NCAR/TN-381+IA
NCAR TECHNICAL NOTE

October 1992

# A User's Guide to the Penn State/NCAR Mesoscale Modeling System

David O. Gill

MESOSCALE AND MICROSCALE METEOROLOGY DIVISION

NATIONAL CENTER FOR ATMOSPHERIC RESEARCH
BOULDER, COLORADO

# Table of Contents

# Table of Contents

# Table of Contents

# List of Figures

# Preface

The Pennsylvania State University/National Center for Atmospheric Research (PSU/NCAR) Mesoscale Modeling system (the MM4 system) is a jointly supported tool for research into regional numerical weather prediction. The standard MM4 modeling package employs a Cressman multi–scan isobaric and surface analysis, with a hydrostatic predictive component using a leap frog integration of the flux form of the primitive equations on sigma coordinates. An experimental version has expanded the data ingest routines to allow hybrid isentropic–isobaric + surface analyses. Experimental versions of the model allow split–explicit time integration, several cumulus parameterizations coupled with an explicit moisture scheme, multiple levels of movable nests, relaxation of the hydrostatic assumptions, additional planetary boundary layer schemes, and microphysical packages. Due to the developmental nature of the modeling system, periodic upgrades in documentation are required to keep the manuals in accord with the programs. This document supersedes *Penn State/NCAR Mesoscale Model User's Manual – Version 8* by Haagenson *et al* (1990).

The MM4 system is a series of FORTRAN programs and accompanying C–shell scripts that runs under UNICOS on the Cray Y-MP8/864 at NCAR. Each of these shells is a stand-alone script which must be submitted in a specific sequence to the Cray. The MM4 package is a flexible modeling system, implying that a good deal of user interaction is required. Each of the major components of the modeling system has a section in this document describing its standard usage; the initial forecast for any particular case involves running them all.

Support for users of the MM4 system is limited. Such support entails providing all available documentation about MM4, helping new users execute the programs in a black box mode, and providing technical assistance when errors occur during a standard run. Neither NCAR nor PSU can support users wishing to undertake research by modifying the programs or by porting the code to a different computing environment.

Where possible, text was lifted directly from FORTRAN programs or C-shells to avoid transcription errors. You may forward detected misgivings in this document to *gill@NCAR.UCAR.edu* for inclusion in any subsequent User's Guide releases.

David O. Gill
19 October 1992

# Acknowledgments

A User's Guide for a research model is never quite finished. Between the original draft and this version, fully six program were modified to such a degree that documentation changes were necessary. To that end I would like to thank Phil Haagenson and Bill Kuo for the allotment of time that somehow did not end.

Significant improvements in the document were suggested by the reviewers. I would like to specifically thank Leigh Angus, Sue Chen, Yong-Run Guo, Phil Haagenson, Hope Hamilton, Kevin Manning, and John Street. They have added to the document's readability as well as technical accuracy.

# Chapter 1 Introduction

During the 1970s, a numerical hydrodynamical model suitable for predicting mesometeorological flows was developed (Anthes and Warner, 1978) at The Pennsylvania State University (PSU). By the 1980s, this program and the accompanying auxiliary support code for pre- and post-processing of the data had become research tools at both PSU and the National Center for Atmospheric Research (NCAR). For an overview of key areas of research conducted with MM4, see Anthes (1990)

The current version of the modeling system has been modified (during the COS to UNICOS conversion) to take advantage of some capabilities available on the NCAR computers and to streamline the process of generating a forecast. Figure 1 shows the general flow of the set of programs required for a typical forecast. The rectangular boxes down the center represent the programs that are run on the NCAR machines, and top to bottom they represent the order in which the job decks are typically run. The parallelograms on the left represent all of the archived data available from the Scientific Computing Division (SCD) at NCAR, such as global analyses and rawinsonde observations. The bold arrows show the path of most of the internally computed data sets that are output from one component and used as input in the next program. The ellipses on the right show the files located in the ~mesouser/Decks subdirectories on the NCAR Cray Y-MP 8/864 that the user must copy to their own disk and modify for each job submission.

The modeling system is maintained by the MesoUser manager at NCAR. All questions concerning the MM4 modeling package should be sent to *mesouser@NCAR.UCAR.edu* on the internet. The MesoUser manager is charged with: 1) maintaining a working set of programs and the accompanying set of C-shell scripts; 2) addressing and notifying users of uncovered problems; 3) testing bug fixes and enhancements manager to the standard package prior to general release; and 4) providing user service. The MesoUser manager keeps all of the modeling package on shavano (one of the Cray Y-MPs) at NCAR so that those with accounts on the SCD supercomputers have access to the MM4 system. Each major component of the modeling system has a subdirectory under ~mesouser/Decks.

There are currently two streams of data from which the analysis codes may pull in the first-guess background fields and the observations. The Data Support group of SCD has maintained synoptic global analyses from NMC and ECMWF, and global coverage for upper air and surface stations. These files are available several weeks after the analyses are generated or the observations are recorded. On shavano are the catalog archive files that need to be searched occasionally to see when new data becomes available. These catalog listing files are located on ~mesouser/catalogs.

The other data source available to the modeling package is in real time. This data is supplied by UNIDATA and is also maintained by the Data Support group in SCD. The observation coverage is not global, but the amount of data is steadily increasing. The global forecast that supplies the background first-guess fields is the medium range forecast (MRF) product.

This document focuses essentially on explaining the use of the individual programs.

This user's manual will not attempt to explain the more technical aspects of any section of code. There is a chapter for each program of the modeling system. Each of the chapters reviews the user input required for the successful run, compares and contrasts the particular job deck with aspects of the others, and provides hints gleaned from experience. Examples of metacode that are provided incidental to the primary purpose of a program are discussed and displayed (samples of available output from the primary graphics programs are displayed with little explanation).

Appendices are provided to allow the user to have more specific information provided. Appendix A details the standard output file and the record header from each program. A sample FORTRAN code is provided to read each file, and a brief description of each variable is given. Appendix B discusses the user provided ASCII input files (both the optional and required files). A FORTRAN subroutine (where appropriate) is provided to remove the ambiguity of the format statements, and a description of all variables is given. Appendix C attempts to list all MM4 modeling system colloquialisms contained in this document.

The remainder of this chapter addresses the fundamental aspects of the modeling system: the different programs, the various vertical and horizontal grids, and examples of domain configurations. Following this is chapter 2, which introduces the user to the novelties particular to the MM4 modeling system. Chapter 3 describes the sample jobs that are available to the user. Chapter 4 through chapter 12 each concentrate on a specific component of the modeling system.

To provide better service to the users of the modeling system, a formal registration procedure has been established. Registered users of the modeling package receive regular updates of new capabilities, discovered bugs and their influence, bug fixes, and information concerning the annual workshop. To initiate the registration procedure, send e-mail to the MesoUser manager at NCAR (*mesouser@NCAR.UCAR.edu*), stating an interest in becoming a registered user of the MM4 modeling system. The registration materials will be forwarded to you.

## 1.1 Horizontal Grid Specification

The MM4 modeling system allows a part of the domain to utilize a higher horizontal resolution, called a nest or a fine grid. The ratio of the coarse-grid distance to the fine-grid distance is 3::1, and may not be modified if the user is planning a two-way interacting nested model run. The horizontal domain is the Arakawa B–Grid (Arakawa and Lamb, 1977), where the horizontal wind components are defined on the "dot" points and the temperature, moisture and pressure fields are defined on the "cross" points (figure 2). The nested grid is defined by the coarse-grid dot points on which the nested domain starts and stops. The nested dimensions must be set up to allow the fine grid to begin and end on a coarse-grid dot point. The nest/coarse boundary interface should be no closer than approximately ten grid points to the nearest coarse-grid outer boundary.

Throughout the document, references will be made to specific horizontal grids. The coarse grid is the outer-most horizontal domain that is used by the model during the

forecast. The fine grids are the inner domains used by the forecast, with higher resolution than the coarse grid. The expanded domain is the grid used for analyzing the isobaric data. This domain may be larger than the coarse grid, and is cut down to provide the coarse grid at the end of RAWINS. Even when the coarse and expanded domains are the same size, it is conventional to talk about the analysis programs running on the expanded domain.

Throughout the entire modeling system, the definition of direction in the different grids is consistent. Each domain is a rectangular array of grid points. Moving from the left edge to the right edge in a domain is moving in the J-direction or the x-direction. Moving from the lower edge to the upper edge in a domain is moving in the I-direction or the y-direction. Users need to remember this counter-intuitive index juxtaposition.

## 1.2 Vertical Coordinates

The TERRAIN program outputs gridded values of terrain elevation and land use categories which require information about the horizontal domain. Information concerning vertical coordinates is mandatory for every other program (DATAGRID, RAWINS, INTERP, INIT, MM4, TRAJEC and VERIFY). Figure 3a shows the vertical distribution of the data output from DATAGRID to be used as input to either RAWINS or INTERP. These are the mandatory pressure levels up to a user defined value (minimum 30 mb). Accompanying the mandatory level wind, temperature, height, and humidity are values of surface temperature and surface geostrophic winds, sea level pressure, sea surface temperature, and snow cover.

The background fields for RAWINS is the mandatory pressure level data from DATAGRID. Additional data for RAWINS are available from rawinsonde and surface observations. The user may specify additional isobaric levels in the master input file that are to be analyzed and enhanced by the available observations in RAWINS. Figure 3b shows a representative selection of new pressure levels. These values, in conjunction with the mandatory level data, give 50 mb vertical resolution throughout the atmosphere.

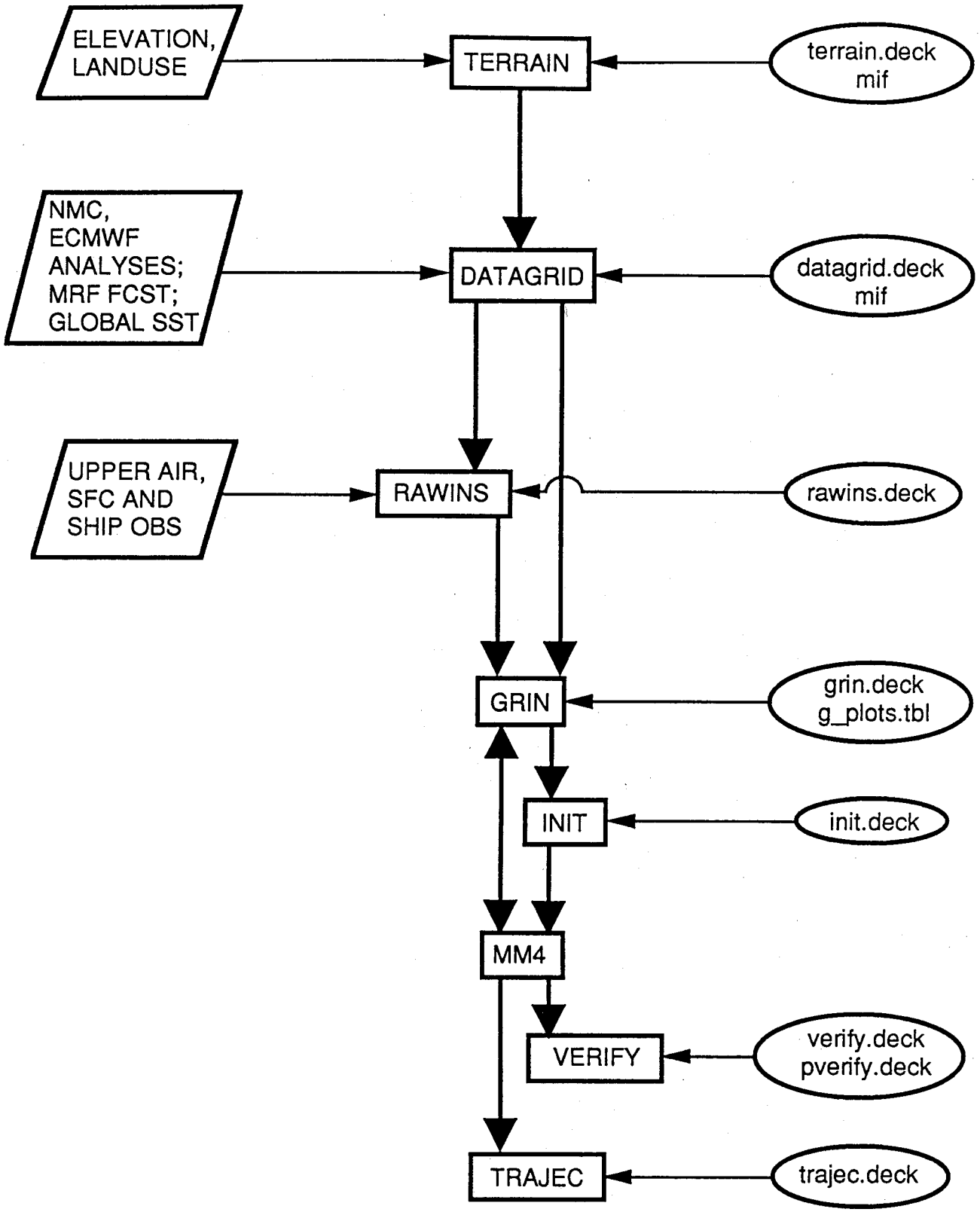



Figure 1. Program flow for the MM4 system: center rectangles denote individual components of the modeling system, bold vertical arrows indicate the flow of generated data, parallelograms show the available NCAR archived data, and the ellipses show the user modified shell scripts and input files that are typically required for each job.

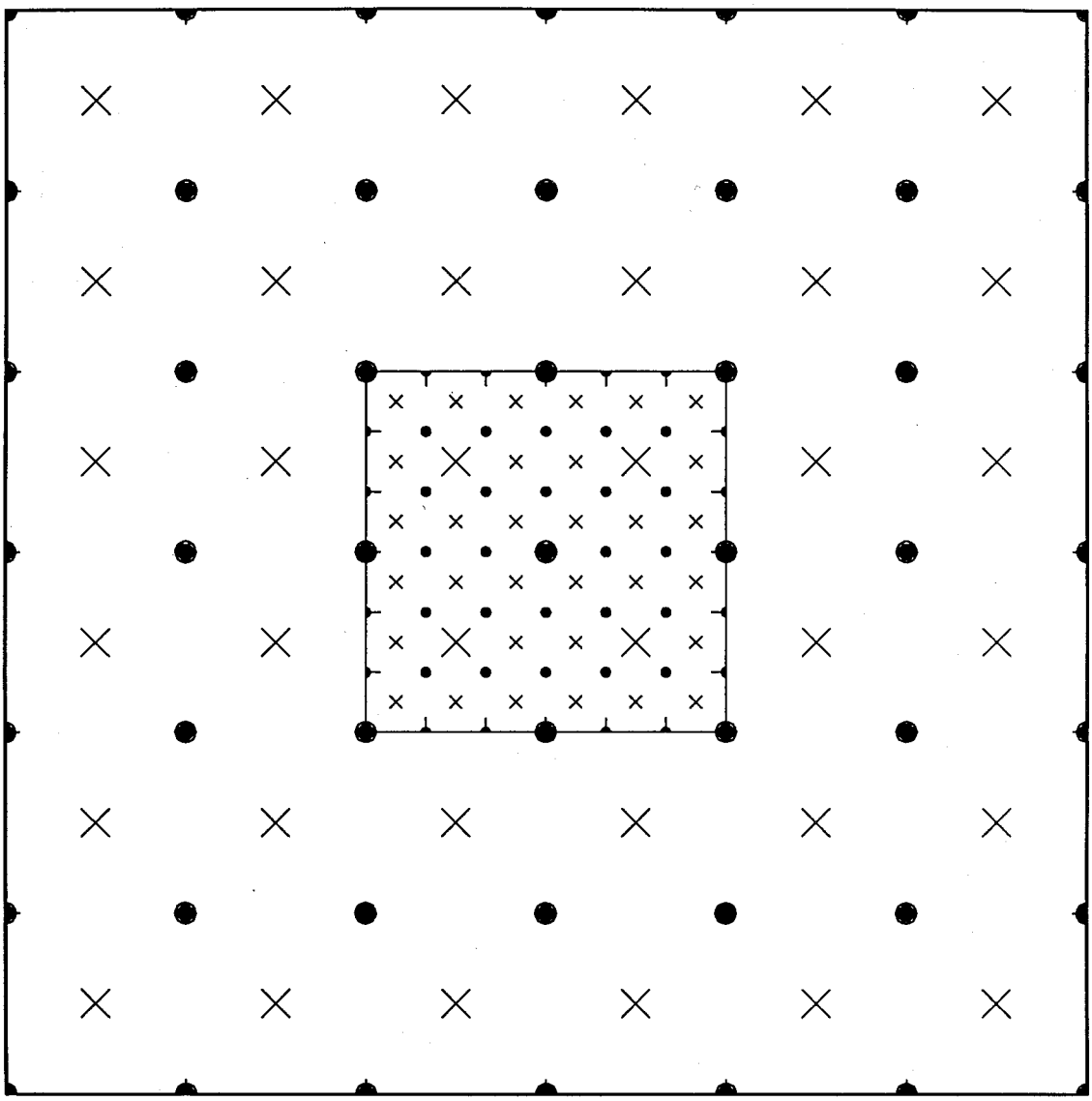


Figure 2. Grid representation showing the horizontal staggering of the dot (•) and cross (×) points. The horizontal components of the winds are defined on the dot points, and the temperature, moisture and pressure fields are defined on the cross points (the Arakawa B–Grid). The smaller inner box shows a representative mesh staggering for a 3::1 coarse-grid distance to fine-grid distance ratio.

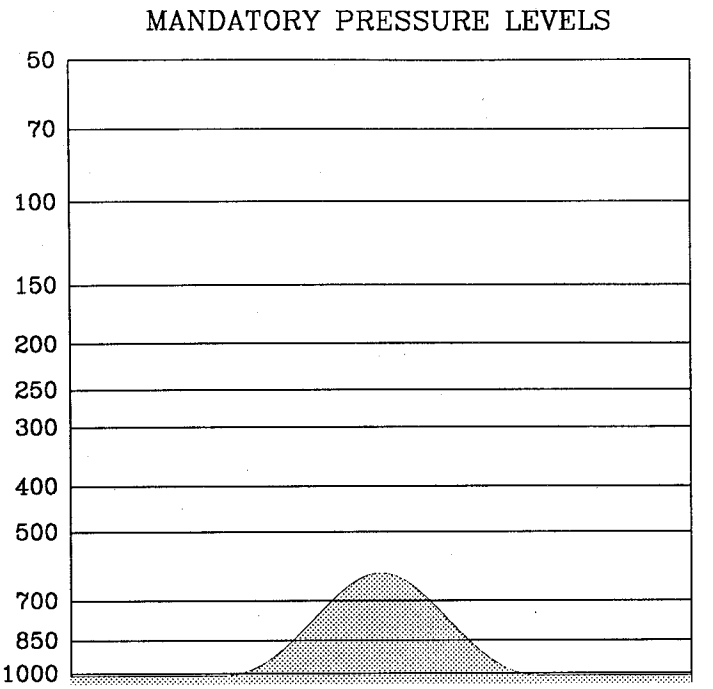



Figure 3a. Vertical cross section detailing the structure of the mandatory pressure levels, typical of data from DATAGRID going into RAWINS. These levels represent all of the mandatory pressure levels up to PTOP in the master input file.

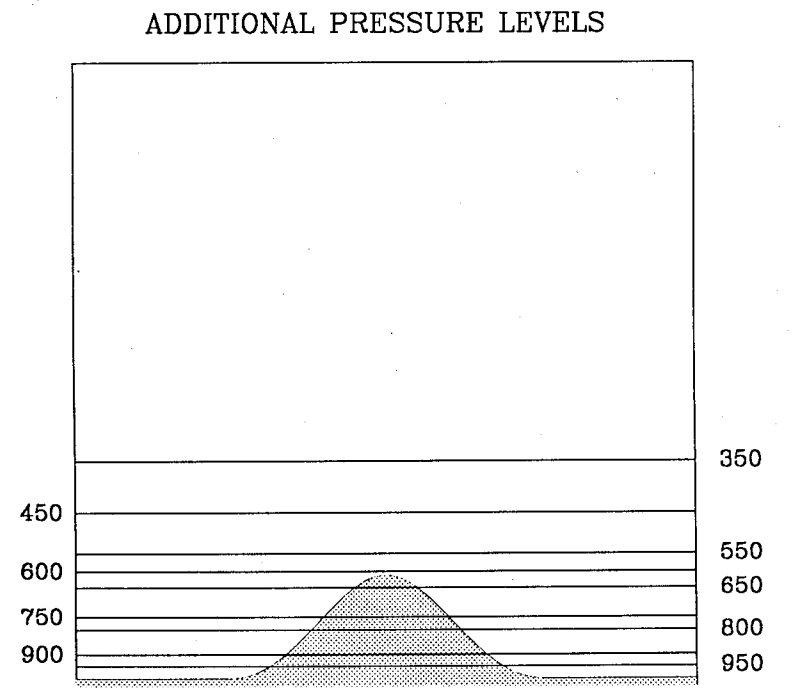



Figure 3b. Vertical cross section detailing the structure of the optional additional pressure levels that were created and analyzed in program RAWINS (these values are the new pressure levels chosen in the master input file).

The model uses $\sigma$, a terrain following normalized pressure coordinate defined as

$$\sigma = \frac{p - p_{top}}{p_{sfc} - p_{top}},$$

where $p_{top}$ is the user defined model lid and $p_{sfc}$ is the surface pressure. From this definition, $\sigma = 1$ corresponds to the level where $p = p_{sfc}$, and $\sigma = 0$ corresponds to the model lid. Figure 4 shows a simple vertical distribution of 11 evenly distributed $\sigma$ levels (solid lines are full levels, dashed lines are half layers). The deformation of the $\sigma$ surface is always more pronounced near high terrain and approaches an isobaric surface at $p_{top}$. (The sample job decks available from the ~mesouser/Decks subdirectories use 24 unevenly spaced sigma levels. This amount of detail would have rendered the figure unintelligible.) The model computations are performed on the half $\sigma$ layers, where the values of $u$, $v$, $T$, and $q_v$ are taken to be representative of the respective variables through the depth of the $\sigma$ layer.

## 1.3 Available Map Projections

The domain of the MM4 modeling system is globally locatable. There are three map projections on which the MM4 system can place data: Lambert conformal, polar stereographic, and Mercator (which should not extend to either pole). The choice for the projection is based upon the area of interest in the domain, specifically, which projection minimizes the distortion and error throughout the gridded domain.

Tropical studies are conducted using the Mercator projection (figure 5a,b), mid-latitude experiments use the Lambert conformal mapping (figure 6a,b), and studies of the polar regions employ the polar stereographic projection (figure 7a,b). A poor choice for domain configuration with respect to projection will not adversely affect any of the programs prior to the forecast model. The required horizontal and vertical interpolations in the analysis programs do not use the map scale information.

The predictive component can become unstable in the areas of largest projection distortion. Since the model will probably abort in this instance, suspect results are not the issue. The user would need to either tighten the model time step appreciably (incurring significantly increased cost), or the entire domain would need to be re-positioned. Moving the forecast domain requires re-running every pre-processor job with the new grid size and domain configuration. As both of these alternatives are time consuming and costly, every effort should be extended to insure the robustness of the initial domain selection.

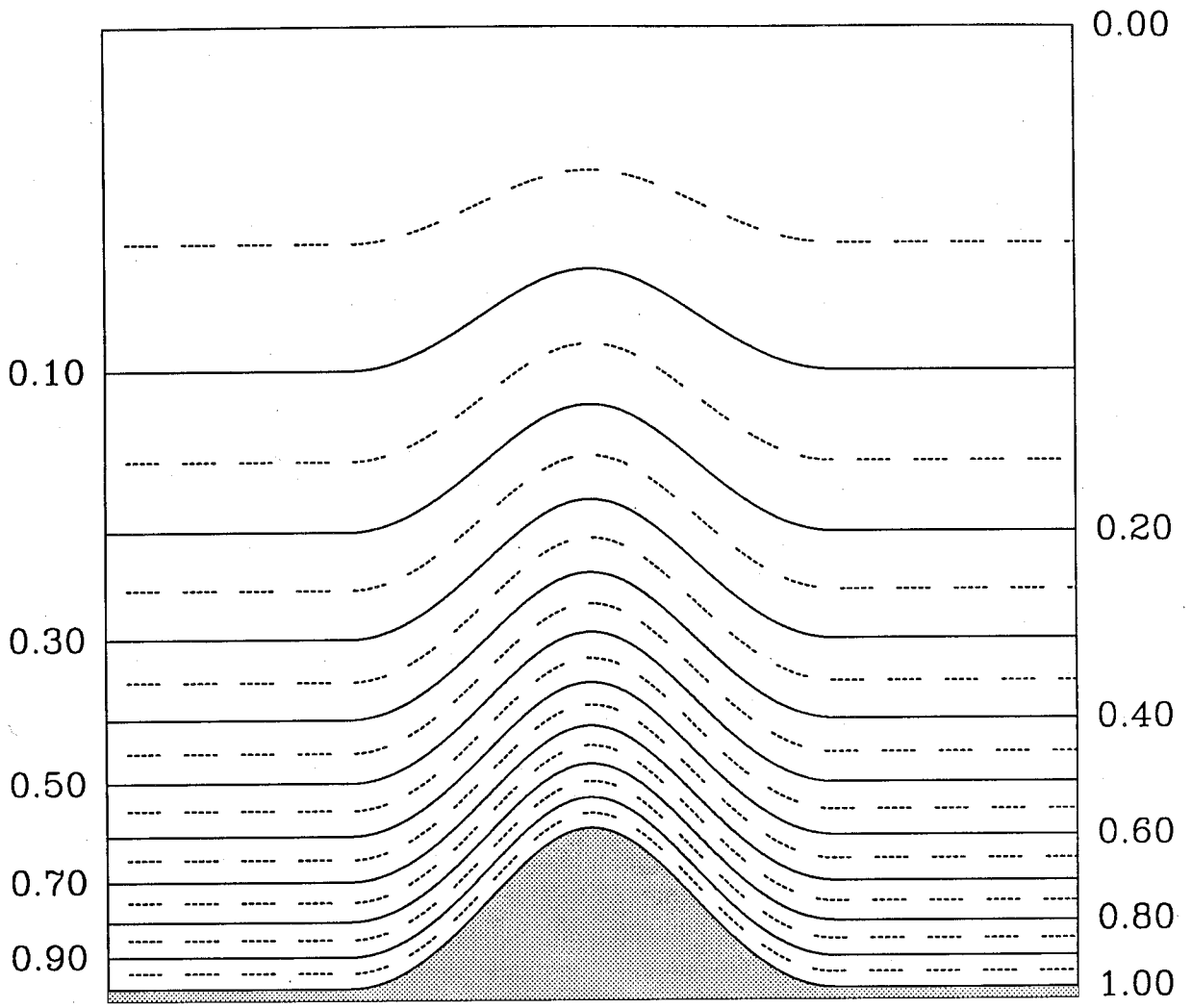



Figure 4. Vertical cross section detailing the structure of the sigma coordinate system of the mesoscale model MM4 with equi-spaced 11 full sigma levels (solid line) and the effective 10 half sigma layers (dashed line). All variables except for sigma dot are defined on the half layers (computational layers).

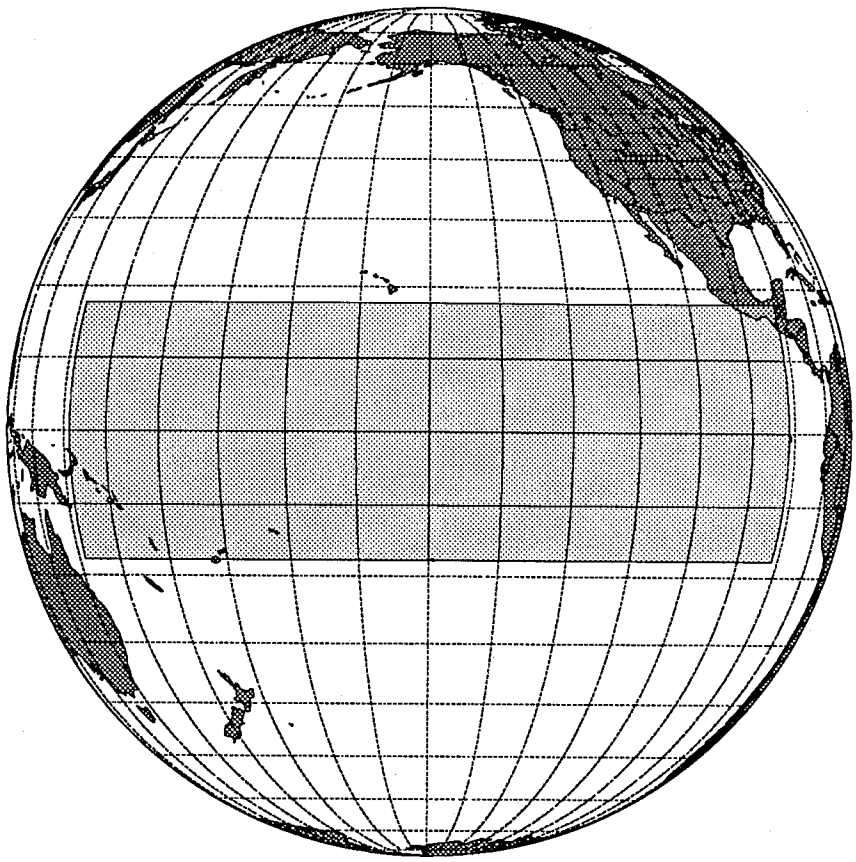


Figure 5a. The lightly shaded region depicts the horizontal extent of a sample domain on the globe, using the Mercator projection.

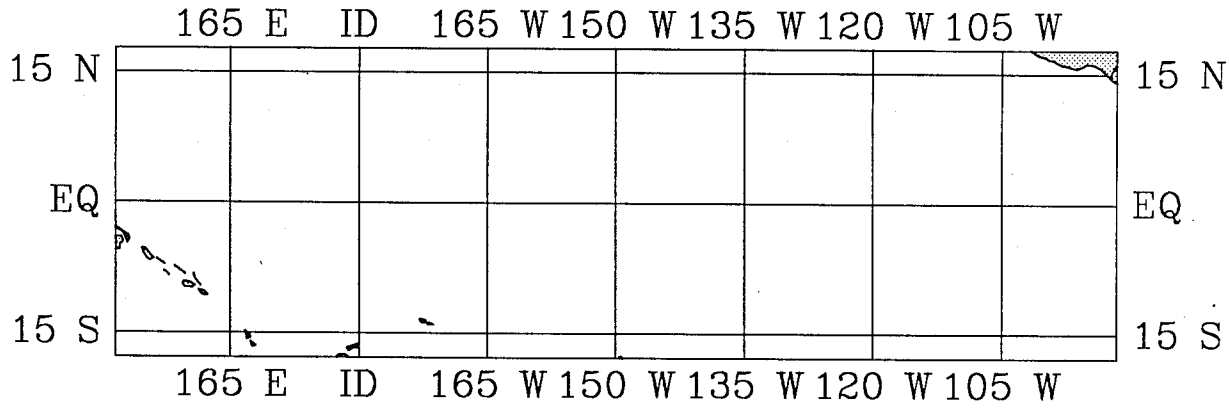



Figure 5b. The more typical presentation of the Mercator projection in figure 5a. This would be the domain used for both analysis and forecast, as well as the map background displayed for horizontal plots.

Figure 6a. The lightly shaded region depicts the horizontal extent of a sample domain on the globe, using the Lambert conformal projection.

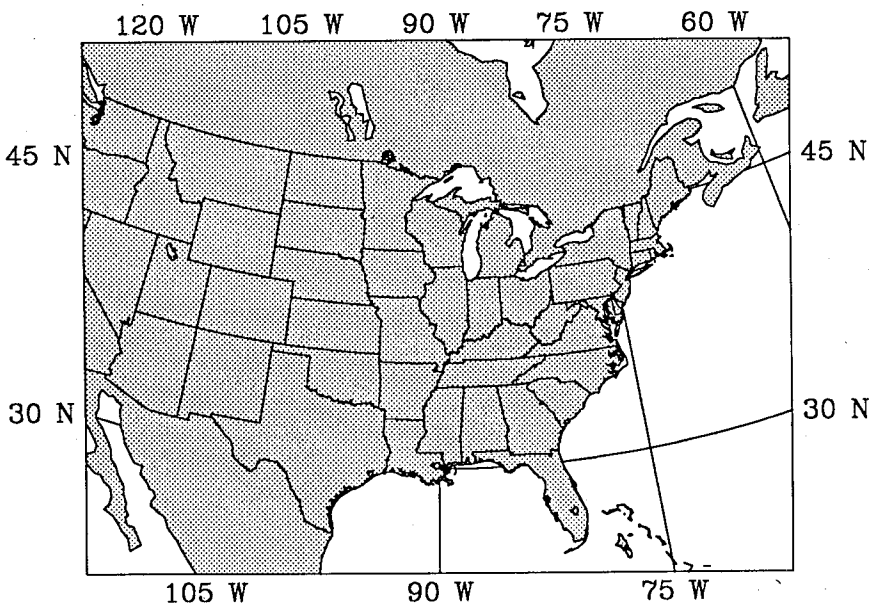



Figure 6b. The more typical presentation of the Lambert conformal projection in figure 6a. This would be the domain used for both analysis and forecast, as well as the map background displayed for horizontal plots.

Figure 7a. The lightly shaded region depicts the horizontal extent of a sample domain on the globe, using the polar stereographic projection.

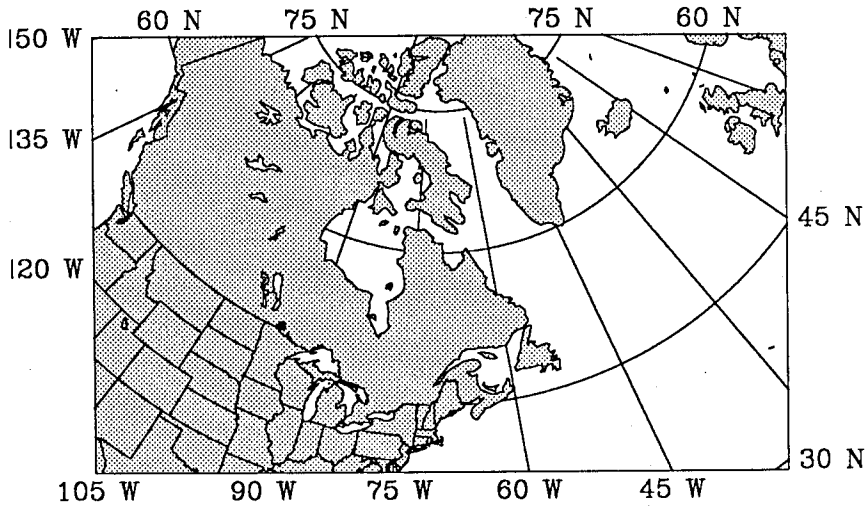



Figure 7b. The more typical presentation of the polar stereographic projection in figure 7a. This would be the domain used for both analysis and forecast, as well as the map background displayed for horizontal plots.

# Chapter 2 Getting Started

This section is for new users of the MM4 modeling system. The assumption is made that the user has SCD accounts on the Cray and the MS, and that the user is familiar with the UNICOS environment and the NCAR facilities. General questions related to the NCAR computer system should be directed to the SCD consulting office at (303) 497–1278 (weekdays from 9:00 AM to 5:00 PM mountain time) or email may be sent to *consult1@NCAR.UCAR.edu* at any time. Questions specific to the MM4 system package should be directed to *mesouser@NCAR.UCAR.edu.*

Section 2.1 of this chapter describes the files that are currently available in the MesoUser supported program directories. A brief overview of the standard naming convention of these files is given. In section 2.2, the archived data that are input into the analysis programs are introduced. The MesoUser manager catalog of available of data sets is discussed. Section 2.3 describes a utility in the modeling system shell scripts that ftp's input files to the shavano disks. The next section deals with a MesoUser supported utility to choose trial grid configurations. The last portion of the chapter, section 2.5, describes how users can access the FORTRAN source code for each of the modeling system components.

## 2.1 MesoUser File Naming Conventions

All of the modeling system components are controlled by job decks. The individual user–level modeling system programs are FORTRAN source codes that are maintained by UPDATE (a Cray source code control system). UPDATE resembles a batch editor, allowing a large group of users access to a common original source deck with a set of maintained bug fixes and enhancements, plus any individual modifications.

The job decks, the modifications files, and various table files for each component of the package are located in separate subdirectories on shavano. Though the list of files maintained by the MesoUser manager initially appears intimidating, generating a standard forecast requires user modification of no more than two files in each section. Following is a list of the currently supported (and therefore fairly standard) modeling system files located in the mesouser account on shavano (an asterix next to the file signifies that it must be modified during a standard run).

- TERRAIN
  - ~mesouser/Decks/Terrain/domain.f
  - ~mesouser/Decks/Terrain/mif *
  - ~mesouser/Decks/Terrain/t_conbw.tbl
  - ~mesouser/Decks/Terrain/t_conbwlu.tbl
  - ~mesouser/Decks/Terrain/t_conco.tbl
  - ~mesouser/Decks/Terrain/t_concolu.tbl
  - ~mesouser/Decks/Terrain/t_mapbw.tbl
  - ~mesouser/Decks/Terrain/t_mapco.tbl
  - ~mesouser/Decks/Terrain/t_new.mods

~mesouser/Decks/Terrain/t_stand.mods
~mesouser/Decks/Terrain/t_true.mods
~mesouser/Decks/Terrain/terrain.deck *

- DATAGRID
  ~mesouser/Decks/Datagrid/d_1time.mods
  ~mesouser/Decks/Datagrid/d_ecmwf.deck
  ~mesouser/Decks/Datagrid/d_ecmwf.f
  ~mesouser/Decks/Datagrid/d_new.mods
  ~mesouser/Decks/Datagrid/d_stand.mods
  ~mesouser/Decks/Datagrid/d_true.mods
  ~mesouser/Decks/Datagrid/datagrid.deck *

- RAWINS
  ~mesouser/Decks/Rawins/r_1time.mods
  ~mesouser/Decks/Rawins/r_conbw.tbl
  ~mesouser/Decks/Rawins/r_conco.tbl
  ~mesouser/Decks/Rawins/r_mapbw.tbl
  ~mesouser/Decks/Rawins/r_mapco.tbl
  ~mesouser/Decks/Rawins/r_nbogv7.mods
  ~mesouser/Decks/Rawins/r_new.mods
  ~mesouser/Decks/Rawins/r_stand.mods
  ~mesouser/Decks/Rawins/r_true.mods
  ~mesouser/Decks/Rawins/rawins.deck *

- GRIN
  ~mesouser/Decks/Grin/g_colorBW.tbl
  ~mesouser/Decks/Grin/g_defaults.nml
  ~mesouser/Decks/Grin/g_mapBW.tbl
  ~mesouser/Decks/Grin/g_new.mods
  ~mesouser/Decks/Grin/g_plots.tbl *
  ~mesouser/Decks/Grin/g_stand.mods
  ~mesouser/Decks/Grin/graph.deck
  ~mesouser/Decks/Grin/grin.deck *
  ~mesouser/Decks/Grin/i_1way.mods
  ~mesouser/Decks/Grin/i_diff_ter.mods
  ~mesouser/Decks/Grin/i_firstg.mods
  ~mesouser/Decks/Grin/i_new.mods
  ~mesouser/Decks/Grin/i_rwdg.mods
  ~mesouser/Decks/Grin/i_stand.mods
  ~mesouser/Decks/Grin/i_v72v8.deck
  ~mesouser/Decks/Grin/record.header

- MM4
  ~mesouser/Decks/MM4/ehtran
  ~mesouser/Decks/MM4/ehtran.f
  ~mesouser/Decks/MM4/m_grell.mods

~mesouser/Decks/MM4/m_new.mods
~mesouser/Decks/MM4/m_stand.mods
~mesouser/Decks/MM4/mm4.deck *

- INIT
  ~mesouser/Decks/Init/init.deck *
  ~mesouser/Decks/Init/init_new.mods
  ~mesouser/Decks/Init/init_stand.mods
  ~mesouser/Decks/Init/m_stand.mods

- TRAJEC
  ~mesouser/Decks/Trajec/ptrajec.deck *
  ~mesouser/Decks/Trajec/tj_color.tbl
  ~mesouser/Decks/Trajec/tj_map.tbl
  ~mesouser/Decks/Trajec/tj_new.mods
  ~mesouser/Decks/Trajec/tjs_new.mods
  ~mesouser/Decks/Trajec/trajec.deck *

- VERIFY
  ~mesouser/Decks/Verify/pverify.deck *
  ~mesouser/Decks/Verify/v_color.tbl
  ~mesouser/Decks/Verify/v_contur.tbl
  ~mesouser/Decks/Verify/v_d.tbl
  ~mesouser/Decks/Verify/v_new.mods
  ~mesouser/Decks/Verify/verify.deck *
  ~mesouser/Decks/Verify/vprep.deck

Most of the files that are in each of the ~mesouser/Decks subdirectories with the extension ".deck" are the C-shell job decks that are executed by the user on shavano (there are some C-shell files that are called internally by the other shells). All of the job decks in the previous list with an asterix next to them are user level C-shells. These programs are set up to run in three modes: interactive, batch submission from shavano with the "qsub" command, or batch submission from a remote site (since the remote job entry procedure is site dependent, the only technique described will be batch submission from the Cray). Interactive use is expensive and limited to a small amount of CPU time. It should be used with discretion, particularly when dealing with the MM4 system.

The modification files have a standard naming convention. Where unique, the first letter of the modification file is the first letter of the program ("t" for TERRAIN, "d" for DATAGRID, "r" for RAWINS, "i" for INTERP, "g" for GRAPH, "m" for MM4, "v" for VERIFY). TRAJEC ("tj") and INIT ("init") are the exceptions. The two modification files that are maintained as supported packages of the modeling system are called "stand" for standard and "new" for the newer modifications. The last five characters in the modification file names are a period, ".", followed by the string "mods". For example, the standard mods for RAWINS would be found in r_stand.mods. There are several other modification files in each of the subdirectories that are either for special use or are in some other way not required for a black box model run. Improvements are introduced to friendly

users by modifying the appropriate *_new.mods file. (Mail is sent to all registered users concerning the recent modification history of the various system UPDATE files.)

