CMAQv5.3 Developer Guide

1 Developers' Guide for the Community Multiscale Air Quality (CMAQ) Modeling System

Consistent with CMAQ model version 5.3 (2019 Release)

Prepared in cooperation with the:
Community Modeling and Analysis System
Institute for the Environment
University of North Carolina at Chapel Hill
Chapel Hill, NC

1.1 Disclaimer

The information in this Developer Guide has been funded wholly or in part by the United States Environmental Protection Agency. The draft version of this document has not been subjected to the Agency's peer and administrative review, nor has it been approved for publication as an EPA document. The draft document is currently being edited and reviewed by the Community Modeling and Analysis System Center. Mention of trade names or commercial products does not constitute endorsement or recommendation for use.

2 Motivation

The evolution and robustness of CMAQ depends on contributions from the vibrant CMAS community. The development team at EPA is excited to work with potential contributors and integrate community submissions into the CMAQ code base. In order to facilitate this process, we describe in this document our development process and how external developers may submit new code features.
The information contained here should be read prior to starting a project within the CMAQ framework. Instructions are tailored for external developers but can also be used by developers in the CMAS-Center or within EPA.

3 Summary of Developer Workflow

The public CMAQ release repository is located on GitHub (https://github.com/USEPA/CMAQ). Users should refer to this repository for bug fixes, issues, documentation and major releases for CMAQ. Users can use the watch and star buttons on the public CMAQ release repository page to be notified of updates and changes. Developers interested in submitting code changes should read this Developer Guide and then contact the EPA CMAQ development team as soon as possible to discuss their motivation and plans for submitting a code change (CMAQ_Team@epa.gov).

In order to facilitate incorporation of a contribution, developers should follow the instructions on code requirements and repository layout as described in the code management instructions. Documentation of

the assumptions and results of the new code is a very important part of a meaningful code submission. If the submission involves a detailed new feature, developers are encouraged to publish the use of their feature in a peer-reviewed journal before submission.

To begin, the developer should fork the public CMAQ release repository within GitHub. This will create a copy of the public CMAQ release repository under your name (https://github.com/{user_name}/CMAQ). Developers should use standard git commands to clone the appropriate version branch (5.3, 5.2.1, 5.2, ..) from your forked repository to your local machine and then to create a new feature or bug fix branch. Developers will add, commit and push changes to their new feature or bug fix branch on their forked repository, not to the public release version of the repository.

Once a feature or bug fix branch meets requirements for code consistency, benchmark testing, model output evaluation, and documentation including release notes, the developer may submit a pull request from their local feature or bug fix branch of their fork of the CMAQ repository on Github to the CMAQ public repository. This process is described in the Nuts and Bolts section below, and in the following tutorial, which also provides instructions on how to keep a fork up to date with changes on the public release repository.

Contributions will undergo a thorough code review within EPA before being incorporated in the next model release. Depending on the size, scope, and importance of the contribution, the CMAQ development team may or may not agree to support the update through future releases. Decisions regarding ongoing support will be made on a case-by-case basis with input from the developer who submits the contribution. The following sections outline the CMAQ code development and review process in greater detail.

4 Development Life-cycle

4.1 Public Release Versions

CMAQ uses a number versioning system for each release version branch, with major and minor increments. For example, in the case of hypothetical version 14.0 the first number (major version) and second number (minor version) refer to a stable release version. The minor version (second number) of CMAQ increments when one or many new science developments have been adopted. Although these changes may significantly affect model results, the model will still be generally compatible with inputs developed for versions of the same major number. The major version (first number) of CMAQ increments when significant development changes to the code base have been adopted such that backward compatibility or comparability is no longer expected. Modifications to the publically released version without increment are prohibited in order to ensure consistency among published literature referring to a particular model version. In between published releases the development team may publish solutions to model bugs and issues in the public repository under the folder DOCS/Known_Issues. The README located in this folder describes existing known issues, their scope and impact, and how they may be solved.

4.2 Development Versions

Prior to the public release of each major CMAQ version, the unofficial source code is released to the public as a development version that is not intended for regulatory or research application use. The purpose of releasing the development version to the public is to give community members: - a reasonable amount of time to complete any pending feature submissions they would like to submit for the stable release. - a role in helping to test, troubleshoot, and debug the unofficial code before the stable release. - an opportunity to comment on the code improvements made in the new version. - the ability to take advantage of improvements for preliminary studies of their own interest. - a reasonable amount of time to ensure the new version is compatible with any features the member may have submitted in the past. The unofficial (or ***beta***) version of the code will first be vetted internally and then released generally 6 months in advance of the corresponding stable CMAQ release; this period is known as the *beta-phase*. At this time, EPA will announce the deadline for community contributions. This deadline will be chosen in order to balance both the time needed by

developers to submit their contributions and the time needed by EPA to incorporate submissions before public release. Version numbering for the beta series will append the letter 'b' and an incrementing number to the expected version number of the stable release. The number of beta versions is variable among releases. For example, before the hypothetical release of CMAQv14.3 the following series of version numbers could be expected:

```
v14.3.b0  (First tested internal EPA version)
v14.3.b1  (Release to public after minor changes)
v14.3.b2  (....incremental testing, ....)
v14.3.b3  (....bug squashing, and  ...)
v14.3.b4  (....documentation updates...)
v14.3     (Stable Public Release)
```

As stated previously, the "Known Issues" section of the documentation will be continually updated as problems are identified in the released code-base. These updates will not be implemented in the default model code and so the version numbering will in general, not increment between public releases. The instrumented versions of the code (e.g. DDM, ISAM, STM, etc) should be released with the stable version.

5 Making Contributions

5.1 Get in touch

Community members with an idea for a code contribution are encouraged to contact the EPA development team well before the *beta-phase* in order to plan appropriately for the testing and inclusion of the contribution. The EPA team may be interested in knowing information including but not limited to the following: - What science module or bug do you intend to address? What work do you intend to contribute to CMAQ? - Are you comfortable with the development strategy including code consistency, benchmarking, configuration testing, compiler testing, model output validation, documentation and merging? - Are you able to provide ongoing support and technical guidance for your proposed contribution?

5.2 Nuts and Bolts

As described above, the CMAQ development process follows a "Forking Workflow." Atlassian has provided a helpful explanation. Developers should follow the guidance at GitHub Help and Atlassian in order to:
- fork the CMAQ repo: https://help.github.com/articles/fork-a-repo/#platform-linux
- clone their newly-created fork: https://help.github.com/articles/cloning-a-repository/#platform-linux
- create a feature branch: https://www.atlassian.com/git/tutorials/using-branches
- add and commit changes to the new feature branch: https://www.atlassian.com/git/tutorials/saving-changes
- push the feature branch to the forked repo: https://help.github.com/articles/pushing-to-a-remote/
- submit a pull request to the public CMAQ repo: https://help.github.com/articles/creating-a-pull-request-from-a-fork/

Developers should run and test their contribution before submitting the pull request so that the results of the test can be included in the documentation of the pull request.

5.3 Code Review

CMAQ Developers at EPA will review all code submissions in order to ensure code stability and consistency, and prevent degradation of model performance. After review, the EPA team will either accept the submission, recommend specific improvements to the submission, or in some cases reject the submission. To avoid outright

rejection, we urge developers to contact the EPA team early in the development process and maintain contact throughout to help ensure the submission is compatible with the CMAQ code base and is a robust addition.

5.3.1 Code Consistency

Please refer to the code management instructions. Examples of small, but important guidelines include: - Eliminate global memory references (across modules). In other words, no common blocks across modules, no hidden data paths, and no "back doors." - All subroutines should be named in a manner which prevents namespace conflicts. - In general, variable names should be self-descriptive (e.g. NCELLS rather than N). - Use the Fortran declaration IMPLICIT NONE to maintain some control on typographic errors and undefined variables. The use of IMPLICIT NONE forces the developer to declare all internal variables. This is standard in Fortran 90. - In general, it is expected that MKS units are used for input and output variables, as these units have been standardized throughout the CMAQ system. If you use alternative units, please document this exhaustively.

5.3.2 Benchmark Testing

Dataset: The U.S. EPA Southeast US 12km domain July 1-14, 2011 testing dataset is provided with the CMAQv5.3 Release. This dataset is distributed for benchmarking and testing the model installation. It is available from CMAS; please go to https://www.epa.gov/cmaq/cmaq-inputs-and-test-case-data for instructions on how to download the test dataset.

Before making code changes, developers should test multiple compilers (if they have access to them; see the following section on **Compiler Tests**), multiple processor configurations, and single processor configuration runs for a single simulation day to verify their results match the previous stable release, and/or that their results are computationally and physically reasonable. After implementing their code changes, developers should repeat these tests and share the results as part of the pull request documentation.

5.3.2.1 Compiler Tests

Compiler tests use the default benchmark configuration with different compilers and MPI configurations. It is important for the user community that CMAQ always compile with Intel Fortran, Gnu Fortran and Portland Group Fortran compilers. If a developer has access to more than one compiler, it is critical that they test all of them. Some errors will cause different behaviors depending on the choice of compiler and may not be detectable with all of the compilers. See appendix 1 for an example of a Compiler Test.

5.3.2.2 Model Performance Tests

Configuration tests use one compiler to test the impact of a model change on results. See appendix 2 for an example of important information to collect when testing science options. The developer should consider submitting similar information with their pull request.

Several tools exist to document the effects of compiler choice and code change on model results. Examples include: **m3diff -** Quantify min, max, mean differences between two different model runs **VERDI -** Create absolute difference plots for multiple variables, timesteps, layers (see spatial differences)
In addition, we recommend utilizing **1:1 Scatter Plots** to demonstrate the differences between two model runs in a concise layout.

5.3.3 Documentation Requirements

Documentation is of course an integral part of the integration of any contribution into the CMAQ code base. The following documentation products are helpful for expediting the review and integration process: - A Release Note written by the developer which describes the motivation, algorithm and impacts of the

contribution is required to ensure proper documentation of CMAQ.
- If the contribution is a new feature, developers are encouraged to publish its use in a peer-reviewed journal before submitting it to the CMAQ Public Repository.

CMAQ Documentation Resources:
Documentation for CMAQv5.3 is available at https://github.com/USEPA/CMAQ/tree/master/DOCS. Materials include: - User Guide which describes code structure and regular operation of the model. - Release Notes describing code improvements relevant for this model release. - Tutorials that give specific instructions for common tasks like running CMAQ or adding chemical tracers.

5.4 Ongoing Support

Depending on the size, scope, and importance of the contribution, the CMAQ development team may or may not have the resources to support it through future releases. For example, bug fixes and minor, but helpful, changes to the existing code will likely be incorporated into the general code base and supported. Large code additions, like a new process module or an instrumented version of CMAQ may require more effort to support than can be provided by resources of the EPA Office of Research and Development. However, if the feature is particularly of interest for the CMAQ user community, it may be supported. Decisions regarding ongoing support will be made on a case-by-case basis.

6 Copyright Information

Contact EPA (CMAQ_Team@epa.gov) with questions and concerns.

