NCAR/TN-392 + STR
NCAR TECHNICAL NOTE

March 1994

The Penn State/NCAR Mesoscale Model (MM5)

Source Code Documentation

Philip L. Haagenson
Jimy Dudhia
David R. Stauffer
Georg A. Grell

MESOSCALE AND MICROSCALE METEOROLOGY DIVISION

NATIONAL CENTER FOR ATMOSPHERIC RESEARCH
BOULDER, COLORADO

NCAR/TN-392 + STR
NCAR TECHNICAL NOTE

March 1994

The Penn State/NCAR Mesoscale Model (MM5)

Source Code Documentation

Philip L. Haagenson
Jimy Dudhia
David R. Stauffer
Georg A. Grell

MESOSCALE AND MICROSCALE METEOROLOGY DIVISION

NATIONAL CENTER FOR ATMOSPHERIC RESEARCH
BOULDER, COLORADO

TABLE OF CONTENTS

Section 4: COMMON BLOCKS

Section 5: SUBROUTINES

List of Figures

Preface

This technical note describes the Fortran code (version 1) of the PSU/NCAR mesoscale model (MM5). It is intended to provide a detailed documentation of the model code for users who want to understand the code in in great detail or modify it. Version 1 is the first version of MM5 which succeeds a version 8 of MM4. Since the MM5 modeling system is a research tool undergoing constant improvement and modification at both NCAR and the Pennsylvania State University, newer versions of MM5 will continue to be developed. Users who want to know more about the overall MM5 modeling system should obtain a copy of *A User's Guide to the Penn State/ NCAR Mesoscale Modeling System*, by D.O. Gill (1992).

Acknowledgments

The critical review provided by David Gill and Kevin Manning improved the presentation of this document. We also thank Gary Bates and Ronald Errico for their valuable contributions, and are grateful to Mary Ann O'Meara and Debbie Witman for their editorial assistance.

Computing for this model development was undertaken on the Cray Y-MP at NCAR, supported by the NSF. The second author also wishes to acknowledge the Department of Energy for support through grant DEA105-90 ERG1070.

Section 1

INTRODUCTION

This document describes the general program design and organization of the PSU/NCAR mesoscale model (MM5). The basic model was originally developed at the Pennsylvania State University (PSU) and has undergone many changes and improvements at PSU and NCAR since documentation regarding the previous version of the model (MM4) was written (Hsie 1987 and Anthes et al. 1987). Description of the model structure and organization is given in enough detail to enable users to modify the code for their own research. This document should be read with the companion Technical Note *Description of the Fifth Generation Penn. State/NCAR Mesoscale Model (MM5)*, by G. Grell et al. (1994), which describes the governing equations, physical parameterizations, and numerical algorithms used in the model. In the Introduction, a general description of the model and source-code structure is given. Section 2 provides detailed information concerning allocation of memory. Section 3 describes the PARAMETER statements used in the model and section 4 the common block structure. The individual subroutines are discussed in section 5 and the subroutine arguments are defined in section 6. In section 7, variables in the NAMELIST records are defined. Flow charts of MM5 are shown in Appendix A and the UNICOS script from the standard job deck is presented in Appendix B.

1.1 General Description of the Model

The MM5 is a grid-point model with finite differences centered in space and time. Second-order finite differences are used for the advection terms, and an Asselin time filter is applied to all prognostic variables. The model can be either hydrostatic or non-hydrostatic. The hydrostatic option uses split semi-explicit time integration for efficient treatment of the fast gravity modes and the non-hydrostatic option uses semi-implicit time integration for the sound-wave modes.

The vertical coordinate, sigma, is defined as:

$$\sigma = (p - p_t) / p^*, \quad p^* = p_s - p_t ,$$

where p is pressure, p_s is surface pressure, and p_t is the pressure at the top of the model (Fig. 1). For the non-hydrostatic option, p, p_s, and p_t are defined in terms of reference pressures (see Grell et al. 1993). The prognostic variables are p*, p' (pressure perturbation), T, T_g (ground temperature), U, V, W (vertical velocity), q_v (water vapor), q_c (cloud water), q_i (cloud ice), q_r (rain water), and q_{ni} (snow). T_g, q_v, q_c, q_i, q_r, and q_{ni}, are optional; W and p' are for the non hydrostatic option. The variables are staggered horizontally such that U and V are defined on dot points (Fig. 2) and all other variables are defined on cross points. In the vertical, all variables except W are defined on the half-sigma levels.

1.2 Source-Code Structure

The source code of the model is written in Cray FORTRAN and stored on the NCAR mass storage system (MSS) in Cray Update form. The code is both multi-tasked and vectorized. A computer with large in-core memory (larger than one megaword) is needed to run the model because all data, arrays, and prognostic calculations are contained in the main memory of the computer. Locally parameterized dimensions in PARAMETER statements enable users to vary both the horizontal and vertical resolution, and also the number of domains for multi-nest levels. All major two-dimensional and three-dimensional arrays are stored in very large storage arrays (**ALLARR** and **INTALL**) which have parameterized dimensions and are located in common block /HUGE/. Further details concerning the memory and source code structure are given in section 2.

1.3 Options

Most input variables used to control selection of options are read in from four NAMELIST records in subroutine PARAM. Other input parameters regarding options affecting

memory requirements (**IFDDA**, **INHYD**, **MAXNES**, etc.) are specified in PARAMETER statements. Users of MM5 should make sure that certain PARAMETER statements regulating model dimensions (for both coarse and nested domains) are compatible with those previously specified in the preprocessor programs of the MM5 modeling system.

1.4 Style Convention

All source-code variable names are shown in **BOLD UPPER-CASE** font. The names of subroutines, common blocks, and NAMELIST records are shown in standard upper-case font, but not bold upper case. In addition, common block names are bracketed with slashes /NAME/ and NAMELIST record names are delineated with a dollar sign $NAME. Examples of source code or UNICOS script taken directly from the FORTRAN or job deck appear in `typewriter` font.

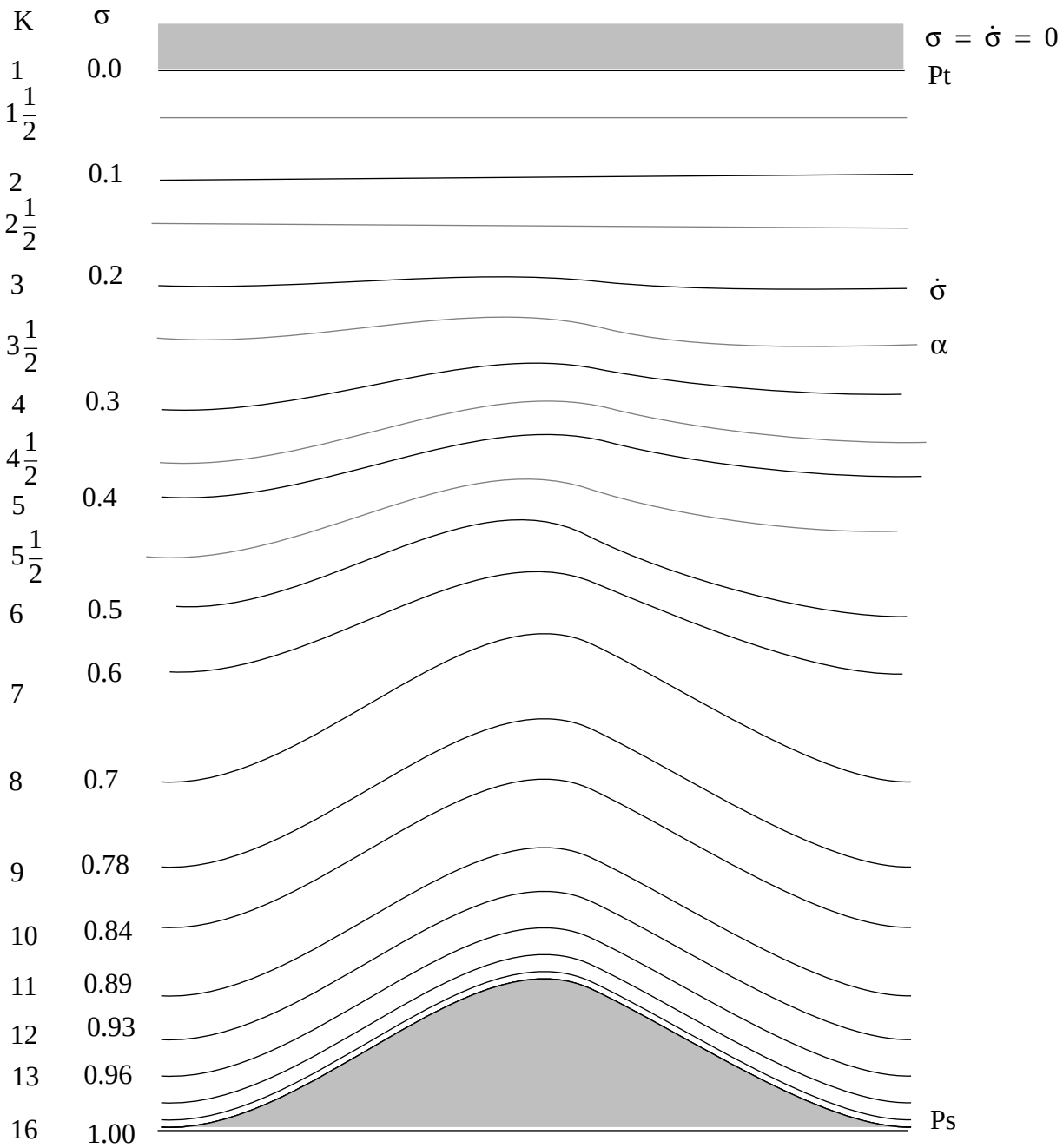

Figure 1. Schematic representation of the vertical structure of the model. The example is for 15 vertical layers. Dashed lines denote half-sigma levels, solid lines denote full-sigma levels.

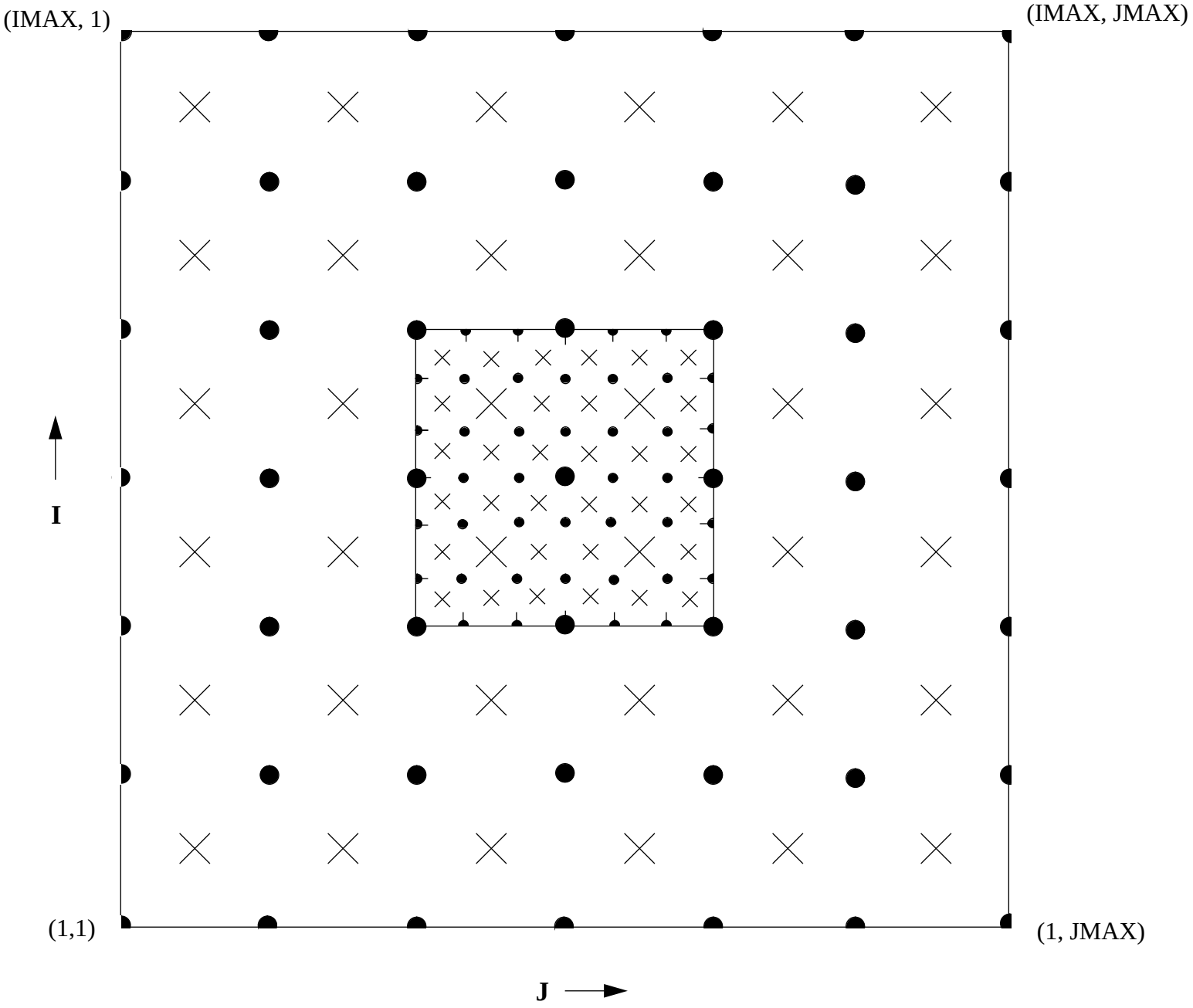

Figure 2. Schematic representation showing the horizontal Arakawa B-grid staggering of the dot (●) and cross (x) grid points. The smaller inner box is a representative mesh staggering for a 3:1 coarse-grid distance to fine-grid distance ratio.

Section 2

MEMORY ORGANIZATION & CODE STRUCTURE

The model is written in standard FORTRAN, but there are some Cray-specific aspects that will be discussed in this section.

2.1 Pointers

MM5 makes use of Cray FORTRAN `POINTERs` to associate model variables with values for a given nest. For instance, three-dimensional array **UA**, appearing in the model physics and dynamics code, is used with a pointer to distinguish which part of the memory it is located in. It has one location for each nest.

`POINTERs` are addresses to identify an array or variable name (the pointee) with a part of the memory, e.g.,

```
POINTER (IAUA, UA(MIX, MJX, MKX))
```

at the top of a subroutine indicates that the location of 3-D array **UA** should be taken to start at address **IAUA** in the memory. Only variables with values that change on different nests require pointers. Physical and model constants and variables local to a subroutine do not have pointers.

Typically in the model the value of **IAUA** and the ~300 other pointers depends upon which nest is being calculated in that part of the program. There would be one value of **IAUA** for each nest and the array **UA** is dimensioned by the maximum **IX** and **JX** of any nest **(MIX**, **MJX**, **MKX)**.

The values of all ~300 (=**NUMVAR**) pointers for each domain are calculated in routine ADDALL at the beginning of a run or restart and stored in array **IAXALL (NUMVAR**, **MAXNES)** where **MAXNES** is the maximum number of nests during the simulation.

When calculations shift from one nest to another, routine ADDRX1C is called to reset the values of **IAUA** etc. for the next nest, taking the values from array **IAXALL**.

Sometimes information from two nests, a coarser and finer nest, are required concurrently such as in initializing nests or calculating nest inputs and feedbacks. In this case ADDRX1N is called to define the pointers for the finer mesh such as

```
POINTER(INAUA, UAN(MIX, MJX, MKX))
```

so that **UA** represents the coarser mesh and **UAN** represents the finer mesh u-component of velocity.

The pointers are divided into 16 common blocks as follows (see section 4 in document for description of the corresponding pointees).

Three-dimensional prognostic variables

```
      COMMON/ADDR1/IAUA, IAUB, IAVA, IAVB, IATA, IATB, IAQVA,
     1        IAQVB, IAQCA, IAQCB, IAQRA, IAQRB, IAQIA, IAQIB,
     2        IAQNIA, IAQNIB
```

Two-dimensional model variables

```
      COMMON/ADDR2/IAPA, IAPB, IARC, IARN, IAF, IAMX, IAMD, IAHT,
     1        IALT, IALO, IALD, IAAL, IATH, IASH, IAZN, IATM,
     2        IAMA, IAEM, IAZO, IAHO, IAMO, IAPL, IARE, IAHX,
     3        IAQX, IAUT, IAPR, IATGA, IATGB, IASA, IAPSI, IARP,
     4        IAGS, IAGL, IACG, IASC, IAEF, IACR, IASR
```

Boundary variables

```
      COMMON/ADDR3/IAUE, IAUW, IAUET, IAUWT, IAVE, IAVW, IAVET, IAVWT,
     1        IATE, IATW, IATET, IATWT, IAQE, IAQW, IAQET, IAQWT,
     2        IAQCE, IAQCW, IAQCET, IAQCWT, IAQRIAQRW, IAQRET,
     3        IAQRWT IAQIE, IAQIW, IAQIET, IAQIWT, IAQNIAQNIW,
     4        IAQNIET,IAQNIWT,IAUN, IAUS, IAUNT, IAUST, IAVN, IAVS,
     5        IAVNT, IAVST, IATN, IATS, IATNT, IATST, IAQN, IAQS,
     6        IAQNT, IAQST, IAQCN, IAQCS, IAQCNT, IAQCST, IAQRN,
     7        IAQRS, IAQRNT, IAQRST, IAQIN, IAQIS, IAQINT, IAQIST,
     8        IAQNIN, IAQNIS, IAQNINT, IAQNIST, IAPE, IAPW, IAPET,
     9        IAPWT, IAPN, IAPS, IAPNT, IAPST, IAUJ1, IAUJL, IAUJ2,
     0        IAUJLX, IAVJ1, IAVJL, IAVJ2, IAVJLX, IAUI1, IAUIL,
     1        IAUI2, IAUILX, IAVI1, IAVIL, IAVI2, IAVILX
```

Other variables (scalars)

```
      COMMON/ADDR4/IAIL, IAJL, IAKL, IAKP1, IAKM, IAILX, IAILM,
     1        IAJLX, IAJLM, IAKTU, IAKTR, IAXT, IADT, IADT2, IADTM,
     2        IADX, IADX2, IADX4, IADX8, IAD16, IADXQ, IAXK, IAXKM,
     3        IAC0, IAC1, IAC3, IADMI, IADMA, IAQMI, IAQMA, IAQME,
     4        IAQMR, IAFNG, IAGNG
```

Nest scalars

```
      COMMON/ADDRNN/INN1, INN2, INN3, INN4, INN5, INN6, INN7, INN8,
     1        INN9, INN10, INN11, INN12, INN13, INN14, INN15, INN16,
     2        INN17, INN18, INN19, INN20, INN21, INN22, INN23, INN24,
     3        INN25, INN26, INN27
```

Split explicit arrays

```
      COMMON/ADDRSP/ISPM, ISPDTA, ISPAM, ISPAN, ISPDS,
     1        ISPHS, ISPZMX, ISPZMR, ISPBZ, ISPCZ, ISPHB
```

FDDA grid-nudging variables

```
      COMMON/ADDR5/IFUBO, IFVBO, IFUBT, IFVBT, IFTBO, IFTBT, IFQBO,
     1        IFQBT, IFVOR
      COMMON/ADDR6/IFPSO, IFPST, IFPSC, IFPSD, IFPSB, IFWDT, IFWCS,
     1        IFWQ, IFDMI, IFDMS, IFIPR, IFIPS, IFMPD, IFMPX
      COMMON/ADDR7/IFSFCO, IFBLWN, IFBLWT, IFBLWS, IFBLPC,
     1        IFBLPD, IFWXY, IFSFT, IFTIB, IFTIE, IFNTB, IFNTE, IFIQC,
     2        IFIDD, IFIDC, IFIDH
```

FDDA observation-nudging variables

```
      COMMON /ADDR8/ IFSVWT, IFVAR, IFRIO, IFRJO, IFRKO, IFTIM, IFERR
```

Three-dimensional nonhydrostatic arrays

```
      COMMON/NONHYD/INHWA, INHWB, INHWTS, INHPPA, INHPPB, INHUTS,
     1        INHVTS, INHPTS
```

Boundary nonhydrostatic arrays

```
      COMMON/NONHYDB/INHWE, INHWW, INHWET, INHWWT, INHPPE, INHPPW,
     1        1INHPPET, INHPPWT, INHWN, INHWS, INHWNT, INHWST,
     2        INHPPN, INHPPS, INHPPST, INHPPNT
```

Nonhydrostatic reference state arrays

```
COMMON/NHCNS/INHT0, INHPS0
```

Navy PBL variables

```
COMMON/ADDRV/IATURK, IATHSB, IATHQW, IAQWSB, IAUTV, IAVTV,
1        IATTV, IAQVTV, IAQCTV
```

Atmospheric radiation array

```
COMMON/RADIAT/IRTT
```

Upper radiative boundary condition

```
COMMON/UPRAD/IUPR
```

This is the sequence of the pointers in **IAXALL** and also of the pointees in the memory where they are EQUIVALENCED to large arrays **ALLARR (IRHUGE**, **MAXNES)** and **INTALL (IIHUGE,MAXNES)**, where **IHUGE** is the sum dimension of all the pointees on a single domain. The arrays **ALLARR** and **INTALL** are written to the save files and read in for restarts, but are not used directly during the calculations. They are stored in /HUGE/ while **IAXALL** is stored in /ADDR0/.

If a user adds variables or arrays with pointers, several changes need to be made,

(i) Add the new common blocks, or add to the existing ones.
(ii) Change **NUMVAR** by the number of new pointers.
(iii) Change **IHUGE** by the total dimension of the new variables (and IIHUGE if new integer).
(iv) Change ADDALL to calculate the starting location of the new variables.
(v) Change ADDRX1C and ADDRX1N to equate the new pointers with the relevant part of **IAXALL**.

2.2 Multi-Tasking

On a multiprocessor computer such as a Cray Y-MP it is possible to perform certain tasks in parallel on different processors and MM5 is written to take advantage of this when available. This is done mostly for the outer **J** loops so that different north-south slices are done by different processors in parallel, i.e. at the same time. This saves wall-clock time but not CPU time. In fact if the tasks on different processors are not well-balanced it may cost more CPU time due to idle waiting time. Much of MM5's calculation is multi-tasked so that on average it uses about 7 of the 8 processors simultaneously.

Multi-tasking therefore differs from vectorization in that it applies to the outer loop while vectorization applies to the inner loop. MM5 is also vectorized as far as possible.

When multi-tasking, some variables have to be declared as SHARED or PRIVATE.

SHARED means that all processors should have access to the same part of the computer memory. If multi-tasking over a **J** loop, arrays that have a **J** in their dimension should normally be shared. Constants that are independent of **J** should also be shared.

PRIVATE means that each processor should have its own private copy of an array with its own memory location that is not available to other processors. Arrays that are dimensioned **(I**, **K)** or **(I)** or **(K)** should be private if they are storing information relevant to only one **J** slice. Arrays that store only local **J** slices should be private. Variables that change their values in the **J** loop should generally be private. Most variables are automatically private unless declared shared, but there are cases where private needs to be declared.

A typical multi-tasking directive in the code is as follows for **K**-loop 210:

```
CMIC$   DO ALL AUTOSCOPE NUMCHUNKS (8)
CMIC$1  PRIVATE (HELP1, HELP2, HSCR1N,HSCR2N, HELP1N, HELP2N)
CMIC$2  SHARED (ISOUTHO, JWESTO, IYY, JXX, IYYN, JXXN,
CMIC$2            ISOUTH, JWEST, IRA)
        DO 210 K=1, KZZ
```

where the '`NUMCHUNKS(8)`' aids efficiency by assigning about 1/8 of the **K** indices to each processor. This works best on a dedicated eight-processor machine such as a Y-MP and with loops where the load is equally balanced among the **K** values, i.e. it is not spending significantly different times on each value of **K**. Loops that are unbalanced, such as those with conditional statements dependent on the loop index, do not use the '`NUMCHUNKS`' directive.

In MM5, multi-tasking takes place in subroutines SOLVE1, SOLVE3 and SOUND. All of the physics routines are called from these routines inside multi-tasked **J** loops, so new physics routines developed by the user need not be multi-tasked themselves. However, care should be taken that shared variables are not changed in the new routines.

Other multi-tasked routines are BDYUV, BDYVAL, FEEDBK, DOTS, STOTNDI, STOTNDT, BDYOVL1, INITNEST and the split-explicit routines DIVG, SPCORT, SPCORU, SPDIFF, SPDIVG, SPGEOP, SPGRAD, SPLITF, SPSTEP2.

On the Cray Y-MP the `cf77` command should include a switch for multi-tasking to recognize the `CMIC$` directives, thus `cf77 -Zu` should be used.

Section 3

PARAMETERS

Section 3.1 lists (alphabetically) and defines the PARAMETERs that are used for defining the dimensions of major arrays in MM5. The PARAMETERs listed in section 3.2 are either used as constants or are for setting the dimensions of minor arrays.

3.1 Major PARAMETERs

IARASC will Arakawa-Schubert cumulus scheme be used, (0=no;1=yes).

IEXMS will explicit moisture scheme be used, (0=no:1=yes).

IFDDA will FDDA be employed, (0=no;1=yes).

IHUGE total first dimension of common block variables **ALLARR** and **INTALL**

(= **MIX**∗**MJX**∗**MKX**∗**NVARX**+**MIXM**∗**MJXM**∗**MKXM**∗**NVARMX**

+**MIX**∗**MJX**∗**NVARSX**+**MIX**∗**MKX**∗**NSPGD**∗8+**MIX**∗**MKX**∗**NSPGX**∗8

+**MIXM**∗**MKXM**∗**NSPGX**∗8+**MJX**∗**MKX**∗**NSPGD**∗8+**MJX**∗**MKX**∗**NSPGX**∗8

+**MJXM**∗**MKXM**∗**NSPGX**∗8+**MIX**∗4∗**NSPGX**+**MJX**∗4∗**NSPGX**

+**MIX**∗**MKX**∗8+**MJX**∗**MKX**∗8+**NVARI**+**NVARP**+**NVARNI**+**NVARNR**+**MEXALL**

+13∗**MIXF**∗**MJXF**∗**MKXF**+(19+**NVAR**) ∗**MIXF**∗**MJXF**

+**NTIM**∗**NVAR**∗**MIXF**∗**MJXF**+5∗**NVAR**+5∗**NIOBF**+4∗1∗**NIOBF**+6∗**NIOBF**

+**NTIM**∗**NVAR**+**NVAR**∗**NCHA**+**NCHA**+**NTIM**

+3∗**MIXNH**∗**MJXNH**∗**KXP1NH**+5∗**MIXNH**∗**MJXNH**∗**MKXNH**

+4∗**MIXNH**∗**MKXNH**∗**NSPGX**+4∗**MJXNH**∗**MKXNH**∗**NSPGX**

+4∗**MIXNH**∗**KXP1NH**∗**NSPGX**+4∗**MJXNH**∗**KXP1NH**∗**NSPGX**

+**MIXNH**∗**MJXNH**∗**MKXNH**+**MIXNH**∗**MJXNH**+9∗**MIXV**∗**MJXV**∗**MKXV**

+4∗**MIXIC**∗**MJXIC**∗**MKXIC**+8∗**MIXIC**∗**MKXIC**∗**NSPGX**

+8∗**MJXIC**∗**MKXIC**∗**NSPGX**+**MIXR**∗**MJXR**∗**MKXR**)

IICE will mixed phase precipitation physics be employed (0=no;1=yes).

IIHUGE first dimension of common block variable **INTALL** (=**NVARI**+**NVARNI**+**NVARNR**+**NSELIT**+**Z***MIXR**+**MJXF**+3***NVAR+NTIM***NVAR**+**NCHA**+**NCHA***NVAR**)

INAV will the Navy PBL be employed, (0=no;1=yes).

INHYD will this simulation be non-hydrostatic (0=no;1=yes).

IRATIO = **IRAX**.

IRAX ratio between coarse grid and fine grid (=3).

IRDDIM will full radiation (LWRAD and SWRAD) be employed (0=no;1=yes).

IRHUGE first dimension of common block variable ALLARR =(**IHUGE-IIHUGE**)

KX = **MKX**.

KXP1 = **MKX**+1.

KXP1NH = (**KXP1**-1)∗**INHYD**+1.

MAXNES maximum number of domains.

MAXSES maximum number for **MAXNES** (default value = 10).

MEXALL = **NSPLIT**∗3+**MKX**∗**NSPLIT**+**MIX**∗**MJX**∗**NSPLIT**∗2+3∗**MKX** ∗**MKX**∗**MKX**∗**KXP1**+**MKX.**

MIX maximum dimension (all domains) in the y-direction.

MIXA maximum dimension of Arakawa-Schubert moisture arrays in y-direction (= (**MIX**-1) ∗**IARASC**+1).

MIXF = (**MIX**-1) ∗**IFDDA**+1.

MIXIC = (**MIX**-1)∗**IICE**+1.

MIXM maximum dimension of moisture arrays in y-direction (= (**MIX**-1)∗**IEXMS**+1)

MIXNH = (**MIX**-1)∗**INHYD**+1.

MIXR = (**MIX**-1)∗**IRDDIM**+1.

MIXV = (**MIX**-1)∗**INAV**+1 (for Navy PBL).

MJX maximum dimension (all domains) in the x-direction.

MJXA maximum dimension of Arakawa-Schubert moisture arrays in x-direction (= (**MJX**-1)∗**IARASC**+1).

MJXF = (**MJX**-1)∗**IFDDA**+1.

MJXIC = (**MJX**-1)∗**IICE**+1.

MJXM maximum dimension of moisture arrays in x-direction (= (**MJX**-1) ∗**IEXMS**+1)

MJXNH = (**MJX**-1) ∗**INHYD**+1.

MJXR = (**MJX**-1) ∗**IRDDIM**+1.

MJXV = (**MJX**-1) ∗**INAV**+1 (for Navy PBL).

MKX maximum dimension (all domains) in the vertical direction.

MKXA maximum dimension of Arakawa-Schubert moisture arrays in vertical direction (= (**MKX**-1) ∗**IARASC**+1).

MKXF = (**MKX**-1) ∗**IFDDA**+1.

MKXIC = (**MKX**-1) ∗**IICE**+1.

MKXM maximum dimension of moisture arrays in vertical direction (= (**MKX**-1) ∗**IEXMS**+1).

MKXNH = (**MKX**-1) ∗**INHYD**+1.

MKXR = (**MKX**-1) ∗**IRDDIM**+1.

MKXV = (**MKX**-1) ∗**INAV**+1 (for Navy PBL).

NCHA maximum number of changes (one variable, one date) when using **INOPRO** option.

NIOBF maximum number of observations within observation-nudging time window (= 10000-1) ∗**IFDDA**+1).

NLNES maximum number of domain levels.

NSPGD number of dot-point slices affected by sponge or relaxation boundary conditions = 5).

NSPGM = **NSPGX**∗8.

NSPGX number of cross-point slices affected by sponge or relaxation boundary conditions (= 5).

NSPLIT number of vertical modes to be used in the split-explicit scheme (=2).

NTIM maximum number of surface-analysis times within analysis-nudging period.

NUMPROGS space allocation for record-header information (for all the modeling system programs).

NUMVALS space allocation for record-header variable definitions.

NUMVAR = **NVARX**+**NVARMX**+**NVARSX**+72+**NVARI**+**NVARP**+27+11+46+26+9+20+1.

NVAR maximum number of variables stored in array **SFCOBS** (= 8).

NVARI number of integer data stored in common /ADDR4/ (= 11).

NVARMX number of prognostic 3-D water-substance variables stored in common /ADDR1/ (= 4).

NVARNI number of nested integer variables (= 21).

NVARNR storage space for nesting (= 6∗**MAXSES**).

NVARP number of real data stored in common /ADDR4/ (= 23).

NVARSX Number of 2-D variables stored in common /ADDR2/ (= 39).

NVARX number of prognostic 3-D variables **UA**, **UB**, **VA**, **VB**, **TA**, **TB**, **QVA**, **QVB** (=8).

3.2 Minor PARAMETERs

G gravitational acceleration (= 9.8 m s^{-2}).

GAMMA standard atmosphere lapse rate (= 6.5E-3 K m^{-1}).

IA = **MBOTH**+2.

IFCT a parameter used for the positive-definite advection scheme (= 1).

IOR a parameter used for the positive-definite advection scheme (= 2).

IW1 = 2∗(**NIMSL**+**MEQU**)+3.

IWW = **IA**∗(**IA**+2)+2∗(**NIMSL**+**MEQU**).

KDIM = **MKX**.

KNUM number of cloud types (= 6 in ARASCH and = 1 in SHALLOW).

KNUMS dimension of **XMB** (= 1).

MBOTH = **MEQU**+**MNEQU**.

MEQU = 0 (needed for ZX4LP).

MINP = **MEQU**+**NIMSL**+2.

MNEQU = **KNUM**.

N1 = **MJX**.

N2 = **MIX.**

N1OS = **N1**∗**NONOS**+1-**NONOS**.

N2OS = **N2**∗**NONOS**+1-**NONOS**.

NCDIAG = (**NIJMAX**+1) ∗**NK**∗4.

NF number of nested grid points used for interpolation to coarse-grid location (= 9).

NIJMAX = larger of **MIX** or **MJX** for split-explicit - (currently set to **MJX**).

NIMSL = 2∗**MNEQU**.

NK = **KX**.

NK1 = **NK**+1.

NONOS = 1 (used in SINT).

NVERTS 7+8∗**NK**+7∗**NK**∗**NK**+**NK1**∗**NK1**.

P0 standard atmosphere sea-level pressure (= 101.325 cb).

PCONST constant added to sea-level pressure (= 10 mb).

R gas constant for dry air (= 287.04 J kg^{-1} K^{-1}).

RATE standard atmosphere lapse rate (= 6.5E-3 K m^{-1}).

RGAS gas constant for dry air (= 287.04 J kg^{-1} K^{-1}).

T0 reference temperature in VTLAPS (= 288.15 K).

TC reference temperature in SEAPRS (= 273.16+17.5 K).

T0L tolerance allowed in VCHEKE (= 1.E-9).

TSTRAT reference temperature for stratosphere in VTLAPS (= 218.15 K).

XKAPPA = R / c_p (= .287).

ZSTRAT reference height for stratosphere in VTLAPS (= 10769 m).

Section 4

COMMON BLOCKS

This section first describes common blocks that are not associated with pointer addresses (4.1) and then those that are associated with pointers (4.2). The name of each common block (listed alphabetically) is followed by a brief description of the block's general contents. The definition, dimension, and units for all variables (or constants) within each common block are also given in alphabetical order along with the beginning pointer > address for those having pointers.

4.1 Common Blocks Without Pointers

4.1.1 ADDR0

/ADDR0/ holds a storage array for pointer addresses.

IAXALL (**NUMVAR**, **MAXNES**): storage array for pointer addresses.

4.1.2 ARASCH1

/ARASCH1/ holds arrays used in the Arakawa-Schubert convection scheme.

KDT (**MIXA, MJXA, MAXNES**); time-step counter denoting how long convection has been active.

OUTTQ (**MIXA**, **MJXA**, **MKXA**, **MAXNES**); q_v tendency (kg kg^{-1} s^{-1}).

OUTTT (**MIXA**, **MJXA**, **MKXA**, **MAXNES**); T tendency (K s^{-1}).

PRETT (**MIXA**, **MJXA**, **MAXNES**); rainfall rate (mm s^{-1}).

4.1.3 ASSEL

/ASSEL/ stores constants for the Asselin filter.

XMUT (**MIX**, **MJX**, **MAXNES**); constants for Asselin filter.

XMUU (**MIX**, **MJX**, **MAXNES**); constants for Asselin filter.

XNUT (**MIX**, **MJX**, **MAXNES**); constants for Asselin filter.

XNUU (**MIX**, **MJX**, **MAXNES**); constants for Asselin filter.

4.1.4 BLANK

Two BLANK common blocks contain matrix operation arrays and some other variables used for the split-explicit scheme.

A (**KX**, **KX**); a matrix operator used in the calculation of array **AM** for vertical modes.

A1 (**KX**, **KX**); local array in VMODES that holds a variable used in the matrix operations.

A2 (**KX**, **KX**); local array in VMODES that holds a variable used in the matrix operations.

A3 (**KX**, **KX**); local array in VMODES that holds a variable used in the matrix operations.

A4 (**KX**, **KX**); local array in VMODES that holds a variable used in the matrix operations.

ALPHA1 = **HYDROS**∗**TBARH**/**PS** (K cb^{-1}) in Daley's variational scheme.

ALPHA2 = **HWEIGH** in Daley's variational scheme.

CPFAC (**MKX**); summed sigma-weighted inverse of the **TAU** matrix (kg J^{-1}) for vertical modes.

D1 (**KX**, **KX**); local array in VMODES that holds a variable used in the matrix operations.

D2 (**KX**, **KX**); local array in VMODES that holds a variable used in the matrix operations.

DSIGMA (**MKX**); thickness of the sigma layer.

E1 (**KX**, **KX**); local array in VMODES that holds a variable used in the matrix operations.

E2 (**KX**, **KX**); local array in VMODES that holds a variable used in the matrix operations.

E3 (**KX**, **KX**); local array in VMODES that holds a variable used in the matrix operations.

G1 (**KX**, **KX**); local array in VMODES that holds a variable used in the matrix operations.

G2 (**KX**, **KX**); local array in VMODES that holds a variable used in the matrix operations.

G3 (**KX**, **KX**); local array in VMODES that holds a variable used in the matrix operations.

HWEIGH (**MKX**); a weighting factor for temperature in Daley's variational scheme.

HYDROC (**MKX**, **KXP1**); hydrostatic matrix (K) for vertical modes used as a multiplier of the log (**SIGMA**∗**P**S+**PT**) vector.

HYDROR (**MKX**, **MKX**); matrix inverse of **HYDROS** for vertical mode calculations.

HYDROS (**MKX**, **MKX**); hydrostatic matrix for vertical modes used as a multiplier of the T vector.

PD = **PS - PTOP**, where **PS** is the average value of p*.

PS p* (cb) averaged.

PT top of model (cb).

R gas constant for dry air (= 287.04 J kg^{-1} K^{-1}).

S1 (**KX**, **KX**); local array in VMODES that holds a variable used in the matrix operations.

S2 (**KX**, **KX**); local array in VMODES that holds a variable used in the matrix operations.

SIGMAH (**KXP1**); half-sigma levels.

TAU (**KX**, **KX**); local matrix variable in VMODES that is coupled with the gas constant and has units (J kg^{-1}).

TBARF (**KXP1**); average temperature (K) on the full-sigma levels.

TBARH (**MKX**); average temperature (K) on the half-sigma levels.

T**HETAF** (**KXP1**); average potential temperature (K) on the full-sigma levels.

THETAH (**KX**); average potential temperature (K) on the half-sigma levels.

TWEIGH (**KX**); sigma weighted **TBARH** (K).

VARPA1 (**MKX**, **KXP1**); array used in Daley's variational scheme for determination of p_s changes ($cb^2\ K^{-1}$).

VARPA2 (**KXP1**, **KXP1**); array used in Daley's variational scheme for determination of p_s changes (cb^2).