Every program deck will copy the standard and new modification files from the mesouser subdirectories. If the user has made some changes to either of these files and would like to have these personal copies of the files incorporated into the program source code (in lieu of the default versions) there are two ways to proceed.

If the user's additions to the default modification file are expected to be a long term requirement, the edited file may be placed into the user's home directory, with the same name as the original. For example, assume the new modification file for MM4 was changed and was required to remain constant during some extended production run. Putting the edited file into the user's home directory on shavano, under the name m_new.mods, would force the MM4 C-shell to use this version of the UPDATE directives, not the default copy in the mesouser directory.

If some testing was still under way concerning changes to the modification files, placing the edited copy in the current working directory would force the mm4.deck to use this version of the modifications, instead of the default copy in the mesouser directory, or the copy in the user's home directory. The precedence for including modification files into the system programs is: 1) check the current working directory, 2) search the home directory, and 3) use the MesoUser file.

Another file to be modified in the previous list, under the TERRAIN section, is the master input file (MIF). This file is used by both the TERRAIN program and the DATAGRID program. The identical file is used by both programs, the user should keep the same MIF file for both the TERRAIN and DATAGRID job submissions. There are no defaults for the switches in the master input file, the user must place a modified copy of the MIF file in the current working directory for both the TERRAIN and DATAGRID runs. Both programs will abort if the master input file is not found. (Section 2.3 discusses how to access the master input file from a front-end machine and appendix B.1 describes the MIF file in detail.)

Most of the table files (filenames that end with a period "." and the extension "tbl") are for internal use by the various programs. Those with the string "color" set up a color table for the subsequent metacode generation. The table files with the string "map" refer to the map information file: style of labels, color filled or outlined, choice of color indices, and other qualities specific to the graphical presentation of the map background. Since no documentation exists describing these files, users are not encouraged to modify the color table files or the map information files.

The table file, listed under the GRIN section, that does need to be modified with every different case is the g_plots.tbl file. This table contains the ordered list of graphical output that the user generates during a grin.deck run. The g_plots.tbl file must reside on the current working directory for the graph.deck shell to access.

## 2.2 Archived Data

There are two suites of available data with which to initialize the forecast model: the historical set which lags several weeks in availability and the UNIDATA set which is in near real-time. The historical latitude-longitude analysis data set provides global first guess fields at all mandatory levels. The user chooses either the NMC or the ECMWF analyses. The UNIDATA archive provides the MRF global 48 hour forecast wwhich supply the background fields to the analysis programs. The historical data sets have global coverage for the upper air and surface stations, while the UNIDATA observation files give only partial coverage.

### 2.2.1 Historical Analyses and Observations

For historical forecast simulations, before the user can initialize the model with data, the NCAR MS archives must be accessed. The typical model run requires between 6 and 10 separate analysis and observation files. Users may choose either the NMC or ECMWF analyses as the background fields (when both are available).

The MesoUser manager has set up several catalogs of the archived analysis and observation files. These volumes are collected from the SCD listings, and compiled to give users current information concerning the availability of the data for the time periods requested. These lists are maintained on shavano and are updated periodically by the MesoUser manager.

From shavano, the catalogs maintained by the MesoUser manager are located in ~mesouser/catalog. There are catalogs for the NMC and ECMWF analyses used by the DATAGRID program, a catalog of the available snow dates for DATAGRID, and catalogs of upper air and surface observations for the RAWINS program.

To serve as an example for retrieving the required MS volume names from the catalogs, a specific 6 day period is chosen. Since this time period was used in the bench-marking runs for the modeling system components, the user can clearly see where the information from the catalogs needs to be incorporated into the job decks. The specific dates were chosen since they require more than one input volume from several of the archived data sets. Below are fragments of these catalogs that include the chosen time periods of the bench-mark forecast, 15 January through 21 January 1988.

## ECMWF Analyses Catalog: ~mesouser/catalog/catalog.ecm

```
-          00Z
K0499K          840419-850226, 00Z ONLY, 13188 BLKS
K0500K          850227-851231, 00Z ONLY, 12934 BLKS
K2177K          860101-860630, 00Z ONLY,  7600 BLKS
K2178K          860701-861231, 00Z ONLY,  7728 BLKS
K7370K          870101-870630, 00Z ONLY,  7602 BLKS
K7371K          870701-871231, 00Z ONLY,  7728 BLKS
Y02397          880101-880630, 00Z ONLY,  7644 BLKS, 77(MB)
Y02398          880701-881231, 00Z ONLY,  7728 BLKS, 78(MB)
-
-          12Z
K0506K          840419-850226, 12Z ONLY, 13188 BLKS
K0507K          850227-851231, 12Z ONLY, 12936 BLKS
K2179K          860101-860630, 12Z ONLY,  7602 BLKS
K2180K          860701-861231, 12Z ONLY,  7728 BLKS
K7372K          870101-870630, 12Z ONLY,  7602 BLKS
K7373K          870701-871231, 12Z ONLY,  7728 BLKS
Y02395          880101-880630, 12Z ONLY,  7644 BLKS, 77(MB)
Y02396          880701-881231, 12Z ONLY,  7728 BLKS, 78(MB)
```

## NMC Analyses Catalog: ~mesouser/catalog/catalog.nmc

```
K4497K          1987JUL16-1987JUL31, 4064 BLKS, NMC LOST: 28JLY0Z
K4498K          1987AUG01-1987AUG15, 3933 BLKS
K4499K          1987AUG16-1987AUG31, 3408 BLKS, NMC LOST: 29-31AUG
K4500K          1987SEP01-1987SEP15, 2884 BLKS, NMC LOST: 1-4SEP
K4501K          1987SEP16-1987SEP30, 3932 BLKS
-
K0720K          1987OCT01-1987OCT15, 3930 BLKS
K0722K          1987OCT16-1987OCT31, 3801 BLKS, NMC LOST: 17OCT, 18OCT0Z
K0787K          1987NOV01-1987NOV15, 4054 BLKS
K0977K          1987NOV16-1987NOV30, 4052 BLKS
K2310K          1987DEC01-1987DEC15, 4010 BLKS, NMC LOST: TEMPS ON 1DEC12Z
K2311K          1987DEC16-1987DEC31, 4323 BLKS
-
K2645K          1988JAN01-1988JAN15, 4051 BLKS
K2646K          1988JAN16-1988JAN31, 4323 BLKS
K6743K          1988FEB01-1988FEB15, 3782 BLKS, NMC LOST: 13FEB0Z, 15FEB12Z
K6744K          1988FEB16-1988FEB29, 3782 BLKS
```

## Surface Observations Catalog: ~mesouser/catalog/catalog.sfc

```
K6532K          1987DEC13-1987DEC19, 3831 BLKS                  LIST A
K6533K          1987DEC20-1987DEC26, 3779 BLKS                  LIST A
K6534K          1987DEC27-1988JAN02, 3741 BLKS                  LIST A
-                                                               LIST A
K6535K          1988JAN03-1988JAN09, 3664 BLKS                  LIST A
K6536K          1988JAN10-1988JAN16, 3676 BLKS                  LIST A
K6382K          1988JAN17-1988JAN23, 3715 BLKS                  LIST A
K6433K          1988JAN24-1988JAN30, 3731 BLKS                  LIST A
K6434K          1988JAN31-1988FEB06, 3690 BLKS                  LIST A


K6547K          1987DEC13-1987DEC19, 3933 BLKS                  LIST B
K6548K          1987DEC20-1987DEC26, 3892 BLKS                  LIST B
K6549K          1987DEC27-1988JAN02, 3896 BLKS                  LIST B
-                                                               LIST B
K6550K          1988JAN03-1988JAN09, 3935 BLKS                  LIST B
K6551K          1988JAN10-1988JAN16, 3922 BLKS                  LIST B
K6735K          1988JAN17-1988JAN23, 3970 BLKS                  LIST B
K6736K          1988JAN24-1988JAN30, 4024 BLKS                  LIST B
K6737K          1988JAN31-1988FEB06, 3920 BLKS                  LIST B
```

## Snow Cover Dates Catalog: ~mesouser/catalog/catalog.snowd

```
1987
 011012  011712  012412  013112  020712  021412  022112  022812  031412  032112
 032812  040412  041112  041812  042512  050212  050912  051612  052312  053012
 060612  061312  062012  062712  070412  071112  071812  072512  080112  080812
 081512  082212  082912  090512  091212  091912  092612  100312  101012  101712
 102412  110712  111412  112112  112812  120512  121212  121912  122612
1988

 010912  011612  012312  013012  020612  021312  022012  022712  030512  102912
 110512  111212  111912  112612  120312  121012  121712  122412
```

## Upper Air Observation Catalog: ~mesouser/catalog/catalog.raob

```
K6500K          1987NOV10-1987NOV27, 5110 BLKS                  LIST A87
K6501K          1987NOV28-1987DEC15, 5090 BLKS                  LIST A87
-                                                               LIST A88
K6503K          1987DEC16-1988JAN02, 5071 BLKS                  LIST A88
-                                                               LIST A88
K8637K          1988JAN03-1988JAN20, 5159 BLKS                  LIST A88
K8641K          1988JAN21-1988FEB07, 5215 BLKS                  LIST A88
K8642K          1988FEB08-1988FEB27, 5271 BLKS                  LIST A88
-                                                               LIST A88
K8644K          1988FEB28-1988MAR16, 5258 BLKS                  LIST A88
K8645K          1988MAR17-1988APR03, 5220 BLKS                  LIST A88
K8647K          1988APR04-1988APR21, 5289 BLKS                  LIST A88
```

Most of the catalogs are quite lengthy and must be searched for the correct dates. In the current example of 15 through 21 January 1988, the MS names of the observation and analysis files can be taken from the catalog fragments, along with the dates on the NMC tapes that have snow cover data. The DATAGRID and RAWINS programs need to be provided with the volume names that contain the data. The programs are capable of processing through these files to find the correct data required for the time periods that are listed in the master input file (for DATAGRID) or that come through the record headers (for RAWINS). In the catalog listing files, the file names are listed on the left hand side; the path on the MS for the files is `/DSS` (Data Support Services, the data management group in SCD). The propensity for the volumes to begin with the letter "K" led to the moniker KTAPE, and it now refers to all of the archived data. The letter "Y" prefixes much of the recent data and the files are now archived on cartridges instead of tapes.

The DATAGRID program uses both the name of the analysis data set (either NMC or ECMWF) and the time period in the NMC file that has the snow cover data. In this example, the ECMWF analysis requires two files: `/DSS/Y02397` for the 00Z time periods and `/DSS/Y02395` for the 12Z time periods (the included dates are `880101` through `880630`, 1 January 1988 through 30 June 1988). The ECMWF analysis has fewer missing time periods than the NMC analyses, but the ECMWF data inventory is not as extensive. The NMC analysis data covering the 15 to 21 January time period is contained in the files `/DSS/K2645K` and `/DSS/K2646K`. The NMC data is not split between 00Z and 12Z as is the ECMWF archive, but two volumes are required to completely cover the analysis periods.

Information about missing NMC time periods and information concerning the number of time periods to pull from the first of a multi-volume NMC analysis are available in the catalog, and must be given to the DATAGRID program. In this example, there are no missing times in the NMC data, and the number of user requested time periods in the first volume of the NMC data set is two (00Z 15 January and 12Z 15 January). A choice is made to run the DATAGRID program with either the NMC data or the ECMWF data, and only those data volumes from that particular gridded analysis need to be inserted into the datagrid.deck C-shell.

If the user is interested in having the predictive model use the available snow cover data during the forecast, snow data must be put into the DATAGRID program. Regardless of whether the user chose NMC or ECMWF analyses to supply the first guess fields, NMC data supplies the snow data. The snow cover availability date prior to the test time period is 12Z 9 January (`010912` under the year 1988), and the snow date during the selected time period is 12Z 16 January (`011612`). The DATAGRID local input file requires information for snow data both prior to the forecast and during the forecast.

Other than the DATAGRID job, the only other program that requires data from the catalogs is RAWINS. The synoptic upper air rawinsonde files required for this example are `/DSS/K8637K` and `/DSS/K8641K`. The catalog of surface data is not as easy to interpret as the other listings, as there are several distinct products contained inside the catalog. At the top of the surface catalog is a brief statement explaining that `LIST A` refers to the 6 hourly data (06Z, 18Z) and `LIST B` refers to the 3 hourly observations (03Z, 09Z, 15Z, 21Z). With this information, the 6 hourly files (data available at 6 hour intervals)

covering the test periods are /DSS/K6536K and /DSS/K6382K, and the 3 hourly files (data available at 3 hour intervals) are /DSS/K6551K and /DSS/K6735K. The rawins.deck C-shell must have all six of these volume names as input to cover the example dates with upper air and surface observations.

The text of the fragment listings was lifted directly from the catalogs on shavano. First time users of the modeling system should log on to shavano and take a look at the catalogs in ~mesouser/catalog. Users need to be comfortable and accurate with the lists, as there can be no forecast until the analyses and observations are correctly ingested. The catalogs are ASCII text and can be ftp'ed to the user's front-end machine. The user need only be concerned about updating the personal catalog list of the observations and analyses files when a required date does not exist.

## 2.3 Automatic FTP Set Up

To allow the standard version of the modeling system to access files that are generated and maintained on the user's front-end machine, a file called ftp.remote needs to be in the user's home directory. This is essentially the information that ftp (file transfer protocol, with TCP/IP) uses for remote access. For example, if user Raboof (with password *iluvshav* on a unit front-end machine) has a computer called raindrop (with complete internet address of raindrop.NCAR.UCAR.edu), and if Raboof stores all of the modeling system input files in the directory ~raboof/ympfiles on raindrop, the ftp.remote file on shavano would appear as:

```
open raindrop.NCAR.UCAR.edu   front-end machine
user raboof iluvshav          user name and password on front-end machine
cd ympfiles                   storage directory on front-end machine
```

Access privileges to this file need to be restricted. Issue the following command in the home directory from shavano:

```
chmod 600 ftp.remote
```

This file is optional; the MM4 system does not require its use. The ftp.remote file simply supplies another place that the C-shell may look to find some requested information (the master input file for TERRAIN and DATAGRID, and various user supplied subjective meteorological datasets for RAWINS). A user more familiar with UNIX can modify the ftp constructs to use rcp in the shell scripts. This avoids the security problem of a published password.

## 2.4 Choosing the Domain

The TERRAIN program together with the master input file generates several frames of metacode of the map background, terrain elevation, and land use categories. When the

user is anticipating a nested model run, the TERRAIN program also creates similar plots for the fine grid. For new users, it is not entirely intuitive how to properly choose grid sizes, domain centers, and projections.

The MesoUser manager supports a simple FORTRAN program to help choose the domain. The file ~mesouser/Decks/Terrain/domain.f is a FORTRAN source code, not a C-shell script. This program can be run interactively on shavano or on the user's local machine (NCAR GKS is required). The user responds to the program's queries, and the generated output is one frame of GKS metacode.

The domain mapping program may be run from shavano, and the generated metacode either ftp'ed back to the user's front-end machine for display, or the metacode translator may be run directly from shavano. To use the domain.f program, copy the program to a temporary directory, compile and load it, then run the program by responding to the prompts. Assuming the user has a vt100 graphics terminal for display purposes, the following shows the few short commands required to run the modeling system domain program and view the output.

```
cd $TMPDIR
cp ~mesouser/Decks/Terrain/domain.f .
ncargf77 domain.f
a.out
ictrans -d vt100 gmeta
```

On the next several pages are the interactive sessions for three domains, and the metacode that was generated. The domains were chosen to show the ease with which the choices required in the TERRAIN program and in the master input could be settled. The program prompts begin with the word **enter**, and the user response is underneath each query.

If the user sets up the master input file (MIF) for the TERRAIN and DATAGRID programs with the identical horizontal grid information as is chosen with this program, the domains will be coincident. Note that all of the coarse-grid domains are centered on the specified latitude-longitude values. This is not a requirement for the fine grid. Users should exercise care in selecting the forecast domain. A poorly chosen domain will suffice until the model integration becomes unstable.

The first example is the sample domain used in the standard decks supported by the MesoUser manager (figure 8). This is a Lambert conformal projection, centered at 40 N latitude, and 90 W longitude. The grid size is 90 km, and the location of a fine grid is specified (in coarse-grid units).

```
enter imax,jmax (i refers to y, j refers to x direction)
46,61
enter grid distance in km
90
enter center latitude, center longitude
40,-90
enter the lower left corner i,j
18,25
enter the upper right corner i,j
28,35
enter type of projection:  (1)lambert (2)polar (3) mercator
1
```

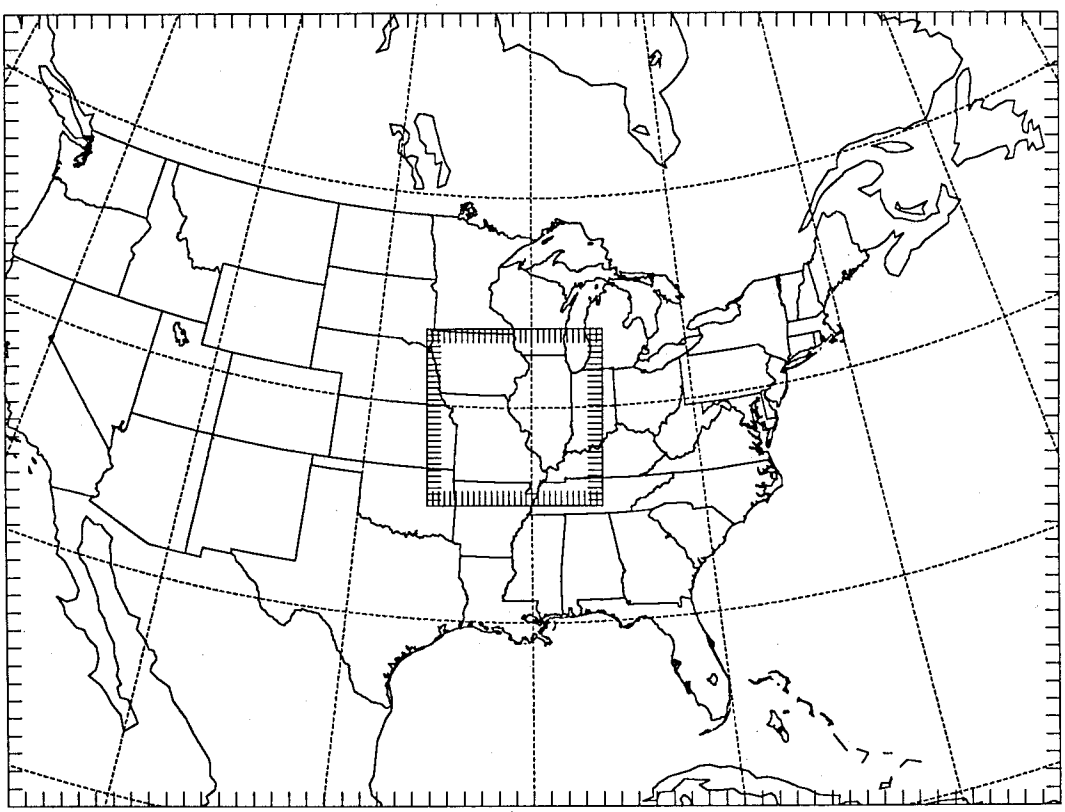


Figure 8. Lambert conformal projection domain generated with ~mesouser/Decks/Terrain/domain.f, using the specifications set forth in the text (note the inner nested region).

The second example is a domain that MM4 could use to generate a forecast for the polar region of the northern hemisphere (figure 9). This is a polar stereographic projection, centered at 90 N latitude, and 90 W longitude. The grid size is 100 km, and no fine grid is specified.

```
enter imax,jmax (i refers to y, j refers to x direction)
101,101
enter grid distance in km
100
enter center latitude, center longitude
90,-90
enter the lower left corner i,j
1,1
enter the upper right corner i,j
101,101
enter type of projection:  (1)lambert (2)polar (3) mercator
2
```

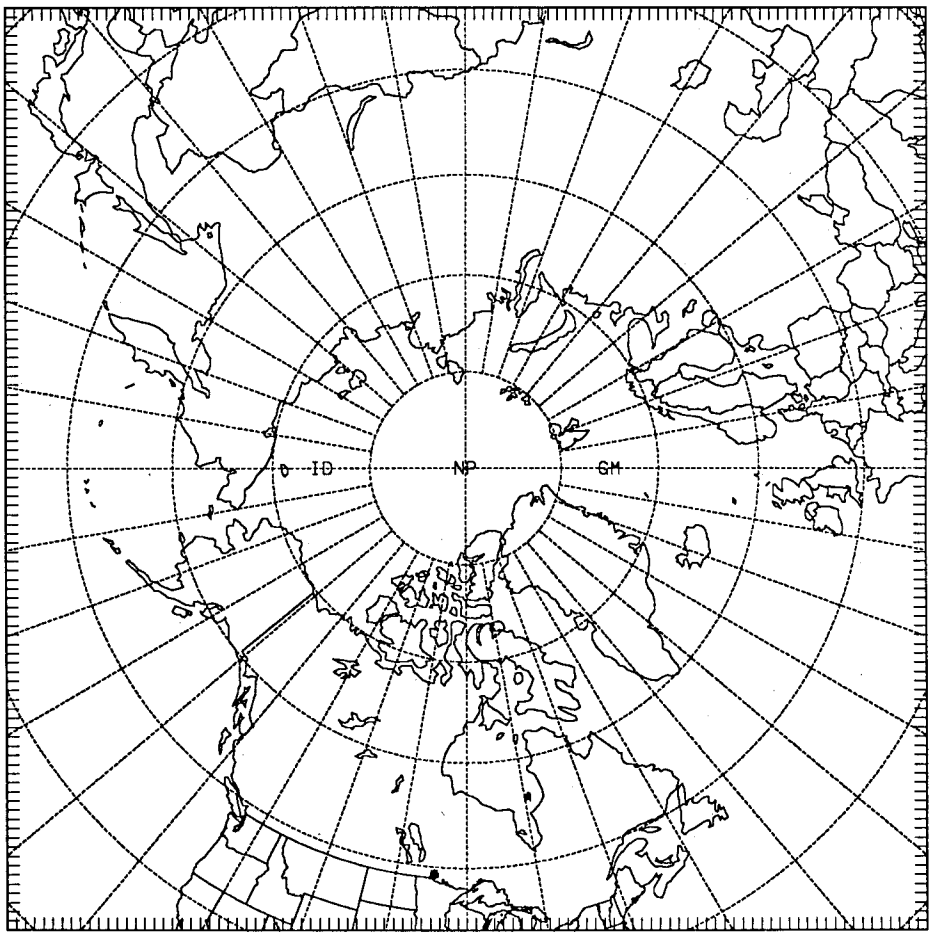



Figure 9. Polar stereographic projection domain generated with ~mesouser/Decks/Terrain/domain.f, using the specifications set forth in the text.

The last example is a domain around the entire globe between 65 N and 65 S (figure 10). This is a Mercator projection, centered at 0 N latitude and 180 W longitude (the equator and the date line). The grid size is approximately 3° at the equator. The fine grid is specified to depict a possible El Niño domain.

```
enter imax,jmax (i refers to y, j refers to x direction)
61,121
enter grid distance in km
333.3
enter center latitude, center longitude
0,180
enter the lower left corner i,j
24,41
enter the upper right corner i,j
38,98
enter type of projection:  (1)lambert (2)polar (3) mercator
3
```

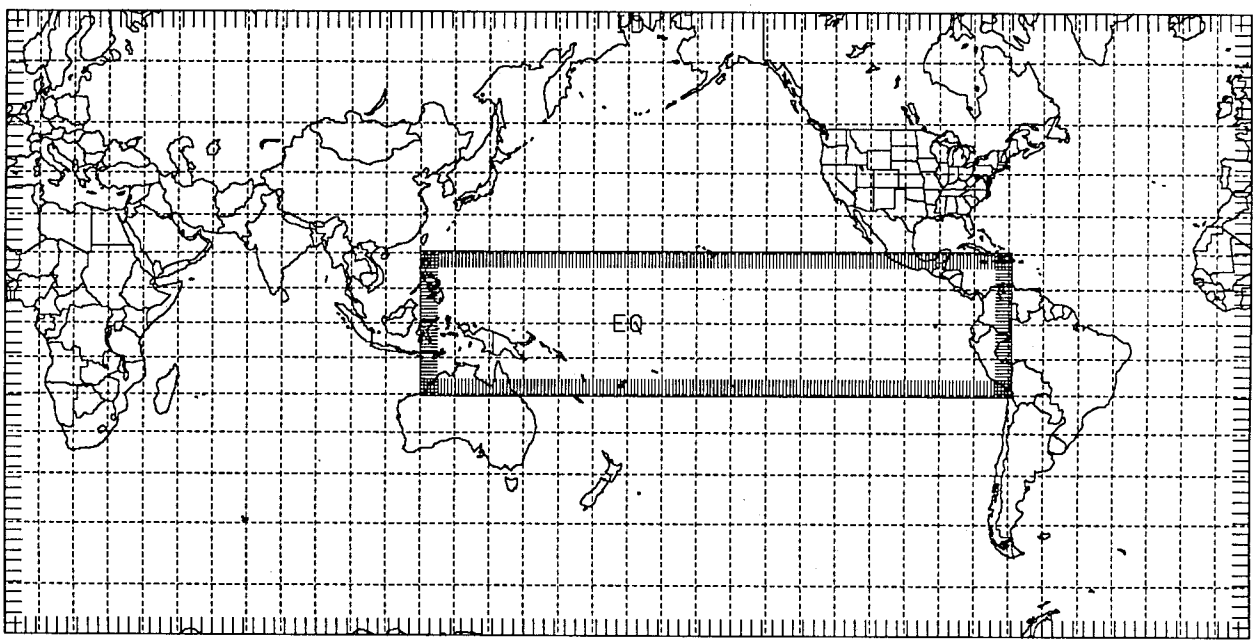



Figure 10. Mercator projection domain generated with ~mesouser/Decks/Terrain/domain.f, using the specifications set forth in the text (note the inner nested region).

## 2.5 Modeling System Source Code

The MesoUser manager maintains archives of the COMPILE-able source code on the MS at NCAR. When modifications warrant, these files are updated to reflect the newer bug fixes. To help users access these files, a C-shell script was written to pull these files from the MS and optionally spool them to a laser printer at NCAR.

Consistent with the other MesoUser supported decks, users may process the job in an interactive mode, submit the file to a batch queue from shavano, or remotely send the job to shavano via MIGS or IRJE. The computational time involved is less than 1 CPU second. Assuming the user will send a batch job from shavano, the user must modify the internal switches on the Cray, then submit the deck with qsub. Following is a sequence of commands to request source code.

```
cd $TMPDIR
cp ~mesouser/Util/gsource.deck .
vi gsource.deck
qsub gsource.deck
```

The internal shell variables need to be set so that the user may choose which of the 10 available source files are requested. The user also sets the flag to determine whether the files are placed onto a shavano disk only, or additionally sent to a printing device. Below are the shell variable assignments inside the gsource.deck file to select source code for INTERP and MM4, and to have both of these files sent to the printer.

```
#
   set TERRAINsw  = no
   set DATAGRIDsw = no
   set RAWINSsw   = no
   set INTERPsw   = yes
   set GRAPHsw    = no
   set INITsw     = no
   set MM4sw      = yes
   set MMZIGGYsw  = no
   set TRAJECsw   = no
   set VERIFYsw   = no
#
#  to obtain hard copies of source code from NCAR SCD high speed laser
#  printer, turn print switch = yes.    For users located outside Boulder,
#  you will receive the output from NCAR SCD through mail.
#
   set PRINTsw = yes
#
```

# Chapter 3 Sample Jobs

The job decks that are maintained by the MesoUser manager have been set up to process data for the time period 00Z 15 January through 00Z 21 January 1988. This time period was chosen more because it required several overlapping archived data sets than for any interesting atmospheric phenomena. These programs were run to bench mark the modeling system components during the COS to UNICOS conversion. Currently, the sample jobs are the supported modeling system decks for users of the MM4 package. Users should take advantage of the information contained inside the bench-mark C-shells.

- The MS output file names in the individual job decks serve as a direct link to where specific data was created, and subsequently these names indicate where the generated data is later used as input.
- The MS path names of the SCD archived data used in the job decks, along with the example catalog fragments contained in this document (section 2.2.1), provide a verification process. This cross-reference capability aids users in the interpretation and placement of the file names from the various catalogs.
- The MS files created during the bench-mark runs are available to users. These files represent an additional resource to consider when including personal modifications to the default UPDATE files. Users may test personal codes that process the modeling system data by verifying that the user programs accept these MS volumes, or that the standard programs are able to ingest the user generated data.
- For new users, one of the most confusing aspects of the modeling package is the grid-size parameterizations used in the pre-processors, model and post-processors. Concrete examples of the rules discussed in several sections of this document detail the use of the coarse, fine, and expanded grids, along with the logical flags that permit nested and expanded domains.

## 3.1 What to Modify in the C-Shell Job Decks

The modeling system job decks are maintained as C-shell scripts. (In this document, the phrases C-shell scripts, C-shell job decks, shells, and decks all refer to the MesoUser maintained system packages.) The user may define shell variables that are expanded for use as file names, directories, or titles for graphics. Users with syntax questions for C-shell constructs are referred to the man pages for "csh".

The modeling system package of C-shell job decks are structured similar to each other. The procedure to modify any one of them is nearly transportable to the rest of the job decks. This section describes the generic structure of the MesoUser C-shell decks, and which specific parts must be modified by the user. The examples will be drawn from the TERRAIN, DATAGRID, RAWINS, GRIN, INIT, MM4, TRAJEC, and VERIFY sample job decks located in the ~mesouser/Decks subdirectories. These programs represent the MesoUser modeling system package.

### 3.1.1 QSUB

Each job deck has the several lines of header commands for the network queueing system (NQS) on shavano. Inclusion of these commands allows the decks to be submitted into the batch queue locally from shavano with the "qsub" command, or remotely from the user's front-end machine through IRJE or MIGS. The `QSUB` commands inside the C-shell files are preceded by the "#" character; when these jobs are run interactively, the `QSUB` commands are interpreted as comments.

```
# QSUB -r  TERRAIN          # request name
# QSUB -q  reg              # job queue class
# QSUB -eo                  # stdout and stderr together
# QSUB -lM 3Mw              # maximum memory
# QSUB -lT 1000             # maximum time in seconds
# QSUB                      # no more QSUB commands
```

The only mandatory command is the definition of the class in which the user would like to have the job run. These are currently `econ` (economy), `reg` (regular), or `prem` (premium). There are special queues that have resources allocated for specific projects, queues for multi-tasked jobs, queues with no GAU accrual (and only stand-by priority), and queues for select large models. The rates at which the different classes consume general accounting units (GAUs) change periodically.

If the program begins execution and subsequently fails with either of the two following error messages,

```
not enough space
Cpu limit exceeded
```

the user should resubmit the C-shell file after augmenting the memory allocation to increase space (`# QSUB -lM`) or the time limit to allow more CPU time (`# QSUB -lT`), respectively.

Other than specifying the queue and the occasional limitation error, there is no need for the user to spend much time with these settings. Further information can be found in the UNICOS manual pages for "qsub". The different job classes have various restrictions related to maximum time, maximum available memory, number of available slots in the job mix, and groups of users permitted in the different queues. Of more interest to users is the fluctuating cost of the different job classes with respect to each other compared to the relative turn around times. Questions related to any of these matters should be referred to the NCAR SCD consultants.

### 3.1.2 Experiment Name: ExpName

After the `QSUB` heading commands, the next user modification is the shell-variable assignment for `ExpName`. The front-end jobs TERRAIN, DATAGRID and RAWINS use the `ExpName` information to define possible names of files that are required to be local, but

might be located on remote machines. The TERRAIN, RAWINS, and GRIN C-shells put the ExpName string into the title of the fiche and film that are generated. Every job deck uses the shell variable ExpName to create the subdirectory under /usr/tmp/$LOGNAME which the C-shell uses for the current working directory (users can override this option with $TMPDIR, as discussed further in this section). Following is the C-shell fragment included in the job decks that creates the directory and moves the process to that location.

```
#       this should be the user's case or experiment
#
set ExpName = TEST
#
#       have the temporary disk as default
#
if ( ! -d /usr/tmp/$LOGNAME ) mkdir /usr/tmp/$LOGNAME
if (   -d /usr/tmp/$LOGNAME/$ExpName ) then
   chdir  /usr/tmp/$LOGNAME/$ExpName
else
   mkdir  /usr/tmp/$LOGNAME/$ExpName
   chdir  /usr/tmp/$LOGNAME/$ExpName
endif
```

With these commands, the job deck executes from a user-defined directory on shavano. All of the files accessed and generated by the C-shell will reside in this directory. Unfortunately, this system is not failsafe; data tends to age off before it is successfully written to the MS by the application. Programs that generate large amounts of metacode also lose files before they have been queued to the plotting device. The alternative is the $TMPDIR directory. This is a pre-defined directory that UNICOS created when the job execution began. No files are scrubbed from $TMPDIR until all of the initiated processes have completed, then all files are removed.

The GRIN program submits all metacode requests to the Text and Graphics System (TAGS) processor from the relative safety of the predefined $TMPDIR directory. The incidental metacode from the TERRAIN program (under 20 frames), and the incidental metacode from the RAWINS program (soundings + auto bogus plots) are small enough so that the temporary disk scrubber is not a serious threat to the metafiles before they are safely queued to the plotting device. The difficulty is with the predictive model. If the user chooses the ExpName directory option, model simulations requiring more than a day to complete are liable to be corrupted (or gone completely) due to scrubbing.

The user can insert the following command after the job deck directory instructions to force the shell to move the execution to $TMPDIR:

```
cd $TMPDIR
```

The side effects associated with the $TMPDIR choice need to be understood. After the process in $TMPDIR completes, all files are removed and the directory is deleted. Users who can absolutely insure that all created files are sent to the MS upon program

termination (either a successful completion or possibly an ungraceful exit), find the $TMPDIR choice preferable, particularly for extended model runs.

When the machine is full, the scrubber is activated often, as frequently as every few minutes. Jobs requiring much less time than the model may have files removed if the current working directory was not defined as $TMPDIR. Users choosing to run programs from $TMPDIR should not submit the jobs directly from shavano, but through MIGS or IRJE. The NQS on shavano attempts to return the standard out and standard error files to the originating machine. These files are typically too large to fit in the user's permanent space on shavano, and shavano's temporary disks are too volatile for even short storage. MIGS and IRJE will safely handle the standard output after the program completes executing.

### 3.1.3 Job Decks Switches

Actions inside the job decks that require input from the user are made with respect to the choices of the deck switches that the user has implemented. For example, these decisions allow: 1) TERRAIN to get the correct resolution of gridded elevation and land use categories from the archived data; 2) DATAGRID to know whether to expect NMC, ECMWF or MRF latitude – longitude input; 3) RAWINS to anticipate either inputting or outputting the auto bogus file; 4) GRIN, MM4 and TRAJEC to be aware of the existence of a nested domain; 5) INIT to choose the number of model iterations; and 6) VERIFY to prepare for a precipitation verification. The job decks have all of the switch settings available to the user, with the inactive lines commented. Following are the switch settings for each of the modeling system components and a brief explanation of the options.

3.1.3.1 Description of TERRAIN C-shell Switches

| | |
|---|---|
| `MAPBGsw` | generate only the map background, or process the terrain elevation and land use information; choices are `MapOnly` (for deactivating data processing) and `TER` (for activating the terrain processing) |
| `NESTsw` | write created nested terrain elevation and land use file to MS; choices are `NoNEST` (for deactivating the nest shell directives) and `NEST` (for activating the nest shell directives) |
| `TLUResC` | resolution of the input terrain elevation and land use gridded data for the coarse domain; choices include `60` (1° resolution), `30` (30-minute resolution), `10` (10-minute resolution) |
| `TLUResN` | resolution of the input terrain elevation and land use gridded data for the nested domain; choices include `60` (1° resolution), `30` (30-minute resolution), `10` (10-minute resolution) |

```
#
#       type of terrain job
#
#  set MAPBGsw = MapOnly
   set MAPBGsw =  TER
#
#  set NESTsw = NoNEST
   set NESTsw =   NEST
#
if ( $MAPBGsw == MapOnly ) then
   set NTypeC = 1
   set NTypeF = 1
   set GetTer = F
else
#       type of lat/lon terrain grid to use
#                       TLURes  = 60     FOR 1 DEGREE DATA
#                       TLURes  = 30     FOR 30 MIN DATA
#                       TLURes  = 10     FOR 10 MIN DATA
#
   set GetTer  = T
#  set TLUResC = 60
   set TLUResC = 30
#  set TLUResC = 10
#
#  set TLUResN = 60
#  set TLUResN = 30
   set TLUResN = 10
endif
```

3.1.3.2 Description of DATAGRID C-shell Switches

`Analysis` the source for the first-guess data; choices are `NMC` for the National Meteorological Center's global 2.5° × 2.5° analyses, `ECMWF` for the European Centre for Medium Range Forecasting's global 2.5° × 2.5° analyses, `UNIDATA` for the Medium Range Forecast 2.5° × 5.0° analyses and 48 hour forecast

`SST` the source for the sea surface temperature analyses; choices are `NMC` for the National Meteorological Center's global 2.5° × 2.5° analyses, `Navy` for the U.S. Navy's northern-hemispheric 64 × 64 octagonal analyses, `Clim` for the monthly mean of the National Meteorological Center's global analyses available at 2.0° × 2.0°resolution

`unidate` when choosing UNIDATA input, the 8 digit MDATE (YYMMDDHH) of the initial time of the 48 hour MRF forecast to use as input to DATAGRID

`sstdate` when choosing UNIDATA input to DATAGRID, the sea surface temperature MDATE is necessary (this is typically different than the time of the MRF initial time)

```
#
#       type of datagrid job
#
   set Analysis  = NMC
#  set Analysis  = ECMWF
#  set Analysis  = UNIDATA
#
   set SST       = NMC
#  set SST       = Navy
#  set SST       = Clim
#
if ( $Analysis == UNIDATA ) then
   set unidate   = 92071400
   set sstdate   = 92071300
endif
```

3.1.3.3 Description of RAWINS C-shell Switches

| | |
|---|---|
| `Submit` | which of the possible RAWINS runs is this; choices are `0` for the first of an expected single submission, `1` for the first of an expected dual submission to generate an auto bogus file, `2` for all subsequent runs of an expected multi-run RAWINS submission to incorporate the auto bogus information into the analysis |
| `INOBS` | the shell must know if the user will acquire the UNIDATA observations from one time period only (`UNIOBS`); or the traditional archived observations, available for all of the analysis time periods in RAWINS (`ARCHIVE`) |
| `FDDAsw` | let the shell know if this is an FDDA run; choices are `NoFDDA` and `FDDA` |
| `SFCsw` | are the surface observations (available at 3 hour intervals and 6 hour intervals) to be incorporated into the analysis; choices are `NoSFC` and `SFC` |
| `BOGUSsw` | allow the shell to search for some additional bogus files if none are in the current working directory; choices are `NoBOG` for no bogus files required in the program, `Konly` when only the KBOGUS file is required, `Nonly` when only the NBOGUS file is required, `KandN` when both the KBOGUS and NBOGUS files need to be accessed |
| `MapCol` | the user may choose the style of auto bogus metacode; choices are `BW` for black and white fiche, `CO` for color film |
| `unidate` | when the user chooses the UNIDATA observations, the 8 digit MDATE (YYMMDDHH) of the observation is required |

```
#
#       type of rawins job
#
   set Submit  = 0
#  set Submit  = 1
#  set Submit  = 2
#
#  set INOBS   = UNIOBS
   set INOBS   = ARCHIVE
#
#  set FDDAsw  = NoFDDA
   set FDDAsw  = FDDA
#
```

```
#  set SFCsw    = NoSFC
   set SFCsw    = SFC
#
   set BOGUSsw = NoBOG
#  set BOGUSsw = Konly
#  set BOGUSsw = Nonly
#  set BOGUSsw = KandN
#
#  set MapCol   = BW
   set MapCol   = CO
#
if ( $INOBS == UNIOBS ) then
   set unidate = 92071400
endif
```

3.1.3.4 Description of GRIN C-shell Switches

| | |
|---|---|
| `ForBsw` | tell GRIN if this is a front-end (DATAGRID or RAWINS into model input) or a back-end (forecast data) job; choices are `FRONT` (have the shell dispose most of the generated data), or `BACK` (plot metacode, no archiving) |
| `FDDAsw` | for the front-end jobs from RAWINS, when a nested domain is requested, GRIN tries to generate a fine-grid surface FDDA file for the mesoscale model; choices are `NoFDDA` or `FDDA` |
| `NESTsw` | for the front-end jobs, does the shell need to archive and plot a fine domain; choices are `NoNEST` or `NEST` |
| `VERSION` | for the back-end jobs, is this Version 8 data, or the older Version 7 model output format that needs to be converted; choices are 8 or 7 |
| `Split` | how many time periods to process for INTERP; this is also the number of metafiles into which the generated plots from GRAPH are split |
| `GraphVersion` | which version of the GRAPH executable the user chooses to acquire; the current version is 5, the GRAPH version that supports the non-hydrostatic model output is 6 (note that this is an environment variable, implying that it exports the variable definition) |

```
#
#       type of grin job
#
   set ForBsw           = FRONT
#  set ForBsw           = BACK
#
#  set FDDAsw           = NoFDDA
   set FDDAsw           =   FDDA
#
#  set NESTsw           = NoNEST
   set NESTsw           =   NEST
#
   set VERSION          = 8
#  set VERSION          = 7
#
   set Split            = 13
#
   setenv GraphVersion = 5
```

3.1.3.5 Description of MM4 C-shell Switches

`STARTsw` the shell needs to know if this is a standard program start or if this is a restart (for file naming conventions); choices are `Begin` for a forecast starting at the 0 hour, `ReStart` for a restart

`INITsw` has the initial conditions file gone through the vertical mode interpolation program INIT; choices are `INIT` and `NoINIT`

`FDDAsw` does the shell need to access the FDDA files on the MS; choices are `NoFDDA`, or `FDDA`

`NESTsw` if this is a nested run, the shell must grab the fine-grid initial conditions from the MS, possibly the fine-grid FDDA surface and analyses; choices are `NoNEST` and `NEST`

`DIAGSsw` if there are diagnostic files created, they may be archived to the MS; choices are `NoDIAGS` and `DIAGS`

`GRELLsw` access the additional cumulus parameterization modification file ~ mesouser/Decks/MM4/m_grell.mods (the IMOIST flag in the namelist must be set to explicit moisture, and the moist parameter dimensions must be set up); choices are `NoGRELL` and `GRELL`

```
#
#       type of mm4 job
#
   set STARTsw = Begin
#  set STARTsw = ReStart
#
   set INITsw  = NoINIT
#  set INITsw  =   INIT
#
   set FDDAsw  = NoFDDA
#  set FDDAsw  =   FDDA
#
   set NESTsw  = NoNEST
#  set NESTsw  =   NEST
#
   set DIAGsw  = NoDIAGS
#  set DIAGsw  =   DIAGS
#
   set GRELLsw = NoGRELL
#  set GRELLsw =   GRELL
```

3.1.3.6 Description of TRAJEC C-shell Switches

| | |
|---|---|
| `VERSION` | which version of the model output to expect, choices are 7 and 8; version 7 data requires the user to provide an additional namelist for the missing header information |
| `PTRAJCsw` | the TRAJEC code requires that the model data be split into separate files only once, interactive use allows users to bypass this step; choices are `PTRA` to run the pre-trajectory code, `NOPTRA` to bypass the pre-trajectory code |
| `NESTsw` | the user may choose to have only one domain acquired (either coarse or fine), or both domains acquired (coarse and fine); choices are `NoNEST` to process only a single domain, or `NEST` to process both domains |
| `SIGsw` | trajectories can be computed on sigma surfaces or on isentropic surfaces; choices are `NOSIG` and `SIG` |

```
#
#       type of trajec job
#
   set VERSION  = 8
#  set VERSION  = 7
#
   set PTRAJCsw =   PTRA
#  set PTRAJCsw = NOPTRA
#
   set NESTsw   = NONEST
#  set NESTsw   =   NEST
#
   set SIGsw    = NOSIG
#  set SIGsw    =   SIG
```

3.1.3.7 Description of INIT C-shell Switches

`Niter` the number of iterations of the loop consisting of the VMODES program followed by a single time step of the model; recommended value is 3

```
#
#       type of init job
#
   set Niter = 3
#
```

3.1.3.8 Description of VERIFY C-shell Switches

`PCPsw` precipitation verification requires additional file handling; choices are `NOPCP` for no precipitation verification, and `PCP` to allow precipitation verification

```
#
#       type of verify job
#
   set PCPsw = NOPCP
#  set PCPsw =   PCP
#
```

### 3.1.4 MS File Names

The MS file names follow the deck switches in the modeling system standard C-shells. The job is expected either to initially pull these MS volumes onto the current working directory, or upon completion, to archive local files to the MS.