7 Appendix

7.1 Appendix 1: Compiler Tests

Compiler flags:

- PGI: -Mfixed -O3 -Mextend
- GCC: -ffixed-form -ffixed-line-length-132 -O3 -funroll-loops -finit-character=32
- Intel: -fixed -132 -O3 -override-limits -fno-alias -mp1 -fp-model precise -fp-model source -shared-intel -openmp

7.1.1 Compilation Testing Manifest Table (Example)

Scenario	**Compiler**	**netCDF**	**I/O API**	**MPI YN (#P)**	**MPI**	**CMAQv5.1 Timing (hh:mm:ss)**	**CMAQv5.2 Timing (hh:mm:ss)**	Notes
Gfortran Serial	Gfort version 4.8.1	4.3.3	3.1	N	N/A	8:19:51	7:35:30	UNC module gcc/4.8.1
Gfortran mvapich	Gfort version 4.8.1	4.3.2	3.1	Y (16)	mvapich2 1.7	0:45:55	0:42:40	

Scenario	Compiler	netCDF	I/O API	MPI YN (#P)	MPI	CMAQv5.1 Timing (hh:mm:ss)	CMAQv5.2 Timing (hh:mm:ss)	Notes
Intel Serial	Intel Fortran version 16.2.0	4.3.2	3.1	N	N/A	6:01:42	5:10:16	UNC module intel/16.2
Intel Open-MPI (EPA Config)	Intel Fortran v15.0.0	4.3.2	3.1	Y (16)	openMPI 1.42	0:34:27		UNC module openmpi_intel/15.0
Intel OpenMPI	Intel Fortran v16.2.0	4.3.2	3.1	Y (16)	openMPI 1.4.2	0:35:29		UNC module openmpi_intel/16.2
Intel mvapich2	Intel Fortran v16.2.0	4.3.2	3.1	Y (16)	mvapich2 1.7	0:36:34		UNC module mvapich2_intel/16.2
Portland Serial	PGI Fortran v16.1	4.3.2	3.1	N	N/A	7:33:36	6:26:31	UNC module pgi/16.1
Portland OpenMPI	PGI Fortran v15.7	4.3.2	3.1	Y (16)	openMPI 1.4.2	0:40:20	0:36:16	UNC module openmpi_pgi/15.7

7.2 Appendix 2: Model Performance Test Metadata

Scenario	Description	Mechanism	Notes	Timing (16PE) hh:mm:ss
Benchmark Case	Online emissions processing, inline photolysis, inline lightning from MCIP RC, no windblown dust, surface HONO, bidirectional NH3 and Hg, no potential vorticity scaling	cb05e51_ae6_aq	Done; LTNGNO InLine, LTNGPARM = N, LOG_START = 2.0	0:40:20
MOSAIC	Benchmark case with MOSAIC and additional stomatal flux files activated	cb05e51_ae6_aq	Done. set CTM_MOSAIC = Y; set CTM_FST = Y	0:44:02
Dust	Benchmark case with dust, including new MODIS FP input	cb05e51_ae6_aq	Done. setenv CTM_WB_DUST Y; setenv CTM_ERODE_AGLAND Y; setenv CTM_WBDUST_BELD BELD3	0:38:28

Scenario	Description	Mechanism	Notes	Timing (16PE) hh:mm:ss
Hourly NLDN	Benchmark with lightning NOx calculated using hourly bNLDN strikes	cb05e51_ae6_aq	Done; LTNGNO InLine, LTNGPARM = Y, USE_NLDN Y	0:40:18
POA Sensitivity	Benchmark with new POA mechanism	cb05e51_ae6nvPOA_aq	Done	0:34:42

8 Code Management and Development

As a public domain model, CMAQ is the product of contributions from many developers, whose numbers are only expected to increase with the number of users worldwide. Some degree of standardization is necessary for management and archiving of these development versions, as well as to compile and execute the code once it is ready for use, and to submit it to the CMAS Center for archiving and benchmark testing. This chapter provides guidance on source code management, coding guidelines for new code development, the compilation of new source code using the build scripts, and guidelines for writing shell scripts usable by CMAQ. Much of this information is derived from Chapter 18 (Young, 1999) in Byun and Ching (1999), with updates where appropriate, particularly for new versions of the model code and for the Fortran 90 standard. The chapter also includes the procedure that is in place for distributing code versions other than the operational CMAQ that are submitted to the development code archives.

8.1 Source Code Management

8.1.1 The need for a configuration-management tool

Faced with a large and growing community that uses and develops a wide variety of programs, modules, and codes, it is imperative to systematically manage the cross-community access to this software. Typically, successful management of software involves the following:

- A repository – a place where all of the public code resides.
- The concept of archived code – codes that have been deposited into the repository in such a manner that anyone can extract the exact code at a later time. This involves some kind of transformation program to maintain master copies of the codes with embedded change tables.
- The concept of revision control – archiving codes results in modifying the tags or unique revision identifiers in the change tables in the master copies in order to recover the exact code at a later date.
- The concept of released code – codes that have reached some state of maturity and have been designated with some kind of "released" status. They can be used with reasonable expectation of reliability. The paradigm used employs the following scenario:
    1. A user modifies or develops code. The code may be one subroutine or many, possibly constituting whole science modules. The code may originate from "scratch," or be extracted from the repository and modified.
    2. After testing or reaching a point of being satisfied with his/her results, he/she decides to save it in the repository so that others can have access to it.
    3. Some archived codes may still be in an experimental, or development, state, while others may be reasonably stable and more completely tested. The latter may be designated as "released." There is no enforceable means to control access based on an experimental or released state. The community will have, and should have, access indiscriminately, well aware that using development-state code is risky.

4. As the user continues to work with the codes, he/she may make enhancements or discover and fix errors. The upgrades are then installed in the repository, which automatically assigns unique revision identifiers.
5. The repository is located where it is conveniently accessible to all users, and is maintained by an administrator who sets and enforces general access rules.

8.1.2 Choice of a configuration-management tool

Prior to CMAQ version 5.0.2, CMAQ developers used CVS for versioning, and distributed tarballs included CVS artifacts (e.g., files with names ending with ',v'). Starting with version 5.0.2, CMAQ developers switched to git.

8.1.3 git Explained

git is a version control system that supports distributed workflows. Every Git directory is a full repository with complete history and version tracking.

- It works on virtually all UNIX and Linux platforms and on many PCs.
- It is publicly available and free and is distributed under the terms of the GNU General Public License.
- If you would like to contribute changes to the EPA CMAQ repository, use the following steps
  1. Create a github account https://github.com/
  2. Go to the EPA github site and Fork your own copy of the EPA CMAQ to your github account
  3. create a directory called CMAQv5.2.1 on the machine where you would like to obtain a copy of the code
  4. `git clone -b 5.2.1 https://github.com/<your github name>/CMAQ.git CMAQv5.2.1` - Get a clone or copy of the 5.2.1 branch of the CMAQ repository from your github site.
  5. This will place a copy of the files from the 5.2.1 Branch into the CMAQv5.2.1 directory
  6. `cd CMAQv5.2.1` go into the CMAQv5.2.1 directory
  7. `git status` To confirm the status of the files in the repository and the branch that is currently checked out
  8. `git checkout -b 5.2.1_update` To copy the 5.2.1 branch into a new branch called 5.2.1_update
  9. To edit the config_cmaq.csh file take the following steps: `vi config_cmaq.csh` - or use the Atom, TextWrangler or other Editor
  10. To see what changes you made use the following command `git diff config_cmaq.csh`
  11. To stage the change use the following command. `git add config_cmaq.csh`
  12. To commit changes to the local repostitory use the command: `git commit -m "changed config_cmaq.csh to fix issue X"`
  13. To commit changes to your Github repository on the branch 5.2.1_update use the command: `git push`
  14. If you get a message that the push was rejected similar to the following:

```
! [rejected]        5.2.1_update -> 5.2.1_update (fetch first)
  error: failed to push some refs to 'https://github.com/CEMPD/CMAQ.git'
  hint: Updates were rejected because the remote contains work that you do
  hint: not have locally. This is usually caused by another repository pushing
  hint: to the same ref. You may want to first integrate the remote changes
  hint: (e.g., 'git pull ...') before pushing again.
 hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

  15. This means the files have been changed on your Github repository since you last did a clone. Use the following command to get the changes that have been made to the remote git repository: `git pull`
  16. You will be asked to merge the files if there are no changes that conflict with your file changes. IF successful you will see a message similar to the following, that indicates what files were changed. `Merge made by the 'recursive' strategy.`

```
config_cmaq.csh | 4 ++--
1 file changed, 2 insertions(+), 2 deletions(-)
```

17. Retry the push command to place the changes that you committed to the local repository on your Github repository: `git push`
18. Go to the fork of the EPA CMAQ on your github page and submit a pull request to ask that the changes that you have made be incorporated into the EPA github site.

8.2 Guidelines for Developing New CMAQ Source Code

8.2.1 Object-oriented concepts

To make the CMAQ system robust and flexible, object-oriented concepts were incorporated into the design of the system. The incorporation of these ideas helps developers avoid introducing errors when code modifications are needed. Additionally, the system can easily and efficiently be modified, allowing the user to quickly create models for different applications. The implementation language for CMAQ is Fortran 90, which imposes limits on how far one can go in terms of object-oriented design. In particular, because Fortran is a static language, objects cannot be instantiated dynamically; they must be declared explicitly in the source code to be created at compile time. However, to encourage a user community that will be contributing code for future enhancements, every attempt has been made to adhere to the Fortran 90 standard.

8.2.2 Global name data table

To implement modularity and data independence, we have employed design ideas that draw heavily from the object-oriented concept of ''inheritance"and code re-use. The data structures in the codes that deal with the chemical mechanism, I/O API, logical file names, general constants, and pointers are determined by Fortran declarations in data and parameter statements in the CMAQ system. These data structures pertain to a particular application and are meant to apply globally—not just to one particular CCTM through all its subroutines, but also to all the models that supply data to CCTM for that application. These data structures are contained in Fortran INCLUDE files, which are essentially header files, included in the declaration sections near the top of the Fortran code source files. The inclusion of these source files is made automatic by using a generic string that represents the INCLUDE file and that is parsed and expanded to the actual INCLUDE file during a preprocessing stage in the compilation. The Fortran global INCLUDE files contain name tables that define:

1. The chemical mechanism;
2. The I/O API interface, including logical file names;
3. The global modeling constants; and
4. Other constants or parameters that apply across the model.

To effect the implementation of the INCLUDE files into the code, a special compiling system, Bldmake, was developed (Fine et al., 1998), which reads a configuration file that, based on the application, completely determines the model executable to be built. The ASCII configuration file can be generated either by the CMAQ system or by the users following a few, simple syntactical rules. In addition to the global INCLUDE files, the configuration file contains module commands that tell Bldmake to extract the codes for that module from the model code repository for compilation.

8.2.3 Thin Interface

As mentioned in Chapter 4, CMAQ is designed to be robust and flexible with respect to the interchange of modules and the elimination of cross-module data dependencies. Consequently, the concept of a "thin interface" has been employed in the design, which applies principally to the class-drivers (i.e. the top level call to a science module). At a minimum, the thin interface implementation implies the following requirements:

- Eliminate global memory references (across modules). This implies no common blocks across modules, no hidden data paths, and no "back doors."
- Each module reads and interpolates its required data independently. The I/O API helps to ensure this kind of data independence.
- Standardized argument list (CGRID, Date, Time, TimeStep) for calling the class-driver. See the example in Section 9.2.6. These requirements attempt to incorporate the object-oriented idea of encapsulation in the CMAQ design. Rumbaugh et al. (1991) suggest that "Encapsulation (also information hiding) consists of separating the external aspects of an object, which are accessible to other objects, from the internal implementation details of the object, which are hidden from other objects. Encapsulation prevents a program from becoming so interdependent that a small change has massive ripple effects. The implementation" "of an object can be changed without affecting the applications that use it."

The encapsulation design makes the CMAQ system safer and enables the transaction processing, plug-and-play capability. This design also makes it easier for a user to trace data and usage within a module, particularly at the class-driver level.

8.2.4 Coding guidelines

To maintain the object-oriented concepts implemented in the CMAQ system design, we have established a small set of coding guidelines that apply to those who develop CMAQ science modules and affect the low-level design of the models. We have developed standards to control data dependencies at the class-driver level, but we have not propagated these coding standards to the submodule level.

1. The models are generally coded in Fortran (both Fortran 90 and Fortran 77 conventions are used by various developers). It is possible to link in subroutines written in the C language, although this has not been done within the current CMAQ implementation. While the Fortran 90 compiler will compile Fortran 77 code, the reverse is not true. Thus the Makefiles are set up to invoke the Fortran 90 compiler.
2. To enable code compatibility between the Fortran 77 compiler and Fortran 90 code, the following guidance is provided: Line length beyond 72 characters is permissible in Fortran 90 (with line continuation indicated by an ending '&'), but not in Fortran 77; therefore, insertion of the '&' in column 73 of the first line and in column 6 of the next line of the Fortran 90 code will ensure compatibility with both compilers (the '&' at the beginning of a line is "in principle" ignored by the Fortran 90 compiler, but interpreted as a continuation character by the Fortran 77 compiler if it appears in column 6).
3. The modules must be controlled by a top-level class-driver routine, whose calling arguments must be the computational concentration grid array (CGRID), the current scenario date (Date), scenario time (Time), and the controlling time step vector (TimeStep). (See Section 9.2.3 above.)
4. The class-driver is also responsible for any temporal integration required within the module. (The time steps for process integration at the module level are usually shorter than those of the CCTM synchronization time step.)
5. Any reads and writes for the module should be done at the level of the class-driver routine. Although not absolutely necessary, this is strongly suggested because it is usually much easier to control the timing of the data accesses at the highest level of the module where the current scenario date and time are known.
6. Use the Fortran declaration IMPLICIT NONE to maintain some control on typographic errors and undefined variables. The use of IMPLICIT NONE forces the developer to declare all internal variables. This is standard in Fortran 90.
7. Use the global INCLUDE files for chemical mechanism data, and other data where available.
8. Use the I/O API for external data references where appropriate. For an illustration of these rules, see the code template provided in Section 9.2.6.

At the submodule level, there are no strict I/O or coding standards. Here it is envisioned that individual researchers/programmers use their own coding styles for their algorithms. However, the following suggestions are offered to facilitate the potential incorporation of a module into the CMAQ system:

- In general, it is expected that MKS units are used for input and output variables, as these units have been standardized throughout the CMAQ system. Within a submodule subroutine, whatever units are most convenient can be used. However, the developer must be responsible for any unit conversions to MKS for input and output, and thus avoid potential errors.
- For efficiency and performance considerations, operations may need to be done on groups of grid cells (a block of cells) at a time. If there are N cells in the block and the entire domain contains M cells, then the entire domain can be decomposed into M/N blocks. The default value of N is set to 500. For operations in the horizontal (x,y), the cell constraint becomes X×Y N, where X = number of cells in the x-direction, and Y = number of cells in the y-direction. For operations in both the horizontal and vertical, the constraint becomes X×Y×Z N, where Z = number of cells in the z-direction. There may be some operations, such as for some horizontal advection schemes, where this decomposition into blocks becomes more difficult or impossible.

8.2.5 Documentation guidelines

Appropriate documentation is critical to the ease of use and maintainability of code developed for CMAQ. The official released version of CMAQ contains extensive in-line documentation and references to pertinent technical information whenever possible. Given the increasing number of new developers and code modules, the following guidelines are provided for new code developed for CMAQ:

- The code revision history should be initiated or updated as appropriate for new and modified code, indicating the author, date, and nature of the revision. The revision history appears at the top of the subroutine.
- Complete references to the pertinent technical documents should be provided whenever possible, and listed in comment lines immediately following the revision history notes. They should be cited in comments preceding, or embedded in-line with, the relevant code segments.
- In-line documentation of the variable definitions indicating units is highly recommended in both subroutines and INCLUDE files, to facilitate the correct implementation of any code modifications in the future. This information is generally included in comments embedded in-line with the declaration of each variable.

8.2.6 Science process code template

The following example from CMAQ v4.7 illustrates a science process class-driver Fortran 90 subroutine. Code developers should follow this template, where appropriate, to maximize the benefit from the design concepts implemented in CMAQ. This template is generic and demonstrates many of the available features. Some class drivers and most other subprograms within a module may not have, nor require, most or any of these features. (The numbers at the left-hand margin refer to footnotes and are not part of the code, and the text within "< >" indicates code removed from the example for brevity in this section)

Example of Science Process Class-Driver

