Operational Guidance for the Community Multiscale Air Quality (CMAQ) Modeling System

Version 4.7.1 (June 2010 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 27599

Operational Guidance for the Community Multiscale Air Quality (CMAQ) Modeling System

Disclaimer

Foreword

The Community Multiscale Air Quality (CMAQ) modeling system is being developed and maintained under the leadership of the Atmospheric Modeling and Analysis Division 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.

Contents

Figures

Tables

Abbreviations

AE	Aerosol chemistry phase
AIRS	Aerometric Information Retrieval System
AQ	Aqueous Chemistry Phase
ASCII	American Standard Code for Information Interchange
BC	Boundary Condition
BCON	Boundary Conditions Processor
BEIS3	Biogenic Emissions Inventory System
CB-IV	Carbon Bond-IV
CB05	Carbon Bond 2005
CCTM	CMAQ Chemistry-Transport Model
CEMPD	Center for Environmental Modeling for Policy Development
CMAS	Community Modeling and Analysis System
CMAQ	Community Multi-Scale Air Quality model
CO	Carbon Monoxide
CPU	Central Processing Unit of a computer
CSQY	Cross Section and Quantum Yield
CTM	Chemistry-Transport Model
CVS	Concurrent Versions System
EBI	Euler Backward Iterative chemistry solver
EPA	Environmental Protection Agency
FAQ	Frequently Asked Question
FTP	File Transport Protocol
GB	Gigabyte
GIS	Geographic Information System
GMT	Greenwich Mean Time
GPL	Gnu General Public License
GUI	Graphical User Interface
HTML	HyperText Markup Language
HYSPLIT	Hybrid Single-Particle Lagrangian Integrated Trajectory model
I/O API	Input/Output Application Programming Interface
IC	Initial Concentration
ICON	Initial Conditions Processor
IDA	Inventory Data Analyzer
IE	Institute for the Environment
IPR	Integrated Process Rate
IRR	Integrated Reaction Rate
IRR/MB	Integrated Reaction Rates/Mass Balance Analysis
JPROC	Photolysis Rate Processor
MCIP	Meteorology-Chemistry Interface Processor
MECH	Chemical Mechanism Reader
MEPSE	Major Elevated Point Source Emission
MM5	Fifth-Generation Penn State/NCAR Mesoscale Model

MP	Mechanism Processor
NAAQS	National Ambient Air Quality Standards
NASA	National Aeronautics and Space Administration
NCAR	National Center for Atmospheric Research
NET	National Emission Trends
netCDF	network Common Data Form
NO_x	Oxides of Nitrogen
NOAA	National Oceanic and Atmospheric Administration
NR	Non-Reactive
NWS	National Weather Service
PACP	Process Analysis Control Process
PAVE	Package for Analysis and Visualization of Environmental Data
PBL	Planetary Boundary Layer
PC	Personal Computer
PM2.5	Particulate matter up to 2.5 microns
PM10	Particulate matter up to 10 microns
PPM	Piecewise Parabolic Method
ppm	Parts per million (by weight)
ppmV	Parts per million (by volume)
PROCAN	Process Analysis Processor
QC	Quality Control
RADM	Regional Acid Deposition Model
SAPRC-99	State Air Pollution Research Center mechanism, version 1999
SAS	Statistical Analysis System
SCC	Source Classification Code
SCCS	Source Code Control System
SMVGEAR	Sparse Matrix Vectorized Gear
SO_2	Sulfur Dioxide
SPC	Species
TOMS	Total Ozone Mapping Spectrometer
TR	Tracer
UAM	Urban Airshed Model
UNC	University of North Carolina
UTC	Universal Time Coordinate
UTM	Universal Transverse Mercator
VOC	Volatile Organic Compounds
WRF	Weather Research and Forecasting model
WRF-ARW	Weather Research and Forecasting model – Advanced Research WRF

Page left intentionally blank

1. 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.[1]

The EPA Community Multiscale Air Quality (CMAQ) modeling system is a third-generation air quality model. It is available online at http://www.cmaq-model.org. 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 highly transparent and modular to facilitate the model's extensibility through community development by all members of the air quality modeling community.

[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

1.1 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 (i.e., *x-y-z* array that is fixed in space and covers a particular domain, i.e., the 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) (Skamrock 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 model to estimate the magnitude, location, and temporal variability of pollution sources. Open-source models such as the Sparse Matrix Operator Kernel Emissions (SMOKE) model (IE, 2008) (http://www.smoke-model.org/) and the Consolidated Community Emissions Processing Tool (CONCEPT) (http://www.conceptmodel.org/) are available for computing emissions inputs to CMAQ from annual, county-level 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 the 2005 update to the Carbon Bond mechanism (CB05) (Yarwood et al., 2005), and the Statewide Air Pollution Research Center, Version 1999

(SAPRC-99) mechanism (Carter, 1990, 2000). Additional details about the gas-phase chemistry in CMAQ are provided in Chapter 2 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.). It is possible to add new emissions species to CMAQ that are not supported in the distributed version of the software by using the chemical mechanism compiler (CHEMMECH) that is one of the CMAQ utility programs (see Chapter 5).

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 (http://www.baronams.com/products/ioapi/AA.html) to control the internal and external data flows to the model, and the network Common Data Form (netCDF) library (http://www.unidata.ucar.edu/software/netcdf) 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 an unprecedented level of flexibility for optimizing model performance for different air quality model applications.

A Note about the 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.

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 of 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 use of 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

1.2 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 five main CMAQ programs are

- The initial conditions processor (ICON)
- The boundary conditions processor (BCON)
- The clear-sky photolysis rate calculator (JPROC)
- The Meteorology-Chemistry Interface Processor (MCIP)
- The CMAQ Chemistry-Transport Model (CCTM)

Ancillary support programs distributed with CMAQ include

- The code builder/manager (M3BLD)
- The chemical mechanism compiler (CHEMMECH)
- The process analysis preprocessor (PROCAN)

Sections 1.2.1 through 1.2.3 describe the CMAQ system concept, followed in Section 1.2.4 by summaries describing the programs listed above.

1.2.1 Installation overview

All CMAQ source code except MCIP is distributed in a Linux tar file as a Concurrent Versions System (CVS) (http://ximbiot.com/cvs/wiki/) source code management archive. CVS is an open-source, network-transparent version control system for managing source code in a software development environment. The distributors of CMAQ use CVS to control access to the distributed source code; CVS maintains a protected version of the original source code and uses copies of the code when building executables. “Executables” are 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.

CVS must be installed on the user’s Linux system before installing CMAQ. 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 M3BLD controls the extraction of copies of CMAQ source code from CVS based on the configuration options specified by the user in UNIX C-shell scripts. After exporting the CMAQ source code from CVS, M3BLD 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.

As noted above, MCIP is not distributed in a CVS archive. Instead, when the CMAQ code is downloaded, MCIP appears in its own directory with source code and makefiles for installation.

1.2.2 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 simply 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. Discussions concerning which CMAQ options must be selected at compilation versus at execution are part of Chapter 5.