One of the conventions used to denote input files from output files is the name given to the shell variable holding the MS path locations. Shell variables starting with the string "In" are files coming in from the MS; they are input to the program. Shell variables starting with the string "Out" are files that are to be archived to the MS; they are output from the program. The same MS volume can be assigned as output from one of the C-shell job decks and then assigned as input in some other subsequent program (for example, the expanded domain terrain and land use file is output from the TERRAIN program and used as input for DATAGRID).

Another convention used is whether the file refers to the outer grid (either the expanded domain or the coarse domain) or the fine grid. Shell variables ending with a capital "C" hold the names of outer domain files (expanded or coarse), while the shell variables ending with the capital letters "F" or "N" refer to the fine (nested) grid. Following are each of the shell variables that are used to store the MS file names. The program in which the variables occur, the C-shell variable name, whether this is an input or output file, and a brief description of each file is included.

| | | | |
|---|---|---|---|
| TERRAIN | `OutTerC` | output | expanded domain terrain elevation and land use |
| TERRAIN | `OutTerN` | output | fine domain terrain elevation and land use |
| | | | |
| DATAGRID | `InTerr` | input | expanded domain terrain elevation and land use |
| DATAGRID | `InAnly` | input | global NMC or NGM first-guess analysis |
| DATAGRID | `InSnow` | input | additional NMC analysis containing snow data |
| DATAGRID | `InAnly00` | input | 00Z ECMWF first-guess analysis |
| DATAGRID | `InAnly12` | input | 12Z ECMWF first-guess analysis |
| DATAGRID | `InSST` | input | either the Navy or NMC climatology SST |
| DATAGRID | `OutDatg` | output | expanded domain DATAGRID analysis |
| | | | |
| RAWINS | `InRaobs` | input | global upper air rawinsonde |
| RAWINS | `InDatg` | input | expanded domain DATAGRID analysis |
| RAWINS | `InSfc3h` | input | global 3 hourly surface observations |
| RAWINS | `InSfc6h` | input | global 6 hourly surface observations |
| RAWINS | `OutRawAnl` | output | coarse domain RAWINS enhanced analysis |
| RAWINS | `OutRaobs` | output | rawinsonde observations inside domain |
| RAWINS | `OutSobs` | output | quality checked surface observations for FDDA |
| RAWINS | `OutUobs` | output | quality checked upper air observations for FDDA |
| RAWINS | `OutVPPlt` | output | Stüve plot of rawinsonde observations in domain |
| RAWINS | `Out4dSfc` | output | coarse domain surface analysis for FDDA |
| RAWINS | `OutABPlt` | output | auto bogus plots |
| RAWINS | `OutAB` | both | auto bogus file |

| | | | |
|---|---|---|---|
| GRIN | InData | input | DATAGRID, RAWINS or model data to process |
| GRIN | InTer | input | fine domain terrain elevation and land use |
| GRIN | InFDDA | input | coarse domain surface analysis for FDDA |
| GRIN | OutInit | output | coarse domain model initial conditions |
| GRIN | OutBdy | output | coarse domain model boundary conditions |
| | | | |
| INIT | InBdy | input | coarse domain model boundary condition |
| INIT | InMMC | input | coarse domain model initial conditions |
| INIT | OutInit | input | initialized coarse domain model initial conditions |
| | | | |
| MM4 | InBdy | input | coarse domain model boundary conditions |
| MM4 | InMMC | input | coarse domain model initial conditions |
| MM4 | InRstC | input | coarse domain restart |
| MM4 | InMMF | input | fine domain model initial conditions |
| MM4 | InRstF | input | fine domain restart |
| MM4 | In4DSfcC | input | coarse domain surface analysis for FDDA |
| MM4 | In4DSfcF | input | fine domain surface analysis for FDDA |
| MM4 | OutMMC | output | coarse domain model forecast |
| MM4 | OutRstC | output | coarse domain save file |
| MM4 | OutMMF | output | fine domain model forecast |
| MM4 | OutRstF | output | fine domain save file |
| | | | |
| PVERIFY | mmin | input | $\sigma$-level model initial conditions |
| PVERIFY | mmout | input | $\sigma$-level model forecast output |
| PVERIFY | OutVprep1 | output | pressure-level model initial conditions |
| PVERIFY | OutVprep2 | output | pressure-level model forecast |
| | | | |
| VERIFY | VFcst | input | pressure-level model forecast |
| VERIFY | VObs | input | pressure-level model initial conditions |
| VERIFY | Vprecip | input | observed precipitation data (various forms) |
| VERIFY | VerOut1 | output | non-standard analysis diagnostics |
| VERIFY | VerOut2 | output | non-standard forecast diagnostics |
| VERIFY | Verplt | output | diagnostic plots of verification statistics |
| | | | |
| TRAJEC | InDataC | input | single domain forecast data |
| TRAJEC | InDataN | input | second of two domains of forecast data |
| TRAJEC | OutData | output | diagnostic trajectory data |

### 3.1.5 Parameterized Dimensions

Following the shell variable assignments for both the switches and the MS file names are the UPDATE modifications to the parameterized dimensions. These are direct modifications to the source code, implying that strict FORTRAN syntax must be observed.

Examples of each of the parameterized dimensions are given in the sections covering the programs individually. The TERRAIN, DATAGRID, and RAWINS codes all use similar constructs to allocate space for the internal variables. The TERRAIN and RAWINS job decks require both a gridded domain parameterization as well as a parameterization for the maximum number of expected observations.

Neither the GRIN nor the MM4 decks use the parameterization UPDATE deck associated with the pre-processors. Both the INTERP and MM4 codes allow domains of different sizes to be processed concurrently (the coarse and fine grids). This requires that the user specify information for each of the separate domains and also allocate space to contain the largest of the separate grids.

The INIT and VERIFY codes process only a single domain per run. Users specify the coarse-grid or fine-grid domain dimensions for the horizontal space allocation, depending on the input domain. The TRAJEC code allows the user to input either the coarse-grid or fine-grid domain individually, or both domains simultaneously. Only TRAJEC jobs with two concurrent input domains require the `IMXN` and `JMXN` parameter values. All of the job deck C-shells have FORTRAN parameter statements that are occasionally not used, and must be defined as 1.

### 3.1.6 Local Input Files

The local input files for each of the programs are contained inside the C-shell job decks, and are physically located after the UPDATE parameter statements. The variables in the local input files are described in some detail in the sections of this document specific to those individual programs.

The local input files are FORTRAN namelist structured files. When modifying the namelist files, users need to exercise care with respect to the Cray extensions to this non-standard syntax. The first column must be a blank character, a comma must follow all assignments, character strings may not be used, and comments are preceded with a semicolon.

After the user is comfortable with the implications of the various flags in the respective local input files, the comments in the namelist records make the options relatively easy to interpret and modify. As with the switch settings of the C-shell variables, the available options that the user may choose are commented out. Together with the mnemonic help of the variable name, the availability of comments about the other optional choices, and a brief description of the particular flag, enough information is usually presented so that the requested selection is understood.

### 3.1.7 No More User Modifications Required

After the user has 1) modified the QSUB commands to set the proper class for the job, 2) changed the `ExpName` flag to be consistent with the rest of the experiment, 3) correctly set the shell variables for the switches and MS file names, 4) modified the parameter statements, and 5) set up the local input file, there are no more user modifications required

in the decks.

Knowing what has to be modified is as important as knowing when to quit. The portion of the job decks that the user must modify is the initial 20% of the C-shell. Physically after the local input file, the individual decks digest the information from the user and set about the business of processing the job. This subsequent section of each of the decks handles the input of data, generates the executable from the FORTRAN source code, and disposes the data and plots to the appropriate devices. Users need only to modify this processing section to adjust the default settings for the data management, include additional options in the executable generation, or redirect the metafile destination.

## 3.2 Running the Sample Jobs

As was stated before, the sample jobs generate data to allow the model to produce a forecast over the lower continental U.S. during the period 00Z 15 January through 00Z 21 January 1988 (see figure 8 for the domain defined for the bench-mark runs). The master input file specifies which times to generate during the analyses. Users may modify personal copies of the standard decks maintained by the MesoUser manager to run for only two synoptic times, allowing up to a 12-hour forecast from the model.

This section is for users who would like to spend 10 CPU minutes to generate all of the pre-processor data through the model input. It will be assumed that the user will run the jobs directly from shavano in the batch queues with the "qsub" command. The forecast length can be significantly reduced from the available 12 hours of boundary conditions.

After the user has logged onto shavano, two directories must be created. The user should issue all commands from the created `TEST` directory.

```
mkdir /usr/tmp/$LOGNAME
mkdir /usr/tmp/$LOGNAME/TEST
cd /usr/tmp/$LOGNAME/TEST
```

The first step in any forecast is to run the TERRAIN program. The TERRAIN job deck and the master input file must be in the current working directory. The C-shell must have the MS file names modified; the user can simply search for the string `/GILL` to find each of the volume names. The master input file (MIF) must be modified to process only two times periods. The `IFILES = 13` line can be changed to `IFILES = 2`. The TERRAIN job can be submitted to shavano after both files have been modified.

```
cp ~mesouser/Decks/Terrain/terrain.deck .
cp ~mesouser/Decks/Terrain/mif .
vi terrain.deck
vi mif
qsub terrain.deck
```

For the programs that generate graphical output, the compile time is quite large. During 12-hour experiment (two time periods), compiling will dominate most of the CPU

time of the pre-processors. The TERRAIN program executes in 27 seconds. The user can search for the 99999 stop in the output listing, which will be called `TERRAIN.o*` and will be located in the `TEST` directory. An output volume from both the expanded domain and a fine-grid domain are generated. Each of these files contain the gridded values of terrain elevation and the land use categories. These data files are archived to the MS, and some incidental metacode is sent to TAGS.

After the TERRAIN program runs, the next program in the series is DATAGRID. This program uses the master input file, but since the variables are identical to those defined during the TERRAIN run, the same copy of the MIF should be used. The user needs to make a local copy of the DATAGRID C-shell. The MS file names should be modified to reflect the choice made during the TERRAIN program run, by changing all occurrences of the string `/GILL`.

```
cp ~mesouser/Decks/Datagrid/datagrid.deck .
vi datagrid.deck
qsub datagrid.deck
```

The DATAGRID job deck acquires the NMC analysis for the requested time periods and interpolates the first-guess data onto the model expanded domain. The expanded domain terrain and land use file is required as input. The generated DATAGRID output file is coarse resolution with mandatory level data. The DATAGRID output file is archived to the MS for later input to RAWINS; there is no metacode file created. Users can certify successful completion of the DATAGRID job by paging through the created listing file, called `DATGRID.o*`. This job deck requires 15 seconds to run, and the 9999 stop from GETDAT will be displayed upon successful completion.

The DATAGRID data set is enhanced by objectively incorporating the upper air and surface observations into the first-guess fields. The master input file used in the TERRAIN and DATAGRID programs is not needed for any of the subsequent programs. The RAWINS C-shell must be accessed from the mesouser directories. Similar to the other job decks, the MS file names should be changed to the convention previously chosen by the user.

```
cp ~mesouser/Decks/Rawins/rawins.deck .
vi rawins.deck
qsub rawins.deck
```

The RAWINS job objectively analyzes the two synoptic time periods from DATAGRID, but also has the local input file flags activated to generate the objectively analyzed surface FDDA file for the model. This requires the surface analysis to run for the additional 03, 06, and 09Z times (the sample bench-mark runs have been modified to only run for 12 hours, 00 - 12Z). Since the submittal switch is not set for the auto bogus capability, the only metacode to be generated is for the upper air soundings. The expanded domain is cut down to the coarse grid when the gridded data is output. The successful RAWINS job for this test case will require 185 CPU seconds and will issue the 99 stop from RAWINS on completion, which is verified from the returned output listing `RAWINS.o*`. Among the

several files written to the MS for this RAWINS run are the FDDA surface file and the output analysis file which will both be used as input to other programs.

After the isobaric and surface analyses are complete, the data needs to be put into the model input format. The GRIN program is run after RAWINS (or after DATAGRID if the user elected not to enhance the analysis with observations). Both the GRIN deck and the table file of plots, g_plots.tbl, are required from the mesouser account. For this test experiment, the number of time periods to process must be changed in the GRIN C-shell from `Split = 13` to `Split = 2`, as well as the MS file naming convention modified to be consistent with the user's previous volumes. The table file can be left unmodified, but it must be in the current working directory (the graphics request table, g_plots.tbl, is more fully discussed in Appendix B.6).

```
cp ~mesouser/Decks/Grin/grin.deck .
cp ~mesouser/Decks/Grin/g_plots.tbl .
vi grin.deck
qsub grin.deck
```

The bench-mark case is set up to generate a nested MM4 forecast (the sample MM4 job is set up as a single domain to avoid accidental expensive runs) requiring that the GRIN deck generate a coarse-grid and a fine-grid model initial condition. INTERP uses the coarse-grid data to create the required model boundary conditions. Since the FDDA code was used in RAWINS, a fine-grid file for FDDA surface analysis nudging is produced for the model. All of these files are ingested by the GRAPH program to generate plots that define the model's initial state of the atmosphere. The interpolation program requires from 1 to 10 seconds to generate a time period of model input, but the GRAPH program typically takes 1 CPU second per generated frame of metacode. Both the INTERP and GRAPH programs issue the `99999` stop upon successful completion of the code. From the output listing `GRIN.o*`, the user can page through the file and count the six separate calls to the GRAPH code.

After the initial conditions and boundary conditions have been created and archived to the MS, the mesoscale model can be run. Since the sample jobs are simply example decks and not sensitivity experiments, the MM4 deck has been set up to be relatively cheap to run. The user needs to copy the mesouser version of the model to the local directory. The MS file names must be consistent with the previous job decks. To keep the MM4 run inexpensive, the maximum forecast time `TIMAX = 1440` should be lowered. This value is the forecast time in minutes, so a one hour forecast would be `TIMAX = 60`. If the user intends to run VERIFY on the forecast data, a 12-hour forecast should be generated, `TIMAX = 720`. Remember to modify the MS filenames to avoid corrupting the archived bench-mark datasets.

```
cp ~mesouser/Decks/MM4/mm4.deck .
vi mm4.deck
qsub mm4.deck
```

This MM4 deck is set up to generate a forecast on the coarse mesh with Anthes–Kuo

cumulus parameterization, the Blackadar PBL, and Davies–Turner relaxation boundary conditions. The output listing from MM4 in `MM4.o*` consists of forecast stability information at each time step of the integration plus some user requested horizontal and vertical sections at specified time intervals. The predictive model successfully completes with the 99999 stop from MM4. Any data generated by the model is sent to the MS.

During this exercise, the user has generated files to allow a two-way interacting coarse-grid / fine-grid forecast over the domain. Because of data sets generated during the RAWINS and GRIN decks, the user may select coarse and/or fine-grid analysis nudging for FDDA in the model. The model forecast may be lengthened to a maximum of `TIMAX = 720` minutes, as two synoptic periods were chosen in the original master input file read by DATAGRID.

# Chapter 4 TERRAIN

The program that begins any complete forecast simulation is TERRAIN. This program horizontally interpolates (or analyzes) gridded latitude - longitude terrain elevation and land use categories onto the chosen expanded domain. The final data is defined for both the expanded grid (which is the same size as the coarse grid if RAWINS will not use the expanded analysis option) and the fine grid (if this is to set up data for a nested-model run).

The TERRAIN scripts, modification files and color tables (for the graphical output) are located in the ~mesouser/Decks/Terrain subdirectory on shavano. To run the TERRAIN program, the master input file (MIF) and the TERRAIN C-shell executable, (terrain.deck) need to be modified. The color tables, map information tables, and contouring information tables are all available for the user to modify, but this is not recommended. Appendix A.1 gives a specific description of the format and content of the TERRAIN output file, and lists the available land use categories.

## 4.1 Local Input File

The master input file contains information concerning: the physical extent of the domain; expanded-grid, coarse-grid and fine-grid particulars; and the projection (see Appendix B.1 for more discussion of the master input file). Information pertaining only to the TERRAIN program is contained in the local input file (also a namelist structured file) included with the TERRAIN job deck. Several of the modeling system job decks, including TERRAIN, have namelist values defined through shell variable expansion. This provides an additional assurance of consistency between the C-shell script and the FORTRAN code. Following is a sample TERRAIN local input file:

```
&LOCMIF ;-------------LOCAL MIF FOR PROGRAM TERRAIN-----------
NTYPEC = $NTypeC,          ; COARSE  ||  1=1 DEG, 4=30 MIN, 10=10 MIN,
NTYPEF = $NTypeF,          ; FINE    ||  5=5 MIN INPUT RESOLUTION
IPRNTD = F,                ; T/F: PRINT ALL INCOMING DATA
IPRNT  = F,                ; T/F: PRINT INCOMING DATA WITHIN MAP DOMAIN
IHSTGM = F,                ; T/F: PRINT HISTOGRAM OF DATA DENSITY
IBNDRY = T,                ; T/F: SMOOTH BOUNDARY HEIGHT GRADIENT
MAP    = F,                ; T/F: PRINT GRID HEIGHT DATA
IPFILE = T,                ; T/F: WRITE AN OUTPUT FILE OF TERRAIN
NAME   = 5HTR80K,6HTERNES, ; FILE NAMES FROM TERRAIN(COARSE-NEST)
NTAPE  = 7HNCAR TP,        ; INPUT SOURCE
NFILES = 0,                ; IF SOURCE IS NCAR TAPE SET TO ZERO
RIN    = 1.5,              ; RAD OF INFLUENCE-COARSE MESH
RINF   = 2.3,              ; RAD OF INFLUENCE-FINE MESH
ISURF  = T,                ; T/F: LAND USE DATA
IFANAL = T,                ; TRUE FOR ANALYSIS,  FALSE FOR INTERPOLATION
IFTER  = $GetTer,          ; TRUE FOR TERRAIN; FALSE FOR MAP ONLY
& ;-------------------------------------------------------------
```

There are several switches in the TERRAIN local input file that are typically modified. To adjust the resolution of the input global terrain and land use data, the NTYPEC (coarse-domain resolution) and NTYPEF (fine-grid resolution) may be set to request 1°, 30', 10' or 5' data. By default, these values are effectively set by the user when selecting the shell variables controlling the data resolution. There are simple tests in the TERRAIN code to verify the requested data resolution is consistent with the domain grid spacing. The 5' data set, which requires the user to set ISURF = FALSE in the TERRAIN local input file, is available only over the western hemisphere and has no land use information. Values set by the a shell variable (those with the "$" prefix should not need to be redefined by the user.

The user must choose whether the input values of the terrain and land use are interpolated or analyzed to the model grid points. The interpolation technique will overshoot the input values in areas of steep gradients, and the analysis method will tend toward a smoother terrain and land use field. The interpolation routine used throughout the modeling system, employs pairs of overlapping quadratic fits, incorporating up to 16 nearest neighbor input values (Bleck and Haagenson, 1966). The analysis method is a single-scan Cressman objective analysis, where the user may modify the radius of influence RIN and RINF in the local input file.

## 4.2 Parameterized Dimensions

Most of the modeling system programs use parameterized values to dimension the 2–D and 3–D variables in the FORTRAN code. Since the TERRAIN program generates 2–D terrain and land use fields, only the horizontal dimensions are important: the coarse-grid size (IMXC and JMXC), the fine-grid size (IMXN and JMXN), and the expanded domain horizontal size (IMX and JMX). If this is not a nested run, the nested dimensions should be set to 1 (IMXN=1 and JMXN=1). If an expanded domain is not to be used in the later analysis, the coarse and expanded grids are the same size (IMXC=IMX and JMXC=JMX). Following is an example of the changes in the parameter listing for the terrain.deck file:

```
*D PARAM.4,11
C  IMX,JMX are the expanded grid dimensions if IEXP=TRUE in the master
C          input file
C          IMX = IMXC + 2 * INT(AEXP/DS + 1.), same for JMX
C          If not using the expanded grid, IMX and JMX are the same as
C          the coarse-mesh dimensions.
C  IMXN,JMXN are the nested-grid dimensions
C  IMXC,JMXC are the coarse-mesh dimensions (non-expanded, same as
C          IMAX and JMAX in the master input file)
C  If not using the expanded grid, IMX=IMXC=IMAX, JMX=JMXC=JMAX
C
      PARAMETER(IMX=58,JMX=73)
      PARAMETER(IMXN=31,JMXN=31,IMXC=46,JMXC=61)
```

The TERRAIN deck has an additional set of parameter statements related to the

expected size of the temporary array that holds the input terrain elevation and land use. The size of this temporary array is computed by knowing the latitudinal extent, longitudinal extent and input resolution (1°, 30', 10', 5'). The TERRAIN program overestimates the amount of span the user will need for latitude and longitude. If the user allocates too little space, the TERRAIN returns a comment stating specifically how large to make the ITER variable and stops. Below is the standard setting for the ITER parameter, which is also located in the terrain.deck:

```
*ID COMPARM
*/ ---------------------------------- These parameters may need to be
*/                                    increased for large domains
*/      IHMAX=(XMAXLAT-XMINLAT)/XNC
*/  and JHMAX=(XMAXLON-XMINLON)/XNC  and XNC is 1, 0.5, 10/60, or 5/60
*/  for 1 deg, 30 min, 10 min or 5 min data.
*/      ITER >= MAX(IHMAX,JHMAX)
*/
*D PROGT.30
      PARAMETER(ITER=700)
```

## 4.3 Generated Files

The TERRAIN program generates one or two data files. If the master input file is set up for a fine-grid run NESTGD = TRUE), both an expanded domain (same size as the coarse domain if IEXP = FALSE) and a fine-grid file of terrain and land use are generated. The expanded domain is used by the DATAGRID program for the isobaric gridded first-guess fields, and the fine grid is used by the INTERP executable to generate a nested-domain initial condition for the model.

The TERRAIN program also generates a metacode file that is automatically sent to the Text and Graphics System (TAGS) at NCAR to generate color film. This file may also be sent to a standard black and white laser device, but one of the frames will be completely covered with toner due to the color-shaded instructions.

## 4.4 Hints and Caveats

Local orographic forcing and differential heating are important to mesoscale phenomena that develop near or occur over land. The user should pay close attention to modifiable attributes in the plots from the TERRAIN program. The user can modify the terrain elevation values or the land use categories at any specific $(i,j)$ location. Adjusting the terrain elevation permits a more realistic coastline or mountain range. Subjectively modifying the land use could entail freezing lakes or bays. Since the elevation and land use are used during every time step in the model, the simulation could be very sensitive to the additional information. (In figure 14, note the artificial lake generated in Mexico, 27N 103W, and the inundation of Louisiana and much of Florida.)

Large domains require very large allocations of memory. The data files that input

the global terrain and land use categories are minimized by a simple scheme that predicts the latitude – longitude box surrounding the model domain. The fine-resolution input data may require repeated job submission as the message "not enough space" continues to appear. The maximum memory often needs to be increased in the TERRAIN program (by adjusting the maximum memory parameter on the `# QSUB` command in the terrain.deck file, see section 3.1.1). The TERRAIN program currently generates the only metacode that includes the color-shaded frames. Domains over complex terrain with $10^4$ grid points overflow the default work space allocated to the graphics routines. Commenting out the GKS calls seems to be the only recourse.

Coarse grid distances larger than 180 km when used with a two way interacting nest and then analyzed to the grid points (not interpolated) have resulted in shifted mountain ranges and coasts in both the coarse and fine grid domains. This aberration has not been noticed when using the interpolation option (`IFANAL = FALSE` in the TERRAIN local input file). Report any occurrence of this phase shift to *mesouser@NCAR.UCAR.edu.*

If the TERRAIN program is not able to generate a value for the elevation or the land use category at any particular point, the initialized value of 0 remains. This occasionally happens in large domains where both the date line and Greenwich meridian are crossed, or when domains cross over both the pole and the equator. The TERRAIN program will not halt in this instance. The terrain errors will be interpreted as elevation near sea level. The land use categories are not utilized prior to the model. Any land use value (in this case, 0) outside the normal range (1 – 13) will cause the model indices for index-valued variables as albedo, surface availability, and surface roughness to become erroneous.

Even though the TERRAIN program may absorb an inordinate amount of memory, it is inexpensive to run. Users trying to determine if the TERRAIN run was successful might consider disposing the metacode file to the local front-end machine for viewing, instead of waiting for the film.

Users anticipating a nested-model run should prepare the master input file so that consistent coarse-grid and fine-grid terrain data sets may be generated during the first TERRAIN run. A mismatch between the coarse-grid and fine-grid terrain elevation files (as could happen when piecing together terrain elevation information) will cause mass oscillations between the coarse and fine-domains during the forecast. The coarse-grid model forecast should use a terrain data set specific to that configuration, not simply the coarse grid of a nested-terrain data set. Model forecasts that run with decreased diffusion could see small-scale noise introduced from the higher-than-resolvable scale terrain and land use information.

## 4.5 Incidental Metacode

Even though the terrain and land use data sets are used by subsequent programs, the metacode produced by the TERRAIN executable is arguably more important. The user needs to review the plots carefully before proceeding with the rest of the simulation.

The bench-mark case is a nested forecast, so the graphical output from TERRAIN reflects this request. A nested TERRAIN run will produce 11 frames of metacode, which

are automatically put onto film (the user may bring the metacode to a local workstation, but should not dispose the plotting instructions to fiche or to a laser printer because of the color-filled plots). Following are the figures from the fine-grid bench-mark TERRAIN run:

- the coarse-grid domain map background
- the coarse-grid terrain elevation (color filled contours)
- the coarse-grid terrain elevation contours
- the coarse-grid land use categories
- the coarse-grid dot point locations
- the coarse-grid upper air station locations
- the coarse-grid map background with the fine-grid location overlayed
- the fine-grid domain map background
- the fine-grid terrain elevation contours
- the fine-grid land use categories
- the fine-grid dot point locations

When a coarse-grid-only TERRAIN run is requested, the first 6 plots are received (the last 5 plots deal with the nested configuration). The metacode generated by the bench-mark case TERRAIN program follows on the next several pages. Descriptions of figures 11 - 21 appear only in the figure captions. The figures are displayed as samples of the typical output from the TERRAIN program and are simple to interpret.

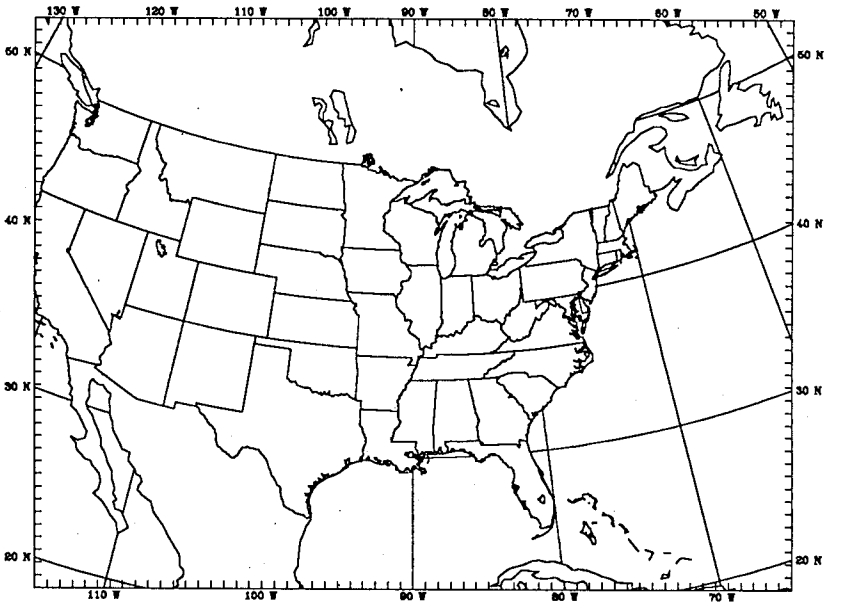



Figure 11. Frame #1 from the TERRAIN program showing the coarse-grid map background. This figure is output for both a coarse-grid-only TERRAIN submittal and nested TERRAIN submittal (generate a coarse-grid and a fine-grid domain).

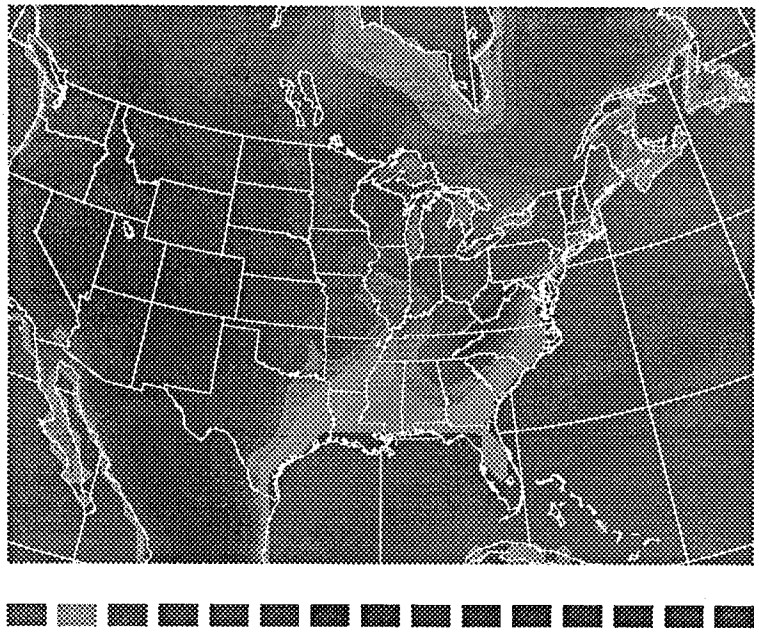


Figure 12. Frame #2 from the TERRAIN program showing the color filled image of the terrain elevation for the coarse grid. This figure is output for both a coarse-grid-only TERRAIN submittal and nested TERRAIN submittal (generate a coarse-grid and a fine-grid domain).

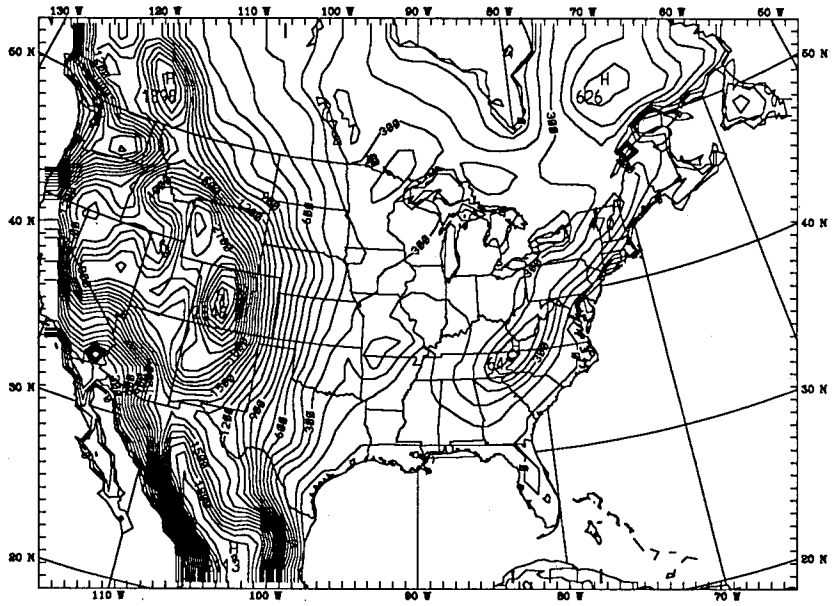



Figure 13. Frame #3 from the TERRAIN program showing the coarse-grid terrain elevation contours. The effect of the option to smooth the boundary gradient is obvious along the California coast. This figure is output for both a coarse-grid-only TERRAIN submittal and nested TERRAIN submittal (generate a coarse-grid and a fine-grid domain).

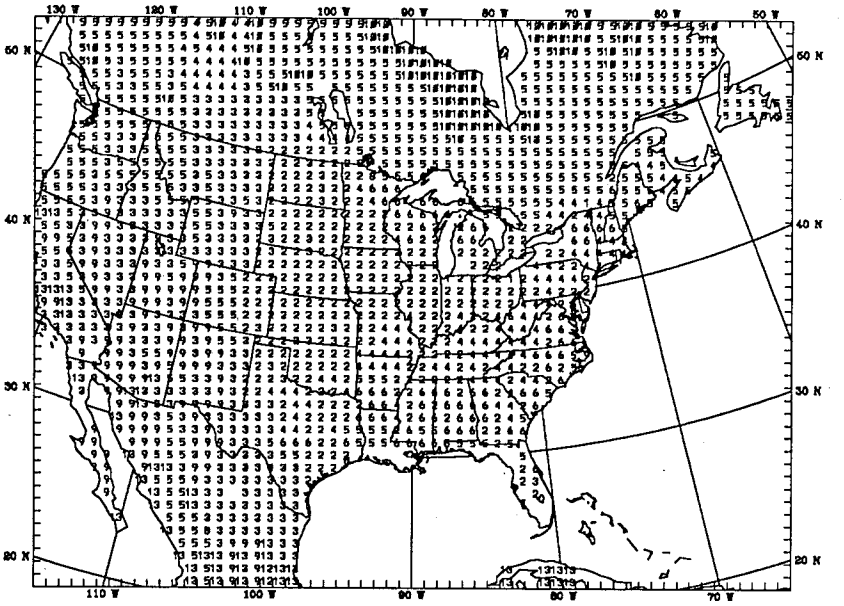



Figure 14. Frame #4 from the TERRAIN program showing the land use categories. This figure is output for both a coarse-grid-only TERRAIN submittal and nested TERRAIN submittal (generate a coarse-grid and a fine-grid domain).

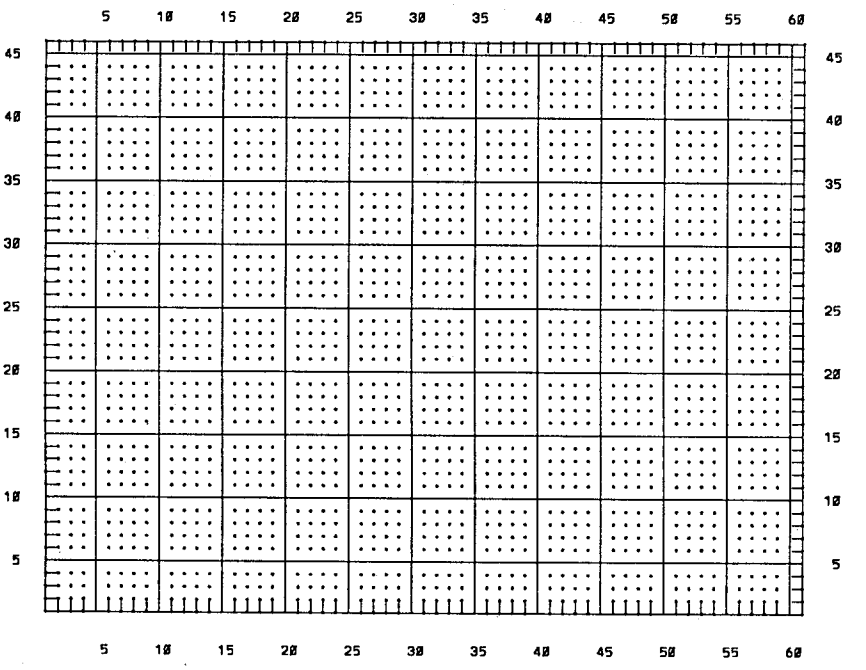



Figure 15. Frame #5 from the TERRAIN program showing the coarse-grid dot point locations. This figure is output for both a coarse-grid-only TERRAIN submittal and nested TERRAIN submittal (generate a coarse-grid and a fine-grid domain).

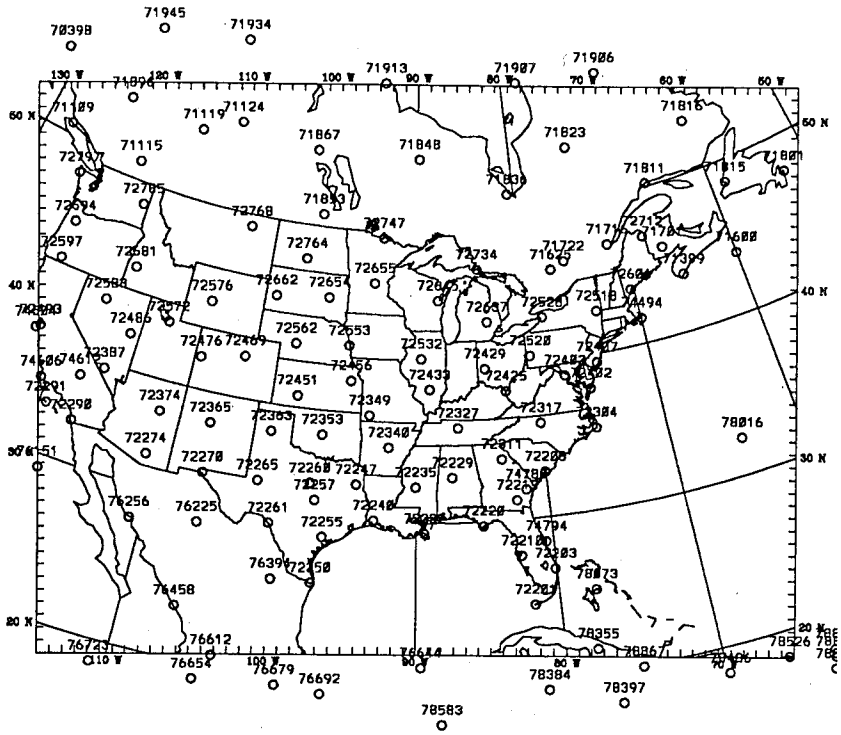


Figure 16. Frame #6 from the TERRAIN program showing the upper air rawinsonde stations located near the expanded domain. This figure is output for both a coarse-grid-only TERRAIN submittal and nested TERRAIN submittal (generate a coarse-grid and a fine-grid domain).