```
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
      SUBROUTINE VDIFF ( CGRID, JDATE, JTIME, TSTEP )

C-----------------------------------------------------------------------
C Asymmetric Convective Model v2 (ACM2) -- Pleim(2006)
C Function:
C   calculates and writes dry deposition.
C   calculates vertical diffusion

C Subroutines and Functions Called:
C   INIT3, SEC2TIME, TIME2SEC, WRITE3, NEXTIME,
C   M3EXIT, EDDYX, TRI, MATRIX, PA_UPDATE_EMIS, PA_UPDATE_DDEP
```

```
C Revision History:
C   Analogous to VDIFFIM (Eddy diffusion PBL scheme)
C 03 Mar 16 G.Sarwar: updated for halogen emissions
C 16 Sep 16 J.Young: update for inline procan (IPR)
C-----------------------------------------------------------------------
...
C-----------------------------------------------------------------------

      USE CGRID_SPCS          ! CGRID mechanism species
      USE GRID_CONF
      USE EMIS_DEFN
      USE DEPV_DEFN
      USE ASX_DATA_MOD
      USE VDIFF_MAP
      USE UTILIO_DEFN
      USE BIDI_MOD
      USE HGSIM
      USE LSM_MOD, Only: n_lufrac
      USE SEDIMENTATION
      USE VDIFF_DIAG
      USE PA_DEFN, Only: LIPR ! Process Anaylsis control and data variables

      IMPLICIT NONE

      INCLUDE SUBST_FILES_ID  ! file name parameters

      CHARACTER( 120 ) :: XMSG = ' '

C Arguments:

      REAL, POINTER :: CGRID( :,:,:,: )              !  concentrations
      INTEGER      JDATE        ! current model date, coded YYYYDDD
      INTEGER      JTIME        ! current model time, coded HHMMSS
      INTEGER      TSTEP( 3 )   ! time step vector (HHMMSS)
                                ! TSTEP(1) = local output step
                                ! TSTEP(2) = sciproc sync. step (chem)
                                ! TSTEP(3) = twoway model time step w.r.t. wrf time
                                !            step and wrf/cmaq call frequency

C Parameters:

C External Functions: None

C Local Variables:
      CHARACTER( 16 ), SAVE :: PNAME = 'VDIFFPROC'
      CHARACTER( 16 ), SAVE :: AERO_GRAV_SETL = 'CTM_GRAV_SETL'
      CHARACTER( 80 ) :: VARDESC                ! env variable description
      LOGICAL, SAVE :: GRAV_SETL
      LOGICAL, SAVE :: FIRSTIME = .TRUE.
      LOGICAL, SAVE :: WRITE_FIRSTIME = .TRUE.
      INTEGER, SAVE :: WSTEP  = 0               ! local write counter
      INTEGER  STATUS                           ! ENV... status

      REAL          :: FCJACMF( NCOLS,NROWS,NLAYS )  ! 1/ mid-full layer vert Jac factor
```

```
      REAL             LRDX3M                          ! loop local RDX3M( L )
      REAL             FCMSF                           ! loop local RMSFX4( C,R )

      REAL, ALLOCATABLE, SAVE :: CNGRD( :,:,:,: )    ! cgrid aero in mixing ratio
      REAL, ALLOCATABLE, SAVE :: DDEP     ( :,:,: )   ! ddep accumulator
      REAL, ALLOCATABLE, SAVE :: ICMP     ( :,:,: )   ! component flux accumulator
      REAL, ALLOCATABLE, SAVE :: DDEPJ    ( :,:,:,: ) ! ddep for mosaic
      REAL, ALLOCATABLE, SAVE :: DDEPJ_FST( :,:,:,: ) ! ddep for stomtal/cuticular pathway

      REAL     :: WRDD( NCOLS,NROWS )                  ! ddep write buffer
      REAL     :: WRDDJ( NCOLS,NROWS,N_LUFRAC+1 )      ! mosaic ddep write buffer
      REAL     :: WRDDJ_FST( NCOLS,NROWS,N_LUFRAC+1 ) ! mosaic stomatal flux write buffer

      REAL, ALLOCATABLE, SAVE :: DDEP_PA  ( :,:,: )   ! ddep for process analysis
      REAL, ALLOCATABLE, SAVE :: EMIS_PA( :,:,:,: )   ! emis for process analysis

      INTEGER, SAVE :: N_SPC_CGRID              ! no. of CGRID species

      REAL     :: EDDYV ( NCOLS,NROWS,NLAYS )   ! from EDYINTB
      REAL     :: SEDDY ( NLAYS,NCOLS,NROWS )   ! flipped EDDYV
      REAL        DTSEC                         ! model time step in seconds

      REAL, ALLOCATABLE, SAVE :: VSED_AE( :,:,:,: )

  C Local Variables

      INTEGER, SAVE :: LOGDEV

      INTEGER     ASTAT
      INTEGER     C, R, L, S, V, I, J, OFF      ! loop induction variables
      INTEGER     MDATE, MTIME, MSTEP           ! internal simulation date&time

      INTERFACE
         SUBROUTINE PA_UPDATE_EMIS ( PNAME, VDEMIS, JDATE, JTIME, TSTEP )
            CHARACTER( * ), INTENT( IN )  :: PNAME
            REAL,           INTENT( IN )  :: VDEMIS( :,:,:,: )
            INTEGER,        INTENT( IN )  :: JDATE, JTIME
            INTEGER,        INTENT( IN )  :: TSTEP( 3 )
         END SUBROUTINE PA_UPDATE_EMIS
         SUBROUTINE PA_UPDATE_DDEP ( PNAME, DDEP, JDATE, JTIME, TSTEP )
            CHARACTER( * ), INTENT( IN )  :: PNAME
            REAL,           INTENT( IN )  :: DDEP( :,:,: )
            INTEGER,        INTENT( IN )  :: JDATE, JTIME
            INTEGER,        INTENT( IN )  :: TSTEP( 3 )
         END SUBROUTINE PA_UPDATE_DDEP
         SUBROUTINE CONV_CGRID ( CGRID, JDATE, JTIME, CNGRD )
            REAL, POINTER :: CGRID( :,:,:,: )
            INTEGER,        INTENT( IN )  :: JDATE, JTIME
            REAL,           INTENT( INOUT ) :: CNGRD( :,:,:,: )
         END SUBROUTINE CONV_CGRID
         SUBROUTINE REV_CGRID ( CNGRD, JDATE, JTIME, CGRID )
            REAL,           INTENT( INOUT ) :: CNGRD( :,:,:,: )
            INTEGER,        INTENT( IN )  :: JDATE, JTIME
            REAL, POINTER :: CGRID( :,:,:,: )
```

```
         END SUBROUTINE REV_CGRID
         SUBROUTINE EDDYX ( EDDYV )
            REAL,           INTENT( OUT ) :: EDDYV( :,:,: )
         END SUBROUTINE EDDYX
         SUBROUTINE VDIFFACMX( dtsec, seddy, ddep, icmp, ddepj, ddepj_fst, cngrd )
            REAL, INTENT( IN )    :: dtsec
            REAL, INTENT( INOUT ) :: seddy( :,:,: )
            REAL, INTENT( INOUT ) :: ddep ( :,:,: )
            REAL, INTENT( INOUT ) :: icmp ( :,:,: )
            REAL, INTENT( INOUT ), OPTIONAL :: ddepj    ( :,:,:,: )
            REAL, INTENT( INOUT ), OPTIONAL :: ddepj_fst( :,:,:,: )
            REAL, INTENT( INOUT ) :: cngrd( :,:,:,: )
         END SUBROUTINE VDIFFACMX
      END INTERFACE

C-----------------------------------------------------------------------
      IF ( FIRSTIME ) THEN

         FIRSTIME = .FALSE.
         LOGDEV = INIT3()

         IF ( .NOT. DEPV_INIT ( JDATE, JTIME, TSTEP, CGRID ) ) THEN
            XMSG = 'Failure initializing deposition velocities module'
            CALL M3EXIT ( PNAME, JDATE, JTIME, XMSG, XSTAT1 )
         END IF

C create global maps
         IF ( .NOT. VDIFF_MAP_INIT( N_SPC_DEPV ) ) THEN
            XMSG = 'Failure initializing index mapping module'
            CALL M3EXIT ( PNAME, JDATE, JTIME, XMSG, XSTAT1 )
         END IF

C Initialize the met data
         CALL INIT_MET( JDATE, JTIME, MOSAIC, ABFLUX, HGBIDI )

        IF ( HGBIDI ) THEN ! Initialize HGSIM module
           CALL INIT_HGSIM(JDATE, JTIME)
        END IF

C Get gravitational settling (sedi) flag.
         GRAV_SETL = .TRUE.          ! default
         VARDESC = 'Using J-,K-mode aerosols gravitational settling'
         GRAV_SETL = ENVYN( AERO_GRAV_SETL, VARDESC, GRAV_SETL, STATUS )
         IF ( STATUS .EQ. 0 ) WRITE( LOGDEV, '(5X, A)' ) VARDESC

C Get diagnostic files flag.
         VDIFFDIAG = .FALSE.          ! default
         VARDESC = 'Writing the VDIFF diagnostic files'
         VDIFFDIAG = ENVYN( VDIFF_DIAG_FILE, VARDESC, VDIFFDIAG, STATUS )
         IF ( STATUS .EQ. 0 ) WRITE( LOGDEV, '(5X, A)' ) VARDESC

C Set output file characteristics based on COORD.EXT and open the dry dep file
         IF ( IO_PE_INCLUSIVE ) THEN
            CALL OPDDEP ( JDATE, JTIME, TSTEP( 1 ), N_SPC_DDEP, ABFLUX )
```

