CMAQv5.2 Operational Guidance Document

06/30/2017

Contents

Version 5.2 (2017 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

0.1 Disclaimer

The information in this operational guidance document 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 has been subjected to review by the Community Modeling and Analysis System Center only; this content has not yet been approved by the EPA. Mention of trade names or commercial products does not constitute endorsement or recommendation for use.

0.2 Foreword

The Community Multiscale Air Quality (CMAQ) modeling system is being developed and maintained under the leadership of the EPA National Exposure Research Laboratory in Research Triangle Park, NC. CMAQ represents over two decades of research in atmospheric modeling and has been in active development since the early 1990's. The first release of CMAQ was made available in 1998 without charge for use by air quality scientists, policy makers, and stakeholder groups to address multiscale, multipollutant air quality concerns. All subsequent CMAQ releases, past, current, and future, will adhere to this same level of code accessibility and scope.

0.3 Introduction

Under the authority of the Clean Air Act, the U.S. Environmental Protection Agency (EPA) has established National Ambient Air Quality Standards (NAAQS) (U.S. EPA, 2008). These standards are designed to protect human health and the environment from high levels of criteria pollutants, such as ozone and particulate matter. Meeting the NAAQS often requires the use of controls on sources of air pollutants. The complex nature of air pollution scenarios requires control strategies to be effective for a variety of air pollutants, geographic regions, and scales. As part of the effort to decrease ambient concentrations of criteria pollutants, the EPA has approved air quality simulation models for use at regional, state, and local scales within the United States. The models have been applied to estimate the ability of various control strategies to improve air quality and ensure cost-effective results.

So-called first-generation air quality models simulated air quality using simple chemistry at local scales, and Gaussian plume formulation was the basis for prediction. Second-generation models covered a broader range of scales (local, urban, regional) and pollutants, addressing each scale with a separate model that often focused on a single pollutant or issue (e.g., ozone, acid deposition). It became apparent, however, that single-pollutant models were not sufficient. Depending on the release characteristics, the pollutants in question, and the surrounding meteorological conditions at the time of pollutant release, modeling scenarios can range from a localized, short-term phenomenon to a long-term regional event. Because some emission sources contribute to the ambient levels of more than one pollutant and can

affect an entire region on various time scales, an integrated modeling approach capable of handling multiple air pollutants and spatiotemporal scales was needed to isolate control strategies that improve overall air quality in a cost-effective manner. New air quality issues identified by the Clean Air Act Amendments of 1990 (such as visibility, fine and coarse particles, and indirect exposure to toxic pollutants) made an integrated modeling approach that could address multiple pollutants even more essential. Third-generation models were needed that could treat multiple pollutants simultaneously and at scales up to continental or larger. Future efforts toward fourth-generation systems will extend linkages and process feedback to include air, water, land, and biota to provide a more holistic approach to simulating transport and fate of chemicals and nutrients throughout an ecosystem

The EPA Community Multiscale Air Quality (CMAQ) modeling system is a third-generation air quality model. It is available online at CMAQ Model Website. The source code for CMAQ is available through a publically-accessible, version-controlled git repository on GitHub (www.github.com/usepa/cmaq) where interested parties may obtain the open-source software and contribute to enhancements of the model. CMAQ is designed for applications ranging from regulatory and policy analysis to understanding the complex interactions of atmospheric chemistry and physics. It is a three-dimensional Eulerian (i.e., gridded) atmospheric chemistry and transport modeling system that simulates ozone, particulate matter (PM), toxic airborne pollutants, visibility, and acidic and nutrient pollutant species throughout the troposphere. Designed as a "one-atmosphere" model, CMAQ can address the complex couplings among several air quality issues simultaneously across spatial scales ranging from local to hemispheric. The CMAQ source code is transparent and modular to facilitate the model's extensibility through community development by all members of the air quality modeling community.

0.4 Model Background and Goals

Air quality models integrate our understandings of the complex processes that affect the concentrations of pollutants in the atmosphere. Establishing the relationships among meteorology, chemical transformations, emissions of chemical species, and removal processes in the context of atmospheric pollutants is the fundamental goal of an air quality model (Seinfeld and Pandis, 1998). In contrast to statistical air quality models that use historical trends in observed atmospheric conditions to predict air pollution, CMAQ uses coupled mathematical representations of actual chemical and physical processes to simulate air quality. The model is based upon the underlying concept of preserving mass through a series of contiguous three-dimensional (3□D) grid cells covering a fixed model grid. A model grid is an *x*-*y*-*z* array that is fixed in space and covers a particular domain, i.e., a geographic area of interest. CMAQ therefore belongs to the Eulerian class of mathematical models that calculate a mass balance within each grid cell by solving the transport across each cell boundary and chemical transformations within each cell during a given time period. As a framework for simulating the interactions of multiple complex atmospheric processes, CMAQ thus requires two primary types of inputs: meteorological information, and emission rates from sources of emissions that affect air quality.

With weather conditions contributing the primary physical driving forces in the atmosphere (such as the changes in temperature, winds, cloud formation, and precipitation rates), representative gridded meteorology forms the basis of all 3-D air quality model simulations. The Fifth-Generation Pennsylvania State University/National Center for Atmospheric Research (PSU/NCAR) Mesoscale Model (MM5) (Grell et al., 1994) and the Weather Research and Forecasting (WRF) model - Advanced Research

WRF (WRF-ARW) (Skamarock et al., 2005) are two meteorological models that are compatible with CMAQ. The meteorology inputs dictate the following CMAQ configuration parameters:

- Horizontal grid coordinate system (e.g., latitude-longitude) and map projection (e.g., Lambert Conformal Conic)
- Horizontal grid resolution (i.e., the size of the cells composing the grid)
- Maximum spatial coverage (horizontal geographic extent, i.e., *the domain*) of the grid
- Maximum vertical grid extent (model top)
- Temporal extent (the starting and ending dates and times and the meteorology update frequency)

To obtain inputs on emissions, CMAQ relies on an emissions processor to estimate the magnitude, location, and temporal variability of pollution sources. Open-source processors such as the Sparse Matrix Operator Kernel Emissions (SMOKE) processor (IE, 2008) SMOKE Website are available for computing emissions inputs to CMAQ from emissions inventories. These emissions inputs must be on the same horizontal and vertical spatial scales and cover the same time period as are used in the air quality model simulation. The emissions inputs to CMAQ must also represent volatile organic compound (VOC) emissions using a chemical parameterization supported by CMAQ; currently supported photochemical mechanisms are recent versions of the Carbon Bond mechanism, the Statewide Air Pollution Research Center (SAPRC) mechanism, and the Regional Atmospheric Chemistry Mechanism (RACM). Additional details about the gas-phase chemistry in CMAQ are provided in the OGD section on CMAQ Chemistry and Transport Modules and in Byun and Ching (1999), Byun and Schere (2006). Those two sources also describe the primary aerosol emissions species that are supported by CMAQ ("aerosol" refers to particulate matter [tiny solid or liquid particles] suspended in the atmosphere.).

CMAQ was designed from the start as a community model. "Community modeling" is the concept that air quality model development should be a collective effort by a broad community of developers. By adopting a standardized modeling architecture, the air quality modeling community can focus its efforts on creating software enhancements and new science modules. CMAQ's modular structure facilitates customization and open-source development by the community. Using the Input/Output Applications Programming Interface (I/O API) library to control the internal and external data flows to the model, and the network Common Data Form (netCDF) library to control the input and output file formats, CMAQ is based around a transparent and platform-independent code infrastructure that promotes extensibility. The modularity of CMAQ also leads to multiple science configuration options (i.e., how various processes are represented) that model users can choose from when setting up new simulations. The trade-off for this flexibility is complexity in the model configuration; the model user is faced with hundreds of different configuration combinations when designing new simulations. To avoid confusing new CMAQ users, this document provides guidance on a basic configuration to use for getting started with the model. For experienced air quality model users, the multiple configuration combinations available provide a high level of flexibility for optimizing model performance for different air quality model applications.

CMAQ is designed to meet the needs of the multiple groups contained within the air quality modeling community: research and regulatory modelers, algorithm and science module developers, air quality forecasters, and planners and policy makers. While each of these groups has distinct individual requirements for CMAQ, they also share a common need for an efficient, transparent, and scientifically

credible tool to simulate air pollution formation and transport. To address these individual and common needs, CMAQ development and maintenance have the following goals:

1. *Scientific Integrity*. Ensure that the model remains state-of-the-science through subjecting it to regular peer reviews
2. *Community Development*. Utilize a design that encourages innovations and enhancements by all members of the air quality modeling community
3. *Multiscale Modeling*. Provide adequate technical formulations to address air quality issues on multiple spatial scales, from urban to hemispheric
4. *One-Atmosphere Design*. Provide robust and integrated science for modeling multiple, coupled air quality issues in a single simulation
5. *Modularity*. Maintain flexibility to add new, or select from existing, science modules to optimize model performance for specific applications
6. *Transparency*. Utilize programming practices that promote understanding of the model formulation at the source-code level
7. *Computational Efficiency*. Provide scientifically acceptable results without compromising the speed at which the results are generated
8. *Open-Source Design*. Enable no-cost distribution and application by the modeling community

0.5 CMAQ OGD List Tables and Figures

0.6 CMAQ OGD Acronyms

1 Overview of CMAQ System Components

CMAQ is a suite of Fortran 90 programs that work in concert to estimate ozone, PM, toxic compounds, and acid deposition throughout the troposphere. The four main CMAQ programs are

- The initial conditions processor ICON
- The boundary conditions processor BCON
- The Meteorology-Chemistry Interface Processor MCIP
- The CMAQ Chemistry-Transport Model CCTM

Utility programs distributed with CMAQ include

- The code builder/manager Bldmake
- The chemical mechanism compiler chemmech
- EBI chemistry solver builder create_ebi
- The inline photolysis preprocessor inline_phot_preproc
- The namelist converter nml
- The preprocessing program jproc

The following sections describe the CMAQ system concept, followed by details of the programs listed above.

1.1 Installation overview

Prior to CMAQ version 5.0.2, CMAQ developers used CVS for source code management, and distributed tarballs (except for MCIP) were CVS archives. Starting with version 5.0.2, CMAQ developers switched to git. All versions of CMAQ from 4.7.1 to the present are available on the U.S. EPA GitHub repository.

CMAQ source codes are used to build "executables": binary files that consist of instructions that have been translated from their original source code (e.g., Fortran) into machine code (also called machine language or object code) so that they are ready to be run (executed). Executable files are created through the use of a specialized program called a compiler.

1.1.1 Installing from GitHub

There are two options for obtaining CMAQ source code from GibHub. 1. Download a zipped code archive from the GitHub web client 2. Clone the repository directly to a Linux server using the command line. For example, to download CMAQ version 5.1, issue the following command (this assumes that git is installed on your system):

1.1.2 Installing from git-based tarballs (CMAQ version 5.0.2 and later)

Users need not install git to install CMAQ. Zip archives of the source code and scripts are available from Git Hub. Click the "Clone or download" botton on the U.S. EPA GitHub repository and select "Download ZIP" to get a zip file of the full CMAQ repository, including scripts and source code. Unzip this file on the Linux file system directory where you would like to install CMAQ.

1.1.3 Installing from CVS-based tarballs (CMAQ version 5.0.1 and earlier)

CVS must be installed on the user's Linux system before installing CMAQ versions earlier than 5.0.2. When the distributed CMAQ tar file is unpacked, a CVS directory tree is installed on the user's machine that contains archived copies of the CMAQ source code. The CMAQ program Bldmake controls the extraction of copies of CMAQ source code from CVS based on the configuration options specified by the user in C-shell build scripts. After exporting the CMAQ source code from CVS, Bldmake then invokes a Fortran 90 compiler to compile the CMAQ source code into binary object files and link them with the necessary precompiled libraries to create binary CMAQ executables. C and Fortran 90 compilers must be installed on the user's Linux system in order to create CMAQ executables.

MCIP was not distributed in a CVS archive. Instead, when older CMAQ versions are downloaded, MCIP appears in its own directory with source code and Makefiles for installation.

1.2 Configuration options

Because the model infrastructure was designed to promote modularity, the user must create new CMAQ executables for each suite of science configuration options for all programs except MCIP. There are

too many combinations of the various chemical mechanisms, horizontal and vertical transport schemes, cloud routines, and chemistry solvers in the CMAQ science configuration to include efficiently in a single executable. The requirement to recompile CMAQ with each science configuration change is offset by the flexibility to add new science to the model or to switch between different model configurations. This point about modularity is most pertinent to CCTM, although there are configuration options that must be selected when compiling the other CMAQ programs as well.

In addition to compile-time configuration options with CMAQ, there are also execution-time configuration options (options that are chosen when the model is run versus when it is compiled). The horizontal domain configuration and the vertical coordinate system are dynamic features in CMAQ that are independent of the executable. In other words, a user can employ a single executable for a simulation that uses any of the supported map projections or grid definitions, without having to recompile the source code into a new executable. A description of which CMAQ options must be selected at compilation versus at execution are is included in Chapter 7: CMAQ Programs and Libraries.

1.3 Chemistry-transport model conceptual formulation

As the chemistry-transport model (CTM) component of CMAQ, CCTM is the final program to be run in the CMAQ modeling sequence. There are three other main programs that prepare input data for CCTM (i.e., ICON, BCON, and MCIP). Before describing each of the CMAQ programs in the summary description section, we present a conceptual formulation of CMAQ and Eulerian air quality modeling to provide a framework for understanding the purposes of the various programs and their relationships to each other and to the overall system.

Eulerian CTMs use coupled ordinary differential equations (ODEs) to predict changes in pollutant concentrations throughout a three-dimensional grid that is fixed in space. The following processes affect changes in the predicted concentrations in each grid cell:

- Emissions from sources
- Horizontal and vertical advection
- Horizontal and vertical diffusion
- Chemical transformations
- Loss processes (deposition)

Mathematically, these processes relate to the concentration change in each grid cell over time ($\frac{\partial C}{\partial t}$) through the *continuity equation,* which is presented in simplified form below:

$$\frac{\partial C}{\partial t} = Adv + Diff + R_c + E_c S_c$$

where,
Adv = advection,
Diff = diffusion,
R_c = chemical transformation of species c,
E_c = emissions of species c,
S_c = loss processes for species c

In CMAQ, the advection and emissions terms are calculated based on input files generated by the meteorology and emissions models, respectively; the diffusion, chemical transformation, and loss process terms are calculated within CCTM.

The Eulerian representation of the area to be modeled is a series of contiguous grid cells that form a limited-area modeling domain on a subset of the globe. A limited-area domain requires that boundary conditions be established to account for advection of pollutants and other chemical species into the modeling domain from areas outside it. CMAQ currently accounts for advection into the domain only from the horizontal (i.e., lateral) boundaries, assuming there is no exchange through the top boundary of the domain (i.e., vertical exchange). These spatial lateral boundary conditions are estimated in CMAQ using the boundary conditions processor, BCON. Similarly, a temporal boundary condition is established with the initial conditions processor, ICON, which estimates the chemical conditions in the first time step of a CMAQ model simulation. Output from these two CMAQ programs is used with output files from the emissions and meteorological models and other CMAQ preprocessors to form the required input data for running CCTM.

1.3.1 Inline processes

In the context of air quality modeling, inline processes refer to those processes that are run at the same time as the chemistry-transport model. The major advantage of including processes inline to the CCTM is that the simulation of one parameter can feedback to the simulation of one or more other parameters. For example, by including the photolysis rate calculations inline in the CCTM, the rates will be influenced by the radiative impacts of the simulated aerosol loading. Other inline features to the CCTM, such as emissions processing, provide efficiency advantages to the modeling operation and facilitate coupled meteorology-chemistry modeling. CMAQ version 5.0 and forward supports emissions calculations for biogenic sources, windblown dust, seasalt, and point source plume rise in the CCTM.

User's must be careful to avoid double counting emissions when designing an inline CCTM simulation. The CCTM does not have the ability to identify which emissions sources were calculated offline and are being input to the model. For example, including biogenic emissions in the input emissions files and configuring the CCTM to calculate the biogenic emissions inline is redundant and will produce erroneous double counting of the emissions source.

Photolysis rates are calculated inline in the CCTM. **The preprocessing program JPROC is no longer a required part of the CMAQ modeling sequence.**

1.4 Summary descriptions of the major CMAQ programs

The major CMAQ components and ancillary programs are described below. More detailed discussions on the formulations of the CMAQ programs are available in Chapter 4, in Byun and Ching (1999), and in Byun and Schere (2006).

Note that the following list of programs is generally the order in which the CMAQ programs are run.

1.4.1 Model Builder (Bldmake)

Bldmake provides an interface to CMAQ source code repository, and to the Fortran 90 compiler for building binary executables. Because Bldmake is required to create all of the CMAQ executables except MCIP (which has its own Makefile procedure), it is the first program that needs to be compiled after installing the CMAQ source code on your system. In addition to creating executables, it also provides the option to generate a Linux Makefile. These are particularly useful for porting the CMAQ code to new operating systems, testing new code in a development environment, or trouble-shooting problems with CMAQ compilation or execution.

1.4.2 Initial Conditions Processor (ICON)

ICON generates a gridded binary netCDF file of the chemical conditions in the modeling domain for the first hour of a simulation. It can generate these initial conditions from either an ASCII file of vertically resolved concentration profiles (distributed with CMAQ) or from an existing CCTM output file. If the profiles in an ASCII file do not have the same vertical structure as the CCTM configuration to be used, ICON will interpolate the data to a vertical structure consistent with CCTM's. Using an existing CCTM output file to generate initial conditions is applicable when extrapolating initial conditions from a coarse to a fine grid simulation, as may occur when setting up nested simulations (simulations with finer-resolution grids that cover part of coarser-resolution grids). As discussed in Chapter 7, the configuration options for ICON include selecting the chemical mechanism to model, defining the horizontal and vertical grids, and choosing whether the initial conditions are generated from an ASCII profile or from an existing CCTM output file.

1.4.3 Boundary Conditions Processor (BCON)

BCON generates a gridded binary netCDF file of the chemical conditions along the horizontal boundaries of the modeling domain. These boundary conditions can be either static or time-varying, and (as with ICON) can be generated from either an ASCII file of vertically resolved concentration profiles or from an existing CCTM output file. Also as with ICON, BCON will interpolate the data in ASCII profiles to a vertical resolution that is consistent with the CCTM configuration. BCON differs from ICON, however, in that it can generate time-varying (i.e., dynamic) boundary conditions. Dynamic boundary conditions are typically extracted either from CCTM outputs from a coarse-grid simulation for nested simulations or from a CCTM simulation using a global-scale model. The file structure of the ASCII input profiles can also support the creation of dynamic boundary conditions, but generally these files are used only for creating static data. The configuration options for BCON include selecting the chemical mechanism to model, defining the horizontal and vertical grids, and choosing whether the boundary conditions are generated from an ASCII profile or from an existing CCTM output file. (discussed further in Chapter 7)

BCON is only used to create boundary conditions inputs for the CCTM from an ASCII profile file or from an existing CCTM output file. Users are responsible for preparing CCTM input boundary conditions from other sources, such as global chemistry transport models.

1.4.4 Meteorology-Chemistry Interface Processor (MCIP)

MCIP uses output files from the MM5 or WRF meteorological models to create netCDF-formatted input meteorology data that are used by SMOKE (the emissions processor that computes emissions inputs to CMAQ) and by CMAQ. MCIP prepares and diagnoses all meteorological fields that are required for SMOKE and CCTM. In addition, MCIP is currently used to calculate the time-varying, species-dependent dry deposition velocities that are used in CCTM. MCIP can be used to uniformly trim cells off the horizontal boundary of the domain defined by the meteorological model, or to window in on a subset of that domain. MCIP can also decrease the vertical resolution of the meteorological data by "layer collapsing," although this option should be used with caution as it can degrade the quality of the data if used incorrectly. Configuration options for MCIP include the time periods over which to extract data from the meteorological model output files, horizontal and vertical grid definitions, and selections for calculating dry deposition velocities and integrating satellite cloud observations into MCIP output. (discussed further in Chapter 7)

1.4.5 CMAQ Chemistry-Transport Model (CCTM)

CCTM integrates the output from the preprocessing programs described above (BCON, ICON, and MCIP), as well as CMAQ-ready emissions inputs (e.g., output from SMOKE), to simulate continuous atmospheric chemical conditions. The modeled concentrations of relevant species can be captured for output at a user-defined time frequency (typically hourly). The CCTM output files are all binary netCDF files of gridded and temporally resolved air pollutant information, such as gas- and aerosol-phase species mixing ratios, hourly wet and dry deposition values, visibility metrics, and integral-averaged concentrations.

The spatial and temporal coverages of CCTM are dictated by the input meteorology information. The science configuration is specific to each application of the model and can be adjusted to optimize model performance both computationally and in the numerical reproduction of observed air quality trends. Configuration options for CCTM include the temporal coverage of the simulation, the chemical mechanism to use in the modeling, the physics scheme to use for modeling pollutant transport, heterogeneous and aqueous chemistry options, inline processing options, and diagnostic options (such as process analysis, discussed in the next paragraph). CCTM has the largest number of configuration options of all the CMAQ programs. (discussed further in Chapter 7)

1.4.6 Chemical Mechanism Compiler (CHEMMECH)

This program creates chemical mechanism namelist files for CMAQ from a mechanism definition file. Chemical mechanisms are represented in CMAQ through a series of namelist files that contain mechanistic and kinetic parameters that describe a photochemical mechanism. CHEMMECH creates the namelist files from an ASCII mechanism-definition file that represents the chemistry as sequential equations of reactants, products, and reaction rate information. This program is needed to modify reaction stoichiometry or kinetics in the existing mechanisms, to add new species and reactions, and to implement entirely new chemical mechanisms in CMAQ.

1.4.7 EBI chemistry solver builder (CREATE_EBI)

The Euler Backward Iterative (EBI) chemistry solver is an optimized numerical solver for CCTM photochemical mechanisms. As the EBI solver is optimized for a specific chemistry mechanism configuration, a new version of the EBI solver is required for new CCTM photochemical mechanisms. The program CREATE_EBI is a CCTM source code generator for new mechanism versions. Mechanism input files for CREATE_EBI are produced by the CMAQ program CHEMMECH. The source code generated by CREATE_EBI may be used to compile a new version of the CCTM for use with updated chemistry namelist files created with CHEMMECH.

The inline photolysis preprocessor creates photolysis reaction parameter tables for the CCTM inline photlysis module.

The nml program converts chemical mechanism csv output files from chemmech to the namelist files required by the CMAQ programs.

JPROC calculates daily clear-sky photolysis rates from look-up tables of molecular absorption cross-section and quantum yield (CSQY) data, and climatologically derived ozone-column and optical depth data.

2 CMAQ Features

2.1 Features of CMAQ for Application Users

The CMAQ modeling system provides a variety of important features to users who are interested in applying the model for investigating scientific research questions or for regulatory applications such as preparation of State Implementation Plans (SIPs).

- CMAQ is designed to address the complex interactions among multiple air quality issues simultaneously. Using a one-atmosphere approach to air quality modeling by applying multiscale and multipollutant modeling techniques, CMAQ can provide independent but dynamically consistent predictions of several different pollutants at varying spatial scales.
- The modularity of the CMAQ design provides flexibility in air quality model configuration for optimizing model performance for different applications and spatial resolutions.
- Close interactions among the development communities for CMAQ and for the meteorology and emissions models provide for a tight integration of the three main components of the air quality modeling system.
- Serial and multiprocessor execution options allow the application user to optimize the performance of CMAQ on various computer platforms.
- Community development expedites the expansion of CMAQ's capabilities through the pursuit of multiple research agendas by a variety of research groups. Application users thus avoid the limitations inherent in having to rely on a single, centralized development group.
- A comprehensive training program is available through the Community Modeling and Analysis System (CMAS) Center website. The CMAS Center is a support resource for users of CMAQ and other modeling systems.

- Members of the large, international community of users connected through the CMAS Center help each other by sharing data and experiences and providing technical support.

2.2 Features of CMAQ for Air Quality Model Developers

Designed under a community-modeling paradigm, CMAQ is distributed as open-source software engineered with a modular code design to facilitate decentralized development. Built around a layered I/O API and netCDF code framework, CMAQ provides a flexible platform for testing new science algorithms, chemistry representations, and optimization techniques. CMAQ provides the following features to scientists interested in developing new algorithms or adding science to the model:

- All CMAQ source code is available through GitHub.
- Developed and distributed following open-source software conventions, CMAQ source code is easily accessible and free to obtain.
- Designed for modularity, CCTM uses standardized input/output (I/O) routines to facilitate extensibility.
- The diverse and continually growing community of CMAQ developers provides an excellent forum for discussing development-related topics of all kinds.

2.3 New Features in CMAQ and MCIP

Each release version of CMAQ contains new features and improvements over the previous release of the model. Details of the new features in each release are available as release notes is provided below. Technical details about these features are contained in the CMAQ Wiki for versions 5.1 and earlier. Beginning with version 5.2, release notes are available on the U.S. EPA GitHub repository. The following links provide details about the new features added to CMAQ since version 5.0.

New Features by CMAQ version

Version 5.2 Version 5.1 Version 5.0.2 Version 5.0.1 Version 5.0

New Features by MCIP version

Version 4.3 Version 4.2 Version 4.1

3 Science Overview

As discussed in Chapter 1, CMAQ is a multipollutant, multiscale air quality modeling system that estimates the transport and chemistry of ozone, PM, toxic airborne pollutants (referred to as “air toxics”), acidic and nutrient pollutant species as well as estimates of visibility degradation and deposition totals. CMAQ includes state-of-the-art technical and computational techniques to simulate air quality from urban to global scales. It can model complex atmospheric processes affecting transformation, transport, and deposition of air pollutants using a system architecture that is designed for fast and efficient computing. While superceded by several science updates since its release, the Science Algorithms of

the EPA Models-3 Community Multiscale Air Quality Modeling System continues to be important reference for the science and design of CMAQ. Chapter 3 of this guide includes links to descriptions of the updates made to CMAQ by release version.

CMAQ has been developed to meet the needs of both the research and application communities. The CMAQ system allows users to easily construct model variations with different characteristics, such as different chemical mechanisms or alternative cloud treatments, in order to address a specific air quality issue (illustrated schematically in Figure 4-1). This modeling configuration allows CMAQ to retain its state-of-the-science status over time because it facilitates the implementation of new science modules as appropriate.

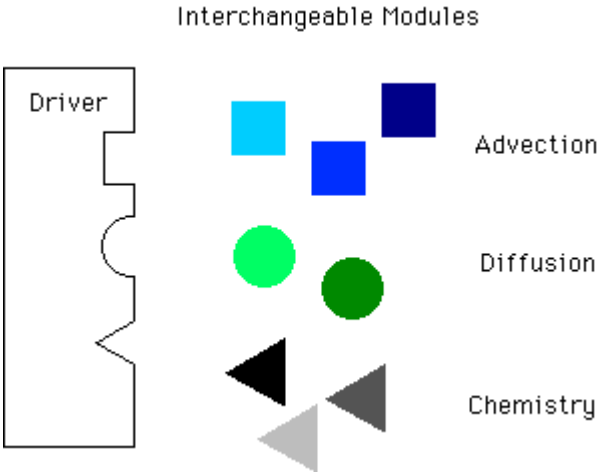


Figure 4□1. CMAQ Modeling Framework

CMAQ can be employed for regulatory applications by using approved standard configurations of the modeling platform that represent the best available modeling technology at a given time. At the same time, the CMAQ modeling system is also a useful tool for the model developer. It is unique in that its components are designed in a flexible, modular fashion with a user interface; model developers can use these design features to create complex modeling situations and scenarios, or to develop entirely new models using a standardized coding framework. Model developers can also perform sensitivity analyses on newly developed modules and perform comparisons with existing systems.

This chapter summarizes the CMAQ modeling system framework and science features in various components of the CMAQ system, including MCIP, ICON, BCON, CHEMMECH, and CCTM. More detailed discussions on these features can be found in Byun and Ching (1999) and Byun and Schere (2006). The next chapter discusses the CMAQ user interface for building and running CMAQ.

3.1 Features Implemented to Achieve the Goals of CMAQ

As noted previously, early air quality model development resulted in separate air quality models that addressed single pollutants or issues, such as ozone or acid deposition. These models had little or no flexibility to be updated with advances in science or to accommodate new regulations. CMAQ was therefore designed to have more adaptability and flexibility for different applications and for changing or improving the modeling methodology. Within the context of the model's science, the following subsections discuss CMAQ's design in terms of (1) accommodating multiple pollutants and multiple

scales, (2) providing flexibility through modularity, and (3) reducing the potential for model simulation error.

As a community model, CMAQ is able to leverage the expertise of model developers in many areas of atmospheric science. This facilitates improving and enhancing the CMAQ modeling system as the state-of-the-science evolves.

3.1.1 Multiple pollutants and multiple scales

With its "one-atmosphere" design, which allows modelers to address the complex interactions among multiple pollutants/air quality issues simultaneously, CMAQ is a dramatic improvement over the earlier, single-pollutant models. The CMAQ system provides state-of-the-science capabilities for modeling multiple air quality pollutants/issues in a single simulation, including tropospheric ozone, PM, air toxics, visibility, and acidic and nutrient pollutant species. The "one-atmosphere" approach is important because the various chemical species interact. For example, ozone and hydroxyl radicals react with emitted species such as anthropogenic and biogenic organics to generate secondary PM species. These PM species can interact with solar radiation to alter photolysis rates, temperature, ventilation, winds, thermal reactions, and temperature- and windspeed-dependent emission rates.

The multiple spatial scale (multiscale) capabilities of CMAQ enable applications from local to hemispheric scales. By combining this multiscale feature with the temporal flexibility of the model, users can perform simulations to evaluate annual and interannual pollutant climatology, as well as shorter-term transport from localized sources. To implement multiscale capabilities in CMAQ, several different issues have been addressed, such as scalable atmospheric dynamics and generalized coordinates that depend on the desired model resolution. Meteorological models may assume hydrostatic conditions for large regional scales, where the atmosphere is assumed to have a balance of vertical pressure and gravitational forces with no net vertical acceleration on larger scales. However, on smaller scales such as urban scales, the hydrostatic assumption cannot be made. A set of governing equations for compressible nonhydrostatic atmospheres is available to better resolve atmospheric dynamics at smaller scales; these are more appropriate for finer-regional-scale and urban-scale meteorology. CMAQ's generalized coordinate system is used so that meteorological fields on different horizontal and vertical coordinates can easily be accommodated and multiple scales can be simulated with the same CTM. The Jacobian used by the generalized coordinate system controls the necessary grid and coordinate transformations (consult Byun, 1999).

3.1.2 Modular flexibility

CMAQ's current coding structure is based on a modularity level that distinguishes from each other CCTM's main driver, science modules, data estimation modules, and control/utility subroutines. Also distinguished from each other are the science models (including submodels for meteorology, emissions, chemistry-transport modeling) and the analysis and visualization subsystems.

In CCTM, the process modules that affect the pollutant concentration fields are classified as listed below. Each bullet contains a description of the process followed by *module name* in parentheses. These modules, with the exception of gencoor, are discussed further later in this section.

Science Modules:

- Horizontal advection (*hadv*)
- Vertical advection (*vadv*)
- Mass conservation adjustments for advection processes (*adjc*)
- Horizontal diffusion (*hdiff*)
- Vertical diffusion (*vdiff*)
- Gas-phase chemical reaction solver (*gas*)
- Aqueous-phase reactions and cloud mixing (*cloud*)
- Aerosol dynamics and size distributions (*aero*)
- Potential vorticity scaling for stratosphere/troposphere exchange (*pv_o3*)
- Meteorology-chemistry coupling (*twoway*)

Control/Utility Modules:

- Model data flow and synchronizing of fractional time steps (*driver*)
- Model horizontal grid system (*grid*)
- Unit conversion (*couple*)
- Initialization (*init*)
- MPI/parallelization (*par*)
- CGRID configuration (*cgrds*)
- Process analysis (*procan*)
- Species namelist utilities (*spcs*)
- Miscellaneous functions (*util*)

Data Estimation Modules:

- Deposition velocity estimation (*depv*)
- Photolytic rate computation (*phot*)

In-line Emissions Modules:

- Calculate emissions (biogenics, dust, lightning, sea salt, plume rise) in-line (*emis*)
- In-line BEIS3 biogenic emissions (*biog*)
- In-line plume rise (*plrise*)

The CMAQ modularity makes it easy to modify or introduce a specific scientific process in CCTM. For example, the *gas* module contains several options for different gas-phase chemistry solvers that can be used to optimize model performance. Without the modular structure, changes to just one scientific process could entail having to modify source code throughout CCTM, thereby greatly increasing the risk of human error.

3.1.3 Quality control features

The CMAQ system was designed to minimize the potential for model simulation error in several significant ways:

- The formal CMAQ peer review process implemented by the EPA ensures that the model retains scientific credibility. Also, informal "review" of the modeling system occurs day-to-day as the broad international user community applies CMAQ for a wide variety of scientific questions and in locations other than North America.
- The modularity of the scientific processes in CMAQ makes modifications and adaptations to the user's needs more straightforward. The potential for error is minimized because the user is not required to change code or declare variables within program modules outside the one of immediate interest.

3.2 CMAQ Input Processors

CCTM uses data from other models and CMAQ input processing programs as input for model simulations Figure 4-2.

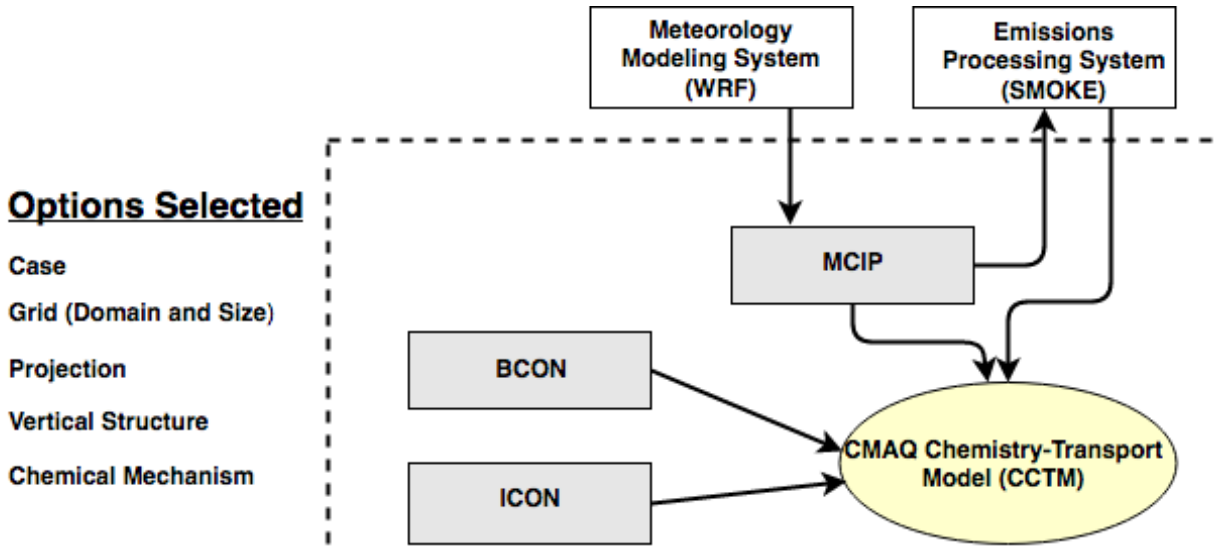


Figure 4-2. CMAQ Chemistry-Transport Model (CCTM) and pre-processors

The input data for CCTM are developed using the three pre-processors shown in grey in Figure 4-2. All of the CMAQ programs shown in Figure 4-2 (bordered by the broken line) require five basic configuration options:

- Case – a unique character string that identifies the simulation
- Grid (Domain and size) – a definition of the horizontal modeling grid that includes the location relative to a fixed map projection and the size of the domain
- Projection –defines a horizontal plane on the spherical surface of the earth, used to specify the general location of the modeling grid on the globe

- Vertical Structure – a definition of the layer boundaries for the vertical grid
- Chemical Mechanism – the name of the photochemical mechanism, aerosol chemistry mechanism, and aqueous chemistry mechanism used for the CMAQ simulation

The choices for these options and how they are selected for each of the CMAQ programs are detailed in Chapter 7.

CMAQ uses the MCIP software to prepare the meteorological fields for CCTM. The ICON and BCON processors generate the initial and boundary conditions for a CCTM simulation. Emissions for CMAQ must be prepared with an emissions data processing system (SMOKE) that generates emissions for direct input to CCTM. Brief descriptions of the various CMAQ pre-processors are presented in this section. Also described is the CHEMMECH processor, not shown in Figure 4.2.

3.2.1 MCIP: Meteorology-Chemistry Interface Processor

MCIP ingests output files from meteorological models, including the Weather Research and Forecasting Model (WRF) and the Fifth-Generation Penn State/NCAR Mesoscale Model (MM5), to create meteorology files that are used within the CMAQ Modeling System. The goal of MCIP is to use as much of the data directly from the meteorological model to ensure physical consistency in the atmospheric state used by the CMAQ Modeling System. The output from MCIP is in the standard I/O API format that is used within the CMAQ Modeling System. MCIP output files can be used by the emissions processor (e.g., for meteorologically varying temperatures for mobile emissions) and by the CCTM to define the atmospheric conditions. An overview of MCIP can be found in Otte and Pleim (2010).

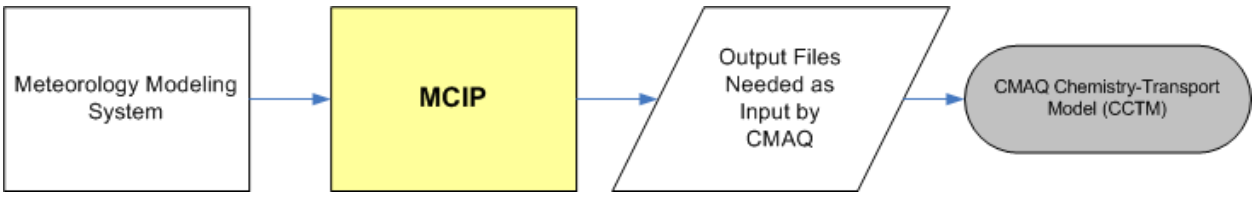


Figure 4□3. Meteorology preprocessing for CMAQ with MCIP

Using output fields from the meteorological model, MCIP performs the following functions:

- Processes all required meteorological fields for CCTM and the emissions model. Meteorological fields such as atmospheric temperature, pressure, humidity, and winds are taken directly from the meteorological model (i.e., "passed through"). Additional fields that are required by the CCTM but are not part of the meteorological model's output stream are computed within MCIP from the available meteorological fields.
- Extracts meteorological model output on the computational domain that is prescribed for the CCTM. The CCTM typically uses a smaller computational domain than the meteorological model, and the lateral boundaries from the meteorological model are generally not used by CCTM.
- Optionally reduces the number of vertical layers from the meteorological model to the CCTM using "layer collapsing". To do this, MCIP uses mass-weighted averaging on higher-vertical-resolution meteorological model output.

- Computes cloud top, cloud base, liquid water content, and cloud coverage for cumuliform clouds using simple convective schemes. The cloud parameters influence CCTM aqueous-phase chemistry and cloud mixing (Walcek and Taylor, 1986; Chang et al., 1987).
- Outputs several files in I/O API format that contain meteorological and geospatial information used by emissions processing and the CCTM.

MCIP is written in FORTRAN, and it runs on a single processor in a Unix/Linux environment. MCIP is driven by a C-shell script with several run-time options that are defined through a FORTRAN namelist. It is typical to use MCIP to process hourly output fields from the meteorological model for each one-day period.

MCIP is often updated concurrently with the CCTM. The changes to MCIP are documented with each update to the software, and a "Frequently Asked Questions" (FAQ) file exists that is specific to MCIP.

3.2.2 ICON and BCON: The initial and boundary conditions processors

To perform air quality simulations, both initial and boundary conditions are required. Initial conditions (calculated in ICON) are needed to provide concentrations of individual chemical species for the first time step throughout the modeling domain. Boundary conditions (calculated in BCON) are needed to provide concentrations of individual chemical species at the lateral boundaries of the modeling domain. In a single run ICON and BCON can generate these concentrations for all of the chemical species required by CMAQ. ICON and BCON require a file that specifies the concentrations of various chemical species in the troposphere and specification of the photochemical chemical and aerosol mechanisms that will be used in the supported CCTM simulation. These processors require two inputs Figure 4□4: a concentration file for the chemical species to be simulated, and the chemical mechanism.

Concentration file: The concentration file used in ICON and BCON can come from one of two sources:

- A time-independent set of vertical concentration profiles that are dependent upon the chemical mechanism being used. This approach is usually taken when no other information about the initial and boundary concentrations is available. CMAQ is currently distributed with IC and BC profiles for the CB05, RACM2, and SAPRC-07T photochemical mechanisms and the CMAQ AERO6 aerosol module. These files are set at the four boundaries (north, east, south, west) of the computational grid and are thus fixed in space.
- Existing CCTM 3-D concentration fields. Usually, this option is selected when performing a nested model simulation and modeling results from a previous CCTM simulation are available from a coarser-grid-resolution simulation. Existing CCTM concentration fields are also used when a CCTM simulation is extended in time in a separate run step. Unlike the profiles discussed in the previous bullet, these CCTM concentration files are spatially and temporally resolved.

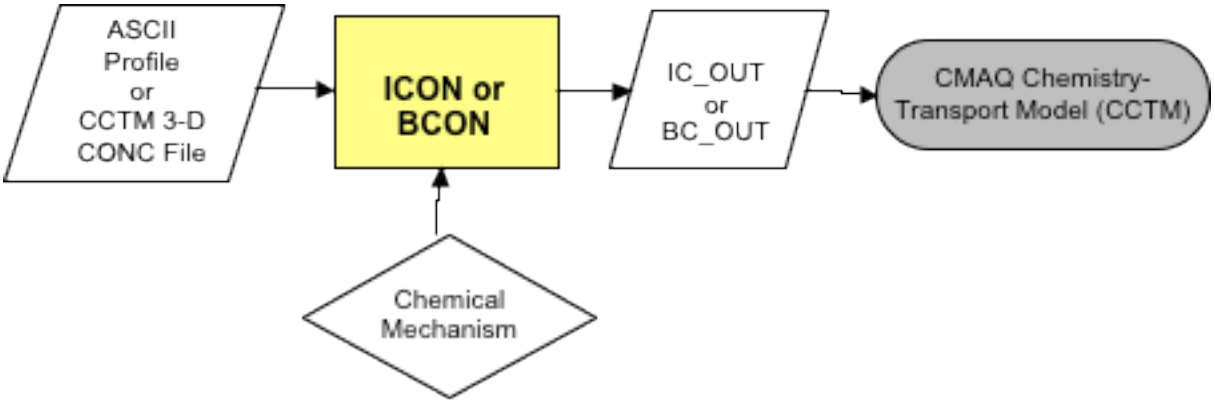


Figure 4□4. Initial and boundary conditions preprocessing for CMAQ

Chemical mechanism: Both the vertical concentration profiles and the CCTM concentration fields have specific chemical mechanisms associated with them, which are a function of how the files were originally generated. Either a generic ASCII input profile or an existing CCTM 3-D concentration file can be used to generate initial and boundary conditions for the CCTM. The user must consider the gas-phase chemical mechanism and aerosol module being used for the CCTM simulation when configuring ICON and BCON. CMAQ includes ASCII input profiles for the RACM2, CB05,and SAPRC-07T photochemical mechanisms and the CMAQ AERO6 aerosol module. Existing CCTM 3□D concentration fields could have been generated using several different chemical mechanisms.

The chemical mechanism used in the CCTM and the CMAQ input processors must be consistent with the mechanism used to generate the concentration fields input to ICON and BCON. In other words, users must generate separate initial and boundary conditions using the same chemical mechanism that will be used for the CCTM simulation.

ICON and BCON can linearly interpolate input concentration profiles from the horizontal or vertical coordinate system used in the profiles to the one needed for the model simulation, if the input data are in the standard I/O API format. If the interpolation is between two different vertical coordinate systems, the mid-layer height of each vertical layer must also be available.

The current ICON and BCON processors cannot generate initial and boundary conditions for the CMAQ model from hemispheric and global model results. A separate processor (GEOS2CMAQ) is available for generating boundary conditions for the CMAQ model using the GEOSSCHEM model results. Another processor is also available for generating boundary conditions for the regional CMAQ model using the hemispheric CMAQ model results. These processors are under development and will be released at a later date. Please contact us if you need the processors now.

3.2.3 CHEMMECH: Chemical mechanism compiler

The release version of CMAQ includes all necessary chemical mechanism information for the pre-configured atmospheric chemistry reactions sets or mechanisms in a released version of CMAQ. Users choose which mechanism to use for compiling and runing the CCTM executable. CCTM implements a chemical mechanism by using its namelist and FORTRAN modules. The files are in ASCII format and include the mechanism parameters required such as the species, reaction stoichiometry, and kinetics

information. The module files are used to compile the CCTM executable while the namelists are read by CCTM at run time.

Advanced users who wish to generate a new chemical mechanism have to use the CHEMMECH utility to convert the mechanism into the files needed by the CCTM program. CHEMMECH uses a mechanism definition file, often named "mech.def", and optionally the mechanism namelist files to generate FORTRAN modules. The "mech.def" is an ASCII file that uses a rigid syntax to define reactions and their rate constants.

This approach defining the CMAQ chemical mechanisms allows the chemical reactions and their species to be a fixed part of the executable code. Modifications to the namelists can change predictions saved to the output files, deposition processes of species, emissions inputs and other options for species without recompiling the executable. The namelists defining a chemical mechanism are used by CCTM as well as the ICON and BCON pre-processors. The FORTRAN modules are required to run utility programs such as create_ebi and inline_phot_preproc and JPROC.

3.2.4 Lightning NO processing in CMAQ

CMAQ is instrumented to estimate the impacts of NO emissions from lightning on air quality. Details of the CCTM lightning NO capability are described in Chapter 7 and in the CMAQv52 Release Notes. Emissions of lightning NO can either be generated inline or read in as an external input file that contains 3-D NO data. There are two ways to estimate lightning NO emissions in the CCTM: * Use observed hourly lightning flash count data from National Lightning Detection Network (NLDN); NLDN flash counts for the years 2002-2016 gridded to 12 km resolution are available from the CMAS center. A lightning parameter file also contains the ocean mask and ICCG data (climatological data for the ratio between inter-cloud to cloud-to-ground flashes); the ocean mask and ICCG data are used in both inline production schemes. The parameter file is available with the NLDN hourly flash data. The default lightning NO production rate is set to 350 moles per flash for both CG and IC flashes, but these values can be modified through the CCTM environment variables (MOLSNCG and MOLSNIC). Download hourly NLDN data and a parameter file for the U.S. * Use linear (log-linear) parameters derived from historical NLDN data and model predicted convective precipitation from the Kain-Fritsch convective scheme. This options is available when observed hourly flash count data (e.g., NLDN) are not available, such as air quality forecasts and future climate applications.

3.2.5 CALMAP: Crop calendar map preprocessor

CMAQ has the capability to estimate windblown dust emissions in-line in the CCTM. The CMAQ dust emissions module uses land cover/land use data to identify dust source regions. The dust module includes a feature to estimate dust blown off by the wind (as opposed to anthropogenic dust emissions) from agricultural areas and the impacts of planting and harvesting cycles on available erodible lands that affect dust emissions. Calmap is a preprocessor to the CCTM that uses crop calendar information to produce gridded crop planting and harvesting dates for input to the CMAQ dust module.

Figure 4-5 is a Calmap schematic showing the data flow through the software. CALMAP reads grid information from the GRIDCRO2D meteorology file (MCIP output), land cover/land use data from

BELD3, and crop calendar data to produce files of planting start dates, planting end dates, and harvesting end dates for different crop types interpolated to the modeling grid. These files are input to the CCTM when it is configured to estimate windblown dust and simulate the impacts of agricultural activity on the windblown dust emissions.

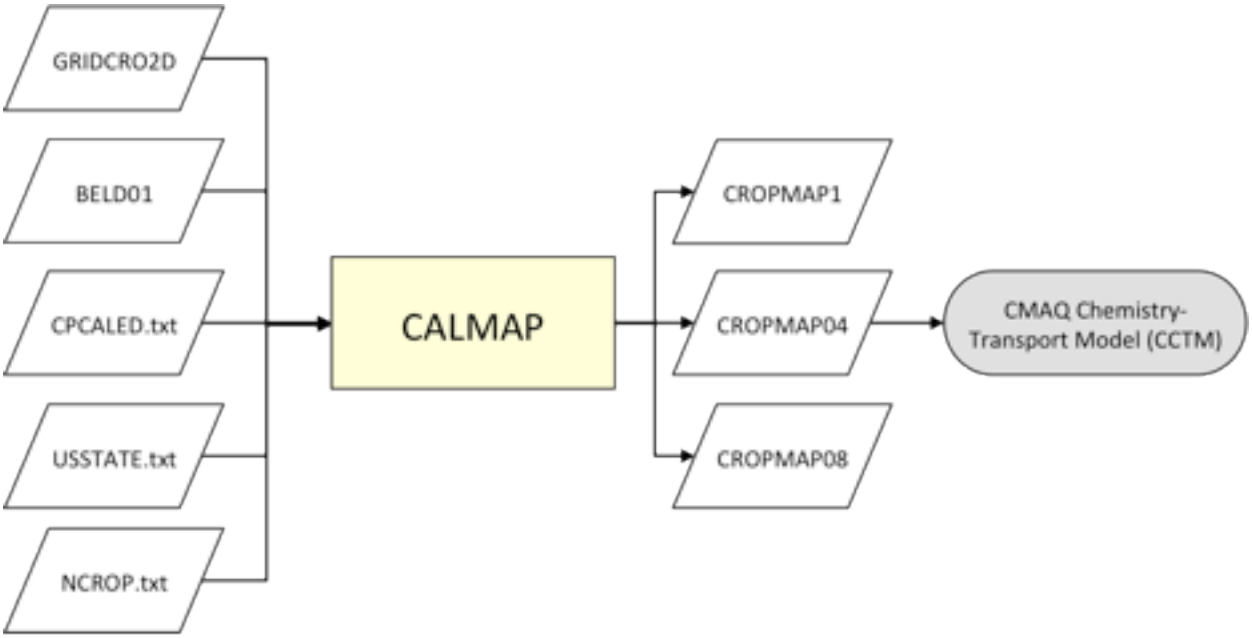


Figure 4□5. Crop calendar data preprocessor for the CCTM

3.3 CCTM: The CMAQ Chemistry-Transport Model

Figure 4□6 illustrates the CMAQ modeling system used to simulate the chemistry and transport of pollutants. This figure also shows how CMAQ relates to other parts of an air quality modeling system: the meteorological model, emissions model, and analysis software. To perform a model simulation, CMAQ needs input data, including meteorological and emissions data. Using this information, CCTM simulates each of the atmospheric processes that affect the transport, transformation, and removal of ozone, particulate matter, and other pollutants. CMAQ uses state-of-the-science techniques to simulate these processes, as described below.

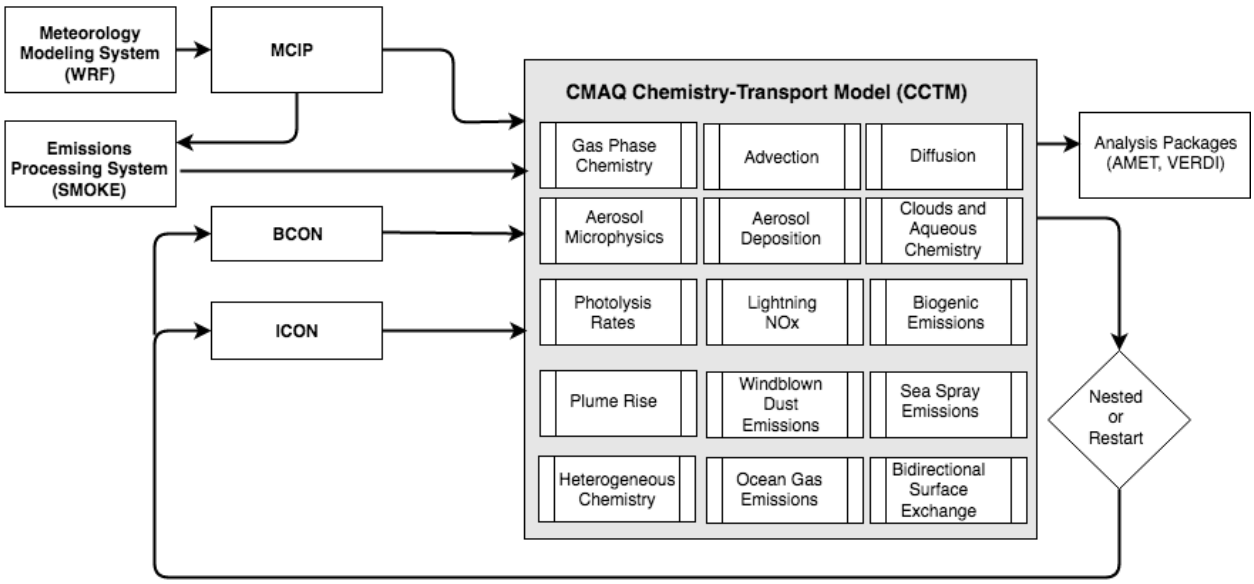


Figure 4-6. CMAQ chemistry-transport model and associated preprocessors

3.3.1 Gas-Phase Chemistry

See the CMAQv5.2 release notes for updates on the chemistry algorithms in CMAQ.

3.3.1.1 Gas-phase chemistry solvers

To determine the time dependent concentrations of species described by a chemical mechanism, the CCTM uses numerical methods to solve ordinary differential equations representing the chemical transformations. Three solution methods are available and differ in terms of accuracy, generalization, and computational efficiency, i.e. model run times . They include the Rosenbrock (ROS3) solver (Sandu et al., 1997), the Euler Backward Iterative (EBI) solver (Hertel et al., 1993), and the Sparse Matrix Vectorized GEAR (SMVGEAR) solver (Jacobson and Turco, 1994). SMVGEAR and ROS3 are considered more accurate, in the order listed. Both solutions are labeled as "generalized" because using either only requires the mechanism's namelist and FORTRAN modules. The EBI solver is more computationally efficient but is less accurate and is not a "generalized" solver so each chemical mechanism requires its own EBI solver. CMAQ includes EBI solvers for each mechanism definitions file releases with a model version. Consult the CMAQ release notes for what mechanisms are in a specific version. If a user creates or modifies a chemical mechanism , they have to create a new EBI solver by the using the create_ebi utility.

3.3.1.2 Photolysis

Photolysis or photodissociation energize and break apart compounds in several key of chemical processes in the atmosphere. It plays in the formation of ozone and particular material that affect human health. Computing the rate of photolysis reactions therefore strongly influences how well an air quality model simulates reality.

The calculation of a photolysis rate must include multiple influences. Clouds, surface features, atmospheric gas and aerosols affect photolysis rates because each scatter and absorb light. A given photolysis reaction depends on molecular properties of the compound involved. Two properties describe the dependence. The absorption cross section represents the effective molecular area of a compound for absorbing solar radiation. The quantum yield gives the chance that the molecule that dissociates after absorbing the radiation. Each property depends on the wavelength of the incident radiation, as well as air temperature and density.

The in-line method (Binkowski et al., 2007) is the preferred method for calculating photolysis rates in the CCTM program of CMAQ model system. The method uses aerosol and ozone predicted within a simulation to calculate the solar radation. Two input files support the calculation. The PHOT_OPTICS file describe the optical properties of clouds, aerosols, and the earth's surface. The OMI file is used to determine how much light is absorbed by atmosphere above the model domain. Both files are included in the released version of CMAQ. Calculating photolysis rates uses an additional input file called the CSQY_DATA file. It contains the cross sections and quantum yields of photolysis rates in a given chemical mechanism. CSQY_DATA files are provided for all chemical mechanisms in a released version of CMAQ. If a user creates a mechanism using new or additional photolysis rates, they have to create a new CSQY_DATA file. The inline_phot_preproc utility produces this file based on the Fortran modules describing the mechanism (see the section on the CHEMMECH utility) and individual files describing the absorption cross-section and quantum yields described for each photolysis reaction.

The CMAQ modeling system includes an additional method to calculate photolysis rates based on look-up tables. The tables gives a mechanism's photolysis rates under cloud free conditions based on a fixed meridional cross-sections of atmospheric composition, temperature, density and aerosols. Each table represents rates as a function altitude, latitude and the hour angle of the sun on a specified Julian date. In model simulations, the method interpolates rates in the table for the date and corrects them to account for clouds described by the meteorological input files. The JPROC utility program creates the table based on the FORTRAN modules describing the chemical mechanism. The utility program also requires files describing each photolysis rates.

3.3.2 Pollution Transport

See the CMAQv5.2 release notes for updates on the transport algorithms in CMAQ.