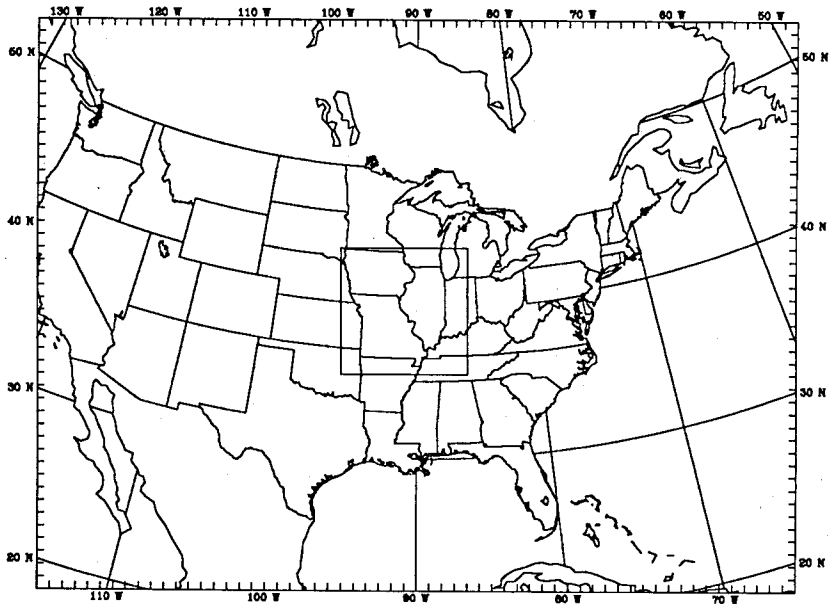


Figure 17. Frame #7 from the TERRAIN program showing the coarse-grid and fine-grid domain locations (small box surrounding Illinois). This figure is output only for a nested TERRAIN submittal (generate a coarse-grid and a fine-grid domain).

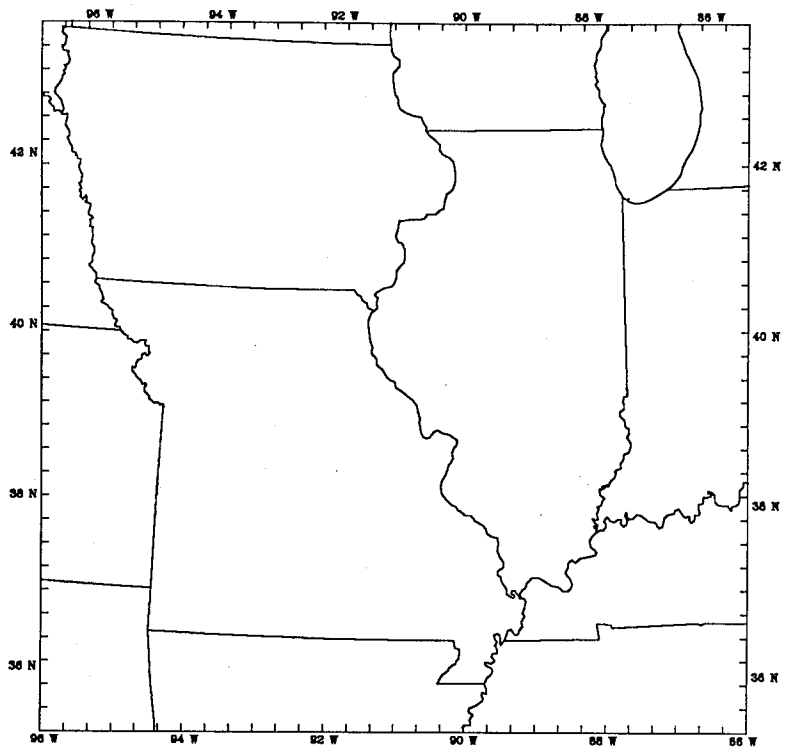



Figure 18. Frame #8 from the TERRAIN program showing the fine-grid map background. This figure is output only for a nested TERRAIN submittal (generate a coarse-grid and a fine-grid domain).

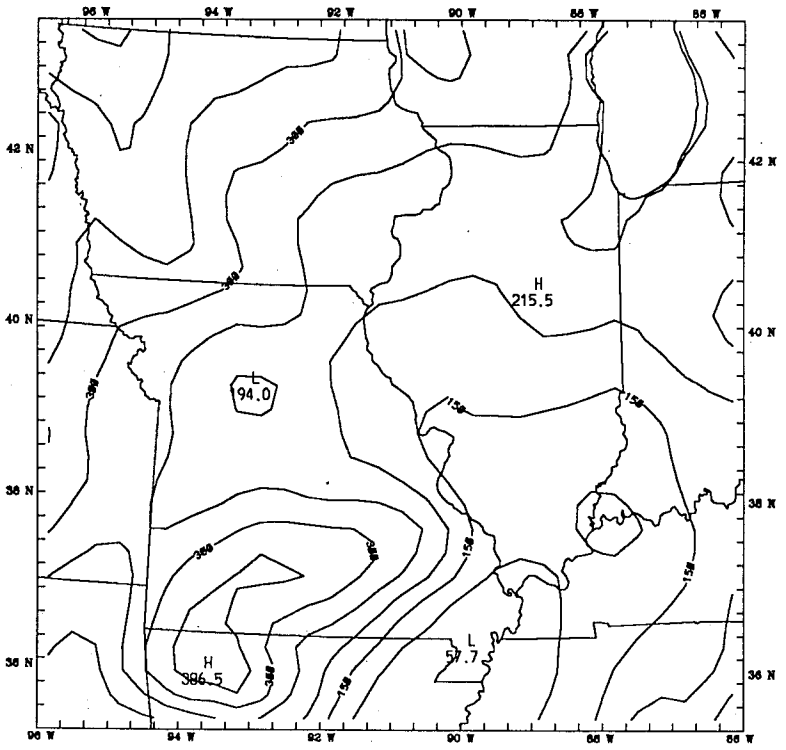



Figure 19. Frame #9 from the TERRAIN program showing the fine-grid terrain elevation contours. This figure is output only for a nested TERRAIN submittal (generate a coarse-grid and a fine-grid domain).

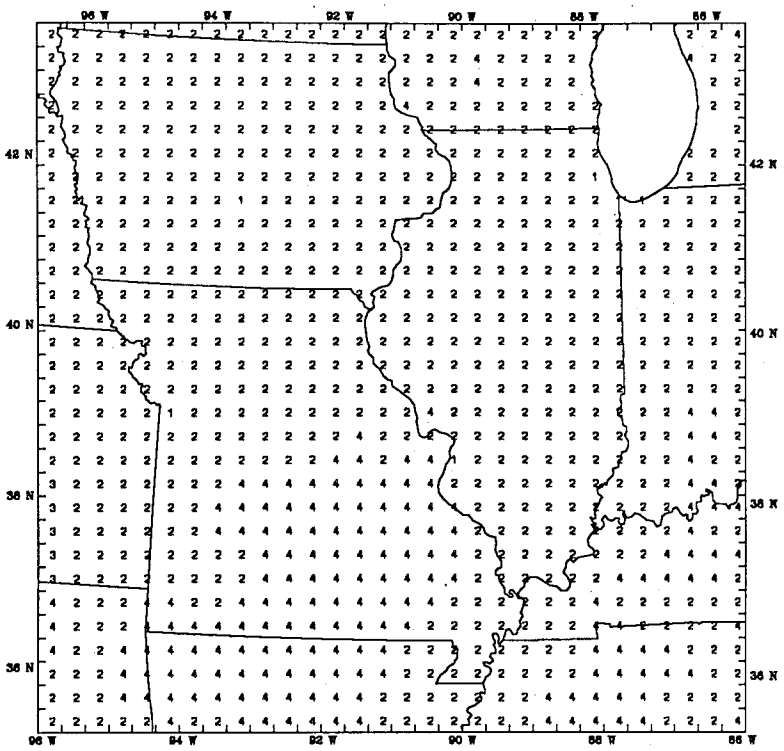



Figure 20. Frame #10 from the TERRAIN program showing the fine-grid land use categories. This figure is output only for a nested TERRAIN submittal (generate a coarse-grid and a fine-grid domain).

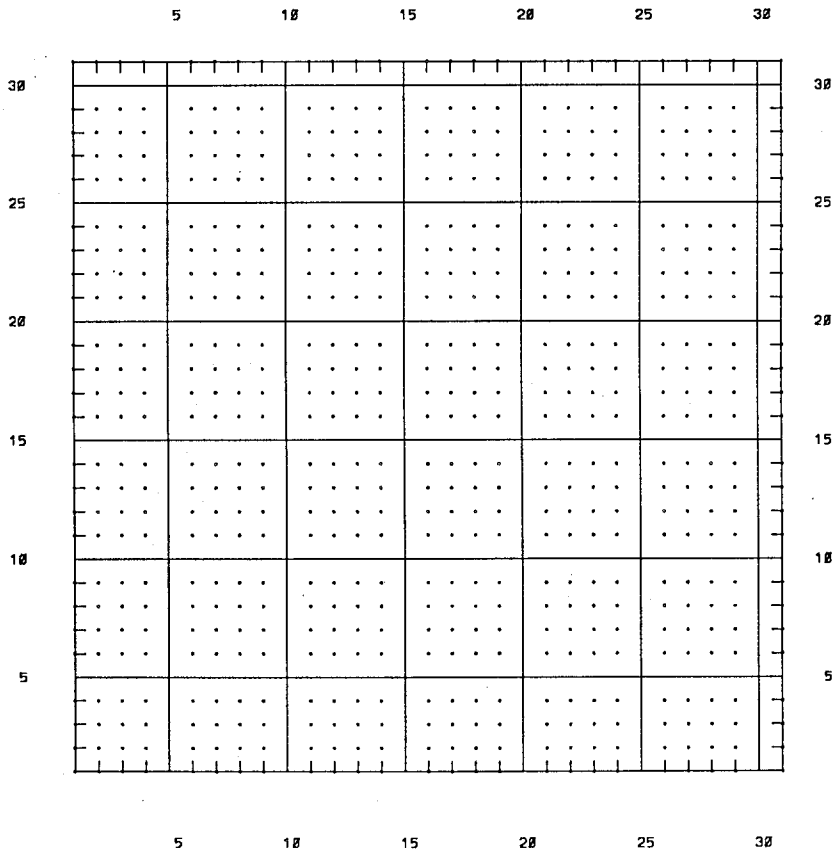



Figure 21. Frame #11 from the TERRAIN program showing the fine-grid dot point locations. This figure is output only for a nested TERRAIN submittal (generate a coarse-grid and a fine-grid domain).

# Chapter 5 DATAGRID

After the physical location of the domain is chosen and the TERRAIN program is successfully completed, the next step to run is DATAGRID. The DATAGRID program is the only other job deck to require the master input file (MIF). All of the information in the MIF is transferred to the subsequent programs through the record header that accompanies the data stream. A sample listing of a master input file is given in Appendix B.1. For a complete description of the DATAGRID output file, including the record header, see Appendix A.2.

The DATAGRID program interpolates the input latitude – longitude meteorological fields (mandatory pressure level wind, temperature, moisture, and height; sea level pressure, surface temperature and surface geostrophic wind) onto the expanded domain (same as the coarse-grid domain if this is not an expanded run). The additional fields of snow cover and sea surface temperature are also interpolated to the model grid. The snow cover data is archived approximately once per week with the NMC data. The sea surface temperature is available from both the NMC historical archives and in real time from the UNIDATA data sets. Both of these sources have sea surface temperature available at 24 hour intervals. A year of the NMC data was put into a monthly-mean file as a climatology when no other sea surface temperature data source is available. There are Navy data sets with sea surface temperatures, but only up through 1988 and only for the northern hemisphere. The final type of data generated by DATAGRID include the terrestrial fields that are either input (terrain elevation and land use categories) or computed (gridded values of latitude, longitude, map scale factors, and coriolis parameter).

For the user to run the DATAGRID program, the SCD inventory must be consulted so that the correct MS volumes are pulled onto the shavano disks for the dates requested during the forecast. Even though the information is available to the user, the MesoUser manager routinely updates the archival files for each catalog. These catalog files for both DATAGRID and RAWINS are maintained in the ~mesouser/catalog subdirectory (see section 2.2.1 for an example of the catalog entries used for the bench-mark case). Following is a sample C-shell fragment to acquire the NMC and ECMWF MS volume names directly without consulting the MesoUser catalogs:

```
#
cd $TMPDIR
#
#       generate catalog listings of the NMC and ECMWF analyses
#
#  acquire listing of all available ECMWF datasets
msread -fch catalog.ecm /DSS/D/DS1100
xprint catalog.ecm
#
#  acquire listing of all available NMC datasets
msread -fch catalog.nmc /DSS/D/DS0820
xprint catalog.nmc
```

The DATAGRID scripts, modification files, other internally activated decks (for the ECMWF data access), and the job deck are located in the ~mesouser/Decks/Datagrid subdirectory on shavano.

To run the model with real-time data, forecast boundary conditions are required. The DATAGRID code accepts the MRF product as the input "analysis" to generate the first-guess fields for the later programs. The required MRF files from UNIDATA are kept in the /wetterhorn/ldm/GRIB subdirectory, available from shavano (all files in the directory are packed as netCDF). To locate the most recent available data, issue the following commands from shavano:

```
cd /wetterhorn/ldm/GRIB
ls -lst *_a?.cdf | head -6
ls -lst *_oTS.cdf | head -1
```

The file names supply the mdate information needed for the DATAGRID shell variables **unidate** and **sstdate**. SCD maintains 3 - 5 days of the MRF data on wetterhorn (implying 6 - 10 separate MRF forecasts). All of these data sets are available for input to DATAGRID. SCD archives the aged data sets to the MS. For information concerning the permenant UNIDATA archive or the real-time data, contact *chifan@NCAR.UCAR.edu.*

To run the datagrid.deck file, the DATAGRID shell needs to be modified, but the same master input file read by TERRAIN must be reused. The next two sections describe the options available in the DATAGRID job deck concerning the local input file and the parameter statements. The shell variable options were discussed in section 3.1.3.2.

## 5.1 Local Input File

After searching the catalogs on shavano or the directories on wetterhorn for the correct volumes corresponding to the requested forecast period, the DATAGRID program can be modified. Information concerning which input to use (NMC or ECMWF analyses, or the MRF forecast), which source to choose for the sea surface temperature (archived NMC, Navy, climatology, or UNIDATA), and chosen dates for the snow cover data are among the options located in the local input file contained within the DATAGRID job deck. Following is a sample DATAGRID local input file:

```
&LOCMIF ;---------LOCAL MIF FOR PROGRAM DATAGRID-------------
;         SOURCE OF FIRST GUESS FIELDS, IF ECMWF, SET IHEMIS=3
IFIRST = 3HNMC,     ; SET TO 3HNMC, 5HECMWF, OR 6HUNIDAT
ISST   = F,         ; T/F USING CLIMATOLOGICAL SEA SFC TEMPS
UNISST = F,         ; TRUE IF USING UNIDATA SEA SFC TEMPS
;         SELECTION OF HEMISPHERE
IHEMIS = 0,         ; 0,1,2--N,S,BOTH,HEMIS; 3=NAVY SST(NORTH HEMIS)
;         PARAMETERS FOR MISSING DATA
IMSG   = F,         ; T/F INTERPOLATE MISSING NMC DATA
ISEQ   = 0,         ; SEQUENCE NUMBER OF MISSING DATA SET
JSEQ   = 0,         ; CONSECUTIVE NUMBER OF MISSING DATA TIMES
IFILE1 = 2,         ; NUMBER OF DATE FILES FROM FIRST INPUT VOLUME
;         SNOW INFORMATION FOR NMC SUBMITTAL
;         99999999= NO SNOW, 88888888= GENERATED SNOW FILE
ISNOW1 = 88010912, ; SNOW COVER DATE-PRIOR TO FCST
ISNOW2 = 88011612, ; SNOW COVER DATE-DURING FCST
ISNOW3 = F,         ; T/F SNOW COVER DATA IS ON EARLIER VOLUME
;         SNOW INFORMATION FOR ECMWF SUBMITTAL
SNOECM = 88011612, ; SNOW COVER DATE ON NMC FILE
;SNOFIL='FLNM=''/HAAGEN/SNOGEN/TESTX2'''L,  ; GENERATED SNOW FILE
;         MISSING NAVY SST DATES, 99999999= NONE ARE MISSING
MISNV1 = 88011800, ; FIRST SST DATE MISSING
MISNV2 = 99999999, ; SECOND SST DATE MISSING
& ;-----------------------------------------------------------------
```

Compared to the TERRAIN local input file, most of the DATAGRID switches are required to specify how the program is to run. The decision to use NMC data (IFIRST = 3HNMC), ECMWF data (IFIRST = 5HECMWF), or the MRF product (IFIRST = 6HUNIDAT) has already been made (the user supplied the information for the input volume names from the MS), but the local input file flags must be set consistently with the expected input data requested in the datagrid.deck shell. Other namelist values related to the input type are the hemisphere and the source for the sea surface temperature. If the user chose ECMWF as the input data set, then the hemisphere flag must be IHEMIS = 3 and Navy sea surface temperatures must be used (ISST = F). If any of the time periods for the Navy sea surface temperature are missing, the user must tell the DATAGRID program through the MISNV1 and MISNV2 flags. The eight digit integer 99999999 signifies no time periods of the Navy sea surface temperaure are missing, otherwise the mdate (YYMMDDHH) is the missing time. A maximum of two Navy sea surface temperature time periods may be missing for DATAGRID to successfully complete. For a UNIDATA input run, the user must set ISST = FALSE and UNISST = TRUE.

The NMC data occasionally has holes (synoptic time periods that were not archived). This information is printed on the NMC catalog listing. There are a couple of ways to deal with missing data. The easiest is to change time periods, but that is not usually feasible. The next step might be to use the ECMWF data, which has far fewer lapses in reported data. If the decision is made to stick with the NMC data, the switches for the missing

data in the DATAGRID local input file are required. The input flag to tell DATAGRID that a time period is missing is ISEQ. If the third time period to process is missing, then ISEQ = 3. The input parameter to tell DATAGRID how many time periods are missing in a row is JSEQ. If only the third time period is missing, then JSEQ = 1. If both the third and fourth time periods are missing, then ISEQ = 3 and JSEQ = 2. If the user would like to have DATAGRID temporally interpolate the data to the missing time periods, set IMSG = TRUE. Consider the lack of diurnal variation and the possible phase-lag effects on the model boundaries.

The remaining user options in the local input file for DATAGRID deal with snow cover. The snow data are only reported over the northern hemisphere. The reporting dates for the snow cover are not regularly spaced; as often as a couple of times a week during the winter months, and as infrequent as a couple of times a month during the warmer seasons. The snow cover may be updated once during the DATAGRID program. The snow cover is considered valid until the next snow cover date is available. The snow cover prior to the forecast period is used until the ISNOW2 snow cover date is reached. The mesoscale model uses only the first snow cover date, the data prior to the forecast. The snow cover dates are given in the standard modeling system mdate format (YYMMDDHH). The time period for the snow cover prior to the forecast is ISNOW1, and the time period for the snow cover during the forecast is ISNOW2.

If the snow cover data prior to the forecast is in a separate MS file not required by DATAGRID for meteorological variables, the flag to force the program to search an entirely different NMC volume is needed (ISNOW3 = TRUE). In this case, ISNOW1 is the snow date on the additional volume and the user must supply this data set's MS name to the DATAGRID job deck (C-shell variable InSnow).

There is no snow cover information in the NCAR archived data from ECMWF, users wishing to have ECMWF as the first-guess field and requiring snow cover data need to access the NMC data volumes on the MS (just as if the NMC first-guess data was to be used). Instead of the ISNOW1, ISNOW2, ISNOW3 flags, the user fills in the SNOECM flag with the mdate (YYMMDDHH) of the snow cover date requested from the NMC file. Users may generate their own snow cover data file, but this is not supported.

## 5.2 Parameterized Dimensions

The DATAGRID program generates 2–D and 3–D arrays of meteorological and terrestrial fields, but only the horizontal dimensions are important: the coarse-grid size (IMXC and JMXC) and the expanded domain horizontal size (IMX and JMX). If an expanded domain is not to be used in the later analysis by RAWINS, the coarse-grid and expanded-grid domains are the same size (IMXC = IMX and JMXC = JMX). The number of vertical levels in DATAGRID is only a function of the chosen value for PTOP in the master input file: all mandatory pressure levels up to and including PTOP are interpolated and output for RAWINS. No parameterization is required for the vertical coordinate in DATAGRID and no parameter values are required for fine-grid domain. Following is an example of the changes to consider for the parameter listing in the datagrid.deck file:

```
*D PARAM.11,12
      PARAMETER(IMX=58,JMX=73)
      PARAMETER(IMXC=46,JMXC=61)
```

## 5.3 Generated Files

The DATAGRID program generates the expanded-domain data set (the same as the coarse grid if **IEXP = FALSE** in the master input file), consisting of the meteorological fields interpolated to the model domain on mandatory pressure levels and several surface fields. The DATAGRID program is the first in the series of modeling system programs that includes a record header. The DATAGRID record header contains information from the MIF and the local input file; essentially metadata since it is data about the data. The data in the record header is required by the subsequent programs, and it is the technique employed to pass information from program to program about the input data. There is no metacode file generated by the DATAGRID program.

## 5.4 Hints and Caveats

The value specified in the master input file as the top level to input for DATAGRID is **PTOP**. This must be a mandatory pressure level (typical values are 100 mb, 70 mb, or 50 mb). This value is used later by the RAWINS program for analysis purposes and by every subsequent program to compute the pressure from the surface pressure and the vertical coordinate value.

The permanent data archive is managed by the Data Support Section of SCD, and they usually store the NMC analyses and observational data on the MS 4 - 6 weeks after the data are available in real time. For most historical purposes, this lag time presents no interruption in coordinated research. The ECMWF archive is only available through 1988. The MesoUser manager is responsible for updating the catalogs so that the most recently made available data sets are referenced in the inventory listings. Questions about these listings should be directed to *mesouser@NCAR.UCAR.edu.*

The UNIDATA archive began in late 1991. Through UNIDATA, the MRF is available 6 hours after the initial time of the forecast. This MRF product consists of a 48 hour forecast at 6 hour intervals. The UNIDATA first-guess files frequently having missing components. It is probable that at least one of the time periods did not get recorded from the 48 hour forecast, or that one (or more) of the variables during the forecast are not complete. Logistical obstacles exist when using the UNIDATA volumes. Questions concerning any of the archived data at NCAR should be directed to *jenne@NCAR.UCAR.edu.*

# Chapter 6 RAWINS

The RAWINS program is similar to the DATAGRID program in several aspects. Both require the MS volume names for the archived data ingest or the date for the UNIDATA products input, and both generate an isobaric analysis (+ some surface fields) suitable for INTERP, which creates input for the model. The RAWINS program does not use the master input file, as all of that information was been sent into RAWINS via the DATAGRID record header. Using the available observations, the primary task for RAWINS is to enhance the coarse mesoscale analysis from DATAGRID. Secondly, RAWINS is responsible for supplying the mesoscale model with analyses and observations for Four Dimensional Data Assimilation (FDDA). Appendix A.3 has specific information pertaining to the analyzed gridded data output format for RAWINS. Appendix A.4 deals with the gridded FDDA surface analysis. Appendix A.5 details the FDDA upper-air and surface observation files created by RAWINS for the observation nudging option in the model. Each of these appendices contains a general description of the variables and a sample FORTRAN code to read the data.

The data from DATAGRID is input into RAWINS on mandatory pressure levels. The list of new pressure levels requested initially in the master input file (and carried by the record header from DATAGRID) is searched and the meteorological fields are vertically interpolated to those additional isobaric levels. The archived observations that are input into the program are unpacked if they are within a couple of degrees of being inside the latitude – longitude box surrounding the expanded domain (this is the same as the coarse-grid domain if `IEXP = F` in the master input file). An objective vertical consistency check is made on the upper-air station observations before they are plotted, written, and stored.

A successive-scan Cressman objective analysis scheme is performed on both the mandatory and additional new pressure levels to incorporate both the large-scale features as well as the small-scale information from individual observations. The observations used in the objective analysis have gone through two more quality-control checking procedures: 1) comparison of groups of observations with neighbors, and 2) comparison of observations to the original analysis.

The primary method in RAWINS for deciding the validity of an observation is to compare groups of neighboring observations among themselves to determine the outliers (buddy check). Even though several of these observations may be different from the first-guess field (but less than the error maximum criteria), if they all tend to corroborate each other (near a strong gradient, for example) they are not removed from the analysis. As a gross error check, the observation's location is determined, and the interpolated value of the first-guess field at that point is calculated. Any absolute difference larger than a user-defined maximum (a function of time, vertical level, terrain elevation, land use category, and meteorological variable) causes the observation to be removed from the analysis. In data sparse regions (over oceans) or when observations are similarly erroneous (several sea level pressures reported at 1090 mb instead of 990 mb), this additional quality-assurance technique complements the buddy check.

Before the analysis is final, a downward vertical pass at each grid point is made through the temperature field from the 500 mb level to the surface, to remove any super-adiabatic layers (they are set slightly stable). If the domain is expanded, RAWINS removes the the additional outer rows and columns of the grid during the final output of the data. The file from RAWINS is not the expanded-grid size, it is the coarse domain.

The RAWINS program is responsible for generating input used for the observation and analysis FDDA capabilities in the predictive model. At the surface, the analysis nudging data is usually required to be at a higher temporal frequency than the rawinsonde 12-hour reporting intervals. There are two ways that the program computes first-guess 3-hour surface analyses: through time interpolation from each of the anchoring 12-hour time periods, or through a lag-time approach which uses the analyzed previous time as the current first guess (a Barnes analysis is available to create the first-guess field directly from the observations, but is currently bypassed). Both of the temporal interpolation schemes employ a Cressman-type objective analysis to insert observations into the generated first-guess field. The data in the observation nudging file for the model have successfully cleared both of the quality checks, in addition to the initial vertical consistency adjustment.

RAWINS requires the rawinsonde observations. If the user has requested that a surface analysis also be performed, 3-hour and 6-hour surface, ship and buoy data are also required. A surface analysis using the 00Z or 12Z surface observations is normally always performed. With the current restrictions for time step size in the PBL scheme in the model, much of the model's CPU time is spent in the boundary layer. The surface analysis is important for proper definition of the boundary layer.

The RAWINS program accepts observations from both the NCAR archive and the real-time data from UNIDATA (packed in netCDF). When RAWINS uses the UNIDATA observations, only one time period of RAWINS is generated as only one time period of observations are ingested. The UNIDATA observations are usually available 90 minutes after 00Z or 12Z. To find the list of available observations from UNIDATA, issue the following commands from shavano:

```
cd /wetterhorn/ldm/decoded
ls -lst *_sao.cdf | head -1
ls -lst *_upa.cdf | head -1
```

The sao file refers to the surface observations and the upa file contains the rawinsonde station reports. The UNIDATA observational archive does not include global coverage. The real-time wetterhorn directory contain 3 - 5 days of observation files. After this aging period, the files are permanently archived to the MS. Users requiring access to these archived volumes should contact *chifan@NCAR.UCAR.edu.*

File names for the SCD archived observations on the MS are located in the inventory catalogs in the ~mesouser/catalog subdirectory on shavano (more information concerning the catalogs is given in section 2.2.1 of this document). The MesoUser manager is responsible for posting updated listings of the current status of the observational data archived by SCD.

Users may access the SCD inventory files on the MS without consulting the MesoUser catalogs. Following is a sample script to access the 12-hour upper-air rawinsonde files, the 6-hour surface, and the 3-hour surface, ship and buoy data files:

```
#
cd $TMPDIR
#
#       generate catalog listings of the observations
#
#  acquire listing of all available RAOB datasets
msread -fch catalog.raob /DSS/D/DS3531
#
#  acquire listing of all available SURFACE datasets
msread -fch catalog.sfc /DSS/D/DS4640
```

The RAWINS scripts, modification files, color tables and map tables are located in the ~mesouser/Decks/Rawins subdirectory on shavano. To run the rawins.deck file, only the RAWINS shell needs to be modified. The master input file is not used by RAWINS or any of the subsequent programs.

## 6.1 Local Input File

After searching the catalogs on shavano or the directories on wetterhorn for the correct volumes corresponding to the observations for the requested period, the RAWINS program can be modified. The switches for the available types of objective analysis, bogus data sets, error criteria, choices of input data, graphical products, and FDDA output files are among the options located in the local input file (contained within the RAWINS job deck).

The majority of Appendix B covers the description of the various types of bogus files available to the RAWINS program (the bogus options must be activated from inside the RAWINS local input file). Appendix B.2 details the auto bogus file's variables, and the way to use the auto bogus file to adjust the analysis. Appendix B.3 deals with the kbogus option, which allows a user to modify point observation values. When the user is required to create new surface or upper-air stations, from some previous subjective analysis, Appendix B.4 covers the use of the nbogus file. Appendix B.5 discusses the nselim option of completely removing an upper-air or surface station report from the analysis.

All of the modeling system local input files are FORTRAN namelist structures, a non-standard extension. Questions concerning the namelist syntax should be directed to a Cray FORTRAN manual. The RAWINS shell requires information concerning the submittal number (`Submit`). This value is imported to the local input file through shell variable expansion (`RWSUMB`). Users should not need to modify local input variables that have been assigned by the shell.

On the following page is the RAWINS local input file used to produce several of the RAWINS bench-mark data sets:

```
&LOCMIF ;---------LOCAL MIF FOR PROGRAM RAWINS-------------
;            OBJECTIVE ANALYSIS OPTIONS
NOBLND  = F,          ; T/F NO OBJECTIVE ANALYSIS
IFAC    = 6HALTERS,   ; ALTERS--FIRST GUESS FIELD USED; ANALYS(NO)
IWTSCM  = 6HBANANA,   ; RAD OF INFLU; BANANA-ELLIPS-CRESMN(CIRC)
IWIND   = 6HMB1000,   ; SFC WIND 1ST GUESS--SFC GEOSTR(SFCGEO) OR 1000MB
IWT     = T,          ; T/F WEIGHT IS SQUARED IN OBJECTIVE ANALYSIS
SMOOTH  = T,          ; T/F SMOOTH FIELDS AFTER OBJECTIVE ANALYSIS
IPOT    = F,          ; T/F OBJECTIVE ANALYSIS ON ISENTROPIC SURFACES
;            DATA INPUT OPTIONS FOLLOW
UNIOBS  = F,          ; TRUE FOR UNIDATA OBS, FALSE FOR ARCHIVED OBS
ALRAWS  = T,          ; T/F USE ALL RAOBS WITHIN DOMAIN
NSTATN  = 0,          ; NUMBER RAOBS WANTED--MODS REQUIRED SUB SETUPS
NSELIM  = F,F,F,F,F,F,F,F,F,F,F,F,F,  ; T/F DELETE STATIONS
NBOGUS  = F,F,F,F,F,F,F,F,F,F,F,F,F,  ; T/F BOGUS COMPLETE STATIONS
KBOGUS  = F,F,F,F,F,F,F,F,F,F,F,F,F,  ; T/F BOGUS INDIVIDUAL LEVELS
IUPPER  = T,          ; T/F ACQUIRE AND ANALYZE SOUNDINGS
ISFCS3  = T,          ; T/F ACQUIRE AND ANALYZE SHIPS AND BUOYS
ISFCS6  = T,          ; T/F ACQUIRE AND ANALYZE SURFACE LAND DATA
IUINTVL = 12,         ; TIME INTERVAL FOR SOUNDING DATA
;            PRINT OPTION FOR SURFACE INPUT DATA
ISPRNT  = F,          ; PRINT SURFACE INPUT OBS
F4D     = T,          ; T/F CREATE EXTRA FDDA SURFACE DATA VOLUME
INTF4D  = 3,          ; TIME INTERVAL FOR FDDA SURFACE DATA (HR)
;            INTERPOLATION FOR NON-STANDARD TIME 1ST GUESS
;            IF INTFRD > 3, SET LAGTEM=FALSE
LAGTEM  = T,          ; TRUE= LAG TIME, FALSE= TEMPORAL INTERPOLATION
;            RWSUBM=0...ONLY ONE SUBMITTAL INTENDED FOR RAWINS
;            RWSUBM=1...FIRST RAWINS SUBMITTAL--TWO ARE INTENDED
;            RWSUBM=2...SECOND RAWINS SUBMITTAL
RWSUBM  = $Submit,    ; SET TO 0,1, OR 2
;            BUDWGT VARIES THE TOLERANCE OF THE BUDDY CHECK
;               BUDWGT=1.0    APPROXIMATES NORMAL ERROR TOLERANCE
;               BUDWGT>1.0    INCREASES THE ERROR TOLERANCE
;               BUDWGT<1.0    DECREASES THE ERROR TOLERANCE
BUDWGT  = 1.25,       ;
;            MAX DIFF ALLOWED BETWEEN FIRST GUESS AND STATION DATA,
;            IF USING BUDDY (BUDWGT>0.1), SET LARGE VALUES FOR ERRMAX
;               SUGGESTED: ERRMXT=15.0, ERRMXW=14.0, ERRMXP=8.0
ERRMXW  = 14.0,       ; MAX WIND DIFFERENCE ALLOWED (m/s)
;                        1000<P     ERRMXW=ERRMXW*1.25
;                        PTOP<P<500 ERRMXW=ERRMXW*1.5
ERRMXT  = 15.0,       ; MAX TEMP DIFF ALLOWED (C)
;                        00Z, TERRAIN < 500m ERRMXT=ERRMXT*1.25
;                        00Z, TERRAIN > 500m ERRMXT=ERRMXT*1.75
;                        12Z, ERRMXT=ERRMXT*1.5
;                        OVER WATER, ERRMXT=ERRMXT*0.75
ERRMXP  = 8.0,        ; MAX SEA LEVEL PRESSURE DIFF ALLOWED (mb)
;            PLOT OPTIONS
IPLOT   = T,T,T,T,T,T,T,T,T,T,T,T,T,  ; PLOT VERTICAL RAOBS
ABFLAG  = T,T,T,T,T,T,T,T,T,T,T,T,T,  ; CONTOUR GRID FIELDS
ABOVER  = T,T,T,T,T,T,T,T,T,T,T,T,T,  ; OVERLAY AUTOBOGUS OBSERVATIONS
& ;-------------------------------------------------------------
```

The size of the local input file for RAWINS is an indicator of the complexity of the interrelated requirements to successfully run the program. There are several values which users rarely modify and can be ignored during most RAWINS submittals. The first seven switches, all for the objective analyses options, fit into this category: NOBLND, IFAC, IWTSCM, IWIND, IWT, SMOOTH, and IPOT.

The switches for the data input options refer to the bogus data and files, the archived observational data, or the UNIDATA real-time set. Once the inventory catalogs have been checked, and the user has filled in the correct file names in the C–shell script, the default switch settings on whether or not to include some archived data should not really be considered. If the user does not want to do an objective analysis, there is no need to run the RAWINS program at all, the original first-guess fields from DATAGRID should suffice. The bogus switches (ALRAWS, NSELIM, NBOGUS and KBOGUS) are logical flags that refer to specific time periods. They allow an additional level of quality control over the observations. Activating these options requires the user to create input files for RAWINS; effectively, the user is subjectively generating new stations to objectively modify the analysis. The other subjective approach to modify the analysis is with auto bogus, which is a subjective enhancement of the maximum error criteria (see Appendices B.2 through B.5 for more information on the bogusing options). Most users attempt to modify their analyses with the auto bogus capability, instead of with the bogus options mentioned here.

If the model is expected to be nudged towards an analysis and/or the observations, the F4D logical needs to be set. For analysis nudging, the time interval (hours) between the generated surface files is required (typically INTF4D = 3). Since surface analyses are not available as first-guess fields for the FDDA files at every time, the user chooses how to generate these initial fields. The two choices are through a lag-time scheme (the final analysis of hour 0 is the first-guess field of hour 3, the final analysis of hour 3 is the first-guess field of hour 6, etc.) or through a temporal interpolation (the first guess of hour 3 is a linear interpolation between the first-guess surface analyses of hours 0 and 12).

Users experienced with the modeling system occasionally want to tinker with the observation quality control subjectively. The most direct to way proceed is with the auto bogus option. The user needs to inform the RAWINS program of this intent with the RWSUBM flag. This method is more structured than the previously mentioned suite of bogus options, but it is tedious. The auto bogus option will allow the user to modify the surface wind, surface temperature, and sea level pressure every 3 hours; upper-level wind and temperature every 12 hours. The relative humidity and the geopotential height are not modified through the auto bogus facility (the geopotential is not an analyzed field in RAWINS).

Without using any of the bogusing options, the analysis can be modified through the maximum error switches. Observations deviating from the specified tolerance are removed. The user may choose to have the observed wind, temperature and sea level pressure observations (not relative humidity or geopotential height) verified against a critical difference (ERRMXW, ERRMXT, ERRMXP). Any observation differing from the first-guess field (the coarse analysis from DATAGRID interpolated to the observation's location) by more than the critical cutoff is discarded from the analysis. Lists of these observations

are printed in the output from RAWINS and comprise the auto bogus output file (re-inserted into the analysis during the second RAWINS submittal). The other tunable error criteria compares observations among themselves with the buddy check (the neighboring observation option) with BUDWGT.

To maximize the capabilities of the buddy check option, the maximum error tolerance should be increased above the value normally used during the auto bogus option. The user should verify that the majority of the discarded observations were removed by the buddy check and not subsequently with the maximum error check. Values for the error maximums and the buddy criteria are given in the namelist. A RAWINS job that uses the buddy check and the error maximum could have the following settings:

```
RWSUBM =  0
BUDWGT =  1.25
ERRMXT = 15.
ERRMXW = 14.
ERRMXP =  8.
```

The buddy check option is activated (BUDWGT > 0.1) implying that the user is not employing the auto bogus options (either submittal 1 or 2). Any significant departure of the observations compared to the analysis (larger than the error maximum criteria) will still cause the observation to be removed from the analysis, but the cutoff values are elevated. A temperature difference of 15.1 °C, even if corroborated with several other large temperature errors (for example, 14.9 °C, 14.8 °C), would not be used in the analysis. The buddy check facility should not be used in conjunction with the auto bogus capabilities.

When RAWINS is not using the buddy check option, the last quality check before an observation is utilized is the error maximum criteria. This requires a more conservative tightening of acceptable observation differences to avoid the inclusion of a spurious value. Typical values for RAWINS job without buddy checking are:

```
BUDWGT =  0.
ERRMXT = 10.
ERRMXW = 10.
ERRMXP =  6.
```

With these settings, the buddy check option is de-activated, any value of the RWSUBM flag is eligible: 0, 1, or 2. The choice to utilize an observation is based solely on the the difference between the observation and the value of the first-guess analysis, interpolated to the observation location.

## 6.2 Parameterized Dimensions

The RAWINS program is the first program to require allocation of space via the parameter statements for the vertical levels of the data. RAWINS reads in the expanded-domain analysis on mandatory levels from DATAGRID (if RAWINS is not doing an analysis on the expanded domain, then the expanded grid and the coarse grid are the

same size) and writes out coarse-domain data that has been interpolated and analyzed (including any additional new pressure levels). The horizontal grid parameter statements are the same as in DATAGRID, the coarse-grid dimensions are given by IMXC and JMXC and the expanded-domain horizontal dimensions are IMX and JMX. If an expanded domain was not used in the RAWINS analysis, the coarse and expanded grids are the same size IMXC = IMX and JMXC = JMX.

The parameterized number of vertical levels in RAWINS, LMX, is the sum of the number of mandatory pressure levels in DATAGRID + the number of new pressure levels requested in the master input file + 1 (for the surface). There are no parameter statements for the fine grid. Following is an example of the changes to consider for the parameter listing in the rawins.deck file:

```
*D PARAM.11,12
      PARAMETER(IMX=58,JMX=73,LMX=24)
      PARAMETER(IMXC=46,JMXC=61)
```

Since RAWINS ingests observations that cover the globe, there are parameter values for the maximum number of these observations which the program needs to handle (for which space needs to be allocated). These two flags have a simple interpretation: IRB is larger than the maximum number of upper-air rawinsonde stations in the domain, and IRS must exceed the total number of station reports (upper-air + surface) in the domain. For example, a domain over the the US and Canada has less than 150 upper-air stations and less than 1200 surface stations, so IRB = 150 and IRS = 1350 would adequately allocate space. A large domain covering much of Asia might contain 500 upper-air stations and 2500 surface stations, requiring IRB = 500 and IRS = 3000. These parameterized switches are located in the rawins.deck file and only need to be increased when the RAWINS program says all of its tables are full. Following are the default values for the maximum number of observations found in the domain:

```
*ID STATPAR
*/
*/      CHANGE MAXIMUM NUMBER OF RAOB STATIONS ==========> IRB
*/      CHANGE MAXIMUM NUMBER OF SFC + RAOB STATIONS ====> IRS
*/
*CDK COMPOR
      PARAMETER(IRB=500)
*D COMP2R.2
      PARAMETER(IRS=3000)
```

## 6.3 Generated Files

The RAWINS program is capable of generating quite a few different types of data sets for the user: gridded isobaric analyses, observations for FDDA, gridded surface analyses for FDDA, the auto bogus input file, and a processed rawinsonde output file. The following paragraphs in this section each describe one of the RAWINS output files.

The gridded analyses are the RAWINS output that are sent to the GRIN program to be vertically interpolated for input into the mesoscale model as initial and boundary conditions. This file has the standard version 8 record header and is the observationally enhanced version of the data originally input from DATAGRID. This is the file that is implied when references are made to RAWINS output.

The observation file for the model for FDDA is initiated by RAWINS. The upper-air observations in this file are defined on the pressure levels valid during the RAWINS submittal (implying the original rawinsonde observations have undergone vertical interpolation). Both the upper-air and surface observations have been quality checked: if an observation is in these files, it was included in the analysis. These files are sent through a pre-processor prior to being input into the mesoscale model when station nudging is used. The model requires observation locations expressed in ($x$, $y$, $\sigma$), which are not available from RAWINS.

Another FDDA option in the model is surface analysis nudging. When the surface tendencies are to be nudged, the analyzed surface file from the coarse domain is ingested by MM4. If nested domain tendencies are also nudged to the surface analysis, the coarse-grid file must have previously been run through the INTERP program.

During the initial RAWINS run when `RWSUBM = 1`, the auto bogus file is created and sent to the user. This ASCII character file lists every observation that was removed from the analysis by RAWINS. The auto bogus file, along with the Stäve vertical-profile soundings and the auto bogus plots, provide information for the user to subjectively decide whether to re-insert that value (at that one level, for that one time) back into the analysis during the second RAWINS submittal (`RWSUBM = 2`).