```
            IF ( ABFLUX .OR. HGBIDI ) CALL OPASX_MEDIA( JDATE, JTIME, TSTEP( 1 ), ABFLUX )
         END IF

C Open vdiff diagnostics file (ioapi header from cgrd)
         IF ( VDIFFDIAG ) THEN
            IF ( .NOT. VDIFF_DIAG_INIT ( JDATE, JTIME, TSTEP( 1 ), GRAV_SETL ) ) THEN
               XMSG = 'Failure initializing vdiff diagnostics module'
               CALL M3EXIT ( PNAME, JDATE, JTIME, XMSG, XSTAT1 )
            END IF
         END IF
C Allocate and initialize dry deposition array

         ALLOCATE ( DDEP( N_SPC_DEPV,NCOLS,NROWS ), STAT = ASTAT )
         IF ( ASTAT .NE. 0 ) THEN
            XMSG = 'Failure allocating DDEP'
            CALL M3EXIT( PNAME, JDATE, JTIME, XMSG, XSTAT1 )
         END IF
         DDEP = 0.0   ! array assignment

         ALLOCATE ( ICMP( LCMP,NCOLS,NROWS ), STAT = ASTAT )
         IF ( ASTAT .NE. 0 ) THEN
            XMSG = 'Failure allocating ICMP'
            CALL M3EXIT( PNAME, JDATE, JTIME, XMSG, XSTAT1 )
         END IF
         ICMP = 0.0   ! array assignment

         IF ( .NOT. EMIS_INIT ( JDATE, JTIME, TSTEP( 1 ) ) ) THEN
            XMSG = 'Failure initializing emissions module'
            CALL M3EXIT ( PNAME, JDATE, JTIME, XMSG, XSTAT1 )
         END IF

C Set up for process analysis
         IF ( LIPR ) THEN
            ALLOCATE ( EMIS_PA( NCOLS,NROWS,EMLAYS,N_SPC_EMIS+1 ), STAT = ASTAT )
            IF ( ASTAT .NE. 0 ) THEN
               XMSG = 'EMIS_PA memory allocation failed'
               CALL M3EXIT ( PNAME, JDATE, JTIME, XMSG, XSTAT1 )
            END IF
            ALLOCATE ( DDEP_PA( NCOLS,NROWS,N_SPC_DEPV ), STAT = ASTAT )
            IF ( ASTAT .NE. 0 ) THEN
               XMSG = 'DDEP_PA memory allocation failed'
               CALL M3EXIT ( PNAME, JDATE, JTIME, XMSG, XSTAT1 )
            END IF
         END IF
C Set up for grav. settling
         IF ( GRAV_SETL ) THEN
            ALLOCATE ( VSED_AE( N_AE_SPC,NLAYS,NCOLS,NROWS ), STAT = ASTAT )
            IF ( ASTAT .NE. 0 ) THEN
               XMSG = 'Failure allocating VSED_AE'
               CALL M3EXIT( PNAME, JDATE, JTIME, XMSG, XSTAT1 )
            END IF
         END IF

         N_SPC_CGRID = SIZE ( CGRID,4 )
```

```
            ALLOCATE ( CNGRD( N_SPC_CGRID,NLAYS,NCOLS,NROWS ), STAT = ASTAT )
            IF ( ASTAT .NE. 0 ) THEN
               XMSG = 'Failure allocating CNGRD'
               CALL M3EXIT( PNAME, JDATE, JTIME, XMSG, XSTAT1 )
            END IF
            CNGRD = 0.0   ! array assignment

            IF ( MOSAIC ) THEN
               ALLOCATE ( DDEPJ( N_LUFRAC,N_SPC_DEPV,NCOLS,NROWS ), STAT = ASTAT )
               IF ( ASTAT .NE. 0 ) THEN
                  XMSG = 'Failure allocating DDEPJ'
                  CALL M3EXIT( PNAME, MDATE, MTIME, XMSG, XSTAT1 )
               END IF
               DDEPJ = 0.0   ! array assignment
               IF ( IO_PE_INCLUSIVE )
     &            CALL OPDDEP_MOS ( JDATE, JTIME, TSTEP( 1 ), N_SPC_DDEP )
               IF ( FST ) THEN
                  ALLOCATE ( DDEPJ_FST( N_LUFRAC,N_SPC_DEPV,NCOLS,NROWS ), STAT = ASTAT )
                  IF ( ASTAT .NE. 0 ) THEN
                     XMSG = 'Failure allocating DDEPJ_FST'
                     CALL M3EXIT( PNAME, MDATE, MTIME, XMSG, XSTAT1 )
                  END IF
                  DDEPJ_FST = 0.0   ! array assignment
                  IF ( IO_PE_INCLUSIVE )
     &               CALL OPDDEP_FST ( JDATE, JTIME, TSTEP( 1 ), N_SPC_DDEP )
               END IF   ! if Fst
            END IF   ! if Mosaic

         END IF   !  if Firstime

         MDATE = JDATE
         MTIME = JTIME
         MSTEP = TIME2SEC( TSTEP( 2 ) )
         DTSEC = FLOAT( MSTEP )
         CALL NEXTIME ( MDATE, MTIME, SEC2TIME( MSTEP / 2 ) )

C Convert non-molar mixing ratio species and re-order CGRID
         CALL CONV_CGRID ( CGRID, MDATE, MTIME, CNGRD )
C read & interpolate met data
         CALL GET_MET ( MDATE, MTIME, MSTEP, MOSAIC, ABFLUX, HGBIDI )

C read & interpolate deposition velocities
         CALL GET_DEPV ( MDATE, MTIME, TSTEP, CGRID )

         IF ( GRAV_SETL ) THEN
C Get gravitational settling velocity for the used aero species:
C AERO_SEDV assumes that every aero species is dry deposited and is diffused (trns)
C Calculate the changes in the layer J-,K-mode aerosol concentrations
            CALL SEDI( MDATE, MTIME, DTSEC, VSED_AE, CGRID, CNGRD )
         END IF

C read & interpolate emissions data => VDEMIS from EMIS_DEFN module
         CALL GET_EMIS ( MDATE, MTIME, TSTEP, CONVPA, CGRID )
```

```
      IF ( LIPR ) THEN
         DO S = 1, N_SPC_EMIS+1
            DO L = 1, EMLAYS
               DO R = 1, MY_NROWS
                  DO C = 1, MY_NCOLS
                     EMIS_PA( C,R,L,S ) = VDEMIS( S,L,C,R )
                  END DO
               END DO
            END DO
         END DO
         CALL PA_UPDATE_EMIS ( 'VDIF', EMIS_PA, JDATE, JTIME, TSTEP )
      END IF

      CALL EDDYX ( EDDYV )

C EDDYV returned = Kz, where Kz is in m**2/sec

      DO L = 1, NLAYS
         LRDX3M = Grid_Data%RDX3M( L )
         DO R = 1, MY_NROWS
            DO C = 1, MY_NCOLS
               FCJACMF( C,R,L ) = LRDX3M * Met_Data%RJACM( C,R,L ) * Met_Data%RJACF( C,R,L )
            END DO
         END DO
      END DO
      DO R = 1, MY_NROWS
         DO C = 1, MY_NCOLS
            FCMSF = Grid_Data%RMSFX4( C,R )
            DO L = 1, NLAYS
               SEDDY( L,C,R ) = FCMSF * FCJACMF( C,R,L ) * EDDYV( C,R,L )
            END DO
         END DO
      END DO

      IF ( WSTEP .EQ. 0 ) THEN
         DDEP = 0.0                      ! array assignment
         ICMP = 0.0                      ! array assignment
         IF ( MOSAIC ) THEN
            DDEPJ = 0.0                  ! array assignment
            IF ( FST ) DDEPJ_FST = 0.0   ! array assignment
         END IF
      END IF

C Calculate the change in concentration and dry dep from vertical diffusion and vsed
C Note: cngrd is the argument keyword (from the INTERFACE); CNGRD is the actual argument
      IF ( .NOT. MOSAIC ) THEN
         CALL VDIFFACMX( DTSEC, SEDDY, DDEP, ICMP,
     &                   cngrd = CNGRD )
      ELSE
         IF ( .NOT. FST ) THEN
            CALL VDIFFACMX( DTSEC, SEDDY, DDEP, ICMP,
     &                      ddepj = DDEPJ, cngrd = CNGRD )
         ELSE
```

```
            CALL VDIFFACMX( DTSEC, SEDDY, DDEP, ICMP,
     &                      ddepj = DDEPJ, ddepj_fst = DDEPJ_FST, cngrd = CNGRD )
         END IF
       END IF

       IF ( VDIFFDIAG ) THEN
          NTICS = NTICS + 1
          NLPCR_SUM = NLPCR_SUM + NLPCR_MEAN    ! array assignment
          DO R = 1, MY_NROWS
             DO C = 1, MY_NCOLS
                NLPCR_MAX( C,R ) = MAX( NLPCR_MEAN( C,R ), NLPCR_MAX( C,R ) )
                NLPCR_MIN( C,R ) = MIN( NLPCR_MEAN( C,R ), NLPCR_MIN( C,R ) )
             END DO
          END DO
          IF ( GRAV_SETL ) THEN
             DTCCR_SUM = DTCCR_SUM + DTCCR_MEAN    ! array assignment
             DO R = 1, MY_NROWS
                DO C = 1, MY_NCOLS
                   DTCCR_MAX( C,R ) = MAX( DTCCR_MEAN( C,R ), DTCCR_MAX( C,R ) )
                   DTCCR_MIN( C,R ) = MIN( DTCCR_MEAN( C,R ), DTCCR_MIN( C,R ) )
                END DO
             END DO
          END IF
       END IF
C Revert non-molar mixing ratio species and re-order CGRID
       CALL REV_CGRID ( CNGRD, MDATE, MTIME, CGRID )

C If last call this hour:  write accumulated depositions:

       WSTEP = WSTEP + TIME2SEC( TSTEP( 2 ) )
       IF ( WSTEP .GE. TIME2SEC( TSTEP( 1 ) ) ) THEN
          MDATE = JDATE
          MTIME = JTIME
          CALL NEXTIME( MDATE, MTIME, TSTEP( 2 ) )
          WSTEP = 0

#ifdef parallel_io
          IF ( WRITE_FIRSTIME ) THEN
             WRITE_FIRSTIME = .FALSE.

             IF ( .NOT. IO_PE_INCLUSIVE ) THEN
                IF ( .NOT. OPEN3( CTM_DRY_DEP_1, FSREAD3, PNAME ) ) THEN
                   XMSG = 'Could not open ' // TRIM(CTM_DRY_DEP_1)
                   CALL M3EXIT( PNAME, JDATE, JTIME, XMSG, XSTAT1 )
                END IF
                IF ( MOSAIC ) THEN
                   IF ( .NOT. OPEN3( CTM_DRY_DEP_MOS, FSREAD3, PNAME ) ) THEN
                      XMSG = 'Could not open ' // TRIM(CTM_DRY_DEP_MOS)
                      CALL M3EXIT( PNAME, JDATE, JTIME, XMSG, XSTAT1 )
                   END IF
                   IF ( FST ) THEN
                      IF ( .NOT. OPEN3( CTM_DRY_DEP_FST, FSREAD3, PNAME ) ) THEN
                         XMSG = 'Could not open ' // TRIM(CTM_DRY_DEP_FST)
                         CALL M3EXIT( PNAME, JDATE, JTIME, XMSG, XSTAT1 )
```

```
                  END IF
               END IF
            END IF
         END IF   ! .NOT. IO_PE_INCLUSIVE
      END IF
#endif

      DO V = 1, N_SPC_DDEP
         S = DD2DV( V )
         DO R = 1, MY_NROWS
            DO C = 1, MY_NCOLS
               WRDD( C,R ) = DDEP( S,C,R )
            END DO
         END DO

         IF ( .NOT. WRITE3( CTM_DRY_DEP_1, DDEP_SPC( V ),
     &                  MDATE, MTIME, WRDD ) ) THEN
            XMSG = 'Could not write ' // CTM_DRY_DEP_1 // ' file'
            CALL M3EXIT( PNAME, MDATE, MTIME, XMSG, XSTAT1 )
         END IF

         IF ( ABFLUX .AND. TRIM( DDEP_SPC( V ) ) .EQ. 'NH3' ) THEN
            DO I = 1, LCMP
               DO R = 1, MY_NROWS
                  DO C = 1, MY_NCOLS
                     WRDD( C,R ) = ICMP( I,C,R )
                  END DO
               END DO
               IF ( .NOT. WRITE3( CTM_DRY_DEP_1, CMPSPC( I ),
     &                 MDATE, MTIME, WRDD ) ) THEN
                  XMSG = 'Could not write ' // CTM_DRY_DEP_1 // ' file'
                  CALL M3EXIT( PNAME, MDATE, MTIME, XMSG, XSTAT1 )
               END IF
            END DO
         ENDIF