Pollutant transport includes both advection and sub-grid-scale diffusion. Advection has to do with pollutant transport that is due to the mean wind fields, while diffusion involves sub-grid-scale turbulent mixing of pollutants. If a pollutant plume is transported primarily by advection, then it may travel a long distance without much change in pollutant concentrations. On the other hand, if a plume is transported primarily by diffusion, then the pollutants will mix more quickly and nearer to the source, which will result in substantial changes to pollutant concentrations.

3.3.2.1 Advection

In CCTM, the advection process is divided into horizontal and vertical components. This distinction is possible because mean atmospheric motion is mostly horizontal. Often, the vertical motion is related to the interaction of dynamics and thermodynamics. The advection process relies on the mass

conservation characteristics of the continuity equation. Data consistency is maintained for air quality simulations by using dynamically and thermodynamically consistent meteorology data from MCIP. When the meteorological data and the numerical advection algorithms are not exactly mass consistent, one needs to solve a modified advection equation (Byun, 1999).

The horizontal advection scheme for CMAQ is the piecewise parabolic method (PPM) (Colella and Woodward, 1984). This algorithm is based on the finite-volume subgrid definition of the advected scalar. In PPM, the subgrid distribution is described by a parabola in each grid interval. PPM is a monotonic and positive-definite scheme. Positive-definite schemes maintain the sign of input values, which in this case means that positive concentrations will remain positive and cannot become negative. These codes are implemented in a global mass-conserving scheme introduced in v4.6 that is similar to the one used in the air quality forecasting version of CMAQ. Inspired by discussions with Robert Yamartino of Cambridge Environmental, the method uses the PPM scheme for horizontal advection, deriving a vertical velocity component at each grid cell that satisfies the continuity equation using the driving meteorological model's air density.

The vertical advection modules solve for the vertical advection with no mass-exchange boundary conditions at the bottom or top of the model. CMAQ also uses PPM as its vertical advection module. Starting in CMAQv5.0, a new method for computing the vertical velocity was implemented that follows the omega calculation in WRF but uses PPM to compute horizontal mass divergence. The two-step process first integrates the continuity equation through the vertical column to get the change in column mass and then solves for omega layer by layer using the horizontal mass divergence (see equation 2.27 in the WRF ARWv3 Technical Note). In CCTM, the PPM algorithm with a steepening procedure is implemented for vertical advection as the default because of the strong gradients in the tracer species that are observed in photochemical air quality conditions.

3.3.2.2 Diffusion

In CCTM, vertical diffusion is represented by the Asymmetric Convective Method (ACM) of Pleim and Chang (1992). ACM2 (Pleim, 2007), an updated version of ACM, was implemented starting in CMAQv5.0. This method recognizes that under convective conditions (when the surface is warming), heated air is transported vertically by buoyancy and mixes with ambient air at each level above the surface until the temperature of the rising air equals the ambient temperature. This process results from fast-moving air in narrow updrafts and slower-moving air in broader downdrafts. Thus, under convective conditions, vertical diffusion is asymmetric. An in-line method for treating biogenic and point-source emissions uses ACM to vertically distribute these emissions during a CMAQ calculation.

Under non-convective conditions (when the surface is cooling), vertical diffusion is represented by an eddy diffusivity approach. Eddy diffusivity is a local mixing scheme and is estimated using the same planetary boundary layer (PBL) similarity-based algorithm as in the Regional Acid Deposition Model (Chang et al., 1987, 1990). In CCTM, the deposition process is simulated as a flux boundary condition that affects the concentration in the lowest vertical model layer. By treating the deposition process as the loss of mass due to the diffusion flux at the bottom of the model, one can relate the bottom boundary condition in the generalized coordinate system to that in the Cartesian coordinate system. CMAQv5 has an improved version of the minimum allowable vertical eddy diffusivity scheme. The new version interpolates between urban and nonurban land cover, allowing a larger minimum vertical diffusivity value for grid cells that are primarily urban.

Horizontal diffusion is implemented with a single eddy diffusion algorithm that is based on local wind deformation and is scaled to the grid cell size. The horizontal eddy diffusivity is assumed to be uniform but dependent on the grid resolution of the model. This diffusivity is larger for a higher-resolution run where the numerical diffusion due to the advection process is smaller.

3.3.3 Particulate matter (aerosols)

See the CMAQv5.2 release notes for updates on the aerosol chemistry algorithms in CMAQ.

Within the air quality community, atmospheric aerosol particles are referred to as particulate matter (PM). PM can be either primary (directly emitted) or secondary (formed in the atmosphere) and from natural or anthropogenic (man-made) sources. Secondary sources include gas-phase oxidation of SO2 to sulfate, condensation of ammonia and nitrate, and oxidation of gas-phase VOCs such as isoprene, monoterpenes, aromatics, and alkanes. Cloud processes also contribute to the formation of PM; for example, aqueous oxidation of sulfur dioxide in cloud droplets is a significant pathway for production of particulate sulfate. CCTM represents PM using three interacting lognormal distributions, or modes. Two modes (Aitken and accumulation) are generally less than 2.5 microns in diameter while the coarse mode contains significant amounts of mass above 2.5 microns. PM2.5 and PM10, species aggregate metrics within the NAAQS, can be obtained from the model mass concentration and size distribution information.

Particulate matter is removed from the atmosphere by wet scavenging, settling, or by dry deposition at the surface, all of which are modeled by CMAQ. In dry deposition, the transfer is by turbulent air motion and by direct gravitational sedimentation of larger particles. The deposition velocity for particles must be calculated from the aerosol size distribution and from meteorological and land use information. CMAQ's dry deposition module calculates the size distribution from the mass and number concentration for each of the three modes and then calculates the dry deposition velocity. Particles in the coarse mode are so large that CMAQ explicitly accounts for their gravitational settling as well. In wet deposition, PM is transferred by rainfall. Wet deposition is calculated within CMAQ's cloud module. Starting in CMAQv4.7, the wet deposition algorithm was modified to include an impaction term in the coarse and accumulation modes.

The 6th generation CMAQ aerosol module (AERO6) expands the chemical speciation of PM. Eight new PM species are added to CMAQ in AERO6: Al, Ca, Fe, Si, Ti, Mg, K, and Mn. Four species that were explicitly treated in previous versions of CMAQ but were not modeled can now be treated as primary anthropogenic species: H2O, Na, Cl, and NH4. The PM emissions mass that remains after speciation into the new components is now input to the model as PMOTHER. AERO6 requires 18 PM emissions species: OC, EC, sulfate, nitrate, H2O, Na, Cl, NH4, NCOM, Al, Ca, Fe, Si, Ti, Mg, K, Mn, and Other (Reff et al., 2009).

CMAQ includes two options for representing POA: nonvolatile or semivolatile. For the nonvolatile POA configuration, mass is tracked separately in terms of its carbon (OC) and non-carbon (NCOM) content. With this approach, mass can be added to the non-carbon species to simulate the aging of POA in response to atmospheric oxidants. Simon and Bhave (2011) document the implementation of the second-order reaction between primary organic carbon and OH radicals. The semivolatile POA configuration segregates POA into several model species based on a combination of volatility and oxidation state (Murphy et al., 2017). There are five POA species at low oxidation state representing

low volatility, semivolatile and intermediate volatility compounds (LVPO1, SVPO1, SVPO2, SVPO3, IVPO1). As the gas-phase species (e.g. VLVPO1) oxidize with OH they form species with higher oxidation state (i.e. LVOO1, LVOO2, SVOO1, SVOO2, SVOO3). The multigenerational aging chemistry for the semivolatile POA configuration is derived from the approach of Donahue et al. (2012) which takes into account the functionalization and fragmentation of organic vapors upon oxidation. The semivolatile POA configuration also includes the option (on by default) of potential secondary organic aerosol from combustion sources (pcSOA). This species is emitted as a VOC (pcVOC) and forms SOA after reaction with OH. The emissions of pcVOC may be zeroed out by the user.

AERO6 uses ISORROPIA in the "reverse mode" to calculate the condensation/evaporation of volatile inorganic gases to/from the gas-phase concentrations of known coarse particle surfaces. It also uses ISORROPIA in the "forward mode" to calculate instantaneous thermodynamic equilibrium between the gas and fine-particle modes. The mass transfer of all semivolatile organic species is calculated assuming equilibrium absorptive partitioning, although some nonvolatile species do exist (e.g. cloud-processed organic aerosol, oligomers, nonvolatile POA (if selected)).

CMAQ can output the reduction in visual range caused by the presence of PM, perceived as haze. CCTM integrates Mie scattering (a generalized particulate light-scattering mechanism that follows from the laws of electromagnetism applied to particulate matter) over the entire range of particle sizes to obtain a single visibility value for each model grid cell at each time step. More detailed descriptions of the PM calculation techniques used in CCTM can be found in Binkowski and Shankar (1995),Binkowski and Roselle (2003), and Byun and Schere (2006).

For easier comparison of CMAQ's output PM values with measurements, time-dependent cutoff fractions may be output by the model (e.g. Jiang et al., 2006). These include quantities for describing the fraction of each mode that would be categorized as PM2.5 (i.e. PM25AT, PM25AC, and PM25CO) and PM1.0 (i.e. PM1AT, PM1AC, and PM1CO) as well as the fraction of particles from each mode that would be detected by an AMS (i.e AMSAT, AMSAC, and AMSCO). . There is also a surface interaction module in the multipollutant version of CMAQ that calculates the flux of mercury to and from the surface (rather than just depositing mercury).

Further discussion on the scientific improvements to the CMAQ PM treatment is available in the release notes for each version of the model.

3.3.4 Clouds and aqueous-phase chemistry

See the CMAQv5.2 release notes for updates on the heterogeneous chemistry algorithms in CMAQ.

Clouds are an important component of air quality modeling and play a key role in aqueous chemical reactions, vertical mixing of pollutants, and removal of pollutants by wet deposition. Clouds also indirectly affect pollutant concentrations by altering the solar radiation, which in turn affects photochemical pollutants (such as ozone) and the flux of biogenic emissions. The cloud module in CMAQ performs several functions related to cloud physics and chemistry. Three types of clouds are modeled in CMAQ: sub-grid convective precipitating clouds, sub-grid nonprecipitating clouds, and grid-resolved clouds. The meteorological model provides information about grid-resolved clouds, with no additional cloud dynamics considered in CMAQ. For the two types of sub-grid clouds, the cloud module in CCTM vertically redistributes pollutants, calculates in-cloud and precipitation scavenging, performs aqueous

chemistry calculations, and accumulates wet deposition amounts. An important improvement in the CMAQv5 convective cloud mixing algorithm corrects a tendency to predict excessive transport from upper layers in the cloud to sub-cloud layers.

CMAQ's standard cloud chemistry treatment estimates sulfate production from five sulfur oxidation reactions, as well as secondary organic aerosol formation from the reaction of glyoxal and methylglyoxal with the hydroxyl radical. The distribution between gas and aqueous phases is determined by instantaneous Henry's law equilibrium, and the bisection method is used to estimate pH (and the distribution of ionic species) assuming electroneutrality. Beginning with CMAQv5.1, two additional cloud chemistry module options, AQCHEM-KMT and AQCHEM-KMTI, were made available along with standard AQCHEM (Fahey et al., 2017). These modules employ a Rosenbrock solver generated using the Kinetic PreProcessor (KPP), version 2.2.3 (Damian et al., 2002) to solve cloud chemistry, ionic dissociation, wet deposition, and kinetic mass transfer between the gas and aqueous phases (Schwartz, 1986). AQCHEM-KMTI also includes an expanded aqueous-phase chemical mechanism that treats SOA formation from biogenic-derived epoxides (Pye et al., 2013) in cloud, in addition to the standard sulfur and □-dicarbonyl oxidation reactions. In all cloud chemistry modules, the parameters for cation content of coarse species have been updated to be consistent with the rest of CMAQ.

3.3.5 Deposition

See the CMAQv5.2 release notes for updates on the air-surface exchange algorithms in CMAQ.

CMAQ optionally calculates the wet and dry deposition of chemical species. Information on the algorithms used can be found in Pleim and Ran (2011), Bash et al (2013), and Pleim et al (2013). For deposition to be considered in a model run, the entry in the namelist file must indicate a deposition surrogate species and a deposition factor. The default namelist files contain the standard configuration for known deposition of species. For wet deposition, the scavenging factor (SCAV_FAC) should be set to 1 and the surrogate species (SCAV_SUR) must be one of the chemicals listed in the HLCONST.F subroutine. If the chemical does not have an appropriate surrogate species listed in the HLCONST.F subroutine, one may be added to the model source code. For dry deposition, the deposition factor (DEPV_FAC) should be set to 1 and the deposition velcoity surrogate (DEPV_SUR) should be set to one of the species listed in the DEPVDEFN.F subroutine. The species listed in this file are further cross-referenced to a table in ASX_DATA_MOD.F where the diffusivity in air, reactivity, mesophyll resistance, LeBas molar volume, and wet surface scavenging surrogate are specified. The wet surface scavenging surrogate must be a chemical in the HLCONST.F subroutine and is typically the same species that is used for wet deposition of the chemical. If a proper deposition velocity surrogate species does not exist in the tables, one can be added to the model source code.

A runtime flag in the CMAQ model controls whether the bidirectional modeules for ammonia and mercury are invoked. The bidirectional modules simulate two-way exchange between the atmosphere and the surface for these species (as opposed to only deposition). The mercury bidirectional module (Bash 2010) is part of the CMAQv5 multipollutant configuration. To use the bidirectional option for ammonia, additional input file are required. The files are created from the Environmental Policy Integrated Climate (EPIC) model (Cooter et al., 2012). There are two time independent files which provide information on the soil and the landcover. A time dependent file contains information on fertilizer application method and amount.

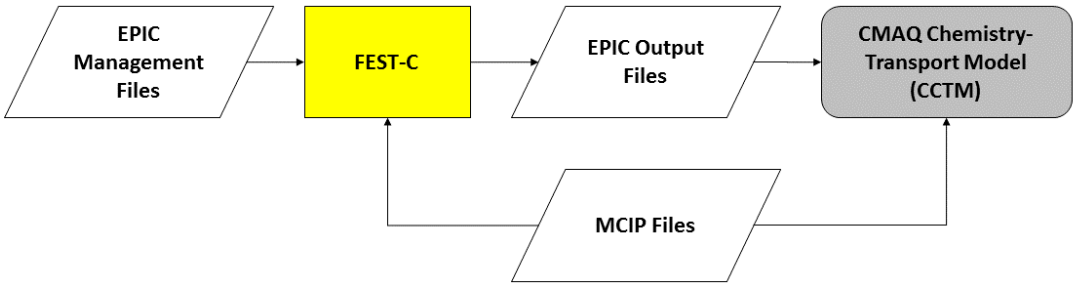


Figure 4□7. Data flow between EPIC, the meteorological model, and CMAQ from Cooter et al. (2012)

An additional configuration option related to deposition is the CMAQ_MOSAIC runtime option. This option outputs land use specific deposition velocities and fluxes.

In previous versions of CMAQ, the effects of HONO heterogeneous chemistry on deposition velocities were included. These effects were removed in CMAQv5.0.

3.3.6 Emissions

See the CMAQv5.2 release notes for updates on the emissions algorithms in CMAQ.

CMAQ includes several in-line options for calculating and processing emissions in the CCTM. The in-line emissions options in CMAQv5 include the following:

- The BEIS3 biogenic emissions model can be used to calculate emissions from vegetation and soils. This biogenic model is based on the same model that is included in SMOKE. User documentation for BEIS can be found in Chapter 6.17 of the SMOKE manual. The temporal allocation of the biogenic emissions in included in CMAQ. However, the calculation of the gridded normalized emissions for winter and summer is a time independent calculation and must be done with normbeis3 prior to running the inline biogenic option in CMAQ. The user must either provide a BIOSEASON file for simulations that are not summer only or winter only (e.g. multiple seasons, spring, fall) or set the SUMMER_YN flag to Y for summer or N for winter. Without the BIOSEASON file, all biogenic emissions will be calculated using summer factors or winter factors. Additional, when using the inline biogenic option, the user must point to the SOILOUT file from one day's simulation as the SOILINP file for the next day. The user must also decide

whether to write over SOILOUT files from previous days or create a uniquely named SOILOUT file for each day. The latter approach is recommended if the user wishes to retain the capability to restart simulations in the middle of a sequence of simulations.

- Plume rise can be calculated for large point sources. Plume rise can be calculated in-line within CMAQ provided the emission files have been processed with SMOKE for in-line processing. The NPTGRPS sets the number of "sectors" for which the user wishes to provide a stack_groups file and an inline emissions file. Optionally, the user can request 2 optional output diagnostic files that include a 3-D file the emissions with plume rise included and a layer fractions (PLAY) file that includes the fractional amount of emission per model layer.
- Windblown dust emissions can be estimated using meteorology and land-cover data
- Updated sea salt emissions. In AERO6 sea salt emissions in the accumulation mode are speciated into Na, Cl, SO4, Ca, K, and Mg. All cations in the coarse-mode sea salt (i.e., Na, Ca, K, and Mg) are lumped into a species called ASEACAT.

3.3.7 Process analysis

See the CMAQv5.2 release notes for updates on the process analysis algorithms in CMAQ.

Process analysis (PA) is a technique for separating out and quantifying the contributions of individual physical and chemical processes to the changes in the predicted concentrations of a pollutant. PA does not have to be activated in a CMAQ simulation but including PA provides additional information that can be useful in interpreting CMAQ results. PA has two components: Integrated process rate (IPR) analysis and Integrated Reaction Rate (IRR) analysis. IPR analysis quantifies the individual physical processes of advection, diffusion, emissions, dry deposition, aerosol processes, and cloud processes) and the overall impact of chemical processes. IRR analysis allows the output of individual chemical reaction rates or user-specified combinations of chemical reactions and species cycling.

PA variables are computed by saving the differential operators associated with each process or reaction, integrated over the model synchronization time step – the same variables that are used in solving the continuity equations within the model. For processes that are solved simultaneously in the same operator, PA uses mass balance to compute the contribution of each process.

As a tool for identifying the relative importance of individual chemical and physical processes, PA has man applications, including: - When attempting to identify the major contributors to the concentration of a chemical species, PA helps to unravel the large number of complex processes that control species concentrations. This is also useful for species that have both production and decay processes occurring in the same time step, because the final concentration may show little change, but individual decay and production rates may be large.
- To characterize the chemical "state" of a particular grid cell, IRR can be used to calculate quantifies such as the production of odd oxygen, the production of new radicals and the termination of radicals. (for example, see Tonnesen and Dennis, 2000). - As a tool for model development, PA can help evaluate the effect of modifications made to a model or process module - For QA purposes, PA can be used to help identify compensating or unresolved errors in the model or input data which may not be reflected in the total change in concentration. For example, if an error in the emissions input data causes the model to calculate negative concentration values in an intermediate step, this could be masked in the

final predicted concentrations if compensated for by larger positive values resulting from the chemistry calculations.

If the user activates PA during CMAQ runtime (CTM_PROCAN=Y), the PA input file (PACM_INFILE) specifies whether IPR, IRR or both analyses are performed, and defines which variables are required for each analysis. The IRR parameters are highly customizable and can be easily modified for new chemical mechanisms, but must be checked carefully before running to ensure that they correspond to the mechanism being used. The PA_REPORT output file should always be reviewed to verify that the calculations are being performed correctly. Note that while the IPR option can be run with any of the chemical solvers, use of IRR in CMAQ requires either the Rosenbrock or the SMVGEAR solvers.

3.4 The CMAQ User Interface

The CMAQ user interface that is distributed with the model source code consists of a series of C-shell scripts for building and running the various CMAQ programs on Linux operating systems. These scripts function primarily to set environment variables that are required by the program Bldmake or by the CMAQ program executables. The scripts can be adapted to work with any Linux shell scripting language (e.g., Bash, Bourne).

CMAQ source code can be viewed and downloaded from the EPA CMAQ GitHub repository. Alternatively, tarballs can be downloaded from the CMAS Center website. Each of CMAQ's programs has separate build and run scripts. The build scripts are used to compile the source code into binary executables. The run scripts are used to set the required environment variables and execute the CMAQ programs. The user can manipulate the CMAQ scripts using a Linux text editor such as emacs, gedit, nano, or vi. There are certain options that need to be set at compilation, and some that can be set before running a simulation. Details about using the scripts to build and run CMAQ are described in Section 5, with further details in Section 7.

The CMAS Center currently supports CMAQ on Linux systems using the Gnu, Portland Group, and Intel Fortran compilers. Community members are encouraged to share their experiences porting CMAQ to other operating systems and compilers.

CMAQ users are strongly urged to use the *same* Fortran compiler for *all* components of the CMAQ system, including the netCDF and I/O API libraries on which CMAQ depends.

3.5 References for Chapter 4: Science Overview

Bash, J. O., E. J. Cooter, R. L. Dennis, J. T. Walker, and J. E. Pleim, 2013: Evaluation of a regional air-quality model with bidirectional NH3 exchange coupled to an agroecosystem model. *Biogeosciences*, **10**, 1635-1645.

Bash, J.O., 2010, Description and initial simulation of a dynamic bi-directional air-surface exchange model for mercury in CMAQ, *J. Geophys. Res.*, **115**, D06305

Binkowski, F.S., and U. Shankar, 1995: The Regional Particulate Model: Part I. Model description and preliminary results. *J. Geophys. Res.*, **100**, 26 191–26 209.

Binkowski, F. S., and S. J. Roselle, 2003: Models-3 Community Multiscale Air Quality (CMAQ) model aerosol component. 1. Model description. "J. Geophys. Res., **108**, 4183, doi:10.1029/2001JD001409.

Binkowski, F.S, , S. Arunachalam, Z. Adelman, and J. Pinto, Examining photolysis rates with a prototype on-line photolysis module in CMAQ, 2007, *J. Appl. Meteor. and Clim.*. **46**, 1252-1256, doi: 10.1175/JAM2531.1

Byun, D. W., 1999: Dynamically consistent formulations in meteorological and air quality models for Multiscale atmospheric studies. Part I: Governing equations in a generalized coordinate system. *J. Atmos. Sci*., **56**, 3789–3807.

Byun, D. W., and J. K. S. Ching, 1999: Science Algorithms of the EPA Models-3 Community Multiscale Air Quality (CMAQ) Modeling System. U. S. Environmental Protection Agency Rep. EPA□600/R□99/030, 727 pp. [Available from Office of Research and Development, EPA, Washington, DC 20460.]

Byun, D., and K. L. Schere, 2006: Review of the governing equations, computational algorithms, and other components of the Models-3 Community Multiscale Air Quality (CMAQ) modeling system. *Appl. Mech. Rev.*, **59**, 51–77. doi:10.1115/1.2128636

Carlton, A.G., B. J. Turpin, K. Altieri, S. Seitzinger, R. Mathur, S. Roselle, R. J. Weber, 2008. CMAQ model performance enhanced when in-cloud SOA is included: comparisons of OC predictions with measurements, Environ. Sci. Technol. , 42, (23), 8799-8802,

Carlton, A.G., P.V. Bhave, S.L. Napelenok, E.O. Edney, G. Sarwar, R.W. Pinder, G.A. Pouliot, M. Houyoux, 2010: Model Representation of Secondary Organic Aerosol in CMAQv4.7. *Env. Sci. & Techno*. **44 (22)**, 8553-8560.

Chang, J. S., P. B. Middleton, W. R. Stockwell, C. J. Walcek, J. E. Pleim, H. H. Lansford, F. S. Binkowski, S. Madronich, N. L. Seaman, D. R. Stauffer, D. Byun, J. N. McHenry, P. J. Samson, and H. Hass, 1990: The regional acid deposition model and engineering model, *Acidic Deposition: State of Science and Technology*, Report 4, National Acid Precipitation Assessment Program.

Colella, P., and P. L. Woodward, 1984: The piecewise parabolic method (PPM) for gas-dynamical simulations. *J. Comput. Phys*.," **54**, 174–201.

Cooter, E.J., Bash, J.O., Benson V., Ran, L.-M., 2012, Linking agricultural management and air-quality models for regional to national-scale nitrogen deposition assessments, *Biogeosciences*, **9**, 4023-4035

Damian, V., A. Sandu, M. Damian, F. Potra, and G.R. Carmichael, 2002: The Kinetic PreProcessor KPP – A Software Environment for Solving Chemical Kinetics, *Computers and Chemical Engineering*, **26**, 1567-1579.

Donahue, N. M., et al. 2012: A two-dimensional volatility basis set – Part 2: Diagnostics of organic-aerosol evolution. *Atmospheric Chemistry and Physics,* **12(2)**, 615-634.

Edney, E. O., T. E. Kleindienst, M. Lewandowski, and J. H. Offenberg, 2007. Updated SOA chemical mechanism for the Community Multi-Scale Air Quality model, EPA 600/X-07/025, U.S. EPA, Research Triangle Park, NC.

Elterman, L., R. Wexler, and D. T. Chang, 1969: Features of tropospheric and stratospheric dust. *Appl. Optics*, **8**, 893–903.

Fahey, K.M., A.G. Carlton, H.O.T. Pye, J. Baek, W.T. Hutzell, C.O. Stanier, K.R. Baker, K.W. Appel, M. Jaoui, J.H. Offenberg, 2017: A framework for expanding aqueous chemistry in the Community Multiscale Air Quality (CMAQ) model version 5.1, *Geosci. Model Dev.*, **10**, 1587-1605.

Fountoukis, C and A. Nenes, 2007: ISORROPIA II: A computational efficient thermodynamic equilibrium model for K+-Ca2+-Mg2+-NH4+-Na+-SO42—NO3—Cl—H2O aerosols, "Atmos. Chem. And Phys., **7**, 4639-4659.

Hertel O., R. Berkowicz, J. Christensen, and O. Hov, 1993: Test of two numerical schemes for use in atmospheric transport-chemistry models. *Atmos. Environ.*, **27A**, 2591–2611

Jacobson, M., and R. P. Turco, 1994: SMVGEAR: A sparse-matrix, vectorized Gear code for atmospheric models. *Atmos. Environ.*, **28**, 2991–3003.

Jiang, W., S. Smyth, É. Giroux, H. Roth, and D. Yin, 2006: Differences between CMAQ fine mode particle and PM2.5 concentrations and their impact on model performance evaluation in the lower Fraser valley. *Atmos. Environ*., **40**, 4973–4985.

Murphy, B. N., et al., 2017: Semivolatile POA and parameterized total combustion SOA in CMAQv5.2: impacts on source strength and partitioning. *Atmospheric Chemistry and Physics Discussions,* 2017: 1-44.

Otte, T. L., and J. E. Pleim, 2010: The Meteorology-Chemistry Interface Processor (MCIP) for the CMAQ modeling system: updates through MCIPv3.4.1. *Geoscientific Model Development*, **3**, 243-256.

Pleim, J.E., J. O. Bash, J. T. Walker, and E. J. Cooter, 2013. *J. Geophys. Res.*, **118**, 3794-3806.

Pleim, J. E., and J. S. Chang, 1992: A non□local closure model in the convective boundary layer. *Atmos. Environ.*, **26A**, 965–981.

Pleim, J.E., and L. Ran, 2011: Surface Flux Modeling for Air Quality Applications. *Atmosphere*, **2**, 271-302.

Pleim, J. E., A. Xiu, P. L. Finkelstein, and T. L. Otte, 2001: A coupled land-surface and dry deposition model and comparison to field measurements of surface heat, moisture, and ozone fluxes. *Water Air Soil Pollut. Focus*, **1**, 243–252.

Pleim, J, 2007: A combined local and nonlocal closure model for the atmospheric boundary layer. Part I: model description and testing, *J. of Appl Met. and Climatology*, **46**, 1383-1395

Pye, H.O.T., R.W. Pinder, I.R. Piletic, Y. Xie, S.L. Capps, Y.H. Lin, J.D. Surratt, Z.F. Zhang, A. Gold, D.J. Luecken, W.T. Hutzell, M. Jaoui, J.H. Offenberg, T.E. Kleindienst, M. Lewandowski, E.O. Edney, 2013: Epoxide pathways improve model predictions of isoprene markers and reveal key role of acidity in aerosol formation, *Environ. Sci. Technol.*, **47(19)**, 11056-11064.

Reff, A., P.V. Bhave, H. Simon, T.G. Pace, G.A. Pouliot, J.D. Mobley, M. Houyoux, 2009: Emissions inventory of PM2.5 trace elements across the United States, *Env. Sci. & Technol*. **43**, 5790-5796.

Sandu, A., J. G. Verwer, J. G., Blom, E. J. Spee, G. R. Carmichael, and F. A. Potra, 1997: Benchmarking stiff ODE solvers for atmospheric chemistry problems. II: Rosenbrock solvers. *Atmos. Environ.*, **31**, 3459–3472.

Schwartz, S.E., 1986: Mass transport considerations pertinent to aqueous-phase reactions of gases in liquid water clouds. In Chemistry of multiphase atmospheric systems, NATO ASI Series, *G6*, 415-471.

Tonnesen, G.S., Dennis, R.L., 2000: Analysis of radical propagation efficiency to assess ozone sensitivity to hydrocarbons and NO x : 1. Local indicators of instantaneous odd oxygen production sensitivity, *J. Geophys. Res.*, **105(D7)**, 9213-9225.

National Oceanic and Atmospheric Administration, 1976: *U.S. Standard Atmosphere*, U.S. Government Printing Office, Washington, DC, NOAA□S/T76□1562.

4 CMAQ Installation and System Requirements

This section describes how to set up and install CMAQ on a Linux system. The installation instructions in this section guide the user through obtaining the CMAQ source code and installing it on your system. Brief instructions for running the CMAQ benchmark case and benchmarking the model are also addressed. Here, the term "benchmarking" refers to the process of verifying that a model has installed correctly on a new computer. CMAQ is distributed with a reference dataset that can be used to benchmark the CMAQ installation; in the distribution, output data from CMAQ are bundled with the input data (including emissions and meteorology) that can be used to reproduce the reference results.

After benchmarking has been successfully completed, the CMAQ system can be configured for other simulations. The same steps that are required to build the model for the benchmark case apply to building it for new simulations. Configuring CMAQ for new applications is covered in Chapter 10.

4.1 System Recommendations

All of the CMAQ programs are written in Fortran and are optimized for use on computers running a version of the Linux operating system (OS). Most personal computers (PCs) running a Linux OS are sufficiently powerful to handle basic CMAQ applications. However, to use CMAQ in a production environment where multiple iterations of the model will be executed for different spatial domains and/or emissions control strategies, either a cluster of multiprocessor PCs on a high-end network or an expandable rack-mounted Linux server is recommended.

CMAQ is distributed and supported for executing on Linux operating systems with the Intel Fortran, Portland Group Fortran (PGF), or Gnu Fortran compilers. CMAQ can be ported to most computers running Linux. Documented configurations include the SGI Altix, Red Hat Enterprise, Fedora, Ubuntu, Mandrake, MacOSX, and Suse operating systems. In addition to the Intel and PGF compilers, CMAQ has been built with Sun and Absoft compilers. Information about these ports and up-to-date hardware recommendations are available through the CMAS Center web site.

4.1.1 Hardware

The minimum hardware requirements for running the CMAQ benchmark case are:

- Linux PC with a single processor

- 1 GB RAM
- 100 GB hard drive storage

Recommendations on production-level hardware configurations for CMAQ are available on the CMAS Center Hardware Blog.

4.1.2 Software

CMAQ requires all of the programs listed in Table 5□1. This list includes the programs distributed with CMAQ. Note that CMAQv5.0 and greater requires I/O API version 3.1. Newer version of CMAQ will not compile with earlier versions of the I/O API library. Table 5□2 lists additional utility software that is not required for running CMAQ, but is useful for model diagnostics and evaluation.

Table 5□1. Software required for running CMAQ

Software	Description	Source
CMAQ Programs		
Bldmake	Executable builder for source code compilation	Contained in the standard CMAQ distribution available at <https://github.com/USEPA/CMAQ/tree/5.2> Release notes and documentation available at <https://github.com/USEPA/CMAQ/tree/5.2/CCTM/d
ICON	Initial conditions preprocessor	“
BCON	Boundary conditions preprocessor	“
MCIP	Meteorology-Chemistry Interface Processor	“
CCTM	CMAQ Chemistry-Transport Model	“
CHEMMECH	Chemical mechanism compiler for modifying or adding reactions to the CMAQ chemistry	“
CREATE_EBI	EBI Chemical solver source code generator	“
Compilers		
IFORT	Intel Fortran 90 compiler	<http://www.intel.com>
PGF90	Portland Group Fortran 90 compiler	<http://www.pgroup.com/>
GFORT	Gnu Fortran compiler	<http://gcc.gnu.org/fortran/>
GCC	Gnu C compiler	<http://gcc.gnu.org/>
Code Libraries		
OpenMPI	Library for the message passing interface; used for multiprocessor CMAQ simulations	<https://www.open-mpi.org>

Software	Description	Source
MPICH	Library for the message passing interface; used for multiprocessor CMAQ simulations	<http://www.mcs.anl.gov/research/projects/mpich2/>
netCDF	Network Common Data Form library for controlling CMAQ file formats*	<http://www.unidata.ucar.edu/software/netcdf/>
I/O API	Input/Output Application Programming Interface for controlling internal and external communications	<https://www.cmascenter.org/ioapi/>
LAPACK	Linear algebra packages for use with the bidirectional mercury module	<http://www.netlib.org/lapack/>
BLAS	Basic Linear Algebra Subprograms	<http://netlib.org/blas/>
For versions prior to CMAQ-5.0.2		
CVS	Concurrent Versions System for managing the distributed archive of the CMAQ source code	<http://ximbiot.com/cvs/cvshome/> or your host's package management system

Note: CMAQ Output File Format CMAQ uses a modified version of the netCDF file format. Although CMAQ output is described as being in the netCDF format, it is actually a hybrid format of the I/O API and the netCDF.

Table 5□2. Optional support software for CMAQ

Software	Description	Source
Evaluation and visual-ization tools		
VERDI	Visualization Environment for Rich Data Interpretation for graphical analysis of netCDF gridded data	<http://www.verdi-tool.org>
PAVE	Package for Analysis and Visualization of Environmental data for graphical analysis of netCDF gridded data	<http://www.cmascenter.org>
IDV	Integrated Data Viewer for 3-D graphical analysis of netCDF gridded data	<http://www.unidata.ucar.edu/software/idv/>

Software	Description	Source
I/O API Tools	Postprocessing tools for manipulating data in the I/O API/netCDF format	<https://www.cmascenter.org/ioapi/>
netCDF Tools	Postprocessing tools for manipulating data in the netCDF format	<http://my.unidata.ucar.edu/content/software/netcdf/index.html>
Source code diagnostics		
GDB	Gnu Fortran debugger	<https://www.sourceware.org/gdb/>
PGDBG	Portland Group Fortran debugger	<http://www.pgroup.com/>
PGPROF	Portland Group Fortran code profiler	<http://www.pgroup.com/>
IDB	Intel Fortran debugger	<https://software.intel.com/en-us/articles/idb-linux>

4.2 Installing and Compiling CMAQ Source Code

Several steps are required to prepare your Linux system for compiling and running CMAQ. The general sequence for installing CMAQ, including the required support software and libraries is listed here.

1. Check for Fortran and C compilers on your Linux system. Install if they are not available.

2. Install Git (or CVS for older versions of CMAQ).
3. Download the source and install the I/O API and netCDF libraries. Follow the instructions in the documentation for each library on how to build them for your Linux system. Note: It is highly recommended that you use the same compiler for these libraries as you will use to build the CMAQ programs.
4. Install a Message Passing Interface (MPI) library on your Linux system.
5. Download the CMAQ source code and scripts from either the EPA GitHub Repository or the CMAS Center (<http://www.cmascenter.org>). After registering to download CMAQ on the CMAS Center Software Clearinghouse, users are redirected to a page that contains links to download Linux tar files of the CMAQ code, scripts, and benchmark data along with various documents describing the installation and execution processes. *Note that GitHub only provides access to source codes and scripts. Benchmark input and output data may only be downloaded from the CMAS Center.*

4.2.1 Distribution contents

The CMAQ distribution includes the following components:

- MODELS - CMAQ source code and libraries

- SCRIPTS - C-shell scripts to build and execute the CMAQ models
- INPUT DATA – datasets necessary to run the tutorial/benchmark case.
- REF DATA – reference output data to compare with datasets produced by the tutorial on a Linux workstation

Starting with CMAQv5.2, the structure of the CMAQ distribution includes: - CCTM - Chemistry Transport Model source code and scripts - PREP - Input pre-processor (e.g., ICON, BCON, MCIP) source code and scripts - UTIL - Utility software (e.g., BLDMAKE, CHEMMECH, NML) source code and scripts - POST - Post-processing and analysis software (e.g., COMBINE, HR2DAY, BLDOVERLAY) source code and scripts - DOCS - This User's Manual, tutorials, and developers guidance

4.2.2 Notes on the CMAQ directory structure

The CMAQ installation includes a dataset for benchmarking the modeling system. Unpacking the various tar files of the distribution in a `CMAQ_HOME` (formerly M3HOME prior to CMAQv5.2) directory installs the CMAQ source code, scripts, and benchmark data files in a directory structure recognized by the default run and build scripts. The `CMAQ_HOME` directory is an arbitrary base location of the CMAQ installation on your Linux system for a specific application. It's up to the user to decide where to install CMAQ and to assign this location to the `CMAQ_HOME` environment variable in the CMAQ build scripts.

Under `CMAQ_HOME`, the `data` directory contains the input and output data for the model, and the `lib` directory contains the compiled binary library files required to build the CMAQ executables. The CMAQ scripts use the following environment variables to alias the locations of these directories:

`CMAQ_LIB = $CMAQ_HOME/lib` (M3LIB before CMAQv5.2) `CMAQ_DATA = $CMAQ_HOME/data` (M3DATA before CMAQv5.2)

The CMAQ scripts require users to select only the location of the `CMAQ_HOME` directory; the other CMAQ directories are referenced relative to `CMAQ_HOME`. While this directory structure is convenient for the benchmark case and most CMAQ applications, other configurations are possible. Detailed instructions for installing and compiling CMAQ are contained in the next section.

4.2.3 Configuring your system for compiling CMAQ

Compiler flag consistency between the Fortran and C compilers used to build netCDF and I/O API is critical for building library files compatible with CMAQ. Table 5-3 lists the suggested compilation options for building netCDF and I/O API libraries that are compatible with CMAQ. Refer to the documentation for these libraries for additional information on installation and compiling.

Table 5□3. NetCDF and I/O API compilation options for CMAQ

Library Type	Intel Fortran	PGI Fortran	Gnu Fortran
netCDF	`CC = icc CPPFLAGS = -DNDEBUG -DpgiFortran CFLAGS = -g -O FC = ifort F77 = ifortFFLAGS = -O2 -mp -recursive CXX = icpc`	`CC = gcc CPPFLAGS = -DNDEBUG -DpgiFortranCFLAGS = -OFC = pgf90FFLAGS = -O -w CXX = g++`	`CC = gcc CPPFLAGS = -DNDEBUG -DgFortran CFLAGS = -OFC = gfortranFFLAGS = -O -wCXX = g++`
I/O API 32-bit	BIN = Linux2_x86ifort	BIN = Linux2_x86pg_pgcc_nomp	N/A
I/O API 64-bit	BIN = Linux2_x86_64ifort	BIN = Linux2_x86_64pg_pgcc_nomp	BIN = Linux2_x86_64gfort

4.2.3.1 config_cmaq.csh

Consistency of configuration variables is critical for building CMAQ itself, not just its libraries. Accordingly CMAQ includes the configuration script `config_cmaq.csh` to help enforce consistent environment settings for CMAQ and its associated libraries Table 5-4 lists the `config_cmaq.csh` variables defined for the build process and suggests values to which to set those variables.

Note that for multiprocessor applications it is recommended that the Fortran MPI wrapper script `mpif90` be specified for the Fortran compiler (myFC). Using this script, instead of a direct call to the Fortran compiler, will ensure that the full suite of MPI components (libraries and include files) for the compiler are included in the parallel build.

Table 5□4. config_cmaq.csh configuration variables

Variable Name	Suggested Value
`CMAQ_HOME`	The central CMAQ installation directory. For example, if you installed the CMAQ source code in the directory `/home/user/CMAQ` set CMAQ_HOME with `export CMAQ_HOME=/home/user/CMAQ` for bash or `setenv CMAQ_HOME /home/user/CMAQ` for csh; note that this variable is M3HOME prior to CMAQv5.2
`CMAQ_DATA`	Automatically set by config_cmaq.csh; note that this variable is M3DATA prior to CMAQv5.2
`CMAQ_LIB`	Automatically set by config_cmaq.csh; note that this variable is M3LIB prior to CMAQv5.2
`M3MODEL`	Automatically set by config_cmaq.csh; deprecated in CMAQv5.2
`compiler`	Set the Fortran compiler type that you will use to compile CMAQ; choices are intel, pgi, or gcc
`IOAPI`	Location of the I/O API library installation on your Linux system
`NETCDF`	Location of the netCDF installation on your Linux system
`MPI`	Location of the Message Passing Interface installation on your Linux system

Variable Name	Suggested Value
`netcdf_lib`	Name of the netCDF library on your system; set to "-lnetcdf" for versions < 4.2.0, "-lnetcdff -lnetcdf" for version 4.2.0 and later
`ioapi_lib`	Name of the I/O API libraryar on your system; set to "-lioapi"
`pnetcdf_lib`	Name of the parallel netCDF library on your system; set to "-lpnetcdf"
`mpi_lib`	Name of the MPI library on your system; set to "-lmpich" for MVAPICH, "-lmpi" for OpenMPI
`myFC`	Set to match the `FC` (Fortran compiler) you used to compile netCDF
`myCC`	Set to match the `CC` (C compiler) you used to compile netCDF
`myFFLAGS`	Fixed-format Fortran compiler optimization flags for your Linux system; suggested values for CMAQ are in the distributed script
`myCFLAGS`	C compiler optimization flags for your Linux system; suggested values for CMAQ are in the distributed script
`myFRFLAGS`	Free form-format Fortran compiler optimization flags for your Linux system; suggested values for CMAQ are in the distributed script
`MPI_INC`	Set to the path to your MPI library INCLUDE files, e.g. `$M3LIB/mpich/include`
`extra_lib`	Set to other libraries required for compiling on your Linux system; users will likely need to change this setting in the distributed script for portability to their system.
`EXEC_ID`	build tag, should be automatically set by config_cmaq.csh

4.2.4 Installing CMAQ on your system

Use the following steps to install CMAQ (with examples using a C-shell environment, a Red Hat Linux system, and the Portland Group Fortran compiler) on a Linux system. CMAQ is not distributed with scripts for installing on Windows or MacOSX.

4.2.4.1 Obtain CMAQ source codes

CMAQ source code can be installed either using git or from tarballs downloaded from the CMAS Center. Both options are described here.

4.2.4.1.1 Git Installation

In the directory where you would like to install CMAQ, issue the following command to clone the official EPA GitHub repository for CMAQv5.2:

```
git clone -b 5.2 https://github.com/USEPA/CMAQ
```

4.2.4.1.2 Tarball Installation

Tarballs of the CMAQ source code are available from both the public GitHub repository and the CMAS Center Software Clearinghouse. In addition to the source code, reference input/output data for testing

the installation of the software are available only from the CMAS Center; *data are not available through GitHub*. You must register/login to access the source codes and data from the CMAS Center.

In the directory where you would like to install CMAQ, unzip, and untar the model distribution file:

```
tar xvzf CMAQv5.2.tar.gz
```

Both of these installation options will produce the following subdirectories on your Linux system:

```
CMAQv5.2/CCTM
CMAQv5.2/PREP
CMAQv5.2/POST
CMAQv5.2/UTIL
CMAQv5.2/DOCS
```

4.2.4.2 Set up the central CMAQ configuration script

Edit the file `CMAQv5.2/config_cmaq.csh` to configure the CMAQ installation for the local computing architecture and compilers.

The first step in editing `CMAQv5.2/config_cmaq.csh` is to set the environment variable `CMAQ_HOME` to point the installation directory location of CMAQ on your Linux system.

Under the "architecture & compiler specific settings" section of the script set the compiler and libraries that you will use to build the CMAQ executables. There are three example compiler configurations built into the config_cmaq.csh script: (1) Intel Fortran (intel), (2) Portland Group Fortran (pgi), and (3) Gnu Fortran (gcc). Set the script variable `compiler` for one of the supported compilers.

Set the names of the I/O API and netCDF libraries using the `ioapi_lib` and `netcdf_lib` script variables. You may also choose to set the pnetcdf_lib if you are using parallel netCDF. Note that for version 4.2 of the netCDF and later, you need to provide both the C and Fortran versions of the netCDF library, in this order: `-lnetcdff -lnetcdf`

Set the name of the MPI library using the `mpi` script variable. For MVAPICH use `-lmpich`; for openMPI use `-lmpi`. If you are using another MPI library, use the variable `mpi` to set the name of the library.

Save and exit from the config_cmaq.csh file and use the source command to invoke the settings in the file:

```
source CMAQv5.2/config_cmaq.csh
```

When you source the config_cmaq.csh script the CMAQ directories, including all required libraries will be installed in the CMAQ working directory. If you encounter errors about libraries not being found, check the settings of the config_cmaq.csh script variables IOAPI, NETCDF, or MPI to ensure that they are correctly point to the locations of these libraries on your Linux system.

4.2.4.3 Install the CMAQ input reference/benchmark data

After you have downloaded the CMAQ reference data from the CMAS Center Software Clearinghouse, navigate to the `$CMAQ_HOME` directory, unzip and untar the `CMAQv5.2.DATA.tar.gz` file:

```
cd $CMAQ_HOME
tar xvzf CMAQv5.2.DATA.tar.gz
```

This will produce the following subdirectories:

```
CMAQv5.2/data/
  bcon/
  bidi/
  cctm/
  crop/
  dust/
  emis/
  icon/
  lightning/
  mcip/
  ocean/
  raw/
  phot/
  raw/
```

4.2.4.4 Install the CMAQ libraries

When you sourced the config_cmaq.csh script above, the directory `CMAQ_LIB` was automatically created along with symbolic links to the I/O API, netCDF, and MPI libraries on your Linux system. The library and include files that were installed in this directory are based on the locations you specified in the config_cmaq.csh script. The locations of the libraries in the CMAQ working directory will be based on the compiler that you selected for your CMAQ build. For example, if you are running on an x86_64 architecture with the Portland Group Fortran compiler, the config_cmaq.csh script will set `CMAQ_LIB` to `$CMAQ_HOME/lib/x86_64/pgi`.

4.2.4.5 Compiling CMAQ

For all CMAQ programs (other than MCIP), the program Bldmake is used to compile the source code into executables. The first step in the compilation of CMAQ is to compile Bldmake. Bldmake will then be used to compile the CMAQ programs. *Note that the compiler paths and flags are all set in the config_cmaq.csh script and then passed along to the build scripts*. None of the CMAQ build scripts contain compiler settings. Instead, the build scripts for each program reference the config_cmaq.csh script using the Linux command "source". See Table 5-4 for a description of the compilation flags in the config_cmaq.csh script.

Before proceeding confirm that you have run the command `source config_cmaq.csh`

All of the CMAQ programs other than CCTM are run in single-processor mode. CCTM may be run either in single-processor (serial) mode or in parallel on multiple processors. Program-specific compilation instructions are provided below. These compilation instructions are for building executables for simulating the test data sets distributed with CMAQ. Additional information about the configuration options for the various CMAQ programs is provided in Chapter 4 and Chapter 7.

4.2.4.5.1 Compile Bldmake

```
As of CMAQv5.2, this is done automatically as part of building bcon, icon, or cctm.
```

4.2.4.5.2 Compile the CMAQ programs

Create the model executables for ICON, BCON, MCIP, and CCTM.

ICON and BCON can be configured for different chemical mechanisms and for different kinds of input data. The configuration options for ICON and BCON are discussed in detail in Chapter 7.

Use the following commands to compile ICON and BCON:

```
cd $CMAQ_HOME/PREP/icon
bldit_icon.csh |& tee build_icon.log

cd $CMAQ_HOME/PREP/bcon
bldit_bcon.csh |& tee build.bcon.log
```

Like the program Bldmake, MCIP is compiled using a Fortran Makefile.

Use the following commands to compile MCIP:

```
cd $CMAQ_HOME/PREP/mcip/src
source ../../../config_cmaq.csh
make |& tee make.mcip.log
```

The CCTM has multiple configuration options that can be changed to optimize model performance for different applications. In addition to selecting the chemical mechanism to model gas-phase chemistry, the user can also select from several different science modules. The science configuration options for CCTM are discussed in detail in Chapter 4 and Chapter 7. The distribution CCTM build script is configured to create a multiprocessor executable for the installation test simulation. For multiprocessor applications, CMAQ uses the message passing interface (MPI) to manage communication between processors in a clustered multiprocessor computing environment. The location of the MPI include and library files on your Linux system are specified in the config_cmaq.csh script.

For single-processor (serial) systems, configure the CCTM build script to create a single-processor executable by commenting out the line that activates the variable "ParOpt" of the CCTM build script. Use the following commands to compile CCTM:

```
cd $CMAQ_HOME/CCTM/scripts
bldit_cctm.csh |& tee build_cctm.log
```

Although not used for the installation test simulation, the program PROCAN can also be compiled using Bldmake. The programs CHEMMECH and CALMAP are also not needed for the test simulation but can be compiled using Makefiles.

4.2.5 Running the CMAQ Installation Test Simulation

After successfully compiling the various CMAQ programs, use the distributed run scripts to generate the CCTM input files and then to run CCTM for the CMAQ benchmark case. CCTM must be run last in the simulation sequence; MCIP must be run first. Note however that CMAQ-ready meteorology data are distributed with the CMAQ test case, which means that MCIP does not actually need to be run to test the model installation. With the exception of MCIP, there are no dependencies among the other CMAQ programs, so they can be run in any order to create input data for CCTM.

To run the test simulation for the various CMAQ programs, change directories to the location of each program and execute the run script.

Run ICON to produce initial conditions:

```
cd $CMAQ_HOME/PREP/icon
./run_icon.csh |& tee run_icon.log
```

Run BCON to produce boundary conditions:

```
cd $CMAQ_HOME/PREP/bcon
./run_bcon.csh |& tee run.bcon.log
```

Check the ICON and BCON log file to ensure that the programs completed successfully.

The CCTM is configured by default to run in multiprocessor mode. This mode requires run time settings that specify the number of processors to allocate to the simulation and the location of the MPI initialization command (mpirun) on your system. Set the number of processors to use for the simulation by setting the NPROCS environment variable in the run.cctm script. You must also set the domain decomposition configuration by setting the variable NPCOL_NPROW. The number of processors (NPROCS) must be equal to the product of the two values selected for NPCOL_NPROW. For example, if you have a system with six processors available to run CMAQ, set NPROCS to 6 and NPCOL_NPROW equal to “3 2”.

For an MPI configuration with 6 processors,

```
setenv NPROCS 6
setenv NPCOL_NPROW “3 2”
```

Most clustered multiprocessor systems require a command to start the MPI run-time environment. The default CCTM run script uses the *mpirun* command. Consult your system administrator to find out how to invoke MPI when running multiprocessor applications. For single-processor computing, set NPROCS to 1 and NPCOL_NPROW to “1 1”

For single-processor computing,

```
setenv NPROCS 1
setenv NPCOL_NPROW to “1 1"
```

After configuring the MPI settings for your Linux system, run the CCTM:

```
cd $CMAQ_HOME/CCTM/scripts
./run_cctm.csh |& tee run_cctm.log
```

4.3 Benchmarking

Benchmarking is the process of confirming that the model source code compiles and executes correctly on a new computer system. CMAQ should be benchmarked on a computing system before the model is used for research or regulatory applications on that system. The purpose of benchmarking is to ensure that there are no inconsistencies introduced into the model solution from local system parameters, such as compilers, processors, or operating systems. While differences are expected in the CMAQ results produced by different operating systems, hardware, and compilers, these differences should be small and within the numerical error of the model. Input and output reference data are packaged with CMAQ to use for benchmarking the model on new systems. After running the test case packaged with the source code, compare the results against the reference data provided in the CMAQ distribution.

4.3.0.1 CMAQ benchmark parameters

The CMAQ benchmark test case is a single day simulation for July 1, 2011 on a 72 column x 100 row x 35 layer 12-km resolution domain over California (CALNEX12 domain). The CCTM configuration parameters for the benchmark test case include the following:

- Multiprocessor simulation
- Horizontal advection: Yamo
- Vertical advection: WRF
- Horizontal diffusion: Multiscale
- Vertical diffusion: ACM2
- Deposition: M3Dry
- Chemistry solver: EBI
- Aerosol module: AERO6
- Cloud module: ACM
- Mechanism: cb05e51_ae6_aq
- Lightning NOx emissions calculated with hourly NLDN strike data
- Dynamic vertical diffusivity
- In-line deposition velocities
- Surface HONO interaction
- In-line biogenic emissions
- In-line plume rise
- In-line windblown dust emissions
- Bi-directional ammonia and mercury fluxes

The system configuration parameters used to generate the benchmark reference data include the following:

- Hardware: Dell C6100 server, 2.93 GHz Intel processor, 12M L3 cache (Model X5670), 48 GM memory
- Operating System: RHEL 5.6
- Compiler: Intel 15.0
- MPI: MVAPICH 2.1.7
- 12 processors

4.3.0.2 CMAQ Benchmark Results

After completing the CMAQ benchmark case, the CCTM output files can be compared with the reference datasets provided in the CMAQ distribution. The reference data for CMAQ are available from the CMAS Center Software Clearinghouse. The reference data may be compared to the results from your simulation using tile plots of differences, grid-cell statistics (minimum/maximum differences), and domain-wide statistics. See Chapter 12 for a list of analysis tools that are available for comparing two model simulations.

Domain-wide differences between the reference data and your simulation results for each model species and each simulation time step (hourly) of less than 1% indicate a successful benchmarking of the software on a Linux system. While larger differences may indicate a problem with the installation, they may also point to a configuration discrepancy in the benchmark simulation. Review the compiler optimization flags and the CCTM configuration that you used to resolve differences with the reference data.

Support for CMAQ is available from the CMAS Center (see Chapter 13).

5 Required Libraries

The CMAQ programs require a set of third-party libraries that must be installed on the users system before CMAQ can be compiled and run. These libraries control the data flow through CMAQ, define the binary file formats used by the CMAQ input and output files, and control how CMAQ functions in a multiple-processor computing environment. The Input/Output Applications Programming Interface (I/O API) and the Network Common Data Form (netCDF) are required for all applications of CMAQ. The Message Passing Interface (MPI) is only required for multiple-processor applications of CCTM. Brief descriptions of these three libraries are provided in this section. For additional information, including how to compile and configure these libraries, refer to the documentation associated with each library.

5.1 Input/Output Applications Programming Interface (I/O API)

The Models-3 Input/Output Applications Programming Interface (I/O API) is an environmental software development library that provides an interface with the data involved in CMAQ applications (Coats, 2005). The I/O API is the core input/output framework of the CMAQ programs, providing a set of commonly used subroutines for passing information between source code modules and for reading and writing data files. Users should download the latest code for the I/O API from the website. In

addition to providing the input/output framework for CMAQ, the I/O API forms part of the binary file format used by the CMAQ programs. ***Starting with CMAQ version 5.0, the I/O API version 3.1 or newer is required to compile and run CMAQ.***

The CMAQ input and output files use a hybrid Network Common Data Form (netCDF)-I/O API file format. The netCDF is described below. The CMAQ data files all use the netCDF convention of self-describing, selective direct access, meaning the modeling system can be more efficient by reading only the necessary parts of the data files. Additionally, netCDF files are portable across computing platforms. This means that the same file can be read, for example, on a Sun workstation, a Red Hat Linux workstation, and on Mac OSX. The I/O API component of the file format is the way that spatial information is defined in the CMAQ data files. The I/O API convention of defining horizontal grids is to use a combination of the map projection and an offset from the projection center to the southwest corner of the modeling domain. After defining the southwest corner of the domain, or the "offset" from the projection center, the I/O API grid definition specifies the size of the horizontal grid cells and the number of cells in the X and Y directions. An additional benefit of the I/O API is that an expansive set of data manipulation utilities and statistical analysis programs is available to evaluate and postprocess the binary CMAQ input/output data files.

For CMAQ users using preconfigured applications of the model, the I/O API system can be essentially transparent. For users who plan to modify the code or implement updated modules for research purposes, a few key elements of the I/O API should be understood, and they are discussed below. This section covers only the barest of necessities in terms of a CMAQ user's interaction with I/O API. For more detailed information about developing new modules for CMAQ using the I/O API code libraries, please refer to the I/O API User's Manual.