W1 (**KX**, **KX**); local array in VMODES that holds a variable used in the matrix operations.

W2 (**KX**, **KX**); local array in VMODES that holds a variable used in the matrix operations.

W3 (**KXP1**, K**X**); local array in VMODES that holds a variable used in the matrix operations.

X1 (**KX**, **KX**); local array in VMODES that holds a variable used in the matrix operations.

4.1.5 CFD

/CFD/ contains scratch arrays used for FDDA.

BLDUM2D (**MIXF**, **MJXF**); scratch array used for analysis nudging.

DUM2D (**MIXF**, **MJXF**); scratch array used for analysis nudging.

SCR2D (**MIXF**, **MJXF**); scratch array used for analysis nudging.

4.1.6 CFDDAGD

/CFDDAGD/ contains nudging coefficients and weighting factors for FDDA. (Second index varies over type of analysis nudging: 1 = 3-D analysis nudging, 2 = surface analysis nudging within PBL).

GP (**MAXSES**, 2); analysis nudging coefficient (s^{-1}) for p*.

GQ (**MAXSES**, 2); analysis nudging coefficient (s^{-1}) for mixing ratio.

GR (**MAXSES**); analysis nudging coefficient (m^2 s^{-1}) for vorticity.

GT (**MAXSES**, 2); analysis nudging coefficient (s^{-1}) for temperature.

GV (**MAXSES**, 2); analysis nudging coefficient (s^{-1}) for wind.

TFAC (**MAXNES**, 2); temporal weighting factor for analysis nudging.

ZFAC (**MAXNES**, 2, **MKXF**); vertical weighting factor for analysis nudging.

4.1.7 DEPAR2

/DEPAR2/ stores arrays used for the finer-domain boundary interpolations.

IG0 (**MJX**, **MIX**); grid-point **I** value.

JG0 (**MJX**, **MIX**); grid-point **J** value.

XIG (9); **I**-direction distance in grid-point units from finer to coarser grid point.

XJG (9); **J**-direction distance in grid-point units from finer to coarser grid point.

4.1.8 HEADER

/HEADER/ contains arrays holding record header information.

MIF (**NUMVALS**, **NUMPROGS**); integer data arrays of record header information.

MRF (**NUMVALS**, **NUMPROGS**); real data arrays of record header information.

4.1.9 HEADERC

/HEADERC/ contains arrays providing character descriptions of record information.

MIFC (**NUMVALS**, **NUMPROGS**); character descriptions of the integer variables in **MIF**.

MRFC (**NUMVALS**, **NUMPROGS**); character descriptions of the real variables in **MRF**.

4.1.10 HUGE

/HUGE/ holds the main storage array for MM5.

ALLARR (**IRHUGE**, **MAXNES**); stores all real variables needed for restart.

INTALL (**IIHUGE, MAXNES**); stores all integer variables needed for restart.

4.1.11 JRG

/JRG/ contains constants that are used for the mixed phase explicit-moisture routine (EXMOISR).

ACRCR used in collection of cloud water by rain (s^{-1} m^{-3}-**BVT**).

ACRCS used in collection of cloud water by snow (s^{-1} m^{-3}-**BVTS**).

ACRIS used in collection of cloud ice by snow (s^{-1} m^{-3}-**BVTS**).

AP used in equation for heterogeneous freezing of cloud droplets (= .66 K).

ATO constant in Fletcher's formula (K^{-1}).

BACRCR used in collection of cloud water by rain.

BACRCS used in collection of cloud water by snow.

BACRIS used in collection of cloud ice by snow.

BP used in equation for heterogeneous freezing of cloud droplets (= 100 m^{-3} s^{-1}).

BR used in formula for fall speed of rain.

BS used in formula for fall speed of snow.

CNP used in equation for heterogeneous freezing of cloud droplets (= 1.E10 m^{-3}).

DEPR1 used for evaporation of rain (m^{-4}).

DEPR2 used for evaporation of rain (m s^{-1}).

DEPR3 used for evaporation of rain.

DEPR4 used for evaporation of rain.

DEPS1 used for sublimation of snow (m^{-4}).

DEPS2 used for sublimation of snow ($m\ s^{-1}$).

DEPS3 used for sublimation of snow.

DEPS4 used for sublimation of snow.

DRAIN density of rain (= 1000 kg m^{-3}).

FRAIN used in fall speed formula ($s^{-1}\ m^{1-\mathbf{BVT}}$).

FSNOW used in fall speed formula ($s^{-1}\ m^{1-\mathbf{BVTS}}$).

HGFR value at which homogeneous freezing of cloud water occurs (= 233 K).

PMS1 used in the equation for melting of snow (m^{-4}).

PMS2 used in the equation for melting of snow ($m\ s^{-1}$).

PMS3 used in the equation for melting of snow.

PMS4 used in the equation for melting of snow.

TOPR top of slope parameter in Marshall-Palmer size distribution for rain (kg m^{-7}).

TOPS top of slope parameter in Marshall-Palmer size distribution for snow (kg m^{-7}).

TNO constant in Fletcher's formula (m^{-3}).

XM01 used in formula for initiation of cloud ice (kg).

XSMAX used in conversion of cloud ice to snow (kg).

4.1.12 LANDUSE

/LANDUSE/ contains data for surface land use categories.

ALBD (13, 2); Albedo (percent).

SCFX (13); Scale factor for calculating the new albedo due to snow cover effects.

SFEM (13, 2); Surface emissivity (fraction).

SFHC (13, 2); Surface heat capacity per unit volume ($J\ m^{-3}\ K^{-1}$).

SFZ0 (13, 2); Surface roughness length (cm).

SLMO (13, 2); Soil moisture availablity (fraction).

THERIN (13, 2); Thermal inertia (0.01 cal cm^{-2} K^{-1} $s^{-0.5}$)

4.1.13 MIC

/MIC/ contains prognostic variables at time t+1, tendencies, and scratch arrays.

HSCR1 (**MIX**, **MJX**); a 2-D scratch (**I**, **J**) array.

HSCR2 (**MIX**, **MJX**); a 2-D scratch (**I**, **J**) array.

HSCR3 (**MIX**, **MJX**); a 2-D scratch (**I**, **J**) array.

HSCR4 (**MIX**, **MJX**); a 2-D scratch (**I**, **J**) array.

IPTN (**MJX**); used for determining the noise ratio for hydrostatic runs.

PHI (**MJX**, **MIX**, **MKX**); geopotential (m^2 s^{-2}).

PSC (**MJX**, **MIX**); p* at time t+1 (cb).

PTEN (**MIX**, **MJX**); the tendency of p* (cb s^{-1}).

PTNTOT (**MJX**); pressure tendency (cb s^{-1}).

PT2TOT (**MJX**); time derivative of the pressure tendency (cb s^{-2}).

QCC (**MJXM**, **MIXM**, **MKXM**); p^*q_c at time t+1 (cb kg kg^{-1}).

QDOT (**MJX**, **MIX**, **KXP1**); vertical sigma velocity, $d\sigma/dt$ (s^{-1}). For the nested domain, the boundary values are computed from coarser-domain tendencies.

QIC (**MJXIC**, **MIXIC**, **MKXIC**); p^*q_i at time t+1 (cb kg kg^{-1}).

QNIC (**MJXIC**, **MIXIC**, **MKXIC**); p^*q_{ni} at time t+1 (cb kg kg^{-1}).

QRC (**MJXM**, **MIXM**, **MKXM**); p^*q_r at time t+1 (cb kg kg^{-1}).

QVC (**MJXM**, **MIXM**, **MKXM**); p^*q_v at time t+1 (cb kg kg^{-1}).

TC (**MJX**, **MIX**, **MKX**); p*T at time t+1 (cb K).

UC (**MJX**, **MIX**, **MKX**); p*U at time t+1 (cb m s^{-1}).

UCC (**MJX**, **MIX**, **MKX**); temporary p*U tendency (cb m s^{-2}).

UCD (**MJX**, **MIX**, **MKX**); temporary p*U tendency (cb m s^{-2}).

VC (**MJX**, **MIX**, **MKX**); p*V at time t+1 (cb m s^{-1}).

VCC (**MJX**, **MIX**, **MKX**); temporary p*V tendency (cb m s^{-2}).

VCD (**MJX**, **MIX**, **MKX**); temporary p*V tendency (cb m s^{-2}).

4.1.14 NESLEV

/NESLEV/ holds variables providing the finer-domain information.

LEVIDN (**MAXSES**); nest level.

NSTTOT total number of active domains.

NUMLV (**NLNES**, **MAXSES**); total number of active domains on given nest level.

NUMNC (**MAXSES**); identifier of mother domain.

4.1.15 NHCNST

/NHCNST/ stores variables (or constants) needed for the nonhydrostatic option.

BET beta parameter for time-averaging in implicit scheme (= 0.4).

DTS short time step (s) in SOUND < (**DX**/(speed of sound)).

GAMMA ratio of heat capacities c_p / c_v.

ISTEP number of short time steps (**DTS**) per leapfrog step (= 4).

P0 reference sea-level pressure (Pa).

TLP lapse rate of reference temperature (K) with respect to log (pressure).

TS0 reference sea-level temperature (K).

4.1.16 NHTENS

/NHTENS/ contains slow tendencies for SOUND routine.

PPTENS (**MIXNH**, **MJXNH**, **MKXNH**); p' slow tendency (Pa s^{-1}).

UTENS (**MIXNH**, **MJXNH**, **MKXNH**); U velocity slow tendency (m s^{-2}).

VTENS (**MIXNH**, **MJXNH**, **MKXNH**); V velocity slow tendency (m s^{-2}).

WTENS (**MIXNH**, **MJXNH**, **KXP1NH**); W slow tendency (m s^{-2}).

4.1.17 PARAM2

/PARAM2/ contains logicals and integers pertaining to selection of model options.

ALBLND albedo over land, used when **ISFPAR** = 0.

BDYTIM time (min) after which boundary conditions are needed or the final time of the present boundary conditions.

IABSOR sponge absorber at top of model, (0=no;1=yes).

IACTIV (**MAXSES**); is nested domain active, (0=no;1=yes).

IBLTYP (**MAXSES**); will bulk PBL or Blackadar PBL be used in the model:
= 0; frictionless.
= 1; bulk PBL.
= 2; multi-level Blackadar PBL.

IBMOIST will initial and boundary conditions be provided for water/ice variables, (0=no;1=yes).

IBOUDY (**MAXSES**); indicates type of lateral boundary conditions:
= 0; fixed.
= 1; relaxation.
= 2; time dependent (from observations or large-scale model).
= 3; time and inflow/outflow dependent. Nonhydrostatic boundary contitions.
= 4; sponge.

ICDCON (**MAXSES**); are drag coefficients constants when using bulk PBL,

(0=no;1=yes--function of terrain height only).

ICLOUD (**MAXSES**); will the radiation effects due to clouds be considered. Used if surface heat and moisture fluxes are calculated (**ISFFLX**=1) and ground temperature is predicted from the budget (**ITGFLG**=1), (0=no;1=yes).

ICOR3D will full Coriolis force including vertical component be considered (nonhydrostatic option only), (0=no;1=yes).

ICUPA (**MAXSES**); what type of cumulus parameterization will be employed:
= 1; none.
= 2; Anthes-Kuo scheme.
= 3; Grell scheme.
= 4; Arakawa-Schubert scheme.

ICUSTB will the stability check in the Kuo cumulus parameterization scheme be activated, (0=no;1=yes).

IDRY (**MAXSES**); is this run a moist or dry forecast, (0=moist;1=dry).

IEXICE will explicit moisture scheme with ice-physics effects be used, (0=no, 1=yes).

IFDRY is this a fake dry run, (no latent heat release) (0=no;1=yes).

IFEED feedback option:
= 0; one-way.
= 1; MM4 method.
= 2; no smoothing.
= 3; light smoothing.

IFPRT is printer output desired, (0=no;1=yes).

IFRAD will radiative cooling of the atmosphere be considered:
= 0; no.
= 1; use simple radiation routine.
= 2; use full radiation (LWRAD and SWRAD).

IFREST is this run restarted from a saved file, (.T. or .F.).

IFSAVE will a saved file will be written for restart, (.T. or .F.).

IFSNOW (**MAXSES**); will snow-cover data be considered, (0=no;1=yes).

IFTAPE will output be saved on files for INTERP, (0=no;1=yes).

IFUPR will upper radiative boundary conditions be used, (0=no;1=yes).

IMOIAV (**MAXSES**); is moisture availability a function of time, (0=no;1=yes).

IMOIST (**MAXSES**); will cumulus parameterization or explicit moisture be used.
= 0; dry case with passive, moisture variables (including q_v).
= 1; no explicit moisture.
= 2; explicit moisture.

IMOVCO (**MAXSES**); counter for how often nested domain is moved.

IMOVE (**MAXSES**); will nested domain be moved, (0=no;1=yes).

IMOVEI (**MAXSES**, 10); number of grid points to move nested domain in the **I**-direction.

IMOVEJ (**MAXSES**, 10); number of grid points to move nested domain in the **J**-direction

IMOVET (**MAXSES**, 10); when in minutes will nested domain be moved.

I**MVDIF** will moist-adiabatic vertical diffusion in clouds be included, (0=no;1=yes).

IOVERW (**MAXSES**); will interpolated nested domain be overwritten with user's own analysis, (0=no;1=yes).

ISFFLX (**MAXSES**); will surface heat and moisture fluxes be calculated, (0=no;1=yes).

ISFOUT will output of surface/terrain parameters be printed (**IFPRT**=1), (0=no;1=yes).

ISFPAR (**MAXSES**); are surface/land-use parameters variable or constant. Used only if (**ISFFLX** = 1 and **ITGFLG**=1), (0=constant;1=variable).

ISHALLO (**MAXSES**); will shallow convection be used, (0=no;1=yes).

ITGFLG (**MAXSES**); indicates the method for calculating ground temperature (**ISFFLX**=1):
= 1; it will be calculated from the budget.
= 2; it will be calculated from a sinusoidal function. (not available)
= 3; it will be determined from specified constants.

ITQPBL tendencies at the boundaries will be computed in HIRPBL (**IBLTYP**=2) when **IBOUDY** = 1, 2, 3, or 4, (0=no;1=yes).

IVMIXM (**MAXSES**); will vertical mixing of momentum be considered (**IBLTYP**=2), (0=no;1=yes).

IXTIMR restart time (min) into forecast.

MASCHK time-step frequency for printout of mass conservation information.

NESTI (**MAXSES**); origin location in mother domain of nested domain in **I**-direction.

NESTIX (**MAXSES**); **I**-dimension of nested domain.

NESTJ (**MAXSES**); origin location in mother domain of nested domain in **J**-direction.

NESTJX (**MAXSES**); **J**-dimension of nested domain.

NTRAD (**MAXSES**); number of time steps between radiation computations, (=**RADFRQ** / **DTMIN**).

NTSAVE number of times output has been written to a file for restart (**IFSAVE**=.T.) and the run is restarting from a saved file (**IFREST**=.T.).

NTTAPE number of times output for INTERP has been written, (**IFTAPE**=1;**IFREST**=.T.).

PRTFRQ interval (min) for printer output (**IFPRT**=1).

PRTTIM time (min) for printer output (computed from **PRTFRQ**).

RADFRQ frequency (min) that solar radiation is computed (**ISFFLX**=1;**ITGFLG**=1).

SAVFRQ interval (min) between save operations (**IFSAVE**=.T.).

SAVTIM time (min) for saving data for restart (computed from **SAVFRQ**).

TAPFRQ interval (min) of output data for INTERP (**IFTAPE**=1).

TAPTIM time (min) for outputting data for INTERP (computed from **TAPFRQ**).

TBDYBE initial time (min) of the present boundary conditions, output from **BDYIN.**

THINLD thermal inertia over land when **ISFPAR** = 0 (cal cm^{-2} K^{-1} $s^{-1/2}$).

XENNES ending time of computations for a given nested domain.

XMAVA moisture availability, when **ISFPAR** = 0.

XSTNES beginning time of computations for a given nested domain.

ZZLND roughness length (m) over land, when **ISFPAR** = 0.

4.1.18 PARAM3

/PARAM3/ holds constants used for physical processes and indices used to specify grid-point location or vertical level.

A (**MKX**); half-sigma levels where (**A** (**K**)=0.5∗ (**SIGMA** (**K**+1)+**SIGMA** (**K**))).

ALPHA constant used in Brown and Campana (1978) scheme (= 0.2495).

BETA constant used in Brown and Campana scheme (= 1.-2.∗**ALPHA**).

CD exchange coefficient for momentum over land (= 0.002).

CDSEA exchange coefficient for momentum over water (= 0.0015).

CH exchange coefficient for heat over land (= 0.002).

CHSEA exchange coefficient for heat over water (= 0.0015).

CP specific heat at constant pressure for dry air (= 1004 J kg^{-1} K^{-1}).

DECLIN solar declination angle for each day of the forecast (radians).

DECTIM time (in min) after which solar declination must be recalculated.

DEGRAD conversion factor from degrees to radians (= $2 * \pi$ / 360 = 0.0174533).

DELTMX maximum time step (s) allowed in the model (= **DT**).

DPD constant to compute the longitude of the sun measured in the ecliptic plane from the vernal equinox (Julian day = 81) (= 360/365=0.986301).

DSIGMA (**MKX**); thickness of sigma levels (i.e. **SIGMA** (**K**+1) - **SIGMA** (**K**)).

EOMEG earth's angular velocity (= 7.2722E-5 s^{-1}).

G gravity (= 9.8 m s^{-2}).

GMT Greenwich Mean Time of the initial data (hours).

GNU constant used in Asselin time filter for all prognostic variables (= 0.1).

GNUHF = **GNU.**

ICON (**MJX**); counter for points at which cumulus parameterization is invoked.

ICONS (**MJX**); number of shallow convection points.

IRDTT a counter for keeping track the total number of free convective points (in HIRPBL) which exceed certain critical values.

ISKF number of input files to be skipped at the beginning of a simulation.

ISPGD number of dot-point slices on the boundary affected by sponge or relaxation boundary conditions (=**NSPGD**-1).

ISPGX same as **ISPGD** but for cross points (=**NSPGX**-1).

JOUT index for variables at time t+1 (temporary storage for variable transfer).

JULDAY Julian date of the initial data set.

JXSEX (**MAXNES**); **J**-index of the north-south vertical slice for printer output.

KARMAN Von Karman constant (= 0.4).

KCHI sigma level of the base of high clouds.

KCLO sigma level of the base of low clouds.

KCMD sigma level of the base of middle clouds.

KTIR sigma level most representative of the atmospheric temperature for downward IR calculations.

KXOUT (**MAXNES**); **K**-level of the horizontal slice for printer output.

K700 sigma level ~ 700 mb, where maximum **THETAE** is regarded as origin of air parcel that produces cloud.

MDATE time and date of the initial data set in yymmddhh format.

OMU constant used in Asselin time filter for all prognostic variables (= .1 - .2 $*$ **GNU**).

OMUHF constant used in Asselin time filter for all prognostic variables (= 1. - 2. $*$ **GNUHF**).

PTOP pressure (cb) at the top of the model.

PTOP4 = 4. $*$ **PTOP**.

QCON (**MKX**); used to compute the vertical interpolation coefficients for q_v (= (**SIGMA** (**K**) - **A** (**K**)) / (**A** (**K**-1) - **A** (**K**))).

R gas constant for dry air (= 287 J kg^{-1} K^{-1}).

ROVCP = **R** / **CP**.

ROVG = **R** / **G** (m K^{-1}).

SIGMA (**KXP1**); full sigma levels.

SOLCON solar constant (= 1395.6 J m^{-2} s^{-1}).

STBOLT Stefan Boltzmann's constant (= 5.67051E-8 J m^{-2} s^{-1} K^{-4}).

TIMAX maximum forecast time (min).

TRAN (46, 13); transmission coefficient, a function of precipitable water and path length (read in).

TWT (**MKX**, 2); coefficient used for vertical interpolation of temperature.

WGTD (**NSPGD**); sponge coefficient for dot-point variables.

WGTX (**NSPGX**); sponge coefficient for cross-point variables.

4.1.19 PARFDDA

/PARFDDA/ contains integers and constants pertaining to selection of FDDA options. For arrays dimensioned (**MAXSES**, 2), second index varies over type of analysis nudging: 1 = 3-D analysis nudging, 2 = surface analysis nudging within PBL.

DCON = 1 / **DPSMX** used for observation nudging.

DIFTIM (**MAXSES**, 2); time (min) between input analyses for analysis nudging.

DIFZ height above ground level (m) of lowest half-level.

DPSMX maximum p* change (cb) allowed within influence range of a surface observation used for observation nudging.

FDAEND (**MAXSES**); time (min) for termination of FDDA.

FDASTA (**MAXSES**); time (min) for initiation of FDDA.

GIP (**MAXSES**); observation-nudging coefficient (s^{-1}) for p*.

GIQ (**MAXSES**); observation-nudging coefficient (s^{-1}) for mixing ratio.

GIT (**MAXSES**); observation-nudging coefficient (s^{-1}) for temperature.

GIV (**MAXSES**); observation-nudging coefficient (s^{-1}) for wind.

I4D (**MAXSES**, 2); will FDDA analysis nudging be employed, (0=no;1=yes).

I4DI (**MAXSES**); will FDDA observation nudging be employed, (0=no;1=yes).

I4DITOT flag set to nonzero if observation nudging is active on any mesh.

IBLNOP (**MAXSES**, 4); flag for choice of surface-analysis nudging options:
= 0; apply surface-analysis nudging correction from the surface layer to every PBL layer.
= 1; compute surface-analysis nudging correction for each PBL layer by differencing surface-layer observed value and model-simulated value at each layer. Second index varies over variable.

IDQ indicator flag used when **IMOISP** = 1.

IDQN indicator flag used when **IMOISP** = 1.

IMOIS (**MAXSES**, 2); will the mixing ratio be nudged from analyses, (0=no;1=yes).

IMOISP (**MAXSES**, 2); will the mixing ratio be nudged, (0=no;1=yes), based on analyzed precipitation.

INDT (**MAXSES**, 2); flag for type of temporal weighting used for analysis nudging (0=no;1=user defined variation; 2=variation for surface-analysis nudging based on data density).

INDXY (**MAXSES**, 2); flag for type of horizontal weighting used for analysis nudging (0=no;1=user defined variation).

INDZ (**MAXSES**, 2); flag for type of vertical weighting used for analysis nudging (0=no;1=user defined variation).

INONBL (**MAXSES**, 4); will PBL fields be nudged from 3-D analyses when not using surface-analysis nudging within PBL. (0=yes; 1=exclude certain variables depending on integer value of second index).

INOPRO (**MAXSES**, 2); flag to withhold processing for specific analyses in surface-analysis nudging.

INT4 (**MAXSES**, 2); will temporal interpolation of analysis be employed for for analysis nudging, (0=no;1=yes).

IONF observation-nudging frequency in coarse grid time steps for observation-nudging calculations.

IPSTR (**MAXSES**, 2); flag for analysis nudging of p* in the hydrostatic version of MM5, (0=no;1=yes).

IROT (**MAXSES**); will vorticity be nudged from analyses, (0=no;1=yes).

ISMOIS (**MAXSES**); will the mixing ratio be nudged from observations, (0=no;1=yes).

ISPSTR (**MAXSES**); flag for observation nudging of p* in the hydrostatic version of MM5, (0=no;1=yes).

ISTEMP (**MAXSES**); will the temperature be nudged from observations, (0=no;1=yes).

ISWIND (**MAXSES**); will the wind field be nudged from observations, (0=no;1=yes).

ITEMP (**MAXSES**, 2); will the temperature be nudged from analyses, (0=no;1=yes).

IUN1 input unit number for 3-D analysis-nudging file.

IUN2 input unit number for 3-D analysis-nudging file.

IWIND (**MAXSES**, 2); will the wind field be nudged from analyses, (0=no;1=yes).

IWINDS (**MAXSES**, 2); will logarithmic-wind adjustment of analyzed surface wind speed be used before applying it throughout the PBL, (0=no;1=yes),

IWTSIG (**MAXSES**); flag for applying observation-nudging correction on pressure (0) or sigma (1) surfaces.

NPFG coarse-grid time-step frequency for select diagnostic print of analysis nudging.

NPFI coarse-grid time-step frequency for select diagnostic print of observation nudging.

NPFV vertical frequency for select diagnostic print of FDDA information.

NSTA number of observations available within time window for observation nudging.

PFREE user-defined pressure level (cb) where terrain effect becomes small.

RINBLW radius of influence (km) for surface analysis nudging where the horizontal weighting function depends on surface data density.

RINFMN multiplier for observation-nudging influence radius (**RINXY**) at the surface.

RINFMX multiplier for observation-nudging influence radius (**RINXY**) at the **PFREE** level.

RINP horizontal radius of influence (km) used when **IMOISP** = 1.

RINSIG vertical radius of influence (in sigma) for distance-weighted nudging corrections (for observation nudging).

RINXY default horizontal radius of influence (km) for distance-weighted nudging corrections (for observation nudging).

TIMANL (**MAXSES**); model time (min) of observed analysis for analysis nudging.

TIMOBP (**MAXSES**); model time (min) of a precipitation analysis.

TWINDO (time window)/2 (min) over which an observation will be used for nudging.

WNDFAC correction factor for surface wind speed adjustment to lowest model half-level in surface-analysis nudging.

XTIMT model time (min) of target analyses for analysis nudging.

XTIM1 model time (min) of initial analyses used in temporal interpolation for analysis nudging.

XTIM2 model time (min) of final analyses used in temporal interpolation for analysis nudging.

XTNEX model time (min) for next call to IN4DGD for input of nudging analysis files.

4.1.20 PBLDIM

/PBLDIM/ holds integers defining PBL top (level) and time step.

KNPBLH (**MJX**); **K**-level where PBL top is located in the north boundary slice.

KPBL2D (**MIX**, **MJX**); top (level) of the PBL.

KSPBLH (**MJX**); **K**-level where PBL top is located in the south boundary slice.

MITER0 = **DT***60/**DTMITE** + .99

4.1.21 PMOIST

/PMOIST/ stores constants used for both explicit and convective moisture calculations.

AVT constant used for terminal velocity of raindrops (= 841.99667 $m^{1-\mathbf{BVT}}$ s^{-1}).

AVTS constant used for terminal velocity of snow (= 11.72 $m^{1-\mathbf{BVTS}}$ s^{-1}).

BETA constant in Fletcher's formula (= 0.6).

BVT constant used for terminal velocity of raindrops (= 0.8).

BVTS constant used for terminal velocity of snow (= 0.41).

CONF condensation threshold (= 1.).

EP1 constant used in computing virtual temperature (= 0.608).

EP2 constant used in computing saturation mixing ratio (= 0.622).

G3PB constant for cloud microphysics, $(\Gamma(3.\mathbf{BVT})=\mathbf{G4PB}/(3.\mathbf{BVT}))$, gamma refers to the gamma-function.

G4PB constant for cloud microphysics, $(\Gamma(4.\mathbf{BVT})=17.837425)$.

G5PB constant for cloud microphysics, $(\Gamma(5.\mathbf{BVT})=1.8273)$.

HYDPRE (**MAXNES**); will water-loading effects be considered in hydrostatic equation (**IMOIST**=2), (0=no;1=yes).

IEVAP (**MAXNES**); will evaporation effects be considered (**IMOIST**=2):
< 0; The evaporation of rainwater only is not considered.
= 0; No evaporation is considered for rain or cloud.
>0; The evaporation is considered.

N0R constant used in Marshall-Palmer distribution of raindrops (=8.E6 m^{-4}).

PPI constant used for cloud (rain) microphysics, (= 1./**PI**∗**N0R** m^4).

PPIS constant used for cloud (snow) microphysics, (= 1./**PI**∗**N0S**∗0.1 m^4).

PRAC constant for accretion of cloud droplets
(= **PI**∗**N0R**∗**AVT**∗**G3PB**∗0.25 $m^{-3-\mathbf{BVT}}$).

PRACS constant for accretion of cloud snow
(= **PI**∗**N0S**∗**AVTS**∗**G3PBS**∗0.25 ∗ **ESI** $m^{-3-\mathbf{BVTS}}$).

PREC1 constant used for evaporation rate of raindrops (= 2∗**PI**∗**N0R**∗0.78 m^{-4}).

PREC2 constant used for evaporation rate of raindrops
= 2∗**PI**∗**N0R**∗0.32∗**AVT** $^{1/2}$ ∗**G5PB** $m^{-3.5-0.5\mathbf{BVT}}$).

PRECS1 constant used for evaporation rate of snow (= 4.∗**N0S**∗.65 m^{-4}).

PRECS2 constant used for evaporation rate of snow
(= 4.∗**N0S**∗**.**44∗**AVTS** $^{1/2}$ ∗**G5PBS** $m^{-3.5-0.5\mathbf{BVTS}}$).

QCK1 constant auto conversion rate (= 1.E-3 kg kg^{-1} s^{-1}).

QCTH threshold for the onset of auto conversion (kg kg^{-1}).

QDCRIT threshold of moisture convergence for the onset of precipitation
(cb kg kg^{-1} s^{-1}).

QWGHT (**MKX**); redistribution function in Kuo scheme for moisture flux due to convection.

RV gas constant for water vapor (= 461.5 J kg^{-1} K^{-1}).

SVPT0 constant used in computing saturation vapor pressure (= 273.15 K) correction.

SVP1 constant used in computing saturation vapor pressure (= 0.6112 cb).

SVP2 constant used in computing saturation vapor pressure (= 17.67).

SVP3 constant used in computing saturation vapor pressure (= 29.65 K).

TO freezing point (= 273.15 K).

TWGHT (**MKX**, 5**: MKX**, 1**: MKX**-3); heating profile for the Kuo cumulus parameterization.

VQFLX (**MKX**, 5**: MKX**, 1**: MKX**-3); weight function for eddy fluxes of convective moisture in Kuo scheme.

VTC constant used for terminal velocity of raindrops (= **AVT**∗**G4PB** / 6 $m^{1-\mathbf{BVT}}$ s^{-1}).

VTCS constant used for terminal velocity of snow (= **AVTS**∗**G4PBS** / 6 $m^{1-\mathbf{BVTS}}$ s^{-1}).

XLF0 latent heat of fusion (= .3337E6 J kg^{-1}).

XLS latent heat of sublimation (= **XLV** (T)+**XLF0** J kg^{-1}).

XLV latent heat of vaporization (= 2.5E6 J kg^{-1}).

XLV0 constant in **XLV** (T) (= 3.15E6 J kg^{-1}), where **XLV** (T)=**XLV0**-**XLV1**∗T.

XLV1 constant in **XLV** (T) (= 2370 J kg^{-1} K^{-1}).

XLVOCP = **XLV** / **CP** (K).

XMMAX mass of largest ice crystal (= 9.4E-10 kg).

XMO mass of initial ice crystal (= $(12.9E\text{-}6 / 16.3)^2$ kg).

XMOIST (**MAXSES**); will moisture effects be used in the thermodynamic equation, (0=no;1=yes).

XN0 constant in Fletcher's formula (= 0.01 m^{-3}).

4.1.22 PSANET

/PSANET/ stores arrays used for finer-domain boundary interpolations.

PSANE (2, **MIX**); two outermost columns (east).

PSANN (2, **MJX**); two outermost rows (north).

PSANS (2, **MJX**); two outermost rows (south).

PSANW (2, **MIX**); two outermost columns (west).

4.1.23 SIZE

/SIZE/ contains size informatin for the model input data and the surface analysis file (used in FDDA).

NUM3D An integer constant that is the number of 3-D fields in the model initial condition set (i.e., u, v, w, t, q_v).

NUM2D An integer constant that is the number of 2-D fields in the model initial condition set (i.e., ground temperature, coriolis, terrain elevation).

NUM1D An integer constant that is the number of 1-D fields in the model initial condition set (currently not used, NUM1D=0).

NUM0D An integer constant that is the number of 0-D fields in the model initial condition set (currenlty not used, NUM0D=0)

NUMLEVEL NUMLEVEL=NUM3D+NUM2D+NUM1D+NUM0D.

NUM3DS An integer constant that is the number of 3-D fields in the surface analysis nudging data set (currently not used, NUM3DS=0).

NUM2DS An integer constant that is the number of 2-D fields in the surface analysis nudging data set (i.e., u, v, w, t, q_v).

NUM1DS An integer constant that is the number of 1-D fields in the surface analysis nudging data set (currently not used, NUM1DS=0).

NUM0DS An integer constant that is the number of 0-D fields in the surface analysis nudging data set (currently not used, NUM0DS=0).

NUMLEVELS NUMLEVELS=**NUM3DS**+**NUM2DS**+**NUM1DS**+**NUM0DS**.

4.1.24 SOUNDL

/SOUNDL/ contains work arrays for SOUND routine.

E (**MIX, MJX, MKXP1**) work array.

F (**MIX, MJX, MKXP1**) work array.

4.1.25 WORKSP

/WORKSP/ holds a working array used in ZX4LP.

DWKSP (2500); working space in ZX4LP.

4.2 Common Blocks With Pointers

(The coarser-domain common blocks appear in alphabetical order paired with each associated finer-domain block).

4.2.1 ADDR1

/ADDR1/ holds three-dimensional prognostic variables at time t (**XXA**) and time t-1 (**XXB**). /ADDR1/ is for the coarser-domain variables.

IAQCA> **QCA** (**MIXM**, **MJXM**, **MKXM**); p^*q_c (cb kg kg^{-1}) at time t.

IAQCB> **QCB** (**MIXM**, **MJXM**, **MKXM**); p^*q_c (cb kg kg^{-1}) at time t-1.

IAQIA> **QIA** (**MIXIC**, **MJXIC**, **MKXIC**); p^*q_i (cb kg kg^{-1}) at time t.

IAQIB> **QIB** (**MIXIC**, **MJXIC**, **MKXIC**); p^*q_i (cb kg kg^{-1}) at time t-1.

IAQNIA> **QNIA** (**MIXIC**, **MJXIC**, **MKXIC**); p^*q_{ni} (cb kg kg^{-1}) at time t.

IAQNIB> **QNIB** (**MIXIC**, **MJXIC**, **MKXIC**); p^*q_{ni} (cb kg kg^{-1}) at time t-1.

IAQRA> **QRA** (**MIXM**, **MJXM**, **MKXM**); p^*q_r (cb kg kg^{-1}) at time t.

IAQRB> **QRB** (**MIXM**, **MJXM**, **MKXM**); p^*q_r (cb kg kg^{-1}) at time t-1.

IAQVA> **QVA** (**MIX**, **MJX**, **MKX**); p^*q_v (cb kg kg^{-1}) at time t.

IAQVB> **QVB** (**MIX**, **MJX**, **MKX**); p^*q_v (cb kg kg^{-1}) at time t-1.

IATA> **TA** (**MIX**, **MJX**, **MKX**); p*T (cb K) at time t.

IATB> **TB** (**MIX**, **MJX**, **MKX**); p*T (cb K) at time t-1.

IAUA> **UA** (**MIX**, **MJX**, **MKX**); p*U (cb m s^{-1}) at time t.

IAUB> **UB** (**MIX**, **MJX**, **MKX**); p*U (cb m s^{-1}) at time t-1.

IAVA> **VA** (**MIX**, **MJX**, **MKX**); p*V (cb m s^{-1}) at time t.

IAVB> **VB** (**MIX**, **MJX**, **MKX**); p*V (cb m s^{-1}) at time t-1.

4.2.2 ADDRN1

/ADDRN1/ is identical to /ADDR1/ except that it is for finer domains instead of the coarser domain, and both variable and address names are slightly changed. For the finer-domain variables, the letter "N" is usually appended to the coarser-domain name such that **UA** becomes **UAN** etc. For the finer-domain addresses, the letter "N" is inserted after the first letter in the coarser-domain name such that **IAUA** becomes **INAUA** etc.

4.2.3 ADDR2

/ADDR2/ holds two-dimensional arrays relating to diagnostic variables, physical processes, and terrestrial information. /ADDR2/ is for the coarser-domain variables.

IAAL> **ALB** (**MIX**, **MJX**); albedo (0< **ALB** <1).

IACG> **CAPG** (**MIX**, **MJX**); thermal capacity of ground slab (J m^{-2} K^{-1}).

IACR> **COSROT** (**MIX**, **MJX**); cos (theta) where theta is angle between y-axis and north.

IAEF> **EF** (**MIX**, **MJX**); other component of Coriolis parameter (s^{-1}).

IAEM> **EMISS** (**MIX**, **MJX**); emissivity.

IAF> **F** (**MIX**, **MJX**); Coriolis parameter (s^{-1}).

IAGL> **GLW** (**MIX**, **MJX**); long-wave radiation (J s^{-1} m^{-2}).

IAGS> **GSW** (**MIX**, **MJX**); shortwave radiation (J s^{-1} m^{-2}).

IAHX> **HFX** (**MIX**, **MJX**); sensible heat flux (J s^{-1} m^{-2}).

IAHO> **HOL** (**MIX**, **MJX**); exchange coefficient for heat and moisture. If HIRPBL is used, **HOL** = PBL height / Monin-Obukov length.

IAHT> **HT** (**MIX**, **MJX**); terrain height times gravity (m^2 s^{-2}).

IAMA> **MAVAIL** (**MIX**, **MJX**); moisture availability.

IAMO> **MOL** (**MIX**, **MJX**); exchange coefficient for momentum when using bulk PBL. If HIRPBL is used, **MOL**=kT* where T* is the eddy temperature (K).

IAMD> **MSFD** (**MIX**, **MJX**); inverse map-scale factor for dot points.

IAMX> **MSFX** (**MIX**, **MJX**); inverse map-scale factor for cross points.

IAPL> **PBL** (**MIX**, **MJX**); planetary boundary layer height (m).

IAPR> **PRW** (**MIX**, **MJX**); precipitable water (cm).

IAPA> **PSA** (**MIX**, **MJX**); p* (cb) at time t.

IAPSI> **PSAINI** (**MIX**, **MJX**); initial p* (cb).

IAPB> **PSB** (**MIX**, **MJX**); p* (cb) at time t-1.

IAQX> **QFX** (**MIX**, **MJX**); latent heat flux (J s^{-1} m^{-2}).

IARC> **RAINC** (**MIX**, **MJX**); accumulated convective rain (cm).

IARN> **RAINNC** (**MIX**, **MJX**); accumulated non-convective rain (cm).

IARP> **RAINP** (**MIX**, **MJX**); scratch array.

IARE> **REGIME** (**MIX**, **MJX**); bulk Richardson number (**BR**) when using bulk PBL, for HIRPBL, **REGIME** stores the PBL classes:
= 1; nighttime stable conditions when **BR** >0.2.
= 2; damped mechanical turbulent conditions when 0.0<**BR**<0.2.
= 3; forced convective conditions when **BR** <0.0 and **HOL** <1.5.
= 4; free convection conditions when **BR** <0.0 and **HOL** >1.5.

IASA> **SATBRT** (**MIX**, **MJX**); land use category.

IASH> **SHC** (**MIX**, **JMX**); dry soil heat capacity (J m^{-3} K^{-1}) when **IMOIAV**=1.

IASR> **SINROT** (**MIX**, **MJX**); sin(theta) where theta is angle between y-axis and north.

IASC> **SNOWC** (**MIX**, **MJX**); snow-cover data.

IATGA> **TGA** (**MIX**, **MJX**); ground temperature (K) at time t.