      END DO

      WRITE( LOGDEV, '( /5X, 3( A, :, 1X ), I8, ":", I6.6 )' )
     &       'Timestep written to', CTM_DRY_DEP_1,
     &       'for date and time', MDATE, MTIME

C Write vdiff diagnostics
      IF ( VDIFFDIAG ) THEN
         IF ( GRAV_SETL ) THEN   ! Write vsed diagnostics

            DO V = 1, N_VSED
               S = VSED_MAP( V )
               DO L = 1, NLAYS
                  DO R = 1, MY_NROWS
                     DO C = 1, MY_NCOLS
                        VSED_BUF( C,R,L,V ) = VSED_AE( S,L,C,R )
                     END DO
                  END DO
```

```
               END DO
               IF ( .NOT. WRITE3( CTM_VSED_DIAG, VSED_NAME( V ),
     &                            MDATE, MTIME, VSED_BUF( 1,1,1,V ) ) ) THEN
                  XMSG = 'Could not write ' // TRIM( VSED_NAME( V ) )
     &                 // ' to ' // CTM_VSED_DIAG
                  CALL M3EXIT( PNAME, MDATE, MTIME, XMSG, XSTAT1 )
               END IF
            END DO

            WRITE( LOGDEV, '( /5X, 3( A, :, 1X ), I8, ":", I6.6 )' )
     &           'Timestep written to', CTM_VSED_DIAG,
     &           'for date and time', MDATE, MTIME

         END IF   ! GRAV_SETL

C Write other diagnostics
         NLPCR_MEAN = NLPCR_SUM / FLOAT( NTICS )
         IF ( .NOT. WRITE3( CTM_VDIFF_DIAG, 'NLP_MEAN',
     &                      MDATE, MTIME, NLPCR_MEAN ) ) THEN
            XMSG = 'Could not write ' //  'NLP_MEAN to ' // CTM_VDIFF_DIAG
            CALL M3EXIT( PNAME, MDATE, MTIME, XMSG, XSTAT1 )
         END IF
         IF ( .NOT. WRITE3( CTM_VDIFF_DIAG, 'NLP_MAX',
     &                      MDATE, MTIME, NLPCR_MAX ) ) THEN
            XMSG = 'Could not write ' //  'NLP_MAX to ' // CTM_VDIFF_DIAG
            CALL M3EXIT( PNAME, MDATE, MTIME, XMSG, XSTAT1 )
         END IF
         IF ( .NOT. WRITE3( CTM_VDIFF_DIAG, 'NLP_MIN',
     &                      MDATE, MTIME, NLPCR_MIN ) ) THEN
            XMSG = 'Could not write ' //  'NLP_MIN to ' // CTM_VDIFF_DIAG
            CALL M3EXIT( PNAME, MDATE, MTIME, XMSG, XSTAT1 )
         END IF
         NLPCR_MAX = 0.0      ! array assignment
         NLPCR_MIN = 9.9E30   ! array assignment
         NLPCR_SUM = 0.0      ! array assignment

         IF ( GRAV_SETL ) THEN   ! Write vsed diagnostics
            DTCCR_MEAN = DTCCR_SUM / FLOAT( NTICS )
            IF ( .NOT. WRITE3( CTM_VDIFF_DIAG, 'SEDI_DTC_MEAN',
     &                         MDATE, MTIME, DTCCR_MEAN ) ) THEN
               XMSG = 'Could not write ' //  'SEDI_DTC_MEAN to ' // CTM_VDIFF_DIAG
               CALL M3EXIT( PNAME, MDATE, MTIME, XMSG, XSTAT1 )
            END IF
            IF ( .NOT. WRITE3( CTM_VDIFF_DIAG, 'SEDI_DTC_MAX',
     &                         MDATE, MTIME, DTCCR_MAX ) ) THEN
               XMSG = 'Could not write ' //  'SEDI_DTC_MAX to ' // CTM_VDIFF_DIAG
               CALL M3EXIT( PNAME, MDATE, MTIME, XMSG, XSTAT1 )
            END IF
            IF ( .NOT. WRITE3( CTM_VDIFF_DIAG, 'SEDI_DTC_MIN',
     &                         MDATE, MTIME, DTCCR_MIN ) ) THEN
               XMSG = 'Could not write ' //  'SEDI_DTC_MIN to ' // CTM_VDIFF_DIAG
               CALL M3EXIT( PNAME, MDATE, MTIME, XMSG, XSTAT1 )
            END IF
            DTCCR_MAX = 0.0      ! array assignment
```

```
            DTCCR_MIN = 9.9E30    ! array assignment
            DTCCR_SUM = 0.0       ! array assignment
         END IF

         CNVCT = 0.0   ! array assignment
         DO R = 1, MY_NROWS
            DO C = 1, MY_NCOLS
               IF ( Met_Data%CONVCT( C,R ) ) CNVCT( C,R ) = 1.0
            END DO
         END DO
         IF ( .NOT. WRITE3( CTM_VDIFF_DIAG, 'CONVCT',
     &                      MDATE, MTIME, CNVCT ) ) THEN
            XMSG = 'Could not write ' //  'convct to ' // CTM_VDIFF_DIAG
            CALL M3EXIT( PNAME, MDATE, MTIME, XMSG, XSTAT1 )
         END IF
         IF ( .NOT. WRITE3( CTM_VDIFF_DIAG, 'LPBL',
     &                      MDATE, MTIME, REAL( Met_Data%LPBL ) ) ) THEN
            XMSG = 'Could not write ' //  'lpbl to ' // CTM_VDIFF_DIAG
            CALL M3EXIT( PNAME, MDATE, MTIME, XMSG, XSTAT1 )
         END IF

         WRITE( LOGDEV, '( /5X, 3( A, :, 1X ), I8, ":", I6.6, I6 )' )
     &          'Timestep written to', CTM_VDIFF_DIAG,
     &          'for date and time (and ntics)', MDATE, MTIME, NTICS
         NTICS = 0

      END IF

      IF ( MOSAIC ) THEN

         DO V = 1, N_SPC_DDEP
            S = DD2DV( V )
            WRDD = 0.0 ! reuse array since it has already been written for hour
            DO R = 1, MY_NROWS
               DO C = 1, MY_NCOLS
                  DO J = 1, N_LUFRAC
                     WRDD( C,R ) = WRDD( C,R ) + DDEPJ( J,S,C,R ) * Grid_Data%LUFRAC( C,R,J )
                     WRDDJ( C,R,J ) = DDEPJ( J,S,C,R )
                  END DO
                  WRDDJ( C,R,N_LUFRAC+1 ) = WRDD( C,R )  ! last array element is total across all la
               END DO
            END DO

            IF ( .NOT. WRITE3( CTM_DRY_DEP_MOS, DDEP_SPC( V ),
     &                   MDATE, MTIME, WRDDJ ) ) THEN
               XMSG = 'Could not write ' // CTM_DRY_DEP_MOS // ' file'
               CALL M3EXIT( PNAME, MDATE, MTIME, XMSG, XSTAT1 )
            END IF

         END DO

         WRITE( LOGDEV, '( /5X, 3( A, :, 1X ), I8, ":", I6.6 )' )
     &          'Timestep written to', CTM_DRY_DEP_MOS,
     &          'for date and time', MDATE, MTIME
```

```
      IF ( FST ) THEN

         DO V = 1, N_SPC_DDEP
            S = DD2DV( V )
            WRDD = 0.0 ! reuse array since it has already been written for hour
            DO R = 1, MY_NROWS
               DO C = 1, MY_NCOLS
                  DO J = 1, N_LUFRAC
                     WRDD( C,R ) = WRDD( C,R ) + DDEPJ_FST( J,S,C,R ) * Grid_Data%LUFRAC( C,R,J )
                     WRDDJ_FST( C,R,J ) = DDEPJ_FST( J,S,C,R )
                     IF ( DDEPJ_FST( J,S,C,R ) .GT. DDEPJ( J,S,C,R ) ) THEN
                        WRITE( LOGDEV,* ) 'FST too big !!!'
                        WRITE( LOGDEV,* ) 'J,S,C,R = ', J, S, C, R
                        WRITE( LOGDEV,* ) 'DDEPJ,DDEPJ_FST: ', DDEPJ( J,S,C,R ), DDEPJ_FST( J,S,C
                        WRITE( LOGDEV,* ) 'DDEP Species: ', DDEP_SPC( V )
                        WRITE( LOGDEV,* ) 'Time and date: ', MTIME, MDATE
                     END IF
                  END DO
                  WRDDJ_FST( C,R,N_LUFRAC+1 ) = WRDD( C,R )  ! last array element is total across
               END DO
            END DO

            IF ( .NOT. WRITE3( CTM_DRY_DEP_FST, DDEP_SPC( V ),
     &                         MDATE, MTIME, WRDDJ_FST ) ) THEN
               XMSG = 'Could not write ' // CTM_DRY_DEP_FST // ' file'
               CALL M3EXIT( PNAME, MDATE, MTIME, XMSG, XSTAT1 )
            END IF

         END DO

         WRITE( LOGDEV, '( /5X, 3( A, :, 1X ), I8, ":", I6.6 )' )
     &          'Timestep written to', CTM_DRY_DEP_FST,
     &          'for date and time', MDATE, MTIME

      END IF   ! FST

   END IF   ! MOSAIC

   IF ( ABFLUX .OR. HGBIDI ) THEN
      CALL WRASX_MEDIA( MDATE, MTIME, ABFLUX )
   END IF

   IF ( LIPR ) THEN
      DO V = 1, N_SPC_DEPV
         DO R = 1, MY_NROWS
            DO C = 1, MY_NCOLS
               DDEP_PA( C,R,V ) = DDEP( V,C,R )
            END DO
         END DO
      END DO
      CALL PA_UPDATE_DDEP ( 'VDIF', DDEP_PA, JDATE, JTIME, TSTEP )
   END IF
```

```
C re-set dry deposition array to zero

            DDEP = 0.0
            ICMP = 0.0
            IF ( MOSAIC ) THEN
               DDEPJ = 0.0   ! array assignment
               IF ( FST ) DDEPJ_FST = 0.0   ! array assignment
            END IF