1.2.3 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 four other main programs that prepare input data for CCTM (i.e., ICON, BCON, JPROC, and MCIP). Before describing each of the CMAQ programs (Section 1.2.4), 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 ($\partial C/\partial t$) through the *continuity equation,* which is presented in simplified form below:

$$\partial C/\partial t = \text{Adv} + \text{Diff} + R_c + E_c - S_c$$

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. To model incoming solar radiation, which provides the energy source for photolysis reactions, the program JPROC calculates clear-sky photolysis rates at various latitude bands and hours based on solar hour angles. Output from these three 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.

1.2.4 Summary descriptions of the major CMAQ programs

The major CMAQ components and ancillary programs are described below. The listing is in the order that you, as a user, will run them. The last component listed, CHEMMECH, is for use by advanced users who wish to define a new or modified chemical mechanism for CMAQ

Model Builder (M3BLD) – The only CMAQ component written in the C programming language, M3BLD provides an interface to the CVS source code archive for exporting the source code, and to the Fortran 90 compiler for building binary executables. Because M3BLD 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

UNIX makefile. These are particularly useful for porting the CMAQ code to new operating systems, testing new code in a development environment, or troubleshooting problems with CMAQ compilation or execution.

Photolysis Rate Processor (JPROC) – JPROC calculates chemical-mechanism-specific clear-sky photolysis rates at fixed altitudes, solar hour angles, and latitude bands from tabulated absorption cross section and quantum yield (CSQY) data. While CMAQ is distributed with CSQY data that support the default chemical mechanisms, updating or adding new CSQY data is straightforward. The only configuration option required for JPROC is the selection of the chemical mechanism to use in the modeling. Output from JPROC is an ASCII look-up table of photolysis rates that CCTM uses to calculate gas-phase chemical transformations and pollutant concentrations.

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). 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.

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.

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.

CMAQ Chemistry-Transport Model (CCTM) – CCTM integrates the output from the preprocessing programs described above (JPROC, 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-definable 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, 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.

Process Analysis Preprocessor (PROCAN) – Process analysis is a technique used to trace the source(s) of a chemical species within a simulation. PROCAN generates Fortran INCLUDE files for building a version of CCTM that can calculate integrated process rates and/or integrated reaction rates (discussed in Section 6.3); these rates can then be used for diagnosing the chemical behavior of CMAQ simulations. This preprocessor uses an input configuration file to select the process analysis options desired, and outputs three INCLUDE files that are used to compile a version of CCTM that is instrumented for process analysis.

Chemical Mechanism Compiler (CHEMMECH) – This program creates chemical mechanism INCLUDE files for CMAQ from a mechanism definition file. Chemical mechanisms are represented in CMAQ through a series of INCLUDE files that contain mechanistic and kinetic parameters that describe a photochemical mechanism. CHEMMECH creates these INCLUDE 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 reactions' stoichiometry or kinetics in the existing mechanisms, and to add new species that require the addition of new reactions.

More detailed discussions on the formulations of the above CMAQ programs are available in Chapter 2 below, in Byun and Ching (1999), and in Byun and Schere (2006).

1.3 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 an unprecedented level of 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 (http://www.cmascenter.org). The CMAS Center is a support resource for users of CMAQ and other modeling systems; it is described in Section 11.1.
- Members of the large, international community of users connected through the CMAS Center can help each other by sharing data and experiences and providing technical support.

1.4 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 valuable, 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:

- 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.

1.5 New Features in Version 4.7

1.5.1 Version 4.7.1

CMAQ version 4.7.1 (CMAQv4.7.1) is an interim release of CMAQ that addresses bugs and expands the capabilities of CMAQ version 4.7. Table 1-1 summarizes the features in this version of the model.

1.5.2 Version 4.7

CMAQ version 4.7 (CMAQv4.7) contains many new features and improvements over the previous release of the model. Table 1-2 provides a summary of these changes. Details are contained in the release notes distributed with CMAQv4.7. Several processes that were previously modeled with other CMAQ components can now be calculated *in-line*, meaning that they can be simulated during the execution of the CMAQ calculation. An example is the photolysis rates calculation. In CMAQv4.7 users have the option of computing clear sky photolysis rates offline using the JPROC program or to calculate them in-line in the CCTM. For the offline approach, which has been the approach used for CMAQ up to this release, the outputs from JPROC are provided to the CCTM during execution. In the in-line approach, the CCTM reads in molecular cross section/quantum yield (CSQY) data and computes photolysis rates during execution.

Also, there were several options discontinued in CMAQv4.7:

- The Plume-in-Grid option (Ping)
- CB-IV chemical mechanism (replaced by CB05)
- Aero3 module and AE3 mechanisms (replaced by Aero4, Aero5, AE4, and AE5)
- Regional Acid Deposition Model (RADM) cloud module
- Stand-alone mercury version of CMAQ (this capability is included in the multipollutant version of CMAQ; see Table 1-1)

For the changes and new features included in the latest version of MCIP, please see the release notes for MCIP version 3.4.1.

1.6 About This Manual

This manual is an operational guidance document for users of the CMAQ modeling system, and is designed to support the installation, configuration, and execution of CMAQ on Linux systems. The users should be familiar with Linux scripting conventions and have some familiarity with the Fortran programming language. Users should also have some familiarity with atmospheric structure, and the physical and chemical processes occurring in the atmosphere.

Introductory and overview chapters present the CMAQ concepts, terminology, installation instructions, and guidance on running the tutorial simulation distributed with the model. The chapters on developing meteorology and emissions for CMAQ present the basic concepts and the availability of models that generate these CMAQ inputs, and describe how to configure these systems to produce input for CMAQ. In brief, the following chapters are included in this manual:

- Chapter 2 provides an overview of the science used in CMAQ
- Chapter 3 describes computer system requirements and installation
- Chapter 4 discusses the Input/Output Applications Programming Interface (I/O API)
- Chapter 5 describes the set of CMAQ programs and libraries
- Chapter 6 describes CMAQ input and output files
- Chapter 7 discusses how to define modeling grids, vertical layers, and chemical mechanisms
- Chapter 8 describes how to set up CMAQ for new simulations
- Chapter 9 provides CMAQ code management and development guidelines
- Chapter 10 describes analysis options for CMAQ output
- Chapter 11 describes how to obtain support for CMAQ.