IATGB> **TGB** (**MIX**, **MJX**); ground temperature (K) at time t-1.

IATH> **THC** (**MIX**, **MJX**); thermal inertia (**IMOIAV**=0) or thermal conductivity (**IMOIAV**=1) based on **SHC** and **MAVAIL** (J m^{-1} s^{-1} K^{-1}).

IATM> **TMN** (**MIX**, **MJX**); constant temperature of infinite reservoir below slab (K)

IAUT> **UST** (**MIX**, **MJX**); friction velocity ($u*$) (m s^{-1}).

IALD> **XLAND** (**MIX**, **MJX**); grid point type of surface, (1=land, 2=water).

IALT> **XLAT** (**MIX**, **MJX**); latitude (degrees).

IALO> **XLONG** (**MIX**, **MJX**); longitude (degrees).

IAZN> **ZNT** (**MIX**, **MJX**); roughness length (m).

IAZO> **ZOL** (**MIX**, **MJX**); if HIRPBL is used, **ZOL** = height / Monin-Obukov length.

4.2.4 ADDRN2

/ADDRN2/ is identical to /ADDR2/ except that it is for finer domains instead of the coarser domain, and both variable and address names are slightly changed. For the finer-domain variables, the letter "N" is usually appended to the coarser-domain name such that **EF** becomes **EFN** etc. For the finer-domain addresses, the letter "N" is inserted after the first letter in the coarser-domain name such that **IAEF** becomes **INAEF** etc.

4.2.5 ADDR3

/ADDR3/ stores finer-domain boundary values and tendencies. /ADDR3/ is for the coarser-domain variables.

IAPE> **PEB** (**MIX**, **NSPGX**); east boundary value of p* (cb).

IAPET> **PEBT** (**MIX**, **NSPGX**); east boundary tendency of p* (cb s^{-1}).

IAPN> **PNB** (**MJX**, **NSPGX**); north boundary value of p* (cb).

IAPNT> **PNBT** (**MJX**, **NSPGX**); north boundary tendency of p* (cb s^{-1}).

IAPS> **PSS** (**MJX**, **NSPGX**); south boundary value of p* (cb).

IAPST> **PSBT** (**MJX**, **NSPGX**); south boundary tendency of p* (cb s^{-1}).

IAPW> **PWB** (**MIX**, **NSPGX**); west boundary value of p* (cb).

IAPWT> **PWBT** (**MIX**, **NSPGX**); west boundary tendency of p* (cb s^{-1}).

IAQCE> **QCEB** (**MIXM**, **MKXM**, **NSPGX**); east boundary value of p^*q_c (cb kg kg^{-1}).

IAQCET> **QCEBT** (**MIXM**, **MKXM**, **NSPGX**); east boundary tendency of p^*q_c (cb kg kg^{-1} s^{-1}).

IAQCN> **QCNB** (**MJXM**, **MKXM**, **NSPGX**); north boundary value of p^*q_c (cb kg kg^{-1}).

IAQCNT> **QCNBT** (**MJXM**, **MKXM**, **NSPGX**); north boundary tendency of p^*q_c (cb kg kg^{-1} s^{-1}).

IAQCS> **QCSB** (**MJXM**, **MKXM**, **NSPGX**); south boundary value of p^*q_c (cb kg kg^{-1}).

IAQCST> **QCSBT** (**MJXM**, **MKXM**, **NSPGX**); south boundary tendency of p^*q_c (cb kg kg^{-1} s^{-1}).

IAQCW> **QCWB** (**MIXM**, **MKXM**, **NSPGX**); west boundary value of p^*q_c (cb kg kg^{-1}).

IAQCWT> **QCWBT** (**MIXM**, **MKXM**, **NSPGX**); west boundary tendency of p^*q_c (cb kg kg^{-1} s^{-1}).

IAQE> **QEB** (**MIX**, **MKX**, **NSPGX**); east boundary value of p^*q_v (cb kg kg^{-1}).

IAQET> **QEBT** (**MIX**, **MKX**, **NSPGX**); east boundary tendency of p^*q_v (cb kg kg^{-1} s^{-1}).

IAQIE> **QIEB** (**MIXIC**, **MKXIC**, **NSPGX**); east boundary value of p^*q_i (cb kg kg^{-1}).

IAQIET> **QIEBT** (**MIXIC**, **MKXIC**, **NSPGX**); east boundary tendency of p^*q_i (cb kg kg^{-1} s^{-1}).

IAQIN> **QINB** (**MJXIC**, **MKXIC**, **NSPGX**); north boundary value of p^*q_i (cb kg kg^{-1}).

IAQINT> **QINBT** (**MJXIC**, **MKXIC**, **NSPGX**); north boundary tendency of p^*q_i (cb kg kg^{-1} s^{-1}).

IAQIS> **QISB** (**MJXIC**, **MKXIC**, **NSPGX**); south boundary value of p^*q_i (cb kg kg^{-1}).

IAQIST> **QINST** (**MJXIC**, **MKXIC**, **NSPGX**); south boundary tendency of p^*q_i (cb kg kg^{-1} s^{-1}).

IAQIW> **QIWB** (**MIXIC**, **MKXIC**, **NSPGX**); west boundary value of p^*q_i (cb kg kg^{-1}).

IAQIWT> **QIWBT** (**MIXIC**, **MKXIC**, **NSPGX**); west boundary tendency of p^*q_i (cb kg kg^{-1} s^{-1}).

IAQN> **QNB** (**MJX**, **MKX**, **NSPGX**); north boundary value of p^*q_v (cb kg kg^{-1}).

IAQNT> **QNBT** (**MJX**, **MKX**, **NSPGX**); north boundary tendency of p^*q_v (cb kg kg^{-1} s^{-1}).

IAQNIE> **QNIEB** (**MIXIC**, **MKXIC**, **NSPGX**); east boundary value of p^*q_{ni} (cb kg kg^{-1}).

IAQNIET> **QNIEBT** (**MIXIC**, **MKXIC**, **NSPGX**); east boundary tendency of p^*q_{ni} (cb kg kg^{-1} s^{-1}).

IAQNIN> **QNINB** (**MJXIC**, **MKXIC**, **NSPGX**); north boundary value of p^*q_{ni} (cb kg kg^{-1}).

IAQNINT> **QNINBT** (**MJXIC**, **MKXIC**, **NSPGX**); north boundary tendency of p^*q_{ni} (cb kg kg^{-1} s^{-1}).

IAQNIS> **QNISB** (**MJXIC**, **MKXIC**, **NSPGX**); south boundary value of p^*q_{ni} (cb kg kg^{-1}).

IAQNIST> **QNISBT** (**MJXIC**, **MKXIC**, **NSPGX**); south boundary tendency of p^*q_{ni} (cb kgkg^{-1} s^{-1}).

IAQNIW> **QNIWB** (**MIXIC**, **MKXIC**, **NSPGX**); west boundary value of p^*q_{ni} (cb kg kg^{-1}).

IAQNIWT> **QNIEWT** (**MIXIC**, **MKXIC**, **NSPGX**); west boundary tendency of p^*q_{ni} (cb kg kg^{-1} s^{-1}).

IAQRE> **QREB** (**MIXM**, **MKXM**, **NSPGX**); east boundary value of p^*q_r (cb kg kg^{-1}).

IAQRET> **QREBT** (**MIXM**, **MKXM**, **NSPGX**); east boundary tendency of p^*q_r (cb kg kg^{-1} s^{-1}).

IAQRN> **QRNB** (**MJXM**, **MKXM**, **NSPGX**); north boundary value of p^*q_r (cb kg kg^{-1}).

IAQRNT> **QRNBT** (**MJXM**, **MKXM**, **NSPGX**); north boundary tendency of p^*q_r (cb kg kg^{-1} s^{-1}).

IAQRS> **QRSB** (**MJXM**, **MKXM**, **NSPGX**); south boundary value of p^*q_r (cb kg kg^{-1}).

IAQRST> **QRSBT** (**MJXM**, **MKXM**, **NSPGX**); south boundary tendency of p^*q_r (cb kg kg^{-1} s^{-1}).

IAQRW> **QRWB** (**MIXM**, **MKXM**, **NSPGX**); west boundary value of p^*q_r (cb kg kg^{-1}).

IAQRWT> **QRWBT** (**MIXM**, **MKXM**, **NSPGX**); west boundary tendency of p^*q_r (cb kg kg^{-1} s^{-1}).

IAQS> **QSB** (**MJX**, **MKX**, **NSPGX**); south boundary value of p^*q_v (cb kg kg^{-1}).

IAQST> **QSBT** (**MJX**, **MKX**, **NSPGX**); south boundary tendency of p^*q_v (cb kg kg^{-1} s^{-1}).

IAQW> **QWB** (**MIX**, **MKX**, **NSPGX**); west boundary value of p^*q_v (cb kg kg^{-1}).

IAQWT> **QWBT** (**MIX**, **MKX**, **NSPGX**); west boundary tendency of p^*q_v (cb kg kg^{-1} s^{-1}).

IATE> **TEB** (**MIX**, **MKX**, **NSPGX**); east boundary value of p*T (cb K).

IATET> **TEBT** (**MIX**, **MKX**, **NSPGX**); east boundary tendency of p*T (cb K s^{-1}).

IATN> **TNB** (**MJX**, **MKX**, **NSPGX**); north boundary value of p*T (cb K).

IATNT> **TNBT** (**MJX**, **MKX**, **NSPGX**); north boundary tendency of p*T (cb K s^{-1}).

IATS> **TSB** (**MJX**, **MKX**, **NSPGX**); south boundary value of p*T (cb K).

IATST> **TSBT** (**MJX**, **MKX**, **NSPGX**); south boundary tendency of p*T (cb K s^{-1}).

IATW> **TWB** (**MIX**, **MKX**, **NSPGX**); west boundary value of p*T (cb K).

IATWT> **TWBT** (**MIX**, **MKX**, **NSPGX**); west boundary tendency of p*T (cb K s^{-1}).

IAUE> **UEB** (**MIX**, **MKX**, **NSPGD**); east boundary value of p*U (cb m s^{-1}).

IAUET> **UEBT** (**MIX**, **MKX**, **NSPGD**); east boundary tendency of p*U (cb m s^{-2}).

IAUI1> **UI1** (**MJX**, **MKX**); U value in the **I**=1 vertical slice (m s^{-1}).

IAUI2> **UI2** (**MJX**, **MKX**); U value in the **I**=2 vertical slice (m s^{-1}).

IAUIL> **UIL** (**MJX**, **MKX**); U value in the **I**=**IL** vertical slice (m s^{-1}).

IAUILX> **UILX** (**MJX**, **MKX**); U value in the **I**=**ILX** vertical slice (m s^{-1}).

IAUJ1> **UJ1** (**MIX**, **MKX**); U value in the **J**=1 vertical slice (m s^{-1}).

IAUJ2> **UJ2** (**MIX**, **MKX**); U value in the **J**=2 vertical slice (m s^{-1}).

IAUJL> **UJL** (**MIX**, **MKX**); U value in the **J**=**JL** vertical slice (m s^{-1}).

IAUJLX> **UJLX** (**MIX**, **MKX**); U value in the **J**=**JLX** vertical slice (m s^{-1}).

IAUN> **UNB** (**MJX**, **MKX**, **NSPGD**); north boundary value of p*U (cb m s^{-1}).

IAUNT> **UNBT** (**MJX**, **MKX**, **NSPGD**); north boundary tendency of p*U (cb m s^{-2}).

IAUS> **USB** (**MJX**, **MKX**, **NSPGD**); south boundary value of p*U (cb m s^{-1}).

IAUST> **USBT** (**MJX**, **MKX**, **NSPGD**); south boundary tendency of p*U (cb m s^{-2}).

IAUW> **UWB** (**MIX**, **MKX**, **NSPGD**); west boundary value of p*U (cb m s^{-1}).

IAUWT> **UWBT** (**MIX**, **MKX**, **NSPGD**); west boundary tendency of p*U (cb m s^{-2}).

IAVE> **VEB** (**MIX**, **MKX**, **NSPGD**); east boundary value of p*V (cb m s^{-1}).

IAVET> **VEBT** (**MIX**, **MKX**, **NSPGD**); east boundary tendency of p*V (cb m s^{-2}).

IAVI1> **VI**1 (**MJX**, **MKX**); V value in the **I**=1 vertical slice (m s^{-1}).

IAVI2> **VI**2 (**MJX**, **MKX**); V value in the **I**=2 vertical slice (m s^{-1}).

IAVIL> **VIL** (**MJX**, **MKX**); V value in the **I**=**IL** vertical slice (m s^{-1}).

IAVILX> **VILX** (**MJX**, **MKX**); V value in the **I**=**ILX** vertical slice (m s^{-1}).

IAVJ1> **VJ**1 (**MIX**, **MKX**); V value in the **J**=1 vertical slice (m s^{-1}).

IAVJ2> **VJ**2 (**MIX**, **MKX**); V value in the **J**=2 vertical slice (m s^{-1}).

IAVJL> **VJL** (**MIX**, **MKX**); V value in the **J**=**JL** vertical slice (m s^{-1}).

IAVJLX> **VJLX** (**MIX**, **MKX**); V value in the **J=JLX** vertical slice (m s^{-1}).

IAVN> **VNB** (**MJX**, **MKX**, **NSPGD**); north boundary value of p*V (cb m s^{-1}).

IAVNT> **VNBT** (**MJX**, **MKX**, **NSPGD**); north boundary tendency of p*V (cb m s^{-2}).

IAVS> **VSB** (**MJX**, **MKX**, **NSPGD**); south boundary value of p*V (cb m s^{-1}).

IAVST> **VSBT** (**MJX**, **MKX**, **NSPGD**); south boundary tendency of p*V (cb m s^{-2}).

IAVW> **VWB** (**MIX**, **MKX**, **NSPGD**); west boundary value of p*V (cb m s^{-1}).

IAVWT> **VWBT** (**MIX**, **MKX**, **NSPGD**); west boundary tendency of p*V (cb m s^{-2}).

4.2.6 ADDRN3

/ADDRN3/ is identical to /ADDR3/ except that it is for finer domains instead of the coarser domain, and both variable and address names are slightly changed. For the finer-domain variables, the letter "N" is usually appended to the coarser-domain name such that **PEB** becomes **PEBN** etc. For the finer-domain addresses, the letter "N" is inserted after the first letter in the coarser-domain name such that **IAPE** becomes **INAPE** etc.

4.2.7 ADDR4

/ADDR4/ contains constants or variables pertaining to the model time step, grid size, advection of mass, and total mass in the domain. /ADDR4/ is for the coarser-domain variables.

IAC0> **C200**; = **KARMAN** * **KARMAN** * **DX**/4.

IAC1> **C201**; = (100.-**PTOP**) / **DXSQ.**

IAC3> **C203**; = 1./**DXSQ**.

IADT> **DT**; time step (s).

IADT2> **DT2**; = 2 * **DT.**

IADTM> **DTMIN**; = **DT**/60. (min).

IADX> **DX**; grid size (m).

IADX2> **DX2**; = 2 * **DX**.

IADX4> **DX4**; = 4 * **DX**.

IADX8> **DX8**; = 8 * **DX**.

IAD16> **DX16**; = 16 * **DX**.

IADXQ> **DXSQ**; = **DX** * **DX**.

IAFNG> **FNUDGE**; relaxation coefficient for relaxation boundary conditions.

IAGNG> **GNUDGE**; relaxation coefficient for relaxation boundary conditions.

IAIL> **IL**; number of dot points in the y-direction.

IAILX> **ILX**; = **IL** - 1.

IAILM> **ILXM**; = **ILX** - 1.

IAJL> **JL**; number of dot points in the x-direction.

IAJLX> **JLX**; = **JL** - 1.

IAJLM> **JLXM**; = **JLX** - 1.

IAKL> **KL**; number of vertical layers (half-levels).

IAKM> **KLM**; = **KL** - 1.

IAKP1> **KLP1**; = **KL** + 1.

IAKTU> **KTAU**; time-step counter.

IAKTR> **KTAUR**; time-step counter used when restarting.

IADMA> **TDADV**; total air mass (kg) advected through lateral boundaries.

IADMI> **TDINI**; total air mass (kg) initially in the domain.

IAQMA> **TQADV**; total water substance (kg) advected through lateral boundaries.

IAQME> **TQEAV**; total water vapor (kg) evaporated from ground.

IAQMI> **TQINI**; total water substance (kg) initially in the domain.

IAQMR> **TQRAI**; total accumulated rainfall (kg) on the ground.

IAXK> **XKHMAX**; maximum horizontal diffusion coefficient ($m^2\ s^{-1}$).

IAXK> **XKHZ**; constant part of the horizontal diffusion coefficient ($m^2\ s^{-1}$).

IAXT> **XTIME**; forecast time (min).

4.2.8 ADDRN4

/ADDRN4/ is identical to /ADDR4/ except that it is for finer domains instead of the coarser domain, and both variable and address names are slightly changed. For the finer-domain variables, the letter "N" is usually appended to the coarser-domain name such that **DT** becomes **DTN** etc. For the finer-domain addresses, the letter "N" is inserted after the first letter in the coarser-domain name such that **IADT** becomes **INADT** etc.

4.2.9 ADDR5

/ADDR5/ holds three-dimensional storage arrays of the observed analysis variables and tendencies that are needed for analysis-nudging FDDA. /ADDR5/ is for the coarser-domain variables.

IFQBO> **QBO** (**MIXF**, **MJXF**, **MKXF**); storage array of observed analysis of p^*q_v (cb kg kg^{-1}).

IFQBT> **QBOTEN** (**MIXF**, **MJXF**, **MKXF**); storage array of observed tendency of p^*q_v (cb kg kg^{-1} min^{-1}).

IFTBO> **TBO** (**MIXF**, **MJXF**, **MKXF**); storage array of observed analysis of p^*T (cb K).

IFTBT> **TBOTEN** (**MIXF**, **MJXF**, **MKXF**); storage array of observed tendency of p^*T (cb K min^{-1}).

IFUBO> **UBO** (**MIXF**, **MJXF**, **MKXF**); storage array of observed analysis for p^*U (cb m s^{-1}).

IFUBT> **UBOTEN** (**MIXF**, **MJXF**, **MKXF**); storage array of observed tendency of p^*U (cb m s^{-1} min^{-1}).

IFVBO> **VBO** (**MIXF**, **MJXF**, **MKXF**); storage array of observed analysis of p^*V (cb m s^{-1}).

IFVBT> **VBOTEN** (**MIXF**, **MJXF**, **MKXF**); storage array of observed tendency of p^*V (cb m s^{-1} min^{-1}).

IFVOR> **VORDIF** (**MIXF**, **MJXF**, **MKXF**); difference of observed and model vorticity (s^{-1}).

4.2.10 ADDRN5

/ADDRN5/ is equivalent to /ADDR5/ except that it is for finer domains instead of the coarser domain and it only contains addresses. The address names are different in that the second letter in the coarser- domain names (always "F") is changed to "G" for the finer-domain address names.

4.2.11 ADDR6

/ADDR6/ contains variable arrays and weighting arrays used in analysis-nudging FDDA. /ADDR6/ is for the coarser-domain variables.

IFDMI> **DMI** (**MIXF**, **MJXF**); horizontal array used for analysis nudging of mixing ratio based on observed precipitation.

IFDMS> **DMSCR** (**MIXF**, **MJXF**); horizontal array used for analysis nudging of mixing ratio based on observed precipitation.

IFIPR> **IPRE** (**MIXF**, **MJXF**); horizontal array used for analysis nudging of mixing ratio based on observed precipitation.

IFIPS> **IPSCR** (**MIXF**, **MJXF**); horizontal array used for analysis nudging of mixing ratio based on observed precipitation.

IFMPD> **MPSD** (**MIXF**, **MJXF**); map-scale factor on dot points used for analysis nudging of vorticity.

IFMPX> **MPSX** (**MIXF**, **MJXF**); map-scale factor on cross points used for analysis nudging of vorticity.

IFPSB> **PSBD** (**MIXF**, **MJXF**); model p* (cb) on dot points at time t-1.

IFPSO> **PSO** (**MIXF**, **MJXF**); observed p* (cb) on cross points.

IFPSC> **PSOC** (**MIXF**, **MJXF**); observed p* (cb) on cross points interpolated in time.

IFPSD> **PSOD** (**MIXF**, **MJXF**); observed p* (cb) on dot points interpolated in time.

IFPST> **PSOTEN** (**MIXF**, **MJXF**); observed p* tendency (cb min^{-1}) on cross points.

IFWCS> **WCS** (**MIXF**, **MJXF**); horizontal weighting array on cross points used for analysis nudging.

IFWDT> **WDT** (**MIXF**, **MJXF**); horizontal weighting array on dot points used for analysis nudging.

IFWQ> **WQ** (**MIXF**, **MJXF**); horizontal array used for analysis nudging of mixing ratio based on observed precipitation.

4.2.12 ADDRN6

/ADDRN6/ is equivalent to /ADDR6/ except that it is for finer domains instead of the coarser domain, and it only contains addresses. The address names are different in that the second letter in the coarser-domain names (always "F") is changed to "G" for the finer-domain address names.

4.2.13 ADDR7

/ADDR7/ stores weighting and p* arrays and time/date identification variables needed for surface-analysis nudging FDDA within the PBL. /ADDR7/ is for the coarser-domain variables.

IFBLPC> **BLPOC** (**MIXF**, **MJXF**); observed p* (cb) on cross points and used for surface analysis nudging.

IFBLPD> **BLPOD** (**MIXF**, **MJXF**); observed p* (cb) on dot points and used for surface analysis nudging.

IFBLWS> **BLWCS** (**MIXF**, **MJXF**); horizontal weighting array on cross points and used for nudging in the PBL.

IFBLWT> **BLWDT** (**MIXF**, **MJXF**); horizontal weighting array on dot points and used for nudging in the PBL.

IFBLWN> **BLWNV** (**NVAR**, **MIXF**, **MJXF**); storage array for horizontal weighting function based on surface data density and used for surface-analysis nudging.

IFIDC> **IDCHK** (**NCHA**, **NVAR**); array identifying what variables at **IDDATE** dates to exclude from surface-analysis nudging.

IFIDD> **IDDATE** (**NCHA**); array identifying what dates to exclude from surface analysis nudging.

IFIDH> **IDHK** (**NVAR**); data-quality flag array used by the **INOPRO** option.

IFIQC> **IQCHK** (**NTIM**, **NVAR**); flag indicating quality of data (based on density) for surface-analysis nudging in PBL (if = 0, data are not used for FDDA).

IFNTB> **NTB** (**NVAR**); position counter from 1 to **NTIM** for **SFCTIM** and **SFCOBS** arrays.

IFNTE> **NTE** (**NVAR**); position counter from 1 to **NTIM** for **SFCTIM** and **SFCOBS** arrays.

IFSFCO> **SFCOBS** (**NTIM**, **NVAR**, **MIXF**, **MJXF**); storage array for surface analyses of each variable at each time used for surface-analysis nudging.

IFSFT> **SFCTIM** (**NTIM**); corresponding model time (min) for each observed surface analysis.

IFTIB> **TIMB** (**NVAR**); beginning bracketing time (min) for temporal interpolation of observed surface analysis.

IFTIE> **TIME** (**NVAR**); ending bracketing time (min) for temporal interpolation of observed surface analysis.

IFWXY> **WXYTOP** (**MIXF**, **MJXF**); horizontal weighting array based on topography and optionally used to compute weighting function for surface analysis nudging.

4.2.14 ADDRN7

/ADDRN7/ is equivalent to /ADDR7/ except that it is for finer domains instead of the coarser domain and it only contains addresses. The address names are different in that the second letter in the coarser- domain names (always "F") is changed to "G" for the finer-domain address names.

4.2.15 ADDR8

/ADDR8/ contains variables providing information concerning location, time, and deviation from forecast of observations used for observation-nudging FDDA. /ADDR8/ is for the coarser-domain variables.

IFERR> **ERRF** (6, **NIOBF**); difference between observed and model values at the observation location (first index varies over variable).

IFRIO> **RIO** (**NIOBF**); coarse-grid location of each nudging observation in the **I** (north-south) direction.

IFRJO> **RJO** (**NIOBF**); coarse-grid location of each nudging observation in the **J** (east-west) direction.

IFRKO> **RKO** (**NIOBF**); half-sigma location of each nudging observation in the vertical direction.

IFSVWT> **SAVWT** (4, **MIXF**, **MJXF**, **MKXF**); storage array used with **IONF** option holding the (weights $*$ corrections) at each grid point for each variable. (First index varies over variable.)

IFTIM> **TIMEOB** (**NIOBF**); model time (hr) of each nudging observation within time window.

IFVAR> **VAROBS** (5, **NIOBF**); storage array for nudging observations within time window (first index varies over variable).

4.2.16 ADDRN8

/ADDRN8/ is equivalent to /ADDR8/ except that it is for finer domains instead of the coarser domain and it only contains addresses. The address names are different in that the second letter in the coarser- domain names (always "F") is changed to "G" for the finer-domain address names.

4.2.17 ADDRNC

/ADDRNC/ is equivalent to /ADDRNN/ except that it is for coarser domains instead of the finer domain and it only contains addresses. The address names are different in that the third letter in the finer-domain names (always "N") is changed to "C" for the coarser-domain address names.

4.2.18 ADDRNN

/ADDRNN/ stores grid-point information relating to the location of the finer-domain boundaries or overlapping areas.

INN23> **IEN** (**MAXSES**); ending grid point (**I**-direction) for overwriting nested boundaries.

INN20> **INM1**; = **INORTH** - 1.

INN21> **INM2**; = **INORTH** - 2.

INN4> **INORTH**; **I**-coordinate (large domain) for north boundary of nested domain (= **NESTI** + (**INX** - 1) / **IRATIO**).

INN19> **INP1**; = **INORTH** + 1.

INN18> **INP2**; = **INORTH** + 2.

INN16> **ISM1**; = **ISOUTH** - 1.

INN17> **ISM2**; = **ISOUTH** - 2.

INN5> **ISOUTH**; **I**-coordinate (large domain) for south boundary of nested domain (= **NESTI**).

INN15> **ISP1**; = **ISOUTH** + 1.

INN14> **ISP2**; = **ISOUTH** + 2.

INN22> **ISTAR** (**MAXSES**); beginning grid point (**I**-direction) for overwriting nested boundaries.

INN26> **ISTO** (**MAXSES**); grid-point location specifying where to overwrite nested boundaries of overlapping nest.

INN2> **JEAST**; **J**-coordinate (large domain) for east boundary of nested domain (= **NESTJ** + (**JNX** - 1) / **IRATIO**).

INN25> **JEN** (**MAXSES**); ending grid point (**J**-direction) for overwriting nested boundaries.

INN8> **JEM1**; = **JEAST** - 1.

INN9> **JEM2**; = **JEAST** - 2.

INN7> **JEP1**; = **JEAST** + 1.

INN6> **JEP2**; = **JEAST** + 2.

INN24> **JSTAR** (**MAXSES**); beginning grid point (**J**-direction) for overwriting nested boundaries.

INN27> **JSTO** (**MAXSES**); grid-point location specifying where to overwrite nested boundaries of overlapping nest.

INN3> **JWEST**; **J**-coordinate (large domain) for west boundary of nested domain (= **NESTJ**).

INN12> **JWM1**; = **JWEST** - 1.

INN13> **JWM2**; = **JWEST** - 2.

INN11> **JWP1**; = **JWEST** + 1.

INN10> **JWP2**; = **JWEST** + 2.

INN1> **NUMNES**; integer defining domain for which addresses are computed.

4.2.19 ADDRSP

/ADDRSP/ holds arrays pertaining to calculation of divergence and geopotential in vertical mode space for the split-explicit scheme.

/ADDRSP/ is for the coarser-domain variables.

ISPAM> **AM** (**MKX**, **NSPLIT**); = **ZMATX**∗**A** (K), where **A** is a 2-D matrix operator defined in the split-explicit routines.

ISPAN> **AN** (**NSPLIT**); = **DSIGMA**∗**ZMATX**

ISPBZ> **BZ** (**MKX**, **KXP1**); used in the transformation of geopotential from sigma space to vertical mode space (J kg^{-1} K^{-1}).

ISPCZ> **CZ** (**MKX**, **MKX**); used in the transformation of geopotential from sigma space to vertical mode space (J kg^{-1}).

ISPDS> **DSTOR** (**MIX**, **MJX**, **NSPLIT**); stores divergence **DELD** in the split-explicit scheme (cb s^{-1}).

ISPDTA> **DTAU** (**NSPLIT**); short-time step (s) for the split-explicit scheme.

ISPHB> **HBAR** (**MKX**); equivalent depths (m^2 s^{-2}) of the vertical modes.

ISPHS> **HSTOR** (**MIX**, **MJX**, **NSPLIT**); stores geopotential **DELH** in the split-explicit scheme (m^2 s^{-2}).

ISPM> **M** (**NSPLIT**); ratio (longtime step / short-time step) for the split-explicit scheme.

ISPZMX> **ZMATX** (**MKX**, **MKX**); a matrix used in the transformation of divergence from sigma space to vertical mode space.

ISPZMR> **ZMATXR** (**MKX**, **MKX**); matrix inverse of **ZMATX.**

4.2.20 ADDNSP

/ADDNSP/ is equivalent to /ADDRSP/ except that it is for finer domains instead of the coarser domain and it only contains addresses.The address names are different in that the second letter in the coarser domain names (always "S") is changed to "O" for the finer-domain address names.

4.2.21 ADDRV

/ADDRV/ is for the Navy PBL scheme which is not used in Version 1.

4.2.22 ADDRVN

/ADDRVN/ is for the Navy PBL scheme which is not used in Version 1.

4.2.23 NHCNS

/NHCNS/ contains reference-state arrays for the nonhydrostatic option. /NHCNS/ is for the coarser-domain variables.

INHPS0> **PS0** (**MIXNH**, **MJXNH**); reference surface pressure (Pa).

INHT0> **T0** (**MIXNH**, **MJXNH**, **MKXNH**); reference temperature (K).

4.2.24 NNCNS

/NNCNS/ is identical to /NHCNS/ except that it is for finer domains instead of the coarser domain, and both variable and address names are slightly changed. For the finer-domain variables, the letter "N" is appended to the coarser-domain name such that **PS0** becomes **PS0N** etc. The address names are different in that the third letter in the coarser- domain names (always "H") is changed to "N" for the finer-domain address names.

4.2.25 NONHYD

/NONHYD/ holds three-dimensional prognostic variables and tendencies for the nonhydrostatic option. /NONHYD/ is for coarser-domain variables.

INHPPA> **PPA** (**MIXNH**, **MJXNH**, **MKXNH**); p*p' (cb Pa) at time t.

INHPPB> **PPB** (**MIXNH**, **MJXNH**, **MKXNH**); p*p' (cb Pa) at time t-1.

INHWA> **WA** (**MIXNH**, **MJXNH**, **KXP1NH**); p*W (cb m s^{-1}) at time t.

INHWB> **WB** (**MIXNH**, **MJXNH**, **KXP1NH**); p*W (cb m s^{-1}) at time t-1.

4.2.26 NNNHYD

/NNNHYD/ is identical to /NONHYD/ except that it is for finer domains instead of the coarser domain, and both variable and address names are slightly changed. For the finer-domain variables, the letter "N" is usually appended to the coarser-domain name such that **PPA** becomes **PPAN** etc. For the finer-domain addresses, the letter "N" replaces the third letter "H" in the coarser-domain name such that **INHPPA** becomes **INNPPA** etc.

4.2.27 NONHYDB

/NONHYDB/ stores nested-domain boundary values and tendencies for the nonhydrostatic option. /NONHYDB/ is for coarser-domain variables.

INHPPE> **PPEB** (**MIXNH**, **MKXNH**, **NSPGX**); p*p' (cb Pa) at east boundary.

INHPPET> **PPEBT** (**MIXNH**, **MKXNH**, **NSPGX**); p*p' (cb Pa s^{-1}) tendency at east boundary.

INHPPN> **PPNB** (**MJXNH**, **MKXNH**, **NSPGX**); p*p' (cb Pa) at north boundary.

INHPPNT> **PPNBT** (**MJXNH**, **MKXNH**, **NSPGX**); p*p' (cb Pa s^{-1}) tendency at north boundary.

INHPPS> **PPSB** (**MJXNH**, **MKXNH**, **NSPGX**); p*p' (cb Pa) at south boundary.

INHPPST> **PPSBT** (**MJXNH**, **MKXNH**, **NSPGX**); p*p' (cb Pa s^{-1}) tendency at south boundary.

INHPPW> **PPWB** (**MIXNH**, **MKXNH**, **NSPGX**); p*p' (cb Pa) at west boundary.

INHPPWT> **PPWBT** (**MIXNH**, **MKXNH**, **NSPGX**); p*p' (cb Pa s^{-1}) tendency at west boundary.

INHWE> **WEB** (**MIXNH**, **KXP1NH**, **NSPGX**); p*W (cb m s^{-1}) at east boundary.

INHWET> **WEBT** (**MIXNH**, **KXP1NH**, **NSPGX**); p*W (cb m s^{-2}) tendency at east boundary.

INHWN> **WNB** (**MJXNH**, **KXP1NH**, **NSPGX**); p*W (cb m s^{-1}) at north boundary.

INHWNT> **WNBT** (**MJXNH**, **KXP1NH**, **NSPGX**); p*W (cb m s^{-2}) tendency at north boundary.

INHWS> **WSB** (**MJXNH**, **KXP1NH**, **NSPGX**); p*W (cb m s^{-1}) at south boundary.

INHWST> **WSBT** (**MJXNH**, **KXP1NH**, **NSPGX**); p*W (cb m s^{-2}) tendency at south boundary.

INHWW> **WWB** (**MIXNH**, **KXP1NH**, **NSPGX**); p*W (cb m s^{-1}) at west boundary.

INHWWT> **WWBT** (**MIXNH**, **KXP1NH**, **NSPGX**); p*W (cb m s^{-2}) tendency at west boundary.

4.2.28 NNNHYDB

/NNNHYDB/ is identical to /NONHYDB/ except that it is for finer domains instead of the coarser domain, and both variable and address names are slightly changed. For the finer-domain variables, the letter "N" is appended to the coarser-domain name such that **PPEB** becomes **PPEBN** etc. For the finer-domain addresses, the letter "N" replaces the third letter "H" in the coarser-domain name such that **INHPPE** becomes **INNPPE** etc.

4.2.29 RADIAT

/RADIAT/ holds a three-dimensional tendency array for radiation. /RADIAT/ is for the coarser-domain variables.

IRTT> **RTTEN** (**MIX**, **MJX**, **MKX**) total radiative temperature tendency (K s^{-1}).

4.2.30 RADIATN

/RADIATN/ is equivalent to /RADIAT/ except that it is for finer domains instead of the coarser domain and it only contains an address. The address name is different in that the letter "N" is inserted after the first letter in the coarser-domain name such that **IRTT** becomes **INRIT**.

4.2.31 UPRAD

/UPRAD/ contains a two-dimensional array for the upper radiative boundary condition. /UPRAD/ is for the coarser-domain variables.

IUPR> **TMASK**(-6:6, -6:6) array for upper radiative boundary condition (m s^{-1} Pa^{-1}).

4.2.32 UPRADN

/UPRADN/ is equivalent to /UPRAD/ except that it is for finer domains instead of the coarser domain and it only contains an address. The address name is different in that the letter "N" is inserted after the first letter in the coarser-domain name such that **IUPR** becomes **INUPR**.

Section 5

SUBROUTINES

In this section, subroutines are presented in alphabetical order and are described following a standard format. A short general description of the subroutine is given first followed by a list of the routines it is called from and any routines it calls. Arguments are then listed in the order they appear in the code with input variables shown in **UPPER CASE** and output variables in **lower case**. Arguments that serve as both input and output are shown in lower case appended with an asterisk, and those that are only used internally (neither input or output) are [**BRACKETED**]. Definitions of the arguments are given in section 6. Common blocks used by the subroutine are listed next, followed by definition of any data in DATA statements. Finally, a detailed description is provided for subroutines that perform many significant tasks, and usage notes are included for a few routines.

5.1 ADDALL

Subroutine ADDALL presets pointer addresses of all variables on all domains in common blocks associated with pointers and puts them into the **IAXALL (NUMVAR, MAXNES)** array.

ADDALL is called from the main program.

Arguments: None.

Common blocks: ADDR0, HUGE, and NESLEV.

Data: None.

5.2 ADDRX1C

Subroutine ADDRX1C sets pointer addresses specific to a given domain and variable by equivalencing the pointers to the common block addresses.

ADDRX1C is called from the main program and from subroutines CHKNST, INIT, INITNEST, INITSAV, NSTLEV1, NSTLEV2, NSTLEV3, OUTPUT, and SHUTDO.

Arguments: **IARR**.

Common blocks: ADDR1, ADDR2, ADDR3, ADDR4, ADDR5, ADDR6, ADDR7, ADDR8, ADDRNC, ADDRSP, ADDRV, HUGE, NESLEV, NHCNS, NONHYD, NONHYDB, RADIAT, and UPRAD.

Data: None.

5.3 ADDRX1N

Subroutine ADDRX1N sets pointer addresses as in ADDRX1C, except in this case, it is for a second domain needed at the same time as the first domain handled by ADDRX1C.

ADDRX1N is called from the main program and from subroutines CHKNST, INITNEST, INITSAV, NSTLEV1, NSTLEV2, and NSTLEV3.

Arguments: **IARR**.

Common blocks: ADDRN1, ADDRN2, ADDRN3, ADDRN4, ADDR5N, ADDR6N, ADDR7N, ADDR8N, ADDNSP, ADDRNN, ADDRVN, HUGE, NESLEV, NNCNS, NNNHYD, NNNHYDB, RADIATN, and UPRAD.

Data: None.

5.4 ARAMB

Subroutine ARAMB sets up matrices for a linear solver routine (ZX4LP) that returns mass fluxes as solutions for the Arakawa-Schubert convection scheme.

ARAMB is called from subroutine ARASCH. It calls subroutine ZX4LP.

Arguments: **[ITEST], DRW, IW1, DDSOL, DPSOL, [DB], [DC], [DA], [IW], IWW, MBOTH, NIMSL, [ITEST2], IA, MINP, XK, F, xmb, KB, [IER], DTIME, FTEST, KNUM.**

Common blocks: None.

Data: None.

5.5 ARAOUT

Subroutine ARAOUT calculates feedback effects of clouds for the Arakawa-Schubert convection scheme.

ARAOUT is called from subroutine ARASCH.

Arguments: **[XMC], XMB, ZU, DELT, DELQ, KDIM, outtem, outq, ier, PW, pre, PWD, ZD, EDT, KNUM.**

Common blocks: None.

Data: None.

5.6 ARAOUTS

Subroutine ARAOUTS calculates feedback effects of clouds for the shallow convection scheme.

ARAOUTS is called from subroutine SHALLOW.

Arguments: **[XMC], XMB, ZU, DELT, DELQ, KDIM, outtem, outq, ier, pre, KNUM.**

Common blocks: None.

Data: None.

5.7 ARASCH

Subroutine ARASCH is the Arakawa-Schubert cumulus parameterization routine. Although it is the main routine for the Arakawa-Schubert scheme, it calls several other related subroutines.