         END IF

         RETURN
         END
```

Notes:

1. Header comments - Highly recommended for internal documentation.
2. USE includes the Fortran source file specified.
3. IMPLICIT NONE must be used in Fortran 90, i.e., implicit declarations are not supported. This dramatically reduces errors due to typos and undefined variables.
4. Chemical mechanism array dimensioning and looping global variables.
5. C preprocessor flags that determine which emissions control dimensioning and looping variables are compiled.
6. Other global array dimensioning and looping global variables, including those for the I/O API. The logical variable LIPR is defined in the SUBST_PACTL_ID INCLUDE file for use at lines labeled (18).
7. Local variable declaration. Note syntax differences from Fortran-77.
8. Declarations for the argument list (standardized).
9. Declarations and PARAMETER statements for local Fortran parameters, illustrating in-line documentation of variables and units. Note syntax differences from Fortran-77.
10. Declarations for external functions not previously declared.
11. Declarations for arrays to hold external file data.
12. Declarations and definitions for local and saved variables, and dynamic memory allocations.
13. Interface is a convenient way to declare calling arguments to a subroutine as input, output, or both in the calling program through the INTENT variable specification (as IN, OUT, or IN OUT). No other declaration of the calling arguments is necessary in the calling program. If IN only, the values of arguments can be passed explicitly in the subroutine call. If OUT, the argument must be passed as a variable.
14. Code section for subroutine initialization and for any local data that need not be set at every entry into the subroutine. Such data would require a SAVE statement in the declarations. For example, FIRSTIME is initialized to .TRUE. in the local variables section.
15. Illustration of memory allocation for a variable declared as allocatable. In this example, NLAYS is accessed from the COORD.EXT file.
16. Illustrates using an I/O API function to set file interpolation time.
17. Meteorological and other data are read and interpolated through a series of subroutine calls. These subroutines in turn use I/O API utilities to perform the time interpolation of the desired met variables, deposited and emitted species.
18. Call to process analysis routine to obtain data for the optional integrated process rates function.
19. Illustrates call to another science process within the module.
20. Main computational loop over the horizontal grid.
21. Time-step loop over subsynchronization time step intervals.
22. Illustrates writing to an I/O API file within a module.
23. Subroutine end

8.3 Compiling CMAQ with New Source Code

The following steps are recommended for compiling CMAQ when a new module has been developed. The procedure creates a Makefile, which can then be modified to add the new module in the appropriate class, but the same steps can be used to obtain a configuration file that can be similarly modified to add the new module.

- On the computational platform of choice, install CMAQ using Git.
- In the $CMAQ_HOME/CCTM/scripts/ subdirectory, modify a file called bldit.cctm by uncommenting the line "set MakeOpt" (remove the leading '#' character).
- Execute the bldit.cctm script. This creates a Makefile as well as a configuration file in the subdirectory $CMAQ_HOME/CCTM/scripts/BLD_CCTM_v52b_{compiler}, where the model code has been copied.
- The Makefile can be modified to compile and link the new module by specifying <full path name>.o for the object file that needs to be linked in. It is essential that a source file with the corresponding name (with extension ".F") reside in the same directory as the specified path name for the object file.
- Issue the "make" command to compile the source code into an executable.

8.4 Guidelines to Writing Shell Scripts for CMAQ

To run a model executable, various UNIX environment variables must be set in the shell that invokes the execute command. Generally, these variables involve the modeling scenario start date and time, the run duration, the output time step interval, various internal code flags that differ among the models, and all the input and output logical (symbolic) file names. There are various ways that external file names can be referenced in the source code, and UNIX platforms can link them by using environment variables. There are I/O API utility functions that allow users to easily access these variables in the code in a generic and portable manner. An additional feature that is provided through the I/O API is the ability to declare a file "volatile" by appending a -v flag in the shell's declaration for the environment variable. By doing this, the I/O API will cause the netCDF file to update (sync) its disk copy after every write and thereby update the netCDF header. Otherwise, netCDF (I/O API) file headers are not updated until the files are closed. This feature is useful, for example, for allowing a user to analyze an open netCDF file using visualization tools while the model is executing. It is also useful in case of a system crash. A CCTM model can be restarted at the scenario time step after the last successful write using the aborted output file as the input initial data.

The following is a sample run script that can be downloaded from the CMAS web site. The build and run scripts are part of the downloaded tar file from this site.

```
#!/bin/csh -f

# ===================== CCTMv5.1 Run Script ======================
# Usage: run.cctm >&! cctm_D51a.log &
#
# To report problems or request help with this script/program:
#             http://www.cmascenter.org
# ================================================================

# ================================================================
#> Runtime Environment Options
# ================================================================

#> Choose compiler and set up CMAQ environment with correct
#> libraries using config.cmaq. Options: intel | gcc | pgi
 setenv compiler intel
 setenv compilerVrsn 13.1
```

```
#> Source the config.cmaq file to set the build environment
 cd ../..
 source ./config_cmaq.csh
 cd CCTM/scripts

#> Set General Parameters for Configuring the Simulation
 set VRSN      = v52               #> Code Version
 set PROC      = mpi               #> serial or mpi
 set MECH      = cb6r3_ae6_aq      #> Mechanism ID
 set EMIS      = 2013ef            #> Emission Inventory Details
 set APPL      = SE52BENCH         #> Application Name (e.g. Gridname)

#> Define RUNID as any combination of parameters above or others. By default,
#> this information will be collected into this one string, $RUNID, for easy
#> referencing in output binaries and log files as well as in other scripts.
 setenv RUNID  ${VRSN}_${compiler}_${APPL}

#> Set the build directory (this is where the CMAQ executable
#> is located by default).
 set BLD      = ${CMAQ_HOME}/CCTM/scripts/BLD_CCTM_${VRSN}_${compiler}
 set EXEC     = CCTM_${VRSN}.exe
 cat $BLD/CCTM_${VRSN}.cfg; echo "    "; set echo

#> Set Working, Input, and Output Directories
 setenv WORKDIR ${CMAQ_HOME}/CCTM/scripts       #> Working Directory. Where the runscript is.
 setenv OUTDIR  ${CMAQ_DATA}/output_CCTM_${RUNID} #> Output Directory
 setenv INPDIR  ${CMAQ_DATA}/SE52BENCH/single_day/cctm_input  #> Input Directory
 setenv LOGDIR  ${OUTDIR}          #> Log Directory Location
 setenv NMLpath ${BLD}             #> Location of Namelists. Common places are:
                                   #>   ${WORKDIR} | ${CCTM_SRC}/MECHS/${MECH} | ${BLD}

# ======================================================================
#> CCTM Configuration Options
# ======================================================================

#> Set Start and End Days for looping
 setenv NEW_START TRUE            #> Set to FALSE for model restart
 set START_DATE = "2011-07-01"     #> beginning date (July 1, 2011)
 set END_DATE   = "2011-07-01"     #> ending date    (July 14, 2011)

#> Set Timestepping Parameters
set STTIME     = 000000            #> beginning GMT time (HHMMSS)
set NSTEPS     = 240000            #> time duration (HHMMSS) for this run
set TSTEP      = 010000            #> output time step interval (HHMMSS)

#> Horizontal domain decomposition
if ( $PROC == serial ) then
   setenv NPCOL_NPROW "1 1"; set NPROCS   = 1 # single processor setting
else
   @ NPCOL  =  4; @ NPROW =  2
   @ NPROCS = $NPCOL * $NPROW
   setenv NPCOL_NPROW "$NPCOL $NPROW";
endif
```

```
#> Vertical extent
set NZ         = 35

#setenv LOGFILE $CMAQ_HOME/$RUNID.log  #> log file name; uncomment to write standard output to a log, o

setenv GRID_NAME SE52BENCH         #> check GRIDDESC file for GRID_NAME options
setenv GRIDDESC $INPDIR/GRIDDESC   #> grid description file

#> Output Species and Layer Options
#>   CONC file species; comment or set to "ALL" to write all species to CONC
     #setenv CONC_SPCS "O3 NO ANO3I ANO3J NO2 FORM ISOP ANH4J ASO4I ASO4J"
     #setenv CONC_BLEV_ELEV " 1 4" #> CONC file layer range; comment to write all layers to CONC

#>   ACONC file species; comment or set to "ALL" to write all species to ACONC
     #setenv AVG_CONC_SPCS "O3 NO CO NO2 ASO4I ASO4J NH3"
     setenv AVG_CONC_SPCS "ALL"
     setenv ACONC_BLEV_ELEV " 1 1" #> ACONC file layer range; comment to write all layers to ACONC
     #setenv ACONC_END_TIME Y      #> override default beginning ACON timestamp [ default: N ]

setenv EXECUTION_ID $EXEC    #> define the model execution id

#> Sychronization Time Step and Tolerance Options
setenv CTM_MAXSYNC 300       #> max sync time step (sec) [ default: 720 ]
setenv CTM_MINSYNC  60       #> min sync time step (sec) [ default: 60 ]
setenv SIGMA_SYNC_TOP 0.7    #> top sigma level thru which sync step determined [ default: 0.7 ]
#setenv ADV_HDIV_LIM 0.95     #> maximum horiz. div. limit for adv step adjust [ default: 0.9 ]
setenv CTM_ADV_CFL 0.95      #> max CFL [ default: 0.75]
#setenv RB_ATOL 1.0E-09       #> global ROS3 solver abs tol [ default: 1.0E-07 ]

#> Science Options
setenv CTM_WB_DUST Y         #> use inline windblown dust emissions [ default: Y ]
setenv CTM_ERODE_AGLAND Y    #> use agricultural activity for windblown dust
                             #>    [ default: N ]; ignore if CTM_WB_DUST = N
setenv CTM_WBDUST_BELD BELD3 #> landuse database for identifying dust source regions
                             #>    [ default: BELD3 ]; ignore if CTM_WB_DUST = N
setenv CTM_LTNG_NO Y         #> turn on lightning NOx [ default: N ]
setenv CTM_WVEL Y            #> save derived vertical velocity component to conc
                             #>    file [ default: N ]
setenv KZMIN Y               #> use Min Kz option in edyintb [ default: Y ],
                             #>    otherwise revert to KzOUT
setenv CTM_ILDEPV Y          #> calculate in-line deposition velocities [ default: Y ]
setenv CTM_MOSAIC N          #> landuse specific deposition velocities [ default: N ]
setenv CTM_FST N             #> mosaic method to get land-use specific stomatal flux
                             #>    [ default: N ]
setenv CTM_ABFLUX Y          #> ammonia bi-directional flux for in-line deposition
                             #>    velocities [ default: N ]; ignore if CTM_ILDEPV = N
setenv CTM_HGBIDI N          #> mercury bi-directional flux for in-line deposition
                             #>    velocities [ default: N ]; ignore if CTM_ILDEPV = N
setenv CTM_SFC_HONO Y        #> surface HONO interaction [ default: Y ]; ignore if CTM_ILDEPV = N
setenv CTM_GRAV_SETL Y       #> vdiff aerosol gravitational sedimentation [ default: Y ]
setenv CTM_BIOGEMIS Y        #> calculate in-line biogenic emissions [ default: N ]
setenv CTM_PT3DEMIS Y        #> calculate in-line plume rise for elevated point emissions
                             #>    [ default: N ]
setenv CTM_ZERO_PCSOA N      #> turn off the emissions of the VOC precursor to pcSOA.
```

```
                              #>    The CMAQ dev team recommends leaving pcSOA mass in the
                              #>    model for production runs. [ default: N ]

#> Process Analysis Options
setenv CTM_PROCAN N           #> use process analysis [ default: N]
#> process analysis global column, row and layer ranges
#> user must check GRIDDESC for validity!
setenv PA_BCOL_ECOL "10 320"
setenv PA_BROW_EROW "10 195"
setenv PA_BLEV_ELEV "1  4"

#> I/O Controls
setenv IOAPI_LOG_WRITE F      #> turn on excess WRITE3 logging [ options: T | F ]
setenv FL_ERR_STOP N          #> stop on inconsistent input files
setenv PROMPTFLAG F           #> turn on I/O-API PROMPT*FILE interactive mode [ options: T | F ]
setenv IOAPI_OFFSET_64 NO     #> support large timestep records (>2GB/timestep record) [ options: YES |
setenv CTM_EMISCHK N          #> Abort CMAQ if missing surrogates from emissions Input files

#> Aerosol Diagnostic Controls
setenv CTM_AVISDIAG Y         #> Aerovis diagnostic file [ default: N ]
setenv CTM_PMDIAG Y           #> What is this [ default: Y ]
setenv CTM_APMDIAG Y          #> What is this [ default: Y ]
setenv APMDIAG_BLEV_ELEV "1 3" #> layer range for average pmdiag
setenv APMDIAG_BLEV_ELEV ""   #> layer range for average pmdiag = NLAYS
setenv AVG_FILE_ENDTIME N     #> What is this [ default: N ]

#> Diagnostic Output Flags
setenv CTM_CKSUM Y            #> cksum report [ default: Y ]
setenv CLD_DIAG Y             #> cloud diagnostic file [ default: N ]
setenv CTM_AERDIAG Y          #> aerosol diagnostic file [ default: N ]
setenv CTM_PHOTDIAG Y         #> photolysis diagnostic file [ default: N ]
setenv CTM_SSEMDIAG Y         #> sea-salt emissions diagnostic file [ default: N ]
setenv CTM_DUSTEM_DIAG Y      #> windblown dust emissions diagnostic file [ default: N ]; ignore if CTM_
setenv CTM_DEPV_FILE Y        #> deposition velocities diagnostic file [ default: N ]
setenv VDIFF_DIAG_FILE Y      #> vdiff & possibly aero grav. sedimentation diagnostic file [ default: N
setenv LTNGDIAG Y             #> lightning diagnostic file [ default: N ]
setenv CTM_AOD Y              #> AOD diagnostic file [ default: N ]
setenv B3GTS_DIAG Y           #> beis mass emissions diagnostic file [ default: N ]
setenv PT3DDIAG N             #> optional 3d point source emissions diagnostic file [ default: N]; ignor
setenv PT3DFRAC N             #> optional layer fractions diagnostic (play) file(s) [ default: N]; ignor
setenv REP_LAYER_MIN -1       #> Minimum layer for reporting plume rise info [ default: -1 ]

set DISP = delete             #> [ delete | keep ] existing output files


# ========================================================================
#> Input Directories and Filenames
# ========================================================================

set ICpath    = $INPDIR/icbc              #> initial conditions input directory
set BCpath    = $INPDIR/icbc              #> boundary conditions input directory
set EMISpath  = $INPDIR/emis/gridded_area #> surface emissions input directory
set IN_PTpath = $INPDIR/emis/inln_point   #> elevated emissions input directory (in-line point only)
set IN_LTpath = $INPDIR/lightning         #> lightning NOx input directory
```

