
error message
   Model 1, 57, 61, 119, 122
   system 8, 11, 58, 92, 95, 119, 120, 121, 122
   error processing 9, 61, 95

F
field list information
   see header,field list
file assign 8, 57, 58, 83, 89, 90–92, 95, 119
first header record 34, 40, 41, 42
format code,history tape
   see history tape,format type
Fortran compiler
   see cft77
Fortran direct access I/O 75, 89, 111
Fortran heap 6, 83
Fortran stack 6, 74, 75, 83, 84, 112, 113, 122

C
ftp 9

G
Gaussian grid
see data structure, gridpoint
Gaussian latitude
see latitude, coordinate
Gaussian quadrature 75, 103, 106, 111
Gaussian weights 34, 40, 42, 45, 59
gpp preprocessor
see preprocessors, gpp
gridpoint data
see data structures, gridpoint

H
header
boundary dataset 57, 67, 70
character record 34, 40, 42, 43
field list 34, 42, 45, 46, 48, 59, 73
history tape
see history tape, header
initial dataset 17, 21, 34, 35, 65, 66, 90
integer record 34, 40, 41, 42
real record 34, 37, 40, 42, 45
heap, Fortran managed memory
see Fortran heap
averaged fields 16, 46, 49, 64, 75
buffer 75, 83, 103, 105
compare program 122
format 32, 35, 37, 40, 48, 49
format type (mftyp) 40, 47
header 38–48, 58, 60, 68, 69, 75, 90, 98, 104
history tape handler 75, 88, 104
instantaneous fields 16, 22, 28, 46, 49, 64, 75, 83
latitude record 32, 36, 38, 41, 48–52, 75, 89
master field list 16, 28, 29, 64, 75, 115, 116
maximum field 16, 46, 64, 75
minimum field 16, 46, 64, 75
naming 7, 15, 20, 26, 40, 62, 93
random order of 75, 102
time samples 21, 35, 36, 37, 60, 90
unit 90
write frequency 15, 22, 26, 27, 42, 48, 55, 63, 83, 97, 99, 107
hybrid vertical coordinate
see vertical coordinate

I
include files 3, 9, 10, 12, 88, 89, 123, 124
in-core 73, 74, 75, 77, 81, 86
indefinite 8, 35, 121
initial run 13, 14, 18, 20, 28, 35, 42, 90, 96, 98, 101,
103
input parameter
default value 13, 15, 20–23, 37, 92, 93, 97
preset value 13, 14, 101
see also namelist input parameter
instantaneous fields
see history tape, instantaneous fields
ishell 9, 69, 94, 95
iteration
see timestep

J
job accounting 60, 114
job sequence number 40, 43, 44

L
latitude
index 48, 79, 80, 85, 111, 112
latitude loop (scan) 81, 104, 105, 106, 110
record
see history tape, latitude record
levels, vertical
see vertical coordinate
libraries, system 8, 11, 94
link to /usr/tmp 16, 22, 38, 92, 93, 94, 95
local data storage, stack-based 74, 75, 83
logical unit assign
see file assign
login 7, 91
longitude coordinate 37, 41, 48, 78, 79, 80, 84, 85, 103, 116, 118

M
Mass Store System
pathname 7, 13, 14, 21, 35, 36, 37, 43, 44, 53, 90, 94, 97, 98
public volume 15, 20, 29, 38
retention period 15, 20, 29, 38, 99
stage to disk 22, 91, 92, 94
virtual volume 15, 20, 29, 38
write password 15, 21, 38
master field list
see history tape, master field list
maximum history tape field
see history tape, maximum field
memory requirement 5, 6, 60, 61, 73, 74, 84, 86, 105, 113, 114, 118, 120, 121, 122
memory-resident 73, 74, 75, 77, 81, 86
meridional wind component 33, 50, 84
minimum history tape field
see history tape, minimum field
Modular Processor
see CCM Modular Processor
multiyear time-variant boundary datasets 14, 21, 30, 36, 68, 70
Index Page 1-3

N
Network Queueing System 1, 5, 6, 7, 119, 120
news for CCM2 users 1

O
orography flag 32, 33, 50, 102
orography standard deviations 20, 35, 36
out-of-core implementation 9, 73, 74, 77–84, 89, 91, 105, 107, 109
output dataset naming see history tape naming