The last type of data files output from RAWINS are the decoded rawinsonde and surface data. These files contains all of the observations available to RAWINS (one file of rawinsonde data, one file of surface reports). The stations are located inside the domain and an objective vertical consistency check has been performed. The files are output prior to the quality-control error checks, so observations thrown out of the analysis by RAWINS (as well as those observations that are incorporated into the analysis) are present in these files. All of the significant level data as well as the mandatory data are present. The upper-air data has not been vertically interpolated to the analysis levels in RAWINS.

In addition to all of the data files, there are two different types of metacode that are generated. When `IPLOT = TRUE` for a particular time period, the upper-air station soundings are plotted. These plots are generated prior to the quality-control checks from the upper-air reports on the original vertical levels (not the vertically interpolated FDDA observation file). These plots are sent to the user rotated on microfiche.

The other metafile generated by the RAWINS program is controlled by the `ABFLAG` and `ABOVER` logical flags. These are usually activated during the first (of the two expected) RAWINS submittals typical of using the auto bogus capabilities. Horizontal maps of analyzed wind, temperature and pressure are overlayed with the observations that RAWINS removed from the analysis. These plots are returned to the user on color microfilm. Without the benefit of color, too much information is displayed on each frame

for easy interpretation.

## 6.4 Hints and Caveats

RAWINS is the first expensive program to run in the series of jobs required to generate a forecast. The printout from the RAWINS code is extensive.

- Every observation that is inside the computed latitude – longitude box surrounding the expanded domain (or coarse domain) is checked for inclusion inside the actual analysis domain. Those falling outside are listed with an explanation.
- Observations with missing levels, or those with levels not extending up to the top analysis level are reported.
- All suspect observations that are modified, or data that is removed out are printed in the listing, along with the observation value, first-guess interpolated approximation, error tolerance, and modified value.
- Every observation of all encountered bogus files, the entire auto bogus file (the original on the first submittal, and the modified on the second submittal), a gridded sample of both the first-guess field and the final analysis on several pressure levels are output.

When having difficulties with RAWINS, first consult the printout to determine what the program thought the problem might have been.

Users considering FDDA in the model typically require the subjective capability of the auto bogus option. Even though the 12-hour upper-air and surface periods may be reasonable, the intermittent 3-hour surface fields are prone to inconsistencies due to the temporal interpolation techniques used to create their first-guess fields. If a large part of the domain is over ocean, there are fewer observations to enhance the analysis. The observation density is recorded in the surface FDDA file so that data sparse areas receive zero weighting in the model nudging coefficients. For a model run nudging towards an analysis, every time period must be subjectively scrutinized. The model nudges towards the wind, temperature and moisture, but does not nudge towards the sea level pressure. After the initial time period, unless the user intends to run the VERIFY program or initialize the model at a later time, spending effort modifying the sea level pressure is wasted.

For the users who will not be performing FDDA in the predictive model, but are using the auto bogus error checking in RAWINS, only the initial time period in the analysis is critical. Spending hours (or days) on the auto bogus file by subjectively enhancing the second or subsequent time periods is essentially worthless to the model, since the analyses interior to the boundary conditions are discarded (unless VERIFY is to be run or if the model is to be initialized at a later time).

Do not use the buddy checking option (`BUDWGT > 0.1` in the RAWINS local input file) and the auto bogus option (`RWSUBM = 1` or `RWSUBM = 2`) at the same time.

The vertical profile sounding plots from RAWINS are sideways on the fiche. To view them it is helpful to cut the fiche in half.

Over very high terrain, the super-adiabatic lapse rate check can introduce cold surface temperature "diamonds" into the analysis due to inadequacies in the surface pressure computation. Since surface air temperature is used to compute the reservoir temperature in the model, users should look for erroneous features not present in the DATAGRID surface temperature field.

## 6.5 Incidental Metacode

There are two types of metacode that the user can request from RAWINS. Both of these options are activated from the RAWINS local input file inside the rawins.deck file. If the logical flag `IPLOT` is set to true, the RAWINS program will generate a Stüve plot for every upper-air station that has successfully passed through the vertical consistency check. Not all of the stations plotted in this file will be in the analysis, as there are more quality-assurance routines before inclusion into the final analysis.

Since the modeling system is usually run in a research mode, with archived observations, every time period generates a plot for each reporting upper-air station. This metacode file is written to fiche and sent to the MS. Since this is a simple black and white line plot file, it may disposed to paper, or viewed on a graphics terminal (the size of the file can be prohibitively large). Figure 22 is a sample of the sideways vertical profile plots from RAWINS.

The upper-air Stüve plots have the 5 digit upper-air identification tag, the date (84 3 28 12 is 12Z 28 March 1984), and the latitude - longitude location. The wind barb speeds are knots: half barbs are 5 kts, full barbs are 10 kts, and flags are 50 kts. The ordinate axis is linear in $p^{-\kappa}$ (mb), the abscissa is temperature (°C).

When the user chooses the auto bogus technique to subjectively tune the analysis, the auto bogus plots in addition to the vertical profile plots should be generated. The user sets the `ABFLAG` and `ABOVER` switches to true, along with the `IPLOT` option for the Stüve plots.

On each auto bogus plot, the final analysis from RAWINS on the expanded domain is plotted and overlayed with the observations that were removed for that variable, at that time, for that level. To aid the user in interpretation of the data, different information on the plots are color coded. Accompanying the plots is an ASCII listing of every observation that was removed (Appendix B.2 describes the auto bogus file). Following are the auto bogus maps for the sea level pressure analysis (figure 23), the surface temperature analysis (figure 24), and the surface wind analysis (figure 25).

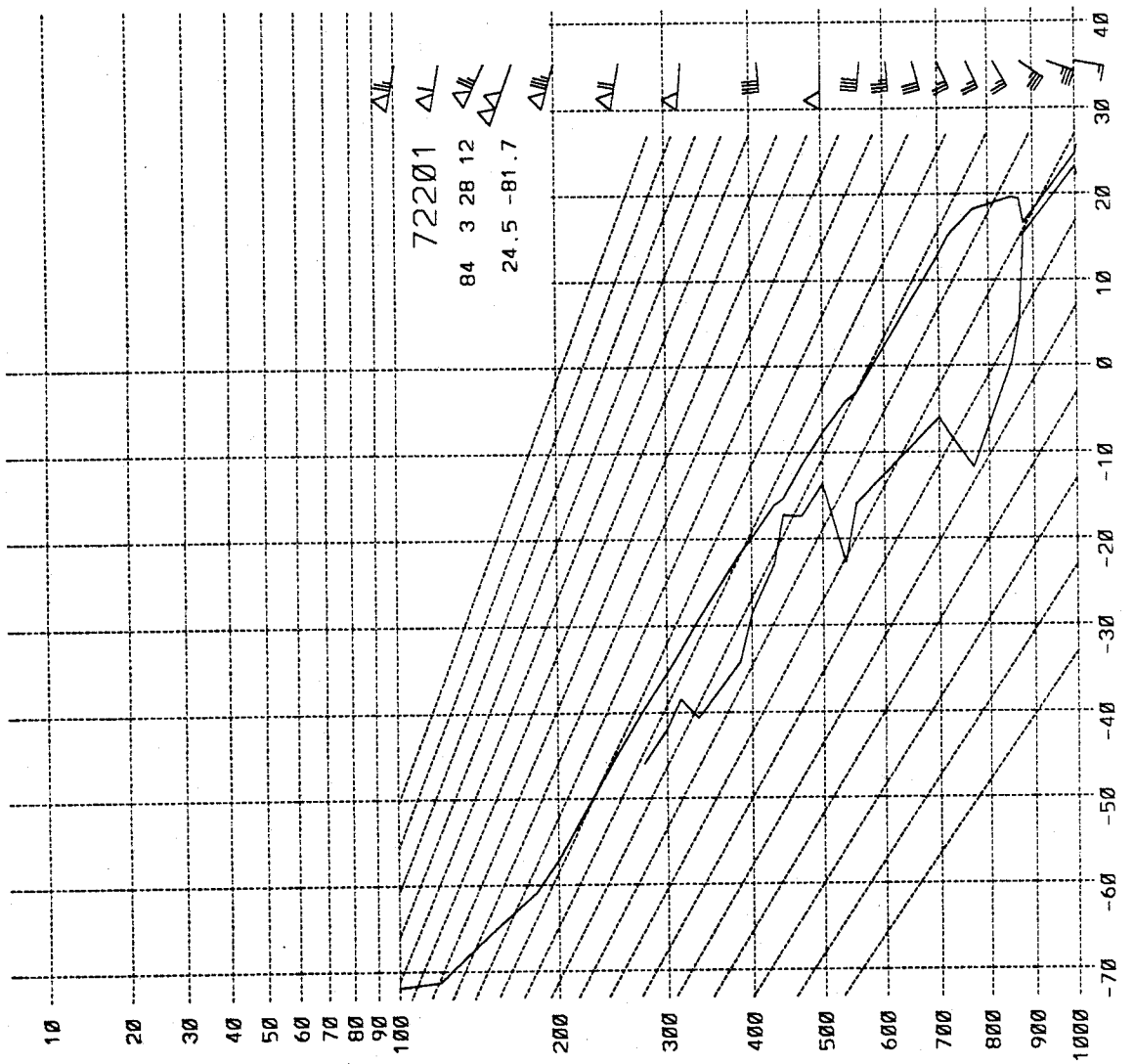



Figure 22. A typical Stüve plot from RAWINS, showing the vertical distribution of temperature, dew point, and wind. The sloping lines are constant potential temperature (K), the ordinate is pressure (mb), and the abscissa is temperature (°C). The wind flags employ a flag for 50 kts, a full barb for 10 kts, and a half-barb represents 5 kts. The 5 digit station identifier (72201), the reporting time (in mdate YY MM DD HH) format, the latitude (24.5 N), and longitude (81.7 W) are given. Each rawinsonde station, for each reporting time, has a plot generated.

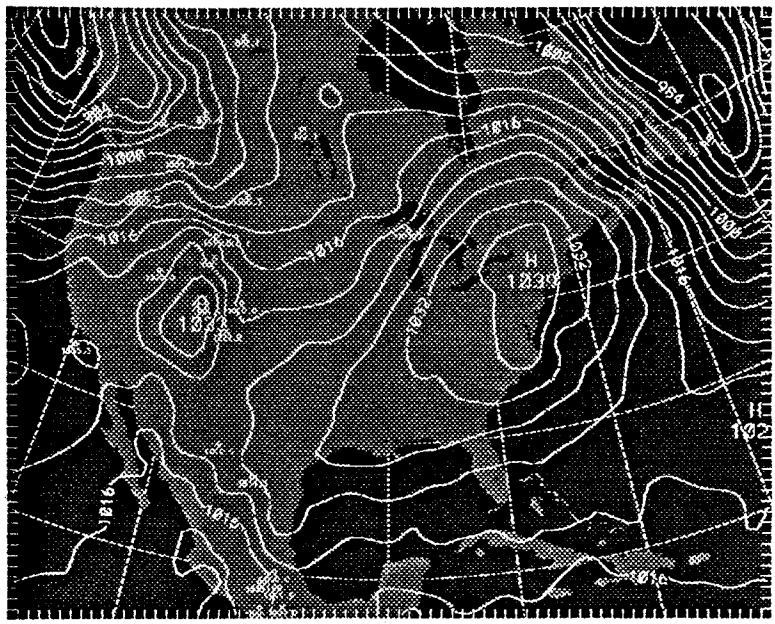


Figure 23. The auto bogus map for the sea level pressure analysis. The final analysis from RAWINS is the contoured field. Observations removed from the final analysis are reported.

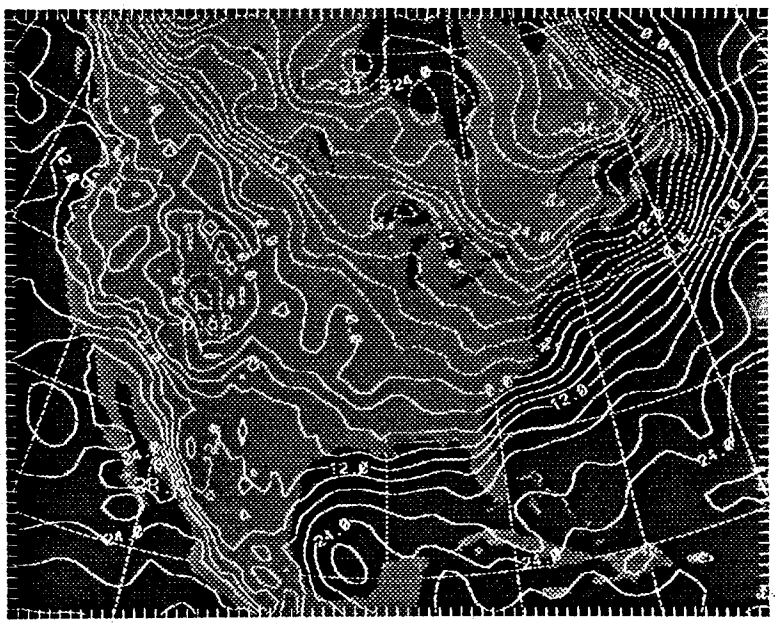


Figure 24. The auto bogus map for the surface temperature analysis. The final analysis from RAWINS is the contoured field. Observations removed from the final analysis are reported.

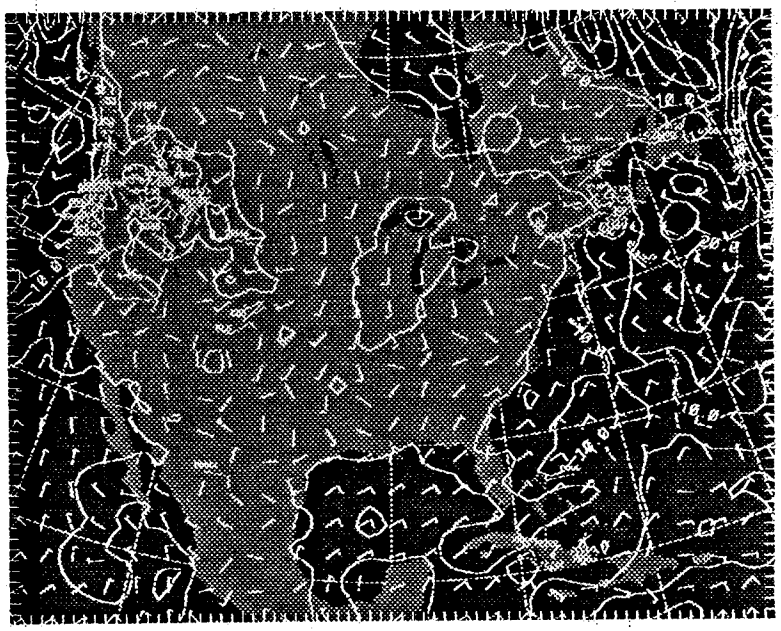


Figure 25. The auto bogus map for the u-component of the surface wind analysis. The final analysis from RAWINS is the regularly spaced wind barb field. Observations removed from the final analysis are the larger barbs. Clusters of flagged observations (in north-western US) indicate large-scale deficiencies in the first-guess field.

The auto bogus metacode is written to film (because of the color information) and to the MS. There are a few mandatory levels plotted along with the surface; the user may modify the source code to plot all the analysis levels. Because of the large amount of flagged observations at the surface, the individual wind components are handled separately. The only available plots are sea level pressure, surface temperature, surface wind (u- and v-components individually), upper-level temperature, and upper-level wind (u- and v-components together). The moisture is not part of the auto bogus option because an error maximum for the objective analysis of relative humidity is not used. The geopotential height is not objectively analyzed in RAWINS.

The user should experiment with the maximum error criteria values (this decides which observations to keep and which to remove). A modification as small as a couple of mb for pressure or a few °C for temperature can result in hundreds more observations (or less observations) being labeled as suspect. Having access to only a couple of observations does not allow the user any flexibility to tune the analysis; alternatively, forcing nearly every good observation back into analysis is not helpful either.

# Chapter 7 INTERP

The GRIN deck is comprised of the two programs INTERP and GRAPH. This functional setup is historical since the DATAFLOW program that preceded GRIN processed both $\sigma$-level and pressure-level data simultaneously, as well as generating metacode for both vertical coordinate systems. Preparing data as model input implies that both the pressure-level data (input) and the $\sigma$-level data (output) are available after the INTERP run. Following this program with GRAPH is efficient. Similarly, having the GRAPH script inside the GRIN deck to process the forecast data allows the user to ignore the temporary step of handling the created pressure-level model output file. Though the INTERP and GRAPH programs may accessed from the same shell, this section deals only with the INTERP program.

The INTERP program vertically interpolates data for subsequent processing. The pressure-level gridded output from DATAGRID and RAWINS are interpolated to the $\sigma$ coordinate when generating model input (a front-end submission). The $\sigma$-level model output is interpolated to pressure-levels for graphical display (the back-end submission). Appendix A.6 gives a description of the model input initial conditions and the model input boundary conditions, both of which are generated during a front-end GRIN job. Appendix A.8 deals with the interpolated model output format, which is generated during the back-end GRIN jobs.

During the front-end submittals (pre-processing, generating model input) the coarse-grid RAWINS data or the expanded-domain DATAGRID output file are used as input into INTERP. For each time period a surface pressure is computed, which is used to define the pressure at each grid point on every $\sigma$-level. Using these pressures, the wind fields and relative humidity are interpolated linear in $p$, and potential temperature is interpolated linear in $ln\ p$ (model input uses mixing ratio and temperature, but the ability to constrain relative humidity and the monotonicity of potential temperature are attractive). There is an option to remove the vertically integrated mean divergence from the coarse-grid wind fields. After the final form of the variables is available on the $\sigma$ surfaces for the time period, the data is written to the initial condition file and is used to produce the boundary condition file. The 2–D and coupled 3–D boundary tendencies of $p^*$, $p^*u$, $p^*v$, $p^*T$, $p^*q_v$ are computed as linearly interpolated temporal differences between the time periods being processed ($n$ time periods produce $n-1$ tendency periods). During the boundary condition generation, a daily mean ground temperature is computed (defined as the reservoir temperature in the mesoscale model).

The back-end submittal (post-processing the model output) interpolates either the coarse-grid or fine-grid model forecast data to pressure levels. While the front-end GRIN deck handles the coarse-grid and fine-grid domains simultaneously (the fine grid is generated internally during a front-end run), the back end can input only one of them. To generate interpolated data for both domains of the nested forecast run, the grin.deck shell must be run twice. The interpolation to pressure levels from the model $\sigma$ layers may require extrapolation at the surface and at the top of the model. Data in these areas

should be interpreted carefully:

- the temperature is extrapolated down using a standard lapse rate (6.5 K/km) from 100 mb above the surface to the pressure level 1001 mb, the potential temperature is interpolated linear in $ln\ p$ between the lowest $\sigma$ layer and the diagnosed value of $\theta$ from the extrapolated value of temperature at 1001 mb; above the top computational layer, $\theta$ is extrapolated linear in $ln\ p$
- the relative humidity at the lowest $\sigma$ layer is computed, the value underground is set to the lowest $\sigma$ layer value at that grid point; above the top computational layer, relative humidity is extrapolated linear in $p$
- the wind-components are vertically constant under the ground, they are set to the lowest $\sigma$ layer value at each $(i,j)$; above the top computational layer, the wind fields are extrapolated linear in $p$
- the geopotential height under ground is interpolated, since it is constrained by the height at sea level (defined as 0) and the height at the surface (defined as the terrain elevation); above the top computational layer, the geopotential height is extrapolated linear in $ln\ p$

The GRIN scripts, modification files, conversion deck, and conversion input file are located in the ~mesouser/Decks/Grin subdirectory on shavano. To run the grin.deck file, both the GRIN C-shell and the g_plots.tbl file (to choose which plots to generate in GRAPH) need to be modified. All of the modification files in the GRIN mesouser directory beginning with "i_" refer to INTERP, and all of the modification files in that directory with the prefix "g_" refer to GRAPH.

## 7.1 Local Input File

No more than three FORTRAN namelist records of the local input file need to be modified in the GRIN deck, and they all refer to the INTERP program. The first is `LOCMIF`, and must be modified for every run. The second is `NONSTAN` and needs to be modified only for non-standard uses of INTERP, such as generating one way nest files for the model or creating files for reanalyzing model output. The third namelist record, `VERSION7`, is used less frequently than either of the other two. Only INTERP jobs with Version 7 DATAGRID or Version 7 RAWINS need to use this namelist (Version 7 front-end data has no record header information). Users with Version 7 model output need to convert their data with the i_v72v8.deck C-shell script, located in the GRIN mesouser directory. A local copy of the record.header file must be modified to fit the model data characteristics (supply the information that would be contained in a standard Version 8 record header). Even though INTERP can handle Version 7 DATAGRID and Version 7 RAWINS (since there are parameterized dimensions), the GRAPH program needs to dynamically allocate space. The amount of space needed is computed from the stated size of the data in the record header. Version 7 data has no such header, therefore GRAPH can not use Version 7 DATAGRID, RAWINS, DATAFLOW (model input) or MM4 data.

Following are samples and descriptions of each of the three records of the local input file.

The local input file is contained inside the grin.deck file. Though not directly associated with the INTERP program or the grin.deck file, users with Version 7 model output need the record.header file. The record.header file is not contained inside the grin.deck file but is located separately on the ~mesouser/Decks/Grin directory. The user should archive the Version 8 model output generated with the i_v72v8.deck processor, as this skips a step the next time the Version 8 format of the forecast data set is required.

```
&LOCMIF ;--------   LOCAL MIF FOR PROGRAM INTERP ---------
;
;       FULL SIGMA LEVELS FOR FRONT END -- USUALLY REQUIRED
;       BOTTOM-UP (SURFACE TO PTOP) ORIENTATION
NEWCOORD=  1.00,0.99,0.98,0.96,0.93,0.89,0.85,   ; TYPICAL HI-RES PBL
           0.80,0.75,0.70,0.65,0.60,0.55,0.50,   ; 40-45 mb
                0.45,0.40,0.35,0.30,0.25,0.20,   ; VERTICAL COVERAGE
                0.15,0.10,0.05,0.00 ;            ; IN TROPOSPHERE
;
;
STANDARD=T,           ; IF FALSE, USE NAMELISTS BELOW
IPROCESS= $Split,     ; NUMBER OF FILES TO PROCESS
IFILINT =1,           ; FILE INTERVAL: 1=EVERY FILE
IBEGIN  =0,           ; START AT WHICH FILE (FRONT) OR XTIME (BACK)
NESTGD  =T,           ; CREATE A NESTED DOMAIN, TRUE/FALSE  (FRONT ONLY)
INEST   =18,          ; I LOCATION IN COARSE OF I=1 IN NEST (FRONT ONLY)
JNEST   =25,          ; J LOCATION IN COARSE OF J=1 IN NEST (FRONT ONLY)
& ;-------------------------------------------------------------------
```

The LOCMIF namelist record contains the switches most often modified for the INTERP program. Since the same program is used for front-end and back-end processing, some of the switches have different interpretations on different submittals, some are completely ignored. For the pre-processing INTERP run, the NEWCOORD refers to the full $\sigma$ levels that the model will use. These are in a bottom–up orientation, where the first value of NEWCOORD must be 1, and the last value of NEWCOORD is 0. These are used to compute the half layers to which the isobaric input data is interpolated. On a back-end job that is interpolating model output to pressure levels for graphical display (STANDARD = TRUE), the default pressure-level values used are contained in the record header information. Users do not need to remember the pressure levels that DATAGRID or RAWINS supplied. On a standard back-end job NEWCOORD is ignored.

What defines a standard front-end or back-end job for the INTERP program is the logical switch STANDARD. The easiest way to remember the setting for this flag is that STANDARD = TRUE unless the user needs to access either of the other two namelist records: NONSTAN or VERSION7. If STANDARD = FALSE, the program will read the additional namelist records. For users taking DATAGRID or RAWINS pressure-level data and generating the default $\sigma$-level model input, or for users reading model output and interpolating it to the default pressure-level data, STANDARD = TRUE. Users with Version 7 pressure-level data

from DATAGRID or RAWINS, special data generation requirements, or non-default switch settings must use STANDARD = FALSE.

The next three values allow the user to request the specific time periods to process in INTERP. The total number of time periods to process is IPROCESS (with the advent of moving nests and the duplicate times, users need to include the count of all duplicate times that will be processed for a given domain). The number of time periods for the INTERP program to process is defined through the shell variable expansion ($Split). To have each time period on a single fiche, the number of time periods is also given to the GRAPH program so that the metacode is split consistently. Users should not modify the IPROCESS flag without understanding the GRAPH shell. The interval for the processed files is IFILINT. For front-end jobs, IFILINT refers to the increment of time periods regardless of the temporal frequency. If IFILINT = 1, then consecutive time periods are used. If IFILINT = 2, then every other time period is processed (for front-end jobs, it does not make sense to choose to degrade the boundary conditions by not using consecutive time periods). For the back-end job submissions, this refers to the actual time interval (in hours) to process. If IFILINT = 2, then every available forecast time that is close to an increment of 2 hours is used. The IBEGIN switch tells the program at which time to begin processing. On the front-end jobs, IBEGIN refers to which time period index is the first to process. With IBEGIN = 1 or IBEGIN = 0 the first time period is used as the initiating point in the sequence to process. To start the processing with the third time period, set IBEGIN = 3. On the back end, IBEGIN refers to the hour of the forecast with which to begin the INTERP processing. These parameters are best described with examples:

Assume this is a front-end job with RAWINS data which is available every 12 hours for 72 hours. This is a total of 7 time periods of data. If the user wants every time period processed:

```
IPROCESS=7,     ; NUMBER OF FILES TO PROCESS
IFILINT =1,     ; FILE INTERVAL: 1=EVERY FILE
IBEGIN  =0,     ; START AT WHICH FILE (FRONT) OR XTIME (BACK)
```

Assume this is the same front-end job, but the user only wants hours 36 and 60 processed:

```
IPROCESS=2,     ; NUMBER OF FILES TO PROCESS
IFILINT =2,     ; FILE INTERVAL: 1=EVERY FILE
IBEGIN  =4,     ; START AT WHICH FILE (FRONT) OR XTIME (BACK)
```

Assume now this is a back-end job with model output data which is available every hour for 72 hours. This is a total of 73 time periods of data (0 – 72 hours). If the user wants every 12 hour time period processed:

```
IPROCESS=7,     ; NUMBER OF FILES TO PROCESS
IFILINT =12,    ; FILE INTERVAL: 1=EVERY FILE
IBEGIN  =0,     ; START AT WHICH FILE (FRONT) OR XTIME (BACK)
```

Assume this is the same back end job, but the user only wants every 6 hours processed between 30 hours and 48 hours:

```
IPROCESS=4,     ; NUMBER OF FILES TO PROCESS
IFILINT =6,     ; FILE INTERVAL: 1=EVERY FILE
IBEGIN  =30,    ; START AT WHICH FILE (FRONT) OR XTIME (BACK)
```

For standard runs, the remaining switches in LOCMIF are operational only during front-end job submittals. They describe if and where to locate a nested domain (identical to the method used in the master input file employing the lower left grid point). Even though the nested information is forwarded from the DATAGRID program through the record header into RAWINS, the information from the record header is discarded; the LOCMIF namelist values and nested parameter statements define the fine-grid domain. This allows the flexibility of creating model input from terrain elevation and nest locations other than that used during the initial analysis, without regenerating the analysis. Users attempting this enhanced terrain elevation and land use strategy should be wary concerning the consistency of the coarse-grid and fine-grid meshes.

The logical flag NESTGD is set to either true or false: NESTGD = TRUE creates the fine-grid initial condition, reads in a fine-grid terrain and land use data set, and generates a fine-grid reservoir temperature for the model boundary conditions; NESTGD = FALSE overrides any nested information carried in the record headers by generating only a coarse-grid initial condition. If the nested flag is true, then the location of the lower left point of the nest in the coarse domain needs to be defined (the requested size of the fine-grid domain comes through the parameter statements). The INEST and JNEST values refer to the I–direction and J–direction indexes of this point, respectively. These values must match the ICNS and JCNS settings in the master input file which generated the input fine-grid terrain data. Tracking down an incorrect fine-grid location specification would be obvious when plotting the terrain elevation against the map background. There are no checks in INTERP to insure that the user is placing subdomains correctly.

Users anticipating a nested-model run should plan ahead so that a consistent coarse-grid and fine-grid terrain data set may be generated during the first TERRAIN run. Mismatch between the coarse-grid and fine-grid terrain elevation files will cause mass flux oscillations between the coarse-grid and fine-grid domains during the forecast. Nests should be designed so as to remain at least 10 coarse-grid units away from the nearest boundary. For a fully two way interactive nest, the model requires a ratio of 3::1 for the coarse-grid size to the fine-grid size. Different integer ratios are available from INTERP, but are usually implemented to drive a one way nest. Modifications to the source code (the flag IRATIO) are required for a one way nest experiment.

```
&NONSTAN  ;----   ONLY REQUIRED FOR NON-STANDARD INTERP JOBS   ----
;
;           *** REPLACE DEFAULTS WITH THESE VALUES ***
IUSE    =0,           ; SUPPORTED NON STANDARD OPTIONS
                      ;    0 = INTERP USES NEWCOORD + REPLACE DEFAULTS
                      ;    1 = OUTPUT DATA IN SPECIAL FORMAT
IVERSION=8,           ; FRONT END ONLY, MM ASSUMED VERSION 8
                      ;    7 = ABSOLUTELY SURE THIS IS VERSION 7
                      ;        FRONT END ONLY.  USER NEEDS TO FILL IN
                      ;        VERSION7 NAMELIST
                      ;    8 = ASSUMES V8, BUT DETECTS V7 DATA, THEN
                      ;        USES V7 DATA FORMAT (ALSO THEN READS
                      ;        VERSION7 NAMELIST)
IDIV    =0,           ; FRONT END ONLY
                      ;    0 = REMOVE INTEGRATED MEAN DIVERGENCE
                      ;    1 = NO WIND MASSAGE, TYPICAL IF RUNNING INIT
                      ;        NEXT
MAKEDATA=2,           ; IF IUSE=1, THEN EXTRA DATA TO DISPOSE
                      ;    1 = MM AS FIRST GUESS TO RAWINS
                      ;    2 = NCAR VAX -- SIGMA
                      ;    3 = 1 WAY NEST
;
;           *** PARAMETERS FOR SPECIAL DATA SETS: ACTIVE IF IUSE=1 ***
ICUT1   =1,           ; I=1 LOCATION OF SUBDOMAIN IN BIG GRID
ICUT2   =49,          ; I=IMAX LOCATION OF SUBDOMAIN IN BIG GRID
JCUT1   =1,           ; J=1 LOCATION OF SUBDOMAIN IN BIG GRID
JCUT2   =61,          ; J=JMAX LOCATION OF SUBDOMAIN IN BIG GRID
& ;-------------------------------------------------------------------
```

The NONSTAN namelist record is used for any non-standard request for the INTERP program. There are two types of non-standard uses available: 1) modify the default values of some switches, and/or 2) output some special data sets. This namelist record is read by INTERP if the user set STANDARD = FALSE in the LOCMIF namelist record.

The default switches that the user may modify are IVERSION (front end only), IDIV (front end only), and NEWCOORD (back end only). If the user has Version 7 DATAGRID or Version 7 RAWINS data, IVERSION = 7 is required, otherwise, IVERSION = 8 should remain the default.

If the user is planning to run the vertical mode initialization program (INIT) prior to using the data for model input, then removing the integrated mean divergence is not recommended. The default is to remove (IDIV = 0) the vertically integrated mean divergence. Setting IDIV = 1 leaves the wind field initial conditions unaltered.

On the back-end GRIN run, NEWCOORD is ignored by default. If the user does not want to interpolate to every pressure level that came from the analysis (for example, if the analyzed data from RAWINS was available every 25 mb), to have the INTERP

program pay attention to the specified NEWCOORD values in the LOCMIF namelist record, set STANDARD = FALSE. With IUSE = 0 in the NONSTAN namelist record, the INTERP code will modify the choice of levels for the interpolated vertical coordinate.

The switches to allow the INTERP program to generate additional data sets are IUSE and MAKEDATA. If IUSE = 0, no non-standard data set is to be produced. If IUSE = 1, a special data set will be generated, and the type of data is defined by the MAKEDATA switch. When IUSE = 1 and MAKEDATA = 1 on a back-end submittal, the model output will be interpolated to mandatory pressure levels and stored in a DATAGRID output format. This allows the model data to be used directly for RAWINS input as the first-guess field. This type of special data is called using the model to generate the first guess. The forecast first-guess fields from the model are reanalyzed (in RAWINS) prior to being used as model input.

Another special type of data is for diagnostic purposes. If IUSE = 1 and MAKEDATA = 2, an ASCII file is created containing the input and interpolated data. This file can be quite large, but it transfers through networks fairly easily since it is a character data set. This file may be generated with either a front-end job (DATAGRID or RAWINS interpolated to model input) or a back-end job (model output interpolated to pressure levels).

The option to have model output turned directly into model input without any reanalysis or vertical coordinate interpolation is available. If the user sets IUSE = 1 and MAKEDATA = 3, the high temporal resolution model output can be used to drive the boundaries of a subsequent forecast (typically with a finer grid than the original). This set up is referred to as a one way nest, since the outer model forecast (high temporal frequency boundary conditions) is not affected by the fine-grid forecast. There are two separate forecasts that take place for both a first guess reanalysis and a one way nested run; both of these options in INTERP are only valid for back-end submissions.

When the four flags ICUT1, ICUT2, JCUT1, and JCUT2 are activated, they give the dot point locations of lower left and upper right grid points of the subdomain to output. When generating a first-guess field (IUSE = 1 and MAKEDATA = 1) or when generating a diagnostic data set (IUSE = 1 and MAKEDATA = 2), the four subdomain flags are active. When generating a one way nest file (IUSE = 1 and MAKEDATA = 3) that does not have a finer grid resolution than the original model output (NESTGD = FALSE), the four flags are active. When generating a one way nest file (IUSE = 1 and MAKEDATA = 3) that has a finer grid resolution than the original model output (NESTGD = TRUE), the subdomain is treated similar to nest. In this case, the nested parameter statements (IXN and JXN) are required as well as the beginning point in the nest (INEST and JNEST), but the four subdomain locations (ICUT1, ICUT2, JCUT1, and JCUT2) are not used.

The use of any of the non-standard options (first-guess reanalysis, diagnostic ASCII data generation, or one way nest) require additional modification files to the standard source deck. These are supported options since the source code modification files are available to all users, but they are not thoroughly tested.

```
&VERSION7 ;----   ONLY REQUIRED FOR VERSION 7 RAWINS OR DATAGRID  -----
;
                          ; EXAMPLE OF PRESSURE VERTICAL COORDINATE
                          ; TYPICAL OF RAWINS (MANDATORY + NEW PRES)
                          ; BOTTOM-UP (1000 mb TO PTOP) ORIENTATION
 OLDCOORD = 1000.,  950.,  900.,  850.,  800.,  700.,  600.,
             500.,  400.,  300.,  250.,  200.,  150.,  100.,
;
                          ; EXAMPLE OF PRESSURE VERTICAL COORDINATE
                          ; TYPICAL OF DATAGRID (MANDATORY ONLY!)
                          ; BOTTOM-UP (1000 mb TO PTOP) ORIENTATION
;OLDCOORD = 1000.,                  850.,          700.,
;            500.,  400.,  300.,  250.,  200.,  150.,  100.,
;
DS         = 160.E3       ; COARSE GRID DISTANCE IN METERS
PTOP       = 100.,        ; TOP OF MODEL IN mb
PROGINT    = 'RAWINS',    ; DATA FROM (6 CHAR): RAWINS, DATGRD
INTVLX     = 4,           ; SUBROUTINE OUTPT X DIRECTION PRINT INTERVAL
INTVLY     = 8,           ; SUBROUTINE OUTPT Y DIRECTION PRINT INTERVAL
KSIGDIG    = 5,           ; SUBROUTINE OUTPT NUMBER OF SIG DIGITS
IFILES     = 2,           ; NUMBER OF TOTAL DATA TIMES IN VOLUME
XLONC      = -90.,        ; CENTER LONGITUDE (WEST NEGATIVE)
PHIC       = 40.,         ; CENTER LATITUDE (SOUTH NEGATIVE)
IPOT       = F,           ; T/F THIS IS RAWINS ISENTROPIC DATA AS INPUT
IPROJ      = 6HLAMCON,    ; LAMCON, POLSTR, MERCAT
& ;-------------------------------------------------------------------
```

The VERSION7 namelist is only for allowing INTERP to use the Version 7 DATAGRID or Version 7 RAWINS files that still exist (this namelist does not have any effect on Version 7 model output which is handled by the i_v72v8.deck and record.header files). To input the Version 7 front-end data, the user must set STANDARD = FALSE in the LOCMIF namelist, and turn IVERSION = 7 in the NONSTAN namelist record. The values in the VERSION7 namelist look very similar to those in the master input file since the MIF values that are carried through the record headers in the Version 8 data are those missing in the Version 7 data. The parameters in the VERSION7 namelist should be set to the values that the user would need in the master input file if the front-end data was to be regenerated as Version 8.

The pressure-level vertical coordinate is stored in the OLDCOORD array. If the data that is input to the INTERP program is Version 7 DATAGRID, fill in only the mandatory pressure levels from 1000 mb up to PTOP (an example is commented out of this type of vertical coordinate), and set PROGINT = 'DATGRD'. If the data that is to be input into the INTERP program is Version 7 RAWINS data, the OLDCOORD array is a meshed list of mandatory and new pressure levels from 1000 mb up to PTOP, and set PROGINT = 'RAWINS'. In the OLDCOORD array, do not include the surface level (1001 mb), sea level (1013 mb), or snow cover level (1023 mb) for either the mandatory or the mandatory + new pressure levels.

```
&MM4STUF ;------ NAMELIST FILE FOR V72V8 PROGRAM
MIF     =  19,          ;  NUMBER OF DATA TIMES
           99,          ;  IX COARSE
           127,         ;  JX COARSE
           1,           ;  IXN NEST
           1,           ;  JXN NEST
           17,          ;  ICNS
           19,          ;  JCNS
           2,           ;  NSTTYP
           8,           ;  INY
           8,           ;  JXX
           5,           ;  KSIGT
           3,           ;  RATIO COARSE/FINE MESH
           4*999,       ;  IGNORE
           6HLAMCON,    ;  MAP PROJECTION
           13*999,      ;  IGNORE
MRF     =  30,          ;  GRID DISTANCE COARSE, KM
           38,          ;  CENTRAL LATITUDE
           -94,         ;  CENTRAL LONGITUDE
           10,          ;  GRID DISTANCE NEST, KM
           300,         ;  AEXP KM
           100,         ;  PTOP MB
           4*999.9,     ;  IGNORE
MLF     =  F,           ;  T/F: THIS WAS A NESTED RUN
           F,           ;  T/F: EXPANDED DOMAIN USED IN ANALYSIS
           F,           ;  T/F: THIS IS THE NESTED MESH
           F,           ;  T/F: THIS DATA WENT THROUGH INIT
           F,           ;  T/F: IGNORE
           F,           ;  T/F: IGNORE
           F,           ;  T/F: IGNORE
           F,           ;  T/F: IGNORE
           F,           ;  T/F: IGNORE
           F,           ;  T/F: IGNORE
NLV     = 14,           ;  NUMBER OF PRESSURE LEVELS USED IN RAWINS
P       = 1000, 950, 900, 850, 800, 700, 600, ; P LEVELS FROM RAWINS
          500, 400, 300, 250, 200, 150, 100, ; NO 1013 OR 1001
INFO    = 0,            ;  1=LOTS OF PRINT OUT, 0=SMALL PRINT OUT
& ;-------------------------------------------------------------------
```

The file record.header that contains the namelist MM4STUF is not used directly by the INTERP program. This namelist is read by the i_v72v8.deck file that converts Version 7 model output into the Version 8 format. Users wanting to convert Version 7 model output to the Version 8 format can not use the VERSION7 namelist record that is contained inside the grin.deck file since it is only for front-end data. If a call is made to the i_v72v8.deck inside the GRIN C-shell (activated by the shell option Version = 7), the user must put the modified copy of record.header (the MM4STUF namelist, located in the

~mesouser/Decks/Grin subdirectory on shavano) in the current directory for that GRIN run.

To fill in the three arrays MIF (integers), MRF (reals), and MLF (logicals), consult Appendix A.7 for a description of those values. These arrays go directly into the record header of the generated Version 8 model output. The next two switches deal with letting the model output carry along the pressure levels from which it was generated. In this example, the data was from RAWINS since the pressure levels include both mandatory as well as new pressure levels. The pressure levels must go from 1000 mb up to PTOP, but do not include the surface (1001 mb), sea level (1013 mb), or snow cover levels (1023 mb). The pressure levels are included in the record header from model output so that users do not need to specify the levels to have the GRAPH program process. These isobaric levels represent what the user would have as the default values for the vertical coordinate for the interpolation and plotting packages. For the standard INTERP run, the arbitrary choice that the user makes for NLV in the record.header file must be equal to KXP in the grin.deck parameter statements.

The i_v72v8.deck file may be run either autonomously or from inside the GRIN deck. Following is an example of using the shell outside of grin.deck file when there are multiple MS volume names for the time periods covering the forecast:

```
cd $TMPDIR
msread MyDataA /GILL/SOME_EXP/MM4OUTA_V7
msread MyDataB /GILL/SOME_EXP/MM4OUTB_V7
cp ~mesouser/Decks/Grin/i_v72v8.deck .
chmod +x i_v72v8.deck
cp ~mesouser/Decks/Grin/record.header .
vi record.header
i_v72v8.deck MyData
```

Even when there is only one MS volume name for the forecast time, the capital letter "A" needs to appended to the local copy of the file:

```
cd $TMPDIR
msread MyDataA /GILL/SOME_EXP/MM4OUT_V7
cp ~mesouser/Decks/Grin/i_v72v8.deck .
chmod +x i_v72v8.deck
cp ~mesouser/Decks/Grin/record.header .
vi record.header
i_v72v8.deck MyData
```

In both cases, the local copy of the original data file has been modified on the Cray disk. After the i_v72v8.deck file completes, the files MyDataA (and MyDataB in the first example) are now Version 8 model output, and the Version 7 files are not in the directory (they still reside on the MS). As was mentioned before, writing these local copies of the Version 8 files directly to the MS is a good idea (with a different bit file name), to avoid modifying the record.header file each time the Version 7 model data is to be input to any of

the post-processing programs. A copy of the record.header file must reside in the current working directory so that the conversion program may access it.