```
set METpath   = $INPDIR/met/mcip          #> meteorology input directory
#set JVALpath  = $INPDIR/jproc            #> offline photolysis rate table directory
set OMIpath   = $BLD                      #> ozone columne data for the photolysis model
set LUpath    = $INPDIR/land              #> BELD landuse data for windblown dust model
set SZpath    = $INPDIR/land              #> surf zone file for in-line seasalt emissions

set ICBC_CASE = 2013ef_v6_13g_s07         #> Version label for the ICBCs
set EMIS_CASE = 2013ef_v6_13g_s07_hg      #> Version Label for the Emissions

# ======================================================================
#> Begin Loop Through Simulation Days
# ======================================================================

set TODAYG = ${START_DATE}
set TODAYJ = `date -ud "${START_DATE}" +%Y%j` #> Convert YYYY-MM-DD to YYYYJJJ
set STOP_DAY = `date -ud "${END_DATE}" +%Y%j` #> Convert YYYY-MM-DD to YYYYJJJ

while ($TODAYJ <= $STOP_DAY )  #>Compare dates in terms of YYYYJJJ

  #> Retrieve Calendar day Information
  set YYYYMMDD = `date -ud "${TODAYG}" +%Y%m%d` #> Convert YYYY-MM-DD to YYYYMMDD
  set YYMMDD = `date -ud "${TODAYG}" +%y%m%d`   #> Convert YYYY-MM-DD to YYMMDD
  set YYYYJJJ = $TODAYJ

  #> Calculate Yesterday's Date
  set YESTERDAY = `date -ud "${TODAYG}-1days" +%Y%m%d` #> Convert YYYY-MM-DD to YYYYJJJ

# ======================================================================
#> Input Files (Some are Day-Dependent)
# ======================================================================

  #> Initial conditions
  if ($NEW_START == true || $NEW_START == TRUE ) then
     setenv ICFILE ICON_20110630_bench.nc
     setenv INITIAL_RUN Y #related to restart soil information file
     rm -rf $LOGDIR/CTM_LOG*${RUNID}*  # Remove all Log Files Since this is a new start
     mkdir -p $OUTDIR
  else
     set ICpath = $OUTDIR
     setenv ICFILE CCTM_CGRID_${RUNID}_${YESTERDAY}.nc
     setenv INITIAL_RUN N
  endif

  #> Boundary conditions
  set BCFILE = BCON_${YYYYMMDD}_bench.nc

  #> Off-line photolysis rates
  #set JVALfile  = JTABLE_${YYYYJJJ}

  #> Ozone column data
  set OMIfile   = OMI_1979_to_2015.dat

  #> Optics file
  set OPTfile = PHOT_OPTICS.dat
```

```
#> MCIP meteorology files
setenv GRID_BDY_2D $METpath/GRIDBDY2D_${YYMMDD}.nc
setenv GRID_CRO_2D $METpath/GRIDCRO2D_${YYMMDD}.nc
setenv GRID_CRO_3D $METpath/GRIDCRO3D_${YYMMDD}.nc
setenv GRID_DOT_2D $METpath/GRIDDOT2D_${YYMMDD}.nc
setenv MET_CRO_2D $METpath/METCRO2D_${YYMMDD}.nc
setenv MET_CRO_3D $METpath/METCRO3D_${YYMMDD}.nc
setenv MET_DOT_3D $METpath/METDOT3D_${YYMMDD}.nc
setenv MET_BDY_3D $METpath/METBDY3D_${YYMMDD}.nc

setenv LAYER_FILE $MET_CRO_3D  # Deprecated: MET_CRO_3D is now read directly in CCTM


#> Emissions files
if ( $CTM_PT3DEMIS == 'N' ) then
   #> Offline 3d emissions file name
   set EMISfile  = emis_mole_all_${YYYYMMDD}_cb6_bench.nc
else
   #> In-line emissions configuration
   set STKCASEG = 12US1_2011ek_cb6cmaq_v6_11g           # Stack Group Version Label
   set STKCASEE = 12US1_cmaq_cb6e51_2011ek_cb6cmaq_v6_11g   # Stack Emission Version Label
   set EMISfile  = emis_mole_all_${YYYYMMDD}_cb6_bench.nc #> Surface emissions
   setenv NPTGRPS 5          #> Number of elevated source groups

   setenv STK_GRPS_01 $IN_PTpath/stack_groups/stack_groups_ptnonipm_${STKCASEG}.nc
   setenv STK_GRPS_02 $IN_PTpath/stack_groups/stack_groups_ptegu_${STKCASEG}.nc
   setenv STK_GRPS_03 $IN_PTpath/stack_groups/stack_groups_othpt_${STKCASEG}.nc
   setenv STK_GRPS_04 $IN_PTpath/stack_groups/stack_groups_ptfire_${YYYYMMDD}_${STKCASEG}.nc
   setenv STK_GRPS_05 $IN_PTpath/stack_groups/stack_groups_pt_oilgas_${STKCASEG}.nc
   setenv LAYP_STTIME $STTIME
   setenv LAYP_NSTEPS $NSTEPS

   setenv STK_EMIS_01 $IN_PTpath/ptnonipm/inln_mole_ptnonipm_${YYYYMMDD}_${STKCASEE}.nc
   setenv STK_EMIS_02 $IN_PTpath/ptegu/inln_mole_ptegu_${YYYYMMDD}_${STKCASEE}.nc
   setenv STK_EMIS_03 $IN_PTpath/othpt/inln_mole_othpt_${YYYYMMDD}_${STKCASEE}.nc
   setenv STK_EMIS_04 $IN_PTpath/ptfire/inln_mole_ptfire_${YYYYMMDD}_${STKCASEE}.nc
   setenv STK_EMIS_05 $IN_PTpath/pt_oilgas/inln_mole_pt_oilgas_${YYYYMMDD}_${STKCASEE}.nc
   setenv LAYP_STDATE $YYYYJJJ
endif

#> Lightning NOx configuration
if ( $CTM_LTNG_NO == 'Y' ) then
   setenv LTNGNO "InLine"    #> set LTNGNO to "Inline" to activate in-line calculation

#> In-line lightning NOx options
   setenv USE_NLDN  Y        #> use hourly NLDN strike file [ default: Y ]
   setenv LTNGPARAM Y        #> use lightning parameter file [ default: Y ]
   if ( $USE_NLDN == Y ) then
      setenv NLDN_STRIKES $INPDIR/lightning/NLDN.12US1.${YYYYMMDD}_bench.nc
   else
      setenv LOG_START 2.0   #> RC value to transit linear to log linear
   endif
   setenv LTNGPARMS_FILE $INPDIR/lightning/LTNG_AllParms_12US1_bench.nc #> lightning parameter file;
```

```
  endif


  #> In-line biogenic emissions configuration
  if ( $CTM_BIOGEMIS == 'Y' ) then
     set IN_BEISpath = ${INPDIR}/land
     set GSPROpath   = ${IN_BEISpath}
     setenv GSPRO      $GSPROpath/gspro_biogenics_1mar2017.txt
     setenv B3GRD      $IN_BEISpath/b3grd_bench.nc
     setenv BIOG_SPRO  B10C6 #> speciation profile to use for biogenics
     setenv BIOSW_YN   N     #> use frost date switch [ default: Y ]
     setenv BIOSEASON  $IN_BEISpath/bioseason.12US1.2006.09apr2012_bench.nc #> ignore season switch fil
     setenv SUMMER_YN  N     #> Use summer normalized emissions? [ default: Y ]
     setenv PX_VERSION Y     #> MCIP is PX version? [ default: N ]
     setenv INITIAL_RUN Y    #> non-existent or not using SOILINP [ default: N ]; default uses SOILINP
     setenv SOILINP    $OUTDIR/CCTM_SOILOUT_${RUNID}_${YESTERDAY}.nc
                             #> Biogenic NO soil input file; ignore if INITIAL_RUN = Y
  endif

  #> Windblown dust emissions configuration
  if ( $CTM_WB_DUST == 'Y' ) then
     # Input variables for BELD3 Landuse option
     setenv DUST_LU_1 $LUpath/beld3_12US1_459X299_output_a_bench.nc
     setenv DUST_LU_2 $LUpath/beld4_12US1_459X299_output_tot_bench.nc
     setenv MODIS_FPAR $LUpath/modis_bench.nc

     if ( $CTM_ERODE_AGLAND == 'Y' ) then
        setenv CROPMAP01 ${INPDIR}/land/BeginPlanting_12km_bench.nc
        setenv CROPMAP04 ${INPDIR}/land/EndPlanting_12km_bench.nc
        setenv CROPMAP08 ${INPDIR}/land/EndHarvesting_12km_bench.nc
     endif
  endif

  #> In-line sea salt emisisions configuration
  setenv OCEAN_1 $SZpath/12US1_surf_bench.nc #> horizontal grid-dependent surf zone file

  #> Bidiretional ammonia configuration
  if ( $CTM_ABFLUX == 'Y' ) then
     setenv E2C_Soilfile  ${INPDIR}/land/2011_US1_soil_bench.nc
     setenv E2C_Fertfile  ${INPDIR}/land/2011_US1_time${YYYYMMDD}_bench.nc
     setenv B4LU_file     ${INPDIR}/land/beld4_12kmCONUS_2006nlcd_bench.nc
     setenv E2C_SOIL ${E2C_Soilfile}
     setenv E2C_FERT ${E2C_Fertfile}
     setenv BELD4_LU ${B4LU_file}
  endif

# ====================================================================
#> Output Files
# ====================================================================
  #> set output file name extensions
  setenv CTM_APPL ${RUNID}_${YYYYMMDD}
  #> set output file names
  setenv S_CGRID         "$OUTDIR/CCTM_CGRID_${CTM_APPL}.nc"         #> 3D Inst. Concenctrations
  setenv CTM_CONC_1      "$OUTDIR/CCTM_CONC_${CTM_APPL}.nc -v"       #> On-Hour Concentrations
```