ARASCH is called from subroutine CUPARA3. It calls subroutines ARAMB, ARAOUT, CLODWD, CLOUDW, ENTR, HEIPRE, KERHEL, MAXIM, MINIM, MOIENE, PRECIP, SOUNDD, and ZUNC.

Arguments: **TI, QI, Z1, TIO, QIO, PIO, KLEV, pre, PI, outtem, outq, DTIME, KBMAX, PCUT, C0, FTEST, PSUR, PSURO, ierrt, KDET, VSP.**

Common blocks: HUGE, NESLEV, PARAM2, PARAM3, PARFDDA, and PMOIST.

Data: **RAD (KNUM)** /0., 20000., 2000., 400., 267., 200./ cloud updraft radius (m).

---- Program Detail ----

ARASCH computes the moisture cycle using an Arakawa-Schubert-type cumulus parameterization scheme that includes downdrafts. The sequence of the basic subroutine tasks is:

1. Initialize some of the arrays.
2. Compute the heights of the sigma levels (HEIPRE).
3. Calculate the moist static energy **HE** and the saturation moist static energy **HES** (MOIENE).
4. Determine the level with the highest moist static energy (MAXIM).
5. Determine the cloud base and level of largest negative buoyancy (MINIM).
6. Begin in-cloud calculations, use **LP** do-loop where **LP** is cloud type.

a) Calculate moist static energy and cloud top (ENTR).

b) Compute normalized mass-flux profile (ZUNC).

c) Calculate the amount of moisture in the updraft (PRECIP).

d) Calculate the precipitation efficiency in terms of windshear.

e) Determine downdraft thermodynamic properties and the normalized mass flux (SOUNDD).

f) Calculate efficiency of generation of kinetic energy for the updraft (CLOUDW) and for the downdraft (CLODWD).

7. Begin new in-cloud and modified environmental calculations, loop again over cloud type.

   a) Calculate cloud effect on environment (KERHEL).

   b) Call HEIPRE and MOIENE for heights and moist static energy.

   c) Call ENTR, ZUNC, PRECIP, and SOUNDD for the moist static energy (in cloud), normalized mass-flux profile, amount of moisture in the updraft, and the downdraft thermodynamic properties.

   d) Call CLOUDW and CLODWD for updraft and downdraft cloud-work functions.

8. Calculate the mass fluxes (ARAMB).

5.8 BDYIN

Subroutine BDYIN reads in the coarsest-grid boundary conditions (boundary data input file).

BDYIN is called from the main program and from subroutine INIT.

Arguments: **IUNIT, tbdybe, bdytim, IX, JX.**

Common blocks: ADDR0, ADDR3, ADDR4, HUGE, NESLEV, and NONHYDB.

Data: None.

5.9 BDYOVL1 [multi-tasked]

Subroutine BDYOVL1 adjusts overlapping grid-point values on and near the nested boundaries for overlapping nests.

BDYOVL1 is called from subroutines NSTLEV1, NSTLEV2, and NSTLEV3.

Arguments: **NESCOU.**

Common blocks: ADDR0, ADDR1, ADDR2, ADDR3, ADDR4, ADDRN1, ADDRN2, ADDRN3, ADDRN4, ADDRNN, HUGE, MIC, NESLEV, PARAM2, PARAM3, PARFDDA, and PBLDIM.

Data: None.

5.10 BDYUV [multi-tasked]

Subroutine BDYUV sets boundary values of U, V according to the type of boundary conditions (fixed or time dependent) specified by the input variable **IBOUDY** defined in PARAM.

BDYUV is called from subroutine BDYVAL. It calls subroutine DOTS.

Arguments: **IB, DTB.**

Common blocks: ADDR1, ADDR2, ADDR3, ADDR4, HUGE, MIC, NESLEV, PARAM3, and PBLDIM.

Data: None.

5.11 BDYVAL [multi-tasked]

Subroutine BDYVAL sets boundary values of the water substance variables equal to 0 if the wind flow is into the domain (inflow), or equal to the value of the first interior point if outflow.

BDYVAL is called from the main program. It calls subroutine BDYUV.

Arguments: **IN, NN, IEXEC.**

Common blocks: ADDR0, ADDR1, ADDR2, ADDR3, ADDR4, HUGE, MIC, NESLEV, PARAM2, PARAM3, PARFDDA, and PBLDIM.

Data: None.

5.12 BLBRGD

Subroutine BLBRGD is an analysis-nudging FDDA routine that calculates the new values of **TIMB** and **TIME**, and position counters **NTB** and **NTE**, for bracketing **XTIME** with two surface analyses.

BLBRGD is called from subroutine BLNUDGD.

Arguments: **NV, MVAR, MTIM, XTIME, SFCTIM, timb, time, ntb, nte*, IQCHK, IN, KTAU, KTAUR, IFREST**.

Common blocks: None.

Data: None.

5.13 BLKPBL

Subroutine BLKPBL calculates the PBL tendencies using a bulk aerodynamic parameterization.

BLKPBL is called from subroutines SOLVE1 and SOLVE3. It calls subroutines SFCRAD and SLAB.

Arguments: **IYY, JXX, KZZ, J, INEST, NST, uten*, vten*, qcten*, scr1*, scr2*.**

Common blocks: ADDR1, ADDR2, ADDR4, ASSEL, HUGE, MIC, NESLEV, NHCNS, NHCNST, NONHYD, PARAM2, PARAM3, PARFDDA, PBLDIM, and PMOIST.

Data: **KZO** /1./ background diffusion coefficient ($m^2\ s^{-1}$). **RIC** /3.05/ critical bulk Richardson number for Deardorff scheme. **SZKM** /1600./ constant used to compute the eddy diffusion coefficients (m^2).

---- Program Detail ----

BLKPBL uses wind speed, PBL height, and the bulk Richardson number to compute the sensible heat, latent heat, and momentum fluxes from the ground. The sequence of the basic subroutine tasks is:

1. Compute wind speed and height for the lowest sigma level.
2. Compute the PBL height and bulk Richardson number.
3. Compute the exchange coefficient for momentum, heat, and moisture.
4. Compute sensible and latent heat flux (**ISFFLX** = 1).
5. Compute ground temperature from surface energy budget (**ITGFLG** = 1).
6. Add terrain effect to the exchange coefficient and compute momentum fluxes.
7. Compute the vertical mixing of momentum, heat, and moisture.

---- User Note ----

Some of the storage spaces for the high-resolution PBL are used to store the variables in BLKPBL. They are: **REGIME, PBL, MOL, HOL, HFX, QFX, ZOL, DUM1, DUM2, SCR1, SCR2. SCR1** is used to sum temperature tendency, and **SCR2** is used to sum moisture tendency. **DUM1** is used to compute average **PSB**, and **DUM2** is the average horizontal wind speed.

5.14 BLNUDGD

Subroutine BLNUDGD is an analysis-nudging FDDA routine that computes the surface-analysis nudging term in the PBL for a general model variable.

BLNUDGD is called from subroutine NUDGD. It calls subroutines BLBRGD, INTPSGD, and MAPSMP.

Arguments: **MTIM, MVAR, J, XB, xten*, pten*, PSTF, PSTO, GX, ID, WXY, WXY2, blwnv*, WPBL, TFAC, ZFAC, GP, IVAR, IN, KPBLT, SFCOBS, SFCTIM, QSATF, IQCHK, timb*, time*, ntb*, nte*, [SCR2D], [BLDUM2D], blpsoc*, blpsod*, REGJ, ZNTJ.**

Common blocks: ADDR4, HUGE, NESLEV, PARAM2, and PARFDDA.

Data: None.

---- Program detail ----

BLNUDGD computes the surface-analysis nudging term within the PBL of the tendency equation (**XTEN**) for **XB**, where **XB** is a general model field at time t-1. The sequence of basic subroutine tasks is:

1. If necessary, calculate new values of **TIMB** and **TIME**, and position counters **NTB** and **NTE**, which bracket **XTIME** with two surface analyses (BLBRGD).
2. If nudging p* from surface-analysis files, perform temporal interpolation (INTPSGD).
3. Calculate the storage array, **BLWNV**, for the horizontal weighting function based on surface data density.
4. Adjust the observed surface wind for the lowest model layer using similarity relationships based on roughness.
5. Adjust observed q_v if supersaturated.
6. Compute analysis-nudging term of the tendency equation within the PBL.
   a) If **ID** = 0, compute the normal nudging term for U, V, T, or q_v.
   b) If **ID** = 1, compute the nudging term in the continuity (**PTEN**) equation in the hydrostatic version of the model.
   c) If **ID** = 3, compute the moisture nudging term using observed precipitation data.

5.15 BLW

Subroutine BLW is an analysis-nudging FDDA routine that computes the horizontal weighting function for surface-analysis nudging within the PBL, based on surface data density (**BLWXY**).

BLW is called from subroutine IN4DGD. It calls subroutine SMTHER.

Arguments: **IL, JL, WTTOP, blxwy*, [BLDUM2D], RINBLW, DX, TIMANLS, MDATES, IN**.

Common blocks: HUGE and NESLEV.

Data: None.

5.16 BUFSLGD

Subroutine BUFSLGD is an analysis-nudging FDDA routine that "decouples" (e.g., u from p*u) and interpolates observed analyzed values for a general model variable to time t-1 for use in calculating the analysis-nudging term.

BUFSLGD is called from subroutines SOLVE1 and SOLVE3.

Arguments: **J, XOB, XOBTEN, xobjk, PSTO, FDTIM, NV, IN.**

Common blocks: ADDR4, HUGE, NESLEV, PARAM2, and PARFDDA.

Data: None.

5.17 BUFVDGD

Subroutine BUFVDGD is an analysis-nudging FDDA routine that computes the observed minus model forecast vorticity difference, **VORDIF**, for use in the analysis nudging of vorticity.

BUFVDGD is called from subroutine SETFD.

Arguments: **UOB, VOB, UOBTEN, VOBTEN, UB, VB, PSTO, FDTIM, MSD, MSX, vordif, PSTF, IN**.

Common blocks: ADDR4, HUGE, NESLEV, PARAM2, and PARFDDA.

Data: None.

5.18 CADJMX

Subroutine CADJMX sets up the indices and matrices needed for the Gaussian elimination method used for the dry-convective adjustment.

CADJMX is called from subroutine CONVAD. It calls subroutine GAUSS.

Arguments: **KB, KE, PI, S, nadj, DSIGMA, DTHDPC, P, [AX], [B], [C], D,** y, KL, MKX.

Common blocks: None.

Data: None.

5.19 CHKNST

Subroutine CHKNST checks whether a nest should be activated and initialized (INITNEST and OUTTAP), moved and initialized (INITNEST and OUTTAP), or deactivated.

CHKNST is called from the main program and from subroutine INIT. It calls subroutines ADDRX1C, ADDRX1N, INITNEST, OUTTAP, and SPINIT.

Arguments: **iexec, NESCOU.**

Common blocks: ADDR0, ADDR4, ADDRNN, HUGE, MIC, NESLEV, PARAM2, PARAM3, PARFDDA, PBLDIM, and PMOIST.

Data: None.

5.20 CLODWD

Subroutine CLODWD calculates the efficiency of generation of kinetic energy for downdrafts in the Arakawa-Schubert convection scheme.

CLODWD is called from subroutine ARASCH.

Arguments: **L1, KS, KMIN, HCD, QES, HES, ZU, Z, TEMPP, KDIM, axd, KNUM.**

Common blocks: None.

Data: None.

5.21 CLOUDW

Subroutine CLOUDW calculates the efficiency of generation of kinetic energy for updrafts in the Arakawa-Schubert convection scheme.

CLOUDW is called from subroutine ARASCH.

Arguments: **L1, HC, QES, HES, ZU, Z, TEMPP, KDIM, ax, KBCON, KNUM, KTOP.**

Common blocks: None.

Data: None.

5.22 CLOUDWS

Subroutine CLOUDWS calculates the efficiency of generation of kinetic energy for shallow clouds in the shallow convection scheme.

CLOUDWS is called from subroutine SHALLOW.

Arguments: **L1, HC, QES, HES, ZU, Z, TEMPP, KDIM, ax, KBCON, KNUM, KTOP.**

Common blocks: None.

Data: None.

5.23 CONADV

Subroutine CONADV computes the amount of dry air and water substance advected through the east-west and north-south lateral boundaries. The advection unit is converted to kg.

CONADV is called from the main program.

Arguments: **ID, IM.**

Common blocks: ADDR1, ADDR2, ADDR4, HUGE, NESLEV, and PARAM3.

Data: None.

5.24 CONMAS

Subroutine CONMAS computes the total amount of dry air and water substance in the domain. Total mass (kg), including advection, evaporation, and rainfall are calculated.

CONMAS is called from the main program. It calls subroutine TMASS.

Arguments: **ID, IM, MC.**

Common blocks: ADDR1, ADDR2, ADDR4, HUGE, NESLEV, and PARAM3.

Data: None.

5.25 CONV3

Subroutine CONV3 is an analysis-nudging FDDA routine that buffers a 2-D array **SLAB(IMAX, JMAX)** into a 4-D array **SFCOBS(MTIM, MVAR, IMAX, JMAX)**, and also places the model-relative time of the 2-D array into the 1-D array **SFCTIM**.

CONV3 is called from subroutine IN4DGD.

Arguments: **SLAB, sfcobs, MIX, MJX, TIMANL, sfctim, MTIM, MVAR, IMAX, JMAX, NT, NV.**

Common blocks: None.

Data: None.

5.26 CONVAD

Subroutine CONVAD is the main routine used for the dry-convective adjustment.

CONVAD is called from subroutines SOLVE1 and SOLVE3. It calls subroutine CADJMX.

Arguments: **ILXM, KL, NUMNES, J.**

Common blocks: HUGE, MIC, NESLEV, PARAM2, PARAM3, PARFDDA, PBLDIM and PMOIST.

Data: **DTDPC** /-0.03/ critical lapse rate (K cb^{-1}).

---- Program Detail ----

CONVAD performs the dry convective adjustment and mixes water vapor whenever the vertical derivative of potential temperature exceeds the critical lapse rate. The sequence of the basic subroutine tasks is:

1. Compute the potential temperature.
2. Determine if multiple-layer adjustment is needed by comparing the derivative over two or more layers to a critical lapse rate, **DTDPC** = -0.03 K cb^{-1}.
3. Perform layer adjustment by calling CADJMX.

5.27 CUP

Subroutine CUP is the Grell cumulus parameterization routine.

CUP is called from subroutine CUPARA2. It calls subroutine MAXIMI and MINIMI.

Arguments: **QCRIT, T, Q, Z1, TN, QO, PO, pre*, P, outtem*, outq*, DTIME, PSUR, VSP, ISTART, IEND, KDET.**

Common blocks: HUGE, NESLEV, PARAM3, and PMOIST.

Data: None.

---- Program Detail ----

CUP computes the moisture cycle using the Grell cumulus parameterization scheme that includes downdrafts. The sequence of the basic subroutine tasks is:

1. Compute the heights of the sigma levels.
2. Calculate the moist static energy **HE** and the saturation moist static energy **HES**.
3. Determine the level with the highest moist static energy (MAXIMI).
4. Determine the cloud base, cloud top, and level of largest negative buoyancy (MINIMI).
5. Calculate the available buoyant energy.
6. Downdraft calculations.
   a) Calculate the precipitation efficiency in terms of windshear.
   b) Determine thermodynamic profiles in downdrafts.
   c) Calculate normalized downdraft strength in terms of precipitation efficiency and normalized condensation and precipitation.
   d) Calculate effects on the environment per unit mass.
7. Recalculate the environmental thermodynamic profiles, available buoyant energy.
8. Calculate mass fluxes and feedbacks

5.28 CUPARA1

Subroutine CUPARA1 is the Kuo-Anthes cumulus parameterization routine.

CUPARA1 is called from subroutines SOLVE1 and SOLVE3.

Arguments: **J, IN, qvten*, tten*.**

Common blocks: ADDR1, ADDR2, ADDR4, ASSEL, HUGE, MIC, NESLEV, NONHYD, PARAM2, PARAM3, PARFDDA, PBLDIM and PMOIST.

Data: **CDSCLD** /0.3/ critical cloud depth in sigma units. If the depth is less than the critical value, the convection is suppressed.
DLT /3.0/ temperature difference (K) used to allow over-shooting of cloud top above level of zero buoyancy.
PERQ /1.0E-3/ perturbation mixing ratio (kg kg^{-1}) to initiate the cloud.
PERT /1.0/ perturbation temperature (K) to initiate the cloud.

---- Program Detail ----

CUPARA1 computes the moisture cycle using a Kuo-Anthes type cumulus parameterization scheme. The sequence of the basic subroutine tasks is:

1. Compute the moisture convergence in a column.
2. Determine if convection exists by comparing the total moisture convergence to the threshold convergence, QDCRIT = 3.0E-7.

3. Compute cloud base and top, and check stability for convection potential.
4. If convection exists, compute convective flux of water vapor and latent heat.
5. If convection does not exist, compute vertical flux divergence term.

5.29 CUPARA2

Subroutine CUPARA2 sets up the variables for the Grell convection scheme (CUP).

CUPARA2 is called from subroutines SOLVE1 and SOLVE3. It calls subroutine CUP.

Arguments: **TA, QVA, PSA, tten*, qten*, rainc*, J, HT, UA, VA, INEST.**

Common blocks: ADDR4, HUGE, NESLEV, NONHYD, PARAM2, PARAM3, PARFDDA, and PMOIST.

Data: None.

5.30 CUPARA3

Subroutine CUPARA3 sets up the variables for the Arakawa-Schubert convection scheme (ARASCH).

CUPARA3 is called from subroutine SOLVE1. It calls subroutine ARASCH.

Arguments: **TA, QVA, PSA, tten*, qten*, rainc*, J, HT, UA, VA, NUMNES.**

Common blocks: ADDR4, ARASCH, HUGE, NESLEV, NONHYD, PARAM2, PARAM3, PARFDDA, and PMOIST.

Data: None.

----User Note----

Set **IARASC** = 1 for this option.

5.31 DECPU

Subroutine DECPU decouples the U, V variables from p*.

DECPU is called from subroutines SOLVE1 and SOLVE3.

Arguments: **KZZ, ften*, FB, PSB, XK, MSF, [SCR], C203, J, IEND, JEND, IND, INEST.**

Common blocks: ADDR3, HUGE, and NESLEV.

Data: None.

5.32 DIFFU

Subroutine DIFFU computes the diffusion term for p*U and p*V using a fourth-order scheme for the interior and a second-order scheme for the boundaries. The diffusion term is calculated on a constant sigma surface.

DIFFU is called from subroutines SOLVE1 and SOLVE3.

Arguments: **C203, FB, FTEN, IEND, IND, INEST, J, JEND, MSF, PSB, SCR, and XK.**

Common blocks: HUGE and NESLEV.

Data: None.

5.33 DIFFUT

Subroutine DIFFUT computes the diffusion term for p*T. The routine computes the derivatives of T along the sigma surfaces.

DIFFUT is called from subroutines SOLVE1 and SOLVE3.

Arguments: **KZZ,ften*, FB, PSB, XK, C201, J, IEND, JEND, INEST.**

Common blocks: HUGE, NESLEV, and PARAM3.

Data: None.

5.34 DIVG [multi-tasked]

Subroutine DIVG computes the divergence from U and V for split-explicit scheme.

DIVG is called from subroutine SPSTEP2.

Arguments: **[U], [V], z, MSFD, MSFX, DX2, KL, IMIN, JMIN, IMAX, JMAX.**

Common blocks: HUGE and NESLEV.

Data: None.

5.35 DOTS

Subroutine DOTS converts cross-point arrays to dot-point arrays using a four-point average for the interpolation.

DOTS is called from subroutines BDYUV, INITNEST, INTPSGD, SETUPGD, and SOLVE3.

Arguments: **SLAB1, slab2, IS1, IS2, ID3, ID4.**

Common blocks: None.

Data: None.

5.36 ENTR

Subroutine ENTR calculates the in-cloud moist static energy and cloud top for the Arakawa-Schubert convection scheme.

ENTR is called from subroutine ARASCH.

Arguments: **CD, KBC, H, hc, HSAT, ENT, KDIM, LP, KB, P, HKB, KNUM, ktop.**

Common blocks: None.

Data: None.

5.37 ENTRS

Subroutine ENTRS calculates the in-cloud moist static energy and cloud top for the shallow convection scheme.

ENTRS is called from subroutine SHALLOW.

Arguments: **CD, KBC, H, hc, HSAT, ENT, KDIM, LP, KB, P, HKB, KNUM, ktop.**

Common blocks: None.

Data: None.

5.38 EQUATE

Subroutine EQUATE puts select values of a 3-D input array **FIN(IXI, JXI, KXI)** into select locations of a 3-D output array **FOUT(IXO, JXO, KXO).**

EQUATE is called from subroutines IN4DGD, INITNEST, OUTTAP and RDINIT.

Arguments: **FIN, IXI, JXI, KXI, fout, IXO, JXO, KXO.**

Common blocks: None.

Data: None.

5.39 ERROB

Subroutine ERROB is an observation-nudging FDDA routine that calculates the difference between the observation and the model forecast at the observation location for all model variables. The difference for each variable and observation location is stored in array, **ERRF**, and used for observation nudging within the specified time window (**TWINDO**).

ERROB is called from subroutine SETFD.

Arguments: **IN, UB, VB, TB, QVB, PSB.**

Common blocks: ADDR4, ADDR8, HUGE, NESLEV, PARAM2, and PARFDDA.

Data: None.

5.40 EXAINT

Subroutine EXAINT is called whenever a nest is initialized or moved. It sets up input arrays for the interpolation routine SINT.

EXAINT is called from subroutines INITNEST, STOTNDI, and STOTNDT. It calls subroutine SINT.

Arguments: **TA, IYY, JXX, tan, IYYN, JXXN, ISOUTH, JWEST, ICRSDOT.**

Common blocks: DEPAR2, HUGE, and NESLEV.

Data: None.

5.41 EXCHANI

Subroutine EXCHANI exchanges boundary values between nests for particular **I**-index values from **JSTARTN** to **JENDN**.

EXCHANI is called from subroutine BDYOVL1.

Arguments: **IARG, IARGN, IOVE, JSTART, JSTARTN, JENDN, K, ARR, arrn*, ARRB, arrnb*, arrbb*, arrbt*, KZZ, NSPG, DT, ISTOP.**

Common blocks: None.

5.42 EXCHANJ

Subroutine EXCHANJ exchanges boundary values between nests for particular **J**-index values from **ISTARTN** to **IENDN**.

EXCHANJ is called from subroutine BDYOVL1.

Arguments: **ARG, JARGN, JOVE, ISTART, ISTARTN, IENDN, K, ARR, arrn*, ARRB, arrnb*, arrbb*, arrbt*, KZZ, NSPG, DT, ISTOP.**

Common blocks: None.

Data: None.

5.43 EXMOISR

Subroutine EXMOISR is an explicit scheme for computing the moisture tendencies that includes an ice option that allows for supercooled liquid water.

EXMOISR is called from subroutines SOLVE1 and SOLVE3.

Arguments: **J, IN, IM, qcten*, qrten*, qiten*, qniten*, qvten*, tten*, IST, IEN.**

Common blocks: ADDR1, ADDR2, ADDR4, JRG, MIC, NHCNST, NONHYD, PARAM3, PBLDIM, and PMOIST.

Data: None.

---- Program detail ----

EXMOISR computes the tendencies of water vapor, cloud water, cloud ice, snow, and rainwater using cloud microphysics parameterization for the autoconversion, accretion, evaporation, deposition/sublimation, melting, and freezing processes. This routine allows for supercooled liquid water and has more sophisticated melting processes than EXMOISS. The sequence of the basic subroutine tasks is:

1. Calculate the saturation vapor pressure.
2. Compute the initiation of cloud ice (**PRI**), freezing of cloud droplets (**NUFCI**), accretion of cloud ice by snow (**PRAI**), collection of cloud water by snow (**PSACW**), deposition/sublimation of cloud ice (**PRD**), autoconversion of cloud ice to snow (**PRCI**), and deposition/sublimation of snow (**PREI**).
3. Compute the autoconversion of cloud to rain (**PRC**), accretion of cloud by rain (**PRA**), and evaporation of rain (**PRE**).
4. Compute melting of snow (**PSMLT**) and evaporation of melting snow (**PMLTEV**).
5. Calculate tendencies ensuring positive water/ice after adding them.
6. Calculate condensation term for vapor to cloud.
7. Compute the fallout terms (with split time steps) for rain and snow.
8. Include effects of melting and freezing of cloud ice.

----User Note----

IICE = 1 to use this option, and **IEXMS** = 1

5.44 EXMOISS

Subroutine EXMOISS is an explicit scheme for computing the moisture tendencies that includes a simple ice option.

EXMOISS is called from subroutines SOLVE1 and SOLVE3.

Arguments: **J, IN, IM, qcten*, qrten*, qvten*, tten*, SCR9, IST, IEN.**

Common blocks: ADDR1, ADDR2, ADDR4, ASSEL, HUGE, MIC, NESLEV, NHCNS, NHCNST, NONHYD, PARAM3, PBLDIM, and PMOIST.

Data: None.

---- Program detail ----

EXMOISS computes the tendencies of water vapor, cloud water, cloud ice, snow, and rainwater using cloud microphysics parameterization for the autoconversion, accretion, evaporation, deposition / sublimation, melting and freezing processes. The sequence of the basic subroutine tasks is:

1. Calculate the saturation vapor pressure
2. Compute the autoconversion (**PRC**), accretion (**PRA**), and evaporation (**PRE**) terms without ice (all production terms are based on variables at time t-1).
3. Compute the initiation of cloud ice (**PRI**) and deposition / sublimation of cloud ice (**PRD**) terms.
4. Re-adjust **PRC, PRA, PRI,** and **PRD** if total production is greater than that available.
5. Compute p*T, $p*q_v$, and $p*q_c$ (at time t+1) without the condensation term, and $p*q_r$ without the fallout term.
6. Compute the condensation and fallout (with split time steps) terms.
7. Include effects of freezing and melting of particles depending on the sign of the vertical motion between sigma layers.

---- User note ----

Initially, **TAOUT, QAOUT, SCR2, SCR3, SCR6,** and **SCR8** are used to store 2-D slices of T, q_v, q_c, q_r, pressure, and air density at time t-1. Set **IEMXS** = 1 for this option.

5.45 FDAOFF

Subroutine FDAOFF resets the analysis-nudging and observation-nudging FDDA switches for a particular domain (**NUMNES**).

FDAOFF is called from subroutine OUTPUT.

Arguments: **NUMNES.**

Common blocks: HUGE, NESLEV, PARAM2 and PARFDDA.

Data: None.

5.46 FEEDBK [multi-tasked]

Subroutine FEEDBK computes the feedback effects from the nested domain to the coarse domain.

FEEDBK is called from subroutines NSTLEV1, NSTLEV2, and NSTLEV3. It calls subroutines MDOTS, SEAPRS, SMTHER, and SMT2.

Arguments: **INEST, IYY, JXX.**

Common blocks: ADDR0, ADDR1, ADDR2, ADDR4, ADDRN1, ADDRN2, ADDRN4, ADDRNN, ADDRV, ADDRVN, HUGE, MIC, NESLEV, NONHYD, NNNHYD, PARAM2, PARAM3, PARFDDA, PBLDIM, and PMOIST.

Data: None.

---- Program detail ----

FEEDBK interpolates (if IFEED = 1) coarse-grid values in the nested domain from the nested values using a nine-point averaging technique. If IFEED > 1, it sets coarse values to coincident nested values. The order of variables for the calculations are: p*, T_g, T, W, and p' (if nonhydrostatic), q_v, q_c, q_r, q_i, q_s, U, and V. The interpolated data are smoothed using a 2-pass smoother-desmoother (SMTHER) if IFEED = 3.

5.47 FILL

Subroutine FILL puts select values of a 2-D input array **HT(MIX, MJX)** into select locations of a 2-D output array **HSCR1(IYYN, JXXN).**

FILL is called from subroutine INITNEST.

Arguments: **HT, hscr1, MIX, MJX, IYYN, JXXN, IEND, JEND.**

Common blocks: None.

Data: None.

5.48 FILSLB

Subroutine FILSLB overwrites new nest values (of a moving nest) with old nest values in regions where they overlap.

FILSLB is called from subroutine INITNEST.

Arguments: **anew, AOLD, ISOUTH, ISOUTHO, JWEST, JWESTO, MIX, MJX, ICRSDOT.**

Common blocks: None.

Data: None.

5.49 GAUSS

Subroutine GAUSS is a routine used for the dry-convective adjustment. It employs the Gaussian elimination method to solve a matrix system.

GAUSS is called from subroutine CADJMX.

Arguments: **AX, B, C, [D], y*, LM, MKX.**

Common blocks: None.

Data: None.

5.50 HADV

Subroutine HADV computes the horizontal advection (flux-divergence) terms using second-order finite differences. The upstream values are optionally used for the q_c and q_r tendencies to ensure positive definiteness.

HADV is called from subroutines SOLVE1 and SOLVE3.

Arguments: **ften*, UA, VA, F, MSF, DXX, J, IND, IN.**

Common blocks: ADDR4, HUGE, NESLEV, and PARAM3.

Data: None.

5.51 HEIPRE

Subroutine HEIPRE calculates heights for the deep (Arakawa-Schubert) and shallow convection schemes.

HEIPRE is called from subroutines ARASCH and SHALLOW.

Arguments: **P, z, T, Z1, KDIM, KS, PSURF.**

Common blocks: None.

Data: None.

5.52 HIRPBL

Subroutine HIRPBL is a high-resolution PBL routine based on a model developed by Blackadar.

HIRPBL is called from subroutines SOLVE1 and SOLVE3. It calls subroutines SFCRAD and SLAB.

Arguments: **IYY, JXX, KZZ, J, IN, NST, uten*, vten*, qcten*, scr1*, scr2*, [KEPBLH], [KWPBLH], qiten*, qniten*.**

Common blocks: ADDR1, ADDR2, ADDR4, ASSEL, HUGE, MIC, NESLEV, NHCNS, NHCNST, NONHYD, PARAM2, PARAM3, PARFDDA, PBLDIM, and PMOIST.

Data:
C1 /0.2721655/ constant ($\sqrt{2/27}$) used in Priestley's equation.
C2 /-0.33333/ constant used in Priestley's equation.
CZO /0.032/ constant used to compute roughness length over water.
ENTRMT /0.2/ entrainment coefficient used for free convection.
KA /2.4E-5/ background molecular diffusivity ($m^2\ s^{-1}$) used to compute latent heat flux.
KZO /1.0/ background diffusion coefficient ($m^2\ s^{-1}$).
OZO /1.0E-4/ constant (m) used to compute roughness length over water.
SZKM /1600./ constant (m^2) used to compute the eddy diffusion coefficients.

---- Program Detail ----

HIRPBL uses horizontal winds, temperature, cloud water, and water vapor at half-sigma levels to compute the bulk Richardson number of surface layers to determine stability classes that define basic parameters affecting the PBL forecast. The algorithms are written to allow the PBL to be described as a discrete, variable number of model layers of arbitrary thickness. The sequence of the basic subroutine tasks is:

1. Convert ground temperature to potential temperature.
2. Decouple flux-form variables to give U, V, T, potential temperature, virtual potential temperature, virtual temperature, q_v, and q_c at cross points at time step **KTAU**-1.
3. Compute the height of the full- and half-sigma levels and the layer thickness.

4. Calculate the critical Richardson number.
5. Begin iteration loop steps.
6. Initialize the vertical tendencies and calculate the bulk Richardson number of the surface layer.
7. Diagnose basic parameters for the appropriate stability classes
   a) stable (night) conditions,
   b) damped mechanical convection,
   c) forced convection, and
   d) free convection.
8. Calculate the frictional velocity and surface radiation (SFCRAD).
9. Compute the surface sensible and latent heat fluxes, and momentum fluxes.
10. Compute ground temperature (SLAB).
11. Compute PBL prognostic terms (explicit computation of surface layer values for classes 1-3, and PBL computations free convection class.)
12. Get PBL tendencies above the surface layer and sum the fluxes.
13. Compute the weighting factor (**WGT**) for simulating reduced mixing rates near the top of the PBL.
14. Compute the surface layer tendencies.
15. Apply vertical diffusion (include moist vertical diffusion if **IMVDIF** = 1).
16. End iteration loop.
17. Convert all tendencies to flux form and store in **SCR1** and **SCR2** (T and q_v).
18. Apply Asselin filter to ground temperature and update it.
19. Convert ground potential temperature to temperature.

---- User Note ----

SCR1, SCR2, SCR3, and **SCR4** store **TTEN**, **QVTEN**, temperature, and virtual temperature respectively. **DUM1** is used in computing the height of the full- and half-sigma levels.

5.53 IN4DGD

Subroutine IN4DGD is an analysis-nudging FDDA routine that reads in the surface and upper-air analyses which are used for analysis nudging during the period **XTIM1** to **XTIM2**.

IN4DGD is called from subroutines INIT, INITNEST, and SETFD. It calls subroutines BLW, CONV3, EQUATE, JULGMT, and NOPRO.

Arguments: **IN, IYY, JXX, KZZ.**

Common blocks: ADDR4, ADDR5, ADDR6, ADDR7, CFD, CFDDAGD, HEADER, HEADERC,HUGE, NESLEV, PARAM2, PARAM3, PARFDDA, and SIZE.

Data: None.

---- Program detail ----

IN4DGD reads in upper-air and surface analyses files and selects data based on **TIMANL**, **DIFTIM, TIMANLS** and **DIFTIMS**. The sequence of the basic subroutine tasks is:

1. Compute **TIMANL**, the model-relative time of the input data (JULGMT), and check for consistency with **XTIME**, the current time step model time.
2. Read in observed upper-air analyses **UBO, VBO, TBO, QBO** and **PSO** at two time levels (**XTIM1** and **XTIM2**).
3. Calculate observed tendencies **UBOTEN, VBOTEN, QBOTEN, TBOTEN,** and **PSOTEN** from **XTIM1** to **XTIM2**.
4. Check whether surface analysis nudging is to be employed. If yes:
   a) Read in surface analyses between **XTIM1** and **XTIM2** and put into **DUM2D** (EQUATE).
   b) Compute horizontal weighting function (BLW) which is used for surface-analysis nudging within the PBL.
   c) Buffer 2-D array **DUM2D** into 4-D array **SFCOBS** (CONV3).
5. Print information concerning the details of the parameter choices related to analysis nudging.

5.54 IN4DOB

Subroutine IN4DOB is an observation-nudging FDDA routine that reads in the observations (all variables) used for observation nudging within the specified time window (**TWINDO**).

IN4DOB is called from subroutines INIT, INITNEST, and SETFD.

Arguments: **INEST, XTIME, KTAU, KTAUR, DTMIN.**

Common block: ADDR8, HUGE, NESLEV, PARAM2, PARAM3, and PARFDDA.

Data: **IEOF** /0/ flag indicator (0 or 1) for end of file. **NMOVE** /0/ initialize (set to 0) number of observations to be removed. **NVOLA** input logical unit number constant for **NVOL**, where **NVOL** = **NVOLA** + **INEST**-1.

---- Program detail ----

IN4DOB reads in observation data files and selects only observations that are contained within the designated time window. The sequence of the basic subroutine tasks is:

1. Define time window limits in model-relative times.
2. Check for old observations that can be discarded.
3. Read in all observations within the time window.
4. Convert the observation time from Julian date and **GMT** form to model-relative (forecast) time.
5. Determine if maximum number of observations exceeds allowable observation array size. If so, stop and increase size parameter **NIOBF**, or optionally, decrease size of time window (generally not a good idea).
6. Print information concerning the details of parameter choices related to observational nudging.

5.55 INIT

Subroutine INIT gets (or specifies) the coarse-grid initial conditions and boundary conditions.

INIT is called from the main program. It calls subroutines ADDRX1C, BDYIN, CHKNST, IN4DGD, IN4DOB, OUTTAP, RDINIT, SOLAR1, and TMASS.

Arguments: **IEXEC, IX, JX.**

Common blocks: ADDR0, ADDR1, ADDR2, ADDR4, ADDR5, ADDR6, ADDR7, ADDR8, ADDRNN, ASSEL, CFD, CFDDAGD, HUGE, LANDUSE, MIC, NESLEV, NHCNS, NHCNST, NONHYD, NONHYDB, PARAM2, PARAM3, PARFDDA, and PBLDIM.

---- Program detail ----

INIT reads and stores initial and boundary conditions for the coarse-grid domain, and initializes some of the input parameter variables. The sequence of the basic subroutine tasks is:

1. Variables used to store rainwater, cloudwater, snow and ice are initialized to 0.
2. Total mass and PBL variables are initialized to 0.
3. Read in the initial conditions (RDINIT).
4. Invert the map scale factor and set the Asselin filter mask.
5. Calculate the total mass of dry air and water substance (TMASS).
6. Specify surface parameters according to land-use category and read in the slab temperature.

7. If FDDA is employed, call subroutine that reads in the observations (for observation nudging) and / or analysis (for analysis nudging).
8. Initialize nests (CHKNST).
9. Read in boundary conditions (BDYIN).
10. Compute solar declination and set up the output times.

5.56 INITNEST [multi-tasked]

Subroutine INITNEST interpolates (or reads in) the nested-grid initial conditions.

INITNEST is called from subroutine CHKNST. It calls subroutines ADDRX1C, ADDRX1N, DOTS, DOTS2, EQUATE, EXAINT, FILL, FILSLB, IN4DGD, IN4DOB, OVLCHK, RDINIT, SEAPRS, SUBCH, and TMASS.

Arguments: **IYY, JXX, KZZ, IYYN, JXXN, KZZN, IYYM, JXXM, KZZM, IYYNM, JXXNM, KZZNM, NESTII, NESTJJ, NESCOU, HTNO, INSTES, ISOUTHO, JWESTO.**

Common blocks: ADDR0, ADDR1, ADDR2, ADDR4, ADDR5, ADDR6, ADDR7, ADDR8, ADDRN1, ADDRN2, ADDRN4, ADDRNN, ADDRV, ADDRVN, ASSEL, CFD, CFDDAGD, HUGE, LANDUSE, MIC, NESLEV, NHCNS, NHCNST, NNCNS, NNNHYD, NONHYD, PARAM2, PARAM3, PARFDDA, PBLDIM, and PMOIST.

---- Program detail ----

INITNEST interpolates (or reads in) and stores the initial conditions for the coarse domain, and initializes some of the input parameter variables. The sequence of the basic subroutine tasks is:

1. Adjust corner points of the nest, if a moving nest (SUBCH).
2. Initialize and set up the coordinates of the nest.
3. Define some constants for grid size and time step.
4. Check if the nest is to be overlaid with other nest (OVLCHK).
5. Recalculate sea-level pressure on coarse domain, interpolate them to the nest, and calculate p_s on the nest based on nested terrain values. Note: EXAINT is used for all interpolations in INITNEST.
6. Interpolate U, V, T, q_v, q_c, and q_r arrays.
7. Interpolate some PBL variables.
8. Interpolate nonhydrostatic arrays: reference temperature, p', and W.
9. Interpolate 2-D terrestrial arrays.
10. Get total mass of dry air and water substance in the nested domain (TMASS).