5.1.1 Files, Logical Names, and Physical Names

The I/O API stores and retrieves data using files and virtual files, which have (optionally) multiple time steps of multiple layers of multiple variables. Files are formatted internally so that they are machine- and network-independent. This behavior is unlike Fortran files, whose internal formats are platform-specific, which means that the files do not transfer using the File Transfer Protocol (FTP) or Network File System (NFS)-mount very well. Each I/O API file has an internal description, consisting of the file type, the grid and coordinate descriptions, and a set of descriptions for the file variables (i.e., names, unit specifications, and text descriptions). According to the I/O API format, files and variables are referred to by names, layers are referred to by numbers (from 1 to the greatest number of layers in the file), and dates and times are stored as integers, using the coding formats *YYYYDDD* (commonly called "JDATE") and *HHMMSS* (commonly called "JTIME"), where

```
YYYYDAY = (1000 * Year) + Julian Day
HHMMSS = (10000 * Hour) + (100 * Minute) + Seconds
```

Rather than forcing the programmer and program-user to deal with hard-coded file names or hard-coded unit numbers, the I/O API utilizes the concept of logical file names. The modelers can define the logical names as properties of a program, and then at run-time the logical names can be linked to the actual file name using environment variables. For programming purposes, the only limitations are that

file names cannot contain blank spaces and must be at most 16 characters long. When a modeler runs a program that uses the I/O API, environment variables must be used to set the values for the program's logical file names. Additional details of how the CMAQ programs use I/O API environment variables are discussed in Chapter 7. The remainder of this section explains some of the rudimentary details of programming in an environment using I/O API data files.

5.1.2 I/O API Data Structure and Data File Types

Each CMAQ data file has internal file descriptions that contain the file type, the file start date and time, the file time step, the grid and coordinate descriptions, and a set of descriptions for the set of variables contained within the file (i.e., names, units specifications, and text descriptions). Some of the elements in a file description, such as the dates and times for file creation and update and the name of the program that created the file, are maintained automatically by the I/O API. The remainder of the descriptive information must be provided at the time of file creation.

All files manipulated by the I/O API may have multiple variables and multiple layers. Each file also has a time-step structure that is shared by all of its variables. There are three kinds of time-step structure supported (Table 6□1). Within a file, all the variables are data arrays with the same dimensions, number of layers, and data structure type, although possibly different basic types (e.g., gridded and boundary variables cannot be mixed within the same file, but real and integer variables can). The data type structures that are supported are listed in Table 6□2. GRDDED3 and BNDARY3 are the most prevalent file types in a CMAQ simulation. Magic number is an indicator associated with the files type.

Table 6□1. Possible Time Step Structures in I/O API Files

File Type	Description
Time-independent	The file's time-step attribute is set to zero. Routines that use time-independent files ignore the date and time arguments.
Time-stepped	The file has a starting date, a starting time, and a positive time step. Read and write requests must be for some positive integer multiple of the time step from the starting date and time.
Circular-buffer	This type of file keeps only two "records", the "even" part and the "odd" part (useful, for example, for "restart" files where only the last data written in the file are used). The file's description has a starting date, a starting time, and a negative time step (set to the negative of the actual time step). Read and write requests must be for some positive integer multiple of the time step from the starting date and time, and they must reflect a specific time step that is in the file.

Table 6□2. Possible Data Type Structures in I/O API Files

File Type	Magic Number	Data Type	Description
CUSTOM3	-1	Custom	User-dimensioned array of REAL4s that the system reads/writes reliably

File Type	Magic Number	Data Type	Description
DCTNRY3	0	Dictionary	Data type stores and retrieves parts of an FDESC.EXT file description
GRDDED3	1	Gridded	Dimension as REAL4 ARRAY (NCOLS, NROWS, NLAYS, NVARS)
BNDARY3	2	Boundary	Dimension as REAL4 ARRAY (SIZE, NLAYS, NVARS)
IDDATA3	3	ID-reference	Used to store lists of data, such as pollution monitoring observations
PROFIL3	4	Vertical profile	Used to store lists of vertical data, such as rawinsonde observations
GRNEST3	5	Nested grid	Preliminary and experimental implementation for storing multiple grids, which need not in fact have any particular relationship with each other beyond using the same coordinate system
SMATRX3	6	Sparse matrix	Sparse matrix data, which uses a "skyline-transpose" representation for sparse matrices, such as those found in SMOKE
KFEVNT3	-3	Cloud event	KF-Cloud files use the same file description data structures (from FDESC3.EXT) and defining parameters (from PARMS3.EXT); the usual I/O API DESC3() call may be used to retrieve file descriptions from the headers. KF-Cloud files, on the other hand, have their own specialized opening/creation, look-up/indexing, input, and output operations
TSRIES3	7	Hydrology Time Series	A hydrology time series file behaves much like a degenerate gridded file, except that the numbers of rows and columns are usually 1, and that there are additional file attributes found in the INCLUDE file ATDSC3.EXT
PTRFLY3	8	Pointer-flyer	A pointer-flyer observation file behaves much like a degenerate gridded file with NCOLS3D and NROWS3D set to 1, and certain mandatory variables and variable-naming conventions to be used by analysis and visualization software

5.1.3 Opening/Creating Data Files in I/O API

The I/O API function `OPEN3` is used to open both new and existing files. `OPEN3` is a Fortran logical function that returns TRUE when it succeeds and FALSE when it fails.

```
LOGICAL FUNCTION OPEN3( FNAME, FSTATUS, PGNAME )
```

where:

```
FNAME (CHARACTER) = file name for query
FSTATUS (INTEGER) = see possible values in Table 6-3
PGNAME (CHARACTER) = name of calling program
```

`OPEN3` maintains considerable audit trail information in the file header automatically, and automates various logging activities. The arguments to `OPEN3` are the name of the file, an integer FSTATUS indicating the type of open operation, and the caller's name for logging and audit-trail purposes. `OPEN3` can be called many times for the same file. FSTATUS values are defined for CMAQ in PARMS3.EXT and are also listed in Table 6-3.

Table 6□3. Possible values for OPEN(3) FSTATUS

FSTATUS	Value	Description
FSREAD3	1	for READONLY access to an existing file
FSRDWR3	2	for READ,WRITE,UPDATE access to an existing file
FSNEW3	3	for READ,WRITE access to create a new file, file must not yet exist
FSUNKN3	4	for READ,WRITE,UPDATE access to a file whose existence is unknown
FSCREA3	5	for CREATE,TRUNCATE,READ,WRITE access to files

In the last three cases, "new" "unknown" and "create/truncate," the code developer may fill in the file description from the INCLUDE file FDESC3.EXT to define the structure for the file, and then call `OPEN3`. If the file does not exist in either of these cases, `OPEN3` will use the information to create a new file according to your specifications, and open it for read/write access. In the "unknown" case, if the file already exists, `OPEN3` will perform a consistency check between your supplied file description and the description found in the file's own header, and will return TRUE (and leave the file open) only if the two are consistent.

An example of how to use the `OPEN3` function is shown below (from the CMAQ INITSCEN subroutine). This program segment checks for the existence of a CCTM concentration (CTM_CONC_1) file, which if found will be open read-write-update. If the CCTM CONC file is not found, a warning message will be generated.

```
IF ( .NOT. OPEN3( CTM_CONC_1, FSRDWR3, PNAME ) ) THEN
MSG = 'Could not open ' // CTM_CONC_1 // ' file for update - '
```

```
& // 'try to open new'
CALL M3MESG( MSG )
END IF
```

File descriptions (i.e., I/O API file type, dimensions, start date, start time, etc.) can be obtained by using DESC3, which is an I/O API Fortran logical function. When DESC3 is called, the complete file description is placed in the standard file description data structures in FDESC3.EXT . Note that the file must have been opened prior to calling DESC3. A typical Fortran use of DESC3 is:

```
IF ( .NOT. DESC3( ' myfile' ) ) THEN
!... error message
ELSE
!... DESC3 commons now contain the file description
END IF
```

5.1.4 Reading Data Files in I/O API

There are four routines with varying kinds of selectivity used to read or otherwise retrieve data from files: READ3, XTRACT3, INTERP3, and DDTVAR3. All four are logical functions that return TRUE when they succeed, FALSE when they fail. The descriptions of the routines are listed in Table 6-4.

Table 6□4. IO API data retrieval routines

Routine	Description
READ3	reads one or all variables and layers from a file for a particular date and time.
XTRACT3	reads a windowed subgrid for one or all variables from a file for a particular date and time.
INTERP3	interpolates the requested variable from the requested file to the date/time
DDTVAR3	computes the time-derivative of the requested variable at the specified date/time

Because it optimizes the interpolation problem for the user, INTERP3 is probably the most useful of these routines. An INTERP3 call to read/interpolate the variable HNO3 to 1230 GMT on February 4, 1995, is outlined below.

```
CHARACTER*16 FNAME, VNAME
REAL*4 ARRAY( NCOLS, NROWS, NLAYS )
...
IF ( .NOT. INTERP3('myfile','HNO3',1995035,123000,NCOLS*NROWS*NLAYS,ARRAY)) THEN
... (some kind of error happened--deal with it here)
END IF
```

With READ3 and XTRACT3, you can use the "magic values" ALLVAR3 (= 'ALL', as defined in PARMS3.EXT) or ALLAYS3 (= -1, as also defined in PARMS3.EXT) as the variable name and/or

layer number to read all variables or all layers from the file, respectively. For time-independent files, the date and time arguments are ignored.

5.1.5 Writing Data Files in I/O API

CMAQ module developers should use the logical function *WRITE3* to write data to files. For gridded, boundary, and custom files, the code may write either one time step of one variable at a time, or one entire time step of data at a time (in which case, use the "magic value" ALLVAR3 as the variable name). For ID-referenced, profile, and grid-nest files, the code must write an entire time step at a time.

```
LOGICAL FUNCTION WRITE3( FNAME, VNAME, JDATE, JTIME, BUFFER)
```

where:

```
FNAME (CHARACTER) = file name for query
VNAME (CHARACTER) = variable name (or ALLVAR3 (='ALL'))
JDATE (INTEGER) = date, formatted YYYYDDD
JTIME (INTEGER) = time, formatted HHMMSS
BUFFER(*) = array holding output data
```

`WRITE3` writes data for the variable with name VNAME, for the date and time (i.e., JDATE and JTIME) to an I/O API-formatted data file with logical name FNAME. For time-independent files, JDATE and JTIME are ignored. If VNAME is the "magic name" `ALLVAR3`, `WRITE3` writes all variables. If FNAME is a dictionary file, `WRITE3` treats VNAME as a dictionary index (and ignores JDATE and JTIME). A typical `WRITE3` call to write data for a given date and time might look like this:

```
REAL*4 ARRAY( NCOLS, NROWS, NLAYS, NVARS )
!...
IF ( .NOT. WRITE3( 'myfile', 'HNO3', JDATE, JTIME, ARRAY ) ) THEN
!...(some kind of error happened--deal with it here)
END IF
IF ( .NOT. WRITE3( 'afile', 'ALL', JDATE, JTIME, ARRAYB ) ) THEN
!...(some kind of error happened--deal with it here)
END IF
```

5.1.6 CMAQ-Related I/O API Utilities

Data files in the CMAQ system can be easily manipulated by using the I/O API utilities. The I/O API utilities (also known as m3tools) are a set of scriptable programs for manipulation and analysis of netCDF-I/O API formatted files. Information regarding the most commonly employed utility routines is listed in Table 6□5. Further information about how to use the utilities are available in the I/O API documentation.

Table 6□5. I/O API data manipulation utilities

Utility	Description
M3XTRACT	extract a subset of variables from a file for a specified time interval
M3DIFF	compute statistics for pairs of variables
M3STAT	compute statistics for variables in a file
BCWNDW	build a boundary-condition file for a sub-grid window of a gridded file
M3EDHDR	edit header attributes/file descriptive parameters
M3TPROC	compute time period aggregates and write them to an output file
M3TSHIFT	copy/time shift data from a file
M3WNDW	window data from a gridded file to a sub-grid
M3FAKE	build a file according to user specifications, filled with dummy data
VERTOT	compute vertical-column totals of variables in a file
UTMTOOL	coordinate conversions and grid-related computations for lat/lon, Lambert, and UTM

5.2 Network Common Data Form (netCDF)

The Network Common Data Form (netCDF) is a set of software libraries and machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data (Unidata, 2009). The netCDF library provides an implementation of the netCDF interface for several different programming languages. The netCDF is used in CMAQ to define the format and data structure of the binary input and output files. CMAQ input and output files are self-describing netCDF-format files in which the file headers have all the dimensioning and descriptive information needed to define the resident data. Users should download the latest code for the NetCDF from the NetCDF website. Compilation and configuration information for the NetCDF is available through the Unidata website.

5.3 Message Passing Interface Library (MPI)

The Message Passing Interface (MPI) is a standard library specification for message passing, or intra-software communication, on both massively parallel computing hardware and workstation clusters. There are different open source MPI libraries available that work well with CMAQ.

- MVAPICH2 is a portable implementation of MPI that is available from Ohio State University.
- OpenMPI is an open-source MPI implelmentation that is developed and maintained by a consortium of academic, research, and industry partners.

5.4 References for Chapter 6: Required Libraries

Coats, C., 2005: The EDSS/Models-3 I/O API. Available online at the I/O API website Unidata, 2009: NetCDF. Available online at NetCDF website

6 CMAQ Programs and Libraries

6.1 Overview

The core CMAQ programs that are needed to perform a basic air quality model simulation are MCIP, ICON, BCON, and CCTM. The relationships among these programs are depicted within the green box in Figure 7-1 CMAQ Core Programs. The blue boxes represent programs that are not part of the CMAQ distribution package but supply data necessary for an air quality simulation (emissions and meteorology data). The yellow boxes represent the standard CMAQ preprocessors: MCIP, ICON, and BCON. The red box represents the CMAQ chemistry-transport model (CCTM), the Eulerian air quality modeling component of CMAQ. Data flows between the CMAQ programs are represented in by arrows. The red arrows illustrate the flow of data from the CMAQ preprocessors and the emissions model to CCTM. The green arrows show the data feedbacks from CCTM to create initial and boundary conditions for nested simulations. The black arrow illustrates the connection between the meteorological model and MCIP. Finally, the blue arrow shows that the output from MCIP can be used to drive an emissions model.

A meteorological model,such as WRF□ARW, generates gridded meteorology for input to both CMAQ and the emissions model. These data are processed for input to CMAQ using MCIP.

An emissions model converts emissions inventories to gridded, hourly emissions formatted for CMAQ. The SMOKE emissions model is currently available for preparing emissions data for CMAQ.

CMAQ includes several “in-line” options to support coupling between meteorology and chemistry processes, and to facilitate operational air quality forecast modeling (see Chapter 4). The user can incorporate photolysis rate calculations and emissions processing during a CCTM simulation. There are several advantages of incorporating these processes directly in a CCTM simulation:

1. Photolysis rate calculations use the aerosol concentrations and meteorology from the CCTM simulation, simulating the feedbacks of the input emissions and resulting air quality on photochemistry
2. Emissions are meteorologically modulated at the synchronization (chemistry) time step rather than being linearly time-interpolated within each simulation hour
3. Disk space may be saved, because a 3□D emissions file is no longer needed for elevated point sources
4. CMAQ can more easily be coupled with a meteorological model, enabling direct emissions modulation by the underlying, freshly computed meteorological variables

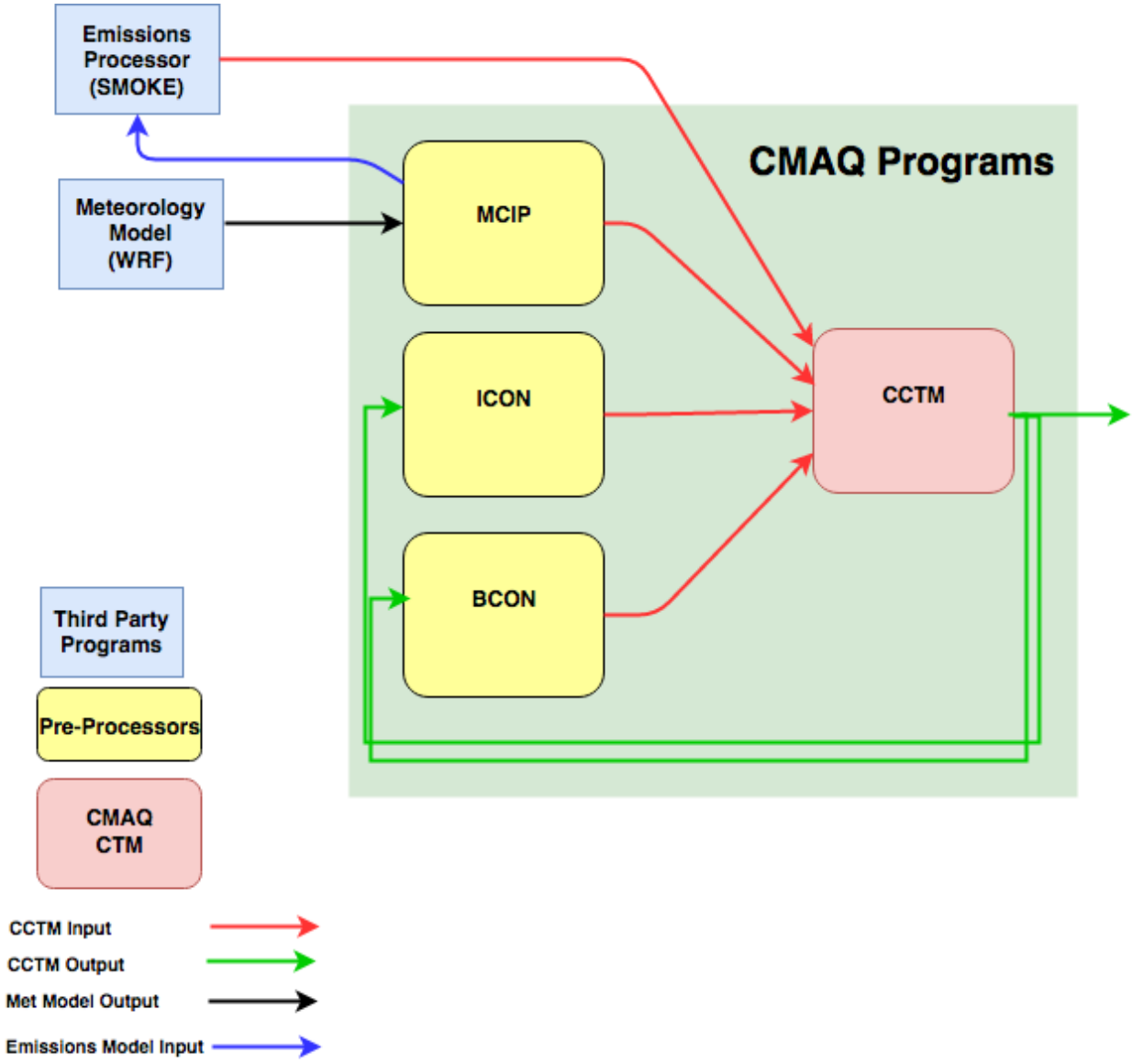


Figure 7□1.CMAQ core programs

MCIP is the first program in the CMAQ distribution package that a user should run when setting up a new simulation. MCIP is used to preprocess the data from a meteorological model for CMAQ and SMOKE.

ICON creates a binary netCDF initial conditions file for input to CCTM. Users have the option to create initial conditions data either from a text file of vertical concentration profiles or from an existing CCTM output file. ICON outputs initial conditions data that are configured for a specific modeling grid and chemical parameterization.

BCON creates a binary netCDF lateral boundary conditions file for input to CCTM. Users have the option to create boundary conditions from either a text file of vertical concentration profiles or from an existing CCTM or larger-scale (e.g., global-scale) output file. BCON outputs boundary conditions data that are configured for a specific modeling grid and chemical parameterization. If derived from an existing CCTM or larger-scale output file, BCON produces dynamic boundary conditions that vary

in time and space. When derived from vertical concentration profiles, BCON produces static boundary conditions for input to CCTM.

CCTM is run last in the sequence of programs. All of the other CMAQ programs, and the emissions and meteorological models, are used to prepare the inputs to CCTM. By using data that are synchronized for a particular modeling time period, model grid, vertical layer configuration, and chemical parameterization, CCTM can produce estimates of pollutant concentrations, wet and dry deposition rates, and visibility metrics.

In addition to the core programs shown in Figure 7□1 CMAQ Core Programs, the CMAQ distribution package also includes utilities (`$CMAQ_HOME/UTIL`) and post-processors (`$CMAQ_HOME/POST`) for utilizing additional technical and diagnostic features in CMAQ. The CMAQ utility programs CHEMMECH CSV2NML, CREATE_EBI, and INLINE_PHOT_PREPROC, and JPROC are used to modify or prepare new chemical mechanisms for CMAQ. The CMAQ preprocessor CALMAP creates maps of the crop calendar for use in estimating windblown dust emissions. The CMAQ post-processors are described in the CMAQv5.2 documentation and are used to prepare CMAQ output data for analysis.

This chapter provides detailed descriptions of the CMAQ programs and utilities. Information about the third-party libraries used by CMAQ—such as I/O API, netCDF, and MPI are available in Chapter 6. When viewing the tables that list each program's input and output files, recall that the various I/O API file formats shown are also described in Chapter 6.

6.2 BCON

6.2.1 Description

The program BCON prepares lateral chemical boundary conditions (BCs) for CCTM from either ASCII vertical profiles or from an existing CCTM output concentration (CONC) file. The BCs created by BCON can be static in both time and space (i.e., time-invariant with uniform concentrations in all boundary grid cells), dynamic in both time and space, or a combination of the two. The ASCII vertical profiles are primarily used to create static BCs. Dynamic BCs can be extracted from CONC files on either the same horizontal grid spacing (i.e., as a windowed modeling domain) or for a finer-resolution model grid (i.e., for a nested simulation), or they can be interpolated from a larger-scale CTM simulation (which is analogous to defining lateral BCs for WRF□ARW).

There are two distinct modes of operation for BCON, and the mode used depends on the nature of the input data. When creating BCON executables, the user must specify whether the input data will be ASCII vertical profiles or a CONC file by selecting either "profile" or "m3conc", respectively, for the setting of the `ModType` variable. This variable determines the input module to use when creating a BCON executable.

CCTM can also be forced with chemical boundary conditions downscaled from global chemistry models (GCMs), such as GEOS-Chem and MOZART. BCON does not support the processing of data from GCMs. BCs derived from GCMs must be calculated with custom codes or scripts that are not available in the CMAQ distribution package. The CAMx developers (Ramboll Environ) have codes available for extracting regional model BCs from both GEOS-Chem and MOZART. Visit the Support Software section of www.CAMx.com to download these utilities.

6.2.2 Files, configuration, and environment variables

Figure 7□2 shows the input and output files and configuration options for BCON. A distinction is made between the options that are invoked at compilation versus those invoked at execution of the program. When compiling BCON, the user specifies a chemical mechanism to configure the gas-phase chemistry and aerosol mechanism used to create the chemical BCs. Setting the `ModMech` and `Mechanism` variables in the BCON compile script configures the program to use a specific set of mechanism INCLUDE files to build an executable. Setting the `ModType` variable in the BCON compile script configures the program to input either a text file of static concentrations or a binary netCDF file of time-dependent concentrations for estimating BCs for CCTM. Separate BCON executables must be prepared for different mechanism and input file configurations.

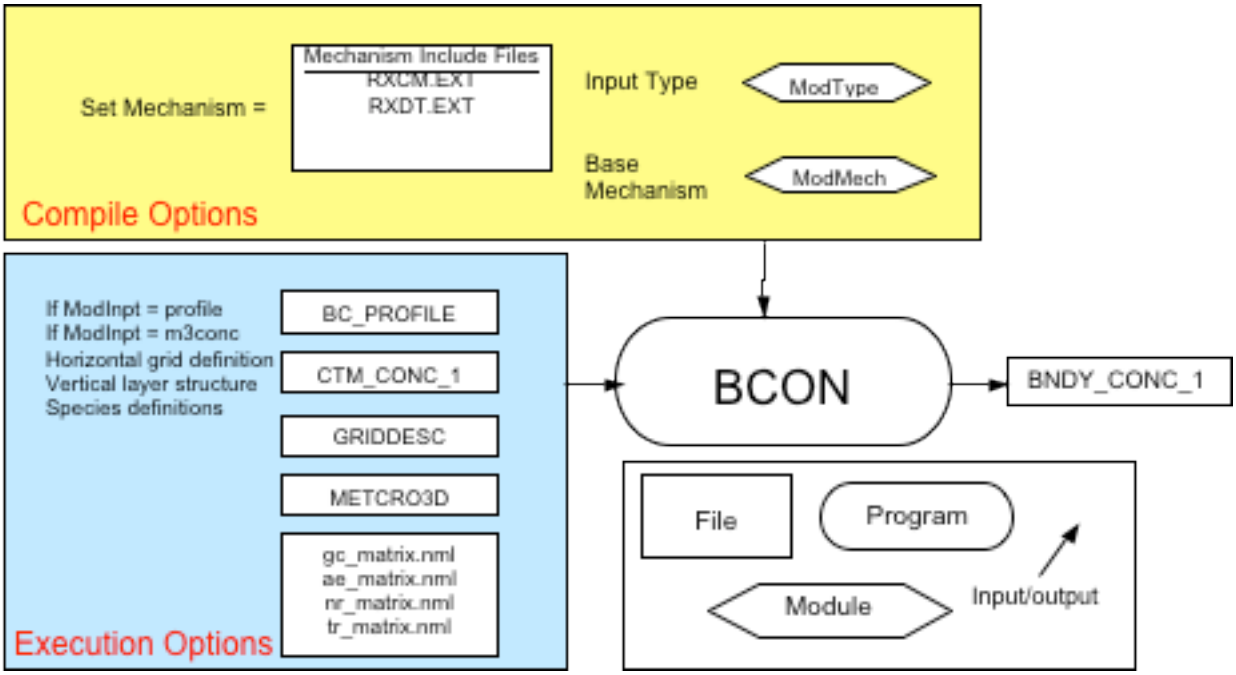


Figure 7□2. BCON input and output files

When BCON is run, it converts a data file of chemical ambient concentrations to BCs on a predefined model grid. Through the specification of the `odInpt` variable in the BCON run script, BCON will input either an ASCII vertical profile file (BC_PROFILE) or an existing CCTM concentration file (CTM_CONC_1); the choice depends on how the user compiled the model. The BC input file provided by the user must have chemical speciation that is consistent with the mechanism configuration of the BCON executable. For example, if BCON was compiled to create BCs using the CB05 mechanism, the input BC profile data must be in terms of the CB05 mechanism. CMAQ is distributed with ASCII vertical profiles representing clean continental BCs for North America for the following chemical mechanisms: cb05e51_ae6_aq, saprc07tb_ae6_aq and racm2_aq6_aq. It is the user's responsibility to generate BC inputs for other mechanism configurations.

The horizontal grid and vertical layer structures for BCON are defined at execution through the input of a grid description (GRIDDESC) file and a meteorology cross-point 3□D (MET_CRO_3D) file, respectively. BCON interpolates between the input vertical layer structure and output layer structure if they are different.

6.2.2.1 BCON input files

Table 7□1. BCON input files

File Name	Format	Description
BC_PROFILE	ASCII	Vertical chemical profiles from which to derive boundary conditions; this file is created by the user; used only when the BC environment variable is set to "profile"
CTM_CONC_1	GRDDED3	CMAQ concentration file from which to derive boundary conditions; this file is output from CCTM; used only when the BC environment variable is set to "m3conc"
MET_CRO_3D_CRS	GRDDED3	Name and location of the coarse-grid MET_CRO_3D file that is required for creating the vertical grid structure if this structure changes between nested simulations; this file is output by MCIP
MET_CRO_3D_FIN	GRDDED3	Name and location of the fine-grid MET_CRO_3D file that is required if the vertical grid structure changes between nested simulations; this file is output by MCIP
GRIDDESC	ASCII	Horizontal grid description file for defining the model grid; this file is output by MCIP or can be created by the user
LAYER_FILE	GRDDED3	3-D cross-point meteorology METCRO3D file for defining the vertical layer structure of the model grid; this file is output by MCIP
gc_matrix.nml	ASCII	Namelist file for defining the gas-phase species that are input to the model through the boundary
ae_matrix.nml	ASCII	Namelist file for defining the aerosol species that are input to the model through the boundary
nr_matrix.nml	ASCII	Namelist file for defining the non-reactive species that are input to the model through the boundary
tr_matrix.nml	ASCII	Namelist file for defining the tracer species that are input to the model through the boundary

6.2.2.2 BCON output files

Table 7□2. BCON output files

File Name	Format	Description
BNDY_CONC_1	BNDARY3	Name and location of the gridded boundary conditions data output on the model grid defined by GRID_NAME

The default location of the BCON output files is the $CMAQ_DATA/bcon directory, controlled by the OUTDIR variable in the run script. The default naming convention for all BCON output files uses the APPL and GRID_NAME environment variables in the file name. For boundary conditions created from existing CCTM CONC files, the Julian date is also used in the file name through the DATE environment variable. All of the file-naming variables for BCON outputs are set in the run script.

6.2.2.3 Compilation Configuration Variables

The configuration options listed here are set during compilation of the BCON executable. When these options are invoked they create a binary executable that is fixed to the specified configuration. To change these options you must recompile BCON and create a new executable.

- `CopySrc`
  Uncomment to copy the source code into a working build (BLD) directory. If commented, onlhy the compiled object and executable files will be placed in the BLD directory.
- `MakeFileOnly`
  Uncomment to build a Makefile to compile the executable. Comment out to both create a Makefile and compile.
- `ModType: [default: module profile]`
  Defines the format of the boundary conditions input files to be used by BCON.
  - `m3conc`: input a `CCTM CONC` file; used for nested simulations or windows of a parent domain
  - `profile`: input an ASCII vertical profiles file
  - `tracer`: use the tracer namelist file to create BCs of tagged tracer species
- `Mechanism: [default: cb05e51_ae6_aq]`
  Specifies the gas-phase, aerosol, and aqueous-phase chemical mechanisms for which to create boundary conditions. The choices for the `Mechanism` variable are the mechanism directory names under the `$CMAQ_HOME/CCTM/src/MECHS` directory. Also see the Mechanism Definitions Table). Examples include:
  - `cb6r3_ae6_aq`: CB6, revision 3 gas-phase mechanism, sixth-generation CMAQ aerosol mechanism with sea salt and speciated PM Other, `aqueous/cloud` chemistry
  - `cb05e51_ae6_aq`: CB05 gas-phase mechanism with CMAQv5.1 updates, sixth-generation CMAQ aerosol mechanism with sea salt and speciated PM Other, `aqueous/cloud` chemistry
  - `cb05tucl_ae6_aq`: CB05 gas-phase mechanism with active chlorine chemistry, updated toluene mechanism, sixth-generation CMAQ aerosol mechanism with sea salt and speciated PM Other, `aqueous/cloud` chemistry
  - `cb05tump_ae6_aq`: CB05 gas-phase mechanism with active chlorine chemistry, updated toluene mechanism, mercury, and air toxics, sixth-generation CMAQ aerosol mechanism with sea salt and speciated PM, `aqueous/cloud` chemistry; this is the CMAQv5 multipollutant mechanism
  - `saprc07tb_ae6_aq`: SAPRC-07 gas-phase mechanism with toluene updates and sixth-generation CMAQ aerosol mechanism
  - `racm2_ae6_aq`: RACM2 gas-phase mechanism with toluene updates and sixth-generation CMAQ aerosol mechanism
- `Tracer [default trac0]`
  Specifies tracer species. Invoking inert tracer species in CMAQ requires defining the tracers using namelist files and compiling the CMAQ programs with these files. The setting for this module corresponds to the directory name in the $CMAQ_HOME/CCTM/src/MECHS directory that contains the namelist files for the tracer configuration. The default setting is to not use any tracers.

  - `trac[n]`

6.2.2.4 Execution Configuration Variables

The environment variables listed here are invoked during execution of the program and are set in the BCON run script.

- `APPL [default: None]` BCON executable identifier. Must match APPL Variable setting in the BCON build script.
- `CFG [default: None]` Configuration identifier for the BCON simulation.
- `MECH [default: None]` CMAQ chemical mechanism. Must match Mechanism variable setting in the BCON build script.
- `EXEC: [default: BCON_${APPL}_${EXECID}]` Executable to use for the simulation. The variable CFG is set in the BCON run script. The variable EXECID is set in the config_cmaq.csh configuration file.
- `GRIDDESC: [default: $CMAQ_HOME/scripts/GRIDDESC1]` Grid description file for setting the horizontal grid definition.
- `GRID_NAME: [default:CMAQ-BENCHMARK]` Name of the grid definition contained in the GRIDDESC file that specifies the horizontal grid for the current application of the model.
- `IOAPI_ISPH: [default: 20]` I/O API setting for spheroid type. See I/O API documentation for setsphere for more information.
- `IOAPI_OFFSET_64: [default: NO]` I/O API setting for large time-step records. If your output time step is going to produce data that are greater than 2GB per time step, then this needs to be set to YES.
- `LAYER_FILE:[default: none]` Name and location of a MET_CRO_3D file for specifying the vertical layer structure for the current application of the model.
- `gc_matrix.nml: [default: none]` Gas-phase species namelist file. This file is used to configure the gas-phase species that will be output by BCON.
- `ae_matrix.nml: [default: none]` Aerosol-phase species namelist file. This file is used to configure the aerosol-phase species that will be output by BCON
- `nr_matrix.nml: [default: none]` Nonreactive species namelist file. This file is used to configure the nonreactive species that will be output by BCON
- `tr_matrix.nml: [default: none]` Tracer species namelist file. This file is used to configure the tracer species that will be output by BCON
- `OUTDIR: [default: $CMAQ_HOME/data/bcon]` Output data directory.
- `BC:` Sets the input file type. The setting of this variable determines how the run script sets the input and output environment variables.
  - `profile`: sets the output file name to include the tag "profile" in the name; uses the variable BC_PROFILE to point to an ASCII vertical profile file for input to BCON.
  - `m3conc`: used for nested simulations; sets the output file name to include a start date in the name; uses the variable `CTM_CONC_1` to point to a CCTM CONC file for input to BCON.
- `DATE:` Sets the Julian date to use in naming the BCON output file for nested runs.

- SDATE:[default: ${DATE}] Julian start date for extracting boundary conditions from a CCTM CONC file for a nested simulation. If SDATE is not set, it will be set automatically from the CTM_CONC_1 file.
- STIME: [default: 000000] Start time for extracting boundary conditions from a CCTM CONC file for a nested simulation. If STIME is not set, it will be set automatically from the CTM_CONC_1 file.
- RUNLEN: [default: 240000] Number of hours of boundary conditions to extract from a CCTM CONC file for a nested simulation. If RUNLEN is not set, it will be set automatically from the CTM_CONC_1 file.

6.2.3 Compiling and Running

6.2.3.1 Compile BCON

Chapter 5 provides an overview of how to install and compile the CMAQ pre-processor programs for a test simulation. Follow those steps (summarized below) to compile new versions of BCON.

1. Compile Bldmake, the CMAQ source code and compilation management program. This needs to be done only once—the first time CMAQ is installed.

- Configure the BCON use the config_cmaq.csh script, which points to the available I/O API and netCDF libraries.
- Configure the BCON build script for your application by setting the compilation configuration variables described above.
- Invoke the build script to create an executable.

```
cd $CMAQ_HOME/PREP/bcon/scripts/
./bldit.bcon |& tee build.bcon.log
```

6.2.3.2 Run BCON

Set the run script settings according to the execution configuration variables described above. Run BCON to produce boundary conditions for the CCTM:

```
cd $CMAQ_HOME/PREP/BCON
./run.bcon |& tee bcon.log
```

6.3 Calmap

6.3.1 Description

The preprocessor program Calmap produces gridded planting start dates, planting end dates, and harvesting end dates for different crop types for estimating the impacts of agricultural activities on windblown dust emissions. The CMAQ windblown dust emissions module can optionally use the output from Calmap to simulate the effects of crop cycles on dust emissions.

6.3.2 Files, configuration, and environment variables

Figure 7-3 shows that Calmap reads five input files to produce eight outputs, only three of which are used by the CCTM. Calmap uses the GRIDCRO2D file, produced by MCIP, to define the modeling grid. The BELD01 file points to a BELD3 "a" file of gridded land cover/land use data containing coverage for several different crop categories. The BELD3 "a" file is an input to the BEIS emissions model and can be generated by the Spatial Allocator. The rest of the inputs to Calmap are crop calendar data for the United States that are packaged with CMAQv5. Calmap converts to the data to I/O API GRDDED3 files on the modeling grid defined in the GRIDCRO2D file. The CROPMAP01 file contains planting start dates for specific crops. The CROPMAP04 file contains planting end dates for specific crops. The CROPMAP08 file contains harvesting end dates for specific crops. Each of these files are input to the CCTM when the erodible agricultural land (CTM_ERODE_AGLAND) feature is turned on.

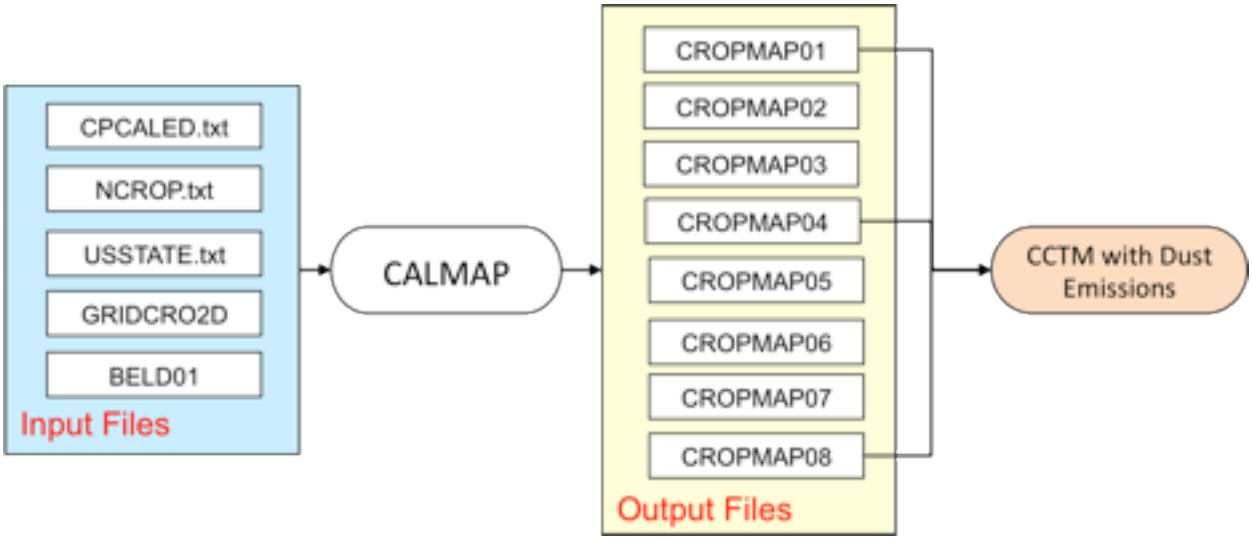


Figure 7-3. Calmap input and output files

6.3.2.1 Calmap input files

Table 7-3. Calmap input files

File Name	Format	Description
GRID_CRO_2D	GRDDED3	Name and location of the time-independent 2-D cross-point meteorology file; output by MCIP
BELD01	GRDDED3	BELD land use "A" data file for calculating windblown dust emissions; produced with BELD land use tiles and the Spatial Allocator
CPCALED	ASCII	Calendar of agricultural activities by state
NCROP	ASCII	Number and names of crop species included in the crop calendar
USSTATE	ASCII	Two digit US state codes included in the crop calendar

6.3.2.2 Calmap output files

Table 7□4. Calmap output files

File Name	Format	Description
CROPMAP01	GRDDED3	Name and location of the gridded planting start dates file.
CROPMAP02	GRDDED3	Name and location of the gridded ??? start dates file; not used by the CCTM.
CROPMAP03	GRDDED3	Name and location of the gridded ??? dates file; not used by the CCTM
CROPMAP04	GRDDED3	Name and location of the gridded planting end dates file.
CROPMAP05	GRDDED3	Name and location of the gridded ??? start dates file; not used by the CCTM
CROPMAP06	GRDDED3	Name and location of the gridded ??? start dates file; not used by the CCTM
CROPMAP07	GRDDED3	Name and location of the gridded harvesting end dates file.

6.3.2.3 Execution Configuration Variables

The environment variables listed here are invoked during execution of the program and are set in the Calmap run script.

- `BASE: [default: $CMAQ_HOME/PREP/agdust/scripts] Base Calmap installation location.`
- `GRID_CRO_2D: [default: none] Directory path and name of the GRID_CRO_2D file for defining t`
- `BELD01: [default: none] Directory path and name of the BELD3 A file for defining the gridde`
- `CROPMAP01-08: [default: none] Directory path and names of Calmap output files. The CROPMAP0`

6.3.3 Compiling and Running

6.3.3.1 Compile Calmap

Calmap is compiled with a Makefile. The configuration options in the Makefile include only the compiler and compiler flags to use for building the executable. The Makefile is located in the directory with the Calmap source code `$CMAQ_HOME/PREP/agdust/src`. To compile Calmap, source the config_cmaq.csh file and invoke the Makefile at the command line.

```
cd $CMAQ_HOME/PREP/agdust/src
source $CMAQ_HOME/config_cmaq.csh
./make |& tee make.calmap.log
```

To port Calmap to different compilers, change the compiler names, locations, and flags in the config.cmaq script.

6.3.3.2 Run Calmap

Set the run script settings according to the execution configuration variables described above. Run Calmap to produce crop calendar input for the CCTM inline windblown dust model:

```
cd $CMAQ_HOME/PREP/agdust/scripts
./run_calmap.csh |& tee run_calmap.log
```

6.4 CCTM

6.4.1 Description

CCTM is the Eulerian chemistry and transport component of CMAQ. It uses input data produced by the other CMAQ programs and from meteorological and emissions models. CCTM produces multiple output files for each simulation. The basic CCTM outputs include instantaneous and average hourly concentration files, wet and dry deposition files, and visibility estimates. Other CCTM outputs can include diagnostic aerosol and cloud files and processes analysis files.

CCTM contains several science configurations for simulating transport, chemistry, and deposition. All of the science configuration options in CCTM, such as the chemical mechanism to be used, are set when compiling the executable. The model grid and vertical layer structure for CCTM are set at execution. The important distinction between selecting the science configuration and the model grid, layer configuration is that CCTM does not need to be recompiled when changing model grids, layers but does need to be recompiled when new science options are invoked.

Optional output files are created when their associated processes are invoked in CCTM. For example, when CCTM is compiled with process analysis turned on, additional output files are created.

CCTM includes options for the in-line processing of emissions and photolysis rates. In-line refers to the handling of processes that had previously been accomplished outside of CCTM, such as emissions processing with SMOKE, with algorithms internal to CCTM. The benefits of in-line emissions processing include the integration of higher time-resolution meteorology in the computation of biogenic emissions and plume rise from point sources and the avoidance of the large data storage burden required for emissions data. The benefit of in-line photolysis rate calculations is the inclusion of predicted gas and aerosol concentrations in the rate calculations.

Both in-line emissions and photolysis are invoked through compile-time configuration options for CCTM. When CCTM is instrumented for in-line emissions calculations, a series of additional input files and environment variables are required at execution. The details of these additional inputs are provided below. In-line photolysis does not require any additional inputs as CCTM includes all of the photolysis rate data internal to the in-line instrumented version of the model.

6.4.2 Files, configuration, and environment variables

Figure 7□4 shows the input and output files and configuration options for CCTM. A distinction is made between the options that are invoked at compilation time versus those invoked at execution of the program. When compiling CCTM, the user specifies a chemical mechanism to configure the gas-phase chemistry and aerosol mechanism used for the air quality calculations. Setting the `Mechanism` variable in CCTM compile script configures the program to use a specific set of mechanism INCLUDE files to build an executable. All of the science processes simulated by CCTM must also be selected during the compilation step for CCTM. Separate CCTM executables must be prepared for different mechanism and science configurations. During the execution step, or when CCTM is run, the user sets the horizontal and vertical grid definitions and the input files used for the simulation. Different spatial domains, vertical grid structures and input files can be used with a single CCTM executable, as long as the input files are consistent with the scientific configuration built into the executable. For example, with the gas-phase photochemical mechanism configuration built into a CCTM executable, different modeling domains can be simulated with the executable as long as the emissions and `IC/BC` files are consistent with the photochemical mechanism configuration built into the executable.

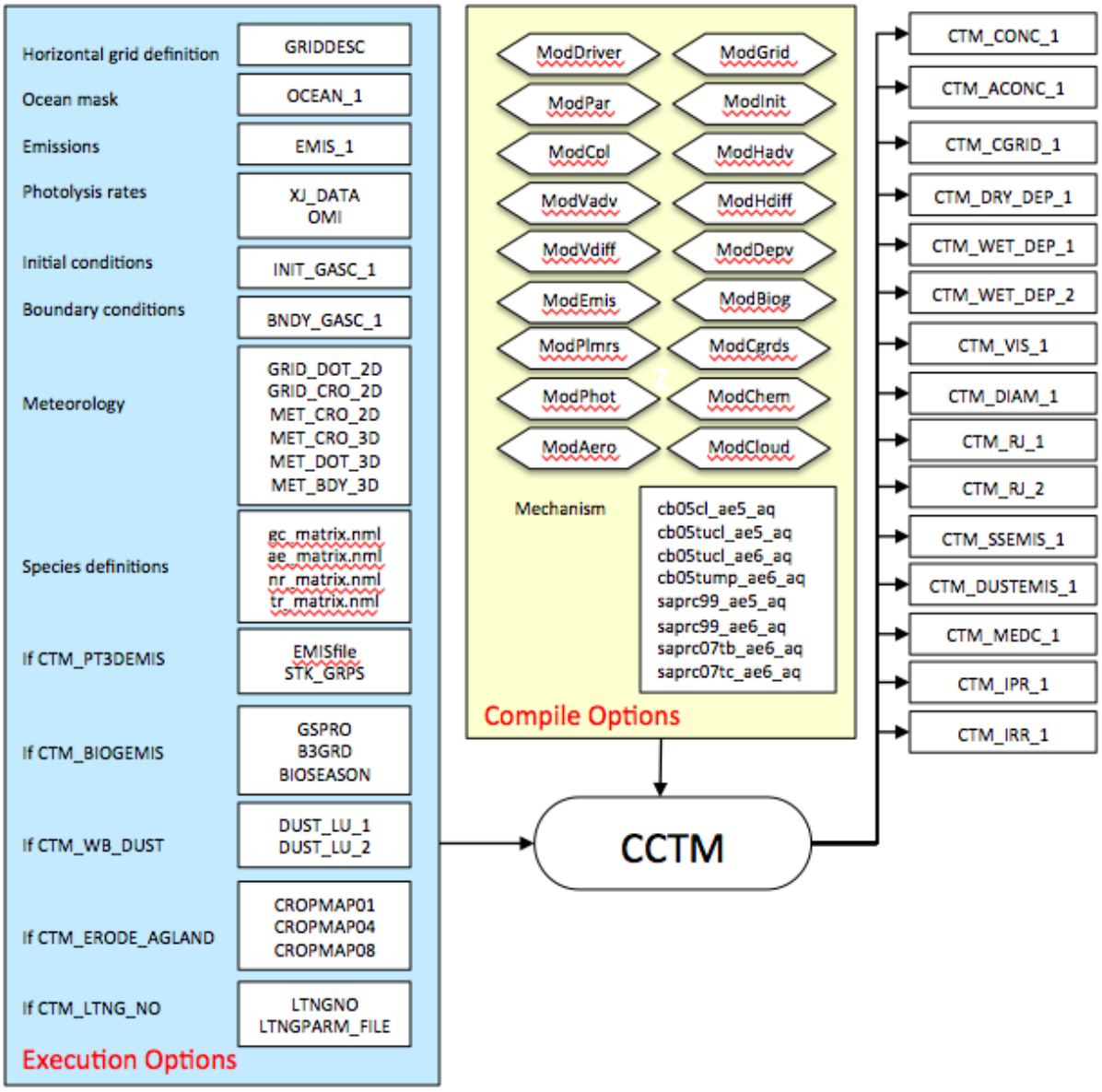


Figure 7□4. CCTM input and output files

6.4.2.1 CCTM input files

Table7□5. Required CCTM input files

File Name	Format	Description
GRIDDESC	ASCII	Map projection and grid definitions
OCEAN_1	GRDDED3	Name and location of the time-independent 2-D file for defining the fraction of each model grid cell covered by open ocean

File Name	Format	Description
EMIS_1	GRDDED3	Name and location of the time-dependent 2-D or 3-D emission file speciated for a particular gas-phase chemical mechanism and PM model; output from an emission model, such as SMOKE or CONCEPT
INIT_[GASC/AERO/NONR/TRAC]	GRDDED3	Name and location of the time-dependent, single-time-step, 3-D initial conditions file speciated for a particular gas-phase chemical mechanism and PM model; output from ICON
BNDY_[GASC/AERO/NONR/TRAC]	BNDARY3	Name and location of the time-dependent, either single-time-step or multi-time-step, 3-D boundary conditions file speciated for a particular gas-phase chemical mechanism and PM model; output from BCON
GRID_CRO_2D	GRDDED3	Name and location of the time-independent 2-D cross-point meteorology file; output by MCIP
GRID_DOT_2D	GRDDED3	Name and location of the time-independent 2-D dot-point meteorology file; output by MCIP
MET_CRO_2D	GRDDED3	Name and location of the time-dependent 2-D cross-point meteorology file; output by MCIP
MET_DOT_3D	GRDDED3	Name and location of the time-dependent 3-D dot-point meteorology file; output by MCIP
MET_CRO_3D	GRDDED3	Name and location of the time-dependent 3-D cross-point meteorology file; output by MCIP
MET_BDY_3D	BNDARY3	Name and location of the time-dependent 3-D boundary meteorology file; output by MCIP
CSQY_DATA	ASCII	Absorption cross section and quantum yield data for calculating inline photolysis rates
OMI	ASCII	Ozone Mapping Instrument vertical ozone column data by latitude for different years
OPTICS_DATA	ASCII	Wavelength, Optical and Surface Albedo Parameters for CMAQ In-Line Photolysis calculation
gc_matrix.nml	ASCII	Namelist file for defining the gas-phase species that are input to the model through the boundary
ae_matrix.nml	ASCII	Namelist file for defining the aerosol species that are input to the model through the boundary
nr_matrix.nml	ASCII	Namelist file for defining the non-reactive species that are input to the model through the boundary
tr_matrix.nml	ASCII	Namelist file for defining the tracer species that are input to the model through the boundary

6.4.2.2 CCTM optional input files

Table 7□6. Optional CCTM input files

File Name	Format	Description
XJ_DATA	ASCII	Name and location of the daily clear-sky photolysis rates file speciated for a particular gas-phase chemical mechanism; output from JPROC - only needed for offline photolysis configuration
STK_GRPS_nn	GRDDED3	Stack parameter file for calculating inline plume rise for point source emissions - nn refers to the sector ID number, where there could be multiple point source stack groups used in a single simulation; produced by the SMOKE program Elevpoint
STK_EMIS_nn	GRDDED3	Emissions for elevated point sources - nn refers to the sector ID number, where there could be multiple point source sectors used in a single simulation; produced by the SMOKE program Smkmerge
GSPRO	ASCII	Speciation profile file. Contains the factors that are used to separate aggregated inventory pollutant emissions totals into emissions of model species required by CCTM
B3GRD	GRDDED3	Normalized biogenic emissions for input to BEIS3; produced by the SMOKE program Normbeis3.
BIOSEASON	GRDDED3	Indicates the biogenic emissions factors to use for each day in a given year. This file can be created using the SMOKE utility program Metscan. The BIOSEASON file is time-dependent and usually contains data for an entire year (365 or 366 days). It uses one variable, SEASON, which is either 0 (grid cell should use winter factors for current day) or 1 (grid cell should use summer factors for current day.
SOILINP	GRDDED3	Biogenic NO soil input restart file; output by the previous day CCTM simulation.
DUST_LU_1	GRDDED3	BELD land use “A” data file for calculating windblown dust emissions; produced with BELD land use tiles and the Spatial Allocator
DUST_LU_2	GRDDED3	BELD land use “TOT” data file for calculating windblown dust emissions; produced with BELD land use tiles and the Spatial Allocator
MODIS_FPAR	GRDDED3	MODIS derived time-varying vegetation land cover data. Can be generated with the Spatial Allocator Raster Tools.

File Name	Format	Description
CROPMAP01	GRDDED3	Gridded planting start dates for estimating dust emissions from erodible cropland; produced by the CMAQ preprocessor Calmap
CROPMAP04	GRDDED3	Gridded planting end dates for estimating dust emissions from erodible cropland; produced by the CMAQ preprocessor Calmap
CROPMAP08	GRDDED3	Gridded harvesting end dates for estimating dust emissions from erodible cropland; produced by the CMAQ preprocessor Calmap
LTNGNO	GRDDED3	Lightning NO emissions file with rate of production (moles/sec) for each model layer at each time step
NLDN_STRIKES	GRDDED3	Hourly observed lightning strikes (km-2) gridded to the CCTM domain.
LTNGPARMS_FILE	GRDDED3	Time-independent, gridded lightning parameters file that includes the regression parameters derived from historical NLDN observations and WRF predicted convective precipitations using Kain-Fritsch convective scheme, ocean masks, and the ratio of intercloud to cloud-to-ground flashes
BELD4_LU	GRDDED3	BELD4 land use file with fractional crop distributions gridded to the modeling domain
E2C_SOIL	GRDDED3	EPIC soil properties file for each crop type gridded to the modeling domain. This time-independent file contains the soil pH for each agricultural crop being modeled for the 1□cm surface layer and 10-cm tilled layer
E2C_FERT	GRDDED3	EPIC crop type file gridded to the modeling domain. This file contains the initial soil ammonium concentrations for the first day of the simulation estimated by EPIC and the fertilizer application depth and rate

6.4.2.3 CCTM output files

Table 7-7 lists the logical file names, formats, and descriptions of the output files that are produced by the base configuration of CCTM. Activating different science modules, in-line deposition, and in-line emissions processing produces additional output files from CCTM. Table 7-8 lists the logical file names, formats, and descriptions of the output files that are produced by optional configurations of CCTM.