Table 1-1. CMAQv4.7.1 Updates

Aqueous Chemistry	**Emissions**
• Mass conservation improvements ▪ Imposed 1 second minimum timestep for the remainder of the cloud lifetime after 100 "iterations" in the solver ▪ Force mass balance for the last timestep in the cloud by limiting oxidized amount to the available mass • Implemented steady-state assumption for OH • Only allow sulfur oxidation to control the aqueous chemistry solver timestep. Previously the reactions of OH, GLY, MGLY, and Hg also controlled the timestep	• Bug fix in EMIS_DEFN.F to include point source layer 1 NH_3 emissions • Bug fix to calculate soil NO "pulse" emissions in BEIS • Remove excessive logging of cases where ambient air temperatures exceed 315.0 K.
	Multi-pollutant Model Updates
Advection	• Bug fix to facilitate in-line emissions option • Mercury ▪ Dicarboxylic acid reduction mechanism applied to aqueous phase oxidized mercury specie ▪ Cl oxidation pathway based on Donohoue et al., 2005, J. Phys. Chem., A 109(34). • Lowered CB05 mechanism unit yields for acrolein from 1,3-butadiene tracer reactions to be consistent with laboratory measurements
• Added divergence-based constraints on advection timestep • Vertical advection now represented with PPM to limit numerical diffusion	
Horizontal Diffusion	**Photolysis**
• Error correction (T. Odman, Ga Tech) ▪ Concentration data may not be correctly initialized if multiple sub-cycle timesteps are required ▪ Fix to initialize concentrations with values calculated in the previous sub-timestep.	• JPROC/phot_table and phot_sat options ▪ Expanded lookup tables to facilitate applications across the globe and vertical extent to 20km ▪ Updated temperature adjustments for absorption cross sections and quantum yields ▪ Revised algorithm that processes TOMS datasets for OMI data format • In-line option ▪ Asymmetry factor calculation updated using values from Mie theory integrated over log normal particle distribution; added special treatment for large particles in asymmetry factor algorithm to avoid numerical instabilities
Model Timestep Determination	
• Error correction (T. Odman, Ga Tech) ▪ The sum of the advection steps for a given layer might not equal the output timestep duration in some cases ▪ Ensured that the advection timesteps sum up to the synchronization timestep	
	Parallel Processing (MPI)
Instrumented Models	• Processor 0 creates the netCDF I/O API file header when opening new output files ▪ On some parallel platforms, message-passing latency causes other processors to not have imminent access to the file header, resulting in a "hung" execution or a crash ▪ Fixed by deferring access by other processors until later in the computation
• DDM-3D • Sulfur Tracking • Primary Carbon Apportionment	

Table 1-2. CMAQv4.7 Updates

Aerosols	Chemistry
• Secondary organic aerosol (SOA) model enhancements ▪ Isoprene SOA ▪ Sesquiterpene SOA ▪ Polymerization ▪ Acid-catalyzed SOA ▪ NO_x-dependent SOA yields ▪ Enthalpy of vaporization ▪ In-cloud SOA formation pathways (glyoxal, methylglyoxal) • Changes in gas-phase chemistry mechanism, emissions speciation, and biogenic emissions model, to represent SOA precursors • Coarse PM ▪ Semivolatile inorganic components (NO_3^-, Cl^-, and NH_4^+) can condense and evaporate from the coarse mode, via dynamic mass transfer ▪ Nonvolatile sulfate can condense on the coarse mode ▪ Variable standard deviation of coarse mode size distribution ▪ Emissions of sea salt from the surf zone • Heterogeneous reaction probability ▪ Rederived parameterization based on Davis et al. (2008)	• HONO enhancements ▪ Heterogeneous reaction on aerosol and ground surfaces ▪ Emissions from mobile sources • Photolysis options (***beta versions***) ▪ In-line photolysis rate module, with aerosol feedback ▪ Photolysis rates adjusted using satellite-derived cloud information (currently table-approach only) • Aqueous chemistry ▪ Added two organic oxidation reactions (glyoxal, methylglyoxal) ▪ Updates to Henry's Law constants based on literature review • Base CB05 mechanism with CL2 chemistry • Multipollutant capability ▪ Include hazardous air pollutants (HAPs) and Hg in single modeling platform
Emissions	**In-Line Options**
• Speciation changes for HONO and benzene • Biogenic emissions: added sesquiterpene emissions • Sea-salt emissions ▪ Updated flux parameterizations & surf zone emissions ▪ Spatial Allocator can be used to produce the ocean file	• Dry deposition ▪ Moved calculation into CCTM • Emissions ▪ Integrated the Biogenic Emissions Inventory System (BEIS) into CCTM ▪ Incorporated plume rise into CCTM • Bidirectional NH_3 and Hg surface flux ▪ Fertilizer NH_3 emissions will be applied through the flux model (under development)
Diagnostic Tools	**Clouds**
• Decoupled Direct Method (DDM) • Primary carbon source apportionment model • Sulfate tracking model	• Convective cloud model ▪ Revised to reduce layer configuration differences ▪ Changed the integration time step • Resolved cloud model • Correction in precipitation flux calculation

1.7 References

Byun, D., and J. K. S. Ching, 1999: Science Algorithms of the EPA Models-3 CommunityMultiscale 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.

Carter, W. P. L., 1990: A detailed mechanism for the gas-phase atmospheric reactions of organic compounds. *Atmos. Environ*., **24A**, 481–518.

Carter, W. P. L., 2000: Implementation of the SAPRC-99 chemical mechanism into the Models-3 Framework. Report to the U.S. Environmental Protection Agency, 29 January 2000. [Available online at http://pah.cert.ucr.edu/~carter/reactdat.htm.]

IE (UNC Institute for the Environment), 2008: SMOKE version 2.5 Users Manual. [Available online at http://www.smoke-model.org/]

Davis, J.M., P.V. Bhave, and K.M Foley, 2008: Parameterization of N2O5 reaction probabilities on the surface of particles containing ammonium, sulfate, and nitrate. Atmos. Chem.Phys. **8**, 5295-5311.

Grell, G. A., J, Dudhia, and D. R. Stauffer, 1994: A Description of the Fifth Generation Penn State/NCAR Mesoscale Model (MM5). NCAR Technical Note NCAR/TN-398+STR, 138 pp.

Seinfeld, J., and S. Pandis, 1998: Atmospheric Chemistry and Physics: From Air Pollution to Climate Change. John Wiley & Sons, New York, NY.

Skamarock, W. C., J. B. Klemp, J. Dudhia, D. O. Gill, D. M. Barker, W. Wang, and J. G. Powers, 2005: A Description of the Advanced Research WRF Version 2. NCAR Technical Note NCAR/TN-468+STR, 88 pp.

U.S. EPA, 2008: National Ambient Air Quality Standards (NAAQS). [Available online at http://www.epa.gov/air/criteria.html.]

Yarwood, G., S. Rao, M. Yocke, and G. Whitten, 2005: Updates to the Carbon Bond chemical mechanism: CB05. Final Report to the U.S. EPA, RT-0400675. [Available online at www.camx.com.]

2. Overview of the Science in the CMAQ Modeling System

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"), visibility, and acidic and nutrient pollutant species. CMAQ uses state-of-the-science techniques and has many new and important features that are not available in previous modeling systems. 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.

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 2-1). 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.