## 7.2 Parameterized Dimensions

The INTERP program has several types of parameter statements for the different functions that the program may perform. The horizontal dimensions, whether for the coarse domain, fine grid, or the expanded domain, begin with the letter "I" or "J". The vertical dimensions, whether referring to $\sigma$, pressure, or $\theta$-level data, begin with the letter "K". The update modifications to the parameter statements are inside the grin.deck file. The user should set any extraneous parameter values to 1; all are used to allocate space, and a zero length array would generate an error.

```
*D PARAM.2,9
      PARAMETER (IX   = 46, JX   =  61) !  COARSE DOMAIN SIZE
      PARAMETER (IEXP = 58, JEXP =  73) !  EXPANDED, REQUIRED DATAGRID
      PARAMETER (IXN  = 31, JXN  =  31) !  NESTED, FRONT END
      PARAMETER (IMAX = 46, JMAX =  61) !  MAXO(IX,IXN), NOT EXPANDED
      PARAMETER (KXS  = 23)  !  VERT COORD: # HALF SIGMA LAYERS
      PARAMETER (KXP  = 21)  !  VERT COORD: # P LVL, 1000->PTOP, NOT SFC
      PARAMETER (KXT  =  1)  !  VERT COORD: # THETA LEVELS (MAXIMUM)
      PARAMETER (KMAX = 23)  !  MAXO(KXS,KXP,KXT)
```

On the front end, the user may input either DATAGRID or RAWINS data to generate model input. The IX and JX values are the coarse-grid size. If the input data is from DATAGRID, the user must specify the expanded-domain size IEXP and JEXP. If the front-end data (DATAGRID or RAWINS) is used to generate both a coarse-grid and a fine-grid model input, the nested domain must also be parameterized. The dimensions IXN and JXN are the I–direction and J–direction dot point size of the fine grid, respectively. The last two horizontal parameters, functionally defined as IMAX=MAXO(IX,IXN) and JMAX=MAXO(JX,JXN), provide space allocation internal to the program. The intrinsic FORTRAN function to return the maximum of two integer values will not work in a parameter statement, this is used to elucidate how the value should be computed.

For back-end job submittals (and front-end jobs with RAWINS input) the expanded parameter statements (IEXP and JEXP) are ignored. For standard back-end INTERP runs the nested parameter statements (IXN and JXN) are ignored. If the fine-grid model output data set is used as input into the INTERP program, set the IX and JX values to the fine-grid size. If the coarse-grid of the model output is read by INTERP, then the parameters IX and JX refer to the coarse-grid size. For a back-end INTERP run, IX and JX refer to the horizontal size of the grid actually being used as input, regardless of the coarse-domain / fine-domain distinction.

There are four vertical coordinate parameter statements: the first three relate to specific coordinate systems, the last is the maximum value of the previous three. These values have the same interpretation on both the front-end and back-end jobs. The parameter KXS is the number of half $\sigma$ layers in the data set. This should be the same as

the number of vertical levels parameterized in the model. The value KXP is the number of mandatory + new pressure levels, from 1000 mb up to PTOP, not including the surface level (1001 mb), sea level (1013 mb), or snow cover level (1023 mb). The dimension KXT is the maximum number of $\theta$ surfaces in the incoming analyses (it may vary between time periods). Since only two of these vertical coordinates are active in INTERP at any time, the other vertical coordinate parameter (usually KXT) is set to 1. To provide space for internal memory management, the maximum size of the vertical coordinate is required (KMAX=MAX0(KXS,MAX0(KXP,KXT))). This is not a legal FORTRAN construct, but only for descriptive purposes.

## 7.3 Generated Files

During a standard front-end run, the INTERP program needs to generate model input (both initial conditions and boundary conditions). If this is a nested-model run, both a coarse-grid initial condition and a fine-grid initial condition are generated. Since there is no fine-grid pressure-level data generated by the RAWINS program, INTERP also handles this task. The fine-grid RAWINS data set is output specifically for the GRAPH program, but is archived to the MS. If the FDDA option was used in the RAWINS program, and a coarse-grid surface file for FDDA was generated, the INTERP program attempts to create a fine grid surface FDDA file also.

For a standard back-end run, model output is interpolated to pressure levels for graphical display (this file is not archived). Only one domain at a time is processed. INTERP must be run once for the coarse-grid and once for the fine-grid domain, if metacode from both is required. There is no metacode generated directly by the INTERP program during either front-end or back-end job submissions.

## 7.4 Hints and Caveats

All horizontal interpolation in INTERP is done with linear interpolations of overlapping quadratic fits between grid points. When trying to generate a consistent fine-grid surface field with this interpolation, errors are pronounced in areas where the value of the interpolated terrain elevation differs widely from the actual values used as input. On the front end, the fine-grid surface temperatures are modified with a standard lapse rate according to this change in height.

All vertical interpolations are linear in $p$ or linear in $ln\ p$. Near the surface, where there are typically several $\sigma$ layers packed together tightly, it is possible for neighboring points on the front end to be generated with different bracketing pressure levels, particularly when the surface pressure is near one of the analyzed levels from DATAGRID or RAWINS. Since there is no feedback of the surface analysis to the upper air analysis (except for a super adiabatic lapse rate removal in RAWINS), the surface level and upper air isobaric levels are not necessarily consistent. This does cause cosmetic problems in the lowest levels of temperature and moisture in the model initial conditions. The effect it has on the reservoir temperature has not been investigated. A solution might be to either increase the order of the interpolation scheme in the lowest several $\sigma$ levels, or to force the inclusion of the

surface value into the computation of fields in the lowest few $\sigma$ levels.

The record.header file is for Version 7 forecast output. This data needs to be converted to be used as input to any of the post-processing programs.

Version 7 DATAGRID and Version 7 RAWINS may be used as input to INTERP, this requires the `NONSTAN` and `VERSION7` namelist records in the INTERP local input file. INTERP does not convert Version 7 front-end data to Version 8 DATAGRID or RAWINS, it accepts Version 7 format front-end data as input.

# Chapter 8 GRAPH

The GRIN deck is comprised of the programs INTERP and GRAPH. This functional setup is historical since the DATAFLOW program that preceded GRIN processed both $\sigma$-level and pressure-level data simultaneously (generating metacode for both vertical coordinate systems). Preparing data as model input implies that both the pressure-level data (input) and the $\sigma$-level data (output) are available after the INTERP run. Sequentially following INTERP with GRAPH is efficient. Similarly, having the GRAPH script inside the GRIN deck to process the forecast data allows the user to ignore the temporary step of handling the created pressure-level model output file. Though the INTERP and GRAPH programs are accessed from the same shell, this section deals only with the GRAPH program.

The GRAPH program computes diagnostics on $\sigma$, pressure, and $\theta$ vertical coordinates systems. GRAPH displays the figures as horizontal contour plots, cross section contour plots, or skew–t plots. Horizontal plots of the data are displayed on the same vertical coordinate as the data ($\sigma$-level data may be interpolated to isentropic surfaces). There is an amount of user interface available to tune the metacode, but the GRAPH program has two primary functions: the program provides a way to quickly look at the modeling system data, and the program provides a method to generate extensive amounts of metacode.

The grin.deck C-shell script, the graph.deck C-shell script, GRAPH program modification files, plot selection table file, and several plotting options files are located in the ~mesouser/Decks/Grin subdirectory on shavano. To run the grin.deck file, both the GRIN C-shell and the g_plots.tbl need to be modified; to run the graph.deck C-shell interactively, only the g_plots.tbl file needs to be modified. All of the modification files (those files matching the "*mods" mask) in the GRIN mesouser directory beginning with "i_" refer to INTERP, and all of the modification files in that directory with the prefix "g_" refer to GRAPH. The user need not generate a GRAPH executable from the modification files unless personal changes are required in the code.

The distinction between the grin.deck and the graph.deck files is important. The grin.deck file is the job deck that runs both the INTERP program and the GRAPH program, it acquires data sets from the MS and returns any created files for archiving, it also disposes the generated metacode files to devices for film processing. The graph.deck file is an interactive script that only runs the GRAPH program, it is the C-shell that is called by the grin.deck file to run the GRAPH executable. The graph.deck shell does not acquire or archive data volumes, nor does it dispose metacode to plotting devices; these tasks are handled directly by the grin.deck file. To generate the standard battery of fiche for the analysis or the model forecast, run the grin.deck file.

## 8.1 Local Input File

The GRAPH program has several input streams. It accepts upto 20 input data volumes. There are table files of information for the generation of the map, the choice of

colors, the type of contouring. The graph.deck shell manufactures a file to input the choice for metacode editing to GRAPH. There are two additional input files for the graph.deck file. The g_plots.tbl file is the required input file that details the sequence of plots to generate. The g_defaults.nml is an optional file of the available plotting defaults that the user may modify. A description of these input files for GRAPH is in Appendix B.6, along with a list of the available diagnostic fields. Neither of these input files is located in the grin.deck file or in the graph.deck file. They are separate files that are located in the ~mesouser/Decks/Grin subdirectory. To use them the user must have a local copy of the input files in the current working directory.

## 8.2 Parameterized Dimensions

The GRAPH program was designed to be an executable that users access, not a source code that is compiled and linked. To have the program allocate space dynamically requires that the data size be computable. The record headers for Version 8 are a necessity (any data without a Version 8 record header can not be used as input to GRAPH). The GRAPH program reads in the first record of the data set and determines the horizontal and vertical size of the arrays, then allocates space for several 3–D and 2–D arrays. As individual subroutines are entered, additional space is placed on the stack, and subsequently discarded on returning from the subroutine.

## 8.3 Generated Files

The only permanent files generated from the GRAPH program are metacode files. All of the metacode that is generated is contained in the "gmeta" file. This file is split into pieces (in a standard GRIN run, this is the `NumSplit` flag). If the gmeta file is split into three pieces, there are three files created: gmeta.split1, gmeta.split2, gmeta.split3. The graph.deck file does the metafile editing (see the man page for "med").

The grin.deck file handles sending these metafiles to TAGS. The default tables that are accessed by the GRIN C-shell generate black and white figures, with simple outlines for the map background. If the user modifies either of these tables (g_map.tbl or g_color.tbl) to include color information, the metacode should not be sent to TAGS as a fiche job, but as a film job.

## 8.4 Hints and Caveats

For users with reasonable metacode transfer rates from shavano to their local (preferably Unix) workstation, the graph.deck file provides an additional capability. The graph.deck file is the C-shell called from the GRIN deck, but the user may run this executable script interactively. This is helpful for a quick look at some data, or for tuning plots for generating slides, or for a series of time snapshots of single fields during the forecast or analysis. Since the graph.deck call from the grin.deck is made without any additional user supplied information, the user should see how to choose this more interactive approach to GRAPH.

- To use the GRAPH program without the rest of the grin.deck file, the first step is to acquire the data from the MS. Since these requests typically take a while to be processed, place them in the background (there are files to edit in the interim). The user must modify the g_plots.tbl to select which diagnostics to plot, which times and vertical levels, contour intervals, and which units are to be used. The g_defaults.nml file is optional. This file permits the GRAPH program to plot a subdomain, modify the contour lines (color, line width, dash pattern), and print out the record header information when debugging. The plot selection and the defaults table are both located in the ~mesouser/Decks/Grin subdirectory on shavano.

```
cd $TMPDIR
# get the data, put the request in the background
msread  MyDataA /GILL/SOME_EXP/MM4OUTA &
msread  MyDataB /GILL/SOME_EXP/MM4OUTB &
# this is the REQUIRED plot request table file, modify it
cp ~mesouser/Decks/Grin/g_plots.tbl .
vi g_plots.tbl
# this is the OPTIONAL modifiable defaults file, edit it too
cp ~mesouser/Decks/Grin/g_defaults.nml .
vi g_defaults.nml
# bring over a copy of GRAPH C-shell executable
cp ~mesouser/Decks/Grin/graph.deck .
chmod +x graph.deck
```

- The graph.deck file may be given input information on the command line, or it will prompt the user. There are three types of information that the C-shell needs: 1) into how many pieces is the metacode to be split after the program completes, 2) how many data input volumes are there, and 3) the data volume name(s). Since there are several ways to handle the data volume naming input, the example below shows three ways the information can be given to the GRAPH shell for the above two data volumes when the generated metacode file is to be split into six pieces. After each of the three examples below, the generated metacode files are the same.

```
# example 1
graph.deck 6 2 MyData

# example 2
graph.deck 6 2 MyDataA MyDataB

# example 3
graph.deck
6
2
MyData
```

Both the skew–t and the cross section plots for the pressure level data need a value for surface pressure. This is not a standard field in the DATAGRID or RAWINS data set, and must be computed. Prior to any user request for pressure level cross sections or pressure level skew–t plots, the GRAPH code needs to plot the surface pressure. Similarly, any diagnostic request for DATAGRID or RAWINS that needs the surface pressure as input (potential temperature, for example), will print out a statement saying `FIELD PSFC NOT FOUND`. The user needs to plot the surface pressure prior to this computation.

The user may split the metacode into as many pieces as desired. The naming convention for the temporary metacode names that are assembled together before being disposed to fiche uses a numbering system. When more than nine time periods are generated, the fiche labels are incorrect, though all of the data is represented.

The GRAPH program accepts a 10 digit MDATE (YYMMDDHHmm). This allows users to specify intervals in minutes, not hours, in the g_plots.tbl file. The choice of the 10 digit MDATE format versus the 8 digit MDATE format (YYMMDDHH) is only required on model output where the output frequency may be less than an hour. To request every 3 hours from 12Z 29 March 1991 through 00Z 30 March 1991, both of the following is valid:

```
TIME LEVELS (MDATE FORMAT YYMMDDHH): FROM 91032912 TO 91033000 BY 3
TIME LEVELS (MDATE FORMAT YYMMDDHH): FROM 9103291200 TO 9103300000 BY 180
```

To request plots every 30 minutes from 12Z 29 March 1991 through 15Z 29 March 1991, the following is an available syntax to use in the g_plots.tbl file:

```
TIME LEVELS (MDATE FORMAT YYMMDDHH): FROM 9103291200 TO 9103291500 BY 30
```

The method to deal with data at a higher frequency than hourly is not as well established for the INTERP program. Users need to experiment with GRIN jobs prior to submitting massive fiche runs to insure that the number of encountered times is consistent with the number of processed periods.

The GRAPH program uses the information in the record header from the input file to allocate space for the program execution. This requires that only data with a Version 8 header can be used as input to GRAPH. Version 7 model output that is converted to Version 8 format (through the i_v72v8.deck file) is eligible for GRAPH processing. Version 7 DATAGRID and Version 7 RAWINS can not be processed with GRAPH.

## 8.5 Sample Metacode

On the next several pages are examples of some of the plots available from the GRAPH program. The plots are from the bench-mark case, and contain figures from the initial conditions and the 12 hour forecast. The GRAPH program allows overlayed figures, dashed line patterns, filled contours, color information, and vertical sections. The following figures are figures generated using the GRAPH program defaults with no overlayed fields (for reproduction clarity).

The bench-mark case has a 46 $\times$ 61 horizontal domain, with 90 km grid spacing. The

following plots from the initial conditions are the model input file (the $\sigma$-level analyses), and the forecast plots are generated with the $\sigma$-level model output. The following plots have the same geographic domain as figure 8. Figure 26 is a depiction of the bench-mark domain detailing the terrain contours. The model requires the normal boundary gradient of elevation set to zero. Figures 27a and 27b show the horizontal plots of sea level pressure for the initial conditions and the 12 hour forecast, respectively. Figures 28a and 28b show the horizontal plots of temperature (just above the boundary layer) for the initial conditions and the 12 hour forecast, respectively. Figures 29a and 29b show the skew-t plots of Altus AFB (OK) for the initial conditions and the 12 hour forecast, respectively.

The horizontal plots generated with GRAPH have several pieces of information important to each figure. The vertical coordinate and the plotted vertical level are displayed. The field name and the chosen field units are at the top-center of plot. The validation date of the plot is either the MDATE (for analyses) or the MDATE + forecast time (for the model output). The number of smoothing passes for the horizontal plot is printed.

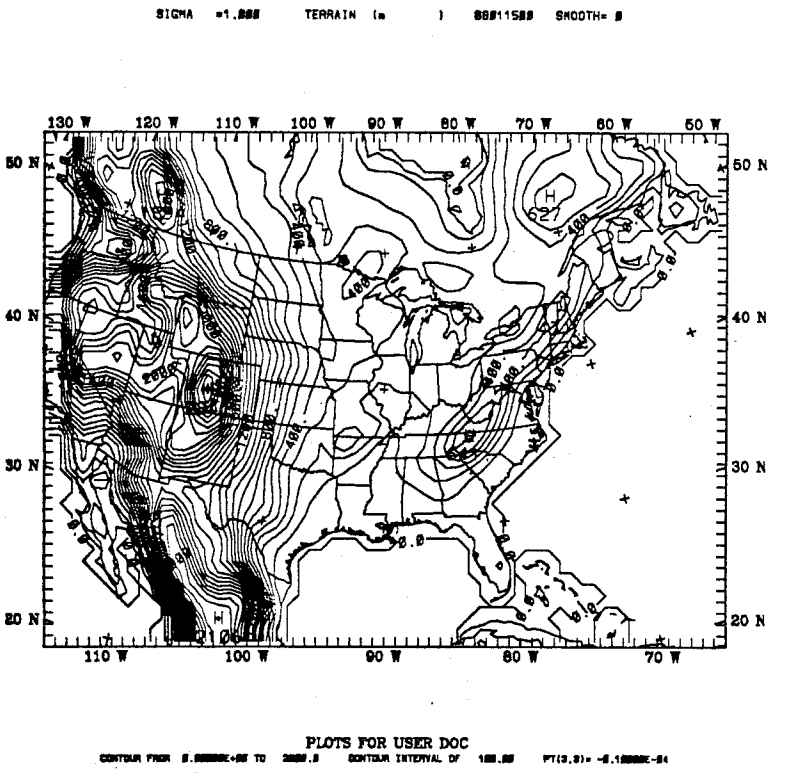



Figure 26. The terrain elevation for the bench-mark domain. Contours depict 100 m intervals. The information at the top of the plot provides that the data is on SIGMA coordinates, the plotted field is the terrain (m), the date is 00Z 15 January 1988 (88011500), and the field has not been smoothed. Below the figure, the user's plot title appears.

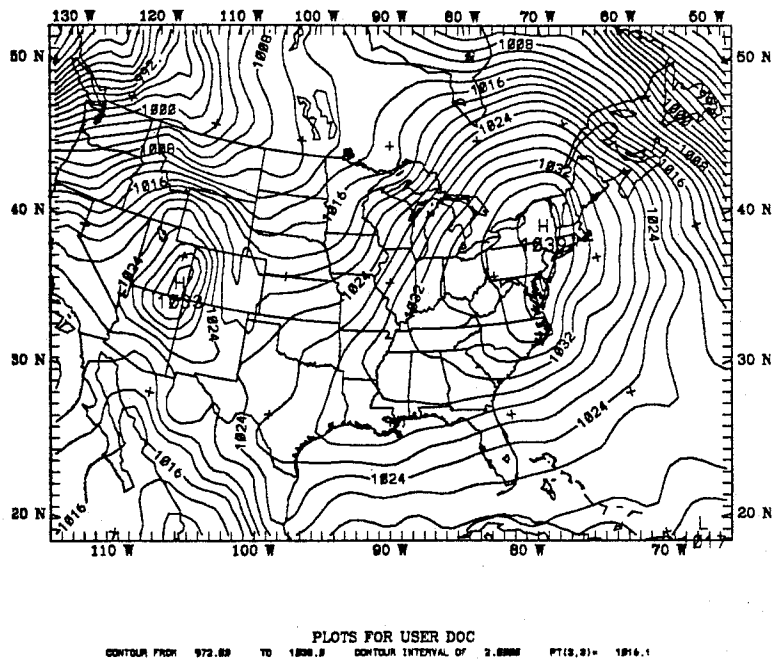



Figure 27a. The sea level pressure map from the initial conditions of the bench-mark case, valid 00Z 15 January 1988. Contours depict 2 mb intervals.

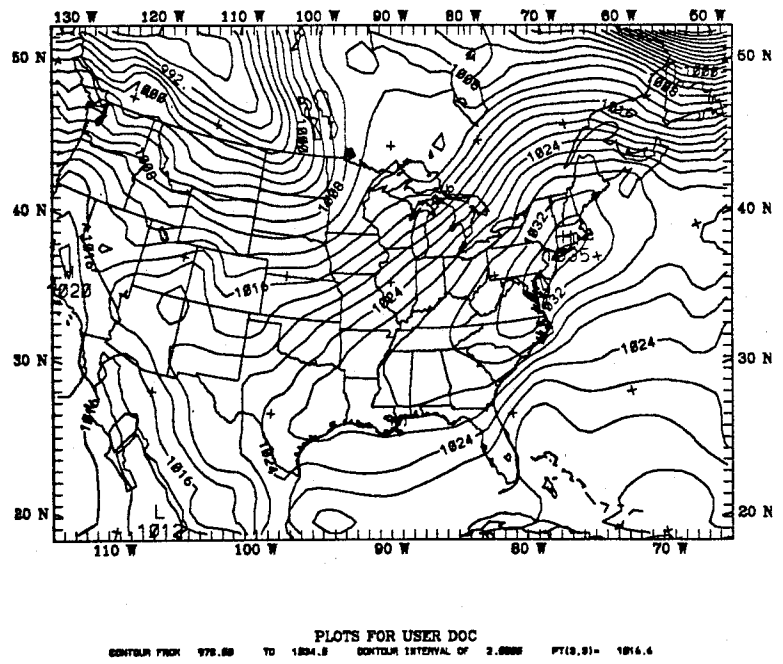



Figure 27b. The sea level pressure map from the 12 hour forecast of the bench-mark case, valid 12Z 15 January 1988. Contours depict 2 mb intervals.

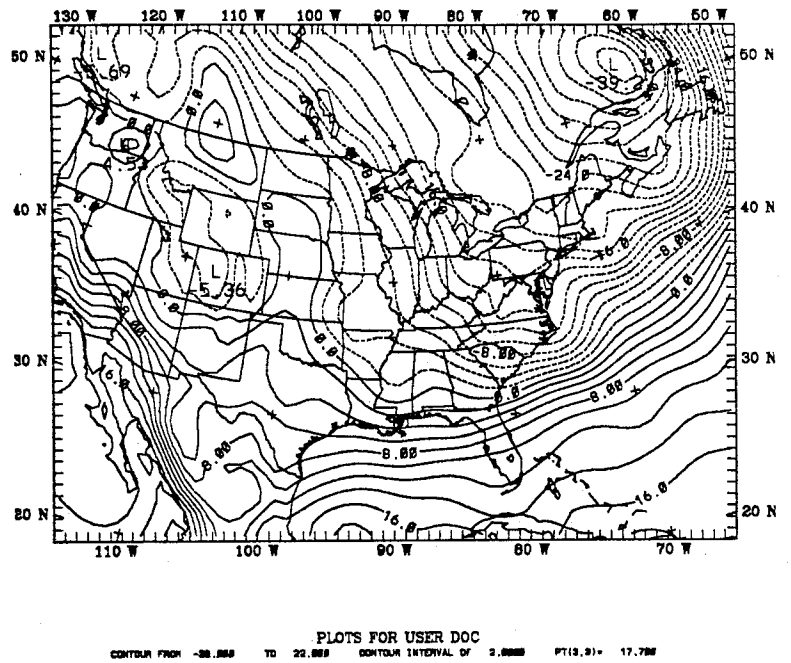



Figure 28a. The temperature map (above the boundary layer, $\sigma = 0.87$) from the initial conditions of the bench-mark case, valid 00Z 15 January 1988. Contours depict 2 °C intervals, with dashed lines representing temperatures below 0 °C.

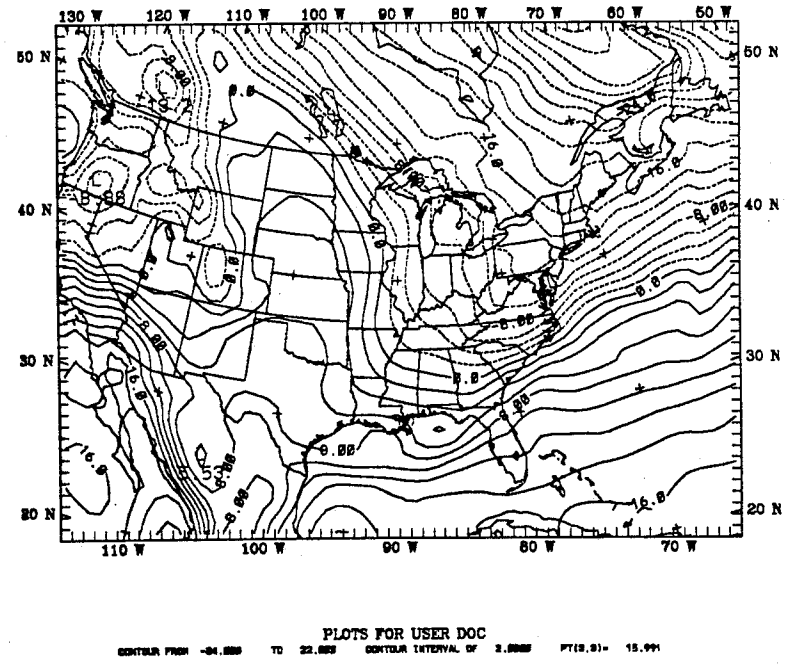



Figure 28b. The temperature map (above the boundary layer, $\sigma = 0.87$) from the 12 hour forecast of the bench-mark case, valid 12Z 15 January 1988. Contours depict 2 °C intervals, with dashed lines representing temperatures below 0 °C.

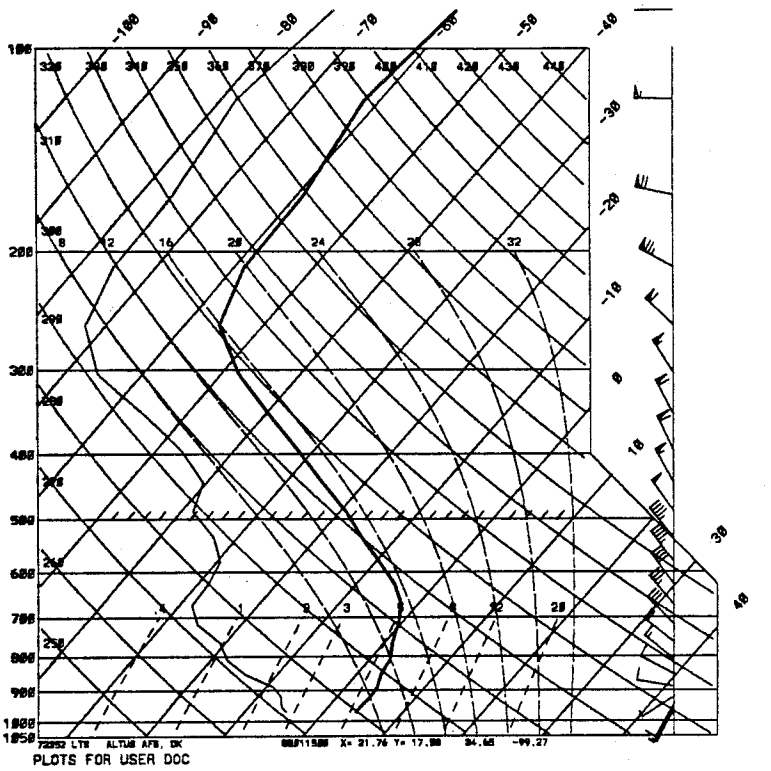



Figure 29a. The skew-t plot of Altus AFB, OK from the initial conditions of the bench-mark case, valid 00Z 15 January 1988. The wind speeds indicate 50 kts for a flag, 10 kts for a full barb, and 5 kts for a half barb. The latitude - longitude and (x,y) location of the sounding in the domain are given.

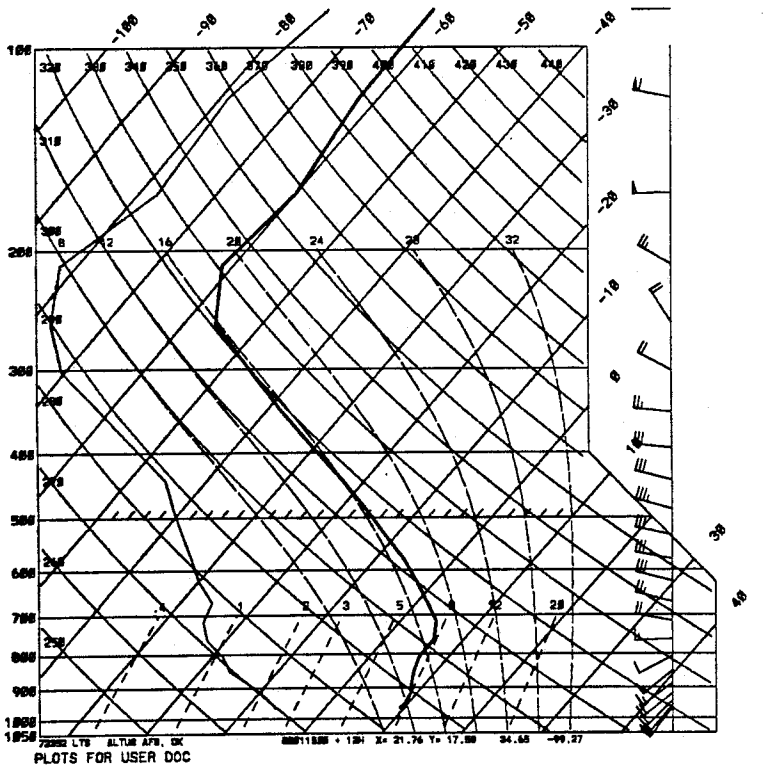



Figure 29b. The skew-t plot of Altus AFB, OK from the 12 hour forecast of the bench-mark case, valid 12Z 15 January 1988. The wind speeds indicate 50 kts for a flag, 10 kts for a full barb, and 5 kts for a half barb. The latitude - longitude and (x,y) location of the sounding in the domain are given.

# Chapter 9 MM4

The predictive component of the modeling system is the MM4 program. The MM4 job deck generates a forecast over the domain specified by the coarse-grid domain, and optionally with a finer mesh over the requested nested-grid subdomain. The same vertical coordinate spacing defined in the INTERP program is used by the model. Because MM4 is a regional model, the forecast may only extend out in time as far as the boundary conditions permit.

The MM4 scripts and modification files are located in the ~mesouser/Decks/MM4 subdirectory on shavano. For the user to run the MM4 program, only the model C-shell, mm4.deck, needs to be modified. Appendix A.7 gives a specific description of the format and content of the MM4 output file and record header (Appendix A.7 contains the formats of the experimental versions of the predictive model).

## 9.1 Local Input File

As with the other programs in the modeling system package, the local input file is contained inside the mm4.deck C-shell file. There are four separate records for the MM4 local input file. The `OPARAM` namelist controls the output information, the `LPARAM` record is responsible for modifying the physics in all of the domains, the `PPARAM` record handles the user tunable parameterizations, and the `FPARAM` is used for the FDDA information.

There are switches in the output namelist section that will rarely require user modification, and the FDDA namelist is only needed when nudging the forecast to observations and/or an analysis. There are no parameterization knobs that are modified frequently except for the forecast length (the duration of the forecast is located in the `PPARAM` namelist). The switches for the different physical options are modified quite frequently as case studies are put through various sensitivity experiments.

The options not associated with FDDA are described in Hsie (1987). Discrepancies exist between that documentation and the current namelists only by addition or omission of some switches (for example, the record header information has nullified the need to directly input the vertical coordinate, grid distance, and model lid values).

As there is no extant user level information on the MM4 version 8 job deck C-shell, this section will specifically address those areas in the local input file which have been modified from Version 7 MM4 job deck. There has been no redefinition of a variable's meaning or use inside the model. Following is a list of the four MM4 Version 8 local input file namelists, and a brief description of the variables.

```
 &OPARAM
;-----------------------------------------------------------------------
;       YOU CAN REMOVE THE UNWANTED DATA FROM THE FOLLOWING LISTING
;       AND USE THE DEFAULT VALUES DEFINED IN SUBROUTINE 'PARAM'.
;-----------------------------------------------------------------------
 RFSTRT = F,         ;REFINED START
 IFREST = F,         ;RESTART
    KTAUR  = 0,
 IFSAVE = T,         ;SAVE DATA FOR RESTART
    SAVFRQ = 360,
    NTSAVE = 0,
 ENDCYC = T,         ;SAVE DATA AT THE END OF THIS JOB
 NESTGD = F,         ;TWO-WAY INTERACTION NESTED GRID
 IFTAPE = 1,         ;OUTPUT FOR GRIN, VERIFY, TRAJEC
    TAPFRQ = 60,
    NTTAPE = 0,
 IFFILT = 0,         ;TIME FILTER
    NDTFIL = 5,
    NTFILT = 0,
 IFPRT  = 0,         ;PRINTER OUTPUT
    PRTFRQ = 720,
 IFTRAJ = 0,         ;OUTPUT FOR TRAJECTORY
    TRAFRQ = 60,
    NTTRAJ = 0,
 IFOMG  = 0,         ;OMEGA OUTPUT
    OMGFRQ = 60,
    NTOMG  = 0,
 IFRAIN = 0,         ;RAINFALL OUTPUT
    RAIFRQ = 60,
    NTRAIN = 0,
 IFPOIN = 0,         ;TIME SERIES OUTPUT
    NDTPOI = 5 ,
    NTPOIN = 0,
 MASCHK = 10,        ;MASS CONSERVATION CHECK
 PBLCHK = F,         ;PBL CHECK
    PBLFRQ = 120,
 ISFOUT = 0,         ;PRINT SURFACE PARAMETERS
 ISKF=0,             ;NUMBER OF TIME PERIODS TO SKIP IN IC
 ISKB=0,             ;NUMBER OF TIME PERIODS TO SKIP IN BC
 &END ;-------------------------------------------------------------------
```

To help the model quickly settle, the user may select the refined start option RFSTRT. This causes the model to use a smaller time step during the initial steps of the forecast. The results are twofold: an initial data imbalance that might cause some instability is checked due to the decreased time step, and the actual forecast period that is degraded with the initial time steps is reduced.

A data set comprised of two successive time steps of model data can be written incrementally to the disk by the program and saved to the MS by the C-shell. This allows the model to pick up where it stopped during a previous forecast integration, and is therefore called a restart. The model is often run as a restart, particularly during expensive integrations. The restart (or save) data is both input to the model (during a restart) and output from the model (when generating the save files). This capability allows the user to submit pieces of the intended forecast to the Cray, and to monitor the results after each portion of the forecast completes. If the user is running a restart (`IFREST = T`), the initial conditions for the model need to be the output save files (both coarse-grid and fine-grid domains). Remember to modify the name of the output save files so that they are distinct from the input restart file names generated during the initial model run. The `KTAUR` value is printed in the output listing of the previous model run after each dump of the save data. It is a good idea to always save the restart files at a frequency that is not prohibitively expensive to regenerate should they need to be used. The `ENDCYC` writes an additional save data set of the last two time steps after the program successfully exits.

The `NESTGD` switch informs the model that there will be a nested grid. This is a logical flag that is used for the two way interacting nests. Even if fine-grid initial conditions are pulled onto the disk and the fine-grid parameter statements are defined to the correct size, there is no nested-domain forecast integrated until `NESTGD = T` is set in this namelist. As a rule of thumb, activating the fine grid costs 4 $\times$ the CPU time of the coarse-grid domain and doubles the memory.

The `OPARAM` namelist controls most of the data set output routines. This information is typically in three parts: the logical flag the user sets to decide whether or not to output this data set, the frequency in minutes at which to write the data, and a switch to allow continuous data sets from multiple model restarts (users are encouraged to not attempt the continuous data sets from separate model runs, most of the post-processors will not accept data in this format).

The standard forecast data set that is input for the GRIN, VERIFY, and TRAJEC programs is controlled by the `IFTAPE` flag. There is no reason to not have this flag turned on during a forecast. The frequency to write the data is hourly by default, `TAPFRQ = 60`, but may be set to any integer time period in minutes. As extreme examples, forecast data available at 1 minute intervals will incur huge I/O costs, and data sets with a frequency of 12 hours provide little temporal continuity for diagnostic purposes. Users with 10 km grid spacing certainly require higher output frequency when compared to experiments where users choose forecasts with 200 km grid distances. The coarse-grid and fine-grid data sets are output at the same frequency, but to separate FORTRAN logical unit numbers. Both the save data sets and these history data sets have been structured in the code to flow over to additional volumes should the amount of data per file become larger than a prespecified size. The additional save data sets and history files generated in this fashion are written to the MS by the MM4 C-shell job deck. None of the other generated data sets of the forecast data are incremented by this method.

There are several available output routines that generate reformatted subsets of the standard history files. The `IFFILT` switch outputs most of the information from the history

files in a j–slice format at comparably high temporal frequency. This file is appropriate for spectral decomposition by subsequent diagnostic programs (the MesoUser manager does not support such diagnostic tools).

The IFTRAJ switch may be used to provide data to the TRAJEC program. The winds, temperature, mixing ratio and pressure are in this subset of data. This file can be smaller than the history file, permitting the trajectory file to be written at a higher output frequency. Even during an explicit moisture run (where cloud water and rain water are predicted), the ability of the history file to roll over onto 10 additional volumes overrides any advantage held by the smaller trajectory file. Most users input the standard history output to TRAJEC.

The IFOMG and IFRAIN dispose only particular fields to a output file, diagnosed vertical motion and the accumulated precipitation, respectively. The precipitation is available from the history volumes, and the vertical motion can be kinematically computed from the history data set. Most users output the standard history files only, instead of the special purpose data sets (each of which requires additional programs).

A time series subroutine is also provided to the user, with the appropriate hooks in the local input file. When the IFPOIN switch is activated (the values IPOUT, JPOUT, and KPOUT from the LPARAM namelist specify the location), the model forecast information at that computational grid point is written. If this option is used, it is typical to write the data at every time step. The code is currently designed to allow only one point in the coarse-grid, and one point in the fine-grid domain to be handled in this fashion.

There are several print options in the code to allow the user to create variable listings during the forecast. The IFPRT logical and the PRTFRQ time interval control the KXOUT horizontal slabs and the JXSEX vertical sections extracted from the initial conditions and forecast data. This option uses the MAPSMP routine (subroutine OUTPT in most other components of the modeling package). A scaled, gridded sample of the 2-D arrays (either horizontal slabs or vertical cross sections) are printed for much of the available data. If the user turns on ISFOUT (the surface parameter print switch), the terrestrial fields are printed in a similar manner.

The MASCHK flag controls print information in the listing file. At the frequency requested, the mass of air and water contained in the domain is output. For the coarse-grid domain, the temporal change of these values tends to be small (several per cent), but the fine-grid domain differences can be significant. Domain mass tendencies for the regional models occurs through mass transport through the boundaries, which is of concern for the outer domain. The inner domain, because of the smaller size is more sensitive to a high or low pressure system propagating through the grid, with regard to the MASCHK flag. Users should not be concerned to see large % differences in the dry air mass. Users can construct scenarios where the mass of water in the fine grid changes by a factor of two (again, in the nested-grid domain, this is of no real concern).

The logical switch PBLCHK prints boundary layer information at selected grid points (maximum of five points) at the frequency PBLFRQ. These switches use the values IXOUT, JXOUT from the LPARAM namelist to choose grid points at which to write such information

as the PBL height, the bulk Richardson numbers, and the regime.

Two flags that were not in the Version 7 OPARAM namelist are the ISKF and ISKB flags. These are the integer number of time periods in the initial conditions to skip (ISKF), and the integer number of boundary condition time periods to skip (ISKB). This allow the user to generate a long series of analyses available for the model, and choose different time periods at which to begin the forecast.

```
&LPARAM
;------FOR LARGE DOMAIN:
IBLTYP(1) = 2,     ;PBL TYPE - 0, 1, 2
IBOUDY(1) = 1,     ;BOUNDARY CONDITIONS - 0, 1, 2, 3, 4
IDRY(1)   = 0,     ;MOIST OR DRY CASE - 0, 1
IMOIST(1) = 1,     ;PASSIVE, CUMULUS PARAM,  OR EXPLICIT - 0, 1, 2
XMOIST(1) = 1,     ;MOIST EFFECTS IN THERMODYNAMIC EQN - 0,1
ISFFLX(1) = 1,     ;SURFACE FLUXES - 0, 1
ITGFLG(1) = 1,     ;SURFACE TEMPERATURE - 1, 3
ISFPAR(1) = 1,     ;SURFACE CHARACTERISTICS - 0, 1
ICLOUD(1) = 1,     ;CLOUD EFFECTS ON RADIATION - 0, 1
ICDCON(1) = 0,     ;CONSTANT DRAG COEFFICIENTS - 0, 1
IFSNOW(1) = 0,     ;SNOW COVER EFFECTS - 0, 1
IMOIAV(1) = 0,     ;DO NOT CHANGE
IVMIXM(1) = 1,     ;VERTICAL MIXING OF MOMENTUM - 0, 1
HYDPRE(1) = 1,     ;HYDROSTATIC EFFECTS OF LIQUID WATER - 0,  1.
IEVAP(1)  = 1,     ;EVAPORATION OF CLOUD WATER AND RAINWATER - <0, 0, >0
IFRAD     = 0,     ;RADIATION COOLING OF ATMOSPHERE - 0, 1
RADFRQ    = 30,    ;FREQUENCY SOLAR RADIATION IS COMPUTED
ICUSTB    = 1,     ;STABILITY CHECK FOR CUMULUS PARAM. - 0, 1
ITQPBL    = 0,     ;GROUND TEMPERATURE FORECAST AT BOUNDARIES - 0, 1
IXOUT(1,1) = 20,   ; I INDEX OF GRID POINT FOR PBL CHECK
JXOUT(1,1) = 20,   ; J INDEX OF GRID POINT FOR PBL CHECK
KXOUT(1)   = 15,   ; K LEVEL OF HORIZONTAL SLICE FOR PRINTER OUTPUT
JXSEX(1)   = 20,   ; J INDEX OF NORTH SOUTH SLICE FOR PRINTER OUTPUT
IPOUT(1)   = 20,   ; I INDEX FOR TIME SERIES OUTPUT
JPOUT(1)   = 20,   ; J INDEX FOR TIME SERIES OUTPUT
KPOUT(1)   = 5,    ; K INDEX FOR TIME SERIES OUTPUT
&END ;-------------------------------------------------------------
```

The LPARAM namelist contains choices for the different physical packages in the model. Most of the switches in the coarse domain need to be set identically to those in the fine domain (the fine-grid boundary condition must be time dependent). If the switch has only two options (0 or 1), then 0 implies to not use the option, 1 implies to use the option. Switches with more than two settings allow the user to choose which version of a required treatment to implement. In the LPARAM namelist, all occurrences of the first element of the array are values of the switch that refer to the coarse-grid, the fine-grid values would be the same flag, but referenced with the second value of the array (IBOUDY(1) refers to

the coarse-grid boundary condition, IBOUDY(2) refers to the fine-grid boundary condition). Those switches without indices refer to both the coarse-grid and fine-grid domains. There were no changes made to this namelist during the Version 8 upgrade process.