Table 7□7. CCTM base output files

File Name	Format	Description
CTM_CONC_1	GRDDED3	Hourly 3-D instantaneous gas- and aerosol-phase pollutant estimates

File Name	Format	Description
S_CGRID	GRDDED3	Simulation-ending 3-D full CGRID (gas- and aerosol-phase pollutants) concentrations for use as a restart file
A_CONC_1	GRDDED3	Hourly 2-D or 3-D integral average gas- and aerosol-phase pollutant estimates
CTM_DRY_DEP_1	GRDDED3	Hourly 3-D gas- and aerosol-phase dry deposition estimates
CTM_WET_DEP_1	GRDDED3	Hourly 3-D gas- and aerosol-phase wet deposition estimates
CTM_VIS_1	GRDDED3	Instantaneous top of the hour, hourly 3-D visibility metrics

Table 7□8. CCTM optional output files

File Name	Format	Description
CTM_SSEMIS_1	GRDDED3	Hourly 2-D sea salt emissions; set the variable `CTM_SSEMDIAG` to `Y` within the CCTM to run script to write this file
CTM_WET_DEP_2	GRDDED3	Name and location of hourly 2-D cloud diagnostics file; set the variable `CLD_DIAG` to `Y` in the CCTM run script to write this file
CTM_DEPV_DIAG	GRDDED3	Hourly 2-D in-line deposition diagnostics file; output when in-line deposition is activated by setting `CTM_ILDEPV` to `Y` and the variable `CTM_DEPV_FILE` is set to `T` or `Y` in the CCTM run script
CTM_IPR_[1-3]	GRDDED3	Hourly 2-D or 3-D integrated process rate (IPR) files; multiple files written when CCTM is configured to run with IPR
CTM_IRR_[1-3]	GRDDED3	Hourly 2-D or 3-D integrated reaction rate (IRR) files; multiple files written when CCTM is configured to run with IRR
CTM_RJ_[1-2]	GRDDED3	Hourly photolysis diagnostic output file; multiple files written when there are a large number of photolytic reactions in a chemical mechanism; set the variable `CTM_PHOTDIAG` to `Y` in the CCTM run script to write this file
B3GTS_S	GRDDED3	Hourly biogenic emissions file; output when in-line biogenic emissions processing is activated by setting `CTM_BIOGEMIS` to `Y` and the variable B3GTS_DIAG is set to “Y” in the CCTM run script
SOILOUT	GRDDED3	Hourly soil NO emissions file; output when in-line biogenic emissions processing is activated by setting `CTM_BIOGEMIS` to `Y`
CTM_DUST_EMIS_1	GRDDED3	Hourly 2-D dust emissions file; output when the CCTM dust module is activated by setting `CTM_WB_DUST` to `Y` and the variable `CTM_DUSTEM_DIAG` is set to `Y` in the CCTM run script

File Name	Format	Description
CTM_LTNG_DIAG_1	GRDDED3	Hourly average 3-D lighting NO emissions file; output when the CCTM lightning module is activated by setting `CTM_LTNG_NO` to `Y` and the variable LTNGDIAG is set to "Y" in the CCTM run script
CTM_LTNG_DIAG_2	GRDDED3	Hourly column toal NO lighting emissions file; output when the CCTM lightning module is activated by setting `CTM_LTNG_NO` to `Y` and the variable LTNGDIAG is set to "Y" in the CCTM run script
CTM_PT3D_DIAG	GRDDED3	Hourly 3-D point-source emissions file; output when in-line emissions processing is activated by setting `CTM_PT3DEMIS` to `Y` and the variable PT3DDIAG is set to "Y" in the CCTM run script
PLAY_SRCID_NAME	GRDDED3	Hourly 3-D layer fractions file; output when in-line emissions processing is activated by setting `CTM_PT3DEMIS` to `Y` and the variable `PT3DFRAC` is set to `Y` in the CCTM run script
CTM_DRY_DEP_MOS	GRDDED3	
CTM_DRY_DEP_FST	GRDDED3	
MEDIA_CONC	GRDDED3	Hourly mercury deposition output file; output when bidirectional mercury flux is activated by setting the variable `CTM_HGBIDI` to `Y` in the CCTM run script
CTM_VDIFF_DIAG	GRDDED3	Hourly vertical diffusion diagnostic data; activate by setting the variable `VDFF_DIAG_FILE` to `Y` in the CCTM run script
CTM_VSED_DIAG	GRDDED3	Hourly gravitational settling diagnostic data; activate by setting the variable `VDFF_DIAG_FILE` to `Y` in the CCTM run script
CTM_AOD_1	GRDDED3	Hourly aerosol optical depth estimates; activate by setting the variable `CTM_AOD` to `Y` in the CCTM run script
CTM_AVIS_1	GRDDED3	Hourly-averaged 3-D visibility metrics; set the variable `CTM_AVISDIAG` to `Y` in the CCTM run script to write this file
CTM_PMDIAG_1	GRDDED3	Hourly instantaneous 3-D aerosol diagnostics for conditions at the top of the hour; dp and sigmas for Aitken and accumulation mode aerosol species; set the variable `CTM_PMDIAG` to `Y` in the CCTM run script to write this file
CTM_APMDIAG_1	GRDDED3	Hourly average 3-D aerosol diagnostics; dp and sigmas for Aitken and accumulation mode aerosol species; set the variable `CTM_APMDIAG` to `Y` in the CCTM run script to write this file

The default location of the CCTM output files is the `$CMAQ_HOME/data/cctm` directory, controlled by the `OUTDIR` variable in the run script. The default naming convention for all CCTM output files uses the `EXEC` and `APPL` environment variables in the file name. All of the variables for naming the CCTM outputs are set in the run script.

6.4.2.4 Compilation Configuration Variables

The configuration options listed here are set during compilation of the CCTM executable. When these options are invoked they create a binary executable that is fixed to the specified configuration. To change these options you must recompile CCTM and create a new executable.

Several of the CCTM science modules have more than one option. Brief descriptions of these options are provided here. For details on the science of the different options refer to the CMAQ Release Notes.

The following five options are invoked by uncommenting the line in the CCTM build script. Comment the line in the script using a "#" to turn the option off.

- `CopySrc`
  Uncomment to copy the source code into a working build (BLD) directory. If commented, onlhy the compiled object and executable files will be placed in the BLD directory.
- `set ParOpt`
  Build an executable for running on multiple processors. Invoking this command requires the availability of the MPI library/INCLUDE files.
- `MakeFileOnly`
  Uncomment to build a Makefile to compile the executable. Comment out to both create a Makefile and compile.
- `set build_parallel_io`
  Uncomment to build CMAQ with true parallel I/O feature (requires ioapi3.2 and pnetcdf)
- `set build_twoway`
  Uncomment to build WRF-CMAQ twoway - this cannot be set for stand-alone CMAQ
- `set potvortO3`
  Uncomment to build CMAQ with potential vorticity free-troposphere O3 scaling

The following configuration settings may have multiple options. Select one option in the CCTM build script.

- `ModDriver: [default: driver/wrf]`
  The CCTM generalized -coordinate driver module.
  - `driver/wrf`
    use WRF-based scheme for mass-conserving advection; select this option when using WRF meteorology

  - `driver/yamo`
    use Yamartino scheme for mass-conserving advection

- `ModGrid: [default: Cartesian]`
  The CCTM model grid configuration module. Currently only Cartesian coordinates are supported by CMAQ. Do not change this module setting.

  - `grid/cartesian`

- `ModInit: [default: init/yamo]`
  The CCTM time-step initialization module that uses a Yamartino scheme for mass-conserving advection. Do not change this module setting.

  - `init/yamo`

- `ModCpl: [default: couple/gencoor_wrf]`
  Mass coupling concentration converstion module options. Unit conversion and concentration coupling module.

  - `couple/gencoor_wrf`
    Coupling scheme compatible with the WRF-based advection scheme; select this option when `ModDriver` is set to `driver/wrf`
  - `couple/gencoor`
    Coupling scheme compatible with the Yamartino advection scheme; select this option when `ModDriver` is set to `driver/yamo`.

- `ModHadv: [default: hadv/yamo]`
  Horizontal advection module. Currently only the Yamartino global mass-conserving hoizontal advection scheme is supported.

  - `hadv/yamo`

- `ModVadv: [default: vadv/wrf]`
  Vertical advection module.

  - `vadv/wrf`
    use the WRF omega calculation with the Piecewise Parabolic Method (PPM) to calculate vertical advection; this module should be used only with WRF meteorology
  - `vadv/yamo`
    use the global mass-conserving scheme to calculate vertical advection

- `ModHdiff: [default: hdiff/multiscale]`
  The only option in CMAQv5 for the horizontal diffusion module is `hdiff/multiscale`, which uses a diffusion coefficient based on local wind deformation. Do not change this module setting.

  - `hdiff/multiscale`

- `ModVdiff: [default: vdiff/acm2]`
  Vertical diffusion and surface exchange module. Do not change this module setting.

  - `vdiff/acm2`
    calculate vertical diffusion using the Asymmetric Convective Model version 2 (ACM2)

- `ModDepv: [default: depv/m3dry]`
  Deposition velocity calculation module. Do not change this module setting.

  - `depv/m3dry`
    CMAQ dry deposition velocity routine

- `ModEmis: [default: emis/emis]`
  CMAQ in-line anthropogenic and natural emissions module. In line emissions are activated in the CCTM run script. Do not change this module setting.

  - `emis/emis`

- `ModBiog: [default: biog/beis3]`
  Calculate biogenic emissions in-line with the BEIS3 model. Inline biogenic emissions are activated in the CCTM run script. Do not change this module setting.

  - `biog/beis3`

- `ModPlmrs: [default: plrise/smoke]`
  Calculate in-line plume rise for large point sources using the Briggs algorithm as it is implemented in SMOKE. Inline emissions plume rise in controlled in the CCTM run script. Do not change this module setting.

  - `plrise/smoke`

- `ModCgrds: [default: spcs/cgrid_spcs_nml]`
  CMAQ model species configuration module.

  - `spcs/cgrid_spcs_nml`
    namelist files used to configure CMAQ model species
  - `spcs/cgrid_specs_icl`
    use Fortran INCLUDE files used to configure CMAQ model species

- `ModPhot: [default: phot/inline]`
  Photolysis calculation module.

  - `phot/inline`
    calculate photolysis rates in-line using simulated aerosols and ozone concentrations
  - `phot/table`
    calculate clear-sky photolysis rates off-line using the CMAQ program JPROC; provide daily photolysis rate look-up tables to CCTM

- `Mechanism: [default: cb05e51_ae6_aq]`
  Chemistry mechanism for gas, aerosol, and aqueous chemistry. See the [CMAQ Mechanism Definitions Table(https://github.com/USEPA/CMAQ/blob/5.2/DOCS/User_Manual/CMAQ_OGD_appendix_A.m
  for a listing of the mechanism choices that are available in CMAQv5.2.

- `Tracer [default trac0]`
  Specifies tracer species. Invoking inert tracer species in CMAQ requires defining the tracers using namelist files and compiling the CMAQ programs with these files. The setting for this module corresponds to the directory name in the `$CMAQ_HOME/CCTM/src/MECHS` directory that contains the namelist files for the tracer configuration. The default setting is to not use any tracers.
  - `trac[n]`
- `ModGas: [default: gas/ebi_${Mechanism}]`
  Gas-phase chemistry solver module.
  - `smvgear`
    use the SMVGEAR chemistry solver
  - `ros3`
    use gas/the Rosenbrock chemistry solver
  - `ebi`
    use the Euler Backward Iterative solver
- `ModAero: [default: aero6]`
  CMAQ aero/aerosol module.
  - `aero6`
    sixth-generation modal CMAQ aerosol model with extensions for sea salt emissions and thermodynamics; includes a new formulation for secondary organic aerosol yields
- `ModCloud: [default: cloud/acm_ae6]`
  CMAQ cloud module for modeling the impacts of clouds on deposition, mixing, photolysis, and aqueous chemistry.
  - `cloud/acm_ae6`
    ACM cloud processor that uses the ACM methodology to compute convective mixing with heterogeneous chemistry for AERO6
  - `cloud/acm_ae6_mp`
    ACM cloud processor that uses the ACM methodology to compute convective mixing with heterogeneous chemistry for AERO6 and air toxics; this is the multipollutant mechanism in CMAQv5
  - `cloud/acm_ae6_kmt`
    ACM cloud processor that uses the ACM methodology to compute convective mixing with heterogeneous chemistry for AERO6 and aqueous chemistry with kinetic mass transfer and Rosenbrock solver
  - `cloud/acm_ae6i_kmti`
    ACM cloud processor that uses the ACM methodology to compute convective mixing with heterogeneous chemistry for AERO6 and aqueous chemistry with kinetic mass transfer and Rosenbrock solver with an extension to simulate the aqueous phase formation of SOA in cloud droplets, see: CMAQv5.1 Aqueous Chemistry
- `ModUtil: [default: util]`
  CMAQ utility modules. Do not change this module setting.

  - util/util

- Tracer: [default: trac0] Add chemically inert tracers to the CCTM, default no tracer species.
- ModPa: [default: procan/pa] Process analsis is controlled in the CCTM run script. Do not change this module setting.

  - procan/pa

- ModPvO3: [default: pv_o3] Potential vorticity parameterization for free-troposphere exhange of ozone. This option is configured using the potvorO3 variable in the CCTM build script. Do not change this module setting.

  - pv_o3

6.4.2.5 Execution Configuration Variables

The environment variables listed below are invoked during execution of the CCTM and are set in the CCTM run script.

- compiler [default: intel]
- compilerVrsn [default: 13.1]
- VRSN [default: v52]
- PROC [default: mpi]
  Sets if the CCTM will run in multi-processor or serial mode.

  - mpi
    Use MPI multi-processor configuration. Additional configuration settings are required when selecting mpi. The CCTM must have been built to support MPI. The run script requires settings for the number of processors and other MPI configuration variables required by the Linux system.
  - serial
    Run the CCTM in serial, single-processor mode.

- MECH [default: None] CMAQ chemical mechanism. Must match Mechanism variable setting in the CCTM build script.
- EMIS [default: 2013ef]
- APPL [default: SE52BENCH]
  CCTM executable identifier. Must match APPL Variable setting in the CCTM build script.
- RUNID [default: $VRSN_compiler_APPL] Run ID used to track version number, compiler, and application case name.
- EXEC [default: CCTM_$APPL_$EXECID]
  The name of the CCTM executable.

6.4.2.5.1 MPI Configuration

- `NPCOL_NPROW [default: 1 1]`
  The numbers of columns and rows for decomposing the modeling domain in an MPI configuration. The product of this pair of numbers must equal the total number of processors allocated to the CCTM simulation. For serial or single-processor MPI runs set to `1 1`. For multi-processor simulations, the number of columns (i.e, the first number in the pair) should be greater than or equal to the number of rows. For example, for an 8 processor MPI simulation, set to `4 2`
- `NPROCS [default: 1]`
  Number of processors to allocate for the CCTM simulation; equal to the product of NPCOL x NPROW. For serial or single-processor MPI runs set to `1`, otherwise set to the product of the two numbers used in NPCOL_NPROW.

6.4.2.5.2 Vertical extent

- `NZ [default: 35]` Set the number of vertical layers.

6.4.2.5.3 Timestep Configuration

- `NEW_START_TRUE [default: TRUE]` For a model restart set to FALSE
- `START_DATE`
  Simulation start date in Gregorian format (YYYY-MM-DD)
- `END_DATE` Simulation end date in Gregorian format (YYYY-MM-DD)
- `STTIME`
  Simulation start time (HHMMSS)
- `NSTEPS [default: 240000]`
  Number of simulation time steps (HHMMSS)
- `TSTEP [default: 010000]`
  Simulation output time step interval (HHMMSS)

6.4.2.5.4 CCTM Configuration Options

- `LOGFILE [default: $BASE/$APPL.log]`
  Uncomment to capture CCTM standard output to a log file; the LOGFILE variable sets the name and location of the log.
- `GRID_NAME [default: CMAQ-BENCHMARK]`
  Name of the grid definition contained in the GRIDDESC file that specifies the horizontal grid for the current application of the model.
- `GRIDDESC [default: $CMAQ_HOME/scripts/GRIDDESC1]`
  Grid description file for setting the horizontal grid definition.
- `CTM_APPL [default: ${RUNID}_${YYYYMMDD}]`
  CCTM log file naming extension.
- `CONC_SPCS [if commented out, all species]`
  Model species to be written to the CCTM CONC file.

- `CONC_BLEV_ELEV [if commented out, all layers]`
  Vertical model layer range for the CONC-file concentrations; this variable sets the lower and upper layers over which to output the CONC file.
- `AVG_CONC_SPCS [if commented out, output all species]`
  Model species for calculating integral average concentrations for each output time step. Options can be any of the standard output species that are written to the CCTM CONC file. The species in this list will be written to the ACONC output file.
- `ACONC_BLEV_ELEV [default: if commented out, all layers]`
  Vertical model layer range for integral average concentrations; this variable sets the lower and upper layers over which to calculate integral average concentrations. For example, setting this variable to "1 5" will produce integral average concentrations for model layers 1 through 5.
- `ACONC_END_TIME [default: N]`
  Change the time stamp of the ACONC file output time step from the default of the beginning of the hour to the end of the hour.
  - `Y`: Set the time stamp to the end of each hour.
  - `N`: Set the time stamp to the beginning of the hour.
- `EXECUTION_ID`
  The name of the CCTM executable; automatically set by the script.

6.4.2.5.5 Synchronization Time Step and Tolerance Options

- `CTM_MAXSYNC [default: 300]`
  Maximum synchronization time step in seconds
- `CTM_MINSYNC [default: 60]`
  Minimum synchronization time step in seconds
- `SIGMA_SYNC_TOP [default: .70]`
  Top sigma level thru which sync step determined
- `ADV_HDIV_LIM [default: .95]` Maximum horizontal division limit for advection time step adjustment
- `CTM_ADV_CFL [default: .95]`
  Maximum Courant–Friedrichs–Lewy (cfl) condition
- `RB_ATOL [default: 1.0E-09]`
  Global Rosenbrock (ROS3) chemistry solver absolute tolerance

6.4.2.5.6 Science Options

- `CTM_WB_DUST [default: Y]`
  Setting to calculate in-line windblown dust emissions in CCTM. Setting this variable to Y requires the availability of gridded land use input files that include the following BELD USGS land use classifications: shrubland, shrubgrass, and sprsbarren. See Chapter 8 for a description of the DUST_LU_1 and DUST_LU_2 input files. Comment out variable or set to Y to turn on; set to N to turn off.

- `CTM_ERODE_AGLAND [default: Y]`
  Setting to use optional erodible agricultural land classifications for computing windblown dust emissions from agricultural land. Setting this variable to Y requires the availability of gridded crop timing data that describe planting start dates, planting end dates, and harvesting end dates for 18 crop types. See Chapter 8 for a description of the CROPMAP01, CROPMAP04, and CROPMAP08 input files. If CTM_WB_DUST is set to N, this setting will be ignored. Set to Y to turn on; comment out variable or set to N to turn off.
- `CTM_WBDUST_BELD [default: BELD3]` Landuse database for identifying dust source regions; ignore if `CTM_WB_DUST = N`
  - `BELD3`
    Use BELD3 landuse data
  - `BELD4` Use BELD4 landuse data
- `CTM_LTNG_NO [default: Y]`
  Setting to activate lightning NO emissions. Setting this variable to Y requires additional variables to define the configuration of the lightning NO emissions calculation. See the settings for `LTNGNO`, `LTNGPARAMS`, `NLDN_STRIKES`, and `LTNGDIAG` below. Set to Y to turn on; comment out variable or set to N to turn off.
- `CTM_WVEL [default: Y]`
  Setting to output the CCTM-calculated vertical velocities to the CONC file. Set to Y to turn on; comment out variable or set to N to turn off.
- `KZMIN [default: Y]`
  If KZMIN is set to Y, CCTM will read the urban land use fraction variable (PURB) from the GRID_CRO_2D meteorology file and use this information to determine the minimum eddy diffusivity in each grid cell. In CMAQv5, grid cells that are predominantly urban use a KZMIN value of 1.0 m2/s and non-urban cells use a value of 0.01 m2/s. If this variable is set to N, the PURB variable will not be used and a uniform KZMIN value of 1.0 m2/s will be used throughout the modeling domain.
- `CTM_ILDEPV [default: Y]`
  Calculate in-line deposition velocities. Comment out variable or set to Y to turn on; set to N to turn off.
- `CTM_MOSAIC [default N]`
  Calculate land use specific deposition velocities and fluxes.
- `CTM_FST [default: N]`
  Use MOSAIC method to get land-use specific stomatal flux.
- `CTM_ABFLUX [default: Y]`
  Activate fertilizer ammonia bidirectional flux for in-line emissions and deposition velocities. If CTM_ILDEPV is set to N this variable is ignored. Setting this variable to Y requires four additional input files that include gridded fractional crop distributions (B4LU_file), soil properties (E2C_Soilfile), fertilizer conditions (E2C_Fertfile), and an agricultural soil initial conditions file (INIT_MEDC_1). Activation of this setting will produce additional variables in the output dry deposition file. See Chapter 8 for a description of the required input files. Set to Y to turn on; comment out or set to N to turn off.
- `CTM_HGBIDI [default: N]`
  Activate mercury bidirectional flux for in-line emissions and deposition velocities. If

CTM_ILDEPV is set to N this variable is ignored. Activation of this setting will produce additional variables in the output dry deposition file. Set to Y to turn on; comment out or set to N to turn off.

- `CTM_SFC_HONO [default: Y]`
  Calculate surface HONO interactions. If CTM_ILDEPV is set to N this variable is ignored. Comment out or set to Y to turn on; set to N to turn off.
- `CTM_GRAV_SETL [default Y]`
  Activate gravitational sedimentation for aerosols. Comment out or set to Y to turn on; set to N to turn off.
- `CTM_BIOGEMIS [default: Y]`
  Calculate biogenic emissions. Comment out or set to Y to turn on; set to N to turn off. If this option is activated, several additional variables must be set (see the In-line biogenic emissions configuration settings)

- `CTM_PT3DEMIS [default: Y]`
  Calculate plume rise for elevated point sources. Set to Y to turn on; comment out or set N to turn off. If this option is activated several additional variables must be set (see the Inline emissions configuration settings) following variables must be set.
- `CTM_ZERO_PCSOA [default: 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.

6.4.2.5.7 Process analysis options

- `CTM_PROCAN [default: N]`
  Activate process analysis in the CCTM. Set this to Y and use $CMAQ_DATA/pacp/pacp.inp to configure the integrated process rate and integrated reaction rate settings for the CCTM. Additional process analysis output files will be created when this setting is activated.
- `PA_BCOL_ECOL [default: None]`
  Modeling grid domain column range for the process analysis calculations. Set to the two digits representing the beginning and ending column number bounding the process analysis domain.
- `PA_BROW_EROW [default: None]`
  Modeling grid domain row range for the process analysis calculations. Set to the two digits representing the beginning and ending row number bounding the process analysis domain.
- `PA_BLEV_ELEV [default: None]`
  Modeling grid domain layer range for the process analysis calculations. Set to the two digits representing the bottom and top layer numbers bounding the process analysis domain.

6.4.2.5.8 I/O Controls

- `IOAPI_LOG_WRITE [default: Y]`
  Set to T to turn on excess WRITE3 logging by the I/O API.
- `FL_ERR_STOP [default: N]`
  Set to T to configure the program to exit if inconsistent headers are found in the input files.

- `PROMPTFLAG [default: N]`
  Turn on I/O-API PROMPTFILE interactive mode. Set to T to require interactive prompts for different I/O API operations.

- `IOAPI_OFFSET_64 [default: N]`
  I/O API setting for large time step records. If your output time step is going to produce data that are >2GB per time step, then this needs to be set to YES.

6.4.2.5.9 Aerosol Diagnostics Controls

- `CTM_AVISDIAG [default: N]`
  Output visibility diagnostics file. Set to Y to turn on; comment out or set to N to turn off.
- `CTM_PMDIAG [default: N]`
  Output aerosol diagnostics and properties file. Set to Y to turn on; comment out or set to N to turn off.
- `CTM_APMDIAG [default: N]`
  Output hourly average aerosol diagnostics and properties file. Set to Y to turn on; comment out or set to N to turn off.
- `APMDIAG_BLEV_ELEV [default: None]`
  Modeling grid domain layer range for the hourly average aerosol diagnostics and properties file. Set to the two digits representing the bottom and top layer numbers to bound the output domain.

6.4.2.5.10 Diagnostic Output Flags

- `CTM_CKSUM [default: Y]`
  Write science processes summaries to the standard output. Impacts run speed and log file output size. Comment out or set to Y to turn on; set to N to turn off.
- `CLD_DIAG [default: N]`
  Output an hourly wet deposition diagnostic file (CTM_WET_DEP_2) that includes convective wet deposition estimates. Set to Y to turn on; comment out or set to N to turn off.
- `CTM_PHOTDIAG [default: N]`
  Output in-line photolysis rates and associated data to diagnostic netCDF output files. The file CTM_RJ_1 contains gridded photolysis rates for O3 (JO3O1D) and NO2 (JNO2) that include both clear-sky and cloud effects, total downward irradiance at the surface (ETOT_SFC_W), aerosol optical depth (TAU_AERO_W), total optical depth (TAU_TOT_W), optical depth of ozone above the model domain (TAUO3_TOP_W), Rayleigh optical depth above the model domain (TAU_RAY_W), and surface albedo (ALBEDO_W). The file CTM_RJ_2 contains gridded photolysis rates for all other photolysis reactions in the selected chemical mechanism. Set to Y to turn on; comment out or set to N to turn off.
- `CTM_SSEMDIAG [default: N]`
  Output the calculated sea salt emissions to a diagnostic netCDF output file (CTM_SSEMIS_1). Set to Y to turn on; comment out or set to N to turn off.

- `CTM_DUSTEM_DIAG [default: N]`
  Output the in-line dust emissions to a diagnostic netCDF output file (CTM_DUST_EMIS_1). The diagnostic file includes not only the total dust emissions, but also dust emissions by land use category and dust model parameters, such as gridded erodible land use fractions. Set to Y to turn on; comment out or set to N to turn off.
- `CTM_DEPV_FILE [default: N]`
  Output an hourly diagnostic file (CTM_DEPV_DIAG) for the in-line deposition velocity calculations. If CTM_ILDEPV is set to N this variable is ignored. Set to Y to turn on; comment out or set to N to turn off.
- `VDIFF_DIAG_FILE [default: N]`
  Output a diffusion and aero gravitational sedimentation diagnostic file. Set to Y to turn on; comment out or set to N to turn off.
- `CTM_AOD [default N]`
  Output an aerosol optical depth (AOD) calculation diagnostics file. Set to Y to turn on; comment out or set to N to turn off.
- `DISP [default: keep]`
  Controls the maintenance of existing log files.
  - `delete` delete output log if it already exists
  - `keep` abort simulation if output log exists

6.4.2.5.11 Inline emissions configuration

- `NPTGRPS [default: 1]`
  The number of input point-source elevated emission sector file groups. A maximum of 9 sectors is allowed.

- `STK_GRPS_##`
  Directory path and file name of the stack groups file for sector ##, where ## = 01, 02,…,NPTGRPS. Each ## refers to one of the plume rise point-source sectors.

- `STK_EMIS_##`
  Directory path and file name of the point emissions file for sector ##, where ## = 01, 02,…,NPTGRPS. Each ## refers to the one of the plume rise point-source sectors.

- `LAYP_STDATE [HHMMSS]`
  Start date for calculating elevated-point-source emissions.

- `LAYP_STTIME [HHMMSS]`
  Start time for calculating elevated-point-source emissions.

- `LAYP_NSTEPS [HHHHHH]`
  Number of time steps for calculating elevated-point-source emissions.

- `CTM_EMLAYS [default: max no of model layers]`
  Number of emissions layers for calculating elevated-point-source emissions. If not set (commented out), the maximum number of model layers will be used.

- `PT3DDIAG [default: N]`
  Output the in-line 3-D point-source emissions to a diagnostic netCDF output file (CTM_PT3D_DIAG). Set to Y to turn on; comment out or set to N to turn off.

- `PT3DFRAC [default: N]`
  Output the in-line 3-D point-source layer fractions to a diagnostic netCDF output file (PLAY_SRCID_NAME). Set to Y to turn on; comment out or set to N to turn off.

- `REP_LAYER_MIN [default: -1]`
  Minimum layer number for reporting plume rise values to the plume rise diagnostics file. Set to `-1` or comment out to report all layers.

6.4.2.5.12 Lightning NOx configuration

CMAQ Lightning NOx Module Documentation

- `LTNGNO [default:InLine]`
  Setting to define whether the lightning emissions calculation will be in-line or off-line. This variable can be set to a gridded netCDF file of lightning NO emissions to use emissions calculated with a preprocessor outside of CCTM. Setting this variable to "inline" activates the in-line emissions calculation in CCTM and requires the LTNGPARMS variable (see below) to define the configuration of the in-line emissions.

- `USE_NLDN [default: Y]`
  Use hourly NLDN strikes file to compute inline lighing NO emissions. Activating this setting requires the NLDN_STRIKES input file. Comment out or set to Y to turn on; set to N to turn off.

- `LTNGPARAMS [default: Y]`
  Use the lightning parameters configuration file to compute inline lightning NO emissions. When the variable `LTNGNO` is set to `inline`, this setting is used to define how the in-line emissions will be calculated. Commenting out this variable or setting it to Y will compute lightning NO from input hourly flash count observations. Setting this variable to N will compute lightning NO strictly from convective precipitation rates in the input meteorology data. When this variable is set to Y, an additional input lightning parameter file (LTNGPARMS_FILE) will need to be available that includes intercloud to cloud-to-ground flash ratios, scaling factors for calculating flashes using the convective precipitation rate, and the moles of NO per flash.

- `NLDN_STRIKES [default: None]`
  Hourly NLDN lightning strike netCDF FILE. Required when `LTNGNO` is set to `Inline` and `USE_NLDN` is set to `Y`; otherwise ignore this setting.

- `LOG_START [default: 0.9]`
  Convective precipitation (RC) value to transition the lightning NO emissions calculation from linear to log linear.

- `LTNGDIAG [default: N]`
  Output a lightning NO emissions diagnostics file. Set to `Y` to turn on; comment out or set to `N` to turn off.

- `LTNGPARMS_FILE [default: None]`
  Lightning parameters output netCDF file; ignore if `LTNGPARAMS = N`

- `LTNGOUT [default: None]`
  Lightning diagnostics output netCDF file; ignore if `LTNGDIAG = N`

6.4.2.5.13 In-line biogenic emissions configuration

- `GSPRO [default: None]`
  Directory path and file name for input ASCII speciation profiles.

- `B3GRD [default: None]`
  Grid-normalized biogenic emissions input netCDF file.

- `BIOG_SPRO [default: None]`
  Profile ID for speciating biogenic VOCs. This profile ID must be present in the GSPRO file.

- `BIOSW_YN [default: Y]`
  Use the frost dates switch file to determine whether to use winter or summer biogenic emissions. Comment out or set to Y to turn on; set to N to turn off.

- `BIOSEASON [default: None]`
  File name for the frost dates switch input netCDF file.

- `SUMMER_YN [default: Y]`
  Toggle for summer season normalized biogenic emissions. This variable is ignored if BIOSW_YN is set to Y. Comment out or set to Y to select summer season biogenic emissions factors; set to N to turn off.

- `PX_VERSION [default: Y]`
  Setting to indicate whether the Pleim-Xiu land-surface model was used for the input meteorology. If this setting is set to Y the input meteorology data must include soil moisture (SOILM), soil temperature (SOILT), and soil type (ISLTYP) variables for use in the calculation of soil NO emissions.

- `INITIAL_RUN [default: N]`
  Set to Y if this is the first time that biogenic NO soil emissions will be calculated. If there is a previously created file, set to N.

- `SOILINP [default: None]`
  Directory path and file name of biogenic NO soil emissions file. If INITIAL_RUN is set to N or F, the soil NO emissions file from the previous day's simulation will be a required input file.

- `B3GTS_DIAG [default: N]`
  Write the inline biogenic emissions (mass units) to a diagnostic netCDF output file (B3GTS_S). Set to Y to turn on; comment out or set to N to turn off.

- `B3GTS_S`
  Diagnostic output netCDF file of biogenic emissions. This variable is ignored if B3GTS_DIAG is set to N.

6.4.2.5.14 Windblown dust emissions configuration

CMAQ Windblown Dust Module Documentation - `DUST_LU_1`
Input BELD "A" landuse netCDF file gridded to the modeling domain. Used if `CTM_WBDUST_BELD` is set to BELD3.

- `DUST_LU_2`
  Input BELD "TOT" landuse netCDF file gridded to the modeling domain. Used if `CTM_WBDUST_BELD` is set to BELD3.

- `MODIS_FPAR`
  Input MODIS FPAR time-varying vegetation netCDF file gridded to the modeling domain.

- `BELD4_LU`
  Input BELD4 landuse netCDF file gridded to the modeling domain. Used if `CTM_WBDUST_BELD` is set to BELD4.

- `CROPMAP01`
  Input beginning planting dates netCDF file gridded to the modeling domain.

- `CROPMAP04`
  Input end planting dates netCDF file gridded to the modeling domain.

- `CROPMAP08`
  Input end harvesting dates netCDF file gridded to the modeling domain.

6.4.3 Compiling and Running

6.4.3.1 Compile CCTM

Chapter 5 provides an overview of how to install and compile the CMAQ pre-processor programs for a test simulation. Follow those steps (summarized below) to compile new versions of CCTM.

1. Compile Bldmake, the CMAQ source code and compilation management program. This needs to be done only once—the first time CMAQ is installed.

- Configure the CCTM build script to use the config_cmaq.csh script, which points to the available `I/O API` and `netCDF` libraries.

- Configure the CCTM build script for your application by setting the compilation configuration variables described above.
- Invoke the build script to create an executable:

```
cd $CMAQ_HOME/CCTM/scripts
./bldit.cctm |& tee build.cctm.log
```

6.4.3.2 Run CCTM

Set the run script settings according to the execution configuration variables described above. Run CCTM using the following command:

```
cd $CMAQ_HOME/CCTM/scripts
./run_cctm.csh |& tee run_cctm.log
```

6.5 CHEMMECH and CSV2NML

6.5.1 Description

The program CHEMMECH generates mechanism source code files for all chemical mechanism-dependent CMAQ programs. Using an ASCII mechanism definition file as input, the Fortran program CHEMMECH creates all of the Fortran files that define the gas-phase chemical mechanisms for the CMAQ programs. The C-Shell script CSV2NML converts a comma-delimited text file that defines the processes (e.g., input as emissions, input through boundary conditions, transport, deposition) impacting the concentrations of each model species to a NAMELIST file for input to the CMAQ programs. In combination the Fortran source and NAMELIST files define chemical mechanisms in the CMAQ programs.

Implementing new mechanisms created by CHEMMECH and CSV2NML in the CMAQ programs is a two-step process. CHEMMECH generates the mechanism RXNS source files that must be used in the compilation of CMAQ source code into an executable. CSV2NML generates species NAMELIST files that are input to the CMAQ programs during execution. Care must be taken to ensure that the RXNS and NAMELIST files are consistent with each other in order to correctly update or implement new mechanisms in CMAQ.

CHEMMECH reads in a mechanism definition (mech.def) text file that lists the stoichiometry and kinetics of a photochemical reaction mechanism. The program converts the mech.def file to two RXNS files, RXNS_DATA_MODULE.F90 and RXNS_FUNC_MODULE.F90 that get compiled with the CMAQ source code into a new executable. The source files created by CHEMMECH must be manually moved to the correct directory location in the CMAQ source code directories to be available during compilation of the CMAQ programs. The mechanism files for CMAQ are found in $CMAQ_HOME/CCTM/src/MECHS/$Mechanism, where $Mechanism is the unique ID of a chemical mechanism (e.g., cb05e51_aq6_aq).

CSV2NML reads in a series of CSV files that define the processes that impact the concentrations of all of the CMAQ species. The CSV files are converted to NAMELIST files that are invoked at execution of the various CMAQ programs. Environment variables in the run scripts for ICON, BCON, and CCTM must be set to point to the NAMELIST files for a particular mechanism.

See Chapter 9 for details on how to update existing mechanisms or create new mechanisms in CMAQ.

6.5.2 Files, configuration, and environment variables

Figure 7□5 shows the input and output files and configuration options for CHEMMECH and CSV2NML. The full set of mechanism files required by the CMAQ programs is generated in two steps. In the first step, the program CHEMMECH is run with the mechanism definition file, mech.def, provided as input. The resulting RXNS files are then input to the CMAQ build scripts to compile CMAQ with a new chemical mechanism configuration. CSV2NML is used to convert the species definition files from CSV format to NAMELIST files. The NAMELIST files are used as inputs to the CMAQ programs ICON, BCON, or CCTM to define the processes that will impact each model species. Three NAMELIST files define the processes for gas-phase species (GC.nml), aerosol species (AE.nml), and nonreactive species (NR.nml).

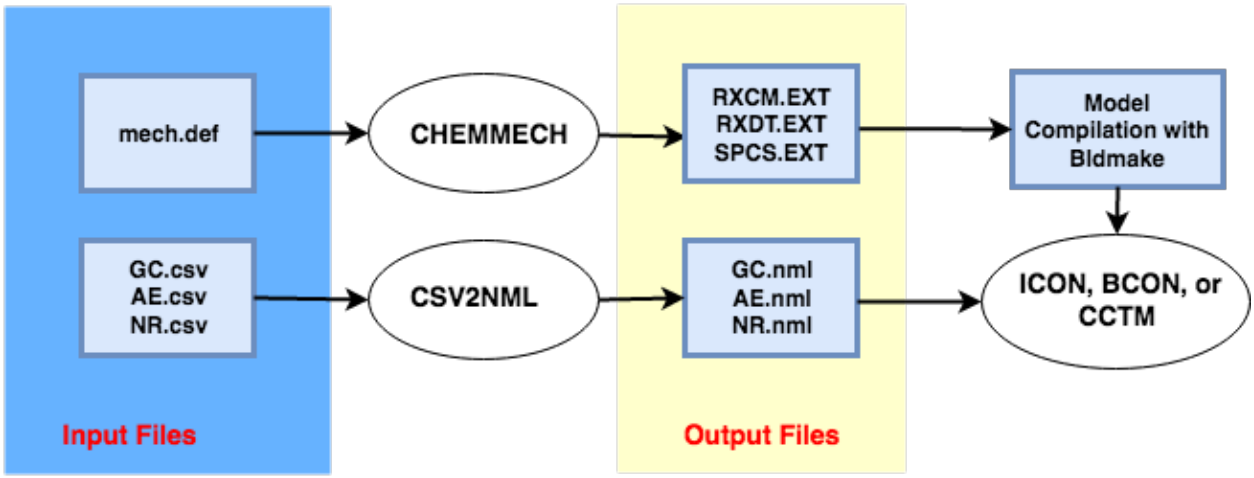


Figure 7□5. CHEMMECH and CSV2NML input and output files

To implement a new mechanism in CMAQ, start with a mechanism definition (mech.def) file and CSV species files from an existing mechanism in the model. Edit the mech.def file to include the new reactions, species, and reaction rates and provide this new mech.def file as input to CHEMMECH. Edit the CSV species files to include the new species and provide these files as input to CSV2NML. Detailed examples of updating an existing mechanism and adding a new mechanism to CMAQ are provided in Chapter 9. Neither CHEMMECH nor CSV2NML requires horizontal grid, vertical layer, or temporal settings.

6.5.2.1 CHEMMECH input files

Table 7-9. CHEMMECH input files

File Name	Format	Description
MCFL (mech.def)	ASCII	CMAQ mechanism definition file; photochemical mechanism listing with both mechanistic and kinetic information about all reactions that compose a chemical mechanism

6.5.2.2 CHEMMECH output files

Table 7□10. CHEMMECH output files

File Name	Format	Description
RXCM.EXT	ASCII	Mechanism common INCLUDE file; lists all of the chemical mechanism variables and parameters
RXDT.EXT	ASCII	Mechanism data INCLUDE file; chemical mechanism definition formatted as DATA blocks to be read in as CMAQ source code
SPCS.EXT	ASCII	Species INCLUDE file; not used

The location of the CHEMMECH output files is set in the run script by the variable Opath. To compile a version of the CMAQ programs that use the INCLUDE files created by CHEMMECH, these output INCLUDE files need to be moved to a new directory under the `$CMAQ_HOME/models/mechs/release` directory. Point the CMAQ build scripts to this new directory through the "Mechanism" variable.

6.5.2.3 CSV2NML input files

Detailed descriptions of the formats of the files shown in Table 7-11 are provided in Chapter 8.

Table 7□11. CSV2NML input files

File Name	Format	Description
GC.csv	ASCII	Gas-phase species process parameters. This file defines the source and sink processes that impact the concentrations of every gas-phase species in the chemical mechanism.
AE.csv	ASCII	Aerosol-phase species process parameters. This file defines the source and sink processes that impact the concentrations of every aerosol-phase species in the chemical mechanism.
NR.csv	ASCII	Nonreactive species process parameters. This file defines the source and sink processes that impact the concentrations of every nonreactive species in the chemical mechanism.

6.5.2.4 CSV2NML output files

Table 7□12. CSV2NML output files

File Name	Format	Description
GC.nml	ASCII	Gas-phase species process parameters. This file defines the source and sink processes that impact the concentrations of every gas-phase species in the chemical mechanism
AE.nml	ASCII	Aerosol-phase species process parameters. This file defines the source and sink processes that impact the concentrations of every aerosol-phase species in the chemical mechanism
NR.nml	ASCII	Nonreactive species process parameters. This file defines the source and sink processes that impact the concentrations of every nonreactive species in the chemical mechanism

6.5.2.5 Execution Configuration Variables

The environment variables listed here are invoked at run time and are set in the CHEMMECH run script. The default run script is called MP.saprc99.csh.

- `Xpath [default: $BASE]`
  Executable directory path
- `EXEC [default: CHEMMECH]`
  Executable name
- `Mechanism [default: None]`
  Name of the output mechanism.
- `Opath [default: ../exts]`
  Output file directory path
- `Mpath [default: ../exts]`
  Mechanism definition file directory path
- `MECHDEF [default: None]`
  Mechanism definition file name
- `MAPPING_ROUTINE [default: None]`

- `SPCSDATX [default: $Opath/SPECIES.ext]`
  Name of output species INCLUDE file
- `RXNS_DATA_MODULE [default: $Opath/RXNS_DATA_MODULE.F90]`
  Name of output mechanism data Fortran file
- `RXNS_FUNC_MODULE [default: $Opath/RXNS_FUNC_MODULE.F90]`
  Name of output mechanism common Fortran file
- `EQNS_KPP_FILE [default: None]`

- `SPCS_KPP_FILE [default: None]`

6.5.3 Compiling and Running

6.5.3.1 Compile Chemmech

To compile CHEMMECH, run the build script:

```
cd $CMAQ_HOME/UTIL/chemmech/scripts
./bldit_chemmech.csh |& tee bldit_chemmech.log
```

To port CHEMMECH to different compilers, change the compiler names, locations, and flags in the config_cmaq.csh script.

6.5.3.2 Run Chemmech

Set the run script settings according to the execution configuration variables described above. Run CHEMMECH using the following command:

```
cd $CMAQ_HOME/UTIL/chemmech/scripts
./run_chemmech.csh |& tee run_chemmech.log
```

6.5.3.3 CSV2NML usage

The CSV2NML script is configured to read in a CSV file from the command line and output a NAMELIST file that can be used with CMAQ. An example of how to use CSV2NML to create a gas-phase species NAMELIST file is include below:

```
cd $CMAQ_HOME/UTIL/nml/scripts
./csv2nml.csh GC.CSV
```

There is also a script to convert an existing namelist file to a CSV

```
cd $CMAQ_HOME/UTIL/nml/scripts
./nml2csv.csh GC.nml
```

6.6 CREATE_EBI

6.6.1 Description

The program CREATE_EBI generates Fortran source code for the mechanism-dependent Euler Backward Iterative (EBI) solver. This utility program allows users to create new EBI solver code when mechanisms are modified or added to CMAQ. The source code generated by CREATE_EBI should be used to build versions of the CCTM that use the new or modified chemistry mechanisms.

See Chapter 9 for details on how to update existing mechanisms or create new mechanisms in CMAQ.

6.6.2 Files, configuration, and environment variables

To implement a new mechanism in CMAQ, start with a mechanism definition (mech.def) file and CSV species files from an existing mechanism in the model. Edit the mech.def file to include the new reactions, species, and reaction rates and provide this new mech.def file as input to the program CHEMMECH. CHEMMECH will output a RXNS_DATA_MODULE.F90 file, which is used as input to CREATE_EBI.

6.6.2.1 CREATE_EBI input files

Table 7-13. CREATE_EBI input files

File Name	Format	Description
RXNS_DATA_SRC.F90	ASCII	CMAQ mechanism reaction listing in Fortran 90 format; output from the program CHEMMECH

6.6.2.2 CREATE_EBI output files

Table 7□14. CREATE_EBI output files

File Name	Format	Description
*.F	ASCII F90	Fortran 90 source code for the CCTM EBI chemistry solver
RXNS_DATA_MODULE.F90	ASCII F90	Mechanism data Fortran source file; chemical mechanism definition formatted as DATA blocks to be read in as CMAQ source code

The location of the CREATE_EBI output files is set in the run script by the variable OUTDIR. To compile a version of the CMAQ programs that use the F90 files created by CREATE_EBI, these output F90 files need to be moved to a new directory under the `$CMAQ_HOME/CCTM/src/gas` directory. Point the CMAQ build scripts to this new directory through the "Mechanism" variable.

6.6.2.3 Compilation Configuration Variables

- `GC_NAME [default: None]`
  Name identifier for gas phase mechanisms
  - `CB6R3`
    Carbon Bond version 6 revision 3
  - `CB05E51`
    Carbon Bond 05 with modifications for CMAQ version 5.1
  - `CB05MP51`
    Carbon Bond 05 multipollutant mechanism for CMAQ version 5.1

  - `CB05TUCL`
    Carbon Bond 05 with modified toluene and chlorine chemistry
  - `CB05TUMP`
    Carbon Bond 05 with modified toluene and multipollutant chemistry
  - `SAPRC07TB`
    SAPRC07 with modified toluene chemistry
  - `SAPRC07TC`
    SAPRC07 with modified toluene chemistry
  - `SAPRC07TIC`
    SAPRC07 with modified toluene chemistry
  - `RACM2`
    RACM2 chemistry

- `AE_NAME [default: None]`
  Name identifier for particle phase mechanisms

  - `AE6`
    CMAQ aerosols version 6
  - `AE6I`
    CMAQ aerosols version 6i

- `AQ_NAME [default: AQ]`
  Name identifier for the CMAQ aqueous phase mechanism

6.6.2.4 Execution Configuration Variables

The environment variables listed here are invoked at run time and are set in the CREATE_EBI run script.

- `EXEC [default: CHEMMECH]`
  Executable name
- `GC_NAME [default: None]`
  Name identifier for gas phase mechanisms

  - `CB6R3`
    Carbon Bond version 6 revision 3
  - `CB05E51`
    Carbon Bond 05 with modifications for CMAQ version 5.1
  - `CB05MP51`
    Carbon Bond 05 multipollutant mechanism for CMAQ version 5.1
  - `CB05TUCL`
    Carbon Bond 05 with modified toluene and chlorine chemistry
  - `CB05TUMP`
    Carbon Bond 05 with modified toluene and multipollutant chemistry
  - `SAPRC07TB`
    SAPRC07 with modified toluene chemistry

  - `SAPRC07TC`
    SAPRC07 with modified toluene chemistry
  - `SAPRC07TIC`
    SAPRC07 with modified toluene chemistry
  - `RACM2`
    RACM2 chemistry

- `AE_NAME [default: None]`
  Name identifier for particle phase mechanisms
  - `AE6`
    CMAQ aerosols version 6
  - `AE6I`
    CMAQ aerosols version 6i
- `AQ_NAME [default: AQ]`
  Name identifier for the CMAQ aqueous phase mechanism
- `OUTDIR [default: ../output]`
  Output file directory path
- `COPYRT_FLAG`
- `CVS_HDR_FLAG`
- `PAR_NEG_FLAG [default: F]` Include PAR negative stoichiometry.
  - `T` for Carbon Bond mechanisms
  - `F` for SAPRC and RACM mechanisms
- `DEGRADE_SUBS`
- `NO2EX_CYCLE`
- `MECH_NO`
  Mechanism name for nitric oxide
- `MECH_NO2`
  Mechanism name for nitrogen dioxide
- `MECH_NO2EX`
  SAPRC, RACM Mechanism name for excited nitrogen dioxide; not in Carbon Bond
- `MECH_O3`
  Mechanism name for ozone
- `MECH_O3P`
  Mechanism name for ground state oxygen atom
  - `O` for Carbon Bond mechanisms
  - `O3P` for SAPRC and RACM mechanisms
- `MECHO_O1D`
  Mechanism name for excited state oxygen atom
- `MECH_OH`
  Mechanism name for hydroxyl radical

- `MECH_HO2`
  Mechanism name for hydroperoxy radical
- `MECH_HONO`
  Mechanism name for nitrous acid
- `MECH_HNO4`
  Mechanism name for peroxynitric acid
  - `PNA` for Carbon Bond mechanisms
  - `HNO4` for SAPRC and RACM mechanisms
- `MECH_PAN`
  Mechanism name for peroxy acetyl nitrate
- `MECH_C2O3`
  Mechanism name for peroxy acetyl radical
  - `C2O3` for Carbon Bond mechanisms
  - `MEC03` for SAPRC and RACM mechanisms
- `MECHO_NO3`
  Mechanism name for nitrate radical
- `MECH_N2O5`
  Mechanism name for dinitrogen pentoxide

6.6.3 Compiling and Running

6.6.3.1 Compile CREATE_EBI

To compile CREATE_EBI, invoke the build file at the command line:

```
cd $CMAQ_HOME/UTIL/create_ebi/scripts
./bldit.create_ebi.csh |& tee build.create_ebi.log
```

To port CREATE_EBI to different compilers, change the `COMPILER` variable in the bldit script.

6.6.3.2 Run CREATE_EBI

Set the run script settings according to the execution configuration variables described above. Run CREATE_EBI using the following command.

```
cd $CMAQ_HOME/UTIL/create_ebi/scripts
./run.create_ebi.csh |& tee run.create_ebi.log
```

6.7 ICON

6.7.1 Description

The program ICON prepares chemical initial conditions (ICs) for CCTM from either ASCII vertical profiles or from an existing CCTM output concentration (CONC) file. ICON creates an output file with a single time step that represents the chemical conditions in each grid cell at the beginning of a CCTM simulation. The ICs can be either spatially uniform or variable across the model grid, depending on the source of the initial chemical concentration data. If deriving ICs from the ASCII vertical profiles, ICON can create only spatially uniform ICs *within* each model layer; it can create different ICs *across* model layers. From CONC files, ICON can extract spatially varying ICs, either on the same grid cell resolution, as a windowed modeling domain, or for a finer-resolution model grid (as for a nested simulation).

There are two distinct modes of operation for ICON, depending on the nature of the input data. When creating ICON executables, the user must specify whether the input data will be ASCII vertical profiles or a CONC file by selecting either "profile" or "m3conc", respectively, for the setting of the `ModType` variable. This variable determines the input module to use when creating an ICON executable.

CCTM can also be forced with initial conditions downscaled from global chemistry models (GCMs), such as GEOS-Chem and MOZART. ICON does not support the processing of data from GCMs. ICs derived from GCMs must be calculated with custom codes or scripts that are not available in the CMAQ distribution package. The CAMx developers (Ramboll Environ) have codes available for extracting regional model ICs from both GEOS-Chem and MOZART. Visit the Support Software section of www.CAMx.com to download these utilities.

6.7.2 Files, configuration, and environment variables

Figure 7□6 shows the input and output files and configuration options for ICON. A distinction is made between the options that are invoked at compilation versus those invoked at execution of the program. When compiling ICON, the user specifies a chemical mechanism to configure the gas-phase chemistry and aerosol mechanism used to create the chemical ICs. Setting the `ModMech` and `Mechanism` variables in the ICON compile script configures the program to use a specific set of mechanism namelist files to build an executable. Setting the `ModType` variable in the ICON compile script configures the program to input either a text file of static concentrations or a binary netCDF file of time-dependent concentrations for estimating ICs for CCTM. Separate ICON executables must be prepared for different mechanism and input file configurations.

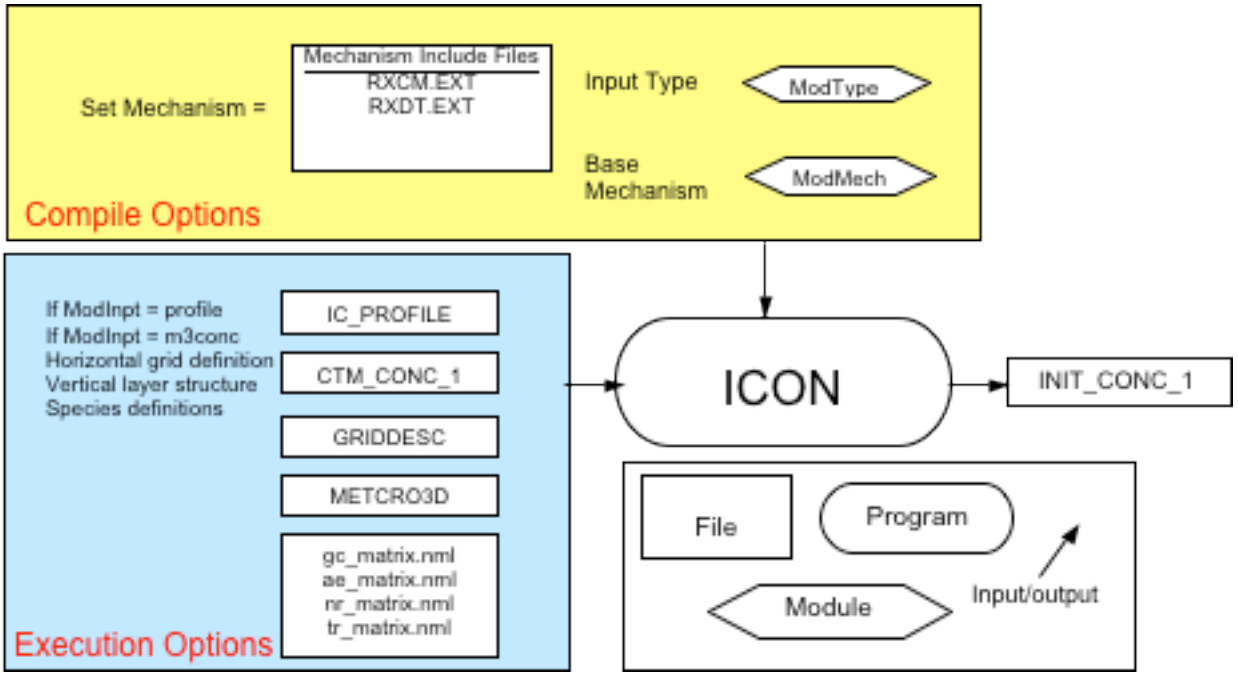


Figure 7□6. ICON input and output files

When ICON is run, it converts a data file of chemical ambient concentrations to ICs on a predefined model grid. Through the specification of the `ModType` variable in the ICON run script, ICON will input either an ASCII vertical profile file (IC_PROFILE) or an existing CCTM concentration file (CTM_CONC_1); the choice depends on how the user compiled the model. The IC input file provided by the user must have chemical speciation that is consistent with the mechanism configuration of the ICON executable. For example, if ICON was compiled to create ICs using the CB05 mechanism, the input IC profile data must be in terms of the CB05 mechanism. CMAQ is distributed with ASCII vertical profiles representing clean continental ICs for North America for the following chemical mechanisms: cb05_ae6_aq, saprc07tb_ae6_aq, racm2_aq6_aq, and saprc99_ae6_aq. It is the user's responsibility to generate IC inputs for other mechanism configurations.

The horizontal grid and vertical layer structures for ICON are defined at execution through the input of a grid description (GRIDDESC) file and a meteorology cross-point 3□D (MET_CRO_3D) file, respectively. ICON interpolates between the input vertical layer structure and output layer structure if they are different.

6.7.2.1 ICON input files

Table 7□15. ICON input files

File Name	Format	Description
IC_PROFILE	ASCII	Vertical chemical profiles from which to derive initial conditions; this file is created by the user; used only when the IC environment variable is set to "profile"

File Name	Format	Description
CTM_CONC_1	GRDDED3	Name and location of the CMAQ concentration file from which to derive initial conditions; this file is output from CCTM; used only when the BC environment variable is set to "m3conc"
MET_CRO_3D_CRS	GRDDED3	Name and location of the coarse-grid MET_CRO_3D file that is required for creating the vertical grid structure if this structure changes between nested simulations; this file is output by MCIP
MET_CRO_3D_FIN	GRDDED3	Name and location of the fine grid MET_CRO_3D file that is required if the vertical grid structure changes between nested simulations; this file is output by MCIP
GRIDDESC	ASCII	Horizontal grid description file for defining the model grid; this file is output by MCIP or can be created by the user
LAYER_FILE	GRDDED3	3-D cross-point meteorology file for defining the vertical layer structure of the model grid; this file is output by MCIP
gc_matrix.nml	ASCII	Namelist file for defining the gas-phase species that are input to the model through the boundary
ae_matrix.nml	ASCII	Namelist file for defining the aerosol species that are input to the model through the boundary
nr_matrix.nml	ASCII	Namelist file for defining the nonreactive species that are input to the model through the boundary
tr_matrix.nml	ASCII	Namelist file for defining the tracer species that are input to the model through the boundary

6.7.2.2 ICON output files

Table 7□16. ICON output files

File Name	Format	Description
INIT_CONC_1	GRDDED3	Name and location of the gridded initial conditions data output on the model grid defined by GRID_NAME

The default location of the ICON output files is the $CMAQ_DATA/icon directory, controlled by the OUTDIR variable in the run script. The default naming convention for all ICON output files uses the APPL and GRID_NAME environment variables in the file name. For initial conditions created from existing CCTM CONC files, the Julian date is also used in the file name through the DATE environment variable. All of the file-naming variables for ICON outputs are set in the run script.

6.7.2.3 Compilation Configuration Variables

The configuration options listed here are set during compilation of the ICON executable. When these options are invoked they create a binary executable that is fixed to the specified configuration. To

change these options you must recompile ICON and create a new executable.