11. Initialize some arrays pertaining to the Asselin filter.
12. Specify some surface parameters according to land-use category.
13. Read in nested initial conditions (RDINIT) if you already have a nested initial conditions file.
14. Initialize some nested FDDA parameters.
15. If FDDA, get observations and / or analyses needed (IN4DOB or IN4DGD).

5.57 INITSAV

Subroutine INITSAV calls the routine SAVREAD that reads restart file information needed for the initialization procedure for a restart. It also initializes some arrays pertaining to the Asselin filter.

INITSAV is called from the main program. It calls subroutines ADDRX1C, ADDRX1N, SAVREAD, SOLAR1, and SPINIT.

Arguments: **iexec.**

Common blocks: ADDR0, ADDR4, ADDRN4, ADDRNN, ASSEL, HUGE, NESLEV, PARAM2, PARAM3, PARFDDA, and PMOIST.

Data: None.

5.58 INTPSGD

Subroutine INTPSGD is an analysis-nudging FDDA routine used for temporal and dot-point interpolation of observed p*.

INTPSGD is called from subroutines BLNUDGD and SETFD. It calls subroutine DOTS

Arguments: **PSO, PSOTEN, psoc, psod, FDTIM, IN.**

Common blocks: ADDR4, HUGE, NESLEV, PARAM2, and PARFDDA.

Data: None.

5.59 INVMTX

Subroutine INVMTX is used to invert some matrices for the split-explicit scheme.

INVMTX is called from subroutine VMODES.

Arguments: **A, NA, v, NV, N, D, IP, ier.**

Common blocks: None.

Data: None.

5.60 JULGMT

Subroutine JULGMT is an FDDA routine that converts standard date and time information (**MDATE**) from the input data into model-relative time (**TIMANL**) and/or the Julian day and **GMT** form used by the FDDA routines.

JULGMT is called from subroutines IN4DGD and SETUPGD.

Arguments: **MDATE, julgmtn*, timanl*, JULDAY, GMT, IND.**

Common blocks: None.

Data: None.

5.61 KERHEL

Subroutine KERHEL calculates net cloud effect per unit mass on the environment for the Arakawa-Schubert convection scheme.

KERHEL is called from subroutine ARASCH.

Arguments: **VAR, R, LP, ZU, HKB, HC, della, P, Z, KB, KDIM, xvar, MBDT, KBEG, xhkb, CD, HCD, ZD, ITEST, KDET, EDT, PSU, KNUM, KTOP.**

Common blocks: None.

Data: None.

5.62 KERHELS

Subroutine KERHELS calculates net cloud effect per unit mass on the environment for the shallow convection scheme.

KERHELS is called from subroutine SHALLOW.

Arguments: **VAR, R, LP, ZU, HKB, HC, della, P, Z, KB, KDIM, xvar, MBDT, KBEG, xhkb, CD, KDET, PSU, KNUM, KTOP.**

Common blocks: None.

Data: None.

5.63 LWRAD

Subroutine LWRAD calculates the longwave component of the atmospheric radiative temperature tendencies and surface flux.

LWRAD is called from subroutines SOLVE1 and SOLVE3.

Arguments: **J**

Common blocks: ADDR1, ADDR2, ADDR4, HUGE, NESLEV, NHCNS, NHCNST, NONHYD, PARAM3, and RADIAT.

Data: None.

---- Program Detail ----

1. Presets column variables and radiation constants.
2. Calculates various path lengths for each model level.
3. Calculates downward fluxes for each model level by upward integration of path lengths and applying broadband emissivity, overlap approximation for CO_2 and assuming radiative grey clouds and precip.
4. Calculates upward fluxes for each model level by downward integration, accounting for ground emissivity.
5. Calculate longwave heating rate from radiative flux divergence.
6. Save downward longwave flux at surface (**GLW**) for ground heat budget.

----User Note----

Set **IRDDIM** = 1 for this option.

5.64 MAPSMP

Subroutine MAPSMP is one of the two main printer output routines for MM5. It prints select values of two-dimensional, five-significant-digit data fields.

MAPSMP is called from subroutines BLNUDGD, NUDGD, NUDOB, and OUTPRT. It calls subroutine VTRAN.

Arguments: **FLD, IYY, JXX, IA, IB, INY, JA, JB, JNX, CONST, ICHOS, NAME, TIME.**

Common blocks: None.

Data: KSIGT /5/ number of significant digits for data printout.

5.65 MAXIM

Subroutine MAXIM determines the level of the maximum value of a 1-D vertical **ARRAY(KDIM)**.

MAXIM is called from subroutines ARASCH and SHALLOW.

Arguments: **ARRAY, KDIM, KS, KE, max.**

Common blocks: None.

Data: None.

5.66 MAXIMI

Subroutine MAXIMI determines the level of the maximum value of each vertical string of a 2-D slice **ARRAY(MIX, MKX)**.

MAXIMI is called from subroutine CUP.

Arguments: **ARRAY, MIX, MKX, KS, KE, max, ISTART, IEND.**

Common blocks: None.

Data: None.

5.67 MINIM

Subroutine MINIM determines the level of the minimum value of a 1-D vertical **ARRAY(KDIM)**.

MINIM is called from subroutines ARASCH and SHALLOW.

Arguments: **ARRAY, KDIM, KS, KEND, kt.**

Common blocks: None.

Data: None.

5.68 MINIMI

Subroutine MINIMI determines the level of the minimum value of each vertical string of a 2-D slice **ARRAY(MIX, MKX).**

MINIMI is called from subroutine CUP.

Arguments: **ARRAY, MIX, MJX, KS, KEND, kt, ISTART, IEND.**

Common blocks: None.

Data: None.

5.69 MM5

MM5 is the main driver program for the predictive component of the MM5 modeling system.

MM5 calls subroutines ADDALL, ADDRX1C, ADDRXIN, BDYIN, BDYVAL, CHKNST, CONADV, CONMAS, INIT, INITSAV, NESLEV1, OUTPUT, PARAM, SHUTDO, SOLVE1, SOLVE3, SPINIT, SPLITF, and STOTNDI.

Arguments: None.

Common blocks: ADDR0, ADDR4, ADDRN4, ASSEL, HUGE, NESLEV, PARAM2, PARAM3, and PARFDDA.

Data: None.

---- Program detail ----

The sequence of the program tasks is:

1. Preset pointer addresses of all variables on all domains (ADDALL).
2. Set pointer addresses specific to a given domain and variable (ADDRX1C).
3. Setup (or define) model parameters (PARAM**)**.
4. If **IFREST**=.T.; read restart file (INITSAV), initialize some arrays, and check whether a nest should be moved, activated, or initialized (CHKNST).
5. Call the routine (INIT) that gets the initial coarse-grid conditions.
6. If **INHYD**=0, initialize arrays for the split-explicit scheme and perform split-explicit calculations at the initial time (SPINIT).
7. Begin forecast loop and read in coarse-grid boundary conditions (BDYIN).
8. Set boundary values for the water-substance variables (BDYVAL).
9. Set pointer addresses specific to a second domain if needed at the same time as the first domain (ADDRX1N).
10. If nests exist, interpolate and store nested-grid values from coarse-grid variables (STOTNDI) at t - Δt.
11. Compute the tendencies of the prognostic variables (call SOLVE1 for the hydrostatic version of the model, and SOLVE3 for the nonhydrostatic version).
12. If **INHYD**=0, perform the split-explicit time integration (SPLITF).
13. Compute amount of dry air and water substance advected through the lateral boundaries (CONADV).

14. Compute total amount of dry air and water substance in the domain.
15. Get the nested-domain tendencies (NSTLEV1).
16. Organize output for post-processor analyses (OUTPUT).
17. Shutdown forecast if out of CPU time or if domains are incompatible (SHUTDO).
18. Check whether a nest should be moved, activated, or initialized (CHKNST).
19. Continue forecast loop.

5.70 MOIENE

Subroutine MOIENE calculates the moist-static energy for the deep (Arakawa-Schubert) and shallow convection schemes.

MOIENE is called from subroutines ARASCH and SHALLOW.

Arguments: **T, Q, Z, h, KDIM, KS.**

Common blocks: None.

Data: None.

5.71 NOPRO

Subroutine NOPRO is an analysis-nudging FDDA routine that uses the information (defined in PARAM) on what times (**IDDATE**) and variables (**IDCHK**), from the surface-analysis nudging files, to withhold from processing and subsequent use for surface-analysis nudging within the PBL.

NOPRO is called from subroutine IN4DGD.

Arguments: **MDATES, IDDATE, IDCHK, MCHA, MVAR, idhk.**

Common blocks: None.

Data: None.

5.72 NSTLEV1

Subroutine NSTLEV1 is a driver subroutine for cycling through the subroutines that calculate the tendencies for all the nests at level one. The routine is similar to the main program since most of the subroutines called are the same as those called from the main program.

NSTLEV1 is called from the main program. It calls subroutines ADDRX1C, ADDRX1N, BDYOVL1, FEEDBK, NSTLEV2, SOLVE1, SOLVE3, SPINIT, SPLITF, STOTNDI, and STOTNDT.

Arguments: **ICOARS, IEXEC.**

Common blocks: ADDR0, ADDR3, ADDR4, ADDRN4, HUGE, NESLEV, NONHYDB, PARAM2, PARAM3, and PARFDDA.

Data: None.

5.73 NSTLEV2

Subroutine NSTLEV2 is a driver subroutine for cycling through the subroutines that calculate the tendencies for all the nests at level two. The routine is similar to the main program since most of the subroutines called are the same as those called from the main program.

NSTLEV2 is called from subroutine NSTLEV1. It calls subroutines ADDRX1C, ADDRX1N, BDYOVL1, FEEDBK, NSTLEV3, SOLVE1, SOLVE3, SPINIT, SPLITF, STOTNDI, and STOTNDT.

Arguments: **ICOARS, IEXEC.**

Common blocks: ADDR0, ADDR4, ADDRN4, HUGE, MIC, NESLEV, PARAM2, PARAM3, PARFDDA, and PBLDIM.

Data: None.

5.74 NSTLEV3

Subroutine NSTLEV3 is a driver subroutine for cycling through the subroutines that calculate the tendencies for all the nests at level three. The routine is similar to the main program since most of the subroutines called are the same as those called from the main program.

NSTLEV3 is called from subroutine NSTLEV2. It calls subroutines ADDRX1C, ADDRX1N, BDYOVL1, FEEDBK, SOLVE1, SOLVE3, SPINIT, SPLITF, STOTNDI, and STOTNDT.

Arguments: **ICOARS, IEXEC.**

Common blocks: ADDR0, ADDR4, ADDRN4, HUGE, MIC, NESLEV, PARAM2, PARAM3, PARFDDA, and PBLDIM.

Data: None.

5.75 NUDGD

Subroutine NUDGD is an analysis-nudging FDDA routine that computes the analysis-nudging term for a general model variable.

NUDGD is called from subroutines SETFD, SOLVE1 and SOLVE3. It calls subroutines BLNUDGD and MAPSMP.

Arguments: **MTIM, MVAR, INVAR, J, XB, XOBJK, xten*, pten*, PSTF, PSTO, GX, ID, WXY, WXY2, TFAC, ZFAC, GP, GR, MSD, IVAR, VORDIF, IN, KPBLT, BLWXY, blwnv*, SFCOBS, SFCTIM, QSATF, IQCHK, timb*, time*, ntb*, nte*, [SCR2D], [BLDUM2D], blpsoc*, blpsod*, REGJ, ZNTJ.**

Common blocks: ADDR4, HUGE, NESLEV, PARAM2, and PARFDDA.

Data: None.

---- Program detail ----

NUDGD computes the analysis-nudging term of the tendency equation (**XTEN**) for **XB**, where **XB** is a general model field at time t-1. The sequence of basic subroutine tasks is:

1. If **ID** = 1, compute the analysis-nudging term in the continuity (**PTEN**) equation (hydrostatic version of the model, and whether the p* fields from the surface-analysis or upper-air files are used for p* nudging depends on the **IPSTR** switch).
2. Define the vertical weighting function / partitioning factors (WPBL) between the surface-analysis nudging in the PBL and the analysis nudging of 3-D fields above the PBL. The partitioning is based on the current time step model PBL depth.
3. Compute surface-analysis nudging term of the tendency equation in the PBL (BLNUDGD).
   a) If **ID** = 0, compute the normal nudging term for U, V, T, or q_v.
   b) If **ID** = 3, compute the moisture nudging term using observed precipitation data.
4. Compute analysis-nudging term of the tendency equation above the PBL.
   a) If **ID** = 0, compute the normal nudging term for U, V, T, or q_v.
   b) If **ID** = 2, compute the rotational (vorticity) nudging term for U and V.
   c) If **ID** = 3, compute the moisture nudging term using observed precipitation data.

5.76 NUDGE

Subroutine NUDGE applies the relaxation boundary conditions to the local tendency term, **FTEN**. The final tendency is computed by a Newtonian term and a diffusion term.

NUDGE is called from subroutines SOLVE1 and SOLVE3.

Arguments: **IP, FCOEF, GCOEF, TBE, XT, ften*, FEB, FWB, FSB, FNB, FEBT, FWBT, FSBT, FNBT, FB, C203, KD, IE, JE, J.**

Common blocks: HUGE and NESLEV.

Data: None.

5.77 NUDOB

Subroutine NUDOB is an observational-nudging FDDA routine that computes the observations-nudging term for a general model variable.

NUDOB is called from subroutines SOLVE1 and SOLVE3. It calls subroutine MAPSMP

Arguments: **J, IVAR, aten*, pten*, IN, A, ISOUTH, JWEST, UB, VB, TB, QVB, PSB, PSBD, KPBLT, PTOP.**

Common blocks: ADDR4, ADDR8, HUGE, NESLEV, PARAM2, and PARFDDA.

Data: None.

---- Program detail ----

NUDOB computes the observation-nudging term of the tendency equation for a general model variable using a one-pass calculation of the weighting factors. The sequence of basic subroutine tasks is:

1. Compute the observation-nudging vertical weighting function **WPBL**, which is based on the current time step model PBL depth.
2. Compute the locations of the observations with respect to current mesh on either dot or cross points, depending on variable type.
3. Compute model p* at the observation locations for later use in the modified Cressman function used to spread observational information near the surface along constant sigma surfaces.
4. Begin outer loop for **NSTA** observations and check for proximity of each observation to grid point value.
5. Compute the temporal, horizontal and vertical weighting functions used to spread each surface observation along constant sigma near the surface, or to spread each observation above the surface along constant pressure (hydrostatic) or height (nonhydrostatic) levels.
6. Nudge model solution to surface-layer observations (default: winds only) on sigma levels.
    a) Loop through grid points within influence range of each surface-layer observation and compute the weighting sums over all observations.
7. Nudge model solution to observations above the model surface layer on pressure levels in **IWTSIG** = 0, or sigma levels if **IWTSIG** = 1:

a) Loop through grid points within influence range and compute the weighting sums over all observations.

8. Apply these sums of nudging weights to get the observation-nudging tendency **ATEN** (general model variable nudging tendency for **UTEN, VTEN TTEN, QVTEN** or **PTEN**).

NOTE: For efficiency, these sums of weights used for observation nudging do not need to be computed every model timestep(see description of parameter **IONF**).

5.78 OUTPRT

Subroutine OUTPRT is one of the two main printer output routines for MM5.

OUTPRT is called from subroutine OUTPUT. It calls subroutine MAPSMP.

Arguments: **IEXEC, N.**

Common blocks: ADDR1, ADDR2, ADDR3, ADDR4, HUGE, MIC, NESLEV, NONHYD, NONHYDB, PARAM2, PARAM3, PARFDDA, PBLDIM, and PMOIST.

Data:
IXN /2/ sampling interval in I-direction.
JXN /2/ sampling interval in J-direction.
KXN /1/ sampling interval in K-direction.
SVP11 /0.611/ constant used for calculation of the saturation vapor pressure (cb).
SVP12 /19.84659/ constant used for calculation of the saturation vapor pressure.
SVP13 /5418.12/ constant used for calculation of the saturation vapor pressure (K).

---- Program detail ----

OUTPRT uses MAPSMP to print output data at time intervals specified in PARAM. The sequence of the basic subroutine tasks is:

1. Print north-south cross sections of U, V, T, W, q_v, q_c, and q_r.
2. Print horizontal slices:
   a) Latitude, longitude, terrain, Coriolis parameter, and map-scale factor.
   b) Surface albedo, surface roughness, moisture availability, thermal inertia, and surface emissivity.
   c) p_s and p_s change.
   d) U, V, T, relative humidity, q_c, and q_r.
   e) PBL variables.
   f) Ground temperature, convective, non-convective, and total rainfall.

5.79 OUTPUT

Subroutine OUTPUT is the driver subroutine for output from MM5. It calls OUTTAP (write forecast data to output file), OUTSAV (write save file), and OUTPRT (primary printout routine).

OUTPUT is called from the main program. It calls subroutines ADDRX1C, FDAOFF, OUTPRT, OUTSAV, and OUTTAP.

Arguments: **IEXEC, NESCOUO.**

Common blocks: ADDR0, ADDR4, HUGE, MIC, NESLEV, PARAM2, PARFDDA, and PBLDIM.

Data: None.

5.80 OUTSAV

Subroutine OUTSAV writes the output data arrays **(ALLARR (IRHUGE),** and **INTALL (IIHUGE))** to a save file for each domain.

OUTSAV is called from subroutines OUTPUT and SHUTDO.

Arguments: **IUTSAV, ALLARR, IRHUGE, INTALL, IIHUGE.**

Common blocks: None.

Data: None.

5.81 OUTTAP

Subroutine OUTTAP writes output data to a standard output file.

OUTTAP is called from subroutines CHKNST, INIT, and OUTPUT. It calls subroutines DELTATIM and EQUATE.

Arguments: **IUTL, NESCOU, IX, JX, KZZ**.

Common blocks: ADDR0, ADDR1, ADDR2, ADDR4, ADDRNN, ADDRV, HEADER, HUGE, MIC, NESLEV, NHCNS, NHCNST, NONHYD, PARAM2, PARAM3, PARFDDA, PBLDIM, V9COM, and V9COMC.

Data: None.

---- Program detail ----

OUTTAP first writes a record header that includes coordinate information, nested information, and constants that define the nonhydrostatic reference state. Next, the 3-D forecast

arrays are written followed by the 2-D forecast arrays, and finally the 2-D terrestrial arrays. EQUATE is used throughout to ensure that arrays from nested domains having different grid size are properly written to the output file.

5.82 OVLCHK

Subroutine OVLCHK calculates which i, j boundary values come from another nest.

OVLCHK is called from subroutine INITNEST.

Arguments: **L.**

Common blocks: ADDRNN, HUGE, NESLEV, PARAM2, and PARFDDA.

Data: None.

5.83 PARAM

Subroutine PARAM defines the model parameters.

PARAM is called from the main program.

Arguments: None.

Common blocks: ADDR0, ADDR4, ADDR5, ADDR6, ADDR7, ADDR8, ARASCH, CFD, CFDDAGD, DEPAR2, HEADER, HUGE, NESLEV, PARAM2, PARAM3, PARFDDA, PMOIST, V9COM, and V9COMC.

Data: **MMD** (12) /31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31/ number of days in month.
QDCRIT /3.0E-7/ precipitation threshold for moisture convergence (cb kg K^{-1} s^{-1}) in the Kuo-Anthes scheme.
VQRANG /5.0E-4/ range limit on vertical eddy flux of water vapor (cb s^{-1}) in the Kuo-Anthes scheme.

[Summer Data Values]

ALBD (13) /18., 17., 19., 16., 12., 14., 8., 14., 25., 15., 55., 12., 20./ surface albedo (percent).
SLMO (13) /.05,.30, .15, .30, .30, .35, 1.0, .50, .02, .50, .95, .50, .15/ surface moisture availability as a fraction of one.
THERIN (13) /3., 4., 3., 4., 4., 5., 6., 6., 2., 5., 5., 5., 3./ thermal inertia (0.01 cal cm^{-2} K^{-1} $s^{-0.5}$).
SFEM (13) /.88, .92, .92, .93, .95, .95, .98, .95, .85, .92, .95, .95, .92/ surface emissivity at 9 micrometers.
SFZO (13) /50., 15., 12., 50., 50., 40., 0.01, 20., 10., 10., 5., 50., 15./ surface roughness length (cm).

SFHC (13) /18.5E5, 25.0E5, 20.8E5, 25.0E5, 29.2E5, 41.8E5, 99999., 29.2E5, 12.0E5, 99999., 99999., 29.2E5, 25.0E5/ surface heat capacity per unit volume (J m^{-3} K^{-1}).

[Winter Data Values]

ALBD (13) /18., 23., 23., 17., 12., 14., 8., 14., 25., 70., 70., 12., 20./ surface albedo (percent).

SLMO (13) /.10, .60, .30, .60, .60, .70, 1.0, .75, .05, .90, .95, .50, .15/ surface moisture availability as a fraction of one.

THERIN (13) /3., 4., 4., 5., 5., 6., 6., 6., 2., 5., 5., 5., 3./ thermal inertia (0.01 cal cm^{-2} K^{-1} $s^{-0.5}$).

SFEM (13) /.88, .92, .92, .93, .95, .95, .98, .95, .85, .92, .95, .95, .92/ surface emissivity at 9 micrometers.

SFZO (13) /50., 5., 10., 50., 50., 40., 0.01, 20., 10., 10., 5., 50., 15./ surface roughness length (cm).

SFHC (13) /18.5E5, 25.0E5, 20.8E5, 25.0E5, 29.2E5, 41.8E5, 99999., 29.2E5, 12.0E5, 99999., 99999., 29.2E5, 25.0E5/ surface heat capacity per unit volume (J m^{-3} K^{-1}).

SCFX (13) /.1, .25, .25, .2, .1, .22, 0., .15, .25, 0., 0., 0., 0./ scale factor for calculating the new albedo due to snow-cover effects.

---- Program detail ----

PARAM defines many of the model parameters regarding various options and contains extensive comments pertaining to model dimensions, FDDA options physical options, time and date, and output parameters. The sequence of the basic subroutine tasks is:

1. Set default values for namelist $OPARAM, $PPARAM, and $LPARAM options including nests.
2. Set FDDA default values for namelist $FPARAM options including nests.
3. Read in user-defined namelist values from $OPARAM, $LPARAM, $PPARAM, and $FPARAM
4. Define some FDDA parameters not included in the namelist.
5. Reset some of the coarse and nested domain options.
6. Specify the Julian date and compute the half-sigma levels.
7. Set the constants used in the model.
8. Set (or calculate) **DT** for the nonhydrostatic version and compute the time steps for radiation computations.
9. Compute the vertical interpolation coefficients for T and q_v, and determine the sigma levels for the lower bounds of low, middle, and high clouds.
10. Specify the coefficients for the sponge and nudging boundary conditions.

11. Read in transmissions if the ground temperature is to be calculated from the surface energy budget.
12. Printout parameter specifications for the model run.

5.84 PARAMR

Subroutine PARAMR sets constants for use with the EXMOISR routine.

PARAMR is called from subroutine PARAM.

Arguments: None.

Common blocks: JRG.

Data: None.

5.85 PRECIP

Subroutine PRECIP calculates the moisture properties of the cloud updraft for the deep (Arakawa-Schubert) and shallow convection schemes.

PRECIP is called from subroutines ARASCH and SHALLOW.

Arguments: **CD, KB, KBCON, KDIM, R, HC, HES, T, QE, QES, pw, qc, qrc, Z, LP, P, QKB, PCUT, C0, ZU, KNUM, KTOP.**

Common blocks: None.

Data: None.

5.86 PRINTSP

Subroutine PRINTSP prints out select values of the correction terms (for T, p*, U, and V) for the split-explicit scheme.

PRINTSP is called from subroutine SPSTEP2.

Arguments: **ARRAY, IX, JX, KX, IINC, JINC, KREF, NAME.**

Common blocks: None.

Data: None.

5.87 QSATGD

Subroutine QSATGD is an analysis-nudging FDDA routine that computes the model-simulated saturation mixing ratio for use in analysis nudging.

QSATGD is called from subroutines SOLVE1 and SOLVE3.

Arguments: **J, qsatf, TB, PSB, PPB, A, PTOP.**

Common blocks: ADDR4, HUGE, NESLEV, and PMOIST.

Data: None.

5.88 RDINIT

Subroutine RDINIT reads in the initial condition data.

RDINIT is called from subroutines INIT and INITNEST. It calls subroutines EQUATE and SHUTDO.

Arguments: **IUNITH, IUNITNH, IX, JX.**

Common blocks: ADDR0, ADDR1, ADDR2, ADDR4, ADDRNN, ASSEL, HEADER, HEADERC, HUGE, MIC, NESLEV, NHCNS, NHCNST, NONHYD, NONHYDB, PARAM2, PARAM3, PARFDDA, PBLDIM, SIZE.

Data: None.

5.89 SAVREAD

Subroutine SAVREAD reads a restart input file and puts the data into data arrays **(ALLARR(IRHUGE)** and **INTALL (IIHUGE)**) for a given domain.

SAVREAD is called from subroutine INITSAV.

Arguments: **ALLARR, IHUGE, INTALL, IIHUGE, IUTSAV.**

Common blocks: None.

Data: None.

5.90 SEAPRS

Subroutine SEAPRS computes sea-level pressure using the relationship: T1/T2 =(p1/p2)**(gamma*R/g), where gamma is the standard atmospheric lapse rate.

SEAPRS is called from subroutines FEEDBK, INITNEST, STOTNDI and STOTNDT.

Arguments: **T, TER, PS, SIG, IYY, JXX, KZZ, PTOP, slp, IDIM, JDIM, ts, t0.**

Common blocks: HUGE and NESLEV.

Data: None.

5.91 SETFD

Subroutine SETFD is a driver subroutine for calls to analysis-nudging and observation-nudging FDDA routines that operate on the entire grid, rather than one particular north-south slice, and can be performed prior to the major east-west J-loops in the SOLVE routines.

SETFD is called from subroutines SOLVE1 and SOLVE3. It calls subroutines BUFVDGD, ERROB, IN4DGD, IN4DOB, INTPSGD, NUDGD, and SETUPGD.

Arguments: **INEST, fdtim, [INFR], [IDARST].**

Common blocks: ADDR0, ADDR1, ADDR2, ADDR3, ADDR4, ADDR5, ADDR6, ADDR7, ADDR8, ASSEL, CFD, CFDDAGD, HUGE, MIC, NESLEV, PARAM2, PARAM3, PARFDDA, PBLDIM, and PMOIST.

Data: None.

---- Program detail ----

The sequence of the basic subroutine tasks is:

1. If necessary, read new analysis data for analysis nudging if **I4D** = 1, (IN4DGD).
2. Perform temporal and dot-point interpolation of p* (INTPSGD).
3. Set up temporal and spatial weighting functions for analysis nudging (SETUPGD).
4. Compute observed and model (forecast) vorticity difference if **IROT** = 1 (BUFVDGD).
5. Call analysis-nudging routines (NUDGD) for first north-south slice to perform necessary setup for subsequent north-south slices.
6. Read observation data for observation nudging if **I4DI** = 1, (IN4DGD).
7. Calculate difference between observation and model forecast at the observation locations within the specified time window (**TWINDO**) if **I4DI** = 1 (ERROB).

5.92 SETUPGD

Subroutine SETUPGD is an analysis-nudging FDDA routine that defines temporal, horizontal and vertical weighting functions used for analysis-nudging FDDA. It also computes model p* on dot points (**PSBD**) if nudging wind.

SETUPGD is called from subroutine SETFD. It calls subroutines DOTS, JULGMT, and UNITY.

Arguments: **PSB, psbdt, IN.**

Common blocks: ADDR4, ADDR5, ADDR6, ADDR7, CFD, CFDDAGD, HUGE, NESLEV, PARAM2, PARAM3, and PARFDDA.

Data: None.

---- Program detail ----

SETUPGD defines the time (**TFAC**), horizontal (**WDT, WCS, BLWDT, BLWCS**), and vertical (**ZFAC**) weighting functions for analysis-nudging. It also computes the horizontal weighting factor for moisture nudging, **WQ**, based on observed precipitation, if **IMOISP** = 1.

5.93 SFCRAD

Subroutine SFCRAD calculates the shortwave and longwave irradiances incident on the surface.

SFCRAD is called from subroutines BLKPBL and HIRPBL. It calls subroutine TRANSM.

Arguments: **IYY, JXX, J, INEST.**

Common blocks: ADDR1, ADDR2, ADDR4, HUGE, MIC, NESLEV, NONHYD, PARAM2, PARAM3, PARFDDA, PBLDIM, and PMOIST.

Data: **CLW** (3) /0.06, 0.22, 0.26/ coefficients used to compute cloud-cover fraction for low, middle, and high clouds.
CSW /4.183E+6/ heat capacity of water per unit volume (J m^{-3} K^{-1}).
LAMG (4) /1.407E-8, -1.455E-5, 6.29E-3, 0.16857/ constants used to compute soil thermal conductivity in moist soil.
TAC (3) /0.98, 0.85, 0.80/ absorption transmission coefficients for low, middle, and high clouds.
TSC (3) /0.80, 0.60, 0.48/ scattering transmission coefficients for low, middle, and high clouds.

---- Program detail ----

The sequence of the basic subroutine tasks is:

1. Determine the cloud fractions of low, middle, and high clouds and calculate the amount of precipitable water.
2. From (1) above, calculate the long-wave irradiance incident from the ground.
3. Calculate the short-wave irradiance from scattering transmissivities obtained from TRANSM.

4. Compute the slab thermal capacity based on soil heat capacity, soil conductivity, and moisture availability.

5.94 SHALLCU

Subroutine SHALLCU is a small driver subroutine for the shallow convection scheme (SHALLOW) and it sets up a few of the variables.

SHALLCU is called from subroutines SOLVE1 and SOLVE3. It calls subroutine SHALLOW.

Arguments: **tten*, qten*, J, NUMNES, SCR1, SCR2, ISTART, IEND.**

Common blocks: ADDR1, ADDR2, ADDR4, HUGE, NESLEV, PARAM2, PARAM3, PARFDDA, and PMOIST.

Data: None.

5.95 SHALLOW

Subroutine SHALLOW is the shallow cumulus parameterization routine. Although it is the main routine for the shallow convection scheme, it calls several other related subroutines.

SHALLOW is called from subroutine SHALLCU. It calls subroutines ARAOUTS, CLOUDWS, ENTRS, HEIPRE, KERHELS, MAXIM, MINIM, MOIENE, PRECIP, and ZUNC.

Arguments: **TI, QI, Z1, TIO, QIO, PIO, KLEV, PRE, PI, outtem, outq, DTIME, KBMAX, PCUT, C0, PSUR, ierrt, RADS, xmb.**

Common blocks: HUGE, NESLEV, PARAM2, PARAM3, PARFDDA, and PMOIST.

Data: None.

---- Program Detail ----

SHALLOW computes the moisture cycle using a shallow cumulus parameterization scheme. The sequence of the basic subroutine tasks is:

1. Initialize some of the arrays.
2. Compute the heights of the sigma levels (HEIPRE).
3. Calculate the moist static energy **HE** and the saturation moist static energy **HES** (MOIENE).
4. Determine the level with the highest moist static energy (MAXIM) and cloud base.
5. Begin in-cloud calculations.
   a) Calculate moist static energy and cloud top (ENTRS).
   b) Compute normalized mass-flux profile (ZUNC).

c) Calculate the moisture properties (non-precipitating) in the shallow updraft (PRECIP).

d) Calculate efficiency of generation of kinetic energy for the shallow updraft (CLOUDWS).

6. Begin new in-cloud and modified environmental calculations.

a) Calculate shallow cloud effect on environment (KERHELS).

b) Call HEIPRE and MOIENE for heights and moist static energy.

c) Call ENTRS, ZUNC, PRECIP, and CLOUDWS for the moist static energy (in cloud), normalized mass-flux profile, amount of moisture in the shallow updraft, and updraft cloud-work function.

7. Calculate feedback effects of clouds to the environment (ARAOUTS).

5.96 SHUTDO

Subroutine SHUTDO shuts down a forecast if running out of CPU time or domains are not assigned correctly. A save file is written (calls OUTSAV).

SHUTDO is called from the main program and subroutine RDINIT. It calls subroutines ADDRX1C and OUTSAV.

Arguments: None.

Common blocks: ADDR0, ADDR4, HUGE, MIC, NESLEV, PARAM2, PARFDDA, and PBLDIM.

Data: None.

5.97 SINT

Subroutine SINT interpolates nested-grid values from coarse-grid values using a positive-definite advection scheme.

SINT is called from subroutine EXAINT.

Arguments: **xf*, N, M, N1STAR, N1END, N2STAR, N2END.**

Common blocks: DEPAR2 and HUGE.

Data: **EP** /1.0E-10/ machine precision value.

5.98 SINTX

Subroutine SINTX interpolates nested-grid boundary values from coarse-grid values (in the x-direction) using a positive-definite advection scheme.

SINTX is called from subroutines STOTNDI and STOTNDT.

Arguments: **xf*, N, M, N1STAR, N1END, JXWANT, ICOMPS, ICRSDOT.**

Common blocks: DEPAR2 and HUGE.

Data: **EP** /1.0E-10/ machine precision value.

5.99 SINTY

Subroutine SINTY interpolates nested-grid boundary values from coarse-grid values (in the y-direction) using a positive-definite advection scheme.

SINTY is called from subroutines STOTNDI and STOTNDT.

Arguments: **xf*, N, M, IYWANT, N2STAR, N2END, ICOMPS, ICRSDOT.**

Common blocks: DEPAR2 and HUGE.

Data: **EP** /1.0E-10/ machine precision value.

5.100 SLAB

Subroutine SLAB calculates the ground temperature tendency from the residual of the surface energy budget. For points over water, the ground temperature tendency is zero; if snow cover effects are considered, the ground temperature is limited to less than 273.16 K for grid points having snow cover.

SLAB is called from subroutines BLKPBL and HIRPBL.

Arguments: **DELTSM, J, INEST.**

Common blocks: ADDR1, ADDR2, ADDR4, HUGE, MIC, NESLEV, NONHYD, PARAM2, PARAM3, PARFDDA, PBLDIM, and PMOIST.

Data: None.

5.101 SMTHER

Subroutine SMTHER is a 2-pass smoother-desmoother routine designed to dampen the short wavelength components.

SMTHER is called from subroutines BLW and FEEDBK.

Arguments: **slab*, IS1, IS2, NPASS, IST, IE, JST, JE.**

Common blocks: None.

Data: None.

5.102 SOLAR1

Subroutine SOLAR1 computes the solar declination angle from the Julian date.

SOLAR1 is called from subroutines INIT, INITSAV, SOLVE1 and SOLVE3.

Arguments: **XTIME, IEXEC.**

Common blocks: HUGE, NESLEV, and PARAM3.

Data: None.

5.103 SOLVE1 [multi-tasked]

Subroutine SOLVE1 computes the tendencies of the prognostic variables for the hydrostatic version (**INHYD** = 0) of MM5.

SOLVE1 is called from the main program and from subroutines NSTLEV1, NSTLEV2, and NSTLEV3. It calls subroutines BLKPBL, BUFSLGD, CUPARA1, CUPARA2, CUPARA3, DECPU, DIFFU, DIFFUT, EXMOISR, EXMOISS, HADV, HIRPBL, MDOTS, NUDGD, NUDGE, NUDOB, QSATGD, SETFD, SHALLCU, SOLAR1, SPONGE, and VADV.

Arguments: **iexec*, INEST, and NN.**

Common blocks: ADDR0, ADDR1, ADDR2, ADDR3, ADDR4, ADDR5, ADDR6, ADDR7, ADDR8, ASSEL, CFD, CFDDAGD, HUGE, MIC, NESLEV, PARAM2, PARAM3, PARFDDA, PBLDIM, and PMOIST.

Data: **DTMITE** /25./ time constant (s) used to compute PBL iteration loop **MITERO**.

---- Program Detail ----

SOLVE1 and SOLVE3 are the main forecast subroutines of MM5. SOLVE1 computes the tendencies of the prognostic variables p*, U, V, q_v and other moisture variables, and T. For each time step, SOLVE1 is used one time for the large domain and **IRAX** (the ratio between the coarse and fine grids) times for a nested domain. In a case where a nested domain is nested within a bigger nest, the biggest nest is considered the large domain. The sequence of the basic subroutine tasks is:

1. Decouple the boundary values of the **UA** and **VA** arrays.
2. Multiply **UA** and **VA** by the inverse of the mapscale factor and initialize some arrays to zero for time t+1.
3. Calculate the boundary values of U, V, q_v, p*, T, q_r, q_c, ice (q_i), and snow (q_{ni}) for time t+1, and compute QDOT at the boundaries.
4. If employing FDDA (**IFDDA** = 1), call SETFD for initial setup.
5. Compute the pressure tendency and vertical sigma velocity (**QDOT**).
6. Apply sponge or relaxation boundary conditions to the pressure tendency (SPONGE or NUDGE).
7. If **IFDDA** = 1, compute the nudging term for p* (NUDOB, NUDGD).
8. Get the forecast pressure at time t+1 (add the tendency) and compute the Bleck noise parameters.
9. Compute geopotential height at the half-sigma levels.
10. Decouple U and V (DECPU), and compute the horizontal diffusion coefficients. Calculate horizontal diffusion and horizontal advection of momentum.
11. Decouple temperature and moisture.
12. Get the temperature tendency:
    a) Compute horizontal and vertical advection (HADV and VADV).
    b) Compute the adiabatic and radiational (LWRAD and SWRAD) cooling terms.
13. Get contributions to the moisture tendency from cumulus parameterizations and advection of water vapor.
    a) If ICUPA = 2, use Anthes-Kuo scheme (CUPARA1).
    b) If ICUPA = 3, use Grell scheme (CUPARA2).
    c) If ICUPA = 4, use Arakawa Schubert scheme (CUPARA3).
14. Compute some horizontal and vertical flux divergence terms needed for the explicit moisture scheme (HADV and VADV).
15. Calculate diffusion (DIFFUT) for moisture and temperature.
16. Compute PBL fluxes (BLKPBL if **IBLTYP** = 1, or HIRPBL if **IBLTYP** = 2).
17. Get the contribution to the moisture tendency from shallow cumulus (call SHALLCU if **ISHALLO** = 1).
18. Add horizontal diffusion and PBL tendencies for T and q_v to **TTEN** and **QVTEN**.
19. Apply sponge or relaxation boundary conditions to T and q_v.
20. If **IFDDA** = 1, get FDDA nudging terms for temperature and moisture from observations (NUDOB) and / or from an analysis (NUDGD).
21. Call EXMOISR or EXMOISS for contributions to the moisture tendency from explicit moisture schemes.

22. Get the forecast T, q_v, q_c, q_r, and q_i at time t+1 (add the tendencies).
23. Get the U and V tendencies.
    a) If **IFDDA** = 1, get FDDA nudging terms for U and V wind components from observations (NUDOB) and / or from an analysis (NUDGD).
    b) Compute the pressure gradient, Coriolis, and geopotential gradient terms.
    c) Compute the vertical flux divergence terms (**VADV**), and add the diffusion and PBL tendencies for U and V to **UTEN** and **VTEN**.
    d) Apply sponge or relaxation boundary conditions to U and V.
    e) Get the forecast U and V at time t+1 (add the tendencies).
24. Perform time-smoothing operations. Update interior of arrays using Asselin filter.
25. Compute the non-convective precipitation and increment the forecast time.
26. Printout noise parameters and number of convective points.
27. Recalculate solar declination angle if forecast time is greater than 24 hours (SOLAR1).