P
packing 20, 21, 41, 46, 48
density 15, 21, 64, 96
pathname,MSS
see Mass Store System,pathname
permanent disk,Cray 1, 20, 35, 37, 90, 118
phis (surface geopotential) 32, 33, 50, 102
physical parameterizations 21, 32, 73–75, 87–89, 100, 102, 106, 115, 118, 119, 125
cloud 107
convective transport 117
gravity wave drag scheme 20, 52, 107
horizontal diffusion 106, 111
initialization 88, 119
planetary boundary layer scheme 89, 107
rayleigh friction 107
surface temperature 107
vertical diffusion 107, 117
physics
adjustment 106, 118
tendency 106
pointer
field,history tape header 46, 48, 59
Fortran 82
three-dimensional array 86, 88, 105
to main buffer 49, 50, 75, 82, 89, 116
preallocation of disk files 58, 68, 91, 92
preprocessors
autotasking 2, 6, 8, 11, 109, 110, 113
segldr UNICOS loader 113, 114, 120, 121
shell 6, 9
Sigma vertical coordinate
see vertical coordinate
single-threaded 2, 3, 6, 11, 48, 75, 109, 110, 111, 112, 113, 119, 121
source code,CCM2 1, 9, 10, 121
spectral
coefficients 105, 112
data structures 73, 77, 86, 87, 106, 111
transform 74, 106
truncation 34, 41, 71, 87, 88, 102, 111
SSD files 58
see also work files
SSD requirement 5, 6, 119
SST,sea-surface temperatures 13, 14, 18, 20, 21, 22, 32, 35, 36, 44, 70, 90, 104
stack
see Fortran stack
stage MSS file to disk 22, 91, 92, 94

Q
q,water vapor constituent
see constituent,first
QSUB directive
see Network Queueing System

R
master 53, 93, 94, 97
naming 93, 94
primary 54, 55, 92, 94, 97
secondary 55, 56, 92, 94, 97
write frequency 16, 22, 24, 25
regeneration run see continuation run,regeneration
relocatable binary files 2, 8, 11
resolution 6, 21, 22, 35, 54, 55, 66, 71, 88, 115, 118
restart dataset 15, 20, 21, 23, 37, 38, 42, 53, 54, 55, 65, 69, 90, 92, 94, 97, 98, 107
restart run see continuation run,restart
retention period,MSS
see Mass Store System,retention period

S
search path see directory search path
second header record 34, 40, 42, 43
Secondary Data Segments 8, 58, 74, 83, 89, 91, 100, 101, 109, 120
segldr UNICOS loader 113, 114, 120, 121
shel 6, 9
Sigma vertical coordinate
see vertical coordinate
single-threaded 2, 3, 6, 11, 48, 75, 109, 110, 111, 112, 113, 119, 121
source code,CCM2 1, 9, 10, 121
spectral
coefficients 105, 112
data structures 73, 77, 86, 87, 106, 111
transform 74, 106
truncation 34, 41, 71, 87, 88, 102, 111
SSD files 58
see also work files
SSD requirement 5, 6, 119
SST,sea-surface temperatures 13, 14, 18, 20, 21, 22, 32, 35, 36, 44, 70, 90, 104
stack
see Fortran stack
stage MSS file to disk 22, 91, 92, 94
statement function 62, 116, 125
static data allocation 84, 121
surface albedo data 20, 35, 36
surface geopotential 32, 33, 50, 59, 102
surface pressure 33, 50, 59, 79, 81, 84, 86, 102, 104
surface type flag 32, 33, 50, 102
symbolic debugger
  see debug

T
temperature 33, 50, 51, 59, 84, 86, 104, 107
  subsurface 33, 52, 81, 102
  surface 33, 52, 107
third header record 34, 37, 40, 42, 45
three-dimensional array
  pointer 86, 88, 105
  time index 105
time filter 17, 103, 106, 107
time integration 37, 74, 79, 88, 102, 103, 106, 111
time samples
  see history tape, time samples
timestep 14, 17, 21, 22, 27, 41, 42, 55, 60, 79, 86, 98,
    103–107, 118, 122
  leapfrog 102, 107
TMPDIR 7, 37
traceback 68
trunckation parameter
  see spectral, truncation

U
unblocked I/O 74, 83, 89, 91
UNICOS ja command 60, 114
UNICOS ln command 16, 22, 38, 92, 93, 94, 95
units of Model fields 46, 47, 50, 59, 107, 115
University Corporation for Atmospheric Research (UCAR) 1, 57

V
vectorization 81, 87, 88, 121, 127
vertical coordinate 33, 34, 35, 37, 40, 41, 42, 45, 46, 49,
    57, 59, 63, 77–78, 80, 81, 103, 106, 118

W
water vapor
  see constituent, first
weights, Gaussian 34, 40, 42, 45, 59
work files 53, 55, 58, 75, 89, 91, 96, 97, 101, 102, 105,
    106, 107, 111, 116
write frequency
  history tape
    see history tape, write frequency
  regeneration data 16, 22, 24, 25
write password, MSS
  see Mass Store System, write password