- `CopySrc`
  Uncomment to copy the source code into a working build (BLD) directory. If commented, onlhy the compiled object and executable files will be placed in the BLD directory.
- `MakeFileOnly`
  Uncomment to build a Makefile to compile the executable. Comment out to both create a Makefile and compile.
- `ModType: [default: module profile]`
  Defines the format of the initial conditions input files to be used by ICON.
  - `m3conc`: input a CCTM CONC file; used for nested simulations or windows of a parent domain
  - `profile`: input an ASCII vertical profiles file
  - `tracer`: use the tracer namelist file to create ICs of tagged tracer species
- `Mechanism: [default: cb05e51_ae6_aq]`
  Specifies the gas-phase, aerosol, and aqueous-phase chemical mechanisms for which to create initial conditions. The choices for the *Mechanism* variable are the mechanism directory names under the `$CMAQ_HOME/CCTM/src/MECHS` directory. Also see the Mechanism Definitions Table). Examples include:
  - `cb6r3_ae6_aq`: CB6, revision 3 gas-phase mechanism, sixth-generation CMAQ aerosol mechanism with sea salt and speciated PM Other, aqueous/cloud chemistry
  - `cb05e51_ae6_aq`: CB05 gas-phase mechanism with CMAQv5.1 updates, sixth-generation CMAQ aerosol mechanism with sea salt and speciated PM Other, aqueous/cloud chemistry
  - `cb05tucl_ae6_aq`: CB05 gas-phase mechanism with active chlorine chemistry, updated toluene mechanism, sixth-generation CMAQ aerosol mechanism with sea salt and speciated PM Other, aqueous/cloud chemistry
  - `cb05tump_ae6_aq`: CB05 gas-phase mechanism with active chlorine chemistry, updated toluene mechanism, mercury, and air toxics, sixth-generation CMAQ aerosol mechanism with sea salt and speciated PM, aqueous/cloud chemistry; this is the CMAQv5 multipollutant mechanism
  - `saprc07tb_ae6_aq`: SAPRC-07 gas-phase mechanism with toluene updates and sixth-generation CMAQ aerosol mechanism
  - `racm2_ae6_aq`: RACM2 gas-phase mechanism with toluene updates and sixth-generation CMAQ aerosol mechanism
- `Tracer [default trac0]`
  Specifies tracer species. Invoking inert tracer species in CMAQ requires defining the tracers using namelist files and compiling the CMAQ programs with these files. The setting for this module corresponds to the directory name in the `$CMAQ_HOME/CCTM/src/MECHS` directory that contains the namelist files for the tracer configuration. The default setting is to not use any tracers.
  - `trac[n]`

6.7.2.4 Execution Configuration Variables

The environment variables listed here are invoked during execution of the program and are set in the BCON run script.

- `APPL [default: None]`
  ICON executable identifier. Must match APPL Variable setting in the ICON build script.
- `CFG [default: None]`
  Configuration identifier for the ICON simulation.
- `MECH [default: None]`
  CMAQ chemical mechanism. Must match Mechanism variable setting in the ICON build script.
- `EXEC: [default: ICON_${APPL}_${EXECID}.exe]`
  Executable to use for the simulation. The variable CFG is set in the ICON run script. The variable EXECID is set in the config_cmaq.csh configuration file.
- `GRIDDESC: [default: $CMAQ_HOME/scripts/GRIDDESC1]`
  Grid description file for setting the horizontal grid definition.
- `GRID_NAME: [default:CMAQ-BENCHMARK]`
  Name of the grid definition contained in the GRIDDESC file that specifies the horizontal grid for the current application of the model.
- `IOAPI_ISPH: [default: 20]`
  I/O API setting for spheroid type. See I/O API documentation for setsphere for more information.
- `IOAPI_OFFSET_64: [default: NO]`
  I/O API setting for large time-step records. If your output time step is going to produce data that are >2GB per time step, then this needs to be set to YES.
- `LAYER_FILE:[default: none]`
  Name and location of a MET_CRO_3D file for specifying the vertical layer structure for the current application of the model.
- `gc_matrix.nml: [default: none]`
  Gas-phase species namelist file. This file is used to configure the gas-phase species that will be output by BCON.
- `ae_matrix.nml: [default: none]`
  Aerosol-phase species namelist file. This file is used to configure the aerosol-phase species that will be output by BCON
- `nr_matrix.nml: [default: none]`
  Nonreactive species namelist file. This file is used to configure the nonreactive species that will be output by BCON
- `tr_matrix.nml: [default: none]`
  Tracer species namelist file. This file is used to configure the tracer species that will be output by BCON
- `OUTDIR: [default: $CMAQ_HOME/data/bcon]` Output data directory.
- `IC:`
  Sets the input file type. The setting of this variable determines how the run script sets the input and output environment variables.
  - `profile`: sets the output file name to include the tag "profile" in the name; uses the variable IC_PROFILE to point to an ASCII vertical profile file for input to ICON.

  - `m3conc`: used for nested simulations; sets the output file name to include a start date in the name; uses the variable CTM_CONC_1 to point to a CCTM CONC file for input to ICON.

- `DATE:`
  Sets the Julian date to use in naming the ICON output file for nested runs.
- `SDATE:[default: ${DATE}]`
  Julian start date for extracting boundary conditions from a CCTM CONC file for a nested simulation. If SDATE is not set, it will be set automatically from the CTM_CONC_1 file.
- `STIME: [default: 000000 ]`
  Start time for extracting boundary conditions from a CCTM CONC file for a nested simulation. If STIME is not set, it will be set automatically from the CTM_CONC_1 file.

6.7.3 Compiling and Running

6.7.3.1 Compile ICON

Chapter 5 provides an overview of how to install and compile the CMAQ pre-processor programs for a test simulation. Follow those steps (summarized below) to compile new versions of ICON.

1. Compile Bldmake, the CMAQ source code and compilation management program. This needs to be done only once—the first time CMAQ is installed.

- Configure the ICON use the config_cmaq.csh script, which points to the available I/O API and netCDF libraries.
- Configure the ICON build script for your application by setting the compilation configuration variables described above.
- Invoke the build script to create an executable:

```
cd $CMAQ_HOME/PREP/icon/scripts
./bldit.icon |& tee build.icon.log
```

6.7.3.2 Run ICON

Set the run script settings according to the execution configuration variables described above. Run ICON to produce initial conditions for the CCTM:

```
cd $CMAQ_HOME/PREP/icon/scripts
./run.icon |& tee icon.log
```

6.8 INLINE_PHOT_PREPROC

6.8.1 Description

The program INLINE_PHOT_PREPROC generates absorption cross-section/quantum yield (CSQY) data files for the CCTM inline photolysis module. This utility program allows users to create new CSQY tables when reaction rate data are modified or added to CMAQ. The data tables generated by INLINE_PHOT_PREPROC should be used to create the photochemistry data tables needed for inline photolysis configurations of the CCTM.

See Chapter 9 for details on how to update existing mechanisms or create new mechanisms in CMAQ.

6.8.2 Files, configuration, and environment variables

To implement new CSQY data in CMAQ, start with individual CSQY data files for each photolysis reaction in an applicable photochemical mechanism. Add to or modify these data to create the CCTM inline CSQY data table.

6.8.2.1 INLINE_PHOT_PREPROC input files

Table 7-17. INLINE_PHOT_PREPROC input files

File Name	Format	Description
RXNS_DATA_MODULE.F90	ASCII	CMAQ mechanism reaction listing in Fortran 90 format; output from the program CHEMMECH
CSQY_DATA_RAW	ASCII	Directory of photolysis reaction-specific absorption cross section and quantum yield data as a function of wavelength
WVBIN_FILE	ASCII	Wavelength bins for which to include CSQY data
FLUX_FILE	ASCII	Solar flux (photons/s/bin) by 0.05nm wavelength bin
WATER	ASCII	Water refractive indices by wavelength
INSOLUBLE	ASCII	Optical properties of soil aerosol material
DUST	ASCII	Optical properties of soil aerosol material
SOLUTE	ASCII	Optical properties of water soluble aerosol material
SOOT	ASCII	Optical properties of soot (BC) aerosol material
SEASALT	ASCII	Optical properties of seasalt aerosol material

6.8.2.2 INLINE_PHOT_PREPROC output files

Table 7□18. INLINE_PHOT_PREPROC output files

File Name	Format	Description
CSQY_DATA	ASCII	Tabulated CSQY data as a function of temperature and wavelength bin

File Name	Format	Description
PHOT_OPTICS	ASCII	Wavelength, Optical and Surface Albedo Parameters for CMAQ In-Line Photolysis calculation.

The location of the INLINE_PHOT_PREPROC output files is set in the run script by the variable OUTDIR. To compile a version of the CMAQ programs that use the files created by INLINE_PHO_PREPROC, copy the output files to a new directory under the `$CMAQ_HOME/CCTM/src/MECHS/$Mechanism` directory. Point the CMAQ build scripts to this new directory with the "Mechanism" variable.

6.8.2.3 Compilation Configuration Variables

- `Mechanism: [default: cb6r3_ae6_aq]`
  Specifies the gas-phase, aerosol, and aqueous-phase chemical mechanisms for which to create initial conditions. The choices for the *Mechanism* variable are the mechanism directory names under the `$CMAQ_HOME/CCTM/src/MECHS` directory. Also see the Mechanism Definitions Table). Examples include:
  - `cb6r3_ae6_aq`: CB6, revision 3 gas-phase mechanism, sixth-generation CMAQ aerosol mechanism with sea salt and speciated PM Other, aqueous/cloud chemistry
  - `cb05e51_ae6_aq`: CB05 gas-phase mechanism with CMAQv5.1 updates, sixth-generation CMAQ aerosol mechanism with sea salt and speciated PM Other, aqueous/cloud chemistry
  - `cb05tucl_ae6_aq`: CB05 gas-phase mechanism with active chlorine chemistry, updated toluene mechanism, sixth-generation CMAQ aerosol mechanism with sea salt and speciated PM Other, aqueous/cloud chemistry
  - `cb05tump_ae6_aq`: CB05 gas-phase mechanism with active chlorine chemistry, updated toluene mechanism, mercury, and air toxics, sixth-generation CMAQ aerosol mechanism with sea salt and speciated PM, aqueous/cloud chemistry; this is the CMAQv5 multipollutant mechanism
  - `saprc07tb_ae6_aq`: SAPRC-07 gas-phase mechanism with toluene updates and sixth-generation CMAQ aerosol mechanism
  - `racm2_ae6_aq`: RACM2 gas-phase mechanism with toluene updates and sixth-generation CMAQ aerosol mechanism
- `COMPILER`
  Compiler to use for building the program
  - `PGF90`
  - `INTEL`
  - `GFORT`

6.8.2.4 Execution Configuration Variables

The environment variables listed here are invoked at run time and are set in the CREATE_EBI run script. - `Mechanism: [default: cb05e51_ae6_aq]`

Specifies the gas-phase, aerosol, and aqueous-phase chemical mechanisms for which to create initial conditions. The choices for the *Mechanism* variable are the mechanism directory names under the `$CMAQ_HOME/CCTM/src/MECHS` directory. Also see the Mechanism Definitions Table). Examples include: - `cb6r3_ae6_aq`: CB6, revision 3 gas-phase mechanism, sixth-generation CMAQ aerosol mechanism with sea salt and speciated PM Other, aqueous/cloud chemistry - `cb05e51_ae6_aq`: CB05 gas-phase mechanism with CMAQv5.1 updates, sixth-generation CMAQ aerosol mechanism with sea salt and speciated PM Other, aqueous/cloud chemistry - `cb05tucl_ae6_aq`: CB05 gas-phase mechanism with active chlorine chemistry, updated toluene mechanism, sixth-generation CMAQ aerosol mechanism with sea salt and speciated PM Other, aqueous/cloud chemistry - `cb05tump_ae6_aq`: CB05 gas-phase mechanism with active chlorine chemistry, updated toluene mechanism, mercury, and air toxics, sixth-generation CMAQ aerosol mechanism with sea salt and speciated PM, aqueous/cloud chemistry; this is the CMAQv5 multipollutant mechanism - `saprc07tb_ae6_aq`: SAPRC-07 gas-phase mechanism with toluene updates and sixth-generation CMAQ aerosol mechanism - `racm2_ae6_aq`: RACM2 gas-phase mechanism with toluene updates and sixth-generation CMAQ aerosol mechanism - `USE_RXNS_MODULES [default: T]` Compatibility flag for CMAQ. Set to "T" for CMAQ version 5.1 and higher; set to "F" for older versions of CMAQ. - `WVL_AE_REFRAC [default: T]` Include spectral values of refractive indices for aerosol species. Only needed for CMAQv5.1 and higher; set to "F" for older versions of CMAQ. - `SPLIT_OUTPUT [default: T]` Split optical and CSQY output data to separate files. Only needed for CMAQv5.1 and higher; set to "F" for older versions of CMAQ.

6.8.3 Compiling and Running

6.8.3.1 Compile CREATE_EBI

To compile CREATE_EBI, invoke the build file at the command line:

```
cd $CMAQ_HOME/UTIL/create_ebi/scripts
./bldit.create_ebi |& tee build.create_ebi.log`
```

To port CREATE_EBI to different compilers, change the `COMPILER` variable in the bldit script.

6.8.3.2 Run CREATE_EBI

Set the run script settings according to the execution configuration variables described above. Run CREATE_EBI using the following command:

```
cd $CMAQ_HOME/UTIL/create_ebi/scripts
./run.create_ebi |& tee run.create_ebi.log
```

6.9 JPROC

6.9.1 Description

The program JPROC calculates daily clear-sky photolysis rates from look-up tables of molecular absorption cross-section and quantum yield (CSQY) data, and climatologically derived ozone-column

and optical depth data. The outputs from JPROC are ASCII look-up tables of daily clear-sky photolysis rates for photochemical reactions in a selected gas-phase photochemical mechanism at different altitudes, latitudes, and hours from noon. The photochemical mechanism from which these rates are derived is selected during compilation of JPROC. The altitudes (meters), latitudes (degrees), and hour angles (from noon) for which the rates are derived are hardwired in the JPROC source code.

CCTM currently uses an in-line photolysis option that calculates photolysis rates using predicted ozone and aerosols. JPROC is notused for the default configuration of ModPhot set to phot/inline). JPROC is required to produce daily photolysis rate look-up tables if CCTM is compiled with *ModPhot* set to phot/table.

6.9.2 Files, configuration, and environment variables

Figure 7□7 shows the input and output files for JPROC. Some options are invoked at compilation, while others are invoked with execution of the program. When compiling JPROC, the user specifies a chemical mechanism to indicate the gas-phase chemistry for which to calculate photolysis rates. Setting the *Mechanism* variable in the JPROC compile script configures the program to use a specific set of mechanism INCLUDE files to build an executable. JPROC executables are hard-wired to a specific mechanism configuration.

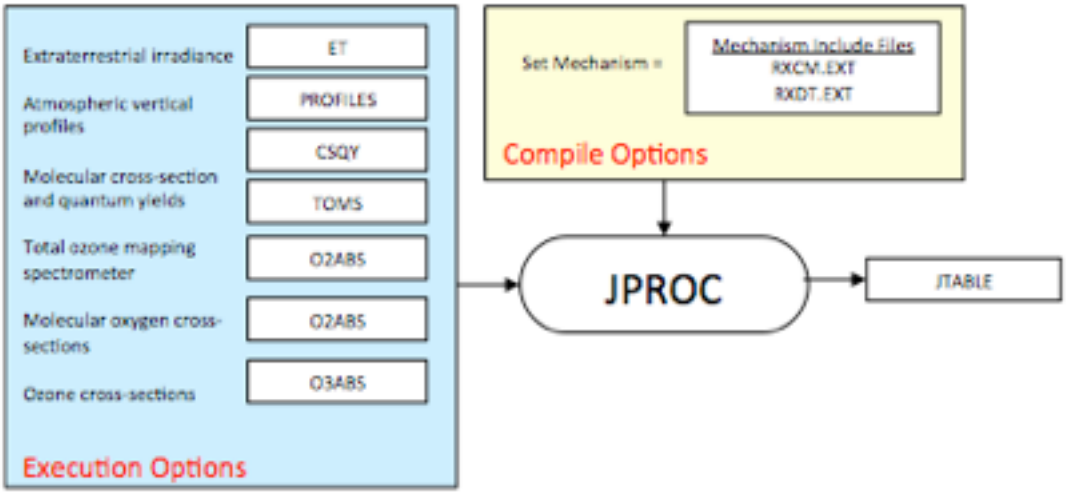


Figure 7□7. JPROC input and output files

While JPROC does not require any technical configuration at execution, such as domain specifications, there are several required and optional input files that the user must provide to the program. For the selected photochemical mechanism, the user must provide a set of molecular absorption CSQY data files that are consistent with the photolysis reactions in the mechanism. CMAQ is distributed with a full set of CSQY files for the Carbon Bond, SAPRC, and RACM photochemical mechanism versions supported by the model. If new mechanisms are added to CMAQ, the user must produce the appropriate CSQY data files for the added mechanism. The user also has the option of using the default atmospheric profiles contained in the PROFILES input file or using Total Ozone Mapping Spectrometer (TOMS) data to replace the climatologically derived ozone column data in the PROFILES file.

6.9.2.1 JPROC input files

Table 7□19. JPROC input files

File Name	Format	Description
ET	ASCII	Extraterrestrial radiation as a function of wavelength
PROFILES	ASCII	Seasonal vertical profiles of ozone concentrations, aerosol attenuation, temperature, air density and Dobson values
TOMS	ASCII	Total ozone column measurements from the Total Ozone Mapping Spectrometer instrument aboard the sun-synchronous polar orbiting Nimbus satellite
O2ABS	ASCII	Absorption CSQY data for molecular oxygen as a function of wavelength
O3ABS	ASCII	Absorption CSQY data for ozone as a function of wavelength
CSQY	ASCII (directory path)	Directory path containing absorption CSQY data for gas-phase photolysis reactions as a function of wavelength

6.9.2.2 JPROC output files

Table 7□20. JPROC output files

File Name	Format	Description
`JTABLE_$Date`	`ASCII`	Daily clear-sky photolysis rates file

The default location of the JPROC output files is the `$CMAQ_HOME/data/jproc` directory, controlled by the `OUTDIR` variable in the run script. The default naming convention for all JPROC output files uses the Date environment variable in the file name, which is aliased to the `STDATE` environment variable in the run script.

6.9.2.3 Compilation Configuration Variables

The configuration options listed here are set during compilation of the JPROC executable. When these options are invoked they create a binary executable that is fixed to the specified configuration. To change these options it is necessary to recompile JPROC and create a new executable.

- `CopySrc`
  Uncomment to copy the source code into a working build (BLD) directory. If commented, onlhy the compiled object and executable files will be placed in the BLD directory.
- `MakefileOnly` Uncomment to build a Makefile to compile the executable. Comment out to create a Makefile and compile.
- `Mechanism: [default: cb6r3_ae6_aq]`
  Specifies the gas-phase, aerosol, and aqueous-phase chemical mechanisms for which to create

initial conditions. The choices for the *Mechanism* variable are the mechanism directory names under the `$CMAQ_HOME/CCTM/src/MECHS` directory. Also see the Mechanism Definitions Table). Examples include:

  - `cb6r3_ae6_aq`: CB6, revision 3 gas-phase mechanism, sixth-generation CMAQ aerosol mechanism with sea salt and speciated PM Other, aqueous/cloud chemistry
  - `cb05e51_ae6_aq`: CB05 gas-phase mechanism with CMAQv5.1 updates, sixth-generation CMAQ aerosol mechanism with sea salt and speciated PM Other, aqueous/cloud chemistry
  - `cb05tucl_ae6_aq`: CB05 gas-phase mechanism with active chlorine chemistry, updated toluene mechanism, sixth-generation CMAQ aerosol mechanism with sea salt and speciated PM Other, aqueous/cloud chemistry
  - `cb05tump_ae6_aq`: CB05 gas-phase mechanism with active chlorine chemistry, updated toluene mechanism, mercury, and air toxics, sixth-generation CMAQ aerosol mechanism with sea salt and speciated PM, aqueous/cloud chemistry; this is the CMAQv5 multipollutant mechanism
  - `saprc07tb_ae6_aq`: SAPRC-07 gas-phase mechanism with toluene updates and sixth-generation CMAQ aerosol mechanism
  - `racm2_ae6_aq`: RACM2 gas-phase mechanism with toluene updates and sixth-generation CMAQ aerosol mechanism

- `Tracer [default trac0]`
  Specifies tracer species. Invoking inert tracer species in CMAQ requires defining the tracers using namelist files and compiling the CMAQ programs with these files. The setting for this module corresponds to the directory name in the `$CMAQ_HOME/CCTM/src/MECHS` directory that contains the namelist files for the tracer configuration. The default setting is to not use any tracers.

  - `trac[n]`

6.9.2.4 Execution Configuration variables

The environment variables listed here are invoked during execution of the program and are set in the JPROC run script.

- `APPL [default: None]`
  JPROC executable identifier. Must match APPL Variable setting in the JRPOC build script.
- `CFG [default: None]`
  Configuration identifier for the JPROC simulation.
- `MECH [default: None]`
  CMAQ chemical mechanism. Must match Mechanism variable setting in the JPROC build script.
- `EXEC: [default: JPROC_${APPL}_${EXEC_ID}]`
  Executable to use for the simulation. The variable CFG is set in the JPROC run script. The variable EXEC_ID is set in the config_cmaq.csh configuration file.
- `STDATE`
  Start Julian date (YYYYDDD) for computing clear sky photolysis rates.

- ENDATE
  End Julian date (YYYYDDD) for computing clear sky photolysis rates.

6.9.3 Compiling and Running

6.9.3.1 JPROC compilation

Chapter 5 provides an overview of how to install and compile the CMAQ programs for the tutorial simulation. Follow the steps outlined in Chapter 5 (summarized below) to compile new versions of JPROC:

1. Compile Bldmake, the CMAQ source code and compilation management program. This needs to be done only once—the first time CMAQ is installed.

- Cnfigure the JPROC build script to use the config_cmaq.csh script, which points to the available I/O API and netCDF libraries.
- Configure the JPROC build script for your application by setting the compilation configuration variables described above.
- Invoke the build script to create an executable:

```
cd $CMAQ_HOME/UTIL/jproc/scripts
./bldit_jproc.csh |& tee build_jproc.log
```

6.9.3.2 Run JPROC

Set the run script settings according to the execution configuration variables described above. Run JPROC to produce offline clear-sky photolysis rates for the CCTM:

```
cd $CMAQ_HOME/UTIL/jproc/scripts
./run_jproc.csh |& tee run_jproc.log
```

6.10 MCIP

6.10.1 Description

The Meteorology-Chemistry Interface Processor (MCIP) processes meteorological model output from either MM5 or WRF□ARW model into I/O API-formatted files that are compatible with CMAQ and SMOKE. MCIP automatically determines whether an input file is generated by MM5 or WRF□ARW by trying to open the file as a netCDF file. If the file can be read as netCDF, MCIP assumes the input is a WRF□ARW dataset; otherwise, MM5 is assumed.

Many of the fields that are simulated by the meteorological model are not modified by MCIP for the emissions model and CCTM, and they are written to I/O API files. Fields that are required for the transformation to CMAQ's generalized coordinate system are calculated within MCIP. The dry deposition velocities are no longer calculated by the current version of MCIP. CMAQv5 can now calculate all deposition velocities, MCIPv3.4 will be the last version of MCIP to calculate those velocities internally.

MCIP can extract both temporal and spatial subsets of the input meteorology files. The run script allows the user to specify the beginning and end dates/times of the MCIP simulation; these dates/times can fall anywhere within the range of the input meteorological time period, but must be consistent with the time granularity of the meteorological files. MCIP cannot perform temporal interpolations to artificially increase the temporal resolution of the meteorology fields. Two types of horizontal domain windowing are allowed with MCIP. The boundary trim option ("BTRIM") uniformly trims grid cells off each of the four lateral boundaries of the input meteorology grid. The nonuniform trim option specifies an offset from the lower left corner of the input meteorology domain and the number of cells in the X and Y directions from the revised origin to extract from the input domain. More information about how to invoke these options is provided in the next section: MCIP. MCIP also provides the capability to reconfigure the vertical layer structure in the input meteorology through interpolation from the input structure to an output structure defined through sigma coordinates in the run script. Commonly referred to as "layer collapsing," this option should be exercised with caution as it can significantly impact the conservation of energy assumption inherent in the meteorology through its effects on the predicted wind fields.

6.10.2 Files, configuration, and environment variables

Figure 7□8 shows the input and output files and configuration options for MCIP. All MCIP configurations are accomplished at execution (rather than at compile time) and via Fortran namelist variables, a distinction from the rest of the CMAQ programs. The user does not need to directly edit the MCIP namelist file. All configuration settings are contained in the MCIP run script, which automatically creates a new namelist file each time the script is executed.

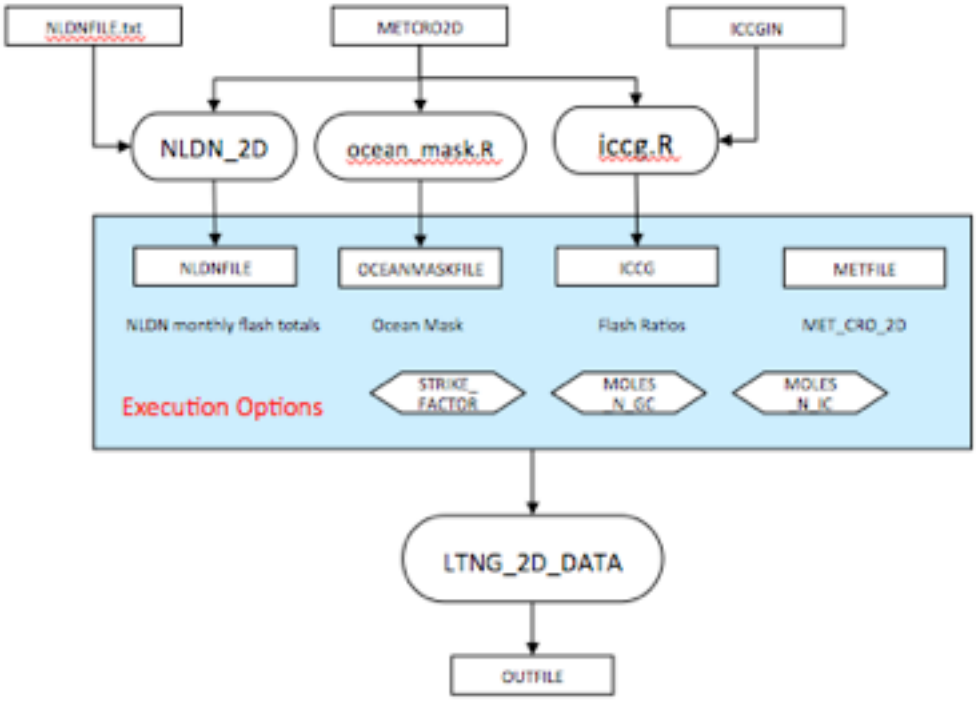


Figure 7□8. MCIP input and output files

6.10.2.1 MCIP Input Files

Table 7□21. MCIP input files

File Name	Format	Description
InMetFiles	binary (MM5) or netCDF (WRF□ARW)	List of MM5 or WRF□ARW output files for input to MCIP
InTerFile	binary	MM5 Terrain file with fractional land use categories; used for calculating land-use-dependent vertical diffusivity. Not necessary with WRF□ARW; this information is included in the WRF-ARW met file.
InSatFiles		GOES satellite cloud data

6.10.2.2 MCIP Output Files

Table 7□22. MCIP output files

File Name	Format	Description
GRIDDESC	ASCII	Grid description file with coordinate and grid definition information
GRID_BDY_2D	BNDARY3	Time-independent 2-D boundary meteorology file
GRID_CRO_2D	GRDDED3	Time-independent 2-D cross-point meteorology file

File Name	Format	Description
GRID_CRO_3D	GRDDED3	Time-independent 3-D cross-point meteorology file
GRID_DOT_2D	GRDDED3	Time-independent 2-D dot-point meteorology file
MET_BDY_3D	BNDARY3	Time-dependent 3-D boundary meteorology file
MET_CRO_2D	GRDDED3	Time-dependent 2-D cross-point meteorology file
MET_CRO_3D	GRDDED3	Time-dependent 3-D cross-point meteorology file
MET_DOT_3D	GRDDED3	Time-dependent 3-D dot-point meteorology file
mmheader	ASCII	Content of MM5 header including configuration information; not generated for WRF□ARW input

The default location of the MCIP output files is the `$CMAQ_HOME/data/mcip/$GridName` directory. Since the default file names do not have any information about the model grid that they are simulating, the name of the grid is set in the output directory path. The default naming convention for all MCIP output files uses only the APPL environment variable in the file name. All of the file-naming variables for the MCIP outputs are set in the run script.

6.10.2.3 Compilation Configuration

All model configuration options for MCIP are set during execution. System compiler options must be set in the provided Linux Makefile to build the program for different operating system/compiler combinations. Example compiler paths, flags, and library locations are provided in the default Makefile.

6.10.2.4 Execution Configuration Variables

The environment variables listed here are invoked during execution of the program and are set in the MCIP run script.

- `APPL [default: None]`
  Application name; scenario ID for file naming
- `CoordName [default: None]`
  Coordinate system name of the MCIP output grid that is written to the GRIDDESC file
- `GridName [default: None]`
  Model grid name of the MCIP output grid that is written to the GRIDDESC file
- `DataPath [default: $CMAQ_DATA]`
  Input/output data directory path
- `InMetDir [default: None]`
  Path of the input data directory containing the MM5 or WRF□ARW output data files
- `InTerDir [default: None]`
  Path of the input data directory containing the MM5 TERRAIN or WRF Geogrid file
- `InSatDir [default: None]`
  Path of the input data directory containing GOES satellite files for using observed cloud cover data.
- `OutDir [default: $CMAQ_HOME/data/mcip]`
  Path of the MCIP output data directory

- `ProgDir [default: $CMAQ_HOME/PREP/mcip/src]`
  Working directory containing the MCIP executable
- `WorkDir [default: $OutDir]`
  Temporary working directory for Fortran links and the namelist file
- `InMetFiles [default: None]`
  List of input meteorology files, including the directory path for each file; without modifying MCIP, up to 300 meteorological model output files are allowed as input to a single MCIP execution
- `InSatFiles [default: None]`
  List of input GOES satellite cloud data files.
- `IfTer [default: F]`
  Binary flag indicating the availability of an input MM5 TERRAIN or WRF Geogrid file; options include T (true) or F (false)
- `InTerFile [default: None]`
  Name and location of input MM5 TERRAIN or WRF Geogrid file
- `LPV: [default: 0]`
  Compute and output potential vorticity. This must be activated to support the CCTM O3 potential vorticity scaling.
  - `0`: Do not compute and output potential vorticity
  - `1`: Compute and output potential vorticity
- `LWOUT [default: 0]`
  Output vertical velocities.
  - `0`: Do not output vertical velocity
  - `1`: Output vertical velocity
- `LUVCOUT [default: 0]`
  Output u- and v-component winds on C-grid.
  - `0`: Do not output u- and v-component winds on C-grid
  - `1`: Output u- and v-component winds on C-grid
- `LSAT [default: 0]`
  Use GOES satellite cloud observations to replace model-derived input on clouds
  - `0`: No satellite data are available
  - `1`: Use GOES observed cloud information to replace model-derived input
- `MCIP_START [format: YYYY-MM-DD-HH:MM:SS.SSSS]`
  Beginning date and time (UTC) of data to output from MCIP. The start date and time must be contained within the input data from MM5 or WRF□ARW.
- `MCIP_END [format: YYYY-MM-DD-HH:MM:SS.SSSS]`
  End date and time (UTC) of data to output from MCIP. The end date and time must be contained within the input data from MM5 or WRF□ARW.
- `INTVL [default: 60]`
  Output interval in minutes. This setting determines the amount of model time contained in each

output time step. The output interval for MCIP can be less frequent than the incoming meteorological model output (e.g., process 30-minute data for CCTM from 15-minute WRF□ARW output).

- `CTMLAYS [default: "-1.0" ]`
  Sigma values of the vertical layers in the 3-D MCIP output. Comma-delimited values for each sigma value must be in descending order starting at 1 and ending with 0. There are a maximum of 100 layers allowed. To use the all of the layers from the input meteorology without collapsing (or explicitly specifying), set CTMLAYS = □1.0.
- `MKGRID [default: T]`
  Determines whether to output static (GRID) meteorology files
- `BTRIM [default: 5]`
  The number of boundary points to remove on each of the four horizontal sides of the MCIP domain. Setting BTRIM = 0 will specify the maximum extent of the input meteorology domain. To remove the MM5 or WRF□ARW lateral boundaries, set BTRIM = 5 (recommended). This setting affects the output MCIP horizontal domain by reducing the input meteorology domain by 2*BTRIM* + *2*NTHIK + 1, where NTHIK is the lateral boundary thickness (from the BDY files). The extra point reflects the conversion from the grid points (dot points) to grid cells (cross points). For windowing a subset domain of the input meteorology, set BTRIM = -1; this setting causes BTRIM to be replaced by the information provided by X0, Y0, NCOLS, and NROWS (see below).
- `X0 [used only if BTRIM = -1]`
  The *x*-coordinate of the lower-left corner of the full MCIP cross-point domain (including the MCIP lateral boundary) based on the input MM5 or WRF□ARW domain. X0 refers to the east-west direction.
- `Y0 [used only if BTRIM = -1]`
  The *y*-coordinate of the lower-left corner of the full MCIP cross-point domain (including the MCIP lateral boundary) based on the input MM5 or WRF□ARW domain. Y0 refers to the north-south direction.
- `NCOLS [used only if BTRIM = -1]`
  Number of columns in the output MCIP domain (excluding MCIP lateral boundaries)
- `NROWS [used only if BTRIM = -1]`
  Number of rows in the output MCIP domain (excluding MCIP lateral boundaries)
- `LPRT_COL [default: 0]`
  Column cell coordinate for diagnostic outputs on the MCIP modeling domain
- `LPRT_ROW [default: 0]`
  Row cell coordinate for diagnostic outputs on the MCIP modeling domain
- `WRF_LC_REF_LAT [default: -999.0]`
  WRF Lambert Conformal reference latitude. Use this setting to force the reference latitude in the output MCIP data. If not set, MCIP will use the average of the two true latitudes. This setting is useful for matching WRF grids to existing MM5 grids.

6.10.3 Compiling and Running

6.10.3.1 Compile MCIP

Chapter 5 provides an overview of how to install and compile the CMAQ programs for the tutorial simulation. Follow the steps outlined in Chapter 5 (summarized below) to compile new versions of **MCIP**.

MCIP is compiled with a Makefile. The configuration options in the Makefile include only the compiler and compiler flags to use for building the executable. The Makefile is located in the directory with the MCIP source code (`$CMAQ_HOME/PREP/mcip/src`). To compile MCIP, source the config_cmaq.csh file and invoke the Makefile at the command line:

```
cd $CMAQ_HOME/PREP/mcip/src/
source $CMAQ_HOME/config_cmaq.csh
./make |& tee make.mcip.log
```

To port MCIP to different compilers, change the compiler names, locations, and flags in the config_cmaq.csh script.

6.10.3.2 Run MCIP

Set the run script settings according to the execution configuration variables described above. Run MCIP to produce meteorology input data for the CCTM:

```
cd $CMAQ_HOME/PREP/mcip/scripts
./run_mcip.csh |& tee run_mcip.log
```

6.11 References for Chapter 7: CMAQ Programs and Libraries

Arya, P., 1984: Parametric relations for the atmospheric boundary layer. Bound.-Layer Meteor., 30, 57–73.

Byun, D. W., and J. K. S. Ching, 1999: Science Algorithms of the EPA Models-3 Community Multiscale Air Quality (CMAQ) Modeling System. U. S. Environmental Protection Agency Rep. EPA□600/R□99/030, 727 pp. [Available from Office of Research and Development, EPA, Washington, DC 20460.]

Hanna, S. R., G. A. Briggs, and R. P. Hosker, 1982: Handbook on atmospheric diffusion, U.S. DOE, DOE/TIC-11223, DE82002045, National Technical Info. Center, Springfield, VA.

Hicks, B.B., 1985: Behavior of turbulence statistics in the convective boundary layer. J. Clim. Appl. Meteor., 24, 607–614.

Irwin, J. S., 1979: Scheme for estimating dispersion parameters as a function of release height, EPA-600/4-79-062, Research Triangle Park, NC.

Niewstadt, F. T. M., 1984: Some aspects of the turbulent stable boundary layer. Bound.-Layer Meteor., 30, 31–55.

Pleim, J. E., A. Xiu, P. L. Finkelstein, and T. L. Otte, 2001: A coupled land-surface and dry deposition model and comparison to field measurements of surface heat, moisture, and ozone fluxes. Water Air Soil Pollut. Focus, 1, 243–252.

Venkatram, A., 1988: Dispersion in the stable boundary layer. Chapter 5, in Lectures on Air Pollution Modeling, A. Venkatram and J. Wyngaard, Eds., American Meteorology Society, Boston, MA.

Weil. J. C., 1988: Dispersion in the convective boundary layer. Chapter 4, in Lectures on Air Pollution Modeling, A. Venkatram and J. Wyngaard, Eds., American Meteorology Society, Boston, MA.

Wesely, M. L., 1989: Parameterization of surface resistances to gaseous dry deposition in regional-scale numerical models. *Atmos. Environ.,* **23**, 1293–1304.

7 CMAQ Input and Output Files

Jump to Input Files Jump to CCTM Output Files

The input files for CMAQ consist of a domain definition file for all programs; two sets of file options for both ICON and BCON; two types of input files (WRF/MM5 and terrain) for MCIP; five mandatory and one optional input file for JPROC; and for CCTM, emissions, initial conditions, and boundary conditions files, six files that define the meteorological conditions to be simulated, and a photolysis rates file. For most CCTM input, a separate data set is required for each horizontal domain that is modeled. When CMAQ is configured for in-line emissions and deposition, there are additional emissions input files that are required.

CMAQ output files include a basic set of files with aerosol and gas-phase species concentrations, wet and dry deposition estimates, and visibility metrics, and an auxiliary set of output files for diagnosing model performance and in-line-calculated emissions.

CMAQ Input Files

This section describes each of the input files required by the various CMAQ programs. The section begins with a description of the grid definition file, which is used by several CMAQ programs, and then goes through a program-by-program listing of the CMAQ input file requirements. Table 8-1 lists the source, file type, and temporal and spatial dimensions of each CMAQ input file. Sample disk space requirements for a desired input data set can easily be calculated from the information in Table 8-1; each data record is four bytes. The I/O API file sizes can be calculated using the number of variables in a CMAQ file and the spatial and temporal coverage of the data. The user should consult the CMAQ release notes for additional file information.

Table 8-1. CMAQ input files

File Name	File Type	Time-Dependence	Spatial Dimensions	Source
General				
GRIDDESC	ASCII	n/a	n/a	user/MCIP
gc_matrix.nml	ASCII	n/a	n/a	CSV2NML
ae_matrix.nml	ASCII	n/a	n/a	CSV2NML

File Name	File Type	Time-Dependence	Spatial Dimensions	Source
nr_matrix.nml	ASCII	n/a	n/a	CSV2NML
tr_matrix.nml	ASCII	n/a	n/a	CSV2NML
ICON				
IC_PROFILE	ASCII	Annual	n/a	user
CTM_CONC_1	GRDDED3	Hourly	X*Y*Z	CCTM
MET_CRO_3D	GRDDED3	Hourly	X*Y*Z	MCIP
BCON				
BC_PROFILE	ASCII	Annual	n/a	user
CTM_CONC_1	GRDDED3	Hourly	X*Y*Z	CCTM
MET_CRO_3D	GRDDED3	Hourly	X*Y*Z	MCIP
JPROC				
ET	ASCII	Annual	n/a	user
PROFILES	ASCII	Annual	n/a	user
O2ABS	ASCII	Annual	n/a	user
O3ABS	ASCII	Annual	n/a	user
TOMS	ASCII	varies	n/a	user
CSQY	ASCII	Annual	n/a	User
MCIP				
InMetFiles	Binary or netCDF	typically hourly, but sometimes sub-hourly	X*Y*Z	MM5 or WRF-ARW
InTerFile InSatFiles	Binary	n/a	X*Y	MM5 or WRF Spatial Allocator
CCTM				
INIT_CONC_1	GRDDED3	Time-invariant	X*Y*Z	ICON/CCTM
BNDY_CONC_1	BNDARY3	Hourly	[2(X+1)+2(Y+1)]*Z	BCON
JTABLE	ASCII	Daily	n/a	JPROC
OMI	ASCII	daily	n/a	
EMIS_1	GRDDED3	Hourly	X*Y*Z	SMOKE
OCEAN_1	GRDDED3	Time-invariant	X*Y	Spatial Allocator
GSPRO	ASCII	Time-invariant	N/a	User
B3GRD	GRDDED3	Time-invariant	X*Y	SMOKE
BIOSEASON	GRDDED3	Time-invariant	X*Y	Metscan
STK_GRPS_nn	GRDDED3	Time-invariant	X*Y	SMOKE
STK_EMIS_nn	GRDDED3	Hourly	X*Y	SMOKE

File Name	File Type	Time-Dependence	Spatial Dimensions	Source
DUST_LU_1	GRDDED3	Time-invariant	X*Y	Spatial Allocator
DUST_LU_2	GRDDED3	Time-invariant	X*Y	Spatial Allocator
MODIS_FPAR	GRDDED3	Daily	X*Y	Spatial Allocator
CROPMAP01	GRDDED3	Time-invariant	X*Y	Cropcal
CROPMAP04	GRDDED3	Time-invariant	X*Y	Cropcal
CROPMAP08	GRDDED3	Time-invariant	X*Y	Cropcal
LTNGNO	GRDDED3	Hourly	X*Y*Z	User
NLDN_STRIKES	GRDDED3	Hourly	X*Y	
LTNGPARMS_FILE	GRDDED3	Time-invariant	X*Y	
BELD4_LU	GRDDED3	Time-invariant	X*Y	
E2C_SOIL	GRDDED3	Time-invariant	X*Y	
E2C_FERT	GRDDED3	Daily	X*Y	
MEDIA_CONC	GRDDED3	Hourly	X*Y	
INIT_GASC_1	GRDDED3		X*Y	
INIT_AERO_1	GRDDED3		X*Y	
INIT_NONR_1	GRDDED3		X*Y	
INIT_TRAC_1	GRDDED3		X*Y	
GRID_CRO_2D	GRDDED3	Time-invariant	X*Y	MCIP
GRID_CRO_3D	GRDDED3	Time-invariant	X*Y*Z	MCIP
GRID_BDY_2D	GRDDED3	Time-invariant	PERIM*Z	MCIP
GRID_DOT_2D	GRDDED3	Time-invariant	(X+1)*(Y+1)	MCIP
MET_BDY_3D	BNDARY3	Hourly	PERIM*Z	MCIP
MET_CRO_2D	GRDDED3	Hourly	X*Y	MCIP
MET_CRO_3D	GRDDED3	Hourly	X*Y*Z	MCIP
MET_DOT_3D	GRDDED3	Hourly	(X+1)*(Y+1)*Z	MCIP

GRIDDESC: Horizontal domain definition

Used by: ICON, BCON, CCTM

The CMAQ grid description file (**GRIDDESC**) is used by all programs except JPROC and MCIP

to define the horizontal spatial grid of the modeling domain. GRIDDESC implements I/O API grid conventions: for more details see the section on Grids and coordinate systems.

A GRIDDESC is an ASCII file that contains two sections: a horizontal coordinate section, and grid description section. GRIDDESC is the logical name for text files that store horizontal coordinate and grid descriptions, and that are read by the DSCGRID() and DSCOORD() utility routines. Each segment has a one-line header (which by convention provides titles for the columns in the data records), a sequence of data records, and a terminal record with name field blank (i.e., ‘’). The GRIDDESC file is generated automatically with MCIP; alternatively, GRIDDESC can be created using a text editor.

The horizontal coordinate section (Table 8-2) consists of text records that provide coordinate-system name, the map projection, and descriptive parameters P_ALP, P_BET, P_GAM, XCENT, and YCENT.

The grid description section (Table 8-3) consists of text records that indicate the grid name, related coordinate-system name (i.e., which GRIDDESC horizontal coordinate name that is defined in the previous section that is applied to this grid), and descriptive parameters XORIG, YORIG, XCELL, YCELL, NCOLS, NROWS, and NTHIK. For a typical CMAQ application, both “dot-point” and “cross-point” grids are defined in the GRIDDESC file; these grids are topological duals in the sense that the vertices (corners) of one correspond to the cell-centers of the other.

Table 8-2. Coordinate sytem description segment of GRIDDESC

Line	Column	Name	Type	Description
1	A	Header	String	Single-quote-delimited header describing section contents; may be blank, i.e., ‘’
2	A	COORD-NAME	String	Name of the coordinate description (required); single quote delimited
3	A	COORDTYPE	Int	I/O API index defining the map projection type (required)
3	B	P_ALP	Double	First map projection descriptive parameter (dependent on projection type)
3	C	P_BET	Double	Second map projection descriptive parameter (dependent on projection type)
3	D	P_GAM	Double	Third map projection descriptive parameter (dependent on projection type)
3	E	XCENT	Double	Longitude for coordinate system center
3	F	YCENT	Double	Latitude for coordinate system center

Table 8-3. Grid definition segment of GRIDDESC

Line	Column	Name	Type	Description
1	A	Header	String	Single-quote-delimited header describing section contents; may be blank, i.e., ‘’
2	A	GRID-NAME	String	Name of the horizontal grid (required); single quote delimited

Line	Column	Name	Type	Description
3	A	COORD-NAME	String	Name of the coordinate description in the previous section (required); single quote delimited
3	B	XORIG	Double	X-coordinate for lower-left (southwest) corner of the grid with respect to (XCENT,YCENT) (dependent on projection type)
3	C	YORIG	Double	Y-coordinate for lower-left (southwest) corner of the grid with respect to (XCENT,YCENT) (dependent on projection type)
3	D	XCELL	Double	X-coordinate grid cell size (dependent on projection type)
3	E	YCELL	Double	Y-coordinate grid cell size (dependent on projection type)
3	F	NCOLS	Int	Number of horizontal grid columns (dependent on projection type)
3	G	NROWS	Int	Number of horizontal grid rows (dependent on projection type)
3	H	NTHIK	Int	Boundary perimeter thickness (number of cells) (optional)

Each data record in these files consists of two or three list-formatted lines (i.e., items are separated by either blanks or commas). Name fields are quoted strings, and appear on the first of these lines. Numeric fields are given in double precision, and occur on either the second line or the second and third lines (this allows you to organize the text so that it is easily viewed in a text editor without running off-screen). The records have the following organization, depending upon whether they are in the first or second segment of GRIDDESC:

```
COORD-NAME COORDTYPE, P_ALP, P_BET, P_GAM XCENT, YCENT
```

or

```
COORD-NAME COORDTYPE, P_ALP, P_BET, P_GAM, XCENT, YCENT
```

and

```
GRID-NAME COORD-NAME, XORIG, YORIG, XCELL, YCELL NCOLS, NROWS, NTHIK
```

or

```
GRID-NAME COORD-NAME, XORIG, YORIG, XCELL, YCELL, NCOLS, NROWS, NTHIK
```

There are at most 32 coordinate systems and 256 grids listed in one of these files. These files are small enough to be archived easily with a study, and have a sufficiently simple format that new ones can easily be constructed “by hand.”

An example of a GRIDDESC file is shown below:

```
' ' 'LAM_40N100W' 2 30.0 60.0 -100.0 -100.0 40.0 ' ' 'M_32_99TUT02' 'LAM_40N100W'
544000.0 -992000.0 32000.0 32000.0 38 38 1 ' '
```

The horizontal coordinate section (first section) in this example GRIDDESC file defines a horizontal coordinate named "LAM_40N100W". The coordinate definition is for a Lambert conformal grid, keyed by the first column of the coordinate description line, which corresponds to the numeric code for the various I/O API-supported grid types (2 = Lambert). The next three parameters (P_ALP, P_BET, and P_GAM) have different definitions for different map projections. For Lambert conformal, P_ALP and P_BET are the true latitudes of the projection cone (30°N and 60°N in the example), and P_GAM (100°W in the example) is the central meridian of the projection. The last two parameters, XCENT and YCENT, are the latitude-longitude coordinates of the grid center of the Cartesian coordinate system, which are 100°W and 40°N in the example. If using WRF-ARW as the meteorological model, the user should be aware of differences from this method.

The example grid definition section above describes a grid named "M_32_99TUT02". The definition of the grid begins with a reference to a coordinate name from the coordinate definition section of the file; in this example, the coordinate named "LAM_40N100W" is referenced in the grid definition. The next two parameters in the grid definition (XORIG and YORIG) are the east-west and north-south offsets from XCENT and YCENT in meters (WRF-ARW usages may differ). The next two parameters (XCELL and YCELL) are the horizontal grid spacing in meters for the X and Y directions (i.e., delta-x and delta-y). The next two parameters (NCOLS and NROWS) are the numbers of grid cells in the X and Y directions. The grid definition concludes with the number of boundary cells, NTHIK, which is typically set to 1.

{gc|ae|nr|tr}_matrix.nml: Species namelist files

Used by: BCON, CCTM, ICON, CHEMMECH

Namelist look-up tables for different classes of simulated pollutants are used to define the parameters of different model species during the execution of the CMAQ programs. Gas-phase (gc), aerosol (ae), non-reactive (nr), and tracer (tr) species namelist files contain parameters for the model species that are included in these different classifications. The species namelist files are used to control how the different CMAQ programs and processes handle the model species. The namelist files define the following processes for each model species:

- Emissions surrogate – which (if any) emitted species is the pollutant mapped to; For GC, NR, and TR species, the variable names in the emission files determine the allowed surrogate values; For AE species, the model source code determines the allowed values.
- Emissions factor – if the pollutant is mapped to an emitted species, uniformly apply a scaling factor to the emissions
- Deposition velocity – which (if any) deposition velocity is the deposition velocity for the pollutant mapped to; Allowed velocities are specified within the model source code.
- Deposition velocity factor – if the pollutant is mapped to a deposition velocity, uniformly apply a scaling factor to this velocity
- Initial/boundary conditions – which initial and boundary condition species is the pollutant mapped to; if not specified, this will default to the species name
- IC/BC Factor – if the pollutant is mapped to an initial/boundary condition species, uniformly apply a scaling factor to the concentrations
- Scavenging - which (if any) species is the pollutant mapped to; Allowed scavenging surrogates are specified within the model source code.

- Scavenging factor - if the pollutant is mapped to an species for scavenging, uniformly apply a scaling factor to the scavenging rate
- Gas-to-aerosol conversion – which (if any) aerosol chemistry species does the gas phase pollutant concentration go into for transformation from the gas-phase to the aerosol-phase
- Gas-to-aqueous Surrogate – which (if any) cloud chemistry species does the gas pollutant concentration go into for simulating chemistry within cloud water
- Aerosol-to-aqueous Surrogate – which (if any) cloud chemistry species does the aerosol pollutant concentration go into for simulating chemistry within cloud water
- Transport – is the pollutant transported by advection and diffusion in the model?
- Dry deposition – Write the pollutant to the dry deposition output file?
- Wet deposition – Write the pollutant to the wet deposition output file?
- Concentration – Write the pollutant to the instantaneous concentration output file?

The namelist files contain header information that describe which class of species are contained in the file, the number of parameters contained in the file, headers describing the parameter fields, and then a series of rows with configuration parameters for every model species. Table 8-4 contains the namelist file format for the gas-phase (GC) species namelist file. The namelist files for the other species classifications (AE, NR, TR) are similar to the format shown in Table 8-4.

Table 8-4. GC species namelist file format

Line	Column	Name	Type	Description
1		!Revision Control System(RCS file)		
2		Header: filename, version, date/time, author		
4		File type	String	&GC_nml
6		Number of group 1 surrogate params	String	n_surr1 = x, where x is the number of surrogates that are specified in pairs of surrogate species and surrogate factor, i.e. emissions, deposition, initial/boundary conditions, and scavenging
7		Number of group2 surrogate params	String	n_surr2 = x, where x is the number of surrogates that are specified only as a surrogate species, i.e. gas-to-aerosol conversion, gas-to-aqueous conversion, and aerosol-to aqueous conversion

Line	Column	Name	Type	Description
8		Number of control params	String	n_ctrl = x, where x is the number of Y/N parameters controlling whether this pollutant is transported and whether it is written to the dry deposition, wet deposition, and/or concentration output files
10		Header ID	String	TYPE_HEADER =
11		HEADER	String	Abbreviated names of file columns, enclosed by single quotes
12		Matrix ID	String	TYPE_MATRIX =
13	1	SPC	String	CMAQ pollutant name, i.e. NO, HNO3, PAR; dependent on chemical mechanism
	2	MOLWT	Integer	Pollutant molecular weight
	3	EMIS_SUR	String	Emissions species name for the CMAQ pollutant
	4	EMIS_FAC	Real	Scaling factor for input emissions
	5	DEPV_SUR	String	Deposition velocity variable name for the CMAQ pollutant
	6	DEPV_FAC	Real	Scaling factor for the deposition velocity
	7	ICBC_SUR	String	IC/BC species name for the CMAQ pollutant
	8	ICBC_FAC	Real	Scaling factor for the IC/BC concentration
	9	SCAV_SUR	String	Wet Deposition Scavenging Coefficient
	10	SCAV_FAC	Real	Scaling factor for Scavenging
	11	G2AE_SUR	String	Gas-to-aerosol transformation species
	12	G2AQ_SUR	String	Gas-to-aqueous transformation species
	13	TRNS	Yes/No	Transport switch. Note: Instead of using one column labeled “TRNS” to turn on/off both advection and diffusion for a pollutant, two separate columns labeled “ADV” and “DIFF” can be used to switch on/off advection and diffusion separately
	14	DDEP	Yes/No	Dry deposition output file switch
	15	WDEP	Yes/No	Wet deposition output file switch
	16	CONC	Yes/No	Concentration output file switch
…	…	…	…	Repeat for the number of gas-phase pollutants in the mechanism being modeling

The namelist files for the other pollutant classes have similar configurations as the gas-phase species configuration shown in Table 8-4. For an example see this link to the GC namelist species file for the cb05e51_ae6_aq mechanism.

IC_PROFILE: Initial conditions vertical profiles

Used by: ICON

ICON can generate initial conditions from two different input file types. The first file type is an ASCII vertical profile file that lists species concentrations at various model layers that are fixed in space and time. To configure ICON to generate initial conditions from ASCII vertical profiles, the "prof" input module is chosen when compiling the program (see Chapter 8). These ASCII-formatted vertical profile files are IC_PROFILE files, and are described in this section. IC_PROFILE files must be developed by the user and can be generated from climatologically averaged observational data or as an a priori estimate from previous modeling studies of the region being modeled. The second file type that ICON can use to generate initial conditions is a concentration file from a previous CMAQ run. These are CTM_CONC_1 files, and are described later in the CTM_CONC_1 section.

IC_PROFILE begins with a header that contains a comment section that describes the data, and a file description section that defines the number of vertical levels in the file, the number of pollutants in the file, and the distribution of the vertical levels. The next entries in IC_PROFILE are the Julian start date and the start time of the data; they are not used by ICON.

Each line in IC_PROFILE corresponds to a different pollutant and begins with the name of the pollutant. The subsequent columns on each line list the chemical concentration at each layer contained in the file. Gas-phase species are in ppmV, and aerosol species are in µg m-3. The layer structure of the IC_PROFILE vertical profiles does not need to correspond exactly to the layer structure that will be modeled; the ICON program will interpolate the data to the correct vertical format for the simulation.

Initial conditions are provided for only the first hour of a model simulation. The initial conditions that are based on an ASCII vertical profile include a gridded file for input to CCTM that has horizontally uniform species concentrations at each model layer. For spatially resolved initial conditions in the horizontal direction, it is necessary to use the other input file type to ICON, an existing CCTM concentration file (CTM_CONC_1).

A detailed description of the vertical profile file format for initial conditions is provided in Table 8-5. The header of the profiles file is list-formatted, while the data section of the file is fixed format.