----User Note----

Set **INHYD** = 0 for this option

5.104 SOLVE3 [multi-tasked]

Subroutine SOLVE3 computes the tendencies of the prognostic variables for the nonhydrostatic version (**INHYD** = 1) of MM5.

SOLVE3 is called from the main program and from subroutines NSTLEV1, NSTLEV2, and NSTLEV3. It calls subroutines BLKPBL, BUFSLGD, CUPARA1, CUPARA2, CUPARA3, DECPU, DIFFU, DIFFUT, DOTS, EXMOISR, EXMOISS, HADV, HIRPBL, NUDGD, NUDGE, NUDOB, QSATGD, SETFD, SHALLCU, SOLAR1, SOUND, and VADV.

Arguments: **iexec*, INEST, and NN.**

Common blocks: ADDR0, ADDR1, ADDR2, ADDR3, ADDR4, ADDR5, ADDR6, ADDR7, ADDR8, ASSEL, CFD, CFDDAGD, HUGE, MIC, NESLEV, NHCNS, NHCNST, NONHYD, NONHYDB, PARAM2, PARAM3, PARFDDA, PBLDIM, and PMOIST.

Data: **DTMITE** /25./ time constant (s) used to compute PBL iteration loop **MITERO**.

---- Program Detail ----

SOLVE3 and SOLVE1 are the main forecast subroutines of MM5. SOLVE3 computes the tendencies of the prognostic variables p', U, V, W, q_v, T and other moisture variables. For each

time step, SOLVE3 is used one time for the large domain and **IRAX** (the ratio between the coarse and fine grids) times for a nested domain. In a case where a nested domain is nested within a bigger nest, the biggest nest is considered the large domain. The sequence of the basic subroutine tasks is:

1. Decouple the boundary values of the **UA** and **VA** arrays.
2. Multiply **UA** and **VA** by the inverse of the mapscale factor and initialize some arrays to zero for time t+1.
3. Calculate the boundary values of U, V, q_v, p*, T, q_r, q_c, ice (q_i), and snow (q_{ni}) for time t+1.
4. If employing FDDA (**IFDDA** = 1), call SETFD for initial setup.
5. Calculate **QDOT** and the total divergence **DIVX**.
6. Decouple U and V (DECPU), and compute the horizontal diffusion coefficients. Calculate horizontal diffusion and horizontal advection of momentum.
7. Decouple temperature, p', and moisture.
8. Get the temperature, p', and W tendencies:
   a) Compute horizontal and vertical advection (HADV and VADV).
   b) Compute the adiabatic and radiational (LWRAD and SWRAD) cooling terms for T and buoyancy term for W.
   c) Remove the total divergence term in temperature, moisture, W, and p'.
9. Get contributions to the moisture tendency from cumulus parameterizations and advection of water vapor:
   a) If **ICUPA** = 2, use Anthes-Kuo scheme (CUPARA1).
   b) If **ICUPA** = 3, use Grell scheme (CUPARA2).
   c) If **ICUPA** = 4, use Arakawa-Schubert scheme (CUPARA3).
10. Compute some horizontal and vertical flux divergence terms needed for the explicit moisture scheme (HADV and VADV).
11. Calculate diffusion (DIFFUT) for moisture, T, and W.
12. Compute PBL fluxes (BLKPBL if **IBLTYP** = 1, or HIRPBL if **IBLTYP** = 2).
13. Get the contribution to the moisture tendency from shallow cumulus (call SHALLCU if **ISHALLO** = 1).
14. Add horizontal diffusion and PBL tendencies for T and q_v to **TTEN** and **QVTEN**.
15. Apply relaxation boundary conditions to T and q_v.
16. If **IFDDA** = 1, get FDDA nudging terms for temperature and moisture from observations (NUDOB) and / or from an analysis (NUDGD).
17. Set **PPTENS** and **WTENS** for SOUND.

18. Call EXMOISR or EXMOISS for contributions to the moisture tendency from explicit moisture schemes.
19. Get the forecast T, q_v, q_c, q_r, q_{ni}, and q_i at time t+1 (add the tendencies).
20. If **IFDDA** = 1, get FDDA nudging terms for U and V wind components from observations (NUDOB) and / or from an analysis (NUDGD).
21. Compute the Coriolis terms.
22. Compute the vertical flux divergence terms (VADV), and add divergence term, diffusion and PBL tendencies for U and V to **UTEN** and **VTEN**.
23. Apply relaxation boundary conditions to U and V.
24. Set **UTENS** and **VTENS** for SOUND.
25. Perform time-smoothing operations on temperature and moisture. Update interior of arrays using asselin filter.
26. Call SOUND which considers the effects of soundwaves for short time steps and includes the interaction of p and momentum.
27. Compute the non-convective precipitation and increment the forecast time.
28. Recalculate solar declination angle if forecast time is greater than 24 hours (SOLAR1).

----User Note----

Set **INHYD** = 1 for this option.

5.105 SOUND [multi-tasked]

Subroutine SOUND considers the effects of soundwaves for short time steps and includes the interaction of pressure and momentum.

SOUND is called from subroutine SOLVE3.

Arguments: **IYY, JXX, ub*, vb*, tb*, PSADOT, ua*, va*, ta*, QVB, PSA, HT, MSFD, MSFX, DX, DTL, INEST, KTAU.**

Common blocks: ASSEL, HUGE, NESLEV, NHCNS, NHCNST, NONHYD, NONHYDB, PARAM2, PARAM3, PARFDDA, PMOIST and UPRAD.

Data: None.

---- Program detail ----

SOUND, which is needed only for the nonhydrostatic version of MM5, considers the effects on soundwaves for short time steps. The sequence of the basic subroutine tasks is:

1. Set some constants and calculate initial arrays for the short time step.
2. Begin short time steps.

3. Calculate the divergence damping term (added temporarily) and the U, V pressure gradients.
4. Establish the U and V boundary conditions and subtract the divergence damping term.
5. Begin the main **I, J** (do 400) loop.
6. Begin the vertical momentum / pressure calculation, column by column.
7. Calculate implicit W equation scheme coefficients **AA**, **B**, and **C** (Ikawa method.)
8. Get RHS for implicit scheme and calculate vertical momentum.
9. Calculate pressure and add the pressure correction (dp' / dt) to temperature.
10. End main **I**, **J** loop.
11. Establish W and p' boundary conditions.
12. End short time steps.
13. Transfer **XXA** values to **XXB** and apply time filter.

5.106 SOUNDD

Subroutine SOUNDD determines the downdraft thermodynamic properties for the Arakawa-Schubert convection scheme and calculates the normalized mass flux.

SOUNDD is called from subroutine ARASCH.

Arguments: **HE, hcd, Z, KDIM, KMIN, PW, LP, HES, QES, QE, T, itest*, qrcd*, pwd*, zd, edt, KDET, KNUM.**

Common blocks: None.

Data: None.

5.107 SPCORT [multi-tasked]

Subroutine SPCORT adds the correction term **DDSUM** to temperature and p* for the split-explicit scheme.

SPCORT is called from subroutine SPLITF.

Arguments: **ta*, psa*, tb*, psb*, DDSUM, AM, AN, GNUHF, KL, ILXM, JLXM, INEST.**

Common blocks: HUGE and NESLEV.

Data: None.

5.108 SPCORU [multi-tasked]

Subroutine SPCORU adds the correction term **DHSUM** to U and V for the split- explicit scheme.

SPCORU is called from subroutine SPLITF.

Arguments: **ua*, va*, PSDOT, ub*, vb*, MSFD, DHSUM, ZMATX, GNUHF, DX2, KL, ILX, JLX, INEST.**

Common blocks: HUGE and NESLEV.

Data: None.

5.109 SPDIFF [multi-tasked]

Subroutine SPDIFF is a small routine used to compute the difference between arrays **IN1** and **IN2** for the split-explicit scheme.

SPDIFF is called from subroutine SPLITF.

Arguments: **out, IN2, IN1, IY, JX, KZ, NT2, NT1, NT.**

Common blocks: HUGE and NESLEV.

Data: None.

5.110 SPDIVG [multi-tasked]

Subroutine SPDIVG calculates the divergence from U and V in vertical mode space for the split-explicit scheme.

SPDIVG is called from subroutine SPINIT and SPLITF.

Arguments: **u*, v*, deld*, ZMATXR, MSFD, MSFX, DX2, IX, JX, KL, IMIN, JMIN, IMAX, JMAX.**

Common blocks: HUGE and NESLEV.

Data: None.

5.111 SPGEOP [multi-tasked]

Subroutine SPGEOP calculates the geopotential in vertical mode space for split-explicit scheme.

SPGEOP is called from subroutine SPINIT and SPLITF.

Arguments: **T, PS, delh, BZ, CZ, SIGMAH, PD, PTOP, KL, ILX, JLX.**

Common blocks: HUGE and NESLEV.

Data: None.

5.112 SPGRAD [multi-tasked]

Subroutine SPGRAD computes the gradient of geopotential for the split-explicit scheme. SPGRAD is called from subroutine SPSTEP2.

Arguments: **PHI, gx, gy, MSFX, DX2, KL, ILX, JLX.**

Common blocks: HUGE and NESLEV.

Data: None.

5.113 SPINIT

Subroutine SPINIT is an initialization routine for the split-explicit scheme. The purpose of the split-explicit scheme is to minimize the computation cost while, at the same time, maintaining model numerical stability. The momentum, pressure and temperature tendencies are computed on 2 time scales with the fastest gravity modes handled with a short time step. The technique still allows the bulk of the model tendency calculations to be performed using a longer time step, and thus no significant increase in the number of required computations.

SPINIT is called from the main program and from subroutines CHKNST, INITSAV, NSTLEV1, NSTLEV2, and NSTLEV3. It calls subroutines SPDIVG, SPGEOP, and VMODES.

Arguments: **PTOP, SIGMA, KV1, IX, JX.**

Common blocks: ADDR1, ADDR2, ADDR4, ADDRSP, blank, HUGE, and NESLEV.

Data: None.

---- Program Detail ----

SPINIT initializes a number of arrays for the split-explicit scheme and performs calculations at the initial time. The sequence of the basic subroutine tasks is:

1. Calculate a mean surface pressure and mean vertical temperature profile (**PS** and **TBARH**) for use in VMODES.
2. Determine the vertical modes of the model (VMODES).
3. Compute the **BZ** and **CZ** arrays needed for transformation of temperature from sigma space to vertical mode space for calculation of geopotential in vertical mode space.

4. Calculate the divergence in vertical mode space (SPDIVG).
5. Calculate the geopotential in vertical mode space (SPGEOP).

5.114 SPLITF [multi-tasked]

Subroutine SPLITF is a driver subroutine for split-explicit time integration.

SPLITF is called from the main program and from subroutines NSTLEV1, NSTLEV2, and NSTLEV3. It calls subroutines SPCORT, SPCORU, SPDIFF, SPDIVG, SPGEOP, SPSTEP2, and XTDOT.

Arguments: **GNUHF, INEST.**

Common blocks: ADDR1, ADDR2, ADDR4, ADDRSP, blank, HUGE, MIC, NESLEV, and PBLDIM.

Data: None.

---- Program Detail ----

SPLITF calls routines that compute the time derivative of divergence and geopotential and add the correction terms to T, p*, U, and V. The sequence of the basic subroutine tasks is:

1. Interpolate p from cross point to dot point (XTDOT).
2. Call SPDIVG twice to calculate the divergence **DELD** at time t (use **UA** and **VA**) and at time t-1 (use **UB** and **VB**).
3. Call SPDIFF to get the **DELD** difference between time t and t-1.
4. Call SPGEOP twice to calculate the geopotential **DELH** at time t (use **TA** and **PSA**) and at time t-1 (use **TB** and **PSB**).
5. Call SPDIFF to get the **DELH** difference between time t and t-1.
6. Perform time integration for the small time steps for fast gravity wave modes (SPSTEP2) and return the correction terms **DDSUM** and **DHSUM** that have been summed over the small time steps.
7. Add the correction **DDSUM** to temperature and p* (SPCORT).
8. Add the correction **DHSUM** to U and V (SPCORU).

5.115 SPONGE

Subroutine SPONGE applies the sponge boundary conditions to the local tendency term, **FTEN**. The final tendency is computed using a weighted average of large-scale (or observed) and model-predicted tendencies.

SPONGE is called from subroutine SOLVE1.

Arguments: **IP, WG, ften*, FEBT, FWBT, FNBT, FSBT, KD, IE, JE, J.**

Common blocks: HUGE and NESLEV.

Data: None.

5.116 SPSTEP2 [multi-tasked]

Subroutine SPSTEP2 is a time integration routine for the split-explicit scheme.

SPSTEP2 is called from subroutine SPLITF. It calls subroutines DIVG, PRINTSP, and SPGRAD.

Arguments: **ddsum, dhsum, [DELD], [DELH], HBAR, PSA, PSDOT, MSFD, MSFX, [WORK], DX2, DTAU, M, IX, JX, ILX, JLX, ILXM, JLXM, INEST.**

Common blocks: HUGE and NESLEV.

Data: None.

---- Program detail ----

SPSTEP2 performs time integration for the small time steps for fast gravity wave modes in the split explicit scheme. It returns the correction terms for temperature and p* (**DDSUM**) and U and V (**DHSUM**) that have been summed over the small time steps. It calls SPGRAD to compute the gradient of geopotential needed for **DHSUM** calculations, and it calls DIVG to compute the divergence needed for the **DDSUM** calculations.

5.117 STOTNDI [multi-tasked]

Subroutine STOTNDI interpolates and stores nested-grid values from coarse-grid variables at the nested-grid boundaries for time t-1. These values are needed later in STOTNDT.

STOTNDI is called from the main program and from subroutines NSTLEV1, NSTLEV2, and NSTLEV3. It calls subroutines EXAINT, MDOTS, SEAPRS, SINTX, and SINTY.

Arguments: **INEST, IYYN, JXXN, IYY, JXX.**

Common blocks: ADDR0, ADDR1, ADDR2, ADDR4, ADDRN1, ADDRN2, ADDRN3, ADDRN4, ADDRNN, DEPAR2, HUGE, MIC, NESLEV, NNNHYD, NNNHYDB, NONHYD, PARAM2, PARAM3, PARFDDA, PBLDIM, and PMOIST.

Data: None.

---- Program detail ----

STOTNDI interpolates values of p*, T, q_v, q_c, q_r, q_i, q_{ni}, U, V, and (if using the nonhydrostatic version) vertical velocity and the pressure perturbation from the coarse grid variables at the nest interface. The interpolation of the variables at time t-1 is accomplished using a positive-definite advection scheme (SINTX for the x-direction and SINTY for the y-direction).

5.118 STOTNDT [multi-tasked]

Subroutine STOTNDT interpolates nested-grid values from coarse-grid variables at the nested-grid boundaries for time t+1. It then also uses the interpolated values for time t-1 (obtained from STOTNDI) to calculate the nested-grid boundary tendencies.

STOTNDT is called from subroutines NSTLEV1, NSTLEV2, and NSTLEV3. It calls subroutines EXAINT, MDOTS, SEAPRS, SINTX, and SINTY.

Arguments: **INEST, IYYN, JXXN, IYY, JXX.**

Common blocks: ADDR0, ADDR1, ADDR2, ADDR4, ADDRN1, ADDRN2, ADDRN3, ADDRN4, ADDRNN, DEPAR2, HUGE, MIC, NESLEV, NNNHYD, NNNHYDB, NONHYD, PARAM2, PARAM3, PARFDDA, PBLDIM, and PMOIST.

Data: None.

---- Program detail ----

STOTNDT interpolates values of p*, T, q_v, q_c, q_r, q_i, q_{ni}, U, V, and (if using the nonhydrostatic version) vertical velocity and the pressure perturbation from the coarse grid variables at the nested interface. The interpolation of the variables at time t+1 is accomplished using a positive-definite advection scheme (SINTX for the x-direction and SINTY for the y-direction. It then uses these values along with the values for time t-1 to calculate the nested-grid tendencies at the interface.

5.119 SUBCH

Subroutine SUBCH adjusts the corner points of a nest within a moving nest.

SUBCH is called from subroutine INITNEST.

Arguments: **NUMNE, NUMN, NESTII, NESTJJ, ISOUTHO, JWESTO.**

Common blocks: ADDRN4, ADDRNN, HUGE, and NESLEV.

Data: None.

5.120 SWRAD

Subroutine SWRAD calculates the shortwave component of the atmospheric radiative temperature tendencies and surface flux.

SWRAD is called from subroutines SOLVE1 and SOLVE3.

Arguments: **J**

Common blocks: ADDR1, ADDR2, ADDR4, HUGE, NESLEV, NHCNS, NHCNST, NONHYD, PARAM3, and RADIAT.

Data: **ALBTAB** (4, 5) look-up table of cloud albedo as a function of liquid water path and zenith angle.
ABSTAB (4, 5) look-up table of cloud absorption as a function of liquid water path and zenith angle.
XMUVAL (4) cosine (zenith angle) for look-up tables.

----User Note----

Set **IRDDIM** = 1 for this option.

---- Program Detail ----

1. Preset column variables and calculates solar zenith angle.
2. Calculate various path lengths for each model level.
3. Calculate downward flux for each model level by integration from top down. Flux depleted by absorption and scattering in clear air and clouds.
4. Absorption in each layer determines heating rate.
5. Save downward shortwave flux at surface (**GSW**) for ground heat budget.

5.121 TMASS

Subroutine TMASS computes the total mass (kg) of dry air and water substance within the domain.

TMASS is called from subroutines CONMAS, INIT, and INITNEST.

Arguments: **tdrym, tqmass, tvmass, tcmass, trmass, IL, JL, KL, IDRY, IMOIST, PSA, QVA, QCA, QRA, DSIGMA, DX, G.**

Common blocks: HUGE and NESLEV**.**

Data: None.

5.122 TRANSM

Subroutine TRANSM uses a lookup table to compute the shortwave transmissivity. The values (which are a function of path length and precipitable water at standard atmospheric pressure) are interpolated and then adjusted for actual pressure.

TRANSM is called from subroutine SFCRAD.

Arguments: **[PATH], PRW, ftabs, ftscat, fbscat, [PSB], J, IYY.**

Common blocks: HUGE, MESLEV, NONHYD, and PARAM3.

Data: None.

5.123 UNITY

Subroutine UNITY sets the values of a 2-D array **SLAB1(MIX, MJX)** TO 1.

UNITY is called from subroutine SETUPGD.

Arguments: **slab1, IMAX, JMAX.**

Common blocks: HUGE and NESLEV.

Data: None.

5.124 VADV

Subroutine VADV computes the vertical advection (flux-divergence) terms for **QVA**, **QCA, QRA, UA, VA,** and nonhydrostatic W and p'. The interpolation is based on linear potential temperature variation for temperature and logarithmic variation for q_v.

VADV is called from subroutines SOLVE1 and SOLVE3.

Arguments: **KZZ, ften*, FA, QDOT, MSF, PSA, [F], J, IND, IN.**

Common blocks: ADDR4, HUGE, NESLEV, and PARAM3.

Data: None.

5.125 VCHEKE

Subroutine VCHEKE checks that eigenvalues are real and positive valued for computations of **HBAR** for the split-explicit scheme.

VCHEKE is called from subroutine VMODES.

Arguments: **ER, EI, NK, numerr*, ANAME.**

Common blocks: None.

Data: None.

5.126 VCHEKI

Subroutine VCHEKI checks for linear algebra errors in the computations of matrix **ZMATX** for the split-explicit scheme.

VCHEKI is called from subroutine VMODES.

Arguments: **IER, numerr*, ANAME.**

Common blocks: None.

Data: None.

5.127 VCHEKT

Subroutine VCHEKT checks if the mean vertical temperature profile **TBARH** is stable for the split-explicit scheme.

VCHEKT is called from subroutine VMODES.

Arguments: **TBARH, SIGMAH, SIGMAF, XKAPPA, PT, PD, NK, numerr*.**

Common blocks: None.

Data: None.

5.128 VMODES

Subroutine VMODES determines the vertical modes of the model for the split-explicit scheme.

VMODES is called from subroutine SPINIT. It calls subroutines INVMTX, VCHEKE, VCKEKI, VCHEKT, VMULTM, VNORML, VORDER, VPRNTM, VPRNTV, and VTLAPS.

Arguments: **LSTAND, SIGMAF, KV1**

Common blocks: ADDR1, ADDR4, ADDRSP, blank, HUGE, and NESLEV.

Data: **LPRINT /.FALSE./ true** denotes linearization about standard atmosphere, **false** denotes linearization about horizontal mean of data.

---- Program Detail ----

VMODES determines the equivalent depths of the vertical modes of the model and also calculates matrices needed for transformation from sigma space to vertical mode space. The sequence of the basic subroutine tasks is:

1. Set up the arrays describing the vertical structure and compute a standard atmospheric temperature environment (VTLAPS) if **LSTAND** = **TRUE**.
2. Check if the vertical temperature structure **TBARH** is stable (VCHEKT).
3. Define diagonal elements for determination of the thermodynamic matrix.
4. Compute the transform from divergence to sigma dot (VMULTM).
5. Define matrices for linearized determination of geopotential--compute the hydrostatic matrices **HYDROS** and **HYDROC**.
6. Get equivalent depths of vertical modes (**HBAR**) and determine the matrix **ZMATX** which is used for transforming divergence from sigma to vertical mode space.
7. Call VCHEKI and VCHEKE to check for linear algebra errors in the **ZMATX** computations and for eigenvalue errors in the **HBAR** computations.
8. Adjust the components of **HBAR** so that they are ordered from largest value to smallest value (VORDER).
9. Normalize the columns of **ZMATX** such that the component with the largest absolute value is positive and the sum of the mass-weighted squares is one (VNORML).
10. Call INVTMX to compute the inverse of **ZMATX** (**ZMATXR**) and **HYDROS** (**HYDROR**).
11. If **LPRINT** is **TRUE**, print out some VMODES arrays (**VPRNTV** and **VPRNTM**).

5.129 VMULTM

Subroutine VMULTM calls four entries which perform matrix multiplication, addition, and subtraction operations for the split-explicit scheme.

VMULTM is called from subroutine VMODES.

Arguments: **a1*, A2, A3, NK.**

Common blocks: None.

Data: None.

5.130 VNORML

Subroutine VNORML normalizes the columns of the matrix **ZMATX** for the split- explicit scheme such that the component with the largest absolute value is positive and the sum of the mass-weighted squares is one.

VNORML is called from subroutine VMODES.

Arguments: **z*, S, NK, NK1.**

Common blocks: None.

Data: None.

5.131 VORDER

Subroutine VORDER adjusts the components of HBAR (equivalent depths of vertical modes) for the split-explicit scheme so that they are ordered from largest value to smallest value.

VORDER is called from subroutine VMODES.

Arguments: **z*, hbar*, [WZ], [WH], NK.**

Common blocks: None.

Data: None.

5.132 VPRNTM

Subroutine VPRNTM prints some 2-D VMODES arrays for the split-explicit scheme.

VPRNTM is called from subroutine VMODES.

Arguments: **A, N1, N2, NAM.**

Common blocks: None.

Data: None.

5.133 VPRNTV

Subroutine VPRNTV prints some 1-D VMODES arrays for the split-explicit scheme.

VPRNTV is called from subroutine VMODES.

Arguments: **A, N, NAM.**

Common blocks: None.

Data: None.

5.134 VTLAPS

Subroutine VTLAPS computes a standard atmosphere temperature environment for the split-explicit scheme (if **LSTAND** = **TRUE**).

VTLAPS is called from subroutine VMODES.

Arguments: **t, SIGMA, R, PT, PD, NK.**

Common blocks: None.

Data: None.

5.135 VTRAN

Subroutine VTRAN transposes the x<-->y orientation of a 2-D array.

VTRAN is called from subroutine MAPSMP.

Arguments: **fyx*, IMAX, JMAX.**

Common blocks: None.

Data: None.

5.136 XTDOT

Subroutine XTDOT interpolates P from cross point to dot point for the split-explicit scheme.

XTDOT is called from subroutine SPLITF.

Arguments: **PX, pd, NI, NJ, NI1, NJ1.**

Common blocks: HUGE and NESLEV.

Data: None.

5.137 ZUNC

Subroutine ZUNC computes the normalized mass-flux profiles for the deep (Arakawa-Schubert) and shallow convection schemes.

ZUNC is called from subroutines ARASCH and SHALLOW.

Arguments: **KBEG, zu, KB, R, Z, KDIM, LP, KNUM, KTOP**

Common blocks: None.

Data: None.

5.138 ZX4LP

Subroutine ZX4LP is used for the Arakawa-Schubert convection scheme to solve a linear programming problem.

ZX4LP is called from subroutine ARAMB.

Arguments: **A, IA, B, C, N, M1, M2, S, psol, DSOL, RW, IW, ier.**

Common blocks: WORKSP.

Data: None.

Section 6

SUBROUTINE ARGUMENTS

This section lists (alphabetically) and defines all the subroutine arguments. The dimension and units are given, and argument variables or constants that also appear in common blocks are appended with a (/). Arguments listed more than once indicate that the variable (or constant) has different definition among different subroutines. For these situations, the description includes the name of the subroutine that the argument refers to (unless several subroutines share one of the multiple argument definitions).

A/ (**MKX**); half-sigma levels where **A**(**K**)=0.5 (**SIGMA** (**K**)+**SIGMA**(**K**+1)).

A/ (**KX**); local input array in VPRNTV and VPRNTM (**A** (**KX**, **KX**)) that will be printed.

A/ (**KX**, **KX**); local input array in INVMTX that holds a matrix that will be inverted.

A an array used in ZX4LP.

A1/ (**KX**, **KX**); output array that is the addition, subtraction, or product of two matrix arrays or a constant times a matrix array.

A2/ (**KX**, **KX**); local input array that holds a variable used for matrix operations.

A3/ (**KX**, **KX**); local input array that holds a variable used for matrix operations.

ALLARR/ (**IHUGE**, **MAXNES**); stores all arrays for all variables needed in memory.

AM/ (**MKX**, **NSPLIT**); = **DSIGMA**∗**A**, where **A** is a 2-D matrix operator defined in VMODES.

AN/ (**NSPLIT**); = **DSIGMA**∗**ZMATX.**

ANAME (8); character string to describe field having flagged errors in the split-explicit scheme.

ANEW (**MIX**, **MJX**); interpolated (coarse to nest) values at nested location after move.

AOLD (**MIX**, **MJX**); nested values before nest is moved.

ARR (**MIX**, **MJX**, **KZZ**); nested input array used for overwriting array **ARRN**.

ARRAY (**IX**, **JX**, **NSPLIT**); local input array (3-D in PRINTSP, 2-D in MAXIMI and MINIMI, 1-D in MAXIM and MINIM).

ARRB (**MIX**, **MJX**, **KZZ**); nested input array used for overwriting array **ARRNB**.

ARRBB (**MIX**, **KZZ**, **NSPG**); nested boundary values (**MJX** = **MIX** in EXCHANI).

ARRBT (**MIX**, **KZZ**, **NSPG**); nested boundary tendencies (**MJX** = **MIX** in EXCHANI).

ARRN (**MIX**, **MJX**, **KZZ**); nested input/output array being overwritten by array **ARR**.

ARRNB (**MIX**, **MJX**, **KZZ**); nested input/output array being overwritten by array **ARRN**.

ATEN (**MIX**, **MKX**); general model tendency for **UTEN**, **VTEN**, **TTEN**, and **QVTEN**.

AX (**MKX**); the main diagonal array in GAUSS and CADJMX.

AX value of updraft cloudwork function (J kg^{-1}) in CLOUDW and CLOUDWS.

AXD value of downdraft cloudwork function (J kg^{-1}).

B (**MKX**); coefficient arrays in GAUSS and CADJMX.

B an array used in ZX4LP.

BDYTIM/ time (min) after which the boundary conditions are needed.

BLDUM2D/ (**MIXF**, **MJXF**); scratch array used for analysis nudging.

BLPSOC (**MIXF**, **MJXF**); observed p* (cb) on cross points and used for surface analysis nudging.

BLPSOD (**MIXF**, **MJXF**); observed p* (cb) on dot points and used for surface analysis nudging.

BLWNV (**NVAR**, **MIXF**, **MJXF**); storage array for horizontal weighting function based on surface data density and used for surface analysis nudging within the PBL.

BLWXY (**MIXF**, **MJXF**); horizontal weighting function based on surface data density and used for surface analysis nudging within the PBL.

BZ/ (**MKX**, **MKX**); used in the transformation of geopotential from sigma space to vertical mode space (J kg^{-1} K^{-1}).

C (**MKX**); coefficient arrays in GAUSS and CADJMX.

C an array used in ZX4LP.

C201/ = (100-**PTOP**)/(**DX**∗**DX**).

C203/ = 1./(**DX**∗**DX**).

CD (**MKX**, **KNUM**); detrainment rate (m^{-1}) in updrafts.

C0 conversion rate (m^{-1}) of liquid water to rain.

CONST constant subtracted from array **FLD**.

CZ/ (**MKX**, **MKX**); used in the transformation of geopotential from sigma space to vertical mode space (J kg^{-1}).

D (**MKX**); working array in GAUSS and CADJMX.

D local variable in INVMTX.

DA (**IA**, **MINP**); working array filled with kernels.

DB (**MBOTH**); working array containing kernels (large-scale forcing).

DC (**NIMSL**); working array containing kernels (large-scale forcing).

DDSOL (**MBOTH**); output from ZX4LP (not needed).

DDSUM (**MIX**, **MJX**, **NSPLIT**); summation of **DELD** over the split-explicit short time steps.

DELD (**MIX**, **MJX**, **NSPLIT**); divergence (cb s^{-1}) in vertical mode space.

DELD (**MIX**, **MJX**, **NSPLIT**, 3); time-step difference of divergence (cb s^{-1}) in vertical mode space

DELH (**MIX**, **MJX**, **NSPLIT**); geopotential (m^2 s^{-1}) in vertical mode space.

DELH (**MIX**, **MJX**, **NSPLIT**, 3); time-step difference of geopotential (m^2 s^{-1}) in vertical mode space

DELLA (**MKX**); change of any thermodynamic variable due to a unit amount of cloud.

DELQ (**MKX**, **KNUM**); change of a moisture variable due to a unit amount of cloud (kg kg^{-1} mb^{-1}).

DELT (**MKX**, **KNUM**); change of a temperature variable due to a unit amount of cloud (K mb^{-1}).

DELTSM time step (s).

DHSUM (**MIX**, **MJX**, **NSPLIT**); summation of **DELH** (m^2 s^{-1}) over the split-explicit short time steps.

DPSOL (**NIMSL**); mass fluxes from ZX4LP.

DRW (**IWW**); local working array.

DSIGMA / (**MKX**); thickness of the sigma layer.

DSOL an array used in ZX4LP.

DT model time step (s).

DTAU / (**NSPLIT**); short time step (s) for the split-explicit scheme.

DTB elapsed time (s) from the initial boundary values.

DTHDPC critical lapse rate (K cb^{-1}).

DTIME time step (s) over which cumulus parameterization is applied.

DTL = **DT** (standard time step for model).

DTMIN / **DT** / 60 (min).

DX / grid length (m).

DX2 / = 2∗**DX**.

DXX the horizontal distance (= **DX4**, **DX**, **DX16**, when **IND**= 1, 2, 3 respectively).

EDT (**KNUM**); variable relating the strength of the downdraft mass flux to the updraft mass flux.

EI (**MKX**); local input array that is checked for negative and non-real eigenvalues in the VMODES calculations.

ENT entrainment rate (m^{-1}).

ER (**MKX**); local input array that is checked for negative and non-real eigenvalues in the VMODES calculations.

F (**MIX**, **MKX**); local working array in VADV.

F (3, **MIX**, **MKX**); input array in HADV.

F (**MKX**); large-scale forcing (J kg^{-1} s^{-1}) in ARAMB).

FA (**MIX**, **MJX**, **KZZ**); input array p***F** at time t.

FB (**MIX**, **MJX**, **MKX**); local name for an input array.

FBSCAT (**MIX**); backscattering coefficient (0 - 1).

FCOEF coefficient for the Newtonian term.

FDTIM model time (min) at time t-1.

FEB (**MIX**, **KD**, **IP**+1); observed boundary values at time **TBE** on the east boundary.

FEBT (**MIX**, **MKX**, **IP**+1); large-scale or observed tendencies at east boundary.

FIN (**MIX**, **MJX**, **MKX**); input array.

FLD (**JXX**, **IYY**); array to hold the data field to be printed.

FNB (**MIX**, **KD**, **IP**+1); observed boundary values at time **TBE** on the north boundary.

FNBT (**MIX**, **MKX**, **IP**+1); large-scale or observed tendencies at north boundary.

FOUT (**MIX**, **MJX**, **MKX**); output array.

FSB (**MIX**, **KD**, **IP**+1); observed boundary values at time **TBE** on the south boundary.

FSBT (**MIX**, **MKX**, **IP**+1); large-scale, or observed tendencies at south boundary.

FTABS (**MIX**); absorption transmissivity (0 - 1).

FTEN (**MIX**, **MKX**); local tendency of **F**.

FTEST scratch variable.

FTSCAT (**MIX**); scattering transmissivity (0 - 1).

FWB (**MIX**, **KD**, **IP**+1); obs boundary values at time **TBE** on the west boundary.

FWBT (**MIX**, **MKX**, **IP**+1); large-scale, or obs tendencies at west boundary.

FYX (**IMAX**∗**JMAX**); local working array.

G / {PARAMETER} gravitational acceleration (= 9.8 m s^{-2}).

GCOEF coefficient for the diffusion term.

GMT / Greenwich Mean Time of the initial data (hours).

GNUHF constant used in Asselin time filter for all prognostic variables (= 0.1).

GP / (**MAXSES**, 2); analysis-nudging coefficient (s^{-1}) for p*.

GR / (**MAXSES**); analysis-nudging coefficient (m^2 s^{-1}) for vorticity.

GX (**MAXSES**, 2); general analysis nudging coefficient for FDDA.

GX (**MIX**, **MJX**, **KL**); gradient of **PHI** (m s^{-2}) in vertical mode space in the x-direction.

GY (**MIX**, **MJX**, **KL**); gradient of **PHI** (m s^{-2}) in vertical mode space in the y-direction.

H (**MKX**); environmental moist static energy.

HBAR / (**MKX**); equivalent depths (m^2 s^{-2}) of the vertical modes.

HC (**MKX**, **KNUM**); updraft moist static energy (m^2 s^{-2}).

HCD (**MKX**, **KNUM**); downdraft moist static energy (m^2 s^{-2}).

HE (**MKX**); environmental moist static energy (m^2 s^{-2}).

HES (**MKX**); saturation moist static energy (m^2 s^{-2}).

HKB moist static energy at cloud base (m^2 s^{-2}).

HSAT (**MKX**); saturation moist static energy (m^2 s^{-2}).

HSC1 (**IYY**, **JXX**); local working array.

HSCR1 (**IYYN**, **JXXN**); terrain height times gravity (m^2 s^{-2}).

HT/ (**MIX**, **MJX**); terrain height times gravity ($m^2\ s^{-2}$).

HTNO (**IYYN**, **JXXN**); g*terrain ($m^2\ s^{-2}$) from old nest before moving.

IA {PARAMETER}; = **MBOTH**+2 in ARAMB.

IA a variable used in ZX4LP.

IA initial sampling point in the first dimension in MAPSMP.

IARG beginning **I**-index value in arrays **ARR**, **ARRB** for the exchange.

IARGN beginning **I**-index value in overwritten arrays for the exchange.

IARR (**NUMVAR**); address location.

IB local name for **IBOUDY** (0 - 4) or last sample point in MAPSMP.

ICHOS flag denoting type of array.

ICOARS integer used as dimension index denoting nest level.

ICOMPS integer flag used for optimization.

ICRSDOT integer used for denoting whether dot or cross point.

ID local name for **IDRY** in CONADV and CONMAS.

ID array dimension in **I**-direction in NUDGE and SPONGE.

ID flag denoting type of nudging in BLNUDGD and NUDGD:
= 0; surface-analysis nudging for U, V, T, or q_v.
= 1; surface-analysis nudging for p*.
= 2; vorticity nudging.
= 3; surface-analysis nudging for q_v using precipitation analysis.

ID3 local first dimension for arrays.

ID4 local second dimension for arrays.

IDARST on / off (1, 0) switch to indicate restart with observation nudging and at least one observation available in time window.

IDCHK/ (**NCHA**, **NVAR**); array identifying what variables at **IDDATE** dates to exclude from surface-analysis nudging.

IDDATE/ (**NCHA**); array identifying what dates to exclude from surface-analysis nudging.

IDHK/ (**NVAR**); data-quality flag array used by the **INOPRO** option.

IDIM used locally as first dimension of an output array.

IDRY/ (**MAXSES**); flag denoting whether this run is a moist or dry forecast, (0=moist; 1=dry).

IE number of points in the **I**-dimension (= **ID** for dot point, **ID**-1 for cross point).

IEN local name for maximum **I**-value.

IEND local name for the maximum **I**-index.

IENDN ending **I**-index of array being overwritten.

IER flag (if non zero) indicating an error in linear algebra routines used for model calculations.

IERRT flag (if non zero) indicating an error in linear algebra routines for the Arakawa-Schubert convection scheme.

IEXEC number of times a subroutine has been called (1=first, >1=subsequent).

IFREST/ is this run is a restart, (.T. or.F.).

IHUGE {PARAMETER} see definition given in PARAMETER list (section 3).

IINC **I**-increment for printing (= 6).

IL/ number of grid points in the **I**-direction (= **IX**).

ILX/ = **IL**-1.

ILXM/ = **ILX**-1.

IM local name for variable **IMOIST**.

IMAX maximum number of points in **I**-direction.

IMIN beginning **I**-index in the **I**-direction.

IMOIST/ (**MAXSES**); indicates if cumulus parameterization or explicit moisture:
= 0; dry case with passive, moist variables.
= 1; no explicit moisture.
= 2; explicit moisture.

IN integer for mesh (1, 2, 3... is coarse, nest, second nest etc.).

IN1 (**MIX**, **MJX**, **MKX**, 3); local input array holding the **DELD** or **DELH** arrays.

IN2 (**MIX**, **MJX**, **MKX**, 3); local input array holding the **DELD** or **DELH** arrays.

IND have **UA** and **VA** been multiplied by **MSFD** (0, 1) in DIFFU.

IND index = 1 for T, 2 for (q_v, q_c, q_r, q_i, q_s,), 3 for (U, V) 4 for (W) in HADV, 4 for (U, V), 5 for W, 6 for p' in VADV.

IND flag for type of input in JULGMT (0= **MDATE**, 1=**TIMANL**, 2=**JULGMTN**).

INEST integer (1, 2, 3...) defining domain for which tendencies are computed.

INFR = **IONF**$*($**IRATIO** $^{\textbf{LEVIDN(INEST)}})$. Determines the time-step frequency for calling certain observation-nudging subroutines.

INSTES integer specifying whether nest is to be moved and will it overlap.

INVAR (**MAXSES**, 2); general on/off analysis-nudging switch for variable **IVAR**.

INY sampling interval in the first dimension.