```
setenv A_CONC_1        "$OUTDIR/CCTM_ACONC_${CTM_APPL}.nc -v"      #> Hourly Avg. Concentrations
setenv MEDIA_CONC      "$OUTDIR/CCTM_MEDIA_CONC_${CTM_APPL}.nc -v" #> NH3 Conc. in Media
setenv CTM_DRY_DEP_1   "$OUTDIR/CCTM_DRYDEP_${CTM_APPL}.nc -v"     #> Hourly Dry Deposition
setenv CTM_DEPV_DIAG   "$OUTDIR/CCTM_DEPV_${CTM_APPL}.nc -v"       #> Dry Deposition Velocities
setenv CTM_PT3D_DIAG   "$OUTDIR/CCTM_PT3D_${CTM_APPL}.nc -v"       #>
setenv B3GTS_S         "$OUTDIR/CCTM_B3GTS_S_${CTM_APPL}.nc -v"    #> Biogenic Emissions
setenv SOILOUT         "$OUTDIR/CCTM_SOILOUT_${CTM_APPL}.nc"       #> Soil Emissions
setenv CTM_WET_DEP_1   "$OUTDIR/CCTM_WETDEP1_${CTM_APPL}.nc -v"    #> Wet Dep From All Clouds
setenv CTM_WET_DEP_2   "$OUTDIR/CCTM_WETDEP2_${CTM_APPL}.nc -v"    #> Wet Dep From SubGrid Clouds
setenv CTM_VIS_1       "$OUTDIR/CCTM_PMVIS_${CTM_APPL}.nc -v"      #> On-Hour Visibility
setenv CTM_AVIS_1      "$OUTDIR/CCTM_APMVIS_${CTM_APPL}.nc -v"     #> Hourly-Averaged Visibility
setenv CTM_PMDIAG_1    "$OUTDIR/CCTM_PMDIAG_${CTM_APPL}.nc -v"     #> On-Hour Particle Diagnostics
setenv CTM_APMDIAG_1   "$OUTDIR/CCTM_APMDIAG_${CTM_APPL}.nc -v"    #> Hourly Avg. Particle Diagnostic
setenv CTM_RJ_1        "$OUTDIR/CCTM_PHOTDIAG1_${CTM_APPL}.nc -v"  #> Photolysis Rxn Diagnostics
setenv CTM_RJ_2        "$OUTDIR/CCTM_PHOTDIAG2_${CTM_APPL}.nc -v"  #> Photolysis Rates Output
setenv CTM_SSEMIS_1    "$OUTDIR/CCTM_SSEMIS.${CTM_APPL}.nc -v"     #> Sea Spray Emissions
setenv CTM_DUST_EMIS_1 "$OUTDIR/CCTM_DUSTEMIS.${CTM_APPL}.nc -v"   #> Dust Emissions
setenv CTM_IPR_1       "$OUTDIR/CCTM_PA_1_${CTM_APPL}.nc -v"       #> Process Analysis
setenv CTM_IPR_2       "$OUTDIR/CCTM_PA_2_${CTM_APPL}.nc -v"       #> Process Analysis
setenv CTM_IPR_3       "$OUTDIR/CCTM_PA_3_${CTM_APPL}.nc -v"       #> Process Analysis
setenv CTM_IRR_1       "$OUTDIR/CCTM_IRR_1_${CTM_APPL}.nc -v"      #> Chem Process Analysis
setenv CTM_IRR_2       "$OUTDIR/CCTM_IRR_2_${CTM_APPL}.nc -v"      #> Chem Process Analysis
setenv CTM_IRR_3       "$OUTDIR/CCTM_IRR_3_${CTM_APPL}.nc -v"      #> Chem Process Analysis
setenv CTM_DRY_DEP_MOS "$OUTDIR/CCTM_DDMOS_${CTM_APPL}.nc -v"      #> Dry Dep
setenv CTM_DRY_DEP_FST "$OUTDIR/CCTM_DDFST_${CTM_APPL}.nc -v"      #> Dry Dep
setenv CTM_DEPV_MOS    "$OUTDIR/CCTM_DEPVFST_${CTM_APPL}.nc -v"    #> Dry Dep Velocity
setenv CTM_DEPV_FST    "$OUTDIR/CCTM_DEPVMOS_${CTM_APPL}.nc -v"    #> Dry Dep Velocity
setenv CTM_VDIFF_DIAG  "$OUTDIR/CCTM_VDIFF_DIAG_${CTM_APPL}.nc -v" #> Vertical Dispersion Diagnostic
setenv CTM_VSED_DIAG   "$OUTDIR/CCTM_VSED_DIAG_${CTM_APPL}.nc -v"  #> Particle Grav. Settling Velocity
setenv CTM_AOD_1       "$OUTDIR/CCTM_AOD_DIAG_${CTM_APPL}.nc -v"   #> Aerosol Optical Depth Diagnostic
setenv CTM_LTNGDIAG_1  "$OUTDIR/CCTM_LTNGHRLY_${CTM_APPL}.nc -v"   #> Hourly Avg Lightning NO
setenv CTM_LTNGDIAG_2  "$OUTDIR/CCTM_LTNGCOL_${CTM_APPL}.nc -v"    #> Column Total Lightning NO

#> set floor file (neg concs)
setenv FLOOR_FILE ${OUTDIR}/FLOOR_${CTM_APPL}.txt

#> create output directory
if ( ! -d "$OUTDIR" ) mkdir -p $OUTDIR

#> look for existing log files and output files
set log_test = `ls CTM_LOG_???.${CTM_APPL}`
set OUT_FILES = "${FLOOR_FILE} ${S_CGRID} ${CTM_CONC_1} ${A_CONC_1} ${MEDIA_CONC}         \
           ${CTM_DRY_DEP_1} $CTM_DEPV_DIAG $CTM_PT3D_DIAG $B3GTS_S $SOILOUT $CTM_WET_DEP_1\
           $CTM_WET_DEP_2 $CTM_VIS_1 $CTM_AVIS_1 $CTM_PMDIAG_1 $CTM_APMDIAG_1             \
           $CTM_RJ_1 $CTM_RJ_2 $CTM_SSEMIS_1 $CTM_DUST_EMIS_1 $CTM_IPR_1 $CTM_IPR_2       \
           $CTM_IPR_3 $CTM_IRR_1 $CTM_IRR_2 $CTM_IRR_3 $CTM_DRY_DEP_MOS                   \
           $CTM_DRY_DEP_FST $CTM_DEPV_MOS $CTM_DEPV_FST $CTM_VDIFF_DIAG $CTM_VSED_DIAG    \
           $CTM_AOD_1 $CTM_LTNGDIAG_1 $CTM_LTNGDIAG_2"
set OUT_FILES = `echo $OUT_FILES | sed "s; -v;;g" `
echo $OUT_FILES
set out_test = `ls $OUT_FILES`

#> delete previous output if requested
if ( $DISP == 'delete' ) then
```

```
   #> remove previous log files
   echo " ancillary log files being deleted"
   foreach file ( $log_test )
      echo " deleting $file"
      /bin/rm -f $file
   end

   #> remove previous output files
   echo " output files being deleted"
   foreach file ( $out_test )
      echo " deleting $file"
      /bin/rm -f $file
   end

else
   #> remove previous log files
   if ( "$log_test" != "" ) then
     echo "*** Logs exist - run ABORTED ***"
     echo "*** To overide, set $DISP == delete in run_cctm.csh ***"
     echo "*** and these files will be automatically deleted. ***"
     exit 1
   endif

   #> remove previous output files
   if ( "$out_test" != "" ) then
     echo "*** Output Files Exist - run will be ABORTED ***"
     foreach file ( $out_test )
        echo " cannot delete $file"
        /bin/rm -f $file
     end
     echo "*** To overide, set $DISP == delete in run_cctm.csh ***"
     echo "*** and these files will be automatically deleted. ***"
     exit 1
   endif
endif

#> for the run control ...
setenv CTM_STDATE      $YYYYJJJ
setenv CTM_STTIME      $STTIME
setenv CTM_RUNLEN      $NSTEPS
setenv CTM_TSTEP       $TSTEP
setenv EMIS_1 $EMISpath/$EMISfile
setenv INIT_GASC_1 $ICpath/$ICFILE
setenv INIT_AERO_1 $INIT_GASC_1
setenv INIT_NONR_1 $INIT_GASC_1
setenv INIT_TRAC_1 $INIT_GASC_1
setenv BNDY_GASC_1 $BCpath/$BCFILE
setenv BNDY_AERO_1 $BNDY_GASC_1
setenv BNDY_NONR_1 $BNDY_GASC_1
setenv BNDY_TRAC_1 $BNDY_GASC_1
setenv OMI $OMIpath/$OMIfile
setenv OPTICS_DATA $OMIpath/$OPTfile
#setenv XJ_DATA $JVALpath/$JVALfile
set TR_DVpath = $METpath
```

```
  set TR_DVfile = $MET_CRO_2D

  #> species defn & photolysis
  setenv gc_matrix_nml ${NMLpath}/GC_$MECH.nml
  setenv ae_matrix_nml ${NMLpath}/AE_$MECH.nml
  setenv nr_matrix_nml ${NMLpath}/NR_$MECH.nml
  setenv tr_matrix_nml ${NMLpath}/Species_Table_TR_0.nml

  #> check for photolysis input data
  setenv CSQY_DATA ${NMLpath}/CSQY_DATA_$MECH

  if (! (-e $CSQY_DATA ) ) then
     echo " $CSQY_DATA  not found "
     exit 1
  endif
  if (! (-e $OPTICS_DATA ) ) then
     echo " $OPTICS_DATA  not found "
     exit 1
  endif

# ======================================================================
#> Execution Portion
# ======================================================================

  #> Print attributes of the executable
  ls -l $BLD/$EXEC; size $BLD/$EXEC
  unlimit
  limit

  date

  #> Executable call for single PE, uncomment to invoke
  # /usr/bin/time  $BLD/$EXEC

  #> Executable call for multi PE, configure for your system
  # set MPI = /usr/local/intel/impi/3.2.2.006/bin64
  # set MPIRUN = $MPI/mpirun
  time mpirun -r ssh -np $NPROCS $BLD/$EXEC

  date

# ======================================================================
#> Finalize Run for This Day and Loop to Next Day
# ======================================================================

  #> Save Log Files and Move on to Next Simulation Day
  mv CTM_LOG_???.${CTM_APPL} $LOGDIR

  #> The next simulation day will, by definition, be a restart
  setenv NEW_START false

  #> Increment both Gregorian and Julian Days
  set TODAYG = `date -ud "${TODAYG}+1days" +%Y-%m-%d` #> Add a day for tomorrow
  set TODAYJ = `date -ud "${TODAYG}" +%Y%j` #> Convert YYYY-MM-DD to YYYYJJJ
```

```
end  #Loop to the next Simulation Day

exit
```

8.5 Testing and Distribution of Development Source Code

The CMAS Center collects, tests, and distributes various operational and development versions of CMAQ through the web site <http://www.cmaq-model.org>. An archive of official releases (both current and past) and development versions of CMAQ is available to the user community. The CMAQ-MADRID and CMAQ-AMSTERDAM developed by AER, Inc. under funding from the Electric Power Research Institute can be downloaded from this archive. As a benefit to the CMAQ community, CMAS periodically updates its documentation on testing such development code versions to include additional feedback as it becomes available, based on users' experiences with these versions. Questions or comments about development versions of CMAQ such as CMAQ-MADRID should be directed to the developers at AER. Questions or comments about downloading the source code and associated documentation, and on the software development guidelines, may be directed to <http://www.cmascenter.org>.

Based on the insights gained from the testing and archiving of a development version of the model such as CMAQ-MADRID, CMAS recommends the following steps as the minimum level of coding and testing practices to be adopted by developers wishing to contribute code to the public CMAQ archive:

1. To make the best use of the CMAQ features in developing new code, the developer should review the coding conventions that are provided in the previous sections of this chapter. Also see the EPA CMAQ Science Document].
2. New code should be built using the current operational CMAQ version as a template whenever possible. This will facilitate consistency in coding practices, including naming conventions, in-line documentation, and the specification of compile time versus run-time parameters.
3. Before submitting source code to the CMAS Center, the developer should verify that the code is consistent with the operational CMAQ version from which it was built, especially in the use of common INCLUDE files (such as horizontal and vertical grid definition files) and run-time parameter settings. Mixing code from different operational versions of the CMAQ model within the same development code version can lead to problems in using the generalized CMAQ scripts.
4. Comprehensive documentation or other references to peer-reviewed literature should be provided for any new science algorithms include in the source code.
5. The developer must document the computational platform used for the testing, including type and speed of the processor(s), the compiler version used, and CPU usage. It is recommended that developers use any combination of the above for testing code intended for release through the CMAS Center, to facilitate benchmarking and portability testing by CMAS staff. Any documentation on potential differences in model outputs between different computing platforms would be useful for end-users who may not be able to duplicate the platform on which the model was initially developed and tested. To this end, code testing and documentation of test results by developers, using more than one platform if available, are highly desirable.
6. The developer should provide all input data for the test case so that interested users may attempt to run the code and reproduce the results on their own platforms.
7. It is recommended that benchmark results from the testing be provided for at least one 5-day simulation. Shorter simulations do not provide adequate results from which to discern model trends beyond the spin-up period.
8. When making incremental changes to model science, the developer should provide documentation of the results, including (a) the results for all variables that show a deviation of greater than 1.0e10-6 ppm for the gas-phase species or 1.0e10-4 µg m-3 for the particulate species from the base model results for the same case, (b) an analysis of what was done to understand these differences, and (c) conclusions of the analysis.
9. Note that more than one simulation may be necessary to adequately demonstrate seasonal or regional

biases, if any, in the results. It is also understood that with models still under development, the analysis may not resolve all differences from the operational model results. It is recommended that these unresolved issues also be documented.

Model developers are also recommended to check the CMAS website to see if there are any additional guidelines that have been recommended since the first set listed above.

8.6 References for Chapter 11: Code Management

Fine, S. S., W. T. Smith, D. Hwang, T. L. Turner, 1998: Improving model development with configuration management, IEEE Computational Science and Engineering, 5(1, Ja-Mr), 56-65.

J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, 1991: Object-Oriented Modeling and Design, Prentice Hall

Young, J. O.," "Integration of Science Code into Models-3, 1999. In *Science Algorithms of the EPA Models-3 Community Multiscale Air Quality (CMAQ) Modeling System*, D. W. Byun and J. K. S. Ching (ed.), EPA/600/R-99/030, U. S. EPA, Research Triangle Park, NC.