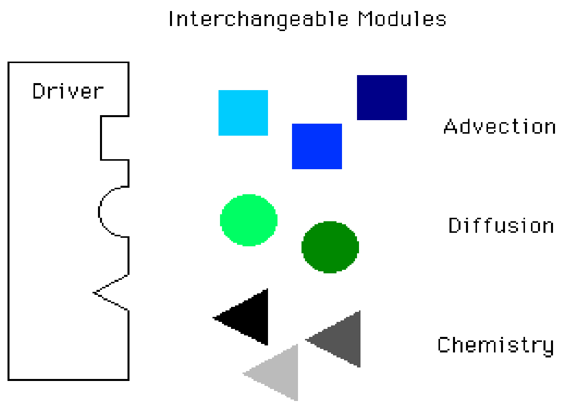


Figure 2-1. CMAQ Modeling Framework

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 describes features of the modeling system that create its adaptability and flexibility (Section 2.1). Sections 2.2 and 2.3 then provide brief descriptions of the key air quality modeling science features in various components of the CMAQ system, including MCIP, ICON, BCON, JPROC, CHEMMECH, PROCAN, and CCTM. More detailed discussions on these features can be found in Byun and Ching (1999) and Byun and Schere (2006). Finally, Section 2.4 discusses the CMAQ user interface for building and running CMAQ.

2.1 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. For example, linkages of CMAQ with exposure modeling, hydrological modeling, climate modeling, and multimedia modeling are currently being explored.

2.1.1 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. The failure of the early, simplistic approach of trying to account for the chemistry of sulfur dioxide from power plants without also treating ozone and hydroxyl radicals demonstrated the need for the "one-atmosphere' approach.

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 vertical coordinates can easily be accommodated and multiple scales can be simulated with the same CTM. The Jacobian matrix used by the generalized coordinate system controls the necessary grid and coordinate transformations (consult Byun, 1999).

2.1.2 Modular flexibility

CMAQ's current coding structure is based on a modularity level that distinguishes from each other the 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 (except for gencoor)are discussed further later in this chapter.)

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 (*chem*)
- Aqueous-phase reactions and cloud mixing (*cloud*)
- Aerosol dynamics and size distributions (*aero*)

Control/Utility Modules:

- Model data flow and synchronizing of fractional time steps (*ctm*)
- Unit conversion (*gencoor*)
- Initialization (*init*)
- Process analysis (*pa*)

Data Estimation Modules:

- Aerosol deposition velocity estimation (*aero_depv*)
- Photolytic rate computation (*phot*)

This modularity makes it easier to modify or introduce a specific scientific process in CCTM. For example, the *chem* 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. In this situation, failing to make the correct modification at even a single location in the source code could lead to erroneous modeling results or program execution failure. CMAQ's modularity makes designing or modifying a model simulation much more user friendly, requiring either less reprogramming or none at all.

2.1.3 Major 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.

Another CMAQ feature that supports quality control is process analysis, which (as noted earlier) is a technique used to trace the source(s) of a chemical species within a simulation. Process analysis is discussed in Section 2.2.5.

2.2 CMAQ Input Processors

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

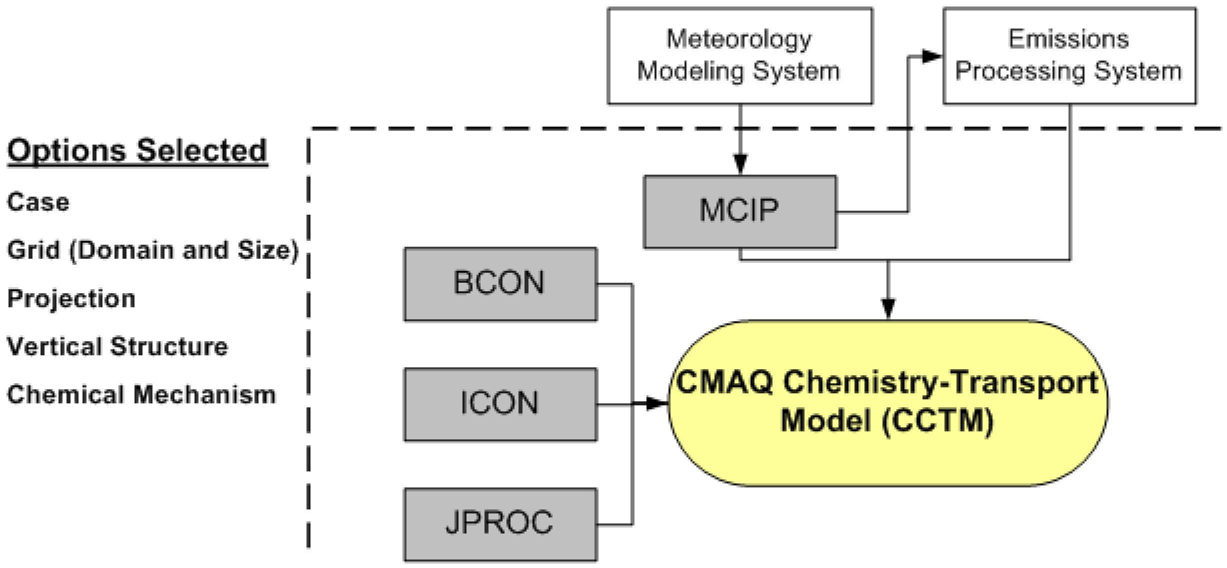


Figure 2-2. CMAQ Chemistry-Transport Model (CCTM) and input processors

The input data for CCTM are developed using the four input processors shown in grey in Figure 2-2. All of the CMAQ programs shown in Figure 2-2 (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 5.

CMAQ uses the MCIP processor to prepare the meteorological fields for CCTM. The ICON and BCON processors generate the initial and boundary conditions for a CCTM simulation. JPROC computes the photolysis rates that are used when simulating photochemical reactions in CCTM. Emissions for CMAQ must be prepared with a modeling system that generates emissions for direct input to CCTM (currently, SMOKE or CONCEPT). Brief descriptions of the various CMAQ input processors are presented in this section. Also described are two processors (CHEMMECH and PROCAN) not shown in Figure 2.1.

2.2.1 MCIP: Meteorology-Chemistry Interface Processor

MCIP uses output files from a meteorological model (such as MM5) to create netCDF-formatted input meteorology files that are used by the emissions model that computes emissions inputs to CMAQ, and by CCTM within CMAQ (Figure 2-3). (The files created by MCIP are not listed individually here.)

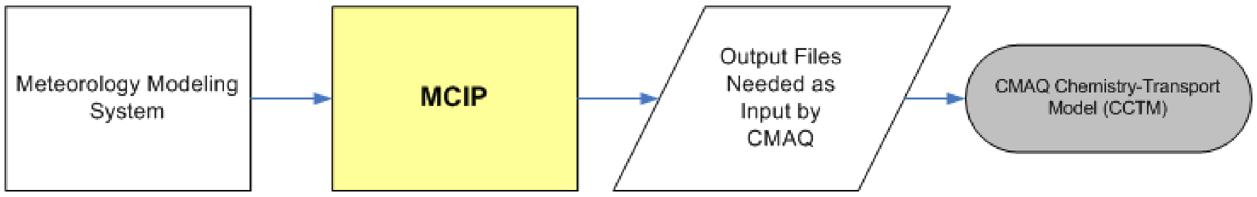

Meteorology Modeling System
MCIP
Output Files Needed as Input by CMAQ
CMAQ Chemistry-Transport Model (CCTM)

Figure 2-3. Meteorology preprocessing for CMAQ with MCIP

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

- Extracts meteorological model output that is specific to the CCTM horizontal grid domain. CCTM uses a smaller computational domain that is entirely contained within the computational domain of the meteorological model, and the lateral boundaries from the meteorological model are generally not used by CCTM.