The flag to choose which boundary layer parameterization to use is IBLTYP. To use a frictionless PBL, IBLTYP = 0. There are no surface fluxes, the land use information is ignored, there is no check on the snow cover, and at each grid point the ground temperature is assumed to remain constant throughout the forecast. The next step of complexity available is to choose a single level bulk–aerodynamic boundary layer, IBLTYP = 1. This formulation assumes a well mixed PBL, throughout the depth of the first $\sigma$-layer. The available land use information and snow cover are used, the ground temperature and surface fluxes are computed. Also available is a multi–level PBL routine when IBLTYP = 2. The bulk PBL scheme requires a single thick sigma layer near the ground (the first half layer $\sigma$ = 0.95). The multi-level PBL requires several thin sigma layers near the ground to resolve the boundary layer (typical half sigma layers in the PBL would be $\sigma$ = 0.995, 0.985, 0.97, 0.945, 0.91, 0.865).

The choices for the boundary conditions are controlled by the IBOUDY flag. The user may choose fixed boundary conditions (IBOUDY = 0); relaxation towards a large scale analysis (IBOUDY = 1); time dependent (IBOUDY = 2); time dependent with an inflow / outflow check (IBOUDY = 3); sponge with relaxation towards a large scale analysis (IBOUDY = 4); or periodic in east / west, constant north / south (IBOUDY = 5). Most model runs have either the relaxation or sponge boundary conditions for the coarse domain. All two way interacting nested forecasts must use time dependent BC for the fine grid.

The switch to allow moisture in the model is the IDRY flag. If IDRY = 0, the simulation is not dry; if IDRY = 1, the simulation is dry. During a dry run, there is no precipitation, no moisture prediction, no cloud water and rain water computation, and no moisture effects in the thermodynamic equations.

When the model run is not dry (IDRY = 0), the user decides which method computes the precipitation. To remove the subgrid scale moisture with a cumulus parameterization technique based upon a predefined heating profile, set IMOIST = 1. Users may choose to explicitly predict the cloud and rain water fields to remove the resolvable scale moisture. When IMOIST = 2, the explicit scheme produces precipitation only from saturated columns (precipitation falling into unsaturated cloud layers continues only after that layer becomes saturated). The option to couple the explicit scheme with a cumulus parameterization is in place. The explicit moisture flag must be set IMOIST = 2, and the MM4 C-shell variable must be activated GRELLsw = GRELL.

The index locations (IXOUT, JXOUT, KXOUT, JXSEX, IPOUT, JPOUT, and KPOUT) were described with the accompanying option in the previous section dealing with the OPARAM namelist. These are the indices to the various requested print options (such as the PBL check) and the choices for the locations of the slabs of the predicted and initial variables (during the intermittent printing of the horizontal and vertical slices).

The remaining flags in the LPARAM namelist are described briefly in the MM4 documentation, in the MM4 C-shell job deck, and in the source code. The logical structure

in the MM4 FORTRAN is the final word on the actual implementation of the various options. Users who are tinkering with the parameterization choices to decide on the surface features and effects, the effects of clouds on radiation, hydrostatic effects and vertical momentum, etc, are assumed to be no longer running the model in a black box mode, and are conducting mesoscale modeling research. Disabling various physical feedbacks provides the capability to more clearly discern cause and effect relationships that are masked in the more complicated runs.

```
&PPARAM
TIMAX  = 1440,    ; LENGTH OF FCST, IN MINUTES
ZZLND  = 0.1,     ; ROUGHNESS LENGTH OVER LAND WHEN ISFPAR=1
ZZWTR  = 0.0001,  ; ROUGHNESS LENGTH OVER WATER WHEN ISFPAR=1
ALBLND = 0.15,    ; ALBEDO OVER LAND WHEN ISFPAR=1
THINLD = 0.04,    ; THERMAL INERTIA OVER LAND WHEN ISFPAR=1
XMAVA  = 0.3,     ; MOISTURE AVAILABILITY WHEN ISFPAR=1
CONF   = 1.0,     ; CONDENSATION THRESHOLD
QCTH   = 0.5E-3,  ; THRESHOLD FOR ONSET OF AUTOCONVERSION (KG/KG)
QCK1   = 1.0E-3,  ; CONSTANT AUTOCONVERSION RATE (KG/KG/S)
&END ;-------------------------------------------------------------
```

The PPARAM namelist contains an entry that is modified nearly every time a forecast is generated. The length of the forecast (in minutes) is defined with the TIMAX flag. For example, a 24 hour forecast would use TIMAX = 1440. This refers to the total extent of a forecast, so a forecast that is to be a total of 24 hours long, but is being restarted at the 12th hour of the original forecast, still uses TIMAX = 1440. In the Version 7 PPARAM namelist, the user was required to fill in the full $\sigma$-levels (SIGMA), the grid distance (DX), and the model lid (PTOP). All of these values are sent through the record header from INTERP into the model directly via the initial condition data.

There are other values in the PPARAM namelist that are switches to allow the user to tune the surface parameterization values to reflect new empirical relations, assumptions about the particular domain, or measured quantities. Additionally, there are switches to allow the user to modify rates and thresholds used in the precipitation computations. Comments concerning the modification of any of these values is beyond the scope of this document.

```
&FPARAM
;
;          ****************************************************
;          ***************** ANALYSIS NUDGING *****************
;          ****************************************************
;
;-----IS THIS A GRID 4DDA RUN
;            3D          3D          SFC         SFC
;          COARSE      FINE       COARSE       FINE
;-------------------------------------------------------
I4D=          0,          0,          0,          0,
DIFTIM=     720,        720,        180,        180,
IWIND=        1,          1,          1,          1,
GV  =      3.E-4,      3.E-4,      3.E-4,      3.E-4,
ITEMP=        1,          1,          0,          0,
GT  =      3.E-4,      3.E-4,          0,          0,
IMOIS=        1,          1,          1,          1,
GQ  =      1.E-5,      1.E-5,      3.E-4,      3.E-4,
IROT=         0,          0,
GR  =       5.E6,       5.E6,
;
;-----DO NOT NUDGE IN THE BOUNDARY LAYER
;           COARSE      FINE
;-----------------------------
INONBL =      0,          0,          ; U WIND
              0,          0,          ; V WIND
              1,          1,          ; TEMPERATURE
              0,          0,          ; MIXING RATIO
;
RINBLW=480,     ; RADIUS OF INFLUENCE FOR SURFACE ANALYSIS (KM)
NPFG=240,       ; NUDGING PRINT FREQUENCY (CGM TIMESTEPS)
```

The only namelist in the MM4 system that was added during the Version 8 and UNICOS conversion is FPARAM. This namelist deals solely with the options associated with the FDDA capabilities in the mesoscale model. The namelist is divided into two sections, options particular to analysis nudging, and those options that pertain to observational nudging. The first section to be described is the analysis nudging option.

There are three distinct parts to the analysis nudging namelist. The first part has to do with which fields are being nudged, in which domain (coarse-grid and/or fine-grid), and where the nudging is to occur. To aid the user with the array index representation and the associated grid designation, the four columns (3D COARSE, 3D FINE, SFC COARSE, SFC FINE) provide a template for the nudging information. The integer flags (I4D, IWIND, ITEMP, IMOIS, IROT) are activated by setting the switch = 1, or deactivated by setting the switch = 0.

The I4D flag is the on / off switch for analysis nudging. If all four values of I4D are

set = 0, then no analysis nudging occurs. The user may specify coarse-grid nudging only, fine-grid nudging only, surface nudging only, or a combination of any of these with this set of four flags. The DIFTIM flag is the time interval in minutes between the individual time periods for which data is available (it is typical to have 12 hours between the upper air analyses and 3 hour intervals between the surface analyses).

The next three integer flags (IWIND, ITEMP, IQ) allow the user to choose which of the fields to nudge towards the analyses (1 implies nudge, 0 implies do not nudge). The associated floating point values with each integer flag (GV, GT, GQ) are the weighting factors that are used directly in the tendency equations. In this example, note that temperature is not nudged at the surface, and that moisture is weakly nudged away from the surface. The next two values (IROT, GR) deal with nudging to the rotational component of the wind (note that this is not available at the surface).

The second part of the analysis nudging namelist allows the user to specify which fields not to nudge in the boundary layer. Even if the user decided to not nudge towards a surface analysis, the forecast could still be nudged to the analyses in the boundary layer. In this example, the temperature is not nudged in the boundary layer, while the boundary layer winds and moisture are nudged to the available analyses.

The third part of the analysis nudging portion of the FPARAM namelist contains a tuning knob and a printing option. The switch RINBLW allows the user to set the radius of influence that the surface observations have over neighboring grid points. Grid boxes without enough observations within a prescribed radius of influence have the nudging weights reduced. The default value was chosen after experimentation based on the surface observation density over the U.S.

```
;
;         ****************************************************
;         *************** OBSERVATION NUDGING ****************
;         ****************************************************
;
;-----INDIVIDUAL OBSERVATION NUDGING
;         COARSE     FINE
;--------------------------------
I4DI   =     0,         0,
ISWIND=      1,         1,
GIV    =   3.E-4,     3.E-4,
ISTEMP=      1,         1,
GIT    =   3.E-4,     3.E-4,
ISMOIS=      1,         1,
GIQ    =   3.E-4,     3.E-4,
;
;
RINXY=150,      ; RADIUS OF INFLUENCE IN HORIZONTAL (KM)
RINSIG=.001,    ; RADIUS OF INFLUENCE IN VERTICAL (SIGMA INDEX UNITS)
TWINDO=40.0,    ; HALF PERIOD OF TIME WINDOW (MINUTES)
NPFI=560,       ; NUDGING PRINT FREQUENCY (CGM TIMESTEPS)
&END ;-------------------------------------------------------------------
```

The second half of the FPARAM namelist is concerned with nudging the model towards individual observations. The observations are at a 4-D point in the atmosphere (given by time, x, y, and sigma index level). There is no switch to deactivate surface or boundary layer nudging. The interpretation of the values is similar to the analysis nudging options. The first array element refers to the coarse-grid domain, the second array element refers to the fine-grid domain.

The I4DI flag allows the user to deactivate the observation nudging option (in this namelist, there is no observation nudging as both values of I4DI are set = 0). The next three integer flags (ISWIND, ISTEMP, ISMOIS) choose which fields are to be nudged towards the observations. The weighting factors for each of the associated fields (GIV, GIT, GIQ), have the same interpretation as in nudging to the analysis.

The next three options decide how large a radius of influence the observation has horizontally (in km, RINXY) and vertically (in sigma index units, RINSIG). This namelist allows grid points 150 km from the observation to know of its influence. With the vertical influence set to 0.001 sigma index units, the user is guaranteed of no direct vertically propagating influence. The time for which the observation is considered valid is chosen with TWINDO. This is the half period time in minutes, before and after the observation was taken, through which the observational value is considered valid.

## 9.2 Parameterized Dimensions

The forecast model has requirements for both horizontal and vertical dimensions,

similar to the INTERP program. To allocate adequate space, the model must know the maximum of several types of dimensions. The MM4 C-shell has a description of each of the parameter values prior to the UPDATE parameter modifications.

Since different physical packages require additional storage (the explicit moisture scheme has 3-D cloud water and rain water), the user must fill in the parameter statements consistent with the choices made in the namelists.

The naming convention for the horizontal dimensions are similar to the rest of the modeling system. The parameter values with an `I` refer to the I–direction, those with a `J` refer to the J–direction. The vertical dimensions all have a `K` in the name.

There are two domains that require the user to parameterize dimensions, the coarse-grid and the fine-grid. Similar to INTERP, the maximum of each dimension (in the I–direction, J–direction and K–direction) between the coarse and fine grid must also be defined in a parameter statement.

For each of the domains, the user must fill in the parameter values considering the physical switches that were activated in the namelists. If the user has chosen a coarse + fine grid model forecast, then the nested parameter statements must be the size of the fine grid (defined in the master input file, TERRAIN, and INTERP). If the user chose the explicit moisture scheme (`IMOIST = 2` in the `LPARAM` namelist), the moist parameter statements are required. The user must include the FDDA parameter statements if analysis nudging is being used.

There are a total of 27 parameter statements in the MM4 model. The MM4 C-shell has a description of each value. Some incorrect combinations of settings of the parameter statements may be detected in the model, but the user should be careful. Unused parameter values should be set = 1.

The user may specifically request a time step to utilize during the integration (a default time step is internally computed). The time step (in seconds) refers to the most coarse resolution data. Most users multiply the coarse grid distance (in km) by 1.5 to obtain a conservative time step for MM4 (in seconds). For example, a 90 km forecast would yield a 135 second time step.

```
*/      ************************************************************
*/              parameter + time step:  USERDEF
*/      ************************************************************
*/      ************************************************************
*/
*ID USERDEF
*/
*/      *************  COARSE GRID DIMENSIONS  **********************
*/
*/  IX, JX, KX        : DIMENSION FOR THE COARSE MESH VARIABLES, THESE
*/                      MUST BE FILLED IN.  IX (JX) IS THE NUMBER OF
*/                      VALUES ON DOT POINTS IN THE Y (X) DIRECTION.
*/                      KX IS THE NUMBER OF HALF SIGMA LAYERS.
*/  IXM, JXM, KXM     : DIMENSIONS FOR CLOUD WATER (QC) AND RAIN WATER
*/                      (QR). EXCEPT WHEN USING THE EXPLICIT MOISTURE
*/                      OPTION, SET THEM EQUAL TO 1.  FOR THE EXPLICIT
*/                      MOISTURE OPTION, THEY MUST BE EQUAL TO
*/                      IX, JX, KX.
*/  IXF, JXF, KXF     : IF USING FDDA CODE, THESE NEED TO BE SET TO
*/                      THE IX, JX, KX VALUES, IF FDDA IS NOT USED,
*/                      SET THEM TO 1
*/
*/      *************  FINE GRID DIMENSIONS  ************************
*/
*/  INX, JNX, KNX     : NESTED DOMAIN DIMENSIONS, IF NESTED GRID
*/                      IS NOT APPLIED, SET THEM EQUAL TO 1.
*/  INXM, JNXM, KNXM  : NESTED DOMAIN CLOUD WATER AND RAIN WATER
*/                      DIMENSIONS.  IF NESTED GRID IS NOT APPLIED,
*/                      OR IF NOT DOING EXPLICIT MOISTURE, THEN
*/                      SET THEM EQUAL TO 1.  IF THIS IS A NESTED +
*/                      AN EXPLICIT MOISTURE RUN, SET THEM TO
*/                      INX, JNX, KNX.
*/  INXF, JNXF, KNXF  : NESTED FDDA DIMENSIONS, IF THIS IS NOT A
*/                      NESTED RUN OR IF FDDA IS NOT USED,
*/                      SET THEM TO 1.    IF THIS IS A NESTED FDDA RUN
*/                      (WITH NUDGING IN THE FINE GRID), THEN SET THEM
*/                      TO INX, JNX, KNX.
*/
*/      *************  MAXIMUM OF COARSE AND FINE GRID DIMENSIONS *****
*/
*/  MIX, MJX, MKX     : MAX(IX,INX), MAX(JX,JNX), MAX(KX,KNX)
*/  MIXM, MJXM, MKXM  : MAX(IXM,INXM), MAX(JXM,JNXM), MAX(KXM,KNXM)
*/  MIXF, MJXF, MKXF  : MAX(IXF,INXF), MAX(JXF,JNXF), MAX(KXF,KNXF)
```

```
*D PARAME.5,7
      PARAMETER (IX   =  46 , JX   =  61 , KX   =  23)
      PARAMETER (IXM  =   1 , JXM  =   1 , KXM  =   1)
      PARAMETER (IXF  =   1 , JXF  =   1 , KXF  =   1)
*D PARAME.10,12
      PARAMETER (INX  =   1 , JNX  =   1 , KNX  =   1)
      PARAMETER (INXM =   1 , JNXM =   1 , KNXM =   1)
      PARAMETER (INXF =   1 , JNXF =   1 , KNXF =   1)
*D PARAME.14,16
      PARAMETER (MIX  =  46 , MJX  =  61 , MKX  =  23)
      PARAMETER (MIXM =   1 , MJXM =   1 , MKXM =   1)
      PARAMETER (MIXF =   1 , MJXF =   1 , MKXF =   1)
*D PARAME.20
      PARAMETER (NSPGX=5,NSPGD=5,IRAX=3)
*/
*/      the following line EXPLICITLY changes the time step
*D PARAM.1409
      DT=135
```

## 9.3 Generated Files

The MM4 program has several switches in the `OPARAM` namelist that allow different types of forecast output (`IFTAPE, IFFILT, IFTRAJ, IFOMG, IFRAIN, IFPOIN`). The gridded history data of the forecast (`IFTAPE`) contains information allowing the user to reconstruct the data archived by the `IFOMG` and `IFRAIN` switches.

Appendix A.7 has a description of the standard model output format (the forecast history activated with the `IFTAPE` flag). There is little reason to generate a forecast without archiving the forecast history data set. Users interested in the other generated data sets need to consult the MM4 source code for more information.

## 9.4 Hints and Caveats

Due to the complicated nature of a research oriented model, the list of hints can hardly be exhaustive. Except when MM4 detects unusual option combinations in the namelists or discrepancies in the parameter statements, there are no diagnostic statements from the model before it stops because of errors. In case of floating point exceptions in the model, the user should inspect the noise ratios that are output in the listing after each time step, to check if anything atypical occurs (increasing values). The next step is to look through the last several archived time periods of the forecast with the GRAPH code. Users unable to make headway after this process should contact the MesoUser manager at NCAR (*mesouser@NCAR.UCAR.edu*).

When the model forecast is being restarted from a previous save file from MM4, remember to change the names of the new save files to be archived to the MS (else they will overwrite the original save files archived previously on the MS).

Most users archive only the standard history data sets (activated with the IFTAPE flag in the OPARAM namelist), not the specialized data (activated with the typewr IFFILT, IFTRAJ, IFOMG, IFRAIN flags). Users interested in the special data formats need to reference the MM4 code. No MesoUser supported programs exists to ingest the time series, filter, vertical motion or rain data.

The nested grid must always have the boundary condition as time dependent (IBOUDY = 2). Most of the other switches in the LPARAM namelist require the coarse grid and fine grid to be defined identically.

The parameter statements must match the options chosen in the MM4 namelists with regards to explicit moisture, nested domains, and analysis nudging. These parameter values must be consistent with the record header data coming from INTERP.

- a forecast with a coarse-grid and a fine-grid domain requires the users to modify the nested parameter statements, to inform the model by activating the NESTGD switch in the OPARAM namelist, and to make the fine-grid initial conditions available to the model executable
- users requesting to explicitly predict resolvable scale moisture must modify the moist parameter statements, and set both of the namelist flags: IDRY = 0 and IMOIST = 2; if the explicit scheme is to be coupled with the Grell cumulus parameterization scheme, the user must set GRELLsw = GRELL in the C-shell
- if the user is doing analysis nudging, the FDDA parameter statements are required, the user must switch on at least one of the I4D flags in the FPARAM namelist, and the FDDA input files must be available to the program

When using the FDDA options with the model, the user should overcome the urge to appreciably modify the weighting factors (GV, GT, GQ, GR for the analysis nudging, and GIV, GIT, GIQ for the observation nudging). Weighting factors that are too large tend to use the model to regenerate the original analyses, and weighting factors that are too small do not noticeably nudge the model towards the analyses.

# II. CCM2 Internals

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.

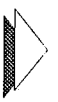
This section is meant for those who wish to make substantive changes to the CCM2 code. Familiarity with advanced aspects of the UNICOS operating system is assumed. We will reference Cray manuals where helpful. Also note the glossary of terms in "Appendix A: Glossary."

## 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.

*CCM2 data structures*



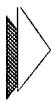
In the previous version of the Community Climate Model, CCM1, a large blank *common* buffer provided storage for all gridpoint fields. It served both as "permanent" cycling storage for the out-of-core[1] implementation and as temporary storage, managed piecemeal by the various physics routines. This 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.

- 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

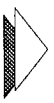
*Use of synchronous I/O, and the Cray SSD*



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.

*"Plug-compatible" physics modules*

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."

*History tape handler*

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.

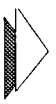
*Multitasking implementationr*

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.

*Random order of history tape*

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.

*Readable standard code*



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.

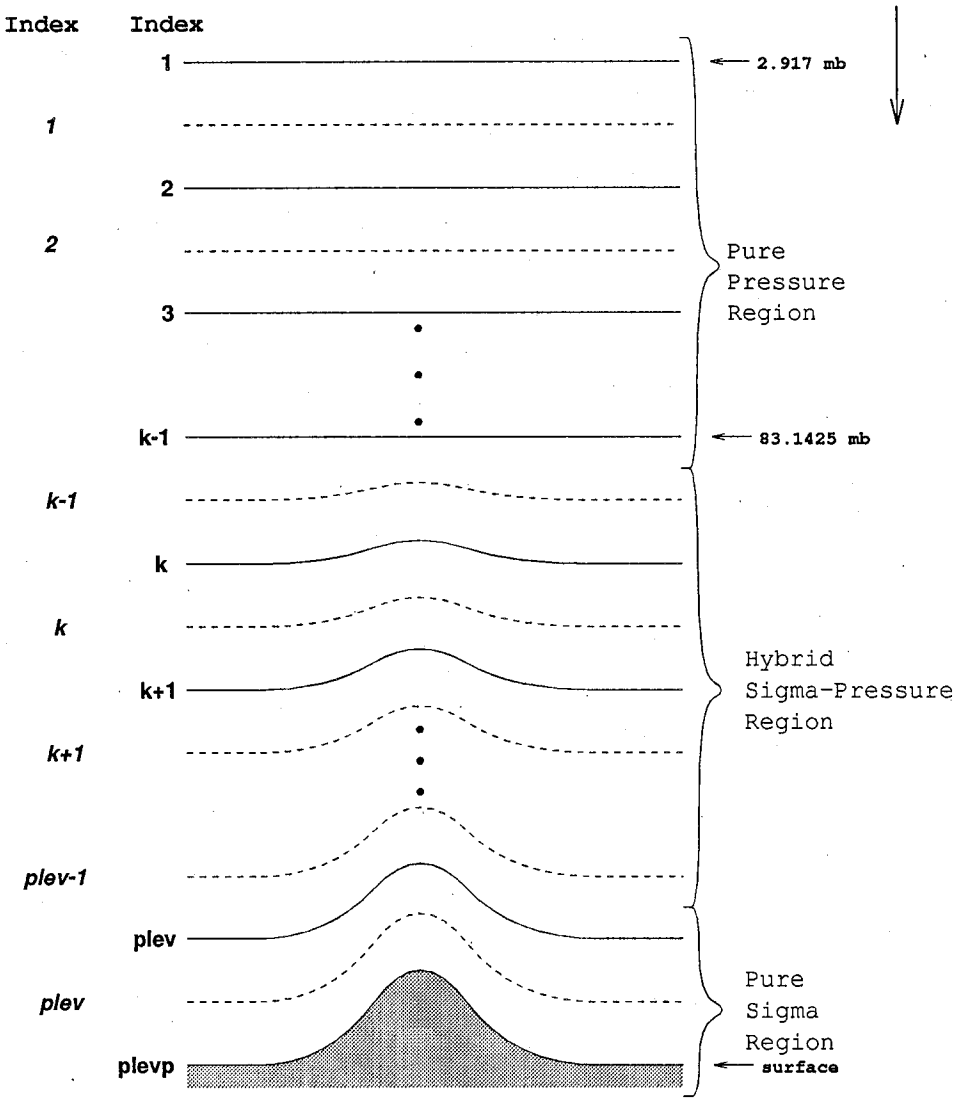



**Figure II.1.** CCM2 Hybrid Vertical Coordinate.

## 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.

*Transported constituents*

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 `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.

*Gridpoint array dimensioning*

Model gridpoint space arrays are almost universally declared with longitudinal dimension "`plond`," where `plond = plon + 1 + 2*nxpt`. `plond`, `plon`, and `nxpt` are all defined in Fortran *parameter* declarations. These and other parameter values used by the Model are tabulated in "Appendix C: CCM2 Parameter Definitions."

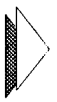
`plon` is the number of actual data points in the longitudinal direction, and `nxpt` is a wrapping number required by the SLT package. Arrays that go through the Fourier transform require two additional longitude points at the end to accommodate storage of the wave 0 (mean) information by the **`fft`** 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 `plond` wherever possible, whether or not it was required.

Actual data start at longitude location `1 + nxpt` for all of the prognostics except surface pressure, and location 1 for all other non SLT-specific arrays with longitude dimension `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 `LINEMS` from `LINDRV` (this is very high in the calling tree: refer to Fig. I.2). Some of the calling and declaration sequence looks like:

*LINDRV:*

```
      call linems(...,ps(1,lat,n3),q3(i1,1,1,j,n3),
     $            b1(nzm1+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

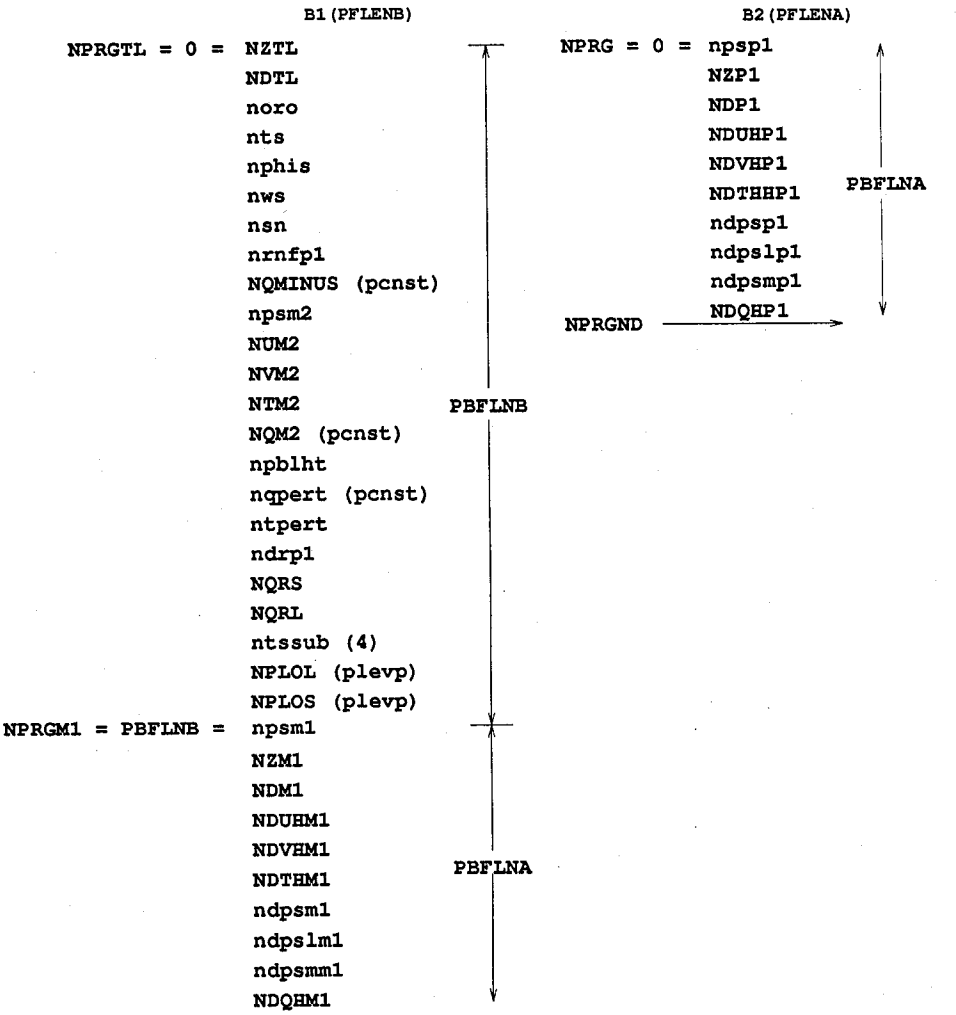



**Figure II.2.** Layout of Main CCM2 buffer.

*Buffer contents*

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*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

(*common*/com3d/), and is copied to and from the Model buffer for the sole purpose of avoiding the **fft** performance penalty.

*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:

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

and the called subroutine references this field mnemonically, thus:

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

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:

```
real buf(pflena) ! model buffer
pointer (pz,z) ! declaration of pointer variable
real z(plond,plev) ! part of model buffer
.
.
.
pz = loc(buf(nzp1+1)) ! array z may now be accessed
```

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.

*Declaration of history tape buffer*

The declaration in `LINDRV` looks like:

```
subroutine lindrv(lhbuf)
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.

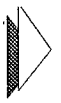
We recommend that the user choose a history tape write frequency (specified by parameter `NHTFRQ`) that divides evenly into the frequency of the longwave radiation calculation (specified by `IRADAE`). Otherwise, the large absorptivity/emissivity arrays must be written to the regeneration datasets.

### (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.

*Use of stack in multitasking*



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.

*Storage in /com3d/*

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.

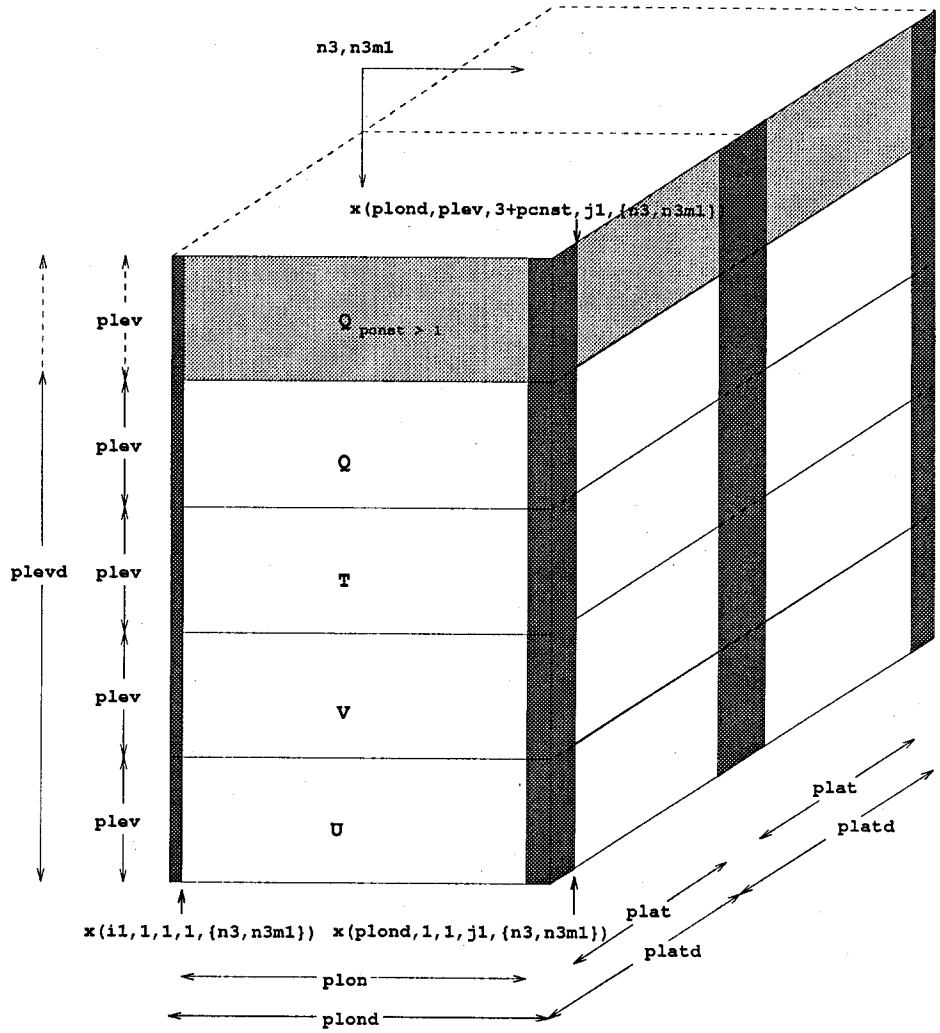



**Figure II.3.** Pictorial Representation of /com3d/ Storage.

If `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 `q3(i1m+i,k,m,j1m+lat,n3)` where `m` is the constituent index, `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,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 `plevd`, defined as `plev*pcnst`, allows dimensioning of u, v, and t as `(plond,plevd,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
     $     v3(plond,plevd,platd,2),        ! v-wind component
     $     t3(plond,plevd,platd,2),        ! temperature
     $     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. Equivalenced 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 (`/comslt/`). 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`, `qxl`, 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`, `qxl`, 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

*Ordering of spectral arrays*

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.

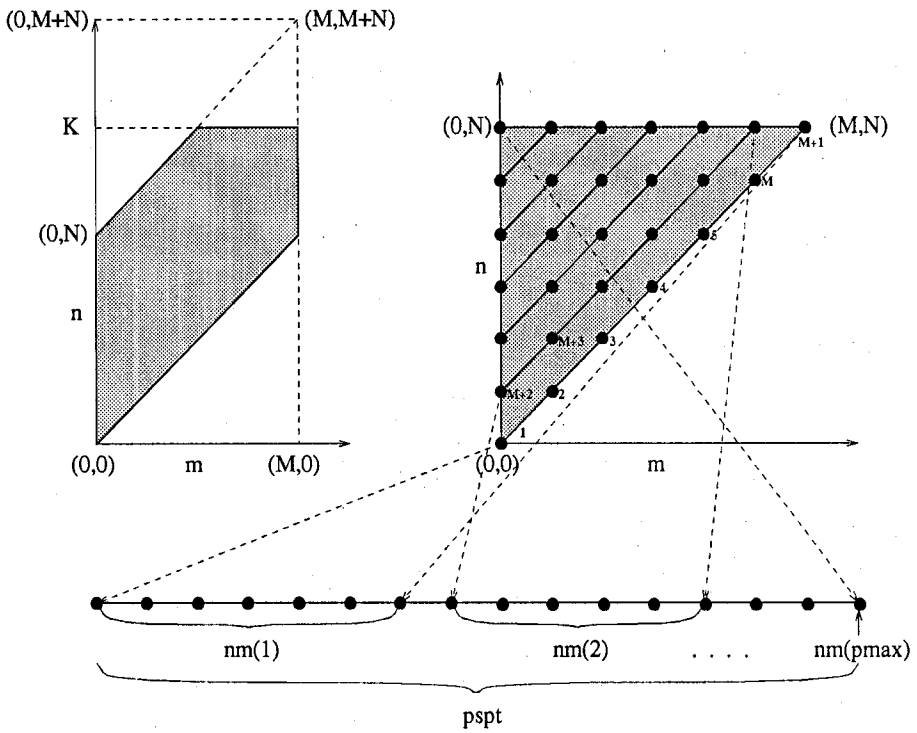


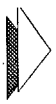

**Figure II.4.** Spectral Storage Arrangement. Notice the linear storage arrangement for spherical harmonic coefficients and associated Legendre polynomials.

## 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.

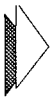
Toward this goal, each physics package has its own *common* block which contains only those physical constants which are required for its own use. For example, `/comvd/` contains eight variables, namely, just those constants that are required by the vertical diffusion package. The various physical parameterizations that make up the physics of CCM2 are hooked to the Model control code using a standard form of initialization and interface routines.

*Parameterization initialization 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.

*Parameterization commons*



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/`, `/comtsc/`, `/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.

*Parameterization interface routines*

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.

*Model parameter statements*

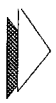
Like the Model *common* blocks, Fortran *parameter* statements are split by functional purpose into separate "include" files. One include file contains the basic grid resolution parameters (`pmgrid.com`). Other *parameter* include files are as follows: `pagrid.com` contains some auxiliary parameters 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 `parslt.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.[1] 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.

*Contents of SSD units*



The units 11, 21, 22,and 60 (Model variables `nra1`, `nrb1`, `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 `nra1`, `nrb1`, and `nabem` for Fortran direct access. The data on `nra1` are the portion of the buffer reserved for time `n+1` data (called `b2` in Subroutine `LINDRV`), between pointers `nprg` and `nprgnd` (see Figure II.2 on page 81). The data on `nrb1` are the buffer `b1` in Subroutine `LINDRV`, containing the half-time-filtered data at `nprgtl` and the `n-1` data at `nprgm1`.

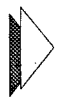
Both the main Model buffer and the history buffer are written to and read from the out-of-core I/O device (SSD) as unblocked files. Record size is computed by rounding up to the nearest multiple of 512 words. This saves an enormous amount of system CPU time which would otherwise have to be spent blocking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

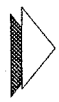
File management in the Model includes acquiring files needed as input from the NCAR Mass Storage System (MSS), accessing data from these files as they reside on temporary Cray disk using standard Fortran I/O of various kinds, creating Model output files on disk, and disposing them to the MSS using a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

*Input datasets*



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.

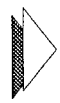
*Initial run, initial dataset:*



Subroutine `INITAL` 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`.

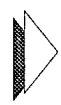
`INITAL` then calls Model routine `RDHDR` to read the header records of the initial dataset. Subroutine `INIDAT` reads and transforms the data. Processing of the initial dataset is then complete.

*Continuation run, regeneration data*



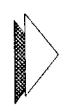
As described in "Restart/Regeneration Datasets" on page 53, the restart dataset contains only a Mass Store System pathname for the set of regeneration datasets last disposed to the MSS. In the case of a restart run, subroutine `RESTRT` uses this pathname to acquire all relevant regeneration datasets, calling Model utility routine `ATTACH`. 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.

*Boundary datasets*



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 `OZNINI` 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.

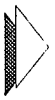
*Output datasets*



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 INITAL for initial runs and by RESTRT for continuation runs to assign the complete set of regeneration datasets.

*System file-handling routines*



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

*Use of $TMPDIR*

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!

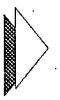
By making a $TMPDIR directory the "current directory" for the Model run, we assure that all local files are protected from scrubbing. The user may choose, via *namelist* parameter LDEBUG, to link the final set of output files from a run into /usr/tmp/$LOGNAME, (see Table I.1 on page 14) to possibly avoid 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.

*Asynchronous mswrite*



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.

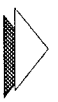
## 2. Use of the NCAR Mass Storage System

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.

*msread and mswrite*

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*



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

/usr/tmp/$LOGNAME disk pathnames as shown below when complete, if *namelist* input parameter LDEBUG is true. This link will be removed when a successful **mswrite** is executed on the file, except for the final output file set of a run. If LDEBUG is false, the default setting, the volume will be disposed directly to the Mass Store, and no links to or removes from /usr/tmp/$LOGNAME will take place. However, the $TMPDIR file will be removed to prevent filling up $TMPDIR on long Model runs.

*Linking into /usr/tmp*

The history file will be linked into the directory

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

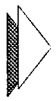
and the restart and regeneration datasets into directory

```
/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 file names*



History tape files are named as follows:

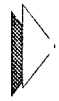
For the primary history tape, "h" plus 4 digits, starting by default with 0001, 0002, .... For example, h0004 is the fourth history tape written for a particular case, assuming that input parameter STFNUM, described below, takes the default value of one.

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 file names*



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.

*Splitting regeneration datasets*

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 pathnames*

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 II.1
## System Routine Calls in CCM2

| System Routine | Description | Model Routines |
|---|---|---|
| **Local File Interface** | | |
| assign | Attach local file to job | atchbnd, attach, restrt, rgnfls, wrapup |
| setf | Preallocate space on disk for local file | prealc |
| **Mass Store Interface** | | |
| msread | Read from Mass Store | attach |
| mserror | Print Mass Store errors | attach, savdis |
| mswrite | Write to Mass Store | savdis |
| **Interface with Run Script** | | |
| getenv | Get value of environment variable | data, igtseq |
| **Miscellaneous System Functions** | | |
| abort | Fatal error stop, issue traceback | endrun |
| date | Return current date as character string | ccm2, inthed, wrthdr |
| clock | Return current time as character string | ccm2, inthed, wrthdr |
| ishell | Issue system (shell) command | iostop (for error processing)<br>prealc (for disk space preallocation)<br>savdis (to remove and link local files) |

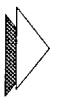
# D. Continuation Run Logic

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.

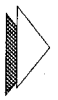
There are three different kinds of continuation runs:

*Restart run*



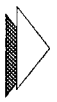
A "restart" run (*namelist* parameter `NSREST = 1`) starts from the last point at which output was disposed, and is exact, that is, the model results for a restart run will be identical to the results of the case if it had not been interrupted.

*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.

*Contents of regeneration datasets*

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/, /comslt/, /comtim/, /comhst/, /comhed/, /comqfl/ and /crdsrf/. The data on work units nrb1 and nra1 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

*How to restart*

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 essentially no loss of resources. If the Model terminates abnormally (e.g., hardware 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.

*How to regenerate*

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 regeneration datasets, and starts as in the restart run.

*How to branch*

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 follows:

```
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).