Table 8-5. IC_PROFILE format description

Line	Column	Name	Type	Description
1-3		Text Header	String	Text description of the contents and source of the initial conditions file (optional)
4	A	NUM_SIGMA_LVL	Int	Number of sigma levels contained in the file (required)
	B	NUM_POLL	Int	Number of pollutants contained in the file (required)
	C	SIGMA_LVL	Real	Vertical coordinate values of sigma-p levels; number of values (n+1) is one more than the NUM_SIGMA_LVL (n) (required)
4	n	…	…	…
5	A	STDATE	String	Julian start date of the file, YYYYDDD (optional)
	B	STTIME	String	Start time of the file, HH (optional)

Line	Column	Name	Type	Description
6	1-10	SPECIES1	String	Pollutant name, enclosed in double quotes (required)
	12-20	LAYER1_IC	Exp	IC concentration for species 1 in lowest sigma layer (required)
	23-31	LAYER2_IC	Exp	IC concentration for species 1 in 2nd sigma layer (required)
	34-42	LAYER3_IC	Exp	IC concentration for species 1 in 3rd sigma layer (required)
	45-53	LAYER4_IC	Exp	IC concentration for species 1 in 4th sigma layer (required)
	…	LAYERX_IC	Exp	IC concentration for species 1 in Xth sigma layer (required)
7	1-10	SPECIES2	String	Pollutant name, enclosed in double quotes (required)
	12-20	LAYER1_IC	Exp	IC concentration for species 2 in lowest sigma layer (required)
	23-31	LAYER2_IC	Exp	IC concentration for species 2 in 2nd sigma layer (required)
	34-42	LAYER3_IC	Exp	IC concentration for species 2 in 3rd sigma layer (required)
	45-53	LAYER4_IC	Exp	IC concentration for species 2 in 4th sigma layer (required)
	…	LAYERX_IC	Exp	IC concentration for species 2 in Xth sigma layer (required)
…	…	…	…	…
Z	1-10	SPECIESZ	String	Pollutant name, enclosed in double quotes (required)
…	12-20	LAYER1_IC	Exp	IC concentration for species Z in lowest sigma layer (required)
…	23-31	LAYER2_IC	Exp	IC concentration for species Z in 2nd sigma layer (required)
…	34-42	LAYER3_IC	Exp	IC concentration for species Z in 3rd sigma layer (required)
…	45-53	LAYER4_IC	Exp	IC concentration for species Z in 4th sigma layer (required)
…	…	LAYERX_IC	Exp	IC concentration for species Z in Xth sigma layer (required)

A sample of the four sections of an IC_PROFILE file is shown below.

```
Example initial condition: The vertical coordinate of the model to generate
these i.c. is the terrain-following sigma coordinate.
The number of sigma layers and defined sigma levels are listed below.
```

```
 6 55 1.00 0.98 0.93 0.84 0.60 0.30 0.00
1988180 00
"SO2 " 0.300E-03 0.200E-03 0.100E-03 0.100E-03 0.200E-04 0.100E-04
```

BC_PROFILE: Boundary conditions vertical profiles

Used by: BCON

As with the ICON program, BCON can generate boundary conditions from two different input file types. The first file type is an ASCII vertical profile file that list species concentrations at various model layers that are fixed in space in time. To configure BCON to generate boundary conditions from ASCII vertical profiles, the "prof" input module is chosen when compiling the program (see Section 5.2 on BCON). These ASCII-formatted vertical profile files are BC_PROFILE files, and are described in this section. BC_PROFILE files must be developed by the user and can be generated from climatologically averaged observational data or as an a priori estimate from previous modeling studies of the region being modeled. The second file type that BCON can use to generate initial conditions is a concentration file from a previous CMAQ run. These are CTM_CONC_1 files, and are described later in the CTM_CONC_1 section.

BC_PROFILE begins with a header that contains a comment section that describes the data, and a file description section that defines the number of vertical levels in the file, the number of pollutants in the file, and the distribution of the vertical levels. The next entries in BC_PROFILE are the Julian start date and the start time of the data; they are not used by BCON. The BCON input consists of four data sections that correspond to the lateral boundaries (i.e., north, south, east, and west) of the model grid. The BCON input profiles contain a field that precedes each data section to indicate which horizontal boundary the data section describes.

The format of the data sections in BC_PROFILE files is the same as in IC_PROFILE files. Each line corresponds to a different pollutant and begins with the name of the pollutant. The subsequent columns on each line list the chemical concentration at each layer contained in the file. Gas-phase species are in ppmV, and aerosol species are in ?g m-3. The layer structure of the BC_PROFILE vertical profiles does not need to correspond exactly to the layer structure that will be modeled; the BCON program will interpolate the data to the correct vertical format for the simulation.

Boundary conditions can either be time-independent (static) or time-dependent (dynamic). Boundary conditions generated with BC_PROFILE's ASCII vertical profiles are both static and spatially uniform along each of the four horizontal boundaries at each model layer. For spatially resolved (in the horizontal direction) and temporally resolved boundary conditions, it is necessary to use the other input file type to BCON, an existing CCTM concentration file (CTM_CONC_1).

A detailed description of the vertical profile file format for boundary conditions is provided in Table 8-6. The header of the profiles file is list-formatted, while the data section of the file is fixed format.

Table 8-6. BC_PROFILE format description

Line	Column	Name	Type	Description
1-3		Text Header	String	Text description of the contents and source of the initial conditions file (optional)

Line	Column	Name	Type	Description
4	A	NUM_SIGMA_LVL	Int	Number of sigma levels contained in the file (required)
	B	NUM_POLL	Int	Number of pollutants contained in the file (required)
	C	SIGMA_LVL	Real	Vertical coordinate values of sigma-p levels; number of values (n+1) is one more than the NUM_SIGMA_LVL (n) (required)
4	n	…	…	…
5	A	STDATE	String	Julian start date of the file, YYYYDDD (optional)
	B	STTIME	String	Start time of the file, HH (optional)
6	A	Direction	String	North, South, East, West; indicates the boundary described by the subsequent data section (required)
7	1-10	SPECIES1	String	Pollutant name, enclosed in double quotes (required)
	12-20	LAYER1_BC	Exp	BC concentration for species 1 in lowest sigma layer (required)
	23-31	LAYER2_BC	Exp	BC concentration for species 1 in 2nd sigma layer (required)
	34-42	LAYER3_BC	Exp	BC concentration for species 1 in 3rd sigma layer (required)
	45-53	LAYER4_BC	Exp	BC concentration for species 1 in 4th sigma layer (required)
	…	LAYERX_BC	Exp	BC concentration for species 1 in Xth sigma layer (required)
8	1-10	SPECIES2	String	Pollutant name, enclosed in double quotes (required)
	12-20	LAYER1_BC	Exp	BC concentration for species 2 in lowest sigma layer (required)
	23-31	LAYER2_BC	Exp	BC concentration for species 2 in 2nd sigma layer (required)
	34-42	LAYER3_BC	Exp	BC concentration for species 2 in 3rd sigma layer (required)
	45-53	LAYER4_BC	Exp	BC concentration for species 2 in 4th sigma layer (required)
	…	LAYERX_BC	Exp	BC concentration for species 2 in Xth sigma layer (required)
…	…	…	…	…
Y	A	Direction	String	North, South, East, West; indicates the horizontal boundary described by the subsequent data section (required)

Line	Column	Name	Type	Description
Z+1	1-10	SPECIESZ	String	Pollutant name, enclosed in double quotes (required)
…	12-20	LAYER1_BC	Exp	BC concentration for species Z in lowest sigma layer (required)
…	23-31	LAYER2_BC	Exp	BC concentration for species Z in 2nd sigma layer (required)
…	34-42	LAYER3_BC	Exp	BC concentration for species Z in 3rd sigma layer (required)
…	45-53	LAYER4_BC	Exp	BC concentration for species Z in 4th sigma layer (required)
…	…	LAYERX_BC	Exp	BC concentration for species Z in Xth sigma layer (required)

A sample of the important sections of a BC_PROFILE file is shown below.

```
6 55 1.00 0.98 0.93 0.84 0.60 0.30 0.00
1988180 00
 North
 "SO2 " 0.300E-03 0.200E-03 0.100E-03 0.100E-03 0.200E-04 0.100E-04
 West
 "SO2 " 0.300E-03 0.200E-03 0.100E-03 0.100E-03 0.200E-04 0.100E-04
```

CTM_CONC_1: CCTM concentration files

Used by: ICON, BCON

An I/O API GRDDED3-formatted CCTM output concentration file, CTM_CONC_1, can be used to create spatially and temporally varying initial and boundary conditions. To configure ICON and BCON to generate initial and boundary conditions from a CCTM concentration file, the "m3conc" input module is chosen when compiling the programs (see Section 5.5 on ICON and Section 5.2 on BCON). The input concentration file must cover a temporal and spatial extent that is consistent with the time period and domain that are being modeled, respectively. Both ICON and BCON require a Julian start date to be specified at execution that identifies the first time step to extract from the input concentration file; BCON also requires a run-length specification to indicate the number of time steps of boundary conditions to extract from the input file. For nested runs, the nested domain for which initial and boundary conditions are being extracted must be on the same projection and fall within the domain contained in the input concentration file.

CSQY: Absorption cross section and quantum yields

Used by: JPROC

CSQY is the logical name for the ASCII data file containing absorption cross section and quantum yield data for unique photolysis reactions. The data in these files are listed as a function of wavelength and correspond to the standard photolysis reactions in each of the chemical mechanisms implemented in

CMAQ. A flexible format allows users to deviate from these standards and test new data with minimal effort.

The ASCII-formatted CSQY files begin with the number and a list of the applicable photolysis reactions. The data section of the CSQY file includes (1) the reaction name/ID; (2) The temperature bands (3) the number of wavelength bins (4) the start, effect and end of the wavelength bin and the photon flux for each inline band. The absorption cross section averaged over UCI Solar Flux(cm), and the quantum yield averaged over UCI Solar Flux as columns, for each wavelength bin, for each temperature, and for each reaction. The CSQY file uses a space-delimited, free-form format for the data section of the file. A detailed description of the CSQY file format is provided in Table 8-7.

Table 8-7. CSQY format description

Line	Column	Name	Type	Description
1	A	CSQY Mechanism Table Name	String	Text name indicating this is the CSQY for the Mechanism Specified. This name is cross-referenced in the chemical mechanism description files (required)
1	A	NPHOTAB	String	Number of Photolysis Reactions
2	A	Comments	String	Preceded by "!", Individual Reaction Rates Listed below
3	A	Name	String	Reaction Name
repeat for all (NPHOTAB) reaction names				
3+NPHOTAB	A	Name	String	Reaction Name
3+NPHOTAB+1	A	NTEMP	Int or Real	Number of Temperature Bands
3+NPHOTAB+2	A	Number	Integer	Index of temperature
	B	TEMP	Real	Temperature in Kelvin
repeat for all NTEMP values				
3+NPHOTAB+NTEMP+1	A	Comments	String	Preceded by "!", column header: I, START_WL_BIN(nm), EFFECT_WL_BIN_(nm), END_WL_BIN_(nm), photon_flux(cm-2*s-1),
3+NPHOTAB+NTEMP+2	A	Number of Inline Photolysis Bands	Integer	

Line	Column	Name	Type	Description
	B	Start Wavelength	Real	Start Wavelength for Bin
	C	Effect Wavelength	Real	Effect Wavelength for Bin
	D	End Wavelength	Real	End Wavelength for Bin
	E	Photon Flux	Real	Photon Flux
3+NPHOTAB+NTEMP+2	A	Comments	String	Preceded by "!", CS = absorption cross sections averaged over UCI Solar Flux
3+NPHOTAB+NTEMP+3	A	Comments	String	Preceded by "!", QY = quantum yields averaged over UCI Solar Flux
3+NPHOTAB+NTEMP+4	A	Comments	String	Preceded by "!", EQY = eCS*eQY/CS averaged over Solar Flux and 77 bins in UCI Model
3+NPHOTAB+NTEMP+5	A	Comments	String	Preceded by "!", header for reactions, CS or EQY, and Wavelength Bins
3+NPHOTAB+NTEMP+6	A	Reaction Name	String	Reaction Name
	B	Quantity	String	CS
	C	TEMP	Real	Temperature
	D	WBIN (1)	Real or EXP	Absorption Cross Section (CS) Value for BIN 1
	E	WBIN (2)	Real or EXP	Absorption Cross Section (CS) Value for BIN 2
	F	WBIN (3)	Real or EXP	Absorption Cross Section (CS) Value for BIN 3
	G	WBIN (4)	Real or EXP	Absorption Cross Section (CS) Value for BIN 4
	H	WBIN (5)	Real or EXP	Absorption Cross Section (CS) Value for BIN 5
	I	WBIN (6)	Real or EXP	Absorption Cross Section (CS) Value for BIN 6
	J	WBIN (7)	Real or EXP	Absorption Cross Section (CS) Value for BIN 7
3+NPHOTAB+NTEMP+7	A	Reaction Name	String	Reaction Name
	B	Quantity	String	EQY
	C	TEMP	Real	Temperature
	D	WBIN (1)	Real or EXP	Absorption Cross Section (EQY) Value for BIN 1
	E	WBIN (2)	Real or EXP	Absorption Cross Section (EQY) Value for BIN 2
	F	WBIN (3)	Real or EXP	Absorption Cross Section (EQY) Value for BIN 3
	G	WBIN (4)	Real or EXP	Absorption Cross Section (EQY) Value for BIN 4
	H	WBIN (5)	Real or EXP	Absorption Cross Section (EQY) Value for BIN 5

Line	Column	Name	Type	Description
	I	WBIN (6)	Real or EXP	Absorption Cross Section (EQY) Value for BIN 6
	J	WBIN (7)	Real or EXP	Absorption Cross Section (EQY) Value for BIN 7

7.0.1 ET: Extraterrestrial irradiance

Used by: JPROC

ET is the logical name for the ASCII data file containing extraterrestrial radiation as a function of wavelength. The extraterrestrial irradiance file has a format similar to that of the CSQY file (CSQY Section). The file begins with a header section; comment lines are preceded with a "!". Like the CSQY file, the header contains a field describing the location on the wavelength interval that the data represent, and a multiplier. The data section uses a space-delimited, free-form format and lists the wavelength of the incoming solar radiation (nm) and the irradiance (photons cm-2 s-1) at each wavelength, with each row corresponding to a specific wavelength interval. A detailed description of the file format is provided in Table 8-8.

Table 8-8 ET file format description

Line	Column	Name	Type	Description
1	A	Comments	String	Preceded by "!", comment lines describe the file contents and document the source of the data (optional)
n	…	…	…	…
n + 1	A	Data Location	String	Field indicating the location of the data as measured across the wavelength band; possible answers: beginning, ending, centered, point (required)
n + 2	A	Multiplier	String	Multiplication factor to apply to photolysis rate equation; line begins with FAC=; factor listed in real or exponential format (required)
n+3	A	Wavelength	Int or Real	Wavelength corresponding to ET data; units = nm (required)
	B	Extra–terrestrial Irradiance	Real or Exp	Estimation of the photon flux reaching the exterior of the earth's atmosphere; units = photons cm-2 second-1 (required)
n+4	A	Wavelength	Int	Wavelength corresponding to ET data; units = nm (required)
	B	Extra–terrestrial Irradiance	Real or Exp	Estimation of the photon flux reaching the exterior of the earth's atmosphere; units = photons cm-2 second-1 (required)

Line	Column	Name	Type	Description
n+X	…	…	…	…

See this link for an example ET file. A sample of the important sections of an ET file is shown below.

```
! Extraterrestrial Irradiance
! Taken from the RADM data---derived from the WMO 1985 report Table 7-4
! Format: wl, et_irradBeginning
! With FAC, units are (photons cm-2 s-1)
FAC=1.0
185.185 3.620E+11
186.916 4.730E+11
```

PROFILES: Atmospheric vertical profiles

Used by: JPROC

PROFILES is the logical name for the ASCII data file containing seasonal and vertical profiles for ozone, aerosol attenuation, temperature, air pressure, and Dobson values. The ASCII-formatted data provided in the PROFILES file are at 19 latitudes (90°N to 90°S) and 51 altitudes (0 to 50 km) in three distinct data sections. The first data section contains seasonal, latitude-dependent vertical profiles of O3 concentrations (molecules cm-3), air temperature (K), and air density (molecules cm-3). The second data section contains monthly Dobson values at the 19 latitude bands. The last data section contains vertical profiles from the 1976 U.S. Standard Atmosphere of air temperature (K), air density (molecules cm-3), ozone concentrations (molecules cm-3), and aerosol attenuation coefficients (km-1).

The first data section of the PROFILES file is divided into 228 (19x3x4) data blocks, with each block representing one of the three variables (O3, air temperature, and air density) at one of the 19 latitude bands, for each of the 4 seasons of the year. The individual data blocks contain 51 values per variable, representing standard conditions at altitudes ranging from 0 to 50 km. The data are ordered, from general to specific, by season (spring, summer, autumn, winter), variable (O3, air temperature, air density), latitude, and altitude. For example, the first block in the PROFILES file contains spring O3 concentration profiles at the latitude band ranging from 90°N to 80°N from 0 to 50 km above sea level; the first value in the block is at 0 km and the last value is at 50 km. The next data block is again spring O3 concentration profiles but at the latitude band ranging from 80°N to 70°N. The next 17 data blocks complete the spring O3 concentration profiles by continuing through the rest of the 19 latitude bands, with the last block representing the 80°S to 90°S latitude band. The 20th data block begins the spring air temperature profiles at the latitude band ranging from 90°N to 80°N and is followed by 18 more data blocks of spring air temperature profiles. The following 19 data blocks follow an identical format for air density and round out the spring profiles. The 19x3 data blocks are then repeated for summer profiles, autumn profiles, and finally winter profiles.

The second data section in the PROFILES file contains monthly average Dobson values. The data are organized in 12 rows (January through December) and 19 columns (90°N to 80°N through 80°S to 90°S).

The last data section in the PROFILES file contains vertical profiles from the 1976 U.S. Standard Atmosphere of temperature, air density, ozone concentrations, and aerosol attenuation coefficients. The data are organized in 51 rows, corresponding to altitude (0 to 50 km), with four columns per row for each of the four variables. See this link for an example PROFILES file. A detailed description of the file format is provided in Table 8-9.

Table 8-9. PROFILES file format description.

Line	Column	Name	Type	Description
1	A	Ozone concentration at Season 1, Latitude 1, Level 1	Exp (E10.3)	Ozone measurements as a function of season, latitude, and vertical level; units = molecules cm-3 (required)
	B	Ozone concentration at Season 1, Latitude 1, Level 2	Exp (E10.3)	Ozone measurements as a function of season, latitude, and vertical level; units = molecules cm-3 (required)
…	…	…	…	…
127	A	Ozone concentration at Season 1, Latitude 19, Level 1	Exp (E10.3)	Ozone measurements as a function of season, latitude, and vertical level; units = molecules cm-3 (required)
	B	Ozone concentration at Season 1, Latitude 19, Level 2	Exp (E10.3)	Ozone measurements as a function of season, latitude, and vertical level; units = molecules cm-3 (required)
…	…	…	…	…
134	A	Temperature profiles at Season 1, Latitude 1, Level 1	Exp (E10.3)	Temperature measurements as a function of season, latitude, and vertical level; units = K (required)
	B	Temperature profiles at Season 1, Latitude 1, Level 2	Exp (E10.3)	Temperature measurements as a function of season, latitude, and vertical level; units = K (required)
…	…	…	…	…
260	A	Temperature profiles at Season 1, Latitude 19, Level 1	Exp (E10.3)	Temperature measurements as a function of season, latitude, and vertical level; units = K (required)
	B	Temperature profiles at Season 1, Latitude 19, Level 2	Exp (E10.3)	Temperature measurements as a function of season, latitude, and vertical level; units = K (required)
…	…	…	…	…
267	A	Air density profiles at Season 1, Latitude 1, Level 1	Exp (E10.3)	Air density measurements as a function of month, latitude, and vertical level; units = molecules cm-3 (required)

Line	Column	Name	Type	Description
	B	Air density profiles at Season 1, Latitude 1, Level 2	Exp (E10.3)	Air density measurements as a function of month, latitude, and vertical level; units = molecules cm-3 (required)
…	…	…	…	…
393	A	Air density profiles at Season 1, Latitude 19, Level 1	Exp (E10.3)	Air density measurements as a function of season, latitude, and vertical level; units = molecules cm-3 (required)
	B	Air density profiles at Season 1, Latitude 19, Level 2	Exp (E10.3)	Air density measurements as a function of season, latitude, and vertical level; units = molecules cm-3 (required)
…	…	…	…	…
1596	A	Air density profiles at Season 4, Latitude 19, Level 51	Exp (E10.3)	Air density measurements as a function of season, latitude, and vertical level; units = molecules cm-3 (required)
1597	A	Average Dobson Values at Latitude 1, Month 1	Real	Average Dobson value as a function of latitude and month (required)
1597	B	Average Dobson Values at Latitude 2, Month 1	Real	Average Dobson value as a function of latitude and month (required)
…	…	…	…	…
1608	A	Average Dobson Values at Latitude 19, Month 12	Real	Average Dobson value as a function of latitude and month (required)
1609	A	Air Temperature at Level 1	Real	Air temperature for a standard atmospheric profile; units = K (required)
	B	Air Density at Level 1	Real or Exp	Air Density for a standard atmospheric profile; units = molecules cm-3(required)
	C	Ozone Concentration at Level 1	Real or Exp	Ozone concentration for a standard atmospheric profile; units = molecules cm-3(required)
	D	Aerosol Attenuation at Level 1	Real or Exp	Aerosol attenuation coefficient for a standard atmospheric profile; units = km-1(required)
1659	A	Air Temperature at Level 51	Real	Air temperature for a standard atmospheric profile, units = K (required)

Line	Column	Name	Type	Description
	B	Air Pressure at Level 51	Real or Exp	Air pressure for a standard atmospheric profile, units = molecules cm-3(required)
	C	Ozone Concentration at Level 51	Real or Exp	Ozone concentration for a standard atmospheric profile, units = molecules cm-3(required)
	D	Aerosol Attenuation at Level 51	Real or Exp	Aerosol attenuation coefficient for a standard atmospheric profile; units = km-1(required)

TOMS: Total ozone mapping spectrometer data

Used by: JPROC

TOMS is the logical name for the ASCII data file containing total ozone column measurements in Dobson units taken by the Total Ozone Mapping Spectrometer instrument aboard the sun-synchronous polar orbiting Nimbus satellite. The data are presented for specific Julian dates as a function of location on the earth (latitude and longitude).

A detailed description of the file format is provided in Table 8-10. The files are fixed format.

** Table 8-10 TOMS Data Profile **

Line	Column	Name	Type	Description
1	A	Julian Day	Int	Julian start day of the file, DDD; preceded by 6 blank spaces (required)
	B	Year	Int	Start year of the file, YYYY; preceded by 9 blank spaces (required)
2		Header	String	80-character line describing the contents of the file (if omitted, needs line placeholder)
3		Header	String	80-character line describing the contents of the file (if omitted, needs line placeholder)
4	A	TOMS Data	Int	TOMS ozone measurements as a function of longitude and latitude, ###; line starts with a space, then space-delimited 25 values per line (required)
…	…	…	…	…

O2ABS: Molecular oxygen absorption cross-section data

Used by: JPROC

O2ABS is the logical name for the ASCII data file containing absorption cross section and quantum yield data for O2 photolysis. The data in this file are listed as a function of wavelength. This file follows the same format as the CSQY files described in this section.

O3ABS: Ozone absorption cross-section data

Used by: JPROC

O3ABS is the logical name for the ASCII data file containing absorption cross section and quantum yield data for O3 photolysis. The data in this file are listed as a function of wavelength. For an example see this link This file follows the same format as the CSQY files described in this section.

InMetFiles: List of MM5 or WRF-ARW output files

Used by: MCIP

MCIP can read MM5 (version 3) binary output files (MMOUT) and WRF-ARW netCDF-based files to generate I/O API-formatted netCDF files for input to CMAQ and emissions modeling. For details about the format of the MMOUT files, visit the MM5 homepage. For information about the format of the WRF output files, visit the WRF-ARW homepage.

InTerFile: Terrain file

Used by: MCIP

MM5 or WRF output file containing fractional land use data. This file is generated by the MM5 program TERRAIN and the WRF program GEOGRID.

InSatFiles: GOES cloud data file

Used by: MCIP

[EPA: need a description of this file]

BNDY_CONC_1: Boundary conditions

Used by: CCTM

CMAQ boundary condition data are of the BNDARY3 file type. Produced by the boundary condition processor, BCON, CCTM reads these data and correlates them with the interior data by the use of a pointer system. This pointer system designates the beginning location of the data in memory that start a new side of the domain (i.e., south, east, north, or west). Figure 8-1 illustrates this input data structure and the relationship of the boundary data to the interior ("main grid") data within CMAQ modules.

Each species being modeled should be in the BNDY_CONC_1 file. If some modeled species are not contained in this file, the boundary condition for these species will default to the value 1 × 10-30. The perimeter of the CMAQ domain is 1 cell wide, where the number of boundary cells = (2*NROW)+(2*NCOL)+4. Figure 8-2 is a graphical example of the CMAQ boundary conditions file; the west and north boundaries have ozone values of 0.035 ppmV, while the east and south boundaries have values of 0.030 ppmV.

Figure 8-1. Illustration of CMAQ boundary condition file

Figure 8-2. Graphical example of a CMAQ gridded boundary conditions file

INIT_CONC_1: Initial conditions

Used by: CCTM

The initial concentrations of each species being modeled must be input to CMAQ. The initial conditions input file type is GRDDED3 and does not vary with time. The file data are looped in this manner: by column, by row, by layer, by variable. Initial conditions files have the same structure as concentration files, so the predicted concentrations from the last hour of day 1 can be used to initialize the following day's simulation. This gives CMAQ users the flexibility to segment simulations in any way they choose. Figure 8-3 is a graphical example of the CMAQ initial conditions file. The file shows spatially varying data that can be used to initialize a following run beginning at the time shown (i.e., June 25, 1996 0:00:00).

<Image:>

Figure 8-3. Graphical example of a CMAQ gridded initial conditions file

JTABLE: Photolysis rates look-up table

Used by: CCTM

Each of the gas-phase mechanisms in CMAQ contains photolysis reactions that require clear-sky reaction rates precomputed from kinetics data at various altitudes and latitude bands. The CMAQ program JPROC (Section 2.2.3) generates photolysis rate look-up tables for input to CMAQ. The photolysis files, called JTABLE, contain a header section that describes the contents of the file, and a data section with clear-sky photolysis rates at different times of the day.

The first line of the header contains the Julian date of the data in the file. This date corresponds to the start date for the simulation that uses the JTABLE data. This is followed by four pairs of data that describe the altitude (m), latitude (degrees), hour angle (from noon), and photolytic reaction name for the data contained in the file. Each data pair begins with a line describing the number of values for each variable and the variable name, followed by the values for each variable. The altitude (variable = LEVELS), latitude (variable = LATITUDES), and hour angle (variable = HOUR ANGLES) variables have fixed values that are hard-coded in the program JPROC (routine jproc.F). The reaction names (variable = PHOTOLYTIC REACTIONS) are mechanism-dependent and vary with the choice of gas-phase mechanism.

The data section of the file contains data blocks that are mapped to the header using a three-digit code. Each block corresponds to an altitude, latitude, and photolysis reaction and contains nine values of clear-sky photolysis rates for the nine hour angles listed in the header. The three-digit code maps the altitude/latitude/reaction number to the data block. For example, the data block that uses the code "3 1 2" corresponds to altitude 3, latitude 1, and reaction 2 as listed in the file header. A detailed description of the JTABLE file format is provided in Table 8-11. The files are list-formatted.

Table 8-11. JTABLE file format description

Line	Column	Name	Type	Description
1	A	JVDATE	String	Julian date of the file; YYYYDDD (required)
	B	Comment	String	Description of the Julian date field (optional)
2	A	JVHT	Int	Number of vertical levels covered by the data (required)
	B	Level Field Indicator	String	The word "LEVELS" (required)

Line	Column	Name	Type	Description
	C	Comment	String	Description of the level field; usually "(m)", for meters (optional)
3	A	XZJV_1	Real	Height of level 1; units in m (required)
	B	XZJV_2	Real	Height of level 2; units in m (required)
…	…	…	…	…
…	x	XZJV_x	Real	Height of level x; units in m (required)
4	A	JVLAT	Int	Number of latitudes covered by the data (required)
	B	Latitude Field Indicator	String	The word "LATITUDES" (required)
	C	Comment	String	Description of the latitudes field; usually "(deg)", for degrees (optional)
5	A	XLATJV_1	Real	Latitude 1; units in degrees (required)
	B	XLATJV_2	Real	Latitude 2; units in degrees (required)
…	…	…	…	…
…	x	XLATJV_x	Real	Latitude x; units in degrees (required)
6	A	JVTMAX	Int	Number of hour angles covered by the data (required)
	B	Hour Angle Field Indicator	String	The words "HOUR ANGLES" (required)
	C	Comment	String	Description of the hour angles field; usually "(from noon)", for hours from noon (optional)
7	A	XHAJV_1	Real	Hour angle 1; units in hours from noon (required)
	B	XHAJV_2	Real	Hour angle 2; units in hours from noon (required)
…	…	…	…	…
…	x	XHAJV_x	Real	Hour angle x; units in hours from noon (required)
8	A	JPHOT	Int	Number of photolytic reactions covered by the data (required)
	B	Reaction Field Indicator	String	The words "PHOTOLYTIC REACTIONS" (required)
	C	Comment	String	Description of the reactions field (optional)
9	A	PHOT_NM_1	String	Single quote-delimited name of photolysis reaction 1 (required)
	B	Delimiter	Char	Comma delimiter separating reaction name from multiplier (required)
	C	ACLD_1	Real	Multiplier for reaction 1 (required)
10	A	PHOT_NM_2	String	Single quote-delimited name of photolysis reaction 2 (required)
	B	Delimiter	Char	Comma delimiter separating reaction name from multiplier (required)
	C	ACLD_2	Real	Multiplier for reaction 2 (required)

Line	Column	Name	Type	Description
…	…	…	…	…
x	A	PHOT_NM_x	String	Single quote-delimited name of photolysis reaction x (required)
	B	Delimiter	Char	Comma delimiter separating reaction name from multiplier (required)
	C	ACLD_x	Real	Multiplier for reaction x (required)
x + 1	A	NHTO	Int	Vertical level cross-reference to header data (required)
	B	NLATO	Int	Latitude cross-reference to header data (required)
	C	NPHOTO	Int	Photolysis reaction cross-reference to header data (required)
x+2	A	XJVAL_1	Real or Exp	Clear-sky photolysis rate at NHTO, NLATO, NPHOTO, and hour angle 1 (required)
	B	XJVAL_2	Real or Exp	Clear-sky photolysis rate at NHTO, NLATO, NPHOTO, and hour angle 2 (required)
	C	XJVAL_3	Real or Exp	Clear-sky photolysis rate at NHTO, NLATO, NPHOTO, and hour angle 3 (required)
…	…	…	…	…
	JVTMAX	XJVAL_(JVTMAX)	Real or Exp	Clear sky photolysis rate at NHTO, NLATO, NPHOTO, and hour angle corresponding to JVTMAX (required)
x + 3	A	NHTO	Int	Vertical level cross-reference to header data (required)
	B	NLATO	Int	Latitude cross-reference to header data (required)
	C	NPHOTO	Int	Photolysis reaction cross-reference to header data (required)
x+4	A	XJVAL_1	Real or Exp	Clear-sky photolysis rate at NHTO, NLATO, NPHOTO, and hour angle 1 (required)
	B	XJVAL_2	Real or Exp	Clear-sky photolysis rate at NHTO, NLATO, NPHOTO, and hour angle 2 (required)
	C	XJVAL_3	Real or Exp	Clear-sky photolysis rate at NHTO, NLATO, NPHOTO, and hour angle 3 (required)
…	…	…	…	…
	JVTMAX	XJVAL_(JVTMAX)	Real or Exp	Clear-sky photolysis rate at NHTO, NLATO, NPHOTO, and hour angle corresponding to JVTMAX (required)
…	…	…	…	…

A sample of the important sections of a JTABLE file is shown below.

```
1999181 (yyyyddd) Julian Date for the file  7 LEVELS (m)  0.0 1000.0 2000.0 3000.0
4000.0 5000.0 10000.0 6 LATITUDES (deg) 10.0 20.0 30.0 40.0 50.0 60.0 9 HOUR ANGLES
(from noon) 0.0 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 6 PHOTOLYTIC REACTIONS 'NO2_CBIV88
```

```
', 1.0 'O3O1D_CBIV88 ', 1.0 'HCHOmol_CBIV88 ', 1.0 'HCHOrad_CBIV88 ', 1.0 'ALD_CBIV88
', 1.0 'ACROLEIN ', 1.0 1 1 1 5.0964761E-01 4.9923715E-01 4.6422747E-01 4.0129572E-01
3.0394882E-01 1.6590215E-01 3.2829735E-02 0.0000000E+00 0.0000000E+00
```

OMI: Ozone Monitoring Instrument Column Data

Used by: CCTM

OMI ozone column data by latitude and longitude for use in the inline photolysis calculations. CMAQ is distributed with ozone columns from 1978 to 2015. The data are 22.5°x10° gridded ozone columns in Dobson units. Table 8-12 lists the format of the OMI data file.

Table 8-12. OMI data format

Line	Column	Name	Type	Description
1		Header		Header with names for each column
2	A	Yeardate	Real	YYYY.??? or YYYY.???? formatted date field.
	B	Latitude 1	Int	80 North latitude
	C	Longitude 1	Int	180.0Z longitude ozone column (DU)
	D	Longitude 2	Int	157.5W longitude ozone column (DU)
	E	Longitude 3	Int	135.0W longitude ozone column (DU)
	F	Longitude 4	Int	112.5W longitude ozone column (DU)
	G	Longitude 5	Int	090.0W longitude ozone column (DU)
	H	Longitude 6	Int	067.5W longitude ozone column (DU)
	I	Longitude 7	Int	045.0W longitude ozone column (DU)
	J	Longitude 8	Int	022.5W longitude ozone column (DU)
	K	Longitude 9	Int	000.0Z longitude ozone column (DU)
	L	Longitude 10	Int	022.5E longitude ozone column (DU)
	M	Longitude 11	Int	045.0E longitude ozone column (DU)
	N	Longitude 12	Int	067.5E longitude ozone column (DU)
	O	Longitude 13	Int	090.0E longitude ozone column (DU)
	P	Longitude 14	Int	112.5E longitude ozone column (DU)
	Q	Longitude 15	Int	135.0E longitude ozone column (DU)
	R	Longitude 16	Int	157.5E longitude ozone column (DU)
	S	Longitude 17	Int	180.0Z longitude ozone column (DU)
3	A	Yeardate	Real	YYYY.??? or YYYY.???? formatted date field.
	B	Latitude 2	Int	70 North latitude
	C	Longitude 1	Int	180.0Z longitude ozone column (DU)
…	…	…	…	…
	S	Longitude 17	Int	180.0Z longitude ozone column (DU)
4	A	Yeardate	Real	YYYY.??? or YYYY.???? formatted date field.
	B	Latitude 3	Int	60 North latitude
	C	Longitude 1	Int	180.0Z longitude ozone column (DU)
…	…	…	…	…
	S	Longitude 17	Int	180.0Z longitude ozone column (DU)
5	A	Yeardate	Real	YYYY.??? or YYYY.???? formatted date field.
	B	Latitude 4	Int	50 North latitude

Line	Column	Name	Type	Description
	C	Longitude 1	Int	180.0Z longitude ozone column (DU)
…	…	…	…	…
	S	Longitude 17	Int	180.0Z longitude ozone column (DU)
…	…	…	…	Repeat for total of 17 latitudes of data
…	…	…	…	Repeat for (1978-2008) there are ~48 days (4 days per month) of data
…	…	…	…	Repeat for (2009-2015) there are 365 days of data

EMIS_1: Emissions

Used by: CCTM

CMAQ can accept emissions inputs from a variety of emissions models and preprocessors. The most commonly used option is the Sparse Matrix Operator Kernel Emissions (SMOKE) modeling system, which is a collection of programs that separately process and merge emissions data for each emissions sector for input to air quality models.

The emissions file sorts the emitted gas-phase and aerosol species by grid cell and time. The file type is GRDDED3, and the units are in moles per second (moles s-1) for gas-phase species and grams per second (g s-1) for aerosol species. The file data are looped as follows: by column, by row, by layer, by variable, and by input time step. CMAQ does not artificially distinguish between surface and elevated emissions sources; elevated sources are provided to CMAQ as vertically resolved emissions. For CCTM configurations that do not use in-line emissions calculations, all emissions estimates are contained within a single input emission file for each day. In v4.7, CMAQ now has the capability to process point-source, sea salt, and biogenic emissions in-line. The supplemental input files to use for calculating the in-line emissions are described in the CMAQv4.7 release notes.

OCEAN_1: Sea salt mask

Used by: CCTM

The CMAQ aerosol models AERO5 and AERO6 can compute sea salt emissions from both open ocean grid cells and surf zone grid cells. The addition of the surf zone option simulates the elevated emissions rates of sea salt in coastal regions where wave action occurs. The OCEAN_1 file contains data on the fraction of each grid cell that is either open ocean (OPEN) or in the surf zone (SURF). When CCTM is compiled with AERO5 or AERO6, it will expect the OCEAN_1 file as input.

GSPRO: Speciation profiles

Used by: CCTM – inline emissions version only

The speciation profile file, GSPRO, contains the factors that are used to separate aggregated inventory pollutant emissions totals into emissions of model species in the form required by CMAQ. If only biogenic emissions are being calculated in-line in CMAQ, the GSPRO file used by CCTM needs to contain split factors only for the biogenic VOC emissions that are input in the B3GRD file. If other emissions sources are being calculated by CCTM, VOC split factors for these other sources must be included in the GSPRO file. The GSPRO file format is listed in the SMOKE user's manual, see: GSPRO documentation.

B3GRD: Gridded, normalized biogenic emissions

Used by: CCTM – inline-emissions version only

An I/O API GRDDED3 file of gridded, normalized biogenic emissions (in grams of carbon or nitrogen per hour, depending on the species) and leaf area index. The B3GRD file contains normalized emissions calculated with both summer and winter emissions factors. The B3GRD file is generated with the SMOKE program NORMBEIS3 using gridded land use data. For additional information about creating the B3GRD file, see the NORMBEIS3 documentation in the SMOKE users' manual.

BIOSEASON: Freeze dates

Used by: CCTM – inline-emissions version only

The BIOSEASON switch file is an I/O API GRDDED3 file used to indicate which biogenic emissions factor to use on each day in a given year for every grid cell in the modeling domain. This file can be created using the Metscan utility program that is distributed with SMOKE. The BIOSEASON file is time-dependent and usually contains data for an entire year (365 or 366 days). It uses one variable, SEASON, which is either 0 (grid cell should use winter factors for current day) or 1 (grid cell should use summer factors for current day). For additional information about creating the BIOSEASON file, see the Metscan documentation in the SMOKE user's manual.

STK_GRPS_nn: Stack groups

Used by: CCTM – in-line emissions version only

The ## mark is unique and represents the sector identification.

The stack groups file is an I/O API netCDF file containing stack parameters for elevated sources. This file can be created using the SMOKE program ELEVPOINT. For additional information about creating the stack groups file, see the ELEVPOINT documentation in the SMOKE user's manual

STK_EMIS_nn: Point source emissions

Used by: CCTM – inline emissions version only

The ## mark is unique and represents the sector identification.

The elevated-point-source emissions file is an I/O API GRDDED3 file with emissions for point sources to be treated as elevated sources by CCTM. The emissions in this file are distributed through the vertical model layers using a plume-rise algorithm contained in CCTM. The elevated-point-source emissions file can be creating using SMOKE. For additional information about preparing point-source emissions for using the CMAQ in-line plume rise calculation, see the ELEVPOINT documentation in the SMOKE user's manual.

DUST_LU_1: Gridded land cover/land use

Used by: CCTM – in-line dust emission version only

The gridded land cover/land use (LCLU) file is an I/O API GRDDED3 file of BELD3 data projected to the modeling domain. This file must contain the following LCLU variables to be compatible with the CMAQ dust module:

- USGS_urban
- USGS_drycrop

- USGS_irrcrop
- USGS_cropgrass
- USGS_cropwdlnd
- USGS_grassland
- USGS_shrubland
- USGS_shrubgrass
- USGS_savanna
- USGS_decidforest
- USGS_evbrdleaf
- USGS_coniferfor
- USGS_mxforest
- USGS_water
- USGS_wetwoods
- USGS_sprsbarren
- USGS_woodtundr
- USGS_mxtundra
- USGS_snowice

These categories are used to determine dust source locations and canopy scavenging factors for estimating dust emission in the model. This file can be created for North America using the Spatial Allocator and BELD4 tiles. The DUST_LU_1 file corresponds to the "a" output file from the Spatial Allocator. See the chapter on creating biogenic inputs to SMOKE of the Spatial Allocator User's Guide for details.

DUST_LU_2: Gridded land cover/land use

Used by: CCTM – in-line dust emission version only

The gridded land cover/land use (LCLU) file is an I/O API GRDDED3 file of BELD3 data projected to the modeling domain. This file must contain the following variables to be compatible with the CMAQ dust module:

- FOREST

This variable is used in combination with the variables in the DUST_LU_1 file to determine canopy scavenging factors for estimating dust emission in the model. This file can be created for North America using the Spatial Allocator and BELD3 tiles. The DUST_LU_2 file corresponds to the "tot" output file from the Spatial Allocator. See the chapter on creating biogenic inputs to SMOKE of the Spatial Allocator User's Guide for details

MODIS_FPAR: MODIS vegetation coverage

Used by: CCTM – in-line dust emission version only

MODIS_FPAR file is an I/O API file with 2-d (row x col) daily values of Fraction of Photosynthetically Active Radiation (FPAR) from MODIS instrument interpolated to the modeling domain. It is required to obtain a dynamic vegetation fraction used by the in-line windblown dust emission module.

CROPMAP01: Gridded planting start dates

Used by: CCTM – in-line dust emission version with crops only

The gridded planting start dates file is an I/O API GRDDED3 file of planting start dates for various crops interpolated to the modeling domain. The variables in this file are planting start dates for different crop types, where each variable is an integer representing the number of days after January 1 that planting stops for each crop. The CMAQ preprocessing program CALMAP reads a crop activity calendar and a GRID_CRO_2D file to generate the CROPMAP08 file.

CROPMAP04: Gridded planting end dates

Used by: CCTM – in-line dust emission version with crops only

The gridded planting end dates file is an I/O API GRDDED3 file of planting end dates for various crops interpolated to the modeling domain. The variables in this file are planting end dates for different crop types, where each variable is an integer representing the number of days after January 1 that planting stops for each crop. The CMAQ preprocessing program CALMAP reads a crop activity calendar and a GRID_CRO_2D file to generate the CROPMAP08 file.

CROPMAP08: Gridded harvesting end dates

Used by: CCTM – in-line dust emission version with crops only

The gridded harvesting end dates file is an I/O API GRDDED3 file of harvesting end dates for various crops interpolated to the modeling domain. The variables in this file are harvesting end dates for different crop types, where each variable is an integer representing the number of days after January 1 that harvesting stops for each crop. The CMAQ preprocessing program CALMAP reads a crop activity calendar and a GRID_CRO_2D file to generate the CROPMAP08 file.

LTNGNO: Lightning NOx emissions

Used by: CCTM – lightning NOx version only

The lightning NOx emissions file is an I/O API GRDDED3 file with 3-d (row x col x layer) hourly NO emissions (moles/s) interpolated to the modeling domain. This is a lightning NO emissions file calculated off-line for input to CMAQ.

LTNGPARMS_FILE: Lightning parameters file

Used by: CCTM – lightning NOx version only

The lightning parameters file is used for calculating in-line NO emissions from hourly observed strike counts. This file contains the following variables interpolated to the modeling grid:

- SLOPE (unitless): linear equation parameter for estimating NO emissions from hourly flash counts
- INTERCEPT: linear equation parameter for estimating NO emissions from hourly flash counts
- SLOPE_lg: logarithmic equation parameter for estimating NO emissions from hourly flash counts
- INTERCEPT_lg: logarithmic equation parameter for estimating NO emissions from hourly flash counts
- ICCG_SUM (unitless): Ratio of intercloud to cloud-to-ground flashes during the summer season
- ICCG_WIN (unitless): Ratio of intercloud to cloud-to-ground flashes during the winter season

- OCNMASK (unitless): Land/water mask to remove spurious flashes over the ocean.

NLDN_STRIKES: Hourly observed lightning strikes

Used by: CCTM – lightning NOx version only

The NLDN lightning strikes file is used for calculating in-line NO emissions from hourly observed strike counts. This file contains the following variables interpolated to the modeling grid:

- NLDNstrk (km-2): hourly flash counts per sq. km.

BELD4_LU – Fractional crop distributions

Used by: CCTM – bidirectional NH3 flux version only

Add content

E2C_SOIL – EPIC soil properties

Used by: CCTM – bidirectional NH3 flux version only

This 3-D file is created by the EPIC model via the FEST-C interface and contains soil properties for Layer 1 (0 to 1 cm depth) and Layer 2 (1 cm to 100 cm depth) for each crop and soil combination in each grid cell. Additional information on the EPIC model and the FEST-C interface are available at https://www.cmascenter.org/fest-c/. The following variables are in this file:

- L1_SoilNum: Soil Number (none)
- L1_Bulk_D: Layer1 Bulk Density (t/m**3)
- L1_Wilt_P: Layer1 Wilting Point(m/m)
- L1_Field_C: Layer1 Field Capacity (m/m)
- L1_Porosity: Layer1 Porosity (%)
- L1_PH: Layer1 PH (none)
- L1_Cation: Layer1 Cation Ex (cmol/kg)
- L2_Bulk_D: Layer2 Bulk Density (t/m**3)
- L2_Wilt_P: Layer2 Wilting Point (m/m)
- L2_Field_C: Layer2 Field Capacity (m/m)
- L2_Porosity: Layer2 Porosity (%)
- L2_PH: Layer2 PH (none)
- L2_Cation: Layer2 Cation Ex (cmol/kg)

E2C_FERT – EPIC crop types and fertilizer application

Used by: CCTM – bidirectional NH3 flux version only

This is a 3-D daily file created by the EPIC model via the FEST-C interface and contains information on fertilizer application rate and depth for each crop and soil combination in each grid cell. Additional information on the EPIC model and the FEST-C interface are available at https://www.cmascenter.org/fest-c/. The file contains many more variables than are used by CMAQ. The following variables are in this file:

- QNO3: N Loss in Surface Runoff (kg/ha)
- SSFN: N in Subsurface Flow (kg/ha)
- PRKN: N Loss in Percolate (kg/ha)
- DN: N-NO3 Denitrification (kg/ha)
- DN2: N-N2O from NO3 Denitrification (kg/ha)
- AVOL: N-NH3 Emission (kg/ha)
- HMN: OC Change by Soil Respiration (kg/ha)
- NFIX: N Fixation (kg/ha)
- YP: P Loss with Sediment (kg/ha)
- QAP: Labile P Loss in Runoff (kg/ha)
- YON: N Loss with Sediment (kg/ha)
- YW: Wind Erosion (ton/ha)
- Q: Runoff (mm)
- HUSC: Heat Unit Schedule (none)
- HU_BASE0: Base Heat Unit (none)
- HU_FRAC: Heat Unit fraction (none)
- L1_DEP: Layer1 Depth (m)
- L1_BD: Layer1 Bulk Density (t/m**3)
- L1_NO3: Layer1 N - Nitrate (kg/ha)
- L1_NH3: Layer1 N - Ammonia (kg/ha)
- L1_ON: Layer1 Organic N (kg/ha)
- L1_P: Layer1 Mineral P (kg/ha)
- L1_OP: Layer1 Organic P (kg/ha)
- L1_C: Layer1 Carbon (kg/ha)
- L1_NITR: Layer1 N - Nitrified NH3 (kg/ha)
- L2_DEP: Layer2 Depth (m)
- L2_BD: Layer2 Bulk Density (t/m**3)
- L2_NO3: Layer2 N - Nitrate (kg/ha)
- L2_NH3: Layer2 N - Ammonia (kg/ha)
- L2_ON: Layer2 Organic N (kg/ha)
- L2_P: Layer2 Mineral P (kg/ha)
- L2_OP: Layer2 Organic P (kg/ha)
- L2_C: Layer2 Carbon (kg/ha)
- L2_NITR: Layer2 N - Nitrified NH3 (kg/ha)
- T1_DEP: Layert (Total Soil Profile) Depth (m)
- T1_BD: Layert Bulk Density (t/m**3)
- T1_NO3: Layert N - Nitrate (kg/ha)
- T1_NH3: Layert N - Ammonia (kg/ha)
- T1_ON: Layert Organic N (kg/ha)
- T1_P: Layert Mineral P (kg/ha)
- T1_OP: Layert Organic P (kg/ha)
- T1_C: Layert Carbon (kg/ha)
- T1_NITR: Layert N - Nitrified NH3 (kg/ha)
- L1_ANO3: Layer1 N-NO3 AppRate (kg/ha)
- L1_ANH3: Layer1 N-NH3 AppRate (kg/ha

- L1_AON: Layer1 ON AppRate (kg/ha)
- L1_AMP: Layer1 MP AppRate (kg/ha)
- L1_AOP: Layer1 OP AppRate (kg/ha)
- L2_ANO3: Layer2 N-NO3 AppRate (kg/ha)
- L2_ANH3: Layer2 N-NH3 AppRate (kg/ha)
- L2_AON: Layer2 ON AppRate (kg/ha)
- L2_AMP: Layer2 MP AppRate (kg/ha)
- L2_AOP: Layer2 OP AppRate (kg/ha)
- UN1: N Uptake by Crop (kg/ha)
- HUI: Heat Unit Index (none)
- LAI: Leaf Area Index (none)
- CPHT: Crop Height (m)

INIT_MEDC_1 – Soil initial conditions file

Used by: CCTM – bidirectional NH3 flux version only

The ASXfile output for the previous day from the bidirectional NH3 and/or Hg model is used to initialize the soil conditions for each simulation day. This file contains soil NH4 concentrations, soil pH, and Soil, vegetation and water Hg.

GRID_CRO_2D: Two-dimensional grid cross-point fields

Used by: CCTM

The GRID_CRO_2D time-independent file contains surface fields at cross points (i.e., at cell centers). It is created by MCIP and used by CCTM. The following variables are in this file:

- LAT: latitude (degrees, where Northern Hemisphere is positive)
- LON: longitude (degrees, where Western Hemisphere is negative)
- MSFX2: squared map-scale factor (m2 m-2)
- HT: terrain elevation (m)
- DLUSE: dominant land use (category)
- LWMASK: land-water mask (1=land, 0=water)
- PURB: urban percentage if cell is based on land (percent)
- LUFRAC_01: land use fraction of NLCD40: Evergreen Needleleaf Forest
- LUFRAC_XX:

GRID_DOT_2D: Two-dimensional grid dot-point fields

Used by: CCTM

The GRID_DOT_2D time-independent file contains surface fields at dot points (i.e., at cell corners). It is created by MCIP and used by CCTM. The following variables are in the GRID_DOT_2D file:

- LAT: latitude (degrees, where Northern Hemisphere is positive)
- LON: longitude (degrees, where Western Hemisphere is negative)
- MSFD2: squared map scale factor (m2 m-2)

GRID_CRO_3D: Three-dimensional grid cross-point fields Used by: CCTM

The GRID_CRO_3D time-independent file contains surface fields at cross points (i.e., at cell centers) that vary by height. It is created by MCIP and used by CCTM for the PT3D. The following variables are in this file: add content

MET_BDY_3D: Three-dimensional meteorological boundary input

Used by: CCTM

The MET_BDY_3D time-dependent file contains 3-D meteorological descriptions at the lateral boundaries (on cross points). It is created by MCIP and used by CCTM and PDM. The following variables may be in the MET_BDY_3D file:

- JACOBF: total Jacobian at layer face (m)
- JACOBM: total Jacobian at layer middle (m)
- DENSA_J: Jacobian-weighted total air density [? J m-2] (kg m-2)
- WHAT_JD: Jacobian- and density-weighted vertical contravariant velocity (kg m-1 s-1)
- TA: air temperature (K)
- QV: water vapor mixing ratio (kg kg-1)
- PRES: air pressure (Pa)
- DENS: air density (kg m-3)
- WWIND: vertical velocity (m s-1)
- ZH: midlayer height above ground (m)
- ZF: full layer height above ground (m)
- QC: cloud water mixing ratio (kg kg-1)
- QR: rain water mixing ratio (kg kg-1)
- QI: ice mixing ratio (kg kg-1)
- QS: snow mixing ratio (kg kg-1)
- QG: graupel mixing ratio (kg kg-1)

MET_CRO_2D: Two-dimensional meteorological cross-point fields

Used by: CCTM

The MET_CRO_2D time-dependent file contains surface and other 2-D meteorological fields at cross points (i.e., at cell centers). It is created by MCIP and used by CCTM and PDM. The following variables may be in the MET_CRO_2D file:

- PRSFC: surface pressure (Pa)
- JACOBS: total Jacobian at surface (m)
- USTAR: cell-averaged horizontal friction velocity (m s-1)
- WSTAR: convective velocity scale (m s-1)
- PBL: planetary boundary layer height (m)
- ZRUF: surface roughness length (m)
- MOLI: inverse Monin-Obukhov length (m-1)
- QFX: latent heat flux (W m-2)
- HFX: sensible heat flux (W m-2)

- RADYNI: inverse aerodynamic resistance (m s-1)
- RBNDYI: inverse laminar boundary layer resistance (m s-1)
- RSTOMI: inverse bulk stomatal resistance (m s-1)
- TEMPG: skin temperature at ground (K)
- TEMP10: 10-m temperature (K)
- TEMP1P5: 1.5-m temperature (K)
- WSPD10: 10-m wind speed (m s-1)
- WDIR10: 10-m wind direction (m s-1)
- GLW: longwave radiation at ground (W m-2)
- GSW: solar radiation absorbed at ground (W m-2)
- RGRND: solar radiation reaching the surface (W m-2)
- RN: incremental (per output time step) nonconvective precipitation (cm)
- RC: incremental (per output time step) convective precipitation (cm)
- CFRAC: total cloud fraction (fraction)
- WBAR: average liquid water content of clouds (g m-3)
- CLDT: cloud-top layer height (m)
- CLDB: cloud-bottom layer height (m)
- SNOCOV: snow cover (1 = yes, 0 = no)
- TEMP2: 2-m temperature (K)
- SOIM1: volumetric soil moisture in top cm (m3 m-3)
- SOIM2: volumetric soil moisture in top m (m3 m-3)
- SOIT1: soil temperature in top cm (K)
- SOIT2: soil temperature in top m (K)
- SLTYP: soil texture type (category)
- LAI: leaf-area index (area area-1)

The following deposition velocities are calculated by MCIP3 by default and written to the MET_CRO_2D file:

- VD_SO2: deposition velocities for SO2 (m s-1)
- VD_SULF: deposition velocities for SO4 (m s-1)
- VD_NO2: deposition velocities for NO2 (m s-1)
- VD_NO: deposition velocities for NO (m s-1)
- VD_O3: deposition velocities for O3 (m s-1)
- VD_HNO3: deposition velocities for HNO3 (m s-1)
- VD_H2O2: deposition velocities for H2O2 (m s-1)
- VD_ALD: deposition velocities for ALD (m s-1)
- VD_HCHO: deposition velocities for HCHO (m s-1)
- VD_OP: deposition velocities for OP (m s-1)
- VD_PAA: deposition velocities for PAA (m s-1)
- VD_ORA: deposition velocities for ORA (m s-1)
- VD_NH3: deposition velocities for NH3 (m s-1)
- VD_PAN: deposition velocities for PAN (m s-1)
- VD_HONO: deposition velocities for HONO (m s-1)

- VD_CO: deposition velocities for CO (m s-1)
- VD_METHANOL: deposition velocities for methanol (m s-1)
- VD_N2O5: deposition velocities for N2O5 (m s-1)
- VD_NO3: deposition velocities for NO3 (m s-1)
- VD_GEN_ALD: deposition velocities for generic aldehyde (m s-1)
- VD_CL2: deposition velocities for CL2 (m s-1)
- VD_HOCL: deposition velocities for HOCL (m s-1)
- VD_HCL: deposition velocities for HCL (m s-1)
- VD_FMCL: deposition velocities for FMCL (m s-1)
- VD_ICL1: deposition velocities for ICL1 (m s-1)
- VD_ICL2: deposition velocities for ICL2 (m s-1)
- VD_HG: deposition velocities for HG (m s-1)
- VD_HGIIGAS: deposition velocities for HGIIGAS (m s-1)

MET_CRO_3D: Three-dimensional meteorological cross-point fields

Used by: CCTM, ICON, BCON

The MET_CRO_3D time-dependent file contains 3-D meteorological descriptions at cross points (i.e., at cell centers). It is created by MCIP and used by CCTM, ICON, BCON, and PDM. The variables that may exist in MET_CRO_3D are the same as those that may be in MET_BDY_3D.

7.0.2 MET_DOT_3D: Three-dimensional meteorological dot-point fields

Used by: CCTM

The MET_DOT_3D time-dependent file contains 3-D meteorological descriptions at dot points (i.e., at cell corners) and at cell faces. It is created by MCIP and used by CCTM and PDM. The following variables may be in the MET_DOT_3D file:

- UWIND: u-component of horizontal wind (m s-1) [dot points; Arakawa-B grid]]
- VWIND: v-component of horizontal wind (m s-1) [dot points; Arakawa-B grid]
- UHAT_JD: contravariant-U*Jacobian*density (kg m-1 s-1) [cell faces; Arakawa-C grid]
- VHAT_JD: contravariant-V*Jacobian*density (kg m-1 s-1) [cell faces; Arakawa-C grid]

CCTM Output Files

The previous section described the output files from JPROC, ICON, BCON, and MCIP that are input to CCTM. In this section, details on the CCTM output files are provided. Except for JPROC (which creates ASCII files), all CMAQ programs produce output files that adhere to the I/O API netCDF format (Chapter 4). The I/O API-formatted CMAQ output files are three-dimensional, gridded, time-stepped binary files that contain headers with metadata describing the file contents. These machine-independent and network transparent binary files are transferable between different computer architectures. In addition to model data output, CMAQ can optionally produce log files that contain the standard output from the various CMAQ processors. If the log file option is not selected by the user, CMAQ will write

all of the log information to the screen along with the standard error, which can be captured to a text file using basic UNIX syntax.