IOVE number of points to be exchanged.

IP number of slices affected by sponge and relaxation boundary condition in SPONGE and NUDGE.

IP (**NK**); working array in INVMTX.

IQCHK/ (**NTIM**, **NVAR**); flag denoting quality of data (density) for analysis nudging in PBL (if = 0, data are not used for FDDA).

IS1 local maximum index value in the **I**-direction.

IS2 local maximum index value in the **J**-direction.

ISKIP number of files or records to skip.

ISOUTH/ **I**-coordinate (large domain) for south boundary of nest (= **NESTI**).

ISOUTHO old **ISOUTH** before moving nest.

IST local name for beginning **I**-value.

ISTART local name for the starting **I**-index.

ISTART beginning **I**-index of overwriting array in EXCHANJ.

ISTARTN beginning **I**-index of array being overwritten.

ISTOP number of corner grid points not being overwritten

ITEST (**KNUM**); scratch array for testing.

ITEST2 (**KNUM**); scratch array for testing.

IUNIT unit number from which the data are read.

IUNITH unit number of initial condition input file.

IUTL unit number of standard model output.

IUTSAV output unit number in OUTSAV and input unit number in SAVREAD.

IVAR variable code where **IVAR** = 1, 2, 3, 4, 5 denotes U, V, T, q_v, p* respectively.

IW (**IW1**); working array needed for ZX4LP.

IW1 {PARAMETER} = 2*(**NIMSL**+**MEQU**)+3.

IWW {PARAMETER} = **IA***(**IA**+2)+2*(**NIMSL**+**MEQU**).

IX number of points in **I** (y) direction.

IXI first dimension for input array.

IXO first dimension for output array.

IY number of points in **I** (y) direction.

IYWANT **I**-index for which interpolation is to be performed.

IYY standard first dimension for variables.

IYYM dimension for $p*q_c$ and $p*q_r$.

IYYN standard first dimension for variables (nest).

IYYNM **I**-dimension of nested domain.

J	index for **J**-slice currently being calculated.
JA	initial sampling point in the second dimension.
JARG	beginning **J**-index value in arrays **ARR**, **ARRB** for the exchange.
JARGN	beginning **J**-index value in overwritten arrays for the exchange.
JB	final sampling point in the second dimension.
JDIM	used locally as second dimension of an output array.
JE	number of points in the **J**-dimension (= **JD** for dot point, **JD**-1 for cross point).
JEN	local name for the maximum **J**-index.
JEND	local name for the maximum **J**-index value.
JENDN	ending **J**-index of array being overwritten.
JINC	**J**-increment for printing (= 6).
JL/	the total number of grid points in the **J**-direction (= **JX**).
JLX/	= **JL**-1.
JLXM/	= **JLX**-1.
JMAX	maximum number of points in **J**-direction.
JMIN	beginning **J**-index in the **J**-direction.
JNX	sampling interval in the second dimension.
JOVE	number of points to be exchanged.
JST	local name for the starting **J**-index.
JSTART	beginning **J**-index of overwriting array.
JSTARTN	beginning **J**-index of array being overwritten.
JULDAY/	Julian day of the initial data set.
JULGMTN	Julian day times 100 plus the **GMT** of the observed analysis used for nudging.

JWEST/ **J**-coordinate (large domain) for west boundary of nest (= **NESTJ**).

JWESTO old **JWEST** before moving nest.

JX number of points in **J** (x) direction.

JXI second dimension for input array.

JXO second dimension for output array.

JXX standard second dimension for variables.

JXXM dimension for p^*q_c and p^*q_r.

JXXN standard second dimension for variables (nest).

JXXNM **J**-dimension of the nested domain.

JXWANT **J**-index for which interpolation is to be performed.

K vertical level.

KB lower limit of unstable layers in CADJMX.

KB updraft originating level in the cumulus parameterization routines.

KBC level of free convection.

KBCON level of free convection.

KBEG updraft originating level.

KBMAX highest possible cloud base level.

KD array dimension in the **K**-direction.

KDET (**MIX**); level where downdraft detrainment begins.

KDIM {PARAMETER} = **MKX**.

KE upper limit of unstable layers in CADJMX.

KE ending do-loop index in the vertical in MAXIM and MAXIMI.

KEND ending do-loop index in the vertical.

KEPBLH (**MIX**); **K**-level where the PBL top is located in east boundary slice.

KL/ = **MKX**, except in split-explicit routines where it sometimes = **NSPLIT**.

KLEV = **KDIM**.

KMIN (**MKX**); downdraft originating level.

KNUM {PARAMETER} number of cloud types (= 6 in ARASCH and 1 in SHALLOW).

KPBLT (**MIXF**); vertical layer index for PBL top.

KREF = 1.

KS (**MIX**); beginning index in the vertical.

KT (**MIX**); flagged level from MIMIMI and MINIM.

KTAU / counter for time steps.

KTAUR / counter for time steps when restarting.

KTOP (**KNUM**); cloud top level.

KV1 = **KXP1**.

KWPBLH (**MIX**); **K**-level where the PBL top is located in west boundary slice.

KX {PARAMETER} = **MKX**, except in PRINTSP where **KX** = **NSPLIT**.

KXI third dimension for input array.

KXO third dimension for output array.

KZZ standard third dimension for variables.

KZZM dimension for p^*q_c and p^*q_r.

KZZN standard third dimension for variables (nest).

KZZNM **K**-dimension of the nested domain.

L nested domain number.

L1 cloud type.

LM order of the matrix and local name for **NADJ**.

LP index of cloud-type loop.

LSTAND use standard atmospheric temperature for the thermodynamic matrix, (.T. or .F.).

M / (**NSPLIT**); ratio (long time-steps / short time-steps) in SPSTEP2.

M / second dimension of 3-D working array in SINT, SINTX, and SINTY.

M1 a variable used in ZX4LP.

M2 a variable used in ZX4LP.

MAX (**MIX**); output level in MAXIMI and MAXIM.

MBDT arbitrary unit mass flux * time-step (kg m^{-2}).

MBOTH {PARAMETER} = **MEQU**+**MNEQU**.

MC local name for variable **MASCHK**, the frequency in timesteps that the mass conservation information will be printed.

MCHA number of **IDDATE** files to check.

MDATE / time and date of initial data set in yymmddhh format.

MDATES dates (as in **MDATE**) of surface analyses.

MINP {PARAMETER} = **MEQU**+**NIMSL**+2.

MIX {PARAMETER} maximum dimension (all domains) in the y-direction.

MJX {PARAMETER} maximum dimension (all domains) in the x-direction.

MKX {PARAMETER} maximum dimension (all domains) in the vertical direction.

MSD (**MIXF**, **MJXF**); map-scale factor at dot points.

MSF (**MIX**, **MJX**); local map scale factor.

MSFD / (**MIX**, **MJX**); map-scale factor at dot points.

MSFX / (**MIX**, **MJX**); map-scale factor at cross points.

MSX (**MIXF**, **MJXF**); map-scale factor at cross points.

MTIM = **NTIM**.

MVAR = **NVAR**.

N a variable used in ZX4LP.

N = **MJX** in SINT, SINTX, and SINTY.

N = **NK** in INVMTX and VPRNTV.

N nest number in OUTPRT.

N1 {PARAMETER} = **MJX**, except in VPRNTM where **N1** = **NSPLIT**.

N1END ending loop index.

N1STAR starting loop index.

N2 {PARAMETER} = **MIX**, except in VPRNTM where **N2** = **NSPLIT**.

N2END ending loop index.

N2STAR starting loop index.

NA = **NK**.

NADJ number of slices between **KB** and **KE**. (= **KE**-**KB**+1).

NAM (8); character string describing field to be printed.

NAME (5); character string describing field to be printed in MAPSMP.

NAME (8); character string describing field to be printed in PRINTSP.

NESCOU domain number.

NESCOUO old domain number.

NESTII = **ISOUTH**.

NESTJJ = **JWEST**.

NI = **IL**.

NI1 = **ILX**.

NIMSL {PARAMETER} = 2∗**MNEQU**.

NJ = **JL**.

NJ1 = **JLX**.

NK {PARAMETER} = **KX**, except in XTDOT where **NK** = 1.

NK1 {PARAMETER} = **NK**+1.

NN N'th time step for the nest from previous large domain forecast.

NPASS number of smoothing passes.

NSPG = **NSPGD** or **NSPGX** (PARAMETERS).

NST N'th time step for the nest from previous large domain forecast.

NT position counter from 1 to **NTIM** in CONV3 for surface analysis.

NT fourth dimension index (1 or 3) for **OUT** array in SPDIFF.

NT1 fourth dimension index (1 or 2) for **IN1** array.

NT2 fourth dimension index (1 or 3) for **IN2** array.

NTB / position counter from 1 to **NTIM** for **TIMB** of **SFCTIM** and **SFCOBS** arrays.

NTE / position counter from 1 to **NTIM** for **TIME** of **SFCTIM** and **SFCOBS** arrays.

NUMERR number of errors in the split-explicit calculations.

NUMN nest number of lower-level nest.

NUMNE nest number of nest being moved.

NUMNES / integer (1, 2, 3...) defining domain for which addresses are computed.

NV variable code where **NV** = 1, 2, 3, 4, 5 denotes U, V, T, q_v, p* respectively in FDDA routines.

NV = **NK** in INVMTX.

OUT (**MIX**, **MJX**, **MKX**, 3); output array holding the **DELD** or **DELH** arrays.

OUTQ (**MIX**, **MKX**); output tendencies of q_v (kg kg^{-1} s^{-1}).

OUTTEM (**MIX**, **MKX**); output tendencies of T (K s^{-1}).

P (**MIX**, **MKX**); pressure (mb), not p*.

PATH (**MIX**); normalized path length for incoming radiation.

PCUT pressure (mb) at which all rain is removed.

PD (**MIX**, **MJX**); = **PSDOT** (cb) in XTDOT.

PD / = **PS - PTOP**, where **PS** is an average value of p* .

PHI / (**MJX**, **MIX**, **KL**); = **DELH**.

PI (**MKX**); Exner's function.

PIO (**MKX**); pressure (mb) after large-scale forcing has been applied.

PO (**MIX**, **MKX**); pressure (mb) after large-scale forcing has been applied.

PRE (**MIX**); precipitation rate (mm s^{-1}).

PRW / precipitable water (cm).

PS (**MIX**, **MJX**); p* (cb).

PSA / (**MIX**, **MJX**); p* (cb) at time t.

PSADOT (**MIX**, **MJX**); **PSA** (cb) interpolated to dot points.

PSB / (**MIX**, **MJX**); p* (cb) at time t-1.

PSBD (**MIXF**, **MJXF**); model p* (cb) on dot points at time t-1.

PSBDT (**MIX**, **MJX**); model p* (cb) on dot points at time t-1.

PSDOT (**MIX**, **MJX**); **PSA** (cb) interpolated to dot points.

PSO / (**MIXF**, **MJXF**); observed p* (cb) on cross points.

PSOC / (**MIXF**, **MJXF**); observed p* (cb) on cross points interpolated in time.

PSOD / (**MIXF**, **MJXF**); observed p* (cb) on dot points interpolated in time.

PSOL an array used in ZX4LP.

PSOTEN / (**MIXF**, **MJXF**); observed p* tendency (cb min^{-1}) on cross points.

PSTF (**MIXF**, **MJXF**); model p* (cb) on dot or cross points.

PSTO (**MIXF**, **MJXF**); observed p* (cb) on dot or cross points.

PSU p_s (mb).

PSUR p_s (mb).

PSURF p_s (mb).

PSURO p_s (mb) after large-scale forcing has been applied.

PT / = **PTOP**.

PTEN / (**MIX**, **MJX**); the tendency of p* (cb s^{-1}).

PTOP/ pressure (cb) at the top of the model.

PW (**MKX**, **KNUM**); normalized condensate (kg kg^{-1}).

PWD (**MKX**, **KNUM**); normalized evaporate (kg kg^{-1}).

PX (**MIX**, **MJX**); = **PSA** (cb).

Q (**MIX**, **MKX**); = q_v (kg kg^{-1}).

QC (**MKX**); local working array.

QCA/ (**MIXM**, **MJXM**, **MKXM**); p^*q_c (cb kg kg^{-1}) at time t.

QCTEN (**MIXM**, **MKXM**); tendency of p^*q_c (cb kg kg^{-1} s^{-1}).

QDOT/ (**MJX**, **MIX**, **KXP1**); vertical sigma-velocity (s^{-1}).

QE (**MKX**); = q_v (kg kg^{-1}).

QES (**MKX**); saturation q_v (kg kg^{-1}).

QI (**MKX**); = q_v (kg kg^{-1}).

QIO (**MKX**); **QI** (kg kg^{-1}) after large-scale forcing.

QITEN (**MIXM**, **MKXM**); tendency of p^*q_i (cb kg kg^{-1} s^{-1}).

QKB q_v (kg kg^{-1}) at cloud base.

QNITEN (**MIXM**, **MKXM**); tendency of p^*q_{ni} (cb kg kg^{-1} s^{-1}).

QO (**MIX**, **MKX**); **QI** (kg kg^{-1}) after large-scale forcing.

QRA/ (**MIXM**, **MJXM**, **MKXM**); p^*q_r (cb kg kg^{-1}) at time t.

QRC (**MKX**, **KNUM**); q_v (kg kg^{-1}) in cloud in PRECIP.

QRCD (**MKX**); q_v (kg kg^{-1}) in downdraft.

QRTEN (**MIXM**, **MKXM**); tendency of p^*q_r (cb kg kg^{-1} s^{-1}).

QSATF/ (**MIXF**, **MJXF**); model saturation mixing ratio (kg kg^{-1}).

QTEN (**MIX**, **MKX**); q_v tendency (cb kg kg^{-1} s^{-1}).

QVA (**MIX**, **MJX**, **MKX**); p^*q_v (cb kg kg^{-1}) at time t.

QVB (**MIX**, **MJX**, **MKX**); p^*q_v (cb kg kg^{-1}) at time t-1.

QVTEN (**MIX**, **MKX**); tendency of p^*q_v (cb kg kg^{-1} s^{-1}).

R cloud updraft radius (m) in the cumulus parameterization routines.

R / gas constant for dry air in VTLAPS (= 287. J kg^{-1} K^{-1}).

RADS radius of shallow cloud (m).

RAINC / (**MIX**, **MJX**); accumulated convective rain (cm).

REGJ (**MIXF**); model PBL regime.

RINBLW / radius of influence (km) for surface analysis nudging where the weighting function depends on surface data density.

RW an array used in ZX4LP.

S (**MKX**); temperature (K) in CADJMX.

S (**NK1**); **SIGMAF** input array in VNORML.

S a variable used in ZX4LP.

SCR (**MIX**, **MKX**); temporary storage array.

SCR1 (**MIX**, **MKX**); temporary storage array.

SCR2 (**MIX**, **MKX**); temporary storage array.

SCR2D / (**MIXF**, **MJXF**); temporary storage array used for analysis nudging.

SCR9 (**MIX**, **MKX**); temporary storage array holding vertical motion.

SFCOBS / (**NTIM**, **NVAR**, **MIXF**, **MJXF**); surface analysis array for each variable at each time used for surface-analysis nudging.

SFCTIM / (**NTIM**); corresponding model time (min) for each observed surface analysis.

SIG (**KZZ**); half-sigma levels.

SIGMA / (**KXP1**); full-sigma levels, except half-sigma levels in VTLAPS.

SIGMAF (**KXP1**); full-sigma levels.

SIGMAH / (**MKX**); half-sigma levels.

SLAB (**IMAX**, **JMAX**); local working array.

SLAB1 (**IYY**, **JXX**); local array used in cross-to-dot interpolation subroutines.

SLAB2 (**IYY**, **JXX**); local array used in cross-to-dot interpolation subroutines.

SLP (**IYY**, **JXX**); sea-level pressure (cb) array.

T (**MIX**, **MJX**, **MKX**); temperature (K) array except in VTLAPS where it is a 1-D array (**TBARH**).

T0 / (**IDIM**, **JDIM**); sea-level temperature (K) in SEAPRS. Also used as a PARAMETER in VTLAPS and as a reference temperature array in common block NHCNS.

TA / (**MIX**, **MJX**, **MKX**); p*T (cb K) at time t.

TA (**IYY**, **JXX**); local input array in EXAINT.

TAN (**IYYN**, **JXXN**); local output array.

TB / (**MIX**, **MJX**, **MKX**); p*T (cb m s^{-1}) at time t-1.

TBARH / (**MKX**); the average temperature (K) on sigma half-levels.

TBDYBE / initial time (min) of the present boundary conditions.

TBE time (min) for boundary values **FEB**, **FWB**, **FSB**, and **FNB**.

TCMASS total mass of cloud water (kg).

TDRYM total mass of the dry air (kg).

TEMPP (**MKX**); temporary array.

TER (**MIX**, **MJX**); g*terrain (m^2 s^{-2}).

TFAC / (**MAXNES**, 2); temporal weighting factor for analysis nudging.

TI (**MKX**); temporary array.

TIMANL / (**MAXSES**); model time (min) of observed analysis for analysis nudging.

TIMANLS model time (min) of observed surface analysis.

TIMB / (**NVAR**); beginning bracketing time (min) for temporal interpolation of observed surface analysis.

TIME / (**NVAR**); ending bracketing time (min) for temporal interpolation of observed surface analysis.

TIO (**MKX**); temporary array after large-scale forcing has been applied.

TN (**MIX**, **MKX**); temporary array after large-scale forcing has been applied.

TQMASS total mass of water substance (kg).

TRMASS total mass of rain water (kg).

TS (**IDIM**, **JDIM**); surface temperature (K) array.

TTEN (**MIX**, **MKX**); tendency of p*T (cb K s^{-1}).

TVMASS total mass of water vapor (kg).

U (3, **MIX**, **MKX**); decoupled U (m s^{-1}) in DECPU.

U (**MIX**, **MJX**, **MKX**); = **UA** or **UB** (cb m s^{-1}) in SPDIVG and DIVG.

UA / (**MIX**, **MJX**, **MKX**); p*U (cb m s^{-1}) at time t.

UB / (**MIX**, **MJX**, **MKX**); p*U (cb m s^{-1}) at time t-1.

UOB (**MIXF**, **MJXF**, **MKXF**); storage array of observed analysis of p*U (cb m s^{-1}).

UOBTEN (**MIXF**, **MJXF**, **MKXF**); storage array of observed tendency of p*U (cb m s^{-1} min^{-1})

UTEN (**MIX**, **MKX**); tendency of p*U (cb m s^{-1}).

V (3, **MIX**, **MKX**); decoupled V (m s^{-1}) in DECPU.

V (**MIX**, **MJX**, **MKX**); = **VA** or **VB** (cb m s^{-1}) in SPDIVG and DIVG.

V (**MKX**, **MKX**); inverted output matrix array from INVMTX.

VA / (**MIX**, **MJX**, **MKX**); p*V (cb m s^{-1}) at time t.

VAR (**MKX**); input variable for q (moisture) or H (moist static energy).

VB / **MIX**, **MJX**, **MKX**); p*V (cb m s^{-1}) at time t-1.

VOB (**MIXF**, **MJXF**, **MKXF**); storage array of observed analysis of p*V (cb m s^{-1}).

VOBTEN (**MIXF, MJXF**, **MKXF**); storage array of observed tendency of p*V (cb m s^{-1} min^{-1})

VORDIF / (**MIXF**, **MJXF**, **MKXF**); difference of observed and model vorticity (s^{-1}).

VSP (**MIX**, **MKX**); wind speed (m s^{-1})

VTEN (**MIX**, **MKX**); tendency of p*V (cb m s^{-1}).

WG (**IP**); weights for sponge boundary conditions.

WH (**MKX**, 2); local working array.

WORK (**MIX**, **MJX**, 3); local working array.

WPBL (**MIXF**, **MKXF**); analysis-nudging vertical weighting function based on model PBL height determining partitioning of types of nudging above and within the PBL.

WTTOP (**MIX**, **MJX**); horizontal weighting array based on topography and optionally used to compute horizontal weighting function for surface-analysis nudging.

WXY (**MIXF**, **MJXF**); general horizontal weighting function for analysis nudging.

WXY2 (**MIXF**, **MJXF**); horizontal weighting function for analysis nudging of mixing ratio based on observed precipitation analyses.

WZ (**MKX**, **MKX**); local working array.

XB (**MIX**, **MJX**, **MKX**); general model field at time t-1.

XF (**MJX**, **MIX**, **NF**); input or output working array to be interpolated.

XHKB moist static energy (m^2 s^{-2}) at cloud base.

XK (**MIX**, **MKX**); horizontal diffusion coefficient (m^2 s^{-1}) in DIFFU and DIFFUT.

XK (**KNUM**, **KNUM**); array containing kernels (J kg^{-1} mb^{-1} s^{-1}) in ARAMB.

XKAPPA {PARAMETER} (= R/c_p = .287).

XMB (**KNUM**); cloud base mass flux (kg m^{-2} s^{-1}).

XMC (**MKX**); cloud mass flux (kg m^{-2} s^{-1}).

XOB (**MIXF**, **MJXF**, **MKXF**); general observed analysis array at time t-1.

XOBJK (**MIXF**, **MJXF**); general north-south slice array for observed variable, decoupled from p* and temporally interpolated to time t-1.

XOBTEN (**MIXF**, **MJXF**, **MKXF**); general observed analysis tendency for period **DIFTIM**.

XT time (min) for variable **FB**.

XTEN (**MIX**, **MKX**); general model tendency field.

XTIME / forecast time (min).

XVAR (**MKX**); variable that has been changed due to the cloud.

Y (**MKX**); right-hand side on input and the solution on output.

Z (**MKX**); height (m).

Z (**NK**, **NK**); = **ZMATX** in VNORML and VORDER.

Z (**MIX**, **MJX**, **KL**); divergence (cb s^{-1}) in DIVG.

Z1 (**MIX**); terrain (m).

ZD (**MKX**, **KNUM**); normalized downdraft mass flux.

ZFAC / (**MAXNES**, 2, **MKXF**); vertical weighting factor for analysis nudging.

ZMATX / (**MKX**, **NSPLIT**); a matrix used in the transformation of divergence from sigma space to vertical mode space.

ZMATXR / (**MKX**, **NSPLIT**); matrix inverse of **ZMATX**.

ZNTJ (**MIXF**); roughness length (m).

ZU (**MKX**, **KNUM**); normalized mass flux for the cloud updraft.

Section 7

NAMELIST RECORDS

This section lists the variables that are contained in the four NAMELIST records $OPARAM, $PPARAM, $LPARAM, and $FPARAM. The definition, dimension, and units for all variables (listed alphabetically for each record) are given along with a brief description of the record's general contents.

7.1 OPARAM

$OPARAM contains variables used mainly for selection of options regarding model output.

IFPRT is printer output desired, (0=no;1=yes).

IFREST is this run restarted from a saved file, (.T. or. F.).

IFSAVE will a saved file will be written for restart, (.T. or .F.).

IFTAPE will output be saved on files for INTERP, (0=no;1=yes).

ISFOUT will output of surface/terrain parameters be printed (**IFPRT**=1), (0=no;1=yes).

IXTIMR restart time (min) into forecast.

LEVIDN (**MAXSES**); nest level.

MASCHK time-step frequency for printout of mass conservation information.

NUMNC (**MAXSES**); mother domain.

PRTFRQ interval (min) for printer output (**IFPRT**=1).

SAVFRQ interval (min) between save operations (**IFSAVE**=.T.).

TAPFRQ interval (min) of output data for INTERP (**IFTAPE**=1).

7.2 PPARAM

$PPARAM contains variables used for selection of options related to physical processes, model numerics, and vertical resolution.

ALBLND albedo over land, used when **ISFPAR** = 0.

CONF condensation threshold (= 1.).

IABSOR sponge absorber at top of model, (0=no;1=yes).

IFEED feedback option:
= 0; one-way.
= 1; MM4 method.
= 2; no smoothing.
= 3; light smoothing.

ISOLVE = 1 (standard numerics).

PTOP pressure (cb) at the top of the model.

QCK1 constant autoconversion rate (= 1.E-3 kg kg^{-1} s^{-1}).

QCTH threshold for the onset of autoconversion (kg kg^{-1}).

SIGMA (**KXP1**); full-sigma levels.

THINLD thermal inertia over land when **ISFPAR** = 0 (cal cm^{-2} K^{-1} $s^{-1/2}$).

TIMAX maximum forecast time (min).

TISTEP model **DT**, use 3∗**DX** for ISOLVE = 1.

XMAVA moisture availability, used when **ISFPAR** = 0.

ZZLND roughness length (m) over land, used when **ISFPAR** = 0.

ZZWTR roughness length (m) over water, used when **ISFPAR** = 0.

7.3 LPARAM

$LPARAM contains variables (mainly integers) pertaining to selection of options regarding physical processes, boundary conditions, nested domain specifications, PBL processes, and printer output.

HYDPRE (**MAXSES**); will water-loading effects be considered in hydrostatic equation (**IMOIST**=2), (0=no;1=yes).

IACTIV (**MAXSES**); is nested domain active, (0=no;1=yes).

IBLIQ are **QCTEN** and **QRTEN** at nested boundary available:
= 0; no, when **IDRY**(1) = 1, or when **IDRY**(1) = 0 and **IMOIST**(1) = 1.
= 1; yes, when **IDRY**(1) = 0 and **IMOIST**(1) = 2.

IBLTYP (**MAXSES**); will bulk PBL or Blackadar PBL be used in the model:
= 0; frictionless.
= 1; bulk PBL.
= 2; multi-level Blackadar PBL.

IBMOIST will initial and boundary conditions be provided for water / ice variables, (0=no;1=yes).

IBOUDY (**MAXSES**); indicates type of lateral boundary conditions:
= 0; fixed.
= 1; relaxation.
= 2; time dependent (from observations or large-scale model).
= 3; time and inflow / outflow dependent. Nonhydrostatic boundary conditions.
= 4; sponge.

IBVAP is **QVTEN** at the nested boundaries available:
= 0; no when **IDRY**(1) = 1.
= 1; yes when **IDRY**(1) = 0.

ICDCON (**MAXSES**); are drag coefficients constants when using bulk PBL, (0=no;1=yes--function of terrain height only).

ICLOUD (**MAXSES**); will the radiation effects due to clouds be considered. Used if surface heat and moisture fluxes are calculated (**ISFFLX**=1) and ground temperature is predicted from the budget (**ITGFLG**=1), (0=no;1=yes).

ICOR3D will full Coriolis force including vertical component be considered (nonhydrostatic option only), (0=no;1=yes).

ICUPA (**MAXSES**); what type of cumulus parameterization will be employed:
= 1; none.
= 2; Anthes-Kuo scheme.
= 3; Grell-type scheme.
= 4; Arakawa-Schubert scheme.

ICUSTB will the stability check in the Kuo cumulus parametrization scheme be activated, (0=no;1=yes).

IDRY (**MAXSES**); is this run a moist or dry forecast, (0=moist;1=dry).

IEVAP (**MAXSES**); will evaporation effects be considered (**IMOIST**=2):
< 0; The evaporation of rainwater is not considered.
= 0; The evaporation is not considered for cloud or rain.
> 0; The evaporation is considered.

IEXICE will explicit moisture scheme with simple ice-physics effects be used, (0=no;1=yes).

IFDRY is this a fake dry run (no latent heat release), (0=no;1=yes).

IFRAD will radiative cooling of the atmosphere be considered:
= 0; no.
= 1; use simple radiation routine.
= 2; use full radiation (LWRAD and SWRAD).

IFSNOW (**MAXSES**); will snow-cover data be considered, (0=no;1=yes).

IFUPR will upper radiative boundary conditions be used, (0=no;1=yes).

IMOIAV (**MAXSES**); is moisture availability a function of time, (0=no;1=yes).

IMOIST (**MAXSES**); will explicit moisture be used:
= 0; dry case with passive, moisture variables (including q_v, q_c, q_r).
= 1; no explicit moisture.
= 2; explicit moisture.

IMOVCO (**MAXSES**); counter for how often nest is moved.

IMOVE (**MAXSES**); will nest be moved, (0=no;1=yes).

IMOVEI (**MAXSES**, 10); how many grid points to move nest in the **I**-direction.

IMOVEJ (**MAXSES**, 10); how many grid points to move nest in the **J**-direction.

IMOVET (**MAXSES**, 10); at what time will nest be moved.

IMVDIF will moist-adiabatic vertical diffusion in clouds be included, (0=no:1=yes).

IOVERW (**MAXSES**); will interpolated nested domain be over-written with user's own analysis, (0=no;1=yes).

ISFFLX (**MAXSES**); will surface heat and moisture fluxes be calculated, (0=no;1=yes).

ISFPAR (**MAXSES**); are surface/land-use parameters variable or constant. Used only if (**ISFFLX** = 1 and **ITGFLG**=1), (0=constant;1=variable).

ISHALLO (**MAXSES**); will shallow convection be used, (0=no;1=yes).

ITGFLG (**MAXSES**); indicates method for calculating ground temperature (**ISFFLX**):
= 1; it will be calculated from the budget.
= 2; it will be calculated from a sinusoidal function.
= 3; it will be determined from specified constants.

ITQPBL tendencies at the boundaries will be computed in HIRPBL (**IBLTYP**=2) when **IBOUDY** = 1, 2, 3, or 4, (0=no;1=yes).

IVMIXM (**MAXSES**); will vertical mixing of momentum be considered (**IBLTYP**=2), (0=no;1=yes).

JXSEX (**MAXSES**); **J**-index of the north-south vertical slice for printer output.

KXOUT (**MAXSES**); **K**-level of the horizontal slice for printer output.

NESTI (**MAXSES**); origin location of nest in **I**-direction in mother domain.

NESTIX (**MAXSES**); **I**-dimension of nest.

NESTJ (**MAXSES**); origin location of nest in **J**-direction in mother domain.

NESTJX (**MAXSES**); **J**-dimension of nest.

RADFRQ frequency (min) that solar radiation is computed, (**ISFFLX**=1;**ITGFLG**=1).

XENNES ending time of computations for a given nest.

XMOIST (**MAXSES**); will moisture effects be used in the thermodynamic equation, (0=no;1=yes).

XSTNES beginning time of computations for a given nest.

7.4 FPARAM

$FPARAM contains variables used for selection of FDDA options. (For arrays with second index = 2 this varies over type of analysis nudging: 1 = 3-D analysis nudging, 2 = surface analysis nudging within PBL).

DIFTIM (**MAXNES**, 2); time (min) between input analyses for analysis nudging.

DPSMX maximum p* change (cb) allowed within influence range of a surface observation used for observation nudging.

FDAEND **(MAXSES**); time (min) for termination of FDDA.

FDASTA (**MAXSES**); time (min) for initiation of FDDA.

GIQ (**MAXSES**); observation-nudging coefficient (s^{-1}) for mixing ratio.

GIT (**MAXSES**); observation-nudging coefficient (s^{-1}) for temperature.

GIV (**MAXSES**); observation-nudging coefficient (s^{-1}) for wind.

GQ (**MAXSES**, 2); analysis-nudging coefficient (s^{-1}) for mixing ratio.

GR (**MAXSES**, 2); analysis-nudging coefficient ($m^2\ s^{-1}$) for vorticity.

GT (**MAXSES**, 2); analysis-nudging coefficient (s^{-1}) for temperature.

GV (**MAXSES**, 2); analysis-nudging coefficient (s^{-1}) for wind.

I4D (**MAXSES**, 2); will FDDA analysis nudging be employed, (0=no;1=yes).

I4DI (**MAXSES**); will FDDA observation nudging be employed, (0=no;1=yes).

IMOIS (**MAXSES**, 2); will the mixing ratio be nudged from analyses, (0=no;1=yes).

INONBL (**MAXSES**, 4); will PBL fields be nudged from 3-D analyses when not using surface-analysis nudging within PBL. (0=yes; 1=exclude certain variables depending on integer value of second index).

IONF observation-nudging frequency in coarse grid time steps for observation-nudging calculations.

IROT (**MAXSES**); will vorticity be nudged from analyses, (0=no;1=yes).

ISMOIS (**MAXSES**); will the mixing ratio be nudged from observations, (0=no;1=yes).

ISTEMP (**MAXSES**); will the temperature be nudged from observations, (0=no;1=yes).

ISWIND (**MAXSES**); will the wind field be nudged from observations, (0=no;1=yes).

ITEMP (**MAXSES**, 2); will the temperature be nudged from analyses, (0=no;1=yes).

IWIND (**MAXSES**, 2); will the wind field be nudged from analyses, (0=no;1=yes).

IWINDS (**MAXSES**, 2); will logarithmic-wind adjustment of analyzed surface wind speed be used before applying it throughout the PBL, (0=no;1=yes).

NPFG coarse-grid time-step frequency for select diagnostic print of analysis nudging.

NPFI coarse-grid time-step frequency for select diagnostic print of observation nudging.

PFREE user-defined pressure level (cb) where terrain effect becomes small.

RINBLW radius of influence (km) for surface-analysis nudging where the horizontal weighting function depends on surface data density.

RINFMN multiplier for observation-nudging influence radius (**RINXY**) at the surface.

RINFMX multiplier for observation-nudging influence radius (**RINXY**) at the **PFREE** level.

RINSIG vertical radius of influence (on sigma) for distance-weighted nudging corrections (for observation nudging).

RINXY default horizontal radius of influence (km) for distance-weighted nudging corrections (for observation nudging).

TWINDO (time window)/2 (min) over which an observation will be used for nudging.

Appendix A:

MM5 Flow Charts

A flow chart for MM5 showing the calling structure of its subroutines is shown in Figure A1 with the flow charts (expansion) of major subroutines CHKNST**,** NSTLEV1**,** and SOLVE3 shown in Figures A2 and A3. Subroutines appended with a pound sign (#) are expanded elsewhere in the Figure, and those appended with an asterisk (*) are expanded in a different Figure. A few subroutines **(**ADDRX1C**,** ADDRX1N**,** DOTS**,** XTDOT, SKIPF, and EQUATE) are not shown in the flow charts. Information concerning which subroutines call them can be found in section 5.

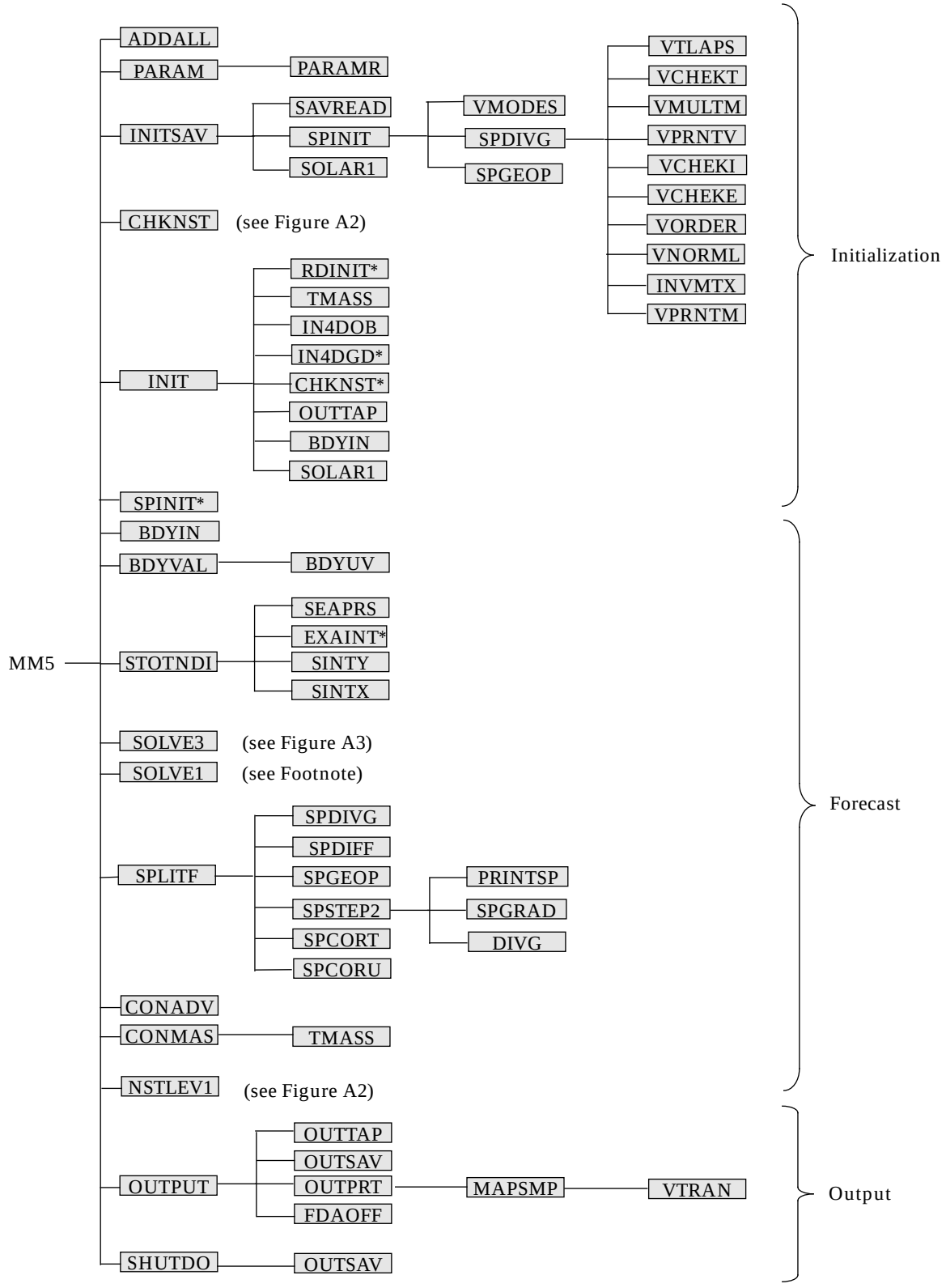

Figure Al. Flow chart of MM5. The expansions of the subroutines appended with an asterisk are shown in Figure A2.

Note: The flow chart structure of subroutine SOLVE1 is very similar to that of SOLVE3 except that subroutine SOUND is not called but subroutine SPONGE is called.