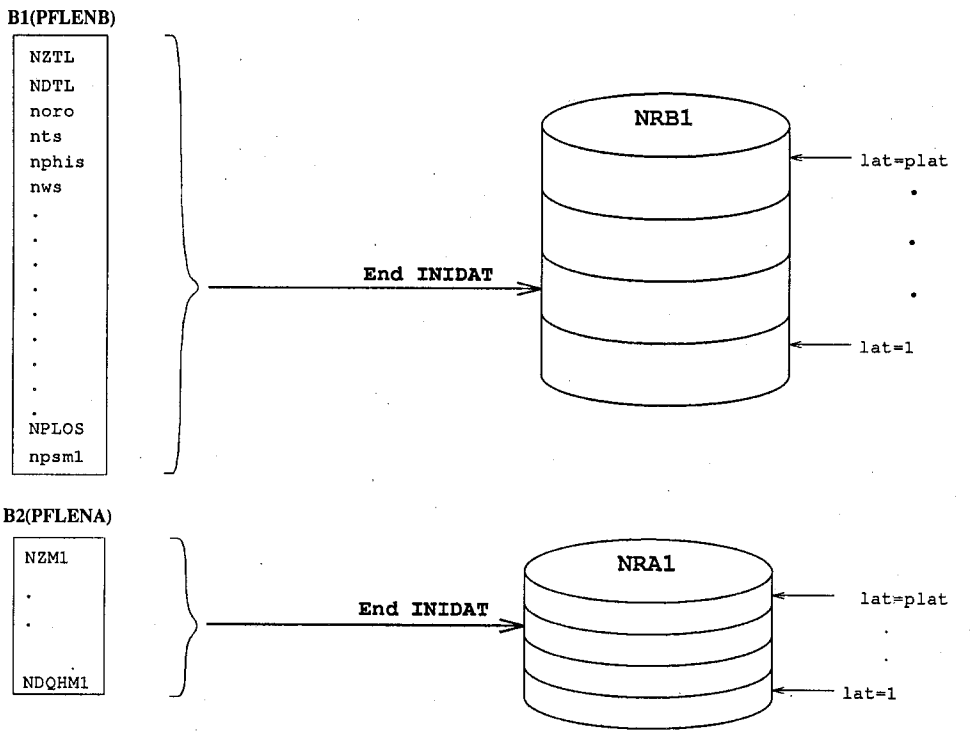



**Figure II.5.** Use of SSD Work Units in Initialization Phase.

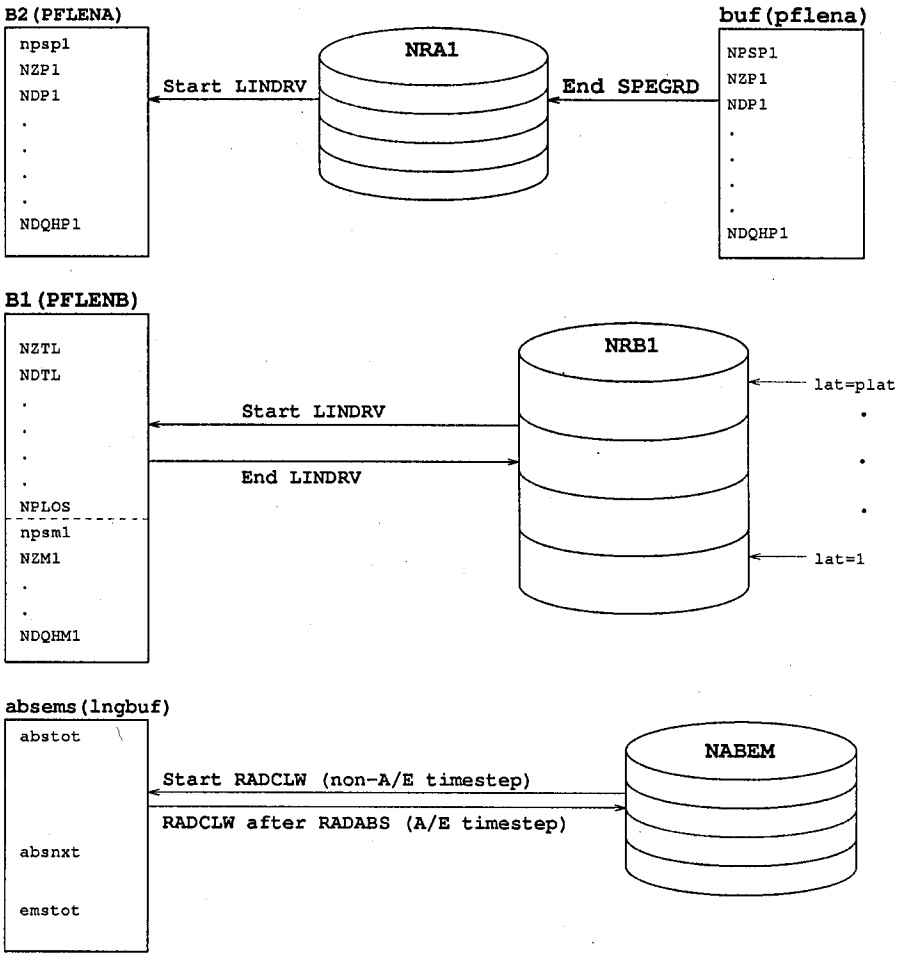



**Figure II.6.** Use of SSD Work Units in Time-Stepping Phase.

## 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 INITAL and the routines beneath INITAL 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*

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.

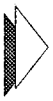
## 2. Computational Loops: Code and Data Flow

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

*STEPON*



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 GRDINI 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 GRDINI (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.

*ADVNCE*

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 OZNINT).

*CALDYI*

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

*SCAN1*

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*

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 (uxl, uxr, qxl, 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

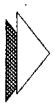
does not have a great effect on the memory hi-water mark. However, this situation changes rapidly as additional constituents are added and the size of the arrays qxl and qxr overwhelms the size of the local database in the biggest branch in the SCAN1 calling tree.

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*

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.

*SCAN2*



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 n3m1 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 (n3m1), and the data just computed in time index n3m1 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*

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 nra1, nrb1, 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 nrb1 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-

cussion 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.

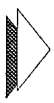
*SPEGRD*

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 nra1 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

*DYNDRV*



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*

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 precomputing 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

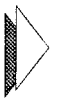
before being output to the history file. In order to accomplish a forward as opposed to leapfrog timestep at `nstep=0`, it is necessary to copy time level `n` temperature and constituents to time level `n-1` after the initial adjustment (see prior discussion of `INIDAT`). The second physics driver invoked from `LINEMS` is `TPHYS`; it controls the physics that produce tendencies. Details of how this routine works are outlined below under section "`TPHYS`."

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).

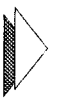
`LINEMS` also converts the wind field between straight meters/second and meters/second multiplied by cosine(latitude). These conversions are necessary because the SLT code uses the direct meters/second form while most of the rest of the model uses the cosine(latitude) form.

*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*

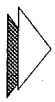


`RADCTL` controls the radiative transfer code. Units are *cgs* while the rest of the model uses *mks* units. `RADOUT` does the conversion from *cgs* to *mks* for requisite radiation fields written to the history tape. Frequency of longwave absorptivity and emissivity calculations is controlled by o*namelist* 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*

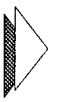


`SLTB1` drives the SLT constituent forecast calculations. Each call to this routine fills one latitude line of three-dimensional array `qfcst`. For more detail on the algorithms of the SLT, reference *Description of the NCAR Community Climate Model (CCM2)* (Hack *et al.*, 1992).

# 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.

*Multitasked performance*



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.

*Deterministic results in a multitasked environment*

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*

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

```
CMIC$ DO 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.

*Multitasked loops*

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:

```
CMIC$ DO ALL SHARED(...) PRIVATE(...)
      do irow=1,plat/2
      ...
      end do
```

Computations within the loop for each value of `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.

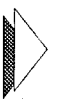
In the SLT routines which drive multitasked loops (`SCANSLT` and `SLTINI`), each latitudinal iteration may be done in parallel. In the spectral dynamics driven by `DYNDRV`, the Gaussian quadrature and semi-implicit timestep computations are parallelized over diagonals of the spectral truncation (remember 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,

```
subroutine xxx(lat)
if (lat.eq.1) then
... code to initialize static variables ...
end if
```

**will not work** inside multitasked portions of the Model. If `lat = 2` happens to be the first process to reach routine `xxx`, the static workspace of the routine will not be properly initialized. To guarantee that routine `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.

*I/O in the multitasked code*



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.

*Forcing sequential execution*

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 "CMIC$ GUARD" statements are necessary.

## 2. Memory Management

*Use of the Fortran stack*

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.

*Memory usage in 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-
DYNDRV@196
DYNDRV@211
SCAN1@570
SCAN2@216
SCANSLT@293
SLTINI@134

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:

```
Command    Started    ...   Memory
  Name       At             HiWater
========  ========          =======
ccm.xx.2  17:16:52   ...     13052
```

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 **`segldr`** 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.

*Add Model field to history tape*

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 `pflds` 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.

*Definition of master field list*

The master field list consists of two arrays, `fieldn(2,pflds)` and `iflds(3,pflds)`, 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:

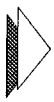
word 1 — Level indicator,
0=single-level,
1=multi-level at interfaces,
2=multi-level at levels

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`.

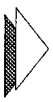
A complete master field list appears in "Appendix D: Master Field List."

*Add diagnostic field*



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.

*Add a constituent*



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 ($Kg/m^2/sec$) 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 qmin 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 qmin and can be reset by the user in the routine INITCOM. The array element qmin(1) is reserved for water vapor.

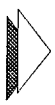
## 4. Changing Resolution

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.

*Parameterization data interface*



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 sepa-

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.

*Accommodating restart*

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 -lQ` 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.

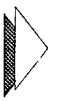
*Running flint*

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*



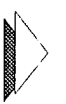
The **-Rb** command line argument to **cft77** turns on array bounds checking. During model execution of code compiled with this option, a message will be printed whenever an array is referenced outside its declared dimensions. This option disables **all** vectorization, resulting in an enormous increase 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 **-c1** 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.

*Comparing history tapes*

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

*Goals of the 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.

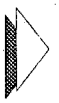
Another debugging tool to be used on the CCM is the Cray bounds checker. There are no constructs in the standard CCM2 code that generate large numbers of "warnings" or errors in using these tools. We strongly recommend that users adhere to this rule in their own code.

### (a) Code Appearance

*Upper/lower case*

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.

*Comments*

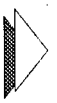


Executable code will be commented to explain the function of the statements following the comments. These comments (one or more lines) will be set off by blank comment lines ("C" in column 1 only). On those comment lines containing text, the text will start in column 3. It is suggested that comments 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.

*Declarative statements*



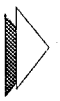
For each program module, following the *subroutine* or *function* statement, a block of comments delineated by "C---..." comment lines will briefly describe the function of the module. Following this, a block of comments labelled "Code `history`" indicates the originator of the current version of the 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

separating common blocks, etc. An "included" file must not itself contain an "include." The *common* variables and *parameter*s 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.

*Delineating declarative statements*



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
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.

*Order of declarative statements*

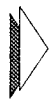
Following the "`Code History`" section, declarative statements should appear in the order *implicit none*, main Model *parameter* statements, *common* statements, type declarations of arguments, local parameters preceded by their type statements, type declarations of local variables, *equivalence* statements, *data* statements, *save* statements, *external* statements and statement functions.

*Continuation character*

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

### (b) Other Fortran Considerations

*I/O statements*

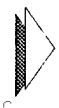


Printed output may be generated using *format* statements or as list-directed output (`write (6,*)`). For efficiency reasons, list-directed output should not be used for printout which is repeated throughout the run. It may be used for diagnostic printout preceding an abnormal termination.

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."

*Fortran literals*

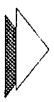


Obscure "magic numbers" should not appear totally undocumented in calculations. These values should be assigned to local variables and commented as to their use.

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

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

*Module and common block naming*



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`."

*Parameter and variable naming*

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.

### (d) Code Structure

*Code indenting*

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 labels*

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.

*Argument lists*

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

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

*Portability issues*

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 callable 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 prarmeterizations 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 managment 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%/SEP1',
  BNDTI = '/CCM2/T42/%data%/tibds',
  BNDTVS = '/CCM2/T42/%data%/tvbds',
  BNDTVO = '/CCM2/T42/%data%/ozn',
  IRT=35,
  NSVSN = 'rstrt',
  NSREST=0,
  NSWRPS='passwd',
  NDENS  =     1,
  NNBDAT = 000901,
  NNBSEC = 0,
  NNDBAS = 0,
  NNSBAS = 0,
  MFILT  =   5,
  DTIME  =  1200.,
  NESTEP =  4,
  NHTFRQ =  4,
  IRAD   =  4,
  IRADAE =  4,
  SSTCYC = .T.,
  OZNCYC = .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
INITAL:HEADER READ AND CHECKED CORRECTLY
1 Layer Locations (*1000)
   1    2.9170            0.0000            2.9170
                4.8093            0.0000            4.8093
   2    7.9292            0.0000            7.9292
               13.0731            0.0000           13.0731
   3   21.5539            0.0000           21.5539
               32.5591            0.0000           32.5591
```

| | | | | | |
|---|---|---|---|---|---|
| 4 | 49.1834 | 0.0000 | 49.1834 | | |
| | 63.9471 | 0.0000 | 63.9471 | | |
| 5 | 83.1425 | 0.0000 | 83.1425 | | |
| | 81.6768 | 17.3664 | 99.0432 | | |
| 6 | 79.9308 | 38.0541 | 117.9849 | | |
| | 78.0201 | 60.6928 | 138.7129 | | |
| 7 | 75.7738 | 87.3088 | 163.0826 | | |
| | 73.3671 | 115.8237 | 189.1908 | | |
| 8 | 70.5752 | 148.9037 | 219.4789 | | |
| | 67.6476 | 183.5918 | 251.2394 | | |
| 9 | 64.2963 | 223.2996 | 287.5959 | | |
| | 60.8624 | 263.9851 | 324.8475 | | |
| 10 | 56.9838 | 309.9406 | 366.9244 | | |
| | 53.1095 | 355.8459 | 408.9554 | | |
| 11 | 48.7913 | 407.0096 | 455.8009 | | |
| | 44.5995 | 456.6760 | 501.2755 | | |
| 12 | 39.9895 | 511.2977 | 551.2872 | | |
| | 35.6607 | 562.5875 | 598.2482 | | |
| 13 | 30.9631 | 618.2465 | 649.2096 | | |
| | 26.7266 | 668.4428 | 695.1694 | | |
| 14 | 22.1902 | 722.1927 | 744.3829 | | |
| | 18.3069 | 768.2030 | 786.5099 | | |
| 15 | 14.2039 | 816.8173 | 831.0212 | | |
| | 10.9421 | 855.4653 | 866.4074 | | |
| 16 | 7.5413 | 895.7590 | 903.3003 | | |
| | 5.1470 | 924.1285 | 929.2755 | | |
| 17 | 2.6838 | 953.3137 | 955.9975 | | |
| | 1.3519 | 969.0938 | 970.4457 | | |
| 18 | 0.0000 | 985.1122 | 985.1122 | | |
| | 0.0000 | 992.5282 | 992.5282 | | |
| 19 | 0.0000 | 1000.0000 | 1000.0000 | | |

reference pressures (Pa)

| | | | | |
|---|---|---|---|---|
| 1 | 291.7000 | | | |
| | | 480.9300 | 501.2200 | |
| 2 | 792.9200 | | | |
| | | 1307.3100 | 1362.4700 | |
| 3 | 2155.3900 | | | |
| | | 3255.9100 | 2762.9500 | |
| 4 | 4918.3400 | | | |
| | | 6394.7100 | 3395.9100 | |
| 5 | 8314.2500 | | | |
| | | 9904.3200 | 3484.2400 | |
| 6 | 11798.4900 | | | |
| | | 13871.2900 | 4509.7700 | |

```
  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  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
     7.596    5.659    4.116    2.877    1.850    1.028
 GRAVITY WAVE EQUIVALENT DEPTHS (M) FOR MEAN STATE =
 11920.920 4117.718 1551.801  731.851  377.519  219.308  131.642   81.575   49.533   29.538   17.501   10.256
     5.883    3.265    1.728    0.844    0.349    0.108
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  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
```

```
   SDS save file A (nra1)              =  11
   SDS save file B (nrb1)              =  21
   History tape buffer (sunit)         =  22
   SDS abs/ems save file (nabem)       =  60
   History tape number  1              =  20
   Restart dataset unit (nsds)         =   7
   Master regeneration unit (nrg)      =   8
   Regeneration dataset units (nrg1)   =   9  10  12  13  14
   Abs/ems unit for restart (nrg2)     =  15  16  17  18  19
```

| | NSTEP | RMSZ | RMSD | RMST | STPS | STQ | COURANT HOR | COURANT VERT |
|---|---|---|---|---|---|---|---|---|
| NSTEP = | 0 | 8.833474687402251E-05 | 7.333832180666193E-06 | 252.716 | 9.84934E+04 | 2.767275591143027E+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 | NLONW |
|---|---|---|---|---|---|---|---|---|---|
| 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 | DATE | TIME | SEQ NO. |
|---|---|---|---|---|
| CURRENT | /JQUSER/ccm2/test/hist/h0001 | 10/02/92 | 12:25:26 | CI9290 |
| FIRST | /JQUSER/ccm2/test/hist/h0001 | 10/02/92 | 12:25:16 | CI9290 |
| INITIAL | /CSM/ccm2/367/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%/tvbds | 12/23/91 | 15:01:43 | - 1 - |
| OZONE BOUNDARY | /CCM2/T42/%data%/ozn | 09/21/92 | 13:34:26 | |

FIELD LIST

| FLD NO. | NAME | FLG. | FLD PT. | PACK. | UNITS |
|---|---|---|---|---|---|
| 1 | PHIS | 0 | 3 | 1 | M2/S2 |
| 2 | PS | 10 | 131 | 1 | PA |
| 3 | T | 12 | 259 | 1 | K |
| 4 | U | 12 | 2563 | 1 | M/S |
| 5 | V | 12 | 4867 | 1 | M/S |
| 6 | Q | 12 | 7171 | 1 | KG/KG |
| 7 | TA01 | 12 | 9475 | 1 | KG/KGS |
| 8 | VD01 | 12 | 11779 | 1 | KG/KGS |
| 9 | DC01 | 12 | 14083 | 1 | KG/KGS |
| 10 | DTH | 12 | 16387 | 1 | K/S |
| 11 | ORO | 0 | 18691 | 1 | FLAG |
| 12 | WET | 10 | 18819 | 1 | M |
| 13 | SNOWH | 10 | 18947 | 1 | M |
| 14 | PRECL | 10 | 19075 | 1 | M/S |
| 15 | PRECC | 10 | 19203 | 1 | M/S |
| 16 | SHFLX | 10 | 19331 | 1 | W/M2 |
| 17 | LHFLX | 10 | 19459 | 1 | W/M2 |
| 18 | QFLX | 10 | 19587 | 1 | KG/M2/S |
| 19 | PBLH | 10 | 19715 | 1 | M |
| 20 | USTAR | 10 | 19843 | 1 | M/S |
| 21 | TPERT | 10 | 19971 | 1 | K |
| 22 | QPERT | 10 | 20099 | 1 | KG/KG |
| 23 | DTV | 12 | 20227 | 1 | K/S |
| 24 | FSNS | 10 | 22531 | 1 | W/M2 |
| 25 | FLNS | 10 | 22659 | 1 | W/M2 |
| 26 | FLNT | 10 | 22787 | 1 | W/M2 |
| 27 | FSNT | 10 | 22915 | 1 | W/M2 |
| 28 | CLOUD | 12 | 23043 | 1 | FRACTION |
| 29 | EFFCLD | 12 | 25347 | 1 | FRACTION |
| 30 | FLNTC | 10 | 27651 | 1 | W/M2 |
| 31 | FSNTC | 10 | 27779 | 1 | W/M2 |
| 32 | FLNSC | 10 | 27907 | 1 | W/M2 |
| 33 | FSNSC | 10 | 28035 | 1 | W/M2 |
| 34 | OMEGA | 12 | 28163 | 1 | PA/S |
| 35 | DQP | 12 | 30467 | 1 | KG/KGS |
| 36 | TAUX | 10 | 32771 | 1 | N/M2 |
| 37 | TAUY | 10 | 32899 | 1 | N/M2 |
| 38 | SRFRAD | 10 | 33027 | 1 | W/M2 |
| 39 | QRS | 12 | 33155 | 1 | K/S |
| 40 | QRL | 12 | 35459 | 1 | K/S |

| | | | | | |
|---|---|---|---|---|---|
| 41 | CLDTOT | 10 | 37763 | 1 | FRACTION |
| 42 | CLDLOW | 10 | 37891 | 1 | FRACTION |
| 43 | CLDMED | 10 | 38019 | 1 | FRACTION |
| 44 | CLDHGH | 10 | 38147 | 1 | FRACTION |
| 45 | TS1 | 10 | 38275 | 1 | K |
| 46 | TS2 | 10 | 38403 | 1 | K |
| 47 | TS3 | 10 | 38531 | 1 | K |
| 48 | TS4 | 10 | 38659 | 1 | K |
| 49 | SOLIN | 10 | 38787 | 1 | W/M2 |
| 50 | UTGW | 12 | 38915 | 1 | M/S2 |
| 51 | VTGW | 12 | 41219 | 1 | M/S2 |
| 52 | TAUGWX | 10 | 43523 | 1 | N/M2 |
| 53 | TAUGWY | 10 | 43651 | 1 | N/M2 |
| 54 | DTCOND | 12 | 43779 | 1 | K/S |
| 55 | CMFDT | 12 | 46083 | 1 | K/S |
| 56 | CMFDQ | 12 | 48387 | 1 | KG/KGS |
| 57 | CMFMC | 11 | 50691 | 1 | KG/M2S |
| 58 | CMFSL | 11 | 52995 | 1 | W/M2 |
| 59 | CMFLQ | 11 | 55299 | 1 | W/M2 |
| 60 | CNVCLD | 10 | 57603 | 1 | FRACTION |

SIGMA VALUES AT FULL LEVELS

| | | | | |
|---|---|---|---|---|
| 4.80930000000002E-03 | 1.30731000000001E-02 | 3.25591000000001E-02 | 6.39471000000000E-02 | 9.90432000000001E-02 |
| 1.38712900000000E-01 | 1.89190800000000E-01 | 2.51239399999999E-01 | 3.24847500000001E-01 | 4.08955400000000E-01 |
| 5.01275499999998E-01 | 5.98248200000000E-01 | 6.95169400000001E-01 | 7.86509900000002E-01 | 8.66407400000000E-01 |
| 9.29275499999999E-01 | 9.70445699999999E-01 | 9.92528200000002E-01 | | |

SIGMA VALUES AT HALF LEVELS

| | | | | |
|---|---|---|---|---|
| 2.91700000000000E-03 | 7.92920000000003E-03 | 2.15539000000000E-02 | 4.91834000000000E-02 | 8.31425000000001E-02 |
| 1.17984900000000E-01 | 1.63082600000000E-01 | 2.19478900000000E-01 | 2.87595899999999E-01 | 3.66924400000000E-01 |
| 4.55800900000000E-01 | 5.51287199999997E-01 | 6.49209599999999E-01 | 7.44382899999998E-01 | 8.31021199999999E-01 |
| 9.03300299999998E-01 | 9.55997499999999E-01 | 9.85112200000000E-01 | 1.00000000000000E+00 | |

LATITUDES

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| -87.86380 | -85.09653 | -82.31291 | -79.52561 | -76.73690 | -73.94752 | -71.15775 | -68.36776 | -65.57761 | -62.78735 |
| -59.99702 | -57.20663 | -54.41620 | -51.62573 | -48.83524 | -46.04473 | -43.25419 | -40.46365 | -37.67309 | -34.88252 |
| -32.09194 | -29.30136 | -26.51077 | -23.72017 | -20.92957 | -18.13897 | -15.34836 | -12.55776 | -9.76715 | -6.97653 |
| -4.18592 | -1.39531 | 1.39531 | 4.18592 | 6.97653 | 9.76715 | 12.55776 | 15.34836 | 18.13897 | 20.92957 |
| 23.72017 | 26.51077 | 29.30136 | 32.09194 | 34.88252 | 37.67309 | 40.46365 | 43.25419 | 46.04473 | 48.83524 |

```
51.62573    54.41620    57.20663    59.99702    62.78735    65.57761    68.36776    71.15775    73.94752    76.73690
79.52561    82.31291    85.09653    87.86380


GAUSIAN WEIGHTS

  1.78328070742408E-03  4.14703326043206E-03  6.50445796886448E-03  8.84675982637578E-03  1.11681394601145E-02
  1.34630478966701E-02  1.57260304760191E-02  1.79517157756741E-02  2.01348231534985E-02  2.22701738083598E-02
  2.43527025686925E-02  2.63774697150316E-02  2.83396726142516E-02  3.02346570723737E-02  3.20579283548366E-02
  3.38051618371289E-02  3.54722132568683E-02  3.70551285402401E-02  3.85501531786030E-02  3.99537411327251E-02
  4.12625632426233E-02  4.24735151236550E-02  4.35837245293276E-02  4.45905581637576E-02  4.54916279274185E-02
  4.62847965813196E-02  4.69681828162118E-02  4.75401657148335E-02  4.79993885964645E-02  4.83447622348090E-02
  4.85754674415084E-02  4.86909570091461E-02  4.86909570091461E-02  4.85754674415084E-02  4.83447622348090E-02
  4.79993885964645E-02  4.75401657148335E-02  4.69681828162118E-02  4.62847965813196E-02  4.54916279274185E-02
  4.45905581637576E-02  4.35837245293276E-02  4.24735151236550E-02  4.12625632426233E-02  3.99537411327251E-02
  3.85501531786030E-02  3.70551285402401E-02  3.54722132568683E-02  3.38051618371289E-02  3.20579283548366E-02
  3.02346570723737E-02  2.83396726142516E-02  2.63774697150316E-02  2.43527025686925E-02  2.22701738083598E-02
  2.01348231534985E-02  1.79517157756741E-02  1.57260304760191E-02  1.34630478966701E-02  1.11681394601145E-02
  8.84675982637578E-03  6.50445796886448E-03  4.14703326043206E-03  1.78328070742408E-03
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  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
    7.596    5.659    4.116    2.877    1.850    1.028
GRAVITY WAVE EQUIVALENT DEPTHS (M) FOR MEAN STATE =
11920.920 4117.718 1551.801  731.851  377.519  219.308  131.642   81.575   49.533   29.538   17.501   10.256
    5.883    3.265    1.728    0.844    0.349    0.108
NSTEP =     1   8.833879376921819E-05  7.377199813813208E-06 252.704  9.84938E+04  2.769625099790608E+01 0.81  0.25
NSTEP =     2   8.834200366937290E-05  7.387113117799499E-06 252.702  9.84942E+04  2.769359722607692E+01 0.81  0.25
NSTEP =     3   8.834552738571852E-05  7.381205123956465E-06 252.696  9.84945E+04  2.770089423633056E+01 0.81  0.25
NSTEP =     4   8.834834211385572E-05  7.408881265554218E-06 252.692  9.84944E+04  2.770266833791698E+01 0.81  0.25


    ***  HEADER FOR CCM2 HISTORY TAPE  ***

    ***   Primary History Tape   ***


 CASE:     f20i
TITLE:    plx20

    LENHDI    MFTYP     MFILH     MFILTH    NRBD      MAXSIZ    NDAVU     MXXX      NLON      NLONW
     217       43         2         5         3      57730     57730         0       128       128
```

| PLAT | PLEV | PTRM | PTRN | PTRK | NFLDH | NSTEPH | NSTPRH | NITSLF | NDBASE |
|---|---|---|---|---|---|---|---|---|---|
| 64 | 18 | 42 | 42 | 42 | 60 | 4 | 0 | 4 | 0 |

| NSBASE | NNDCUR | NSCUR | NBDATE | NBSEC | NCDATE | NCSEC | MDT | MHISF | MFSTRT |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 4800 | 000901 | 0 | 000901 | 4800 | 1200 | 4 | 0 |

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

| | 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 | /CSM/ccm2/367/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%/tvbds | 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/f20i/hist/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
===============================

Command  Started  Elapsed    User CPU   Sys CPU   I/O Wait I/O Wait   CPU Mem  I/O WMem Kwords   Log I/O Memory   Ex
  Name      At    Seconds     Seconds   Seconds   Sec Lck  Sec Unlck  Avg Mwds Avg Mwds Xferred  Request HiWater  St Ni
Fl  GAU's
======== ======== ======== ========== ========== ======== ========== ======== ======== ======== ======= ======= === ==
== =======
ja       14:19:05  0.0790     0.0005     0.0079   0.0004     0.0689   0.0583   0.0579     0.00       1     124   0
24    0.000002
setf     14:22:08  0.0903     0.0014     0.0151   0.0000     0.0695   0.0310   0.0000     0.00       0      64   0
24    0.000005
sh       14:22:08  0.4021     0.0038     0.0396   0.0000     0.0656   0.0551   0.0000     0.00       0     113   0
24    0.000012
ccm.xx.2 14:19:05  2944.8041   161.4891    11.3307   6.6888     0.8330   2.6393   6.2717  8737.00     247   12908   0
24    0.013817
#  1 CPU          55.1104
#  2 CPU          66.2069
#  3 CPU          90.2325
#  4 CPU          90.3232
# SDS                                                                                  89568.00    2368
pshell   14:19:05    727.4056     0.0002     0.0043   0.0000     0.0000   0.0307   0.0000     0.02       4      64   0
24 F  0.000001


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
```

```
Operating System                 : sn1036 sn1036 6.1 cbh.31 CRAY Y-MP
User Name (ID)                   : jquser (xxxx)
Group Name (ID)                  : ncar (100)
Account Name (ID)                : 09013402 (3028)
    Gaus Allocated               :     924.0000
    Gaus Used, as of 09/30/92    :     341.0849
Job Name (ID)                    : CI9290 (79350)
Report Starts                    : 08/13/92 14:19:05
Report Ends                      : 08/13/92 14:31:12
Elapsed Time                     :         2945      Seconds
User CPU Time                    :          161.2276 Seconds
Multitasking Breakdown

(Concurrent CPUs * Connect seconds = CPU seconds)
 ---------------   ---------------   -----------

               1 *          5.1104 =      5.1104
               2 *          3.1034 =      6.2069
               3 *          3.4108 =     10.2325
               4 *          5.0808 =     20.3232

(Concurrent CPUs * Connect seconds = CPU seconds)
      (Avg.)           (total)          (total)
 ---------------      --------------   -----------

            2.51 *         16.7055 =     41.8730

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 Hours            :     0.01197               CPU Charge        :     0.01197
Avg Memory (mwd)     :     6.20541               Memory Charge     :     0.00044
```

```
Disk Activity (mwd)  :     8.53225        Disk Act Charge   :     0.00142
SDS Memory (mwd)     :    12.00000        SDS Memory Charge :   n/a

Charge before Queue Factor       :                          0.01484 GAUs
      (Excluding MSS/NTWK/TAGS)
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

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

*Valid for T42, 18-level, single constituent run.

## Table C.1
## CCM2 Parameter Definitions

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

*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

| Default Tape | Field Name | NPRGM1 Pointer or Common Location* | NPRGTL Pointer | NPRG Pointer | Field Description | NL | A/I | Units |
|---|---|---|---|---|---|---|---|---|
| ✓ | PHIS | | NPHIS | | surface geopotential | 1 | 1 | $m^2 \cdot s^{-2}$ |
| ✓ | PS | NPSM1 | NPSM2 | NPSP1 | surface pressure | 1 | A | $Pa$ |
| ✓ | T | /com3d/ | | | temperature | N | A | $K$ |
| ✓ | U | /com3d/ | | | zonal wind component | N | A | $m \cdot s^{-1}$ |
| ✓ | V | /com3d/ | | | meridional wind component | N | A | $m \cdot s^{-1}$ |
| | ETADOT | | | | Etadot on half levels | N | A | $1/s$ |
| ✓ | Q (or TRxx) | /com3d/ | | | tracer (first is always specific humidity) | N | A | $Kg \cdot Kg^{-1}$ |
| | HAxx | hqfcst (LINEMS) | | | horizontal advection of tracer | N | A | $Kg \cdot Kg^{-1} \cdot s^{-1}$ |
| | VAxx | vqfcst (LINEMS) | | | vertical advection of tracer | N | A | $Kg \cdot Kg^{-1} \cdot s^{-1}$ |
| | DFxx | dqfx3 (LINEMS) | | | SLT fixer of tracer | N | A | $Kg \cdot Kg^{-1} \cdot s^{-1}$ |
| ✓ | TAxx | ta (LINEMS) | | | total advection of tracer | N | A | $Kg \cdot Kg^{-1} \cdot s^{-1}$ |
| ✓ | VDxx | dqv (VDINTR) | | | diffusion tendency of tracer | N | A | $Kg \cdot Kg^{-1} \cdot s^{-1}$ |
| ✓ | DCxx | dqcond (LINEMS) | | | tracer tendency from adjustment physics | N | A | $Kg \cdot Kg^{-1} \cdot s^{-1}$ |
| | TExx | | | | time tendency of tracer | N | A | $Kg \cdot Kg^{-1} \cdot s^{-1}$ |
| | SSxx | | | | tracer source/sinks (pcnst-1 values) | N | A | $Kg \cdot Kg^{-1} \cdot s^{-1}$ |
| | SFxx | | | | tracer surface fluxes (pcnst-1 values) | N | A | $Kg \cdot m^2 \cdot s$ |
| | DUH | NDUHM1 | | NDUHP1 | u horizontal diffusive heating rate | N | I | $K \cdot s^{-1}$ |
| | DVH | NDVHM1 | | NDVHP1 | v horizontal diffusive heating rate | N | I | $K \cdot s^{-1}$ |
| ✓ | DTH | NDTHM1 | | NDTHP1 | T horizontal diffusion | N | A | $K \cdot s^{-1}$ |

* 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 D.1
## Master Field List

| Default Tape | Field Name | NPRGM1 Pointer or Common Location* | NPRGTL Pointer | NPRG Pointer | Field Description | NL | A/I | Units |
|---|---|---|---|---|---|---|---|---|
| | NDQH | NDQHM1 | | NDQHP1 | q horizontal diffusion | N | A | $Kg \cdot Kg^{-1} \cdot s^{-1}$ |
| ✓ | ORO | | NORO | | surface type flag: = 0 for ocean = 1 for land = 2 for sea ice | 1 | I | flag |
| ✓ | WET | | NWS | | soil moisture | 1 | A | $m$ |
| ✓ | SNOWH | | NSN | | water equivalent snow depth | 1 | A | $m$ |
| ✓ | PRECL | precl (LINEMS) | | | large-scale stable precipitation | 1 | A | $m \cdot s^{-1}$ |
| ✓ | PRECC | precc (LINEMS) | | | convective precipitation | 1 | A | $m \cdot s^{-1}$ |
| | PRECSL | | | | large-scale stable snowfall | 1 | A | $m \cdot s^{-1}$ |
| | PRECSC | | | | convective snowfall | 1 | A | $m \cdot s^{-1}$ |
| | RUNOFF | | NRNFP1 | | soil moisture runoff | 1 | A | $m \cdot s^{-1}$ |
| ✓ | SHFLX | shflx (PHYS) | | | surface sensible heat flux | 1 | A | $W \cdot m^{-2}$ |
| ✓ | LHFLX | lhflx (PHYS) | | | surface latent heat flux | 1 | A | $W \cdot m^{-2}$ |
| ✓ | QFLX | cflx (PHYS) | | | surface water flux | 1 | A | $Kg \cdot m^{-2} \cdot s$ |
| ✓ | PBLH | | | | planetary boundary layer height | 1 | A | $m$ |
| ✓ | USTAR | ustar (VDINTR) | | | surface friction velocity | 1 | A | $m \cdot s^{-1}$ |
| | T10 | | | | 10 m potential temperature | 1 | A | $K$ |
| | Q10 | | | | 10 m specific humidity | 1 | A | $Kg \cdot Kg^{-1}$ |
| | CGH | | | | pbl nonlocal transport, heat | N | A | $K \cdot m^{-1}$ |
| | CGQ | | | | pbl nonlocal transport, humidity | N | A | $1 \cdot m^{-1}$ |
| | CGS | | | | pbl nonlocal transport, multiplier | N | A | $s \cdot m^{-2}$ |

## Table D.1
## Master Field List

| Default Tape | Field Name | NPRGM1 Pointer or Common Location* | NPRGTL Pointer | NPRG Pointer | Field Description | NL | A/I | Units |
|---|---|---|---|---|---|---|---|---|
| ✓ | TPERT | | NTPERT | | pbl plume temperature perturbation | 1 | A | $K$ |
| ✓ | QPERT | | NQPERT | | pbl plume moisture perturbation | 1 | A | $Kg \cdot Kg^{-1}$ |
| | KVH | | | | diffusivity for heat | N | A | $m^2 \cdot s^{-1}$ |
| | KVM | | | | diffusivity for momentum | N | A | $m^2 \cdot s^{-1}$ |
| | DUV | | | | U vertical diffusion | N | A | $m \cdot s^{-2}$ |
| | DVV | | | | V vertical diffusion | N | A | $m \cdot s^{-2}$ |
| ✓ | DTV | | | | T vertical diffusion tendency | N | A | $K \cdot s^{-1}$ |
| ✓ | FSNS | fsns (PHYS) | | | net downward solar flux at surface | 1 | A | $W \cdot m^{-2}$ |
| ✓ | FLNS | flns (PHYS) | | | net upward longwave flux at surface | 1 | A | $W \cdot m^{-2}$ |
| ✓ | FLNT | flnt (PHYS) | | | net upward longwave flux at top of model | 1 | A | $W \cdot m^{-2}$ |
| ✓ | FSNT | fsnt (PHYS) | | | net downward solar flux at top of model | 1 | A | $W \cdot m^{-2}$ |
| ✓ | CLOUD | cld (PHYS) | | | cloud fraction | N | A | fraction |
| | SETLWP | | | | prescribed liquid water path | N | A | $g \cdot m^{-2}$ |
| | CLDLWP | | | | cloud weighted liquid water path | N | A | $g \cdot m^{-2}$ |
| ✓ | EFFCLD | effcld (PHYS) | | | effective cloud fraction | N | A | fraction |
| ✓ | FLNTC | flntc (PHYS) | | | net clearsky upward longwave flux at top | 1 | A | $W \cdot m^{-2}$ |
| ✓ | FSNTC | fsntc (PHYS) | | | net clearsky downward solar flux at top | 1 | A | $W \cdot m^{-2}$ |
| ✓ | FLNSC | flnsc (PHYS) | | | net clearsky upward longwave flux at surface | 1 | A | $W \cdot m^{-2}$ |
| ✓ | FSNSC | fsnsc (PHYS) | | | net clearsky downward solar flux at surface | 1 | A | $W \cdot m^{-2}$ |

## Table D.1
## Master Field List

| Default Tape | Field Name | NPRGM1 Pointer or Common Location* | NPRGTL Pointer | NPRG Pointer | Field Description | NL | A/I | Units |
|---|---|---|---|---|---|---|---|---|
| ✓ | OMEGA | omga (LINEMS) | | | vertical pressure velocity | N | A | $Pa \cdot s^{-1}$ |
| ✓ | DQP | qc (LINEMS) | | | Q tendency from rainout | N | A | $Kg \cdot Kg^{-1} \cdot s^{-1}$ |
| ✓ | TAUX | taux (PHYS) | | | zonal surface stress | 1 | A | $N \cdot m^{-2}$ |
| ✓ | TAUY | tauy (PHYS) | | | meridional surface stress | 1 | A | $N \cdot m^{-2}$ |
| ✓ | SRFRAD | | NDRP1 | | radiative flux absorbed at the surface | 1 | A | $W \cdot m^{-2}$ |
| ✓ | QRS | | NQRS | | solar heating rate | N | A | $K \cdot s^{-1}$ |
| ✓ | QRL | | NQRL | | longwave heating rate | N | A | $K \cdot s^{-1}$ |
| ✓ | CLDTOT | cltot (PHYS) | | | random overlap total cloud cover | 1 | A | fraction |
| ✓ | CLDLOW | cllow (PHYS) | | | random overlap low cloud cover | 1 | A | fraction |
| ✓ | CLDMED | clmed (PHYS) | | | random overlap medium cloud cover | 1 | A | fraction |
| ✓ | CLDHGH | clhgh (PHYS) | | | random overlap high cloud cover | 1 | A | fraction |
| | TOTLWP | | | | total liquid water path | 1 | A | fraction |
| ✓ | TS1 | | NTSSUB | | surface temperature (level 1) | 1 | A | $K$ |
| ✓ | TS2 | | NTSSUB | | subsurface temperature, level 2 | 1 | A | $K$ |
| ✓ | TS3 | | NTSSUB | | subsurface temperature, level 3 | 1 | A | $K$ |
| ✓ | TS4 | | NTSSUB | | subsurface temperature, level 4 | 1 | A | $K$ |
| ✓ | SOLIN | solin (PHYS) | | | solar insolation | 1 | A | $W \cdot m^{-2}$ |
| | UTEND | | | | U tendency | N | A | $m \cdot s^{-2}$ |
| | VTEND | | | | V tendency | N | A | $m \cdot s^{-2}$ |
| | TTEND | | | | T tendency | N | A | $K \cdot s^{-1}$ |
| | QTEND | | | | Q tendency | N | A | $Kg \cdot Kg^{-1} \cdot s^{-1}$ |

## Table D.1
## Master Field List

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

# Index

## T



## U



## V



## W



## Z