Table 8-13. CMAQ Output files

File Name	File Type	Time-Dependence	Spatial Dimensions
General			
Output Log	ASCII	n/a	n/a
CTM_CONC_1	GRDDED3	Hourly	[2(X+1)+2(Y+1)]*Z
CTM_CGRID_1	GRDDED3	1-hour	[2(X+1)+2(Y+1)]*Z
CTM_ACONC_1	GRDDED3	Hourly	[2(X+1)+2(Y+1)]*Z
CTM_DRY_DEP_1	GRDDED3	Hourly	[2(X+1)+2(Y+1)]*Z
CTM_WETDEP_1	GRDDED3	Hourly	[2(X+1)+2(Y+1)]*Z
CTM_VIS_1	GRDDED3	Hourly	[2(X+1)+2(Y+1)]*Z
CTM_AVIS_1	GRDDED3	Hourly	[2(X+1)+2(Y+1)]*Z
Diagnostic and Advanced			
CTM_PMDIAG_1	GRDDED3	Hourly	[2(X+1)+2(Y+1)]*Z
CTM_APMDIAG_1	GRDDED3	Hourly	[2(X+1)+2(Y+1)]*Z
B3GTS_S	GRDDED3	Hourly	[2(X+1)+2(Y+1)]
CTM_DEPV_DIAG	GRDDED3	Hourly	[2(X+1)+2(Y+1)]
CTM_PT3D_DIAG	GRDDED3	Hourly	[2(X+1)+2(Y+1)]*Z
CTM_DUST_EMIS_1	GRDDED3	Hourly	[2(X+1)+2(Y+1)]
CTM_AOD_1	GRDDED3	Hourly	[2(X+1)+2(Y+1)]*Z
CTM_IPR_1-3	GRDDED3	Hourly	[2(X+1)+2(Y+1)]*Z
CTM_IRR_1-3	GRDDED3	Hourly	[2(X+1)+2(Y+1)]*Z
FLOOR	ASCII	Hourly	n/a
MEDIA_CONC	GRDDED3	Hourly	[2(X+1)+2(Y+1)]
CTM_DEPV_MOS	GRDDED3	Hourly	[2(X+1)+2(Y+1)]
CTM_DRY_DEPV_MOS	GRDDED3	Hourly	[2(X+1)+2(Y+1)]
CTM_DEPV_FST	GRDDED3	Hourly	[2(X+1)+2(Y+1)]
CTM_DRY_DEPV_FST	GRDDED3	Hourly	[2(X+1)+2(Y+1)]
CTM_VDIFF_DIAG	GRDDED3	Hourly	[2(X+1)+2(Y+1)]
CTM_VSED_DIAG	GRDDED3	Hourly	[2(X+1)+2(Y+1)]
LTNG_HOURLY	GRDDED3	Hourly	[2(X+1)+2(Y+1)]*Z
LTNG_COL	GRDDED3	Hourly	[2(X+1)+2(Y+1)]
PLAY_SRCID	GRDDED3	Hourly	[2(X+1)+2(Y+1)]
CTM_RJ_1-2	GRDDED3	Hourly	[2(X+1)+2(Y+1)]
SOILOUT	GRDDED3	Hourly	[2(X+1)+2(Y+1)]
CTM_SSEMIS_1	GRDDED3	Hourly	[2(X+1)+2(Y+1)]
CTM_WETDEP_2	GRDDED3	Hourly	[2(X+1)+2(Y+1)]

The previous section described the output files from JPROC, ICON, BCON, and MCIP that are input to CCTM. In this section, details on the CCTM output files are provided. Except for JPROC (which creates ASCII files), all CMAQ programs produce output files that adhere to the I/O API netCDF format

(Chapter 4). The I/O API-formatted CMAQ output files are three-dimensional, gridded, time-stepped binary files that contain headers with metadata describing the file contents. These machine-independent and network transparent binary files are transferable between different computer architectures. In addition to model data output, CMAQ can optionally produce log files that contain the standard output from the various CMAQ processors. If the log file option is not selected by the user, CMAQ will write all of the log information to the screen along with the standard error, which can be captured to a text file using basic UNIX syntax.

CMAQ output log

All of the CMAQ processors generate standard output and standard error during execution. For all of the processors other than CCTM, this diagnostic output information can be captured to a log file at execution using a UNIX redirect command. For example, to capture the standard output and error of a CCTM simulation, use the following command:

```
run.cctm >& tee cctm.log
```

For CCTM, the LOGFILE environment variable allows users to specify the name of a log file for capturing the standard output from the program. If this variable is not set, the standard output is written to the terminal and can be captured using the UNIX redirect command (“>”), as shown in the example above.

CTM_CONC_1: CCTM hourly instantaneous concentration file

The 3-D CCTM hourly concentration file (CONC) is the most commonly referenced CCTM output file. Containing gas-phase species mixing ratios (ppmV) and aerosol species concentrations (µg m-3), CONC files include instantaneous model species concentrations at the end of each model hour. The number and types of species contained in the CONC files depend on the chemical mechanism and aerosol model configurations that are selected when CCTM is compiled. The FORTRAN NameLists within the mechanism directories list the modeled species, and contain a column that specifies which species are written to the CONC files. The GC_*mechname*.nml file lists the gas-phase species, the AE_*mechname*.nml file lists the aerosol species, and the NR_*mechname*.nml lists the nonreactive (inert) species. Species can be removed from the CONC file by editing the CONC column in the NameList file(s) to reduce the number of species that are written to, and thus the size of the CONC file.

CTM_CGRID_1: CCTM restart file

The 3-D CCTM ending concentration file (CGRID) is the CCTM restart file. Containing gas-phase species mixing ratios (ppmV) and aerosol species concentrations (µg m-3), the CGRID file includes model species concentrations at the end of each simulation period. The number and types of species contained in the output CGRID files depend on the chemical mechanism and aerosol model configurations that are selected when CCTM is compiled. This file can be used to initialize CCTM from the simulation period that the model completed. For example, if the CCTM is configure to produce daily output files, a CGRID file will be written out at the end of each simulation day.

CTM_ACONC_1: CCTM hourly average concentration file

The 3-D CCTM integral average concentration file (ACONC) contains average model species concentrations for each model hour, as opposed to instantaneous concentrations at the end of each output time

step. The species written to the ACONC file are set by the user in the CCTM run script using the variable AVG_CONC_SPCS. The model layers that are used to calculate the integral average concentration are also set in the CCTM run script using the variable ACONC_BLEV_ELEV, where BLEV corresponds to the bottom layer number and ELEV corresponds to the top layer number. An example setting for the ACONC_BLEV_ELEV variable is "1 6", which defines layers 1 through 6 as the vertical extent over which to calculate hourly average concentrations.

CTM_DRY_DEP_1: CCTM hourly cumulative dry deposition file

The 2-D CCTM dry deposition file (DRYDEP) includes cumulative hourly dry deposition fluxes (kg hectare-1) for selected model species. CCTM calculates dry deposition for all of the species listed in the dry deposition column of the FORTRAN Namelist files within the mechanism directories. The GC_*mechname*.nml file lists the gas-phase species, the AE_*mechname*.nml file lists the aerosol species, and the NR_*mechname*.nml lists the nonreactive (inert) species. Species can be removed from the dry deposition file by editing the DDEP column in the NameList file(s).

CTM_WETDEP_1: CCTM hourly cumulative wet deposition file

The 2-D CCTM wet deposition file (WETDEP) includes cumulative hourly wet deposition fluxes (kg hectare-1) for selected model species. CCTM calculates wet deposition for all of the species listed in the wet deposition column of the FORTRAN Namelist files within the mechanism directories. The GC_*mechname*.nml file lists the gas-phase species, the AE_*mechname*.nml file lists the aerosol species, and the NR_*mechname*.nml lists the nonreactive (inert) species. Species can be removed from the wet deposition file by editing the WDEP column in the NameList file(s).

CTM_VIS_1: CCTM hourly instantaneous visibility metrics

The 2-D CCTM visibility file contains hourly Mie and reconstructed visual range coefficients (km-1) and normalized extinction coefficients (deciviews).

CTM_AVIS_1: CCTM hourly average visibility metrics

The 2-D CCTM visibility file contains hourly Mie and reconstructed visual range coefficients (km-1) and normalized extinction coefficients (deciviews).

7.1 Diagnostic and Advanced CMAQ Output Files

Along with the basic outputs detailed in the previous section, CMAQ can be configured to output several auxiliary files for diagnosing model performance.

CTM_PMDIAG_1: Instantaneous hourly aerosol diagnostics file

This diagnostic file contains information on the geometric mean diameters and geometric standard deviations for the lognormal modes.

CTM_APMDIAG_1: Average hourly aerosol diagnostics file

This diagnostic file contains information on the geometric mean diameters and geometric standard deviations for the lognormal modes.

B3GTS_S: Biogenic emissions diagnostic file

This optional 2-D CCTM hourly output file contains calculated biogenic emissions in mass units. The B3GTS_S file will be produced only if in-line biogenic emissions are being calculated by CCTM and if the B3GTS_DIAG variable is turned on.

CTM_DEPV_DIAG: CCTM inline deposition diagnostics file

This 2-D CCTM file contains the deposition velocity (m/s) for each chemical species calculated for the final time step for the hour.

CTM_PT3D_DIAG: CCTM PT3D diagnostics file Add content

CTM_DUST_EMIS_1 This optional 2-D CCTM hourly output file contains calculated dust emissions in mass units. The DUST_EMIS_1 file will be produced only if in-line windblown dust emissions are being calculated by CCTM and if the CTM_DUSTEM_DIAG variable is turned on.

CTM_AOD_1 Aerosol optical depths calculated by the CCTM. This file will only be produced if CTM_AOD=Y in the CCTM run script.

CTM_IPR_[1-3] The 3-D CCTM integrated process rate files (IPR) contains hourly concentrations of selected model output species in terms of the model process that contributed to the predicted concentration at each hour. For each grid cell in the process analysis domain (which is most likely a subset of the full modeling domain), the IPR file shows the hourly change in species concentration that is due to particular source/sink processes in the model. The input file procan.inp is used to set the model species for which to capture process analysis information, and the processes to track during the process analysis.

CTM_IRR_[1-3] Process analysis output – integrated reaction rates The 3-D CCTM integrated reaction rate file (IRR) contains hourly concentrations of selected model output species in terms of the gas-phase chemistry pathways that contributed to the predicted concentration at each hour. For each grid cell in the process analysis domain (which is most likely a subset of the full modeling domain), the IRR file shows the hourly change in species concentration that is due to particular gas-phase chemistry reactions or reaction groups. The input file procan.inp is used to select the process analysis domain, the model species for which to capture process analysis information, and the chemistry reactions or groups of reactions to track during the process analysis.

FLOOR: concentration-reset diagnostics file

FLOOR files are optional output diagnostic files which list specific gridboxes/timesteps in which species with negative concentrations are reset to zero.

MEDIA_CONC: Bidirectional soil NH4+ restart file

This 2-D CCTM file contains the the soil NH4 and pH concentrations if using the bidirectional NH3 option and/or the soil, vegetation and water Hg concentrations. This file is used to initialize the next day of the model simulation.

CTM_DEPV_MOS

This 3-D CCTM file contains the deposition velocity (m s-1) for the final time step of the hour for each land use type within a grid cell.

CTM_DRY_DEP_MOS

This 3-D CCTM file contains the total deposition (kg hectare-1) for the hour for each land use type within each grid cell.

CTM_DRY_DEP_FST

This 3-D CCTM file contains the total deposition (kg hectare-1) through the stomatal pathway for the hour for each land use type within each grid cell.

CTM_DEPV_FST

This 3-D CCTM file contains the deposition velocity (m s-1) through the stomatal pathway for the final time step of the hour for each land use type within a grid cell.

CTM_VDIFF_DIAG Add content

CTM_VSED_DIAG Add content

LTNG_HOURLY Hourly 3-D lightning NO emissions calculated in-line by the CCTM.

LTNG_COL Hourly column-total lightning NO emissions calculated in-line by the CCTM.

PLAY_SRCID Add content

CTM_RJ_[1-2]: In-line photolysis output – gridded photolysis rates

The photolysis diagnostic output files (RJ) contain the photolysis rates calculated by CCTM when the in-line photolysis option is used. ### SOILOUT

Name and location of hourly soil NO emissions file; output when in-line biogenic emissions processing is activated by setting CTM_BIOGEMIS to “T” or “Y”.

CTM_SSEMIS_1: Sea salt emissions diagnostic file

This optional 2-D CCTM hourly output file contains calculated sea salt emissions. The SSEMIS file will be produced by CCTM only if the AERO5 aerosol mechanism is being used and if the CTM_SSEMDIAG variable is turned on.

CTM_WET_DEP_2: CCTM cloud diagnostics file

In CMAQ, wet deposition is calculated separately for resolved (grid-scale) clouds and for convective (subgrid) clouds. The WETDEP1 file contains the total wet deposition, i.e., the sum of both resolved-scale and subgrid-scale deposition. The WETDEP2 file contains only subgrid-scale deposition, plus some cloud diagnostic variables. The 2-D CCTM wet deposition file (WETDEP2) includes cumulative hourly wet deposition fluxes (kg hectare-1) for selected model species. CCTM calculates wet deposition for all of the species listed in the wet deposition column of the FORTRAN Namelist files within the mechanism directories. The GC_*mechname*.nml file lists the gas-phase species, the AE_*mechname*.nml file lists the aerosol species, and the NR_*mechname*.nml lists the nonreactive (inert) species. Species can be removed from the wet deposition file by editing the WDEP column in the NameList file(s).

8 Defining Grids, Layers, and Chemistry

This chapter describes how to define new horizontal grids, vertical layers, and chemical mechanisms in CMAQ. These specifications apply to multiple programs in the CMAQ modeling system, including

ICON, BCON, JPROC, and CCTM. When configuring new simulations, users must define the location, extent, and structure of the horizontal and vertical grids, and the chemical mechanism for representing pollutant chemical transformations. CMAQ contains several default options for these parameters that can be used as templates for setting up new configurations. Before deciding to create definitions for new grids and mechanisms, check to see whether the existing options are sufficient for your model simulation. If a predefined choice is not appropriate, then follow the steps described in this section to create a new definition.

Once you have configured a simulation that is suitable for your purposes in terms of the horizontal grid, vertical layers, and chemical mechanism, proceed to Chapter 10 to learn how to develop new model executables for running a CMAQ simulation.

8.1 Grids and coordinate systems

CMAQ is a three-dimensional Eulerian air quality model. The *domain* of a model run (the extent of its area of interest) is divided into three-dimensional cells (or *voxels*), the boundaries of which (the *grid* of the domain) must be rigorously and consistently defined for all functional components of the model (e.g., chemistry, emissions, meteorology). Mathematical algorithms describing atmospheric transport and air/surface exchange govern the flow of material into and out of each grid cell. Mathematical algorithms describing chemical reactions and aerosol dynamics govern the production and loss of material contained in each grid cell.

Horizontal (or *2D*) and vertical components of a model run's grid are treated differently. The horizontal grid specification (setting the x and y dimensions) must be *regular*: the horizontal projection of each grid cell (sometimes referred to as a *pixel*) has the same resolution, and the boundaries of each pixel are time-invariant. By contrast, the vertical grid specification (setting the z dimension) need not be regular; it can vary in space and time.

After determining the horizontal and vertical extent of the domain of interest, a meteorological model must be run for a horizontal domain slightly larger than the CMAQ domain. A larger meteorology domain is necessary for distinguishing the meteorological boundary conditions from the CMAQ boundary conditions.

8.1.1 Supported CMAQ Coordinate Systems

Specifications for CMAQ and MCIP grids are governed by I/O API grid conventions. The choice of horizontal coordinate system, or map projection, for CMAQ is governed by the input emissions inventories and meteorological model fields, which must agree. WRF/ARW support the Lambert Conformal, Polar Stereographic, and Mercator projections, which can be directly passed to CMAQ.

8.1.2 Horizontal Grids

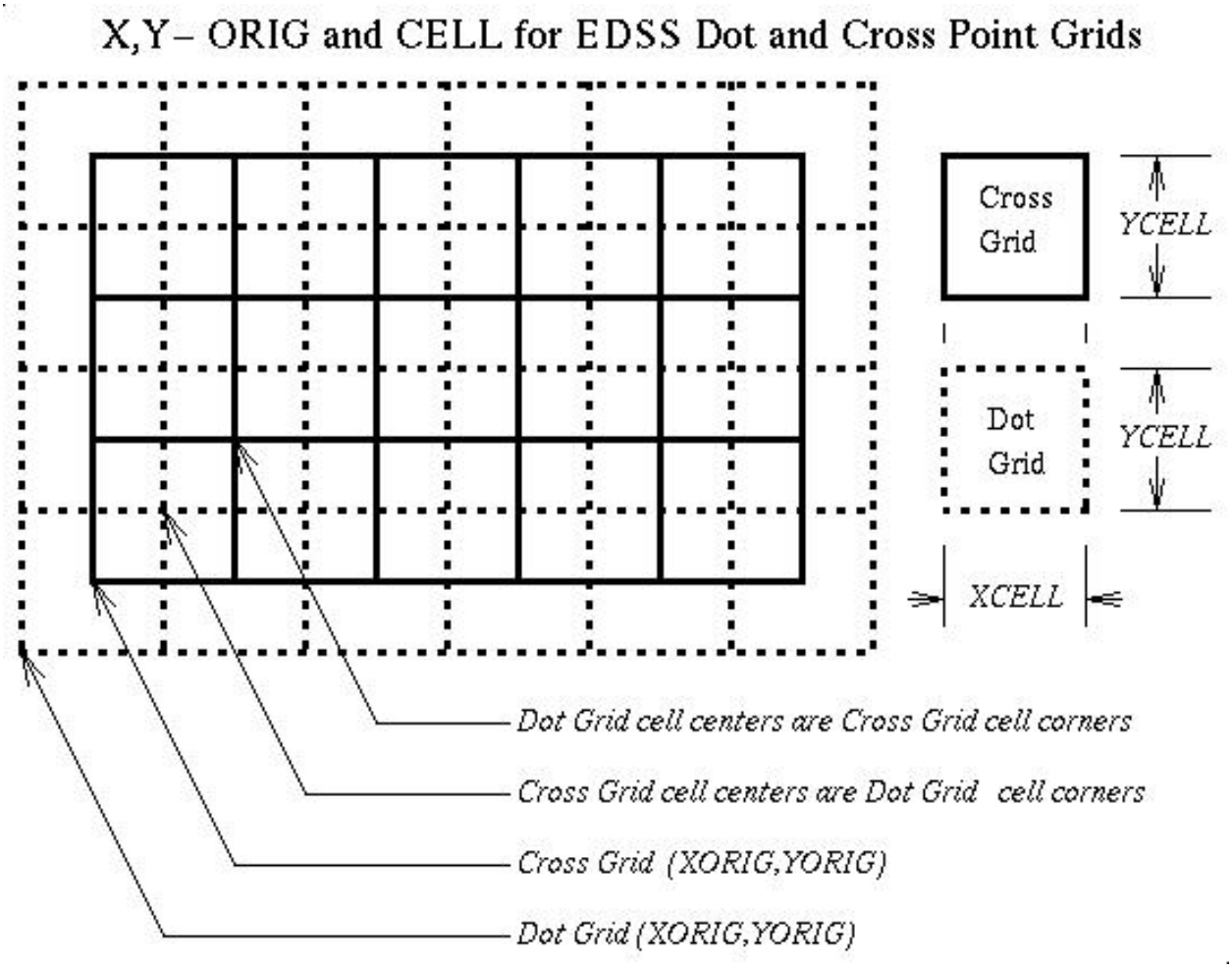
Available horizontal grids for a given CMAQ run are defined at runtime by setting the GRIDDESC and GRID_NAME environment variables to point to an existing grid definition file and to one of the grids

defined in the file, respectively. Horizontal grids are defined by the grid definition file (GRIDDESC), which can be edited by the user (more below).

The extent of the horizontal grid used in CMAQ is limited by the size of the domain of the input meteorology. MCIP and the I/O API Tools can be used to *window* subsets of meteorology data. Choosing the appropriate horizontal grid scale and extent for a CCTM run is largely dependent on the issues to be addressed by the modeling. However, practical consideration should also be paid to the relationship between grid size, output file size, and execution times, i.e., output data volume and run times increase as the number of horizontal grid cells increase.

8.1.2.1 CMAQ horizontal grid conventions

Grid conventions are specified (at length) by the I/O API. In summary, users should be aware that CMAQ uses both "cross-point" and "dot-point" grids.



"Cross-point" is often abbreviated *CRO*, as in `GRID_CRO_2D`. "Dot-point" is often abbreviated *DOT*, as in `MET_DOT_3D`. Similarly, the user should be aware of the grid's projection units. Usually meters, except when using lat-lon coordinate systems.

The terms associated with I/O API grid definitions are listed in Table 9-1.

Table 9-1. I/O API Grid Type Terms

Term	Definition
origin	lower left corner of the cell at column=row=1
X_ORIG	X coordinate of the grid origin (in projection units)
Y_ORIG	Y coordinate of the grid origin (in projection units)
X_CELL	horizontal resolution parallel to the X coordinate axis (in projection units)
Y_CELL	horizontal resolution parallel to the Y coordinate axis (in projection units)
NCOLS	number of grid columns, dimensionality in the X direction
NROWS	number of grid rows, dimensionality in the Y direction

CMAQ is distributed with a GRIDDESC file that contains a definition for a 12-km grid covering California that uses a Lambert Conformal Conic projection. The definition of this grid is below.

- Coordinate: Lambert Conformal
- Latitude 0: 40.0
- Longitude 0: -97.0
- Standard Parallel 1: 33.0
- Standard Parallel 2: 45.0
- X origin = -2,376,000
- Y origin = -792,000
- Rows: 100
- Columns: 72
- dX = 12,000
- dY = 12,000
- Layers = 35

8.1.2.2 Creating or modifying horizontal grids

Creating a grid in CMAQ involves simply adding a few lines of text to the GRIDDESC file. Using a combination of the I/O API GRIDDESC file format documentation and existing grid definitions as examples, new grids can be defined for CMAQ by adding a coordinate and grid description to the GRIDDESC file. Set the GRID_NAME environment variable in the CMAQ run scripts to point to the name of the new grid.

The most common situation for creating a new CMAQ grid definition is encountered when using meteorology and/or emissions data that have not yet been modeled with CMAQ. WRF□ARW outputs can be run through MCIP to generate a GRIDDESC file that can be input directly to both CMAQ and SMOKE. MCIP includes a set of variables for trimming boundary cells from the WRF output, windowing the WRF domain, and setting the reference latitude of the projection. A description of the MCIP variables is provided in Chapter 7. The MCIP variables for defining horizontal grids are provided below.

- **BTRIM**: Sets the number of boundary points to remove on each of the four horizontal sides of the MCIP domain. Setting BTRIM = 0 will specify the maximum extent of the input meteorology domain. To remove the WRF□ARW lateral boundaries, set BTRIM = 5 (recommended).

For windowing a subset domain of the input meteorology, set BTRIM = -1; this setting causes BTRIM to be replaced by the information provided by X0, Y0, NCOLS, and NROWS (see below).

- **X0**: The *x*-coordinate of the lower-left corner of the full MCIP cross-point domain (including the MCIP lateral boundary) based on the input WRF□ARW domain. X0 refers to the number of grid cells in the east-west direction from the origin of the WRF domain to the origin of the MCIP cross-point domain. This setting is only used when BTRIM = -1.
- **Y0**: The *y*-coordinate of the lower-left corner of the full MCIP cross-point domain (including the MCIP lateral boundary) based on the input WRF□ARW domain. X0 refers to the number of grid cells in the north-south direction from the origin of the WRF domain to the origin of the MCIP cross-point domain. This setting is only used when BTRIM = -1.
- **NCOLS**: Number of columns in the output MCIP domain (excluding MCIP lateral boundaries). This setting is only used when BTRIM = -1.
- **NROWS**: Number of rows in the output MCIP domain (excluding MCIP lateral boundaries). This setting is only used when BTRIM = -1.
- **WRF_LC_REF_LAT**: WRF Lambert Conformal reference latitude. Use this setting to force the reference latitude in the output MCIP data. If not set, MCIP will use the average of the two true latitudes.

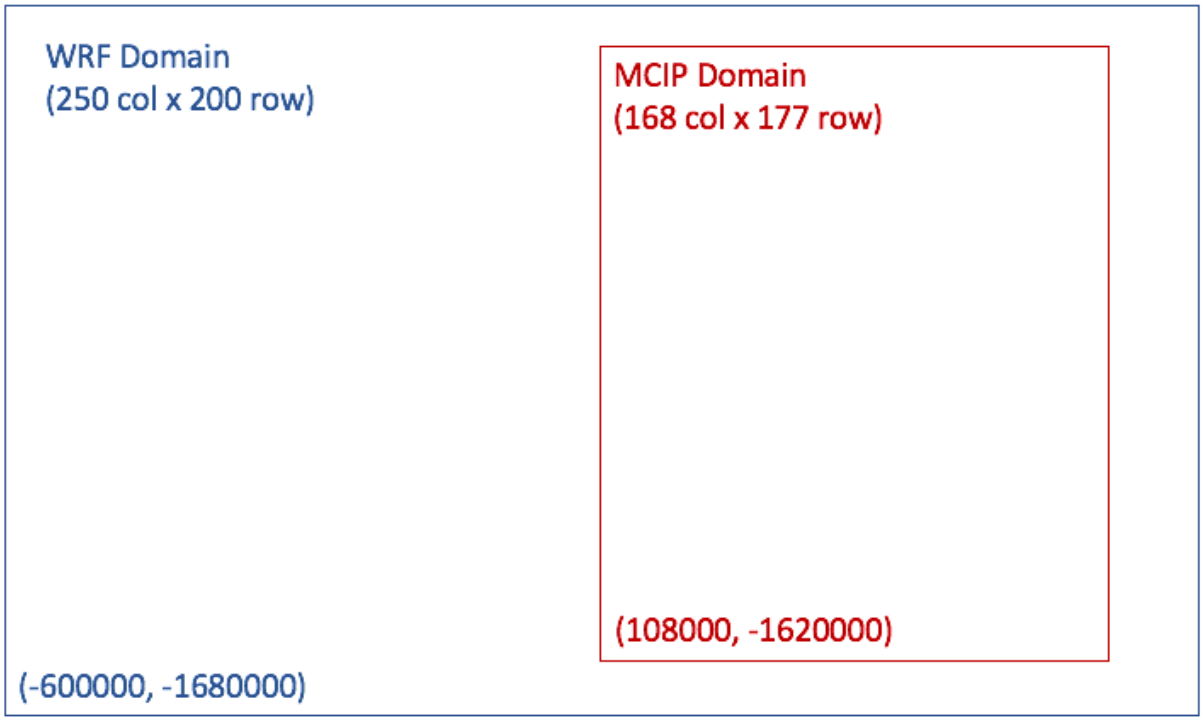
8.1.2.3 Example BTRIM Calculation

Figure 9-2 shows an example of how the BTRIM calculation works for windowing WRF data. The 12-km grid resolution WRF domain has a lower left corner that is offset from the projection center by 600,000 m West and 1,680,000 m South. The CMAQ/MCIP output domain has an offset (lower left corner) that is 108,000 m East and 1,620,000 m South of the projection center.

The MCIP variables X0 and Y0 set the number of grid cells to "trim" from the WRF to domain to get to the MCIP domain lower left corner. Adding "1" in each of these calculations accounts for the addition of the MCIP lateral boundary.

```
`X0 = |(WRF X origin – MCIP X origin)|/(Grid Resolution) + 1`
`X0 = |(–600,000 – 108,000)|/12,000 + 1`
`X0 = 60`

`Y0 = |(WRF Y origin – MCIP Y origin)|/(Grid Resolution) + 1`
`Y0 = |(–1,680,000 – –1,620,000)|/12,000 + 1`
`Y0 = 6`
```



8.1.2.4 Further information on horizontal grids

- If the meteorology data have already been processed by MCIP and the GRIDDESC file is missing, the grid definition of the input meteorology (and emissions) can be determined by using the netCDF utility *ncdump* to view the header of one of the I/O API files and then use that information to manually create a GRIDDESC file.
- Horizontal grid dimensions should be no smaller than 30 rows and 30 columns.
- External boundary thickness should be set to "1".
- A CMAQ grid should be smaller than its parent meteorology grid by at least four grid cells on a side, and preferably by six.
- Horizontal grid spacing for the parent meteorology grid often has a 3:1 ratio, although other ratios have been employed.

8.2 CMAQ Vertical Layers

The vertical structure of CMAQ is inherited from the model used to prepare the meteorological information. WRF-ARW uses a sigma coordinate that is based upon surface pressure, not sea level pressure, and a pressure at the top boundary (e.g., 100 hecto-Pascals). The sigma coordinate is terrain following. Because WRF-ARW is a nonhydrostatic model, the vertical coordinate is time varying.

8.2.1 Vertical layer resolution

Resolving the surface boundary layer requires high resolution (i.e., shallow vertical layers) near the surface for meteorological simulations. To determine mass exchange between the boundary layer and free troposphere, high resolution near the boundary layer top is also preferable. In addition, different cloud parameterizations may perform differently depending on the layering structure. Layer definitions should be appropriate for the topographic features of the simulation domain. Aerodynamic resistance, which influences dry deposition velocities, is a function of layer thickness and the boundary layer stability. For emissions processing, the layer thickness affects the plume rise from major stacks. The vertical extent of the surface-based emission effects is determined by the thickness of the lowest model layer for CCTM. For consistency, CCTM should use the same vertical resolution as the meteorological model used to prepare the input data.

8.2.2 Further information on vertical layers

- CMAQ redefines the vertical coordinates to monotonically increase with height, a capability necessary to handle a generalized coordinate system.
- Although MCIP may be used to reduce the number of vertical layers by collapsing layers, this is ***not recommended,*** as dynamical inconsistencies can develop and lead to misleading results. This is particularly true when cloud processes are important.
- Increasing the number of vertical layers increases the CPU time and the computational complexity.
- Computational limits arise from the Courant number limitation of vertical advection and diffusion processes. When using K-theory, a very shallow layer definition increases CPU time tremendously under the convective conditions.

8.2.3 References for grid and vertical coordinate system topics

- On The Definition of Horizontal and Vertical Grids and Coordinates for Models-3
- Chapter 12 (MCIP) of the 1999 Models-3/CMAQ Science document
- Otte and Pleim 2009 (in GMD) on MCIP

8.3 CMAQ Chemical Mechanisms

The CMAQ modeling system accounts for chemistry in three phases: a gas phase, aerosols (solid or liquid), and an aqueous phase. Refer to the release notes to find the gas□phase chemistry mechanisms available in each version of CMAQ. Several variations of the base gas-phase mechanisms, with and without chlorine, mercury, and toxic species chemistry, are distributed with CMAQ. The modularity of CMAQ makes it possible to create or modify the gas-phase chemical mechanism.

Gas-phase chemical mechanisms are defined in CMAQ through Fortran source files. Located in subdirectories of the $CMAQ_MODEL/CCTM/src/MECHS directory (each corresponding to a mechanism name), these files define the source, reaction parameters, and atmospheric processes (e.g., diffusion, deposition, advection) of the various mechanism species. The species definitions for each mechanism

are contained in namelist files that are read in during execution of the CMAQ programs. The CMAQ mechanism configuration is more similar to the science module configuration than to the horizontal grid or vertical layer configuration in that the mechanism is defined at compilation, resulting in executables that are hard-wired to a specific gas-phase mechanism. To change chemical mechanisms between simulations, a new executable that includes the desired mechanism configuration must be compiled.

8.3.1 Using predefined chemical mechanisms

To select a predefined mechanism configuration in CMAQ, set the *Mechanism* variable in the build scripts to the name of one of the mechanism directories located under $CMAQ_MODEL/CCTM/src/MECHS. Refer to the CMAQv5.2 Photochemical Mechanisms release notes for the list of mechanisms available in CMAQv5.2.

8.3.2 Creating or modifying chemical mechanisms

Creating or modifying mechanisms in CMAQ requires the use of the CMAQ chemical mechanism compiler, CHEMMECH, to produce the required Fortran source (F90) and namelist files. CHEMMECH translates an ASCII mechanism listing to the F90 and namelist files required by CMAQ. Like all of the CMAQ preprocessors, CHEMMECH is a Fortran program that must be compiled prior to use. Distributed with a Makefile for compilation and run scripts for execution, CHEMMECH reads a mechanism definition (mech.def) file and outputs the mechanism F90 and namelist files. See Chapter 7 for a description of CHEMMECH.

To modify an existing mechanism, copy the mech.def file that is contained in one of the existing mechanism directories to a new directory and modify the mech.def file accordingly. Provide this modified mechanism definition file to CHEMMECH as input to produce the mechanism F90 and namleist files needed to compile CMAQ.

To invoke this new mechanism in CMAQ, set the *Mechanism* variable in the CMAQ build scripts to the name of the new mechanism directory and compile new executables.

To create a new mechanism for CMAQ, follow a procedure similar to the above for modifying mechanisms. Use an existing mech.def file as a template to format the new mechanism for inclusion in CMAQ. After formatting the mechanism in the form of the mech.def file, provide this file as an input to CHEMMECH to create the required mechanism input files for CMAQ. Move the resulting mechanism files to a new directory under $CMAQ_MODEL/CCTM/src/MECHS. To invoke this new mechanism, set the *Mechanism* variable in the CMAQ build scripts to the name of the new mechanism directory and compile new executables.

8.3.3 Using species namelist files

The species namelist files define the parameters of the gas, aerosol, non-reactive, and tracer species simulated by the model. The CMAQ programs read the namelist files during execution to define the sources and processes that impact the simulated concentrations of each of the model output species.

The namelist files can be used to apply uniform scaling factors by model species for major model processes. For example, emissions of NO can be reduced by 50% across the board by applying a factor of 0.5 to the emissions scalar column of the gas-phase species namelist file. Similarly, the boundary conditions of O3 can be increased by 50% by applying a factor of 1.5 to the boundary conditions scalar column of the gas-phase species namelist file.

See Chapter 8 for a description of the format of the namelist file.

When mechanisms are modified or created in CMAQ, new namelist files must be created that include the new species in the mechanism. As described above, the program CHEMMECH will generate namelist files from a mech.def mechanism definition file. Alternatively, existing namelist files can be used as templates to guide the manual creation of new files.

8.3.4 Further information on chemical mechanisms

- The same chemical mechanism must be used for CCTM and all of the mechanism-dependent input processors that are part of the CMAQ system.
- The Euler Backward Iterative (EBI) chemistry solver is mechanism-dependent. If a chemical mechanism is modified, then new EBI solver source code must be generated based on the mechanism definition. The CMAQ utility program CREATE_EBI reads the output from CHEMMECH to generate new EBI solver source code.
- The Rosenbrock and SMVGEAR solvers are mechanism-independent choices of chemistry solvers for the CCTM.
- When adding new species to CMAQ, it is important to check that the sources of these new species into the modeling domain are accounted for correctly in the mechanism definition files. If species are added to the domain through the emissions files, the namelist files that define the mechanism species must contain these new species.

9 Developing New CMAQ Simulations

For application users of CMAQ, the CMAQ model builder Bldmake is used only at the beginning of a simulation to compile executables for a specific science configuration. Since the horizontal grid and vertical layer structure are defined dynamically at execution of the model, there is typically no need to recompile the programs when changing these parameters. Compilation is required only when either developing CMAQ simulations for the first time or when the chemistry/science configuration within the model is changed.

A model developer would use Bldmake to check out a working version of the CMAQ source code and create a Makefile to facilitate the interchange of science components within a model, to modify reaction details within an existing chemistry mechanism, or to experiment with source code modifications.

The purpose of this chapter is to demonstrate how to build the executables for the CMAQ programs beyond running the installation test case. Before proceeding with this chapter, review Chapter 5 for an overview of the system requirements for CMAQ. In general, there are three major steps in compiling CMAQ executables:

1. Install third-party libraries (netCDF, I/O API, MPI)
2. Compile Bldmake
3. Configure and build the executables for the various CMAQ programs.

All compilations of libraries and CMAQ must be done with the *same compilers and settings*. The details of these three major steps with respect to creating new CMAQ simulations are covered in this chapter.

9.1 General Introduction to Model Building

Before using CMAQ for operational modeling in a new computing environment, it is recommended that the model be benchmarked using the test dataset that is distributed with the model. Chapter 5 describes how to install and benchmark CMAQ on a Linux system. After benchmarking CMAQ, it can be configured for other simulations. The same steps that are required to build the model for the test case apply to building it for new simulations. However, not all of the steps need to be repeated for a new model configuration unless new code becomes available or bug fixes are identified. In particular, it is not necessary to rebuild any of the libraries that CMAQ uses once working versions are built on a user's system. A single installation of the libraries (netCDF, I/O API, MPI) can be linked to for multiple configurations and applications of the model. Likewise, the CMAQ model builder, Bldmake, only needs to be compiled once and used for all applications of the model.

Except for MCIP, all of the CMAQ programs need to be recompiled when the chemistry mechanism or science configuration of the model change. If the science configuration does not change between applications, the CMAQ executables can be reused for different applications on the same Linux system. MCIP needs to be compiled only once on a user's Linux system and the executables may be reused for all applications of the model, unless new source code, including libraries, become available.

9.2 Configuring New Simulations

The reason for modeling a particular time period evolves from a research question, such as determining why a severe air pollution episode happened, or studying the dominant sources of visibility degradation in a specific geographic region. Once a modeling episode is selected, several steps must be taken before CMAQ can be run.

1. Run and evaluate the WRF meteorology model for the episode of interest
2. Convert the WRF data for input to SMOKE and CMAQ using MCIP
3. Run and evaluate SMOKE to process emissions input data for CMAQ
4. Prepare biogenic emissions input data, either for inline or offline processing
5. Prepare initial and boundary conditions for CMAQ
6. Run and evaluate the CCTM.

The horizontal model grid, vertical layer structure, and model time periods must be consistent across the meteorology data, emissions data, and CMAQ.

Configuring CMAQ for new simulations involves defining the model grid, vertical layers, time periods, initial and boundary conditions, input/output file locations, and science options of the model. The following sections cover these topics in the context of setting up a new CMAQ simulation.

9.2.1 Defining a new horizontal grid

The grid-dependent CMAQ programs (CCTM, ICON, BCON) use a GRIDDESC grid description file to define the map projection and horizontal grid for a simulation. The GRIDDESC file is either output by MCIP or it can be created manually using a text editor. The CMAQ run scripts for the grid-dependent CMAQ programs must refer to a GRIDDESC file that contains the definition of the model grid to be simulated. The name of the grid for the current application must be specified in the CMAQ run script because a single GRIDDESC file can contain multiple grid definitions. Additional information about setting up horizontal grids in CMAQ is contained in Chapter 9.

The following error from any of the CMAQ programs can occur if the name and/or location of the GRIDDESC file and/or the name of the horizontal grid are incorrect in the run script:

```
Failure defining horizontal domain
```

For CCTM, which uses both a GRIDDESC file and gridded input data (emissions, meteorology, ICs/BCs), the grid defined in the GRIDDESC file must be consistent across all of the gridded input files, or the simulation will fail.

To configure a new CMAQ simulation, the following steps must be taken to set up the horizontal model grid:

- Produce emissions and meteorology data on a consistent horizontal model grid to be modeled with CCTM
- Create a GRIDDESC file that defines the horizontal model grid
- Generate ICs and BCs on the horizontal model grid; for nested simulations, generate BCs from a parent grid
- Configure the CCTM script to use the GRIDDESC file and input data on the desired horizontal model grid

9.2.2 Defining a new vertical layer structure

The CMAQ programs that produce 3-D output (CCTM, ICON, BCON) use a MET_CRO_3D file to define a vertical layer structure. The MET_CRO_3D file is output from MCIP and takes the vertical layer structure from the input meteorology. The vertical layer structure must be consistent across all of the CMAQ programs used to complete a CCTM simulation. New vertical layer structures for CMAQ simulations are configured with MCIP when processing raw meteorological model data.

9.2.3 Setting a new episode time period

The temporal extent of a CMAQ simulation is limited by the availability of input meteorology and emission data. Similar to the horizontal grid and vertical layer structures, the time period to model with

CMAQ must be considered when designing the meteorological modeling simulation used to produce CMAQ input data.

The model output time step and run length of each CMAQ simulation are flexible and can be configured in the run script for the applicable CMAQ program. For CCTM, it is possible to have output files with a different number of time steps than the corresponding meteorology or emission input data. For example, it is common to have input meteorology files in 4- or 5-day intervals, input emission files in 1- or 2-day intervals, and CCTM output files in 1-day intervals. The CMAQ scripts allow the user to configure the model to output data with any number of time steps. The number of CCTM output time steps does not need to correspond with the input meteorology and emissions output time steps. To keep CMAQ output file sizes manageable, CMAQ output is typically stored in 1 day (24 hour) blocks.

To configure a new CMAQ simulation, the follow steps must be taken to set up the modeling time period:

- Produce emissions and meteorology data for the time period to be modeled with CMAQ. When deciding upon a modeling period, it is necessary to have a "spin-up" interval prior to the beginning of the initial time of interest. The spin-up period is a sequence of days at the beginning of an air quality simulation that are not used in the analysis of the modeling results. These days are simulated to minimize the impacts of the initial conditions on the CCTM modeling results. Spin-up periods vary in length depending on the size of the modeling domain, the magnitude of the emissions sources within the modeling domain, and the pollutants being studied. As a general rule of thumb for regional modeling, a period of at least 10 days should be considered for a spin-up period.
- Generate ICs for the first time step of the CCTM simulation; if running multiple days in sequence, configure the CCTM run script to use the ICs from ICON for the first model day and to initialize the subsequent days with the previous day's CCTM output
- Generate either time-independent BCs for the grid to be modeled, or for a nested simulation generate temporally resolved BCs for the model time period from CCTM outputs for the parent grid
- Configure the CCTM run script to loop through the days to be modeled, using the correct input files for each model day and writing output files with the desired number of time steps

9.2.4 Initial and boundary conditions

After preparing the meteorology and emissions input data files and determining the model grid and model time periods, the next step for setting up a new CMAQ simulation is creating initial and boundary conditions (ICs and BCs) for CCTM. The ICON processor provides initial chemical fields of individual species concentrations for a specific modeling domain. BCON provides concentrations of individual chemical species for the grid cells surrounding the modeling domain. ICON and BCON both require two inputs: concentration values for the chemical species needed in the simulation, and a predefined chemical mechanism.

As described in Chapter 5, there are two types of input concentrations for ICON and BCON: either (1) tabulated tropospheric vertical profiles or (2) three-dimensional fields from a previous CMAQ or larger-scale CTM simulation, such as GEOS-Chem(2009) or MOZART (2009). The input file type

to use for ICON and BCON depends on whether the simulation is a restart of a previous run and/or whether the simulation is a nest of a larger parent grid. The same chemical mechanism must be selected for ICON, BCON, and CCTM. Both ICON and BCON assume that the input species concentrations are for the selected mechanism. Refer to Chapter 5 for information on how to configure ICON and BCON for the two input file types.

Standard operation of CMAQ uses boundary conditions that are extracted from a global model, such as GEOS-Chem or MOZART for the outermost modeling domain. Nested grids use temporally resolved boundary conditions extracted from the results of a parent-grid CCTM simulation. The initial conditions produced by ICON are time independent for the first spin-up day. If the initial conditions were generated from vertical profile data then they will also be spatially uniform, meaning that for a given layer they will contain the same concentration in every grid cell. The remaining days of a simulation should use spatially heterogeneous output from the previous day's CCTM simulation to initialize each day. See Figure 10-1 for a schematic of the initial and boundary conditions used for a CCTM simulation with a two-day spin-up followed by a two-day period of interest. In this example, each of the days was run separately.

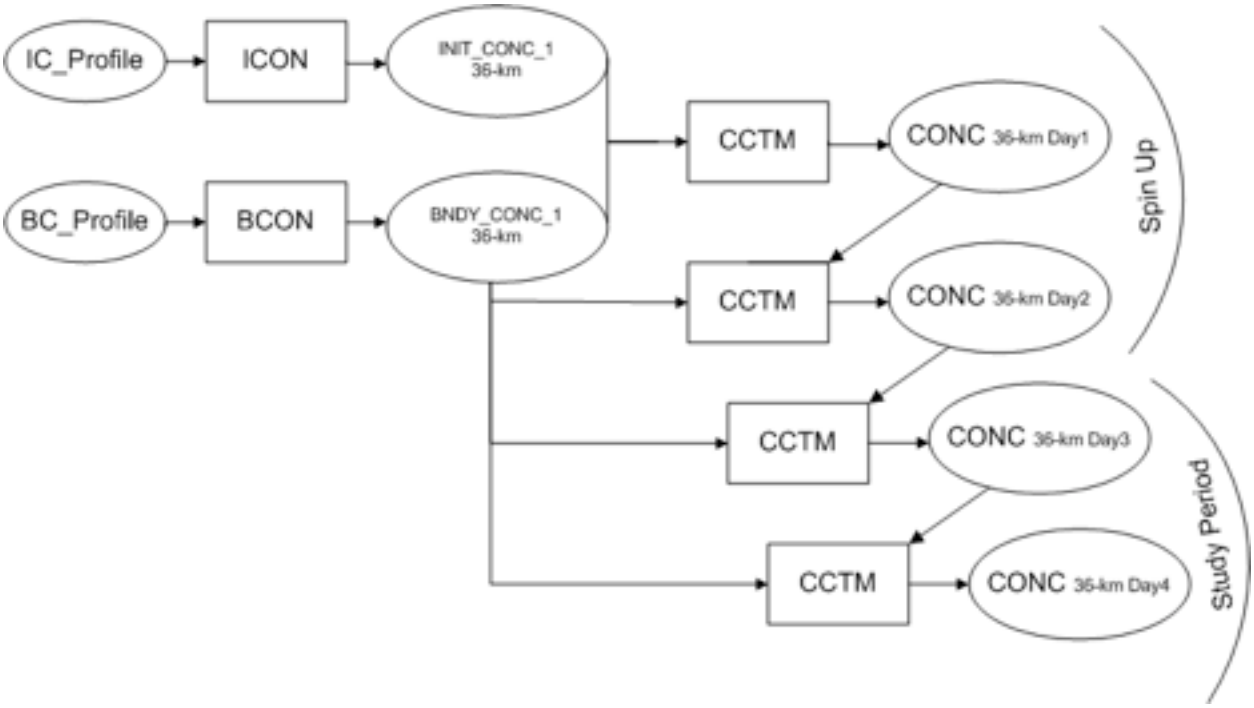


Figure 10-1. 36-km four-day modeling period IC/BC schematic

When using a nested-grid configuration, the fine-grid-resolution grids can use time-varying boundary conditions generated by BCON with input from time-varying output concentrations from the coarse-grid CCTM simulation. The initial conditions for start-up of the fine grid are also generated using concentrations from the coarse grid. Subsequent runs in the period of interest use the last hour of the concentration file generated from the previous day's run. See Figure 10-2 for an example of a one-way, nested simulation. This example uses initial and boundary conditions from the coarser 36-km grid simulation to perform the nested, finer-grid simulation for the same two-day period of interest.

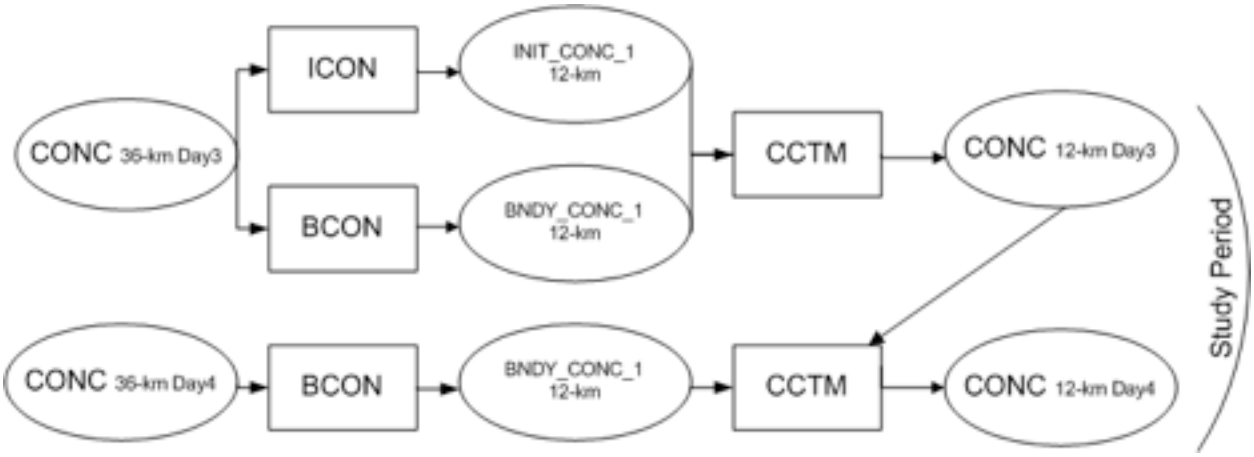


Figure 10-2. 12-km nested two-day modeling period IC/BC schematic

9.2.5 Input/output file names and locations

Configuring the CMAQ run scripts for a new simulation involves creating an application identifier to use in the name of the CMAQ outputs and to specify the correct input and output file names and locations on the system where CMAQ will be run. Application identifiers are selected to uniquely identify the output files with respect to the simulation. For example if you are running a base simulation for the year 2007, a simulation identifier, or APPL setting in CMAQ terms, could be Base07. For time-dependent output files, a date identifier is commonly added to the file name to identify the time period covered by the file. Following the previous example, if you configured a simulation to run on January 5, 2007 (Julian date 2007005), an example file name for a CMAQ version 5.2 CCTM concentration file would be CCTMv52.CONC.Base07.2007005.ncf. Additional identifying information, such as the chemistry mechanism name, versioning identifiers for the input meteorology and emissions, and the operating system version (i.e. Linux2_x86_64) are also commonly added to CMAQ output file names.

Before starting a new CMAQ simulation, the user needs to confirm the existence of the input files for all of the CMAQ programs, and to check the names of the output files and directory locations. A common error that occurs with all of the CMAQ programs is an execution failure caused by missing or misnamed input files.

9.2.6 Science option configuration

The CMAQ scripts as they are distributed use a default science option configuration. While this default configuration is acceptable for all applications of the model, it is not the only configuration that can be used with CMAQ. The ICON and BCON science options depend on the nature of the input data, the chemical mechanism chosen, and whether one-way nests are being used. The CCTM science configuration presents the largest number of combinations of different transport algorithms, chemistry options, and special features, such as process analysis. To see a choice of all the different options available for configuring CCTM, refer to Chapter 7.

Not all of the combinations of the various CCTM configuration options have been tested. It is possible that some combinations of different science modules will not build a working executable. Generally,

here are few rules to follow when configuring CCTM:

- For configurations that use the Yamartino model driver, the Yamartino options must be used for the initialization module, the advection routines, and the concentration adjustment scheme.
- Ensure that there is consistency in the selection of the chemistry mechanism and the aerosol module; for example, the *aero6* aerosol module cannot be used with a chemistry mechanism that is tagged "ae5".
- The EBI chemistry solver is mechanism-dependent and must be consistent with the chemistry mechanism; for example, the EBI solver for CB05 cannot be used with a SAPRC□07 chemistry mechanism.

The availability of different science options in CMAQ creates the flexibility to select a configuration that optimizes the model performance for different applications. Through testing the various science configurations with respect to model accuracy and speed, the CMAQ user can find the combination of science options that gives the best performance for a particular modeling study.

9.3 References for Chapter 10: New Simulations

GEOS-CHEM (2009) http://wiki.seas.harvard.edu/geos-chem/index.php/Main_Page

MOZART (2009), http://www.mpimet.mpg.de/en/wissenschaft/modelle/mozart.html

10 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.

10.1 Source Code Management

10.1.1 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.

10.1.2 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.

10.1.3 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 on the machine where you would like to obtain a copy of the code 4. `git clone –b 5.2 https://github.com/<your github name>/CMAQ.git CMAQv5.2` - Get a clone or copy of the 5.2 branch of the CMAQ repository from your github site.

5. This will place a copy of the files from the 5.2 Branch into the CMAQv5.2 directory 6. `cd CMAQv5.2` go into the CMAQv5.2 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_update` To copy the 5.2 branch into a new branch called 5.2_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_update use the command: `git push` 14. If you get a message that the push was rejected similar to the following: `! [rejected] 5.2_update -> 5.2_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.

10.2 Guidelines for Developing New CMAQ Source Code

10.2.1 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.

10.2.2 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.

10.2.3 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.

10.2.4 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 $X \times Y \leq N$, 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 $X \times Y \times Z \leq N$, 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.

10.2.5 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.

10.2.6 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 accumlator
      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 vsed 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 acros
               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(
                           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_FS
                              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 tota
                     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

10.3 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 .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.