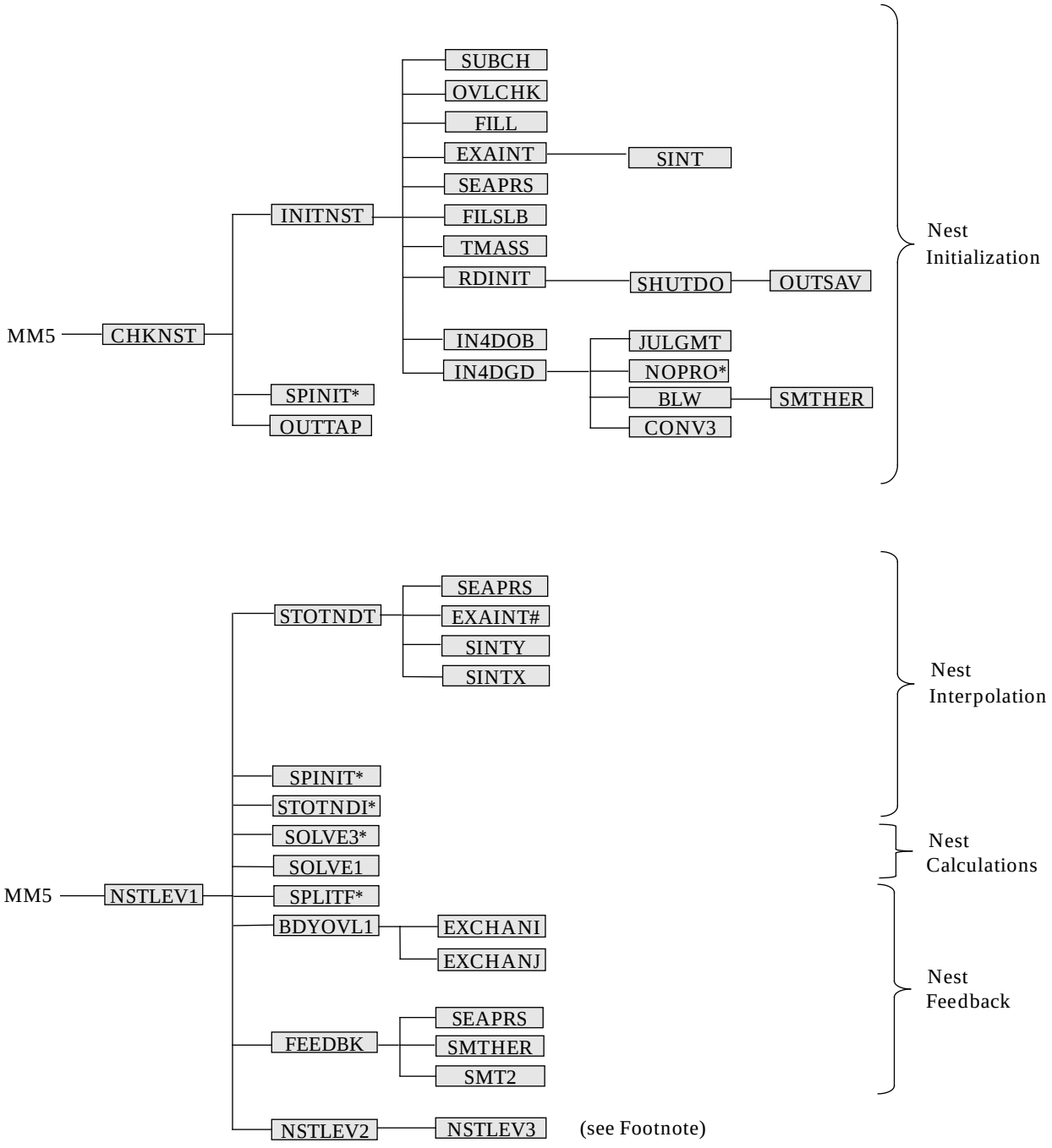

Figure A2. Flow chart for major subroutines CHKNST and NSTLEV1. The expansions of the subroutines appended with an asterisk are shown in Figure A1.

Note: The flow chart structure of subroutines NSTLEV2 and NSTLEV3 is nearly identical to that of NSTLEV1.

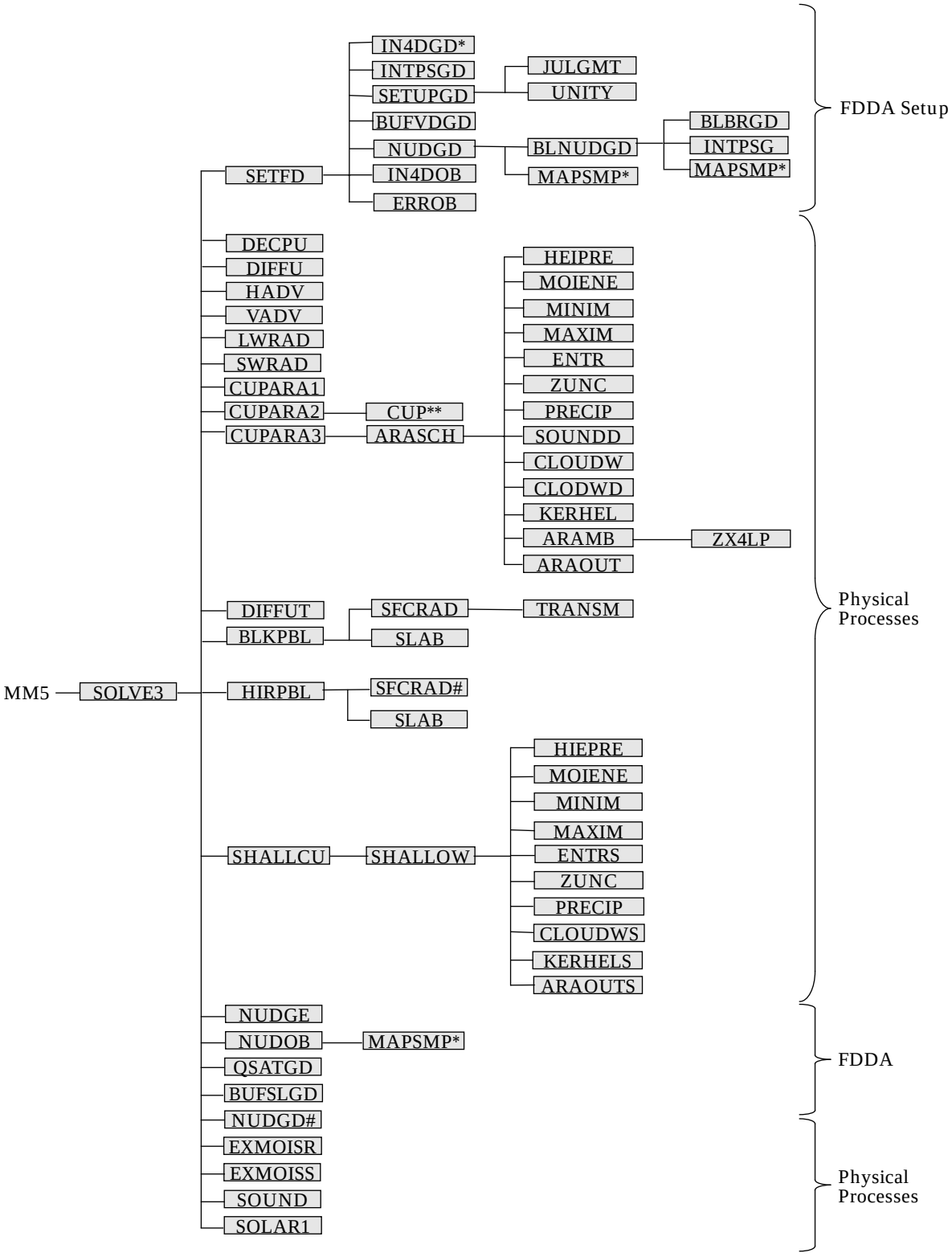

Figure A3. Flow chart for major subroutine SOLVE3. The expansions of the subroutines appended with an asterisk are shown in Figures A1 (for MAPSMP) and A2 (for IN4DGD). The routine appended with a double asterisk calls two subroutines not shown (MAXIMI and MINIMI).

Appendix B:

MM5 Job Deck

This appendix gives an example of the UNICOS script required to submit an MM5 job. It contains 1) UNICOS QSUB directives, 2) input/output Mass Store filenames, 3) UPDATE directives to define the model dimensions and other major PARAMETERs, and 4) the local input file with the NAMELISTs. It also includes the commands for compiling, loading, and running the model.

```
# QSUB -r MM5v1                         # request name
# QSUB -q reg                           # job queue class
# QSUB -eo                              # stdout and stderr together
# QSUB -lM 8Mw                          # maximum memory
# QSUB -lT 5000                         # time limit
# QSUB                                  # no more qsub commands
#
ja
cd $TMPDIR
batchname mm5.out
#
#             ********************************************
#             *******      mm5 batch C shell       *******
#             ********************************************
#
#
#    how many CRAY CPUs to use to run the model, set to 1 if not multitasking
#
setenv NCPUS 1
#
#       this should be the user's case or experiment (used in MS name)
#
  set ExpName = MM5/TEST
  set RetPd   = 365           # MSS retention period in days
#
  set Mods = ( )                # do not change if no user's own mm5 mods
#
# host computer name to rcp user's mm5 mods
  set Host    = username@host.domain:/usr/tmp/username
#
#       type of mm5 job
#
   set recompile = yes         # if yes, recompiled the mm5 code
#   set recompile = no          # if yes, recompiled the mm5 code
#
   set CaseName = CTL      # MSS pathname for this sensitivity test
#
   set STARTsw   = NoReStart   # start model run at hour 0
#  set STARTsw   = ReStart      # restart model run
#
  set FDDAsw    = NoFDDA       # no FDDA input files
#  set FDDAsw    = Anly         # gridded FDDA input files
#   set FDDAsw    = Obs         # obs FDDA input files
#  set FDDAsw    = Both         # gridded and obs FDDA input files
#
#  set HYDROsw   = Hydro        # hydrostatic input files
   set HYDROsw    = NonHydro      # nonhydrostatic input files
#
   set NumDomInp = 1           # number of initial condition input files
   set DomIDInp  = ( 1 )     # domain ID no. for the initial condition inputs
#
   if ( $HYDROsw == NonHydro ) then
     set InitName  =  $ExpName/NH
   else
     set InitName  =  $ExpName/HY
   endif
#
#
#      boundary file
#
```

```
     set InBdy  =  $InitName/BDYOUT_DOMAIN1
#
#      initial conditions
#
     set InMM    = ( $InitName/MMINPUT_DOMAIN1 )
#      set InMM    = ( $InitName/MMINPUT_DOMAIN1 \
#                     $InitName/MMINPUT_DOMAIN2 )
#
#      the input restart file
#
   if ( $STARTsw == ReStart ) then
     set InRst    = ( $InitName/${CaseName}/SAVE_DOMAIN1 \
                    $InitName/${CaseName}/SAVE_DOMAIN2 )
   endif
#
#      4dda surface analyses
#
if (( $FDDAsw == Anly ) || ( $FDDAsw == Both )) then
   set In4DSfc  = ( $ExpName/RW4DDA_DOMAIN1 \
                  $ExpName/RW4DDA_DOMAIN2  )
endif
#
#      4dda observations
#
if (( $FDDAsw == Obs  ) || ( $FDDAsw == Both )) then
   set In4DObs  = ( $ExpName/MM5OBS_T12 )
endif
#
#
#      MSS directory name for history, save, shut down and print out files
#
if ( $STARTsw == ReStart ) then
   set OutMM       = ${InitName}/${CaseName}_RES
else
   set OutMM       = ${InitName}/$CaseName
endif
#
#       update for mm5, data dimensions
#
if ( $recompile == yes ) then
cat >! mm5.mods << EOF
*/
*/      ***************************************************************
*/      ***************************************************************
*/            PARAMETER VALUES TO MODIFY FOR DOMAIN SIZES
*/      ***************************************************************
*/      ***************************************************************
*/
*ID USERDEF
*/
*/  MAXNES          : TOTAL NUMBER OF DOMAINS IN THE FORECAST,
*/                    INCLUDING THE COARSE GRID
*/
*/  NLNES           : NUMBER OF LEVELS OF NEST.  IF THE COARSE GRID
*/                    HAS A FINE GRID, NLNES=2
*/
*/  MIX,MJX,MKX     : MIX (MJX) IS THE MAXIMUM NUMBER OF
*/                    VALUES ON DOT POINTS IN THE Y (X) DIRECTION
*/                    FOR ALL DOMAINS.
*/                    MKX IS THE NUMBER OF HALF SIGMA LAYERS.
```

```
*/
*/
*D PARAME.2,20
C
C
C--- ADDITIONAL MEMORY REQUIREMENTS FOR FDDA RUNS (IFDDA=1),
C--- NONHYDROSTATIC RUNS (INHYD=1), HIGHER ORDER PBL RUNS (INAV=1),
C--- EXPLICIT MOISTURE SCHEME (IEXMS=1), ARAKAWA-SCHUBERT
C--- CONVECTIVE PARAMETERIZATION (IARASC=1),AND ATMOSPHERIC
C--- RADIATION (IRDDIM=1).
C
C
     PARAMETER (IFDDA=0,INHYD=0,INAV=0,IICE=0,
    1          IEXMS=0,IARASC=0,IRDDIM=0)
C
C--- NUMBER OF DOMAINS (MAXNES) AND NESTLEVELS (NLNES)
C
     PARAMETER (MAXNES=1,NLNES=1)
C
C--- MAXIMUM DIMENSIONS OF ANY DOMAIN
C
     PARAMETER (MIX=1,MJX=1,MKX=1)
*/ COMMENT the following line, if running in extended mt class
*D MM5.170,178
EOF
endif
#
#       local namelist values
#
cat >! mmlif << EOF
 &OPARAM                                        ;          <-- MOD2
 ;-----------------------------------------------------------------------
 ;      YOU CAN REMOVE THE UNWANTED DATA FROM THE FOLLOWING LISTING
 ;      AND USE THE DEFAULT VALUES DEFINED IN SUBROUTINE 'PARAM'.
 ;-----------------------------------------------------------------------
 IFREST = F,      ;RESTART
   IXTIMR  =720,
 LEVIDN  = 0,1,1,1,1,1,1,1,1,1, ; level of nest for each domain
 NUMNC   = 1,1,1,1,1,1,1,1,1,1, ; ID of mother domain for each nest
 IFSAVE = T,       ; SAVE DATA FOR RESTART
   SAVFRQ =360.,  ;   ... in minutes.
 IFTAPE = 1,       ; OUTPUT FOR GRIN backend
   TAPFRQ =60.,   ;   ... in minutes.
 IFPRT = 0,        ; 1: print output
 PRTFRQ = 720.,    ; Print output frequency in minutes
 MASCHK = 60,      ; MASS CONSERVATION CHECK (minutes)
 & ;-----------------------------------------------------------------------
 &LPARAM
  iactiv=1,0,0,0,0,0,0,0,0,0, ; in case of restart: was this domain active?
 ;
 ;
 ;************** start physics options *****************
 ;
 IFRAD     = 0 ,   ;RADIATION COOLING OF ATMOSPHERE - 0, 1, 2
 RADFRQ    = 30.,  :RADIATION FREQUENCY IN MINUTES
 ICUSTB    = 0,    ;STABILITY CHECK FOR CUMULUS PARAM. - 0, 1
 IEXICE    = 0,    ;ICE-PHYSICS IN EXPLICIT SCHEME - 0, 1
 IFDRY     = 0,    ;FAKE-DRY RUN - 0, 1
 IMVDIF    = 0,    ;MOIST VERTICAL DIFFUSION IN CLOUDS - 0, 1
 IBMOIST   = 0,    ;BOUNDARY AND INITIAL WATER/ICE SPECIFIED - 0, 1
```

```
ICOR3D    = 0,    ;3D CORIOLIS FORCE (FOR NH RUN ONLY) - 0, 1
IFUPR     = 0,    ;UPPER RADIATIVE BOUNDARY CONDITION (NH) - 0, 1
;
IBOUDY = 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, ;BOUNDARY CONDITIONS
;       (FIXED, RELAXATION, TIME-DEPENDENT, TIME AND INFLOW/OUTFLOW DEPENDENT,
;        SPONGE - 0, 1, 2, 3, 4)
IBLTYP = 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, ;PBL TYPE
;       (FRICTIONLESS, BULK PBL, MULTI-LEVEL PBL - 0, 1, 2)
IDRY   = 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ;MOIST OR DRY CASE - 0, 1
IMOIST = 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ;NON-EXPLICIT,  EXPLICIT, -  1, 2
ICUPA  = 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ;NONE,KUO,GRELL,A-S - 1,2,3,4
ISFFLX = 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ;SURFACE FLUXES - 0, 1
ITGFLG = 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ;SURFACE TEMPERATURE - 1, 3
ISFPAR = 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ;SURFACE CHARACTERISTICS - 0, 1
ICLOUD = 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ;CLOUD EFFECTS ON RADIATION - 0, 1
ICDCON = 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ;CONSTANT DRAG COEFFICIENTS - 0, 1
IFSNOW = 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ;SNOW COVER EFFECTS - 0, 1
IMOIAV = 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ;VARIABLE MOISTURE AVAILABILITY - 0, 1
IVMIXM = 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ;VERTICAL MIXING OF MOMENTUM - 0, 1
HYDPRE = 1.,1.,1.,1.,1.,1.,1.,1.,1.,;HYDRO EFFECTS OF LIQ WATER - 0., 1.
IEVAP  = 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ;EVAP OF CLOUD/RAINWATER - <0, 0, >0
ISHALLO= 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ;SHALLOW CONVECTION - 0, 1
;
;************** end physics options ******************
;
;
;
;************** start nesting options *****************
;
nestix =  46,  46,  46,  46,  46,  46,  46,  46,  46,  46, ; domain size I
nestjx =  61,  61,  61,  61,  61,  61,  61,  61,  61,  61, ; domain size J
nesti  =   1,   1,   1,   1,   1,   1,   1,   1,   1,   1, ; start location I
nestj  =   1,   1,   1,   1,   1,   1,   1,   1,   1,   1, ; start location J
xstnes =   0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,; domain initiation
xennes = 1440.,1440.,1440.,720.,720.,720.,720.,720.,720.,; domain completion
ioverw =   1,   0,   0,   0,   0,   0,   0,   0,   0,   0, ; overwrite domain
;
;************** end nesting options ******************
;
;
;
;************** start moving options ******************
;
imove  =   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, ; move domain 0,1
imovco =   1,   1,   1,   1,   1,   1,   1,   1,   1,   1, ; 1st move #
; imoveI(J,K)=L, ; I-INCREMENT MOVE (DOMAIN J, MOVE NUMBER K) IS L
imovei =   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, ; I move #1
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0, ; I move #2
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0, ; I move #3
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0, ; I move #4
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0, ; I move #5
imovej =   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, ; J move #1
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0, ; J move #2
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0, ; J move #3
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0, ; J move #4
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0, ; J move #5
imovet =   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, ; time of move #1
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0, ; time of move #2
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0, ; time of move #3
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0, ; time of move #4
```

```
        0,   0,   0,   0,   0,   0,   0,   0,   0,   0, ; time of move #5
;
;************** end moving options ******************
;
& ;---------------------------------------------------------------------
&PPARAM
TIMAX  = 1440.,   ; IN MINUTES. 720=12h, 1440=24h, 2160=36h, 2880=48h
ZZLND  = 0.1,     ; ROUGHNESS LENGTH OVER LAND IN METERS
ZZWTR  = 0.0001, ; ROUGHNESS LENGTH OVER WATER IN METERS
ALBLND = 0.15,    ; ALBEDO
THINLD = 0.04,    ; SURFACE THERMAL INERTIA
XMAVA  = 0.3,     ; MOISTURE AVAILABILITY OVER LAND AS A DECIMAL
;                 FRACTION OF ONE
CONF   = 1.0,     ; NON-CONVECTIVE PRECIPITATION SATURATION
TISTEP = 270.,    ; COARSE DOMAIN DT IN MODEL, USE 3*DX
ifeed  = 3,       ; OLD FEEDBACK, NO/LIGHT SMOOTHING IN FEEDBK - 1,2,3
iabsor = 0,       ; SPONGE ON UPPER BOUNDARY (HYD) - 0,1
& ;---------------------------------------------------------------------
&FPARAM
;
;
;       ****************************************************
;       ****************************************************
;
;
;    THE FIRST DIMENSION (COLUMN) IS THE DOMAIN IDENTIFIER:
;       COLUMN 1 = DOMAIN #1, COLUMN 2 = DOMAIN #2, ETC.
;
;      START TIME FOR FDDA (ANALYSIS OR OBS) FOR EACH DOMAIN
;      (IN MINUTES RELATIVE TO MODEL INITIAL TIME)
FDASTA=0.,0.,0.,0.,0.,0.,0.,0.,0.,
;
;      ENDING TIME FOR FDDA (ANALYSIS OR OBS) FOR EACH DOMAIN
;      (IN MINUTES RELATIVE TO MODEL INITIAL TIME)
FDAEND=780.,0.,0.,0.,0.,0.,0.,0.,0.,
;
;
;       ****************************************************
;       ***************** ANALYSIS NUDGING *****************
;       ****************************************************
;
;
;    THE FIRST DIMENSION (COLUMN) OF THE ARRAYS DENOTES THE
;    DOMAIN IDENTIFIER:
;       COLUMN 1 = DOMAIN #1, COLUMN 2 = DOMAIN #2, ETC.
;    THE SECOND DIMENSION (ROW OR LINE) EITHER REFERS TO THE 3D VS
;    SFC ANALYSIS OR WHICH VARIABLE IS ACCESSED:
;       LINE 1 = 3D, LINE 2 = SFC   OR
;       LINE 1 = U, LINE 2 = V, LINE 3 = T, LINE 4 = Q
;
;      IS THIS A GRID 4DDA RUN? 0 = NO; 1 = YES
I4D=0,0,0,0,0,0,0,0,0,0,      ; 3D ANALYSIS NUDGING
   0,0,0,0,0,0,0,0,0,0,      ; SFC ANALYSIS NUDGING
;
;      SPECIFY THE TIME IN MINUTES BETWEEN THE INPUT (USUALLY
;      FROM INTERP) USED FOR GRID FDDA
DIFTIM=720.,720.,0.,0.,0.,0.,0.,0.,0.,      ; 3D ANALYSIS NUDGING
      180.,180.,0.,0.,0.,0.,0.,0.,0.,      ; SFC ANALYSIS NUDGING
;
;      GRID NUDGE THE WIND FIELD? 0 = NO; 1 = YES
```

```
IWIND=1,1,0,0,0,0,0,0,0,0,    ; 3D ANALYSIS NUDGING
     1,1,0,0,0,0,0,0,0,0,    ; SFC ANALYSIS NUDGING
;
;       NUDGING COEFFICIENT FOR WINDS ANALYSES
GV=2.5E-4,1.0E-4,0.,0.,0.,0.,0.,0.,0.,       ; 3D ANALYSIS NUDGING
  2.5e-4,1.0E-4,0.,0.,0.,0.,0.,0.,0.,       ; SFC ANALYSIS NUDGING
;
;       GRID NUDGE THE TEMPERATURE FIELD? 0 = NO; 1 = YES
ITEMP=1,1,0,0,0,0,0,0,0,0,    ; 3D ANALYSIS NUDGING
     1,1,0,0,0,0,0,0,0,0,    ; SFC ANALYSIS NUDGING
;
;       NUDGING COEFFICIENT FOR TEMPERATURE ANALYSES
GT=2.5e-4,1.0E-4,0.,0.,0.,0.,0.,0.,0.,       ; 3D ANALYSIS NUDGING
  2.5e-4,1.0E-4,0.,0.,0.,0.,0.,0.,0.,       ; SFC ANALYSIS NUDGING
;
;       GRID NUDGE THE MIXING RATIO FIELD? = 0 ; NO = 1 ; YES
IMOIS=1,1,0,0,0,0,0,0,0,0,    ; 3D ANALYSIS NUDGING
     1,1,0,0,0,0,0,0,0,0,    ; SFC ANALYSIS NUDGING
;
;       NUDGING COEFFICIENT FOR THE MIXING RATIO ANALYSES
GQ=1.E-5,1.E-5,0.,0.,0.,0.,0.,0.,0.,        ; 3D ANALYSIS NUDGING
  1.E-5,1.E-5,0.,0.,0.,0.,0.,0.,0.,        ; SFC ANALYSIS NUDGING
;
;       GRID NUDGE THE ROTATIONAL WIND FIELD? 0 = NO; 1 = YES
IROT=0,0,0,0,0,0,0,0,0,0,     ; 3D ANALYSIS NUDGING
;
;       NUDGING COEFFICIENT FOR THE ROTATIONAL COMPONENT OF THE WINDS
GR=5.E6,5.E6,0.,0.,0.,0.,0.,0.,0.,          ; 3D ANALYSIS NUDGING
;
;       IF GRID NUDGING (I4D(1,1)=1) AND YOU WISH TO EXCLUDE THE
;       BOUNDARY LAYER FROM FDDA OF COARSE GRID THREE DIMENSIONAL
;       DATA (USUALLY FROM INTERP),
;           0 = NO,  INCLUDE BOUNDARY LAYER NUDGING
;           1 = YES, EXCLUDE BOUNDARY LAYER NUDGING
INONBL =0,0,0,0,0,0,0,0,0,0,         ; U WIND
       0,0,0,0,0,0,0,0,0,0,         ; V WIND
       0,0,0,0,0,0,0,0,0,0,         ; TEMPERATURE
       0,0,0,0,0,0,0,0,0,0,         ; MIXING RATIO
;
;       RADIUS OF INFLUENCE FOR SURFACE ANALYSIS (KM).
;       IF I4D(2,1)=1 OR I4D(2,2)=1, ETC, DEFINE RINBLW (KM) USED
;       IN SUBROUTINE BLW TO DETERMINE THE HORIZONTAL VARIABILITY
;       OF THE SURFACE-ANALYSIS NUDGING AS A FUNCTION OF SURFACE
;       DATA DENSITY.  OVER LAND, THE STRENGTH OF THE SURFACE-
;       ANALYSIS NUDGING IS LINEARLY DECREASED BY 80 PERCENT AT
;       THOSE GRID POINTS GREATER THAN RINBLW FROM AN OBSERVATION
;       TO ACCOUNT FOR DECREASED CONFIDENCE IN THE ANALYSIS
;       IN REGIONS NOT NEAR ANY OBSERVATIONS.
RINBLW=250.,
;
;       SET THE NUDGING PRINT FREQUENCY FOR SELECTED DIAGNOSTIC
;       PRINTS IN THE GRID (ANALYSIS) NUDGING CODE (IN CGM
;       TIMESTEPS)
NPFG=50,
;
;       ******************************************************
;       ********** OBSERVATION NUDGING **********
;       ******************************************************
;
;
```

```
;    INDIVIDUAL OBSERVATION NUDGING.  VARIABLES THAT ARE ARRAYS
;    USE THE FIRST DIMENSION (COLUMN) AS THE DOMAIN IDENTIFIER:
;       COLUMN 1 = DOMAIN #1, COLUMN 2 = DOMAIN #2, ETC.
;
;     IS THIS INDIVIDUAL OBSERVATION NUDGING? 0 = NO; 1 = YES
I4DI   =0,0,0,0,0,0,0,0,0,0,
;
;     OBS NUDGE THE WIND FIELD FROM STATION DATA? 0 = NO; 1 = YES
ISWIND =1,0,0,0,0,0,0,0,0,0,
;
;     NUDGING COEFFICIENT FOR WINDS FROM STATION DATA
GIV  =4.E-4,4.E-4,0.,0.,0.,0.,0.,0.,0.,0.,
;
;     OBS NUDGE THE TEMPERATURE FIELD FROM STATION DATA? 0 = NO; 1 = YES
ISTEMP=1,0,0,0,0,0,0,0,0,0,
;
;     NUDGING COEFFICIENT FOR TEMPERATURES FROM STATION DATA
GIT  =4.E-4,4.E-4,0.,0.,0.,0.,0.,0.,0.,0.,
;
;     OBS NUDGE THE MIXING RATIO FIELD FROM STATION DATA? 0 = NO; 1 = YES
ISMOIS=1,0,0,0,0,0,0,0,0,0,
;
;     NUDGING COEFFICIENT FOR THE MIXING RATIO FROM STATION DATA
GIQ  =4.E-4,4.E-4,0.,0.,0.,0.,0.,0.,0.,0.,
;
;     THE OBS NUDGING RADIUS OF INFLUENCE IN THE
;     HORIZONTAL IN KM FOR CRESSMAN-TYPE DISTANCE-WEIGHTED
;     FUNCTIONS WHICH SPREAD THE OBS-NUDGING CORRECTION
;     IN THE HORIZONTAL.
RINXY=240.,
;
;     THE OBS NUDGING RADIUS OF INFLUENCE IN THE
;     VERTICAL IN SIGMA UNITS FOR CRESSMAN-TYPE DISTANCE-
;     WEIGHTED FUNCTIONS WHICH SPREAD THE OBS-NUDGING
;     CORRECTION IN THE VERTICAL.
RINSIG=0.001,
;
;     THE HALF-PERIOD OF THE TIME WINDOW, IN MINUTES, OVER
;     WHICH AN OBSERVATION WILL AFFECT THE FORECAST VIA OBS
;     NUDGING. THAT IS, THE OBS WILL INFLUENCE THE FORECAST
;     FROM TIMEOBS-TWINDO TO TIMEOBS+TWINDO.  THE TEMPORAL
;     WEIGHTING FUNCTION IS DEFINED SUCH THAT THE OBSERVATION
;     IS APPLIED WITH FULL STRENGTH WITHIN TWINDO/2. MINUTES
;     BEFORE OR AFTER THE OBSERVATION TIME, AND THEN LINEARLY
;     DECREASES TO ZERO TWINDO MINUTES BEFORE OR AFTER THE
;     OBSERVATION TIME.
TWINDO=40.0,
;
;     THE NUDGING PRINT FREQUENCY FOR SELECTED DIAGNOSTIC PRINT
;     IN THE OBS NUDGING CODE (IN CGM TIMESTEPS)
NPFI=20,
;
;     FREQUENCY (IN CGM TIMESTEPS) TO COMPUTE OBS NUDGING WEIGHTS
IONF=2,
& ;---------------------------------------------------------------
EOF
#
#########################################################################
####################                  ######################
####################  END USER MODIFICATION  ######################
```

```
####################                    ######################
###############################################################################
#
#       this is INTERACTIVE or BATCH
#
if ( $?ENVIRONMENT ) then
  echo "environment variable defined as $ENVIRONMENT"
else
  setenv ENVIRONMENT INTERACTIVE
  echo "environment variable defined as $ENVIRONMENT"
endif
#
#       initializations, no user modification required
#
set LETTERS = (A B C D E F G H I J K L M N O P Q R S T U V W X Y Z)
set MesoUser = /u1/mesouser/MM5V1/MM5
#
#       get boundary conditions from MS
#
msread bdyout $InBdy
#
#       loop over how many files of domains to acquire
#
set NUM = 0
while ( $NUM < $NumDomInp )
  @ NUM ++
#
#       initial conditions
#
  msread fort.1$DomIDInp[$NUM] $InMM[$NUM]
#
#       input restart conditions
#
  if ( $STARTsw == ReStart ) then
    msread fort.9$DomIDInp[$NUM] $InRst[$NUM]
  endif
#
#       get analyses for nudging
#
  if (( $FDDAsw == Anly ) || ( $FDDAsw == Both )) then
    cp fort.1$DomIDInp[$NUM] fort.3$DomIDInp[$NUM]
    msread fort.7$DomIDInp[$NUM] $In4DSfc[$NUM]
    cp fort.7$DomIDInp[$NUM] fort.8$DomIDInp[$NUM]
  endif
#
#       observations if OBS nudging
#
  if (( $FDDAsw == Obs ) || ( $FDDAsw == Both )) then
    msread fort.6$DomIDInp[$NUM] $In4DObs[$NUM]
  endif
end

#
#       set up fortran input files for MM5
#
if ( -e assign.mm5 ) rm assign.mm5
setenv FILENV assign.mm5
assign -a mmlif                 fort.7
assign -a ${MesoUser}/ehtran    fort.8
assign -a bdyout                fort.9
```

```
if ( $recompile == yes ) then
#
#       mm5 mod decks to include in update
#
  set NUM = 0
  while ( $NUM < ${#Mods} )
    @ NUM ++
    echo "using $Mods[$NUM] mod deck"
    rcp $Host/${Mods[$NUM]} .
    cat $Mods[$NUM] >> mm5.mods
    rm $Mods[$NUM]
  end

  if ( -e m5_stand.mods ) then
    echo "using local copy of m5_stand.mods"
    cat m5_stand.mods >> mm5.mods
  else if ( -e ~/m5_stand.mods ) then
    echo "using MY copy of m5_stand.mods"
    cat ~/m5_stand.mods >> mm5.mods
  else if ( -e ${MesoUser}/m5_stand.mods ) then
    echo "using standard copy of m5_stand.mods"
    cat ${MesoUser}/m5_stand.mods >> mm5.mods
  else
    echo "not using any copy of m5_stand.mods"
  endif
#
  if ( -e m5_my.mods ) then
    echo "using local copy of m5_my.mods"
    cat m5_my.mods >> mm5.mods
  else if ( -e ~/m5_my.mods ) then
    echo "using MY copy of m5_my.mods"
    cat ~/m5_my.mods >> mm5.mods
  else
    echo "not using any copy of m5_my.mods"
  endif
#
#       make an MM5 source code
#
  if ( ! -e mm5.v1 ) then
    echo "acquiring the mm5.v1 source code"
    cp ${MesoUser}/mm5v1.s mm5.v1
  endif
  nupdate -i mm5.v1 -n mmzin
  nupdate -p mmzin -m 1 -f -i mm5.mods -o "id sq" -c mm5
  set toast = $status
  if      ( $toast != 0 ) then
    echo "error in the update, stopping"
    exit(1)
  else if ( $ENVIRONMENT != BATCH ) then
    echo -n "update complete, continue?     (yes)   "
    set ans = "$<"
    if (( $ans == "n" ) || ( $ans == "no" )) then
      exit (1)
    endif
  endif
#
#       compile MM5
#
  cf77 -c -Wf"-o aggress" mm5.f
  set toast = $status
```

```
  if      ( $toast != 0 ) then
    echo "error in the compile, stopping"
    exit(1)
  else if ( $ENVIRONMENT != BATCH ) then
    echo -n "compile complete, continue?    (yes)   "
    set ans = "$<"
    if (( $ans == "n" ) || ( $ans == "no" )) then
      exit (1)
    endif
  endif
#
#       load MM5
#
  segldr -L /lib,/usr/lib,/usr/local/lib -l etime,imslcnv,imsl -o mm5.exe mm5.o
  set toast = $status
  if      ( $toast != 0 ) then
    echo "error in the segldr, stopping"
    exit(1)
  else if ( $ENVIRONMENT != BATCH ) then
    echo -n "segldr complete, continue?     (yes)   "
    set ans = "$<"
    if (( $ans == "n" ) || ( $ans == "no" )) then
      exit (1)
    endif
  endif
endif

if ( $recompile == no ) then
  msread mm5.exe ${InitName}/${CaseName}/mm5.exe
  chmod +x mm5.exe
else
  mswrite mm5.exe ${InitName}/${CaseName}/mm5.exe
endif

execute:
ls -l
#
#       run MM5
#
date
mm5.exe >&! mm5.print.out
set toast = $status
if ( $toast != 0 ) then
  echo "error in the forecast, stopping"
  debug -s mm5.exe
  if ( $ENVIRONMENT != BATCH ) exit(1)
endif
#
#       if interactive, probably do not want to dispose files
#
if ( $ENVIRONMENT != BATCH ) then
  echo -n "test mm5 run complete, continue?     (no)    "
  set ans = "$<"
  if (( $ans != "y" ) && ( $ans != "yes" )) then
    exit (0)
  endif
endif
#
#       print and save the print output
#
```

```
ja -chls >! acct
cat acct >> mm5.print.out
if ( $ENVIRONMENT == BATCH ) cat mm5.print.out
#
#       history output   41-49
#       save file output 51-59
#       shutdown output  61-69
#
ls -ls
set Tens  = (   4     5     6   )
set Name  = ( MMOUT SAVE SHUTDO )
foreach OutType ( 1 2 3 )
  set OutFileType = $Name[$OutType]
  foreach Units ( 1 2 3 4 5 6 7 8 9 )
    if ( -e fort.$Tens[$OutType]${Units} ) then
      echo ls -l fort.$Tens[$OutType]${Units} >! hold
      set test = ‘source hold‘
      if ( $test[4] < 400000000 ) then
        mswrite -t $RetPd fort.$Tens[$OutType]$Units \
              $OutMM/${OutFileType}_DOMAIN${Units}
      else
       if ($Tens[$OutType] == 4) then
        if ( ! -e split.deck ) then
        cp ${MesoUser}/split.deck .
        chmod +x split.deck
        endif
        mv fort.$Tens[$OutType]${Units} ${OutFileType}_DOMAIN${Units}
        split.deck ${OutFileType}_DOMAIN${Units}

        set Numb = 0
        foreach fil ( ‘ls mmtmp*‘ )
        @ Numb ++
        echo "mswrite $fil \
             $OutMM/${OutFileType}_DOMAIN${Units}_$LETTERS[$Numb] "
        mswrite -t $RetPd $fil \
             $OutMM/${OutFileType}_DOMAIN${Units}_$LETTERS[$Numb]
        end

        rm mmtmp*
       else

        bsplit -400000000 fort.$Tens[$OutType]${Units} \
              small.$Tens[$OutType]${Units}.
        set NUM = 1
        echo ls small.$Tens[$OutType]${Units}.* >! hold
        set test = ‘source hold‘
        foreach split ( $test )
          mswrite -t $RetPd $split \
            $OutMM/${OutFileType}_DOMAIN${Units}_BSPLIT_$LETTERS[$NUM]
          @ NUM ++
        end
       endif

      endif
    endif
  end
end
#
# tar the namelist, mods, source code, executable, and output together
# save the MM5 tar file on MSS
```

```
#
tar -cvf mm5.tar mm5.f mmlif mm5.mods mm5.print.out
echo " mswrite -t $RetPd mm5.tar $ExpName/mm5.tar "
mswrite -t $RetPd mm5.tar $OutMM/mm5.tar
ls -ls
```

Appendix C:

Unit Number Allocations

Model input and output of various types are assigned to FORTRAN logical units as follows. Where the unit number has two digits, the second digit always refers to the domain number. All units are binary unless ASCII is specified. Units designated (I) are input, but are only used internally. Units 1-6 and those ending in zero are not used.

UNIT	I/O	DESCRIPTION
7	I	mmlif: model input namelist file (ASCII)
8	I	ehtran: look-up table for transmissivities
9	I	bdyout: coarse-mesh boundary conditions
11-19	I	Initial conditions and FDDA analyses
21-29		Not used
31-39	(I)	Analyses for FDDA (dummy file copied from 11-19)
41-49	O	Model output file for GRIN
51-59	O	Save-file output for restarts
61-69	I	Individual observations for FDDA
61-69	O	Shut-down save-file output for restarts
71-79	I	Surface analyses for FDDA
81-89	(I)	Surface analyses for FDDA (dummy file copied from 71-79)
91-99	I	Input restart file

References

Anthes, R. A., E.-Y. Hsie and Y.-H Kuo, 1987: Description of the Penn State/ NCAR mesoscale model version 4 (MM4). NCAR Technical Note, NCAR/TN-282+ STR, 66 pp. [Available from NCAR Information Services, P.O. Box 3000, Boulder CO, 80307.]

Gill, D. O., 1992: A user's guide to the Penn State/NCAR mesoscale modeling system. NCAR Technical Note, NCAR/TN-381+IA, 212 pp. [Available from NCAR Information Services, P.O. Box 3000, Boulder CO 80307.]

Grell, G. A., J. Dudhia and D. R. Stauffer, 1993: A description of the fifth-generation Penn State/ NCAR mesoscale model (MM5). NCAR Technical Note, NCAR/TN-398+ STR, 117 pp. [Available from NCAR Information Services, P.O. Box 3000, Boulder, CO 80307.]

Hsie, E.-Y., 1987: MM4 (Penn State/NCAR) mesoscale model version 4 documentation. NCAR Technical Note, NCAR/TN-293+STR, 209 pp. [Available from NCAR Information Services, P.O. Box 3000, Boulder CO 80307.]