- Processes all required meteorological fields for CCTM and the emissions model. The meteorological information, such as atmospheric temperature, pressure, humidity, and winds produced by the meteorological model, is put into a form required by CMAQ.

- "Collapses" meteorological model fields, if coarser vertical resolution data are desired for a given CCTM run. (*This is not a recommended procedure*). To do this, MCIP uses mass-weighted averaging on higher-vertical-resolution meteorological model output.

- Computes dry-deposition velocities for important gaseous species. Dry deposition is the deposition of pollutants from the air onto the surface of the earth. The rate of this removal is determined by various chemical, physical, and biological factors. In addition, dry deposition depends on the type of pollutant, the amount of turbulence or mixing in the atmosphere, and the nature and type of the receiving surface (information describing this surface is generated by MCIP). One measurement used in simulating the dry deposition of pollutants is the dry-deposition velocity; this is analogous to the settling velocity of particles due to gravity, and it is useful in determining surface fluxes. MCIP can compute dry deposition using the surface exchange aerodynamic method (Pleim et al., 2001). This method uses surface resistance, canopy resistance, and stomatal resistance to compute dry deposition velocities. The user also has the option to turn off the MCIP dry-deposition velocity calculation completely; this option is used to process meteorology data for CMAQ simulations that use the in-line dry deposition velocity calculations contained in CCTM.

- 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, as discussed in Section 2.2.5 (Walcek and Taylor, 1986; Chang et al., 1987). First, the cloud base is determined as the lifting condensation level computed from the highest saturated equivalent potential temperature below 700 mb. Then, the cloud top is computed by following a saturated adiabatic lapse rate from cloud base until it reaches a temperature five degrees cooler than the surrounding environment. Once the top and bottom of the cloud are determined, MCIP constructs a vertical profile of the adiabatic liquid water mixing ratio as the difference between the saturated mixing ratio at each level and the source-level mixing ratio. MCIP obtains the cloud coverage fraction by iteratively solving the equations governing the conservation of total water mass and energy conservation for cloud-top mixing, commensurate with the temperature profile.

- Outputs meteorological and geophysical files in the netCDF format, for input to SMOKE and CMAQ.

2.2.2 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 of each processor, ICON and BCON can generate these concentrations for all the chemical species required by CMAQ.. These processors require two inputs (Figure 2-4): 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. The current default profiles available in CMAQ are in RADM2 speciation. 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.

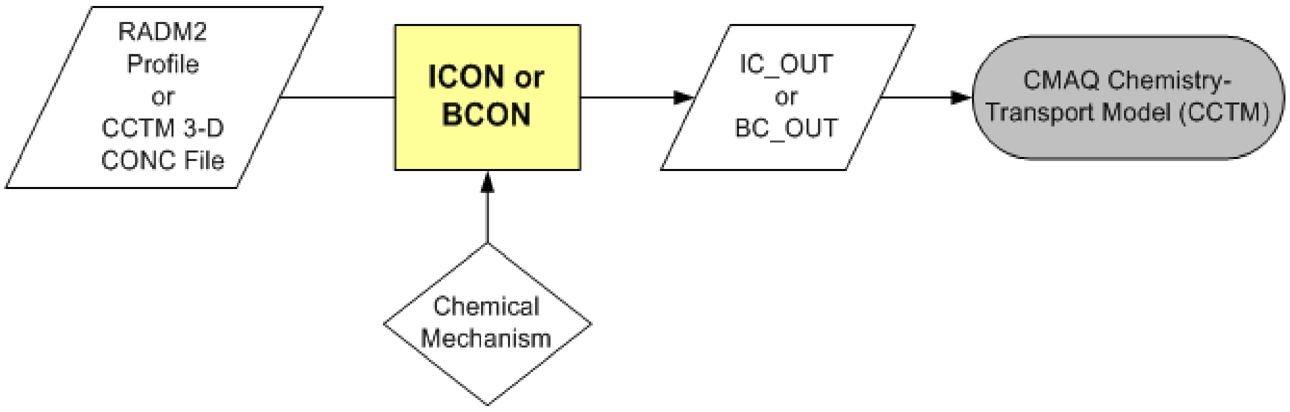

RADM2 Profile or CCTM 3-D CONC File
ICON or BCON
IC_OUT or BC_OUT
CMAQ Chemistry-Transport Model (CCTM)
Chemical Mechanism

Figure 2-4. 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. Figure 2-4 assumes that the RADM2 chemical mechanism was used to generate the default vertical profiles; this is consistent with the profiles distributed with the CMAQ version 4.7 tutorial dataset. Existing CCTM 3-D concentration fields could have been generated using several different chemical mechanisms. The chemical mechanism used in CCTM and CMAQ input processors must be consistent with the mechanism used to generate the concentration fields input to ICON and BCON. Both of these processors can perform mechanism conversions from one photochemical mechanism to another. The default conversion in ICON and BCON maps RADM2 species to either CB05 or SAPRC-99 speciation. Customized mechanism conversions can be coded into ICON and BCON through a Fortran INCLUDE file that maps the species from one photochemical mechanism to another.

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.

2.2.3 JPROC: Clear-sky photolysis rate calculator

For CMAQ, the photolysis rate model, JPROC, is used to generate clear sky photodissociation reaction rates. JPROC requires temperature profiles from the *U.S. Standard Atmosphere* document (NOAA, 1976), a profile of the aerosol extinction coefficients (Elterman, 1969), data on species cross sections and quantum yields (CSQY), extraterrestrial radiance (ET), and standard seasonal profiles of ozone. JPROC can optionally use ozone column totals from the NASA Total Ozone Mapping Spectrometer (TOMS) satellite to produce the photolysis rates for CCTM (Figure 2-5).

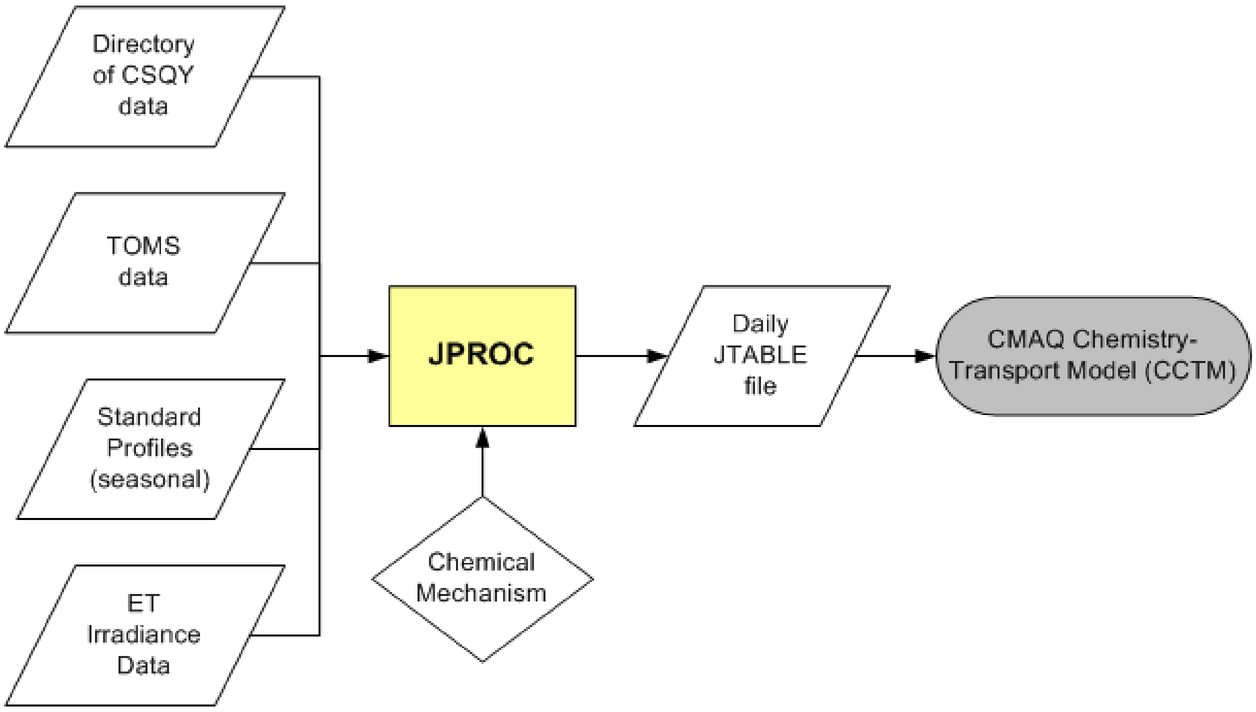


Figure 2-5. Clear sky photolysis rate preprocessing for CMAQ

JPROC uses this information in a radiative transfer model to calculate the actinic flux (photons cm^{-2} min^{-1}) needed for calculating photolysis rates. Currently, JPROC calculates the actinic flux for clear-sky conditions (no clouds present), and CCTM then attenuates for cloudiness when clouds are present. JPROC computes the rate for each photolysis reaction at various latitudes, altitudes, and zenith angles. Within CCTM, the subroutine PHOT interpolates the data generated by JPROC to individual grid cells, and adjusts for the presence of clouds. A beta version of an in-line photolysis module has been included with v4.7 but it is not yet a part of the official release.

2.2.4 CHEMMECH: Chemical mechanism compiler

As released, CMAQ includes all necessary chemical mechanism information for a series of pre-configured atmospheric chemistry parameterizations. All the beginning user has to do is choose which mechanism is desired, as explained in Section 7.4. Advanced users who wish to generate new chemical mechanism information for use in CMAQ can use the program CHEMMECH to convert a mechanism listing file into the files needed by CMAQ.

Gas-phase chemical mechanisms are implemented in CMAQ using Fortran INCLUDE files. These files are in a machine-readable ASCII format and include all of the mechanism parameters required, including gas-phase species, reaction stoichiometry, and kinetics information. To invoke chemical mechanisms in CMAQ, these files are included in the compilation of the various CMAQ programs to generate mechanism-specific executables. CHEMMECH takes a mechanism definition file, often named “mech.def”, and generates the mechanism and species INCLUDE files—RXDT.EXT, RXCM.EXT, and SPC.EXT—that define the chemistry parameters for the CMAQ programs. The file “mech.def” is an ASCII file that is easy to understand and modify. Figure 2-6 shows the relationship between CHEMMECH and other parts of the CMAQ modeling system.

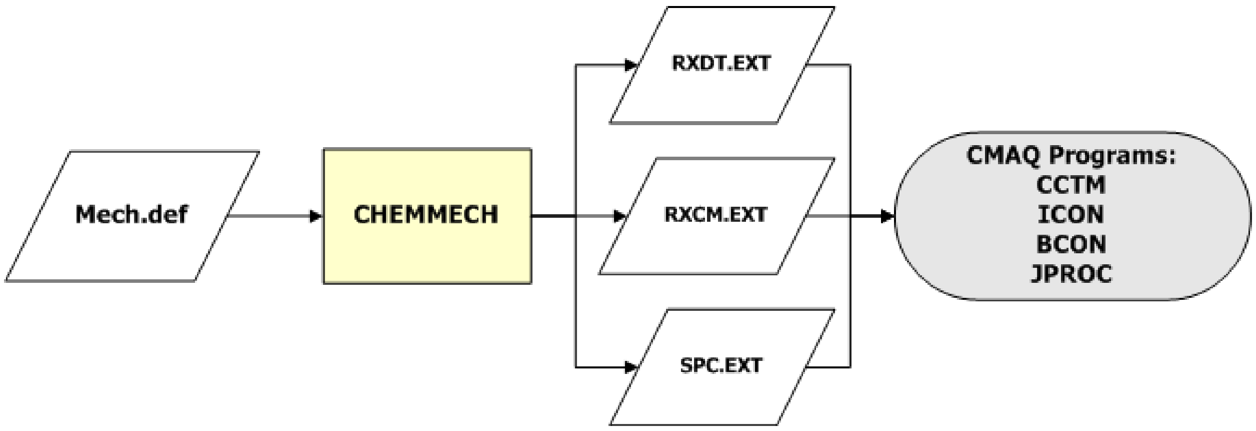


Figure 2-6. Invoking new/modified gas-phase chemical mechanisms in CMAQ

2.2.5 PROCAN: Process-analysis preprocessor

Process analysis (PA) is an accounting system that tracks the quantitative effects of the individual chemical and physical processes that combine to produce the predicted hourly species concentrations output from a CTM simulation. The CMAQ CTM and other Eulerian grid models output concentration fields that are solutions of systems of partial differential equations for the time-rate of change in species concentrations due to a series of individual physical and chemical processes; these results are then combined to obtain a cumulative hourly concentration. PA tracks the mass throughput of these individual processes and provides quantitative information about how each process affected the predicted hourly species concentrations. PA is an optional configuration in CMAQ that is implemented by compiling CCTM with Fortran INCLUDE files that define PA configuration options. A PA-instrumented version of CCTM outputs additional files that contain hourly concentrations by physical and/or chemical process for selected model species.

PROCAN is the PA preprocessor in the CMAQ modeling system. As shown in Figure 2-7, PROCAN takes as input a configuration file (Pacp.inp) that is used to define the model species and processes to be tracked using PA, and outputs three INCLUDE files (PA_CMN.EXT, PA_CTL.EXT, and PA_DAT.EXT) that are used when compiling CCTM.

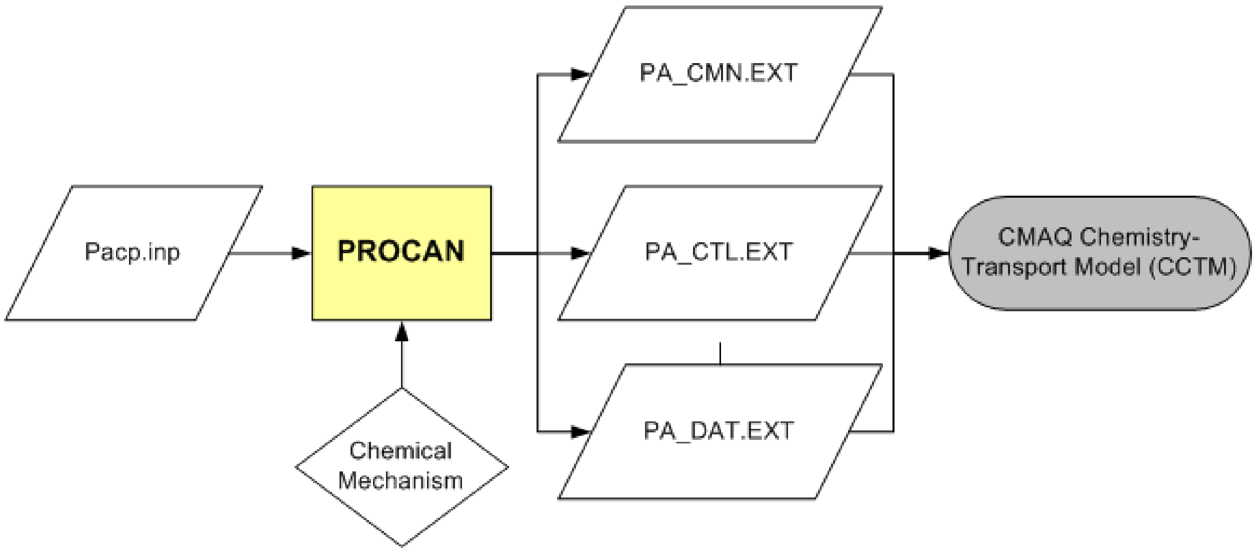


Figure 2-7 Process analysis implementation in the CMAQ modeling system

2.3 CMAQ Chemistry-Transport Model Science Modules

Figure 2-8 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 package. To perform a model simulation, CMAQ needs input information, 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 in the following subsections.

2.3.1 Gas-phase chemistry solvers

Various modules for simulating tropospheric gas-phase chemistry within CMAQ have been developed, ranging from simple linear and nonlinear systems for engineering model prototypes to comprehensive chemistry representations for detailed chemical pathways related to chemical transformation of atmospheric pollutants.. In CMAQv4.7, gas-phase chemistry can be simulated with the CB05 or SAPRC-99 photochemical mechanisms. Because of CCTM's modularity, advanced users can modify the existing photochemical mechanisms, or even add new ones. To compute time-varying species concentrations and their rate of formation or depletion, differential equations governing chemical reaction kinetics and species conservation need to be solved for the entire species set. CCTM uses special techniques called numerical chemistry solvers to compute these concentrations and rates at each time step.

Regarding these solvers, various solution algorithms for the chemical kinetics have been investigated in terms of the optimal balance of accuracy, generalization, and computational efficiency that is required for this component of the atmospheric system. CCTM currently contains three options for solving gas-phase chemical transformations: 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).

CMAQv4.7 includes options for simulating the chemistry of chlorine, mercury, and other toxic compounds in a multipollutant version of the CB05. See the CMAQv4.7 release notes for addition details on the gas-phase chemistry options.

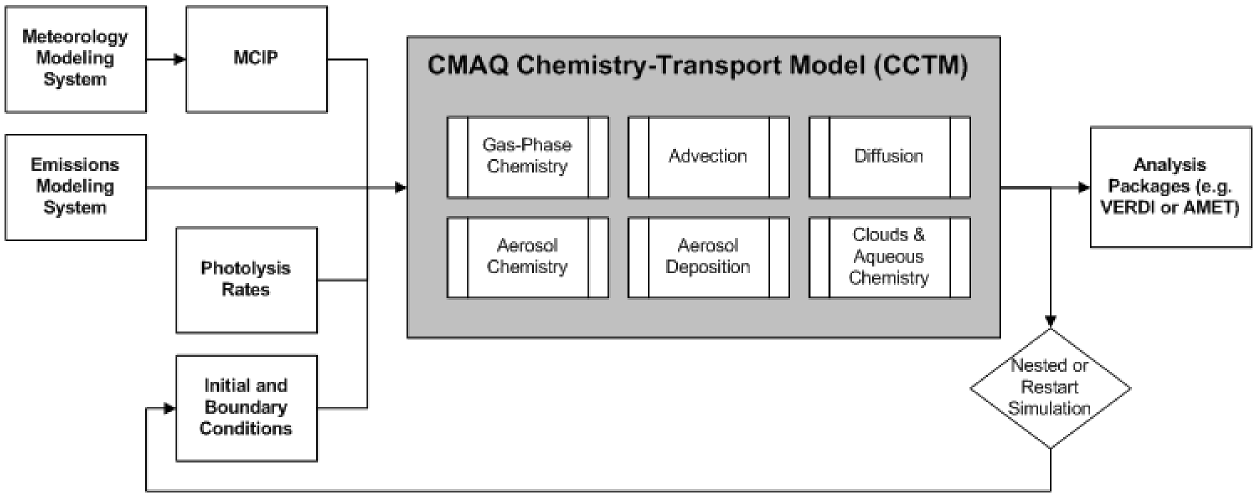

Meteorology Modeling System
MCIP
Emissions Modeling System
Photolysis Rates
Initial and Boundary Conditions
CMAQ Chemistry-Transport Model (CCTM)
Gas-Phase Chemistry
Advection
Diffusion
Aerosol Chemistry
Aerosol Deposition
Clouds & Aqueous Chemistry
Analysis Packages (e.g. VERDI or AMET)
Nested or Restart Simulation

Figure 2-8. CMAQ chemistry-transport model and associated preprocessors

2.3.2 Photolysis

Photolysis (or photodissociation) of trace gases initiates most chemical reactions that take place in the atmosphere. Photolysis splits gas-phase chemical species using energy from sunlight. Photolysis is involved in the formation of smog, an air pollution problem that affects human, animal, and plant health. Simulating photochemical reactions accurately is therefore a key issue that strongly influences air quality model performance.

CCTM uses state-of-the-science techniques to simulate photolytic reactions in the PHOT module. Photolysis reactions and their rates of reaction are driven by sunlight. Similar to kinetic reaction rates for nonphotochemical reactions, the photolysis rate quantifies how much reactant is produced from a photolytic reaction in a given amount of time. The calculation of a photolysis rate must include multiple influences:

The rate of photolysis is a function of the amount of solar radiation (called actinic flux), which varies based on the time of day, season, latitude, and terrestrial features. The amount of solar radiation is also affected by the amount of cloudiness and by aerosol absorption and scattering in the atmosphere.

The photolysis rate also depends on species-specific molecular properties like the absorption cross section (the effective molecular area of a particular species when absorbing solar radiation, which results in a shadow region behind the particle) and quantum yield (the number of molecules that dissociate for each light photon incident on the atmosphere). These molecular properties depend on the wavelength of the incident radiation and the temperature (and hence, on the available photon energy). Thus, estimating the photolysis rate is further complicated by these temperature and wavelength dependencies.

As discussed in Section 2.2.3, the CMAQ modeling system includes an advanced photolysis model (JPROC) to calculate temporally varying photolysis rates for use in simulating photolysis in CCTM.

An in-line photolysis module (Binkowski et al., 2007) was included in CMAQ beginning with version 4.7. The in-line photolysis calculations account for the presence of ambient PM and ozone predicted by CMAQ and use these estimates to adjust the actinic flux, rather than relying on a look-up table of static background PM and ozone values. The in-line method for calculating photolysis rates also uses newer values of the absorption cross sections and quantum yields than those in the table look-up version, and these new values are corrected for ambient temperature. The extinction coefficients for the ambient aerosols are explicitly calculated using a new, efficient parametric algorithm derived from explicit integration of Mie calculations over the lognormal size distributions. Refractive indices are calculated based upon the categories of chemical species present (i.e., water soluble, insoluble, soot-like, water, and sea salt).

2.3.3 Advection and diffusion

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.

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 module 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. 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.

Note that the CMAQ v4.5 advection scheme, with the same horizontal advection but using PPM for the vertical velocity component, is still available in the 2008 release, along with the mixing ratio correction step.

Diffusion: In CCTM, vertical diffusion is represented by the Asymmetric Convective Method (ACM) of Pleim and Chang (1992). ACM2, an updated version of ACM, is used in v4.7. 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. In CMAQv4.7, 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. CMAQv4.7 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 in v4.7 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.

2.3.4 Particulate matter (aerosols)

Within the air quality community, atmospheric aerosol particles are referred to as particulate matter (PM). PM can be either primary or secondary. Primary PM is emitted directly into the atmosphere from natural or anthropogenic (man-made) sources. Secondary PM is formed in the atmosphere, either from precursors that react chemically to form new particles, or from vapor-phase species that condense or deposit onto primary particles that are already present in the atmosphere. 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 lognormal subdistributions, or modes. Two interacting modes (Aitken and accumulation) represent $PM_{2.5}$ (particulate matter of diameter equal to or less than 2.5 microns). A coarse mode represents PM with diameters greater than 2.5 microns and equal to or less than 10 microns. Thus, PM_{10} is modeled as the sum of the $PM_{2.5}$ and coarse-mode PM.

Particulate matter is deposited to the ground by wet or dry deposition, both of which are modeled by CMAQ.

In wet deposition, PM is transferred by rainfall. Wet deposition is calculated within CMAQ’s cloud module.

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. In v4.7, the dry deposition algorithm has been modified to include an impaction term in the coarse and accumulation modes.

In CMAQv4.7, several new pathways for secondary organic aerosol (SOA) formation have been implemented, based on the recommendations of Edney et al. (2007) and the recent work of Carlton et al. (2008). New SOA precursors include isoprene, sesquiterpenes, benzene, glyoxal, and methylglyoxal. The enthalpies of vaporization of these species have been revised based on recent laboratory data (Edney et al., 2007). In previous versions of CMAQ, all SOA was treated as semivolatile. In CMAQv4.7, four types of nonvolatile SOA are simulated.

There is a new 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). In a future release, this methodology will also be applied to ammonia fluxes to the surface. Further discussion is available in the CMAQv4.7 release notes. The release notes also contain more information about all of the other scientific improvements to the representation of PM that are included in v4.7.

Another type of output available from CMAQ is 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, there are three new variables (PM25AT, PM25AC, and PM25CO) that are the fractional amounts of the Aitken, accumulation, and coarse modes, respectively, that are composed of particles less than 2.5 microns in aerodynamic diameter (Jiang et al., 2006).

In the near future, two PM diagnostic tools will be available in an update to CMAQv4.7: one for tracking the sulfate budget, and one for tracking the sources of elemental and primary organic carbon that were part of CMAQv4.6.

2.3.5 Clouds and aqueous-phase chemistry

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 v4.7 convective cloud mixing algorithm corrects a tendency to predict excessive transport from upper layers in the cloud to sub-cloud layers. In v4.7, a version of ACM is used to model convective clouds.

2.3.6 Process analysis

As discussed in Section 2.2.5, CCTM also includes a process analysis (PA) module. Process analysis is a technique for separating out and quantifying the contributions of various individual physical and chemical processes to the changes in the predicted concentrations of a pollutant. This makes PA useful for conducting quality assurance procedures on a model run. With the information PA provides, compensating or unresolvable errors in the model or input data can be identified even if they are not 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 occurrence could be masked in the final predicted concentrations if the negative values are more than compensated for by larger positive values resulting from the chemistry calculations.

In addition to its role in the quality control of air quality modeling runs, PA has other important applications:

It is a very strong analysis tool for identifying the relative importance of processes (chemistry, advection, diffusion, etc.) that change pollutant concentrations.

As a tool for model development, PA can help evaluate the effect of modifications made to a model or process module.

As a tool for regulatory decision-making, PA can help determine whether a decision to control a specific type of emission would produce the desired results, and if the answer is no, it can help determine another type of control that would be more effective.

Note that CMAQ with process analysis will not work with the EBI chemical solver (discussed in Section 2.3.1). The user is required to use either the Rosenbrock or the SMVGEAR solver.

2.4 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 a series of environment variables that are required by M3BLD (described in Section 1.2.4) or by the CMAQ program executables. The scripts can be adapted to work with any Linux shell scripting language (e.g., Bash, Bourne).

Tar files for CMAQ can be downloaded from the CMAS Center web site. The MODELS tar file contains all of the source code for CMAQ, and the SCRIPTS tar file contains all of the C–shell scripts for building and executing the individual programs. Each of CMAQ's programs has separate build and run scripts. The build script is used to check copies of the source code out of a CVS repository and to compile the program; the run script is used to set the required environment variables and run the program. The user can manipulate the CMAQ scripts using a Linux text editor such as *vi* (http://www.vim.org/) or *nedit* (http://www.nedit.org/). 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 Chapter 3, with further details in Chapter 5.

The CMAS Center (see Section 11.1) currently fully supports CMAQ on Linux systems using the Portland Group and Intel Fortran compilers. The CMAS Center no longers supports compiling and running CMAQ on other UNIX operating systems. In any event, good computing practice suggests that the *same* Fortran compiler be used to compile all components of the CMAQ system, including the netCDF and I/O API libraries.

2.5 References

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 Multscale 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,

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.

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.

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 $PM_{2.5}$ concentrations and their impact on model performance evaluation in the lower Fraser valley. *Atmos. Environ