10.4 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. (NEED TO UPDATE)

```
#!/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
# ================================================================

#> Source the config_cmaq.csh file to set the run environment
 source ../../config_cmaq.csh

#> Check that CMAQ_DATA is set:
 if ( ! -e $CMAQ_DATA ) then
    echo "   $CMAQ_DATA path does not exist"
    exit 1
    endif
 echo " "; echo " Input data path, CMAQ_DATA set to $CMAQ_DATA"; echo " "

 set PROC     = mpi #> serial or mpi
 set APPL     = v52b
 set CFG      = CMAQ52b_${PROC}
```

```
 set MECH     = cb05e51_ae6_aq
 set EXEC     = CCTM_${APPL}_$EXEC_ID

#> horizontal domain decomposition
if ( $PROC == serial ) then
   setenv NPCOL_NPROW "1 1"; set NPROCS   = 1 # single processor setting
else
   setenv NPCOL_NPROW "4 4"; set NPROCS   = 16
endif

#> Set the working directory
 set BASE     = $CMAQ_HOME/CCTM/scripts
 set BLD      = ${BASE}/BLD_CCTM_${APPL}

 cd $BASE; date; cat $BLD/cfg.$EXEC; echo "    "; set echo

#> timestep run parameters

 set STDATE   = 2011182       # beginning date
 set STTIME   = 000000        # beginning GMT time (HHMMSS)
 set NSTEPS   = 240000        # time duration (HHMMSS) for this run
 set TSTEP    = 010000        # output time step interval (HHMMSS)
 set YEAR  = 2011
 set YR    = 11
 set MONTH = 07
 set DAY   = 01
 set YMD   = ${YEAR}${MONTH}${DAY}

# ========================================================================
# CCTM Configuration Options
# ========================================================================
#setenv LOGFILE $BASE/$APPL.log  #> log file name; uncomment to write standard output to a log,

setenv GRID_NAME 12CalnexBench         #> check GRIDDESC file for GRID_NAME options
setenv GRIDDESC $CMAQ_DATA/mcip/GRIDDESC  #> grid description file

#setenv CONC_SPCS "O3 NO ANO3I ANO3J NO2 FORM ISOP ANH4J ASO4I ASO4J" #> CONC file species; com
es to CONC
#setenv CONC_BLEV_ELEV " 1 4" #> CONC file layer range; comment to write all layers to CONC

setenv AVG_CONC_SPCS "O3 NO CO NO2 ASO4I ASO4J NH3" #> ACONC file species; comment or set to "A
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 N         #> use inline windblown dust emissions [ default: Y ]
setenv CTM_ERODE_AGLAND Y    #> use agricultural activity for windblown dust [ default: N ]; ig
setenv CTM_WBDUST_BELD BELD3 #> landuse database for identifying dust source regions [ default:
setenv CTM_LTNG_NO Y         #> turn on lightning NOx [ default: N ]
setenv CTM_WVEL Y            #> save derived vertical velocity component to conc file [ default
setenv KZMIN Y               #> use Min Kz option in edyintb [ default: Y ], otherwise revert t
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:
setenv CTM_ABFLUX Y          #> ammonia bi-directional flux for in-line deposition velocities [
setenv CTM_HGBIDI Y          #> mercury bi-directional flux for in-line deposition velocities [
setenv CTM_SFC_HONO Y        #> surface HONO interaction [ default: Y ]; ignore if CTM_ILDEPV =
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 [ def

#> 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

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

#> MPI Optimization Flags
setenv MPI_SM_POOL 16000     #> increase shared memory pool in case many MPI_SEND headers
setenv MP_EAGER_LIMIT 65536  #> set MPI message passing buffer size to max
setenv MP_SINGLE_THREAD yes  #> optimizate for single threaded applications [ default: no ]
setenv MP_STDOUTMODE ordered #> order output by the processor ID
setenv MP_LABELIO yes        #> label output by processor ID [ default: no ]
setenv MP_SHARED_MEMORY yes  #> force use of shared memory for tasks on same node [ default: no
setenv MP_ADAPTER_USE shared #> share the MP adapter with other jobs
setenv MP_CPU_USE multiple   #> share the node with multiple users/jobs
setenv MP_CSS_INTERRUPT yes  #> specify whether arriving packets generate interrupts [ default:

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

# ======================================================================
#> Input/Output Directories
# ======================================================================

set ICpath    = $CMAQ_DATA/icon      #> initial conditions input directory
set BCpath    = $CMAQ_DATA/bcon      #> boundary conditions input directory
set EMISpath  = $CMAQ_DATA/emis      #> surface emissions input directory
set IN_PTpath = $CMAQ_DATA/emis      #> elevated emissions input directory (in-line point only)
set IN_LTpath = $CMAQ_DATA/lightning #> lightning NOx input directory
set METpath   = $CMAQ_DATA/mcip      #> meteorology input directory
set JVALpath  = $CMAQ_DATA/jproc     #> offline photolysis rate table directory
set OMIpath   = $CMAQ_HOME/CCTM/src/phot/inline #> ozone columne data for the photolysis model
set LUpath    = $CMAQ_DATA/dust      #> BELD landuse data for windblown dust model
```

```
set SZpath    = $CMAQ_DATA/ocean     #> surf zone file for in-line seasalt emissions
set NMLpath   = $CMAQ_HOME/CCTM/src/MECHS #> mechanism namelist files

set OUTDIR    = $CMAQ_DATA/cctm      #> output file directory

# ======================================================================
#> Input Files
# ======================================================================

#> Initial conditions
set ICFILE = CCTM_D51a_Linux2_x86_64intel_CGRID.CMAQ51-BENCHMARK_20110630

#> Boundary conditions
set BCFILE = BCON_D51a_12CalnexBench_20110701

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

#> Ozone column data
set OMIfile   = OMI_1979_to_2015.dat

#> Optics file
set OPTfile = PHOT_OPTICS.dat

#> MCIP meteorology files
set EXTN = 110701
setenv GRID_DOT_2D $METpath/GRIDDOT2D_${EXTN}
setenv GRID_CRO_2D $METpath/GRIDCRO2D_${EXTN}
setenv MET_CRO_2D $METpath/METCRO2D_${EXTN}
setenv MET_CRO_3D $METpath/METCRO3D_${EXTN}
setenv MET_DOT_3D $METpath/METDOT3D_${EXTN}
setenv MET_BDY_3D $METpath/METBDY3D_${EXTN}

#> Emissions files

if ( $CTM_PT3DEMIS == 'N' ) then
   set EMISfile  = e3d.${YMD}.12EUS1_279X240.cb05soa.24.ncf #> Offline 3d emissions file name
else
   #> In-line emissions configuration
   set STKCASEG = 12US1_G20_CB05E51
   set STKCASEE = 12US1_cb05e51_G20_CB05E51
   set CASE = 12CalnexBench_cb05e51_G20_CB05E51
   set EMISfile  = emis_mole_all_${YMD}_${CASE}.ncf #> Surface emissions
   setenv NPTGRPS 5          #> Number of elevated source groups
```

```
   setenv STK_GRPS_01 $IN_PTpath/stack_groups_ptnonipm_${STKCASEG}.ncf
   setenv STK_GRPS_02 $IN_PTpath/stack_groups_ptegu_${STKCASEG}.ncf
   setenv STK_GRPS_03 $IN_PTpath/stack_groups_othpt_${STKCASEG}.ncf
   setenv STK_GRPS_04 $IN_PTpath/stack_groups_ptfire_${YMD}_${STKCASEG}.ncf
   setenv STK_GRPS_05 $IN_PTpath/stack_groups_pt_oilgas_${STKCASEG}.ncf
   setenv LAYP_STTIME $STTIME
   setenv LAYP_NSTEPS $NSTEPS

   setenv STK_EMIS_01 $IN_PTpath/inln_mole_ptnonipm_${YMD}_${STKCASEE}.ncf
   setenv STK_EMIS_02 $IN_PTpath/inln_mole_ptegu_${YMD}_${STKCASEE}.ncf
   setenv STK_EMIS_03 $IN_PTpath/inln_mole_othpt_${YMD}_${STKCASEE}.ncf
   setenv STK_EMIS_04 $IN_PTpath/inln_mole_ptfire_${YMD}_${STKCASEE}.ncf
   setenv STK_EMIS_05 $IN_PTpath/inln_mole_pt_oilgas_${YMD}_${STKCASEE}.ncf
   setenv LAYP_STDATE $STDATE
endif

#> Lightning NOx configuration
if ( $CTM_LTNG_NO == 'Y' ) then
#   setenv LTNGNO $IN_LTpath/nox_CMAQ-BENCHMARK.35L.$YMD  #> offline calculated lightning NOx

#> In-line lightning NOx options
   setenv USE_NLDN  N        #> use hourly NLDN strike file [ default: Y ]
   setenv LTNGPARAM N        #> use lightning parameter file [ default: Y ]
   if ( $USE_NLDN == Y ) then
      setenv NLDN_STRIKES
   else
      setenv LOG_START 2.0   #> RC value to transit linear to log linear
   endif
   setenv LTNGPARMS_FILE $CMAQ_DATA/lightning/LTNG_RATIO.$YEAR.$MONTH.12CalnexBench.ioapi #> li
ARAM = N
   setenv LTNGOUT $OUTDIR/$EXEC.LTNGDIAG.${CFG}_${YMD} #> lightning diagnostic file; ignore if
endif

#> In-line biogenic emissions configuration
if ( $CTM_BIOGEMIS == 'Y' ) then
   set GSPROpath = ${CMAQ_DATA}/emis
   setenv GSPRO $GSPROpath/gspro_cb05soa_notoxics_cmaq_poc_09nov2007.txt
   set IN_BEISpath = ${CMAQ_DATA}/emis
   setenv B3GRD     $IN_BEISpath/b3grd.smoke30_beis361.12CalnexBench.2006NLCD_FIA5.1_norm_folia
   setenv BIOG_SPRO     B10C5 # speciation profile to use for biogenics
   setenv BIOSW_YN      N     # use frost date switch [ default: Y ]
   setenv BIOSEASON $IN_BEISpath/bioseason.cmaq.2011_12CalnexBench_wetland100.ghrsst.ncf #> ign
   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 SOILIN
```

```
   setenv SOILINP $OUTDIR/$EXEC.SOILINP.${CFG}_${YMD}  # Biogenic NO soil input file; ignore if
endif

#> Windblown dust emissions configuration
if ( $CTM_WB_DUST == 'Y' ) then
   # Input variables for BELD3 Landuse option
   setenv DUST_LU_1 $LUpath/beld3_12CalnexBench_output_a.ncf
   setenv DUST_LU_2 $LUpath/beld4_12CalnexBench_output_tot.ncf
   setenv MODIS_FPAR $LUPath/modis_fpar_lai_12km_20101201_20111231_daily_ioapi.ncf

   # Input variables for BELD4 Landuse option
   setenv BELD4_LU $LUpath/
   # All other Landuse options (USGS24, MODIS_NOAH, MODIS, NLCD50, NLCD40) read the GRID_CRO_2D
   if ( $CTM_ERODE_AGLAND == 'Y' ) then
      setenv CROPMAP01 ${CMAQ_DATA}/crop/BeginPlanting_12CalnexBench
      setenv CROPMAP04 ${CMAQ_DATA}/crop/EndPlanting_12CalnexBench
      setenv CROPMAP08 ${CMAQ_DATA}/crop/EndHarvesting_12CalnexBench
   endif
endif

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

#> Bidiretional ammonia configuration
if ( $CTM_ABFLUX == 'Y' ) then
   setenv E2C_Soilfile  ${CMAQ_DATA}/bidi/2011_12CalnexBench_soil.nc
   setenv E2C_Fertfile  ${CMAQ_DATA}/bidi/2011_12CalnexBench_time${YMD}.nc
   setenv B4LU_file     ${CMAQ_DATA}/bidi/beld4_12CalnexBench_2006nlcd.ncf
   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 ${CFG}_${YMD}
#> set output file names
 set CONCfile  = $EXEC.CONC.${CTM_APPL}              # CTM_CONC_1
 set ACONCfile = $EXEC.ACONC.${CTM_APPL}             # CTM_ACONC_1
 set CGRIDfile = $EXEC.CGRID.${CTM_APPL}             # CTM_CGRID_1
 set MEDfile   = $EXEC.MEDIA_CONC.${CTM_APPL}        # MEDIA_CONC
 set DD1file   = $EXEC.DRYDEP.${CTM_APPL}            # CTM_DRY_DEP_1
```

```
 set DV1file   = $EXEC.DEPV.${CTM_APPL}              # CTM_DEPV_DIAG
 set PT1file   = $EXEC.PT3D.${CTM_APPL}              # CTM_PT3D_DIAG
 set BIO1file  = $EXEC.B3GTS_S.${CTM_APPL}           # B3GTS_S
 set SOIL1file = $EXEC.SOILOUT.${CTM_APPL}           # SOILOUT
 set WD1file   = $EXEC.WETDEP1.${CTM_APPL}           # CTM_WET_DEP_1
 set WD2file   = $EXEC.WETDEP2.${CTM_APPL}           # CTM_WET_DEP_2
 set AV1file   = $EXEC.PMVIS.${CTM_APPL}             # CTM_VIS_1
 set AV2file   = $EXEC.APMVIS.${CTM_APPL}            # CTM_AVIS_1
 set AD1file   = $EXEC.PMDIAG.${CTM_APPL}            # CTM_PMDIAG_1
 set AD2file   = $EXEC.APMDIAG.${CTM_APPL}           # CTM_APMDIAG_1
 set RJ1file   = $EXEC.PHOTDIAG1.${CTM_APPL}         # CTM_RJ_2
 set RJ2file   = $EXEC.PHOTDIAG2.${CTM_APPL}         # CTM_RJ_2
 set SSEfile   = $EXEC.SSEMIS.$CTM_APPL              # CTM_SSEMIS_1
 set DSEfile   = $EXEC.DUSTEMIS.$CTM_APPL            # CTM_DUST_EMIS_1
 set PA1file   = $EXEC.PA_1.${CTM_APPL}              # CTM_IPR_1
 set PA2file   = $EXEC.PA_2.${CTM_APPL}              # CTM_IPR_2
 set PA3file   = $EXEC.PA_3.${CTM_APPL}              # CTM_IPR_3
 set IRR1file  = $EXEC.IRR_1.${CTM_APPL}             # CTM_IRR_1
 set IRR2file  = $EXEC.IRR_2.${CTM_APPL}             # CTM_IRR_2
 set IRR3file  = $EXEC.IRR_3.${CTM_APPL}             # CTM_IRR_3
 set DVMfile   = $EXEC.DEPVFST.${CTM_APPL}           # CTM_DEPV_FST
 set DVFfile   = $EXEC.DEPVMOS.${CTM_APPL}           # CTM_DEPV_MOS
 set DDFfile   = $EXEC.DDFST.${CTM_APPL}             # CTM_DRY_DEP_FST
 set DDMfile   = $EXEC.DDMOS.${CTM_APPL}             # CTM_DRY_DEP_MOS
 set VDFfile   = $EXEC.VDIFF_DIAG.${CTM_APPL}        # CTM_VDIFF_DIAG   diag
 set VSDfile   = $EXEC.VSED_DIAG.${CTM_APPL}         # CTM_VSED_DIAG    diag
 set AODfile   = $EXEC.AOD_DIAG.${CTM_APPL}          # CTM_AOD_1        diag
#> In-line biogenic emissions output files
if ( $CTM_BIOGEMIS == 'Y' ) then
   setenv B3GTS_S $OUTDIR/$EXEC".B3GTS_S".${CTM_APPL}
   setenv SOILOUT $OUTDIR/$EXEC".SOILOUT".${CTM_APPL}  # Biogenic NO soil output file
endif

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

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

#> look for existing log files

 set test = `ls CTM_LOG_???.${CTM_APPL}`
 if ( "$test" != "" ) then
    if ( $DISP == 'delete' ) then
       echo " ancillary log files being deleted"
```

```
       foreach file ( $test )
          echo " deleting $file"
          rm $file
          end
       else
       echo "*** Logs exist - run ABORTED ***"
       exit 1
       endif
    endif

#> for the run control ...

setenv CTM_STDATE      $STDATE
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 ${BLD}/GC_$MECH.nml
setenv ae_matrix_nml ${BLD}/AE_$MECH.nml
setenv nr_matrix_nml ${BLD}/NR_$MECH.nml
setenv tr_matrix_nml ${NMLpath}/trac0/Species_Table_TR_0.nml

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


#>- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

 source $BASE/outck.q

 ls -l $BLD/$EXEC; size $BLD/$EXEC
 unlimit
 limit

#> 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
 exit
```

10.5 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.

10.6 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.

11 Analysis Tools for CMAQ

Several tools are freely available for visualizing, analyzing, and evaluating CMAQ inputs and outputs. The list includes CMAQ utility tools, m3tools, PAVE, VERDI, Panoply, the Atmospheric Model Evaluation Tool (AMET), netCDF Operators (NCO), UNIDATA Integrated Data Viewer (IDV), and the NCAR Command-line Language (NCL). Several other commercial packages, including MATLAB and IDL, also support the analysis and visualization of CMAQ inputs and outputs. Most visualization and analysis software that supports netCDF file formats will work with CMAQ output data.

This page briefly describes several of these software utilities and provides links to where they may be downloaded. Refer to the documentation for each piece of software for additional information.

Two classes of analysis tools are presented here - Command line utilities for manipulating CMAQ input/output data - netCDF - appendwrf - bldoverlay - block_extract - combine - hr2day - sitecmp - sitecmp_dailyo3 - writesite - m3tools - netCDF Operators

- Visualization tools for graphical display of CMAQ input/output data
- VERDI
- AMET
- PAVE
- IDV
- NCL

11.1 Command Line Data Processors

11.1.1 netCDF

http://www.unidata.ucar.edu/software/netcdf/

Almost all of the CMAQ input and output files use the I/O API netCDF file format. If the user has already built the netCDF library for compiling CMAQ, the ncdump utility should also be available on the user's machine. This utility generates an ASCII representation of the netCDF file using the CDF notation developed by NCAR.

The UNIX syntax for invoking ncdump is the following:

```
ncdump [-h] [-c] [-n name] [inputfile]
```

where:

-h produces only the "header" information in the output file; i.e., the declarations of dimensions, variables, and attribute, but no data values for the variables.

-c produces the "header" information in the output file and the data values for coordinate variables (variables that are also dimensions).

-n name is used to specify a different name for the network Common data form Description Language (CDL) description than the default.

11.1.2 CMAQ Utility Tools

https://github.com/USEPA/CMAQ

Several Fortran-based post-processing tools are provided along with the CMAQ code/scripts distribution. These are located in the $CMAQ_HOME/POST directory in the CMAQ distribution (version 5.2 and later). These tools work directly with the CMAQ outputs and help in processing, formatting, and preparing datasets from various ambient monitoring networks for subsequent evaluation. These networks include the EPA Air Quality System (AQS)AIRS-AQS, Interagency Monitoring of Protected Visual Environments (IMPROVE), Clean Air Status Trends Network (CASTNET), Speciated Trends Network (STN), National Atmospheric Deposition Program (NADP), Mercury Deposition Network (MDN) and the Southeast Aerosol Research and Characterization Study (SEARCH). The formatted observation data files needed for running the sitecmp and sitecmp_dailyo3 utilities are available for 2000 through 2014 from the CMAS Center Data Clearinghouse under the heading "2000-2014 North American Air Quality Observation Data": https://www.cmascenter.org/download/data.cfm.

The various CMAQ utility tools are described below. Documentation and sample run scripts are provided in the $CMAQ_HOME/POST directory for each utility.

- **appendwrf**: This program concatenates variables from multiple WRF input or output files into a single file along the Time (unlimited) dimension. This can be useful in cases where a user may have WRF input or output files that were generated for shorter time periods and wants to combine them into files with longer (e.g. monthly) duration. - **bldoverlay**: This program creates an observation overlay file that can be imported into either PAVE or VERDI. - **block_extract**: This program extracts time series of 1 or more variables from 1 or more (up to 99) IOAPI files for a specified range of cells. - **combine**:This program combines species from raw CMAQ output files or wrfout input files into a new IOAPI output file. Species can be aggregated or transformed into variables of interest (i.e. to match observed quantities from a specific monitoring network). - **hr2day**: This program creates gridded I/O API files with daily values (e.g. daily average, daily sum, maximum daily 8-hr average) from gridded I/O API files containing hourly values. - **sitecmp**: This program generates a csv (comma separated values) file that compares CMAQ generated concentrations with an observed dataset. - **sitecmp_dailyo3**: This program generates a csv (comma separated values) file that compares various daily ozone metrics computed from hourly CMAQ generated and observed ozone concentrations. The metrics included in the output file are daily maximum 1-hr ozone concentrations, daily maximum 1-hr ozone concentrations in the nine cells surrounding a monitor, time of occurrence of daily maximum 1-hr ozone concentrations, daily maximum 8-hr ozone concentrations, daily maximum 8-hr ozone concentrations in the nine cells surrounding a monitor, time of occurrence of daily maximum 8-hr ozone concentrations, the daily

W126 ozone value, and the daily SUM06 ozone value. - **writesite**: This program generates a csv file from an IOAPI data file for a set of species at defined site locations.

11.1.3 M3tools

<https://www.cmascenter.org/ioapi/>

An extensive set of utility programs called *m3tools* that use the I/O API library have been developed and made available for the modeling community. These utility routines assist in manipulating dates and times, performing coordinate conversions, storing and recalling grid definitions, sparse matrix arithmetic, etc., as well as in data manipulation and statistical analyses. All *m3tools* can be run at the command line, and the various options can be provided interactively, or all of them can be stored in a file and executed as scripts.

A list of these utility programs and brief descriptions is provided below.

- **airs2m3**: Imports air quality monitor data from an AIRS AMP350-format ASCII file and puts them into an I/O API "observational data" file
- **bcwndw**: Extracts data from a gridded file to the boundary of a subgrid window (see **m3wndw** later in this list for extracting to the window itself)
- **datshift**: Takes calendar date (form YYYYMMDD) and a number of days D, and reports the date D days later.
- **gregdate**: Computes calendar-style date "Month DD, YYYY", day-of-week (Sunday, Monday, …, Saturday), and whether or not Daylight Saving Time is in effect from Julian date YYYYDDD, or from "yesterday", "today", or "tomorrow"
- **juldate**: Computes Julian date YYYYDDD, day-of-week (Sunday, Monday, …, Saturday), and whether or not Daylight Saving Time is in effect from calendar-style date "Month DD, YYYY", or from "yesterday", "today", or "tomorrow".
- **m3combo**: Computes linear combinations of sets of variables from an I/O API input file, and writes the resulting variables to an I/O API output file
- **m3cple**: Copies to the same grid, or interpolates to another grid, a time sequence of all variables from a source file to a target file, under the optional control of an I/O API coupling-mode "synch file"
- **m3diff**: Computes statistics for pairs of variables and for the results of applying various comparison ("differencing") operations to those variables in a pair of files.
- **m3edhdr**: Edits header attributes/file descriptive parameters
- **m3fake**: Builds a file according to user specifications, filled either with dummy data or with data read in from a set of user-supplied files
- **m3merge**: Merges selected variables from a set of input files for a specified time period, and writes them to a single output file, with optional variable renaming in the process
- **m3pair**: Builds an ASCII file of paired values for two variables from two files, within a user-selected window into the grid, according to user specifications
- **m3stat**: Computes statistics for variables in a file
- **m3tproc**: Computes time period aggregates (e.g., 08:00-16:00 gridded daily maxima) and writes them to an output file. Can be used to create running averages, (e.g., 8-h O3 data from 1-h O3), daily averages, daily maxima, etc.

- **m3tshift**: Copies/time-shifts data from a file
- **m3wndw**: Windows data from a gridded file to a subgrid (see **bcwndw** earlier in this list for extracting to the boundary of the subgrid window)
- **m3xtract**: Extracts a subset of variables from a file for <*time interval*> .Can also be used to concatenate data from two or more files with different time periods into one file
- **m4filter**: Converts first-edition Models-3 files to current version
- **mtxblend**: Uses a sparse-matrix file to interpolate/transform data from an input file to the grid of a "base" file and to merge it with data from the "base" file
- **mtxbuild**: Builds a sparse-matrix transform file from user-supplied ASCII coefficient inputs
- **mtxcalc**: Builds a grid-to-grid sparse-matrix transform file using a subsampling algorithm
- **mtxcple**: Uses a sparse-matrix file to interpolate a time sequence of all variables from a source file to a target file, under the optional control of an I/O API coupling-mode "synch file"
- **presterp**: Interpolates from a 3-D sigma-coordinate file to a new 3-D pressure-coordinate file, using coefficients from PRES_CRO_3D
- **selmrg2d**: Selects multiple 2-D layer/variable combinations from multiple gridded input files, and writes result to merged 2-D gridded output file
- **utmtool**: Performs coordinate conversions and grid-related computations for lat-lon, Lambert, and UTM coordinate systems.
- **vertot**: Computes vertical-column totals of variables in a file

11.1.4 netCDF Operators (NCO)

http://nco.sourceforge.net/

The netCDF Operators (NCO) are a suite of programs known as operators. Each operator is a stand-alone, command-line program that is executed at the UNIX shell level, similar to the commands ls or mkdir. The operators take netCDF files as input, then perform a set of operations (e.g., deriving new data, averaging, hyperslabbing, or metadata manipulation) and produce a netCDF file as output. The operators are primarily designed to aid manipulation and analysis of gridded scientific data. The single command style of NCO allows users to manipulate and analyze files interactively and with simple scripts, avoiding the overhead (and some of the power) of a high-level programming environment.

NCO achieves flexibility by using *command-line options*. These options are implemented in all traditional UNIX commands as single-letter *switches*, e.g., 'ls -l'. NCO supports both short-format (single letter) and long-format (multiletter) options.

An overview of the various netCDF operators is given below.

- **ncap (netCDF Arithmetic Processor)**: ncap and ncap2 arithmetically process netCDF files. The processing instructions are contained either in the NCO script file fl.nco or in a sequence of command-line arguments.
- **ncatted (netCDF Attribute Editor)**: ncatted edits attributes in a netCDF file. ncatted can *append*, *create*, *delete*, *modify*, and *overwrite* attributes (all explained below). Furthermore, ncatted allows each editing operation to be applied to every variable in a file. This saves time when changing attribute conventions throughout a file.

- **ncbo (netCDF Binary Operator)**: ncbo performs binary operations on variables in *file_1* and the corresponding variables (those with the same name) in *file_2* and stores the results in *file_3*. The binary operation operates on the entire files.
- **ncea (netCDF Ensemble Averager)**: ncea performs grid-point averages of variables across an arbitrary number (an *ensemble*) of *input-files*, with each file receiving an equal weight in the average. Each variable in the *output-file* will be the same size as the same variable in any one of the *input-files*, and all *input-files* must be the same size. ncea averages entire files, and weights each file evenly. This is distinct from ncra (discussed later in this list), which averages only over the record dimension (e.g., time), and weights each record in the record dimension evenly; ncea *always averages* coordinate variables, regardless of the arithmetic operation type performed on the noncoordinate variables. All dimensions, including the record dimension, are treated identically and preserved in the *output-file*.
- **ncecat (netCDF Ensemble Concatenator**): ncecat concatenates an arbitrary number of input files into a single output file. A new record dimension acts as the glue to bind the input files data together. Each variable in each input file becomes one record in the same variable in the output file. All *input-files* must contain all extracted variables (or else there would be "gaps" in the output file). Each extracted variable must be constant in size and rank across all *input-files*. The *input-files* are stored consecutively as a single record in *output-file*. Thus, the *output-file* size is the sum of the sizes of the extracted variable in the input files.
- **ncflint (netCDF File Interpolator)**: ncflint creates an output file that is a linear combination of the input files. This linear combination is a weighted average, a normalized weighted average, or an interpolation of the input files. Coordinate variables are not acted upon in any case; they are simply copied from *file_1*.
- **ncks (netCDF Kitchen Sink)**: ncks combines selected features of ncdump, ncextr, and the nccut and ncpaste specifications into one versatile utility. ncks extracts a subset of the data from *input-file* and prints it as ASCII text to stdout, writes it in flat binary format to binary-file, and writes (or pastes) it in netCDF format to *output-file*.
- **ncpdq (netCDF Permute Dimensions Quickly)**: ncpdq performs one of two distinct functions—packing or dimension permutation—but not both. ncpdq is optimized to perform these actions in a parallel fashion with a minimum of time and memory.
- **ncra (netCDF Record Averager)**: ncra averages record variables across an arbitrary number of *input-files*. The record dimension is, by default, retained as a degenerate (size 1) dimension in the output variables.
- **ncrcat (netCDF Record Concatenator)**: ncrcat concatenates record variables across an arbitrary number of *input-files*. The final record dimension is by default the sum of the lengths of the record dimensions in the input files.
- **ncrename (netCDF Renamer)**: ncrename renames dimensions, variables, and attributes in a netCDF file. Each object that has a name in the list of old names is renamed using the corresponding name in the list of new names. All the new names must be unique.
- **ncwa (netCDF Weighted Averager)**: ncwa averages variables in a single file over arbitrary dimensions, with options to specify weights, masks, and normalization.

11.2 Visualization Tools

11.2.1 Visualization Environment for Rich Data Interpretation (VERDI)

http://www.verdi-tool.org

The Visualization Environment for Rich Data Interpretation (VERDI) is a flexible and modular Java-based visualization software tool that allows users to visualize multivariate gridded environmental datasets created by environmental modeling systems such as SMOKE, CMAQ and WRF, namely gridded concentration and deposition fields that users need to visualize and compare with observational data both spatially and temporally. VERDI has been designed keeping most of the functionality of PAVE in mind, and hence can help users analyze and visualize model outputs in a very similar vein, using both command-line driven scripts as well as using a Graphical User Interface (GUI). Further, VERDI is under active development to enhance its features beyond PAVE.

11.2.2 Atmospheric Model Evaluation Tool (AMET)

http://www.cmascenter.org

The Atmospheric Model Evaluation Tool (AMET) is a suite of software designed to facilitate the analysis and evaluation of meteorological and air quality models. AMET matches the model output for particular locations to the corresponding observed values from one or more networks of monitors. These pairings of values (model and observation) are then used to statistically and graphically analyze the model's performance. More specifically, AMET is currently designed to analyze outputs from MM5, WRF, CMAQ, and CAMx as well as MCIP-postprocessed meteorological data (surface only).

The basic structure of AMET consists of two ‘’fields”and two *processes*. The two fields (scientific topics) are MET and AQ, corresponding to meteorology and air quality data. The two processes (actions) are database population and analysis. Database population refers to the underlying structure of AMET; after the observations and model data are paired in space and time, the pairs are inserted into a MySQL database. Analysis refers to the statistical evaluation of these pairings and their subsequent plotting. Practically, a user may be interested in using only one of the fields (either MET or AQ), or may be interested in using both fields. That decision is based on the scope of the study. The three main software components of AMET are MySQL (an open-source database software system), R (a free software environment for statistical computing and graphics), and perl (an open-source, cross-platform programming language).

11.2.3 Package for Analyses and Visualization of Environmental Data (PAVE)

http://paved.sourceforge.net

PAVE is a flexible and distributed application to visualize multivariate gridded environmental datasets. Features include

- baseline graphics with the option to export data to high-end commercial packages
- the ability to access and manipulate datasets located on remote machines

- support for multiple simultaneous visualizations
- an architecture that allows PAVE to be controlled by external processes
- low computational overhead
- no software distribution cost

PAVE is very widely used by the air quality modeling community, and it can produce various types of plots, including scatter plots, time-series plots, 2-D tile plots, 3-D surface plots, bar plots, wind-vector plots, etc. The source code for PAVE is also distributed under the terms of the GNU General Public License Version 2. PAVE can be run at the Linux command prompt, and the various commands/options can be invoked using the graphical user interface (GUI), or all of them can be stored in a script file and executed by running the script. However, note that PAVE is not being updated any more, and CMAS has ceased support for PAVE, and encourages the user community to move towards VERDI (discussed next).

11.2.4 Integrated Data Viewer (IDV)

http://www.unidata.ucar.edu/software/idv/

The Integrated Data Viewer (IDV) from Unidata is a Java™-based software framework for analyzing and visualizing geoscience data. The IDV release includes a software library and a reference application made from that software. It uses the VisAD library (<http://www.ssec.wisc.edu/~billh/visad.html>) and other Java-based utility packages.

The IDV is developed at the Unidata Program Center (UPC), part of the University Corporation for Atmospheric Research in Boulder, CO, which is funded by the National Science Foundation. The software is freely available under the terms of the GNU Lesser General Public License.

The IDV "reference application" is a geoscience display and analysis software system with many of the standard data displays that other Unidata software (e.g., GEMPAK and McIDAS) provides. It brings together the ability to display and work with satellite imagery, gridded data (for example, numerical weather prediction model output), surface observations, balloon soundings, NWS WSR-88D Level II and Level III RADAR data, and NOAA National Profiler Network data, all within a unified interface. It also provides 3-D views of the earth system and allows users to interactively slice, dice, and probe the data, creating cross-sections, profiles, animations and value read-outs of multidimensional data sets. The IDV can display any Earth-located data if they are provided in a supported format.

IDV includes the capability to read I/O API netCDF formatted files and a scripting interface to create and manipulate images and movies. The scripting is accomplished through an XML file: *IDV Scripting Language* (ISL). The ISL file can be opened from a running IDV, or one can be passed to the IDV as a command-line argument:

```
runIDV capture.isl
```

11.3 NCAR Command Language (NCL)

http://www.ncl.ucar.edu

The NCAR Command Language (NCL) is a free, interpreted language designed specifically for scientific data processing and visualization. NCL has robust file input and output. It can read in netCDF, HDF4, HDF4-EOS, GRIB, binary, and ASCII data. The output graphics from NCL are of high quality, and highly customizable.

It runs on many different operating systems, including Solaris, AIX, IRIX, Linux, MacOSX, Dec Alpha, and Cygwin/X running on Windows. It is available for free in binary format.

NCL can be run in interactive mode, where each line is interpreted as it is entered at the user's workstation, or it can be run in batch mode as an interpreter of complete scripts. The user can also use command-line options to set options or variables on the NCL command line.

The power and utility of the language are evident in three areas:

- file input and output
- data analysis
- visualization

NCL has many features common to modern programming languages, including types, variables, operators, expressions, conditional statements, loops, and functions and procedures. The various NCL commands can be executed at the NCL command prompt, or all the commands can be stored in a file and invoked as follows:

ncl commands.ncl

NCL comes with numerous predefined functions and resources that the user can easily invoke. These include functions for data processing, visualization, and various mathematical and statistical analyses, such as empirical orthogonal functions (EOFs) and singular value decomposition (SVD). As an example, contributed.ncl is a library of user-contributed functions within NCL. This library is distributed with NCL, and loading the script at the beginning of the user's NCL script therein can access the functions.

```
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/contributed.ncl"
```

NCL also has a capability to call external Fortran or C routines. This is enabled through an NCL wrapper script called WRAPIT. The wrapper file is a C program that describes all of the arguments and passes them back and forth between the function/procedure the user wants to call, and the NCL script that is calling it. Thus, when the user invokes WRAPIT on the Fortran code that needs to be called from NCL, it creates a special C wrapper file, compiles it and the Fortran file, and generates a *.so file that the user can then load into NCL using the "external" statement.

12 CMAQ Support Resources

Technical and operational user support for CMAQ are available free of charge from the Community Modeling and Analysis System Center (<http://www.cmascenter.org>). The CMAS Center offers an e-mail help desk, and community listservs for posting questions about CMAQ. In addition to these community-based resources, the CMAS Center offers fee-based trainings, and provides a documentation library for CMAQ that includes operational and technical guidance manuals as well as references to primary literature involving CMAQ.

12.1 The CMAS Center

Under contract to EPA, the Center for Environmental Modeling for Policy Development (CEMPD) at the University of North Carolina at Chapel Hill (UNC) Institute for the Environment maintains the CMAS Center for supporting community-based air quality modeling. CMAS is an approach to the development, application, and analysis of environmental models that leverages the complementary talents and resources of the modeling community in order to set new standards for quality in science and in the reliability of the application of the technology.

From research to application to outreach, the CMAS Center advances the community modeling paradigm through the establishment of a centralized resource to serve the members of the national and international environmental modeling community.

12.1.1 CMAS functions

Currently, the following activities are available through the CMAS Center:

- On-line help desk - Get help with the supported CMAS products
- Model clearinghouse - Download the supported CMAS products
- Training courses - Attend a training course on emissions modeling, air quality modeling, or other related topics
- Conferences - Attend the annual CMAS conference to interact with the community
- Development assistance - Add new science to the supported CMAS products
- Documentation - Access on-line documentation for the CMAS products
- Model-related research - Learn about the latest developments in modeling research
- Data clearinghouse - Access air quality modeling data from around the community

12.2 Getting Help with CMAQ

The CMAS Center website (<http://www.cmascenter.org>) includes a help desk with resources that are available to assist with CMAQ-related issues. The CMAS help desk services are free to the community. Many of the services in the help desk benefit from increased usage, such as the listserv discussion groups. E□mail-based CMAQ technical consultation is available to registered CMAS participants only.

The following resources are available through the CMAS Center to address CMAQ-related questions. Community members should use these resources in the order that follows. There is currently a large, searchable database of resolved CMAQ support tickets; before submitting a new ticket to the help desk, be sure to search the database for keywords to see if the issue has been addressed previously. Section 11.3 provides a list of the web pages referenced in Section 11.2.

12.2.1 Documentation

The first place to look for an answer to CMAQ-related questions is the on-line documentation for the software. The CMAS documentation page contains links to available documentation for current and

previous releases of the various kinds of software that CMAS supports. Peruse these on-line manuals, as many of them contain FAQs and discussion specific to the various programs.

12.2.2 Interactive resources

Search the CMAS FAQs and listservs for information about the question that you have. These services are organized by topic to facilitate searching. Look under the CMAS Model Clearinghouse area to find out about new releases, and read through the release notes of past releases for detailed information about the features of the models.

12.2.3 Tutorials/training

General questions regarding model installation or application may be addressed in the online tutorials for the CMAS-supported software. More-specific tutorials will be added over time; users can suggest tutorial topics by contacting the CMAS Center.

The CMAS Center offers quarterly trainings on CMAQ at the Institute for the Environment offices in Chapel Hill, North Carolina, USA. CMAS training staff are also available to travel for on-site training anywhere in the world. The currently available training is an introductory course to CMAQ that covers configuration, compilation, and basic operation of the model. Visit the CMAS training web page to see an agenda, fees, and the schedules for upcoming training courses

12.2.4 E-mail support

E-mail support is available to CMAS users who have a support account, and provides case-specific support for all CMAS-supported software, which includes CMAQ. CMAS e-mail support provides direct access to expert CMAQ users for questions about installation or operational issues. E-mail support also provides direct access to the CMAQ developers for technical questions about model formulation, model science, and code integration. Visit the e-mail support page for an explanation of how to use the system and to register.

12.3 Contacting CMAS

The CMAS Center is available on the web at <http://www.cmascenter.org>. Table 11-1 lists important contacts for the CMAS Center.

Table 11□1. CMAS contact information and important links

Resource	Link
Main website	<http://www.cmascenter.org>
General Questions	cmas@unc.edu
Help Desk	<http://www.cmascenter.org/help_desk.cfm>
Training Information	<http://www.cmascenter.org/training.cfm>

Resource	Link
Conferences and Workshops	<http://www.cmascenter.org/conference.cfm>
Downloads	<http://www.cmascenter.org/download.cfm>
Release Calendar	<http://www.cmascenter.org/release_calendar.cfm>
FAQs	<http://www.cmascenter.org/help/faq.cfm>
CMAQ Home Page	<http://www.cmaq-model.org>

12.4 Appendix A: CMAQ v5.2 Mechanism Table

The table below lists the chemistry mechanisms available in CMAQ v5.2. The entries in the MECHS Module column correspond to the `Mechanism` setting in the CMAQ build scripts. The entries in this column link to documentation for each mechanism. The next two columns in the table define the aerosol and cloud modules that are compatible with each mechanism. The fourth column links to the mechanism definition, showing the details of the stoichiometry and kinetics of each mechanism, the last column links to the species tables from Appendix A. See the CMAQv5.2 release notes for additional information on photochemical mechanisms in CMAQ.

MECHS Module	Aerosol Module	Cloud Module	Mechanism Definition	Species Table
cb05e51_ae6_aq	aero6	acm_ae6 or acm_ae6_kmt	mech_cb05e51_ae6_aq.def	Table A-1
cb05e51_ae6nvPOA_aq	aero6	acm_ae6 or acm_ae6_kmt	mech_cb05e51_ae6nvPOA_aq.def	
cb05eh51_ae6_aq	aero6	acm_ae6 or acm_ae6_kmt	mech_cb05eh51_ae6_aq.def	
cb05mp51_ae6_aq	aero6	acm_ae6_mp	mech_cb05mp51_ae6_aq.def	
cb05tucl_ae6_aq v5.0 v5.1	aero6	acm_ae6 or acm_ae6_kmt	mech_cb05tucl_ae6_aq.def	Table A-2
cb6r3_ae6_aq	aero6	acm_ae6 or acm_ae6_kmt	mech_cb6r3_ae6_aq.def	Table A-3
cb6r3_ae6nvPOA_aq	aero6	acm_ae6 or acm_ae6_kmt	mech_cb6r3_ae6nvPOA_aq.def	
racm2_ae6_aq	aero6	acm_ae6 or acm_ae6_kmt	mech_racm2_ae6_aq.def	Table A-4
saprc07tb_ae6_aq	aero6	acm_ae6 or acm_ae6_kmt	mech_saprc07tb_ae6_aq.def	Table A-5
saprc07tc_ae6_aq	aero6	acm_ae6 or acm_ae6_kmt	mech_saprc07tc_ae6_aq.def	Table A-6
saprc07tc_ae6nvPOA_aq	aero6	acm_ae6 or acm_ae6_kmt	mech_saprc07tc_ae6nvPOA_aq.def	
saprc07tic_ae6i_aq	aero6	acm_ae6	mech_saprc07tic_ae6i_aq.def	Table A-7

MECHS Module	Aerosol Module	Cloud Module	Mechanism Definition	Species Table
saprc07tic_ae6i_aqkmti	aero6	acm_ae6i_kmt	mech_saprc07tic_ae6i_aqkmti.def	
saprc07tic_ae6invPOA_aq	aero6	acm_ae6 or acm_ae6_kmt	mech_saprc07tic_ae6invPOA_aq.def	

13 GLOSSARY

Accumulation mode. Aerosol particles with diameters nominally between 0.1 and 1.0 micrometers. The term is based upon the concept that particles in this size range accumulate in the atmosphere from the growth of much smaller particles. (See also "Aitken mode.") The growth may be from new mass condensing on the smaller particles, or from the coagulation of the smaller particles.

Adaptive grid. A grid structure that varies during model execution according to the value(s) of some model parameter(s). For example, in a photochemistry model, grid resolution may automatically increase in areas of high spatial gradients of NOx. This allows more accurate determination of plume-to-background concentration ratios, which greatly influence photochemical ozone production. In a meteorological model, an adaptive grid may automatically increase the grid resolution in an area of modeling where there is a large atmospheric pressure change across a grid cell.

Air quality modeling system. A computational system environment that combines a set of physical and chemical models that describe atmospheric processes, which are important to trace-gas and particulate matter distributions in the atmosphere. These systems typically include meteorological models, emissions models, chemistry-transport models, and the analysis and visualization tools necessary for supporting decisions related to air quality.

Aitken mode. Aerosol particles with diameters nominally between 0.01 and 0.1 micrometers (μm). Such particles are formed in the atmosphere by nucleation of gas□phase precursors, or by direct emissions from sources. The most common source is combustion (e.g., diesel soot).

Arakawa-B. Horizontal grid staggering system described by Arakawa and Lamb (1977) and used by MM5. Mass variables and momentum variables are defined on separate horizontal grids of spacing equal to Delta-x. The two grids are offset by 0.5 □ Delta-x in the north-south and east-west directions.

Arakawa-C. Horizontal grid staggering system described by Arakawa and Lamb (1977) and used by WRF□ARW and CCTM. Mass variables and each horizontal component of the momentum are defined using separate horizontal grids of spacing equal to Delta-x.

ArcInfo. A high-end geographical information system (GIS) with capabilities for the automation, modification, management, analysis, and display of geographical information.

Automatic quality control (QC). QC correction that is accomplished automatically without user intervention.

Bookmark. In on-line help, a bookmark marks an entry of a help document so it can be quickly accessed later. A list of bookmarks appears in the Bookmarks Menu on the help browser window and also in the bookmark's window. Each item on the list is a hypertext link to a marked entry.

Chemistry-transport model (CTM). A model that simulates various physical and chemical processes that are important for understanding atmospheric trace-gas and particles distributions. The processes include atmospheric transport; vertical mixing; atmospheric chemistry in the gas phase, in the aqueous phase, and in aerosols; cloud mixing and precipitation scavenging; and surface removal processes. Generally, a chemistry-transport model relies on a meteorological model for the description of atmospheric states and motions, and depends on an emissions model for the description of anthropogenic and biogenic emissions that are injected into the atmosphere.

Class. A collection of software modules associated with a science process.

Conforming datasets. Conforming datasets are in I/O API format. Most programs (models, visualization and analysis routines) in the CMAQ system are able to read and write conforming datasets. This capability eliminates the need for data conversion, even within a distributed computing environment.

Conforming programs. Conforming programs generally use the I/O API library routines for reading and writing data. Key data structures are defined by globally shared information. You define this critical data structure information once, and it is automatically made available for use in conforming code throughout the system. This globally shared information is permanently stored as objects in an object data base. In CMAQ, these objects are a stored collection of related information, such as grid dimensions and resolution, coordinate system definitions, chemical mechanism species, and reactions.

‘”Cross point.”’Grid point where all scalars are defined.

Daemon. A process that runs in the background independently and performs a function. An example is a printer daemon that controls the job queue for the printer.

Decision support system. An automated system that provides information for decision making.

Domain. The domain for a CMAQ calculation is the horizontal geographic area (e.g., the eastern United States) that is of interest. CMAQ calculations are performed for physical and chemical processes occurring in the atmosphere above this horizontal geographic area.

‘”Dot point.”’Grid point where wind components are defined

Emission model. This type of model calculates the emission rates of trace gas and particulate matter into the atmosphere under ambient meteorological conditions and socioeconomic activities. Emissions from both natural and manmade sources are calculated. One example of an emissions model is SMOKE.

Environmental modeling system. A set of computational models that mathematically represents a simplified version of real-world phenomena, along with the necessary mechanisms for managing the processing, the data produced by it, and the analysis and visualization tools necessary for making related decisions. For example, this could be the creation and behavior of environmental pollutants as a function of weather and chemical interactions. Researchers use these “scaled-down” versions of the world to perform experiments that would be too dangerous, too costly, or simply impossible in the real world.

Eulerian. Fluid motion specification where one concentrates on what happens at a spatial point, ‘’x,”so that independent variables are taken as a function of time at that point.

Evasion. Loss of material from the land surface (e.g. the upward flux of mercury from the land surface).

Exploratory models. Test-bed models for developing scientific algorithms that are not thoroughly reviewed and evaluated. (See also "operational models" and "screening models.")

Fortran. Formula translator (computer programming language).

Framework. A system of mechanisms to manage the scheduling and execution of computational models, the data produced by them, the analysis and visualization tools necessary for understanding their results for decision making, and the interfaces to all these capabilities.

Generalized coordinate system. A scheme for constructing coordinate systems that allows for choices of horizontal map projection type (e.g., latitude/longitude, Lambert, Mercator, polar stereographic, Universal Transverse Mercator, or matrix) and projection parameters, as well as for choices of various vertical coordinate types (e.g., pressure coordinate, height above ground, height above sea level, sigma-p hydrostatic, sigma-p nonhydrostatic, and sigma-z). The advantage of a generalized coordinate system is that a CMAQ model can adapt to a variety of different possibilities in horizontal and vertical parameters.

Generic grid system. A scheme for constructing grid systems that allows for choices of origin, orientation, and spatial resolution. It allows a model to be expressed in a grid system that is optimal for representing the governing equations. For a regular, rectangular grid system, mapping gridded data to the earth's surface can be achieved by defining the number of rows and columns, cell size, and location and extent. For an irregular grid system, grid intersections (nodes) are described by coordinates from a reference position.

Geocoded. An entity with an identifier associated with geographic boundaries (e.g., state or county code).

Geographic information system (GIS). A computer-based system for managing, analyzing, manipulating, and displaying geographic information in both numeric and graphical ways.

Geospatial. Refers to the spatial extent of a geographic boundary.

Grid cell. The smallest subdivision of a grid.

Grid size. Length of shortest side of a rectangular grid cell.

Grid. A network of conceptual locations at which numerical calculations are performed. This network extends horizontally to cover the domain of interest, and vertically through the troposphere.

Growth factor. An adjustment factor used to estimate the growth in a source's activity level between the inventory base year and a projected year. Valid values are 0.00 □ 99.99.

Heterogeneous, distributed computing environment. A heterogeneous computing environment consists of multiple machines of different types (e.g., supercomputers, graphics workstations, and workstation file servers). A distributed computing environment permits the execution of a large computational task to be shared across multiple machines linked together by a network. Thus, a heterogeneous, distributed computing environment consists of many different kinds of machines networked together.

Hydrostatic. Used to indicate to a model that it is to assume that the atmosphere is in hydrostatic equilibrium (i.e., surfaces of constant pressure and constant mass coincide and are horizontal throughout). Complete balance exists between the force of gravity and the pressure force.

Hypertext link. A hypertext link is a specially designated word or phrase displayed in text that has been saved in Hypertext Markup Language (HTML) format. A hypertext link provides nonsequential

access to other entries in a document set. In CMAQ, the Help facility is done using hypertext. Hypertext linking is done to all help entries.

Input/Output Applications Programming Interface (I/O API). A software library that reads and writes files. It uses an internal data format that is machine independent and that conforms to the widely used University Corporation for Atmospheric Research Network Common Data Format (netCDF). The I/O API files contain self-describing headers with complete information that is necessary to use and interpret the data contained in the file. The I/O API format is slightly more restrictive than the standard netCDF format regarding how the header information must be written. The I/O API library provides a variety of data structure types and a set of reusable access routines that offer selective direct access to the data in terms that are meaningful to the environmental modeler. Supported data types include gridded, boundary, vertical profile, grid nest, time series, and event-driven. For additional information on the I/O API, see Chapter 4.

‘”Internal.”’With respect to CMAQ data, internal means that the data are available within the software; the user does not have to provide them. Examples incude look-up tables, ranges, and lists of state/county names.

Inventory Data Analyzer (IDA). Program used for input and quality control checks of emission inventories.

Inventory. With respect to an emission processing system, inventory refers to a file or a database containing emission data for a specific set of pollutants for a specific time period (typically for the entirety of a specific year) for an area (country, states, counties).

Irix. Operating system for SGI computers.

Keyword. A keyword is a word or phrase, up to 40 characters long, that can be used to locate an entity in help text or a CMAQ object (e.g., dataset, program, study, or model) using a Find screen.

Layer collapsing. A procedure in which the layer structure prepared by a meteorological model is modified by reducing the number of layers. ‘**’This is not recommended.**”

Linux. An open-source, UNIX-like operating system. It is available and redistributable under the terms of the GNU Public License.

Makefile. A Makefile is a list of UNIX commands that perform activities such as mounting files and compiling source code.

Massively parallel processing. Computer processing employing multiple CPUs to execute multiple computational tasks simultaneously. Massively parallel systems employ a large number of CPUs simultaneously on the same task. In contrast, conventional computer design uses a single CPU to perform computational tasks in a strictly linear, sequential order.

Mesoscale. Medium scale. In meteorology, mesoscale denotes medium horizontal and vertical spatial scale. The horizontal scale extends to several hundred kilometers. The vertical scale extends from tens of meters to the depth of the troposphere.

Metadata. Information about data and about the processes that produced them. In particular, information about the data’s structure, internal characteristics and history, and location within a data storage environment, as well as information derived from the data.

Meteorological model. This type of model provides descriptions of atmospheric motions, momentum, moisture, heat fluxes, turbulence characteristics, clouds and precipitation, and atmospheric radiative characteristics. Most meteorological models currently in use for air quality modeling were originally developed for the prediction of weather. CMAQ models require information from a meteorological model that is designed to address specific issues relevant to air quality modeling, such as planetary boundary layer information, cloud distribution and mixing characteristics, precipitation, and surface fluxes.

‘”Mie scattering.”’A generalized particulate light-scattering mechanism that follows from the laws of electromagnetism applied to particulate matter.

Mixed-media. Simultaneously involving more than one environmental pollutant medium, such as air and water.

Model developer. Those scientists and software developers who construct and study theories to explain physical and chemical processes, and use computer models to study their theories.

Model users. Research, production, and quality assurance personnel (i.e., applied scientists and engineers) who generate valid input for the modeling systems, run the modeling systems, maintain audit trails, analyze input and output for correctness, and produce analyses of the modeling results.

Model. A representation of a planned or existing object or condition.

Modeling structure. A design specification that provides the paradigm of operation and the interface specifications for the modules used to construct a particular family of models. In a CMAQ model, for example, the paradigm is that modules act as operators upon a shared concentration field, and four types of interfaces (call interfaces, INCLUDE-file interfaces, I/O interfaces, and UNIX-environment interfaces) must be specified.

Modeling system. A set of computational models and associated data processors that together provide for the simulation of processes of interest.

CMAQ components. The various subsystems within the CMAQ framework. Each component is represented by its own icon. The available components are Dataset Manager, Model Builder, Program Manager, Science Manager, Strategy Manager, Study Planner, and Tools Manager.

CMAQ. The third-generation air quality modeling system. It is a flexible system that addresses multiple air quality issues, such as regional- and urban-scale oxidant and acid deposition.

Module. A subset that is part of a larger program (such as a meteorological model, an emissions model, or CMAQ). In a modular program, all modules of a particular type (e.g., those that compute dry deposition) are interchangeable, allowing you to replace one module with another to determine, for example, how each module affects modeling results. Examples of modules include science modules and analysis and visualization modules.

Monotonic. A quality of strictly increasing or decreasing within some interval.

Multilevel nesting. Multilevel nesting refers to employing nested grids within nested grids, possibly several levels deep.

National Emissions Inventory. A database at EPA containing the information about sources that emit criteria air pollutants and their precursors, and hazardous air pollutants.

Nested grids. Nesting refers to fitting a finer-resolution grid over part of a coarser-resolution grid. The finer-resolution grid receives information (such as boundary conditions) from the coarser-grid simulation.

Nonconforming datasets. Nonconforming datasets are ones that are not in I/O API format. They can be used in the CMAQ framework by programs that are specifically designed to read those datasets. When nonconforming datasets and programs are used, however, you must know how to match programs and datasets, and which data formats and programs are transportable to different machine architectures. Those considerations are automatically managed by the Models□3 framework for those who use conforming datasets and conforming programs.

Nonhydrostatic. Used to indicate that the model does not assume that the atmosphere is in hydrostatic equilibrium. Air is not assumed to have only horizontal motion relative to the earth.

Open-source software. Open-source software began as a marketing campaign for free software. OSS can be defined as computer software for which the human-readable source code is made available under a copyright license (or arrangement such as the public domain) that meets the "open source" definition. This permits users to use, change, and improve the software, and to redistribute it in modified or unmodified form. It is very often developed in a public, collaborative manner. Open-source software is the most prominent example of open-source development and often compared to user-generated content.

Operational models. These models offer fully functional modeling of relevant science processes, such as atmospheric, emissions, and chemical transport processes. They represent the current state-of-the-art that has undergone significant quality assurance review, peer review, and evaluation. The turnaround time for these models is generally much longer than for screening models but short enough to allow episodic studies.

Parameterize. To create an algorithm that describes the average large-scale behavior of a physical phenomenon, rather than describing the subgrid-scale behavior in terms of the underlying physics and chemistry. For example, a parameterized cloud algorithm might describe average cloud behavior over 80-km-square cells, although the individual clouds are much smaller that 80 km across.

Planetary boundary layer. The portion of the atmosphere adjacent to the earth's surface. This layer generally ranges from 100 m to 2 km in height, and is significantly influenced by surface effects (e.g., heating, friction). These effects can cause the layer to be well-mixed, which affects the distribution of pollutants in the air. (See also "troposphere.")

Popup window. A popup window is a special window for displaying an on-line help entry. The window opens when you select a specially designated hypertext link. Pop-up windows are useful for quickly displaying short, concise units of information. You cannot jump or step to other entries from a pop-up window.

Prepare. Read and process a file or a set of data.

Process analysis. Traces the source(s) of a chemical species within a simulation. One example of process analysis is determining whether a species concentration originated within the cell in which it is found or instead within some other cell(s). Another example is determining what chemical constituents were involved in creating a species produced by reaction (rather than transported from somewhere else).

Process. Read a file or a set of data, perform the desired functionality (quality control, reformatting, algebraic operations, etc.) and submit the processed data to the next set of actions.

Quality control (QC). The act of reading data (inventories, files) and checking for correctness, completeness, and consistency. QC may involve automatic correction, substitution, or the filling of missing data. All QC processes are followed by QC reports.

Register data. When you register data, you are making something that already exists (e.g., a file) known to the system.

Rule effectiveness percent. An adjustment to projected estimated emissions data to account for emissions underestimates due to compliance failures and the inability of most inventory techniques to include these failures in an emission estimate. The adjustment accounts for known underestimates due to noncompliance with existing rules, control equipment downtime, or operational problems and process upsets. Valid values: 0 to 100.

Rule penetration percent. An adjustment to projected estimated emissions data to account for emissions underestimates due to failure to implement rules throughout the area of intent. Valid values: 0 to 100.

Scalable. In the context of parallel computer architectures and algorithms, a parallel architecture or algorithm is termed scalable if its performance behaves linearly in terms of the number of processors employed. For example, doubling the number of processors does not cause new communications bottlenecks to appear but doubles the performance achieved.

Scale flexibility. The modeling system's ability to accurately simulate physical and chemical processes that describe atmospheric pollutants at different spatial and temporal scales. A modeling system with scalable algorithms for physical and chemical processes and with a generic grid system has this quality.

Science module. A component that is part of a modeling program (such as a meteorological model, an emissions model, or CMAQ) and that computes data for a discrete category of environmental phenomena (e.g., dry deposition, photochemistry, vertical advection).

Screening models. These models have simplified science processes designed for quick assessment studies. These models are employed when users are willing to sacrifice some accuracy for faster turnaround time. A typical screening study involves making a large number of runs in order to identify episodes or situations that warrant more detailed modeling.

Source Classification Code (SCC). The SCC system is used for classifying emissions sources at the level of individual processes (e.g., automobiles, boilers, solvent sources) within an emissions data inventory.

Source. With respect to air pollution, a point, area, or mobile source that produces and/or emits air pollutants.

Speciation. In CMAQ, speciation refers to using one of the chemical mechanisms available with CMAQ to disaggregate a chemical substance (pollutant) into simpler compounds.

Species. Typically, a chemical substance or group of related substances whose behavior is modeled during environmental simulation modeling.

Sub-grid-scale process. Physical process that occurs on a scale smaller than the grid resolution of the modeling system, such as point-source plumes and convective clouds. Since the scale is smaller than the grid resolution, these processes must be estimated or parameterized.

Summary report. Generally refers to an automatic, short report generated after the execution of a process.

Surface fluxes. The exchange of material, energy, or momentum between the surface of the earth and the atmosphere.

Time step. A time step is a fixed unit of time. A model may have one or more internal time steps for various processors. In the CMAQ framework, a time step is used to indicate the length of time between output of variables from the model or a process within the model. Another term might be "output time interval."

Troposphere. The troposphere is the lowest portion of Earth's atmosphere. It contains approximately 75% of the atmosphere's mass and almost all of its water vapor and aerosols. The average depth of the troposphere is about 11 km (7 miles) in the middle latitudes. It is deeper in the tropical regions (up to 20 km [12 miles]) and shallower near the poles (about 7 km [4 miles] in summer, indistinct in winter). Within the troposphere, temperature decreases with altitude. The lowest part of the troposphere, where friction with the Earth's surface influences air flow, is the planetary boundary layer (PBL). This layer is typically a few hundred meters to 2 km (1.2 miles) deep, depending on the landform and time of day. The border between the troposphere and stratosphere is called the tropopause. Above this layer is a temperature inversion—that is, in the stratosphere temperature increases with altitude.

Visualization. An important aspect of scientific computing that provides a method for presenting easily understandable data quickly and compactly in the form of charts and graphs.

---

[1] Future efforts toward fourth-generation systems will extend linkages and process feedback to include air, water, land, and biota to provide a more holistic approach to simulating transport and fate of chemicals and nutrients throughout an ecosystem

[2] The CVS ‘’modules”file has no intrinsic relationship with the CMAQ classes/module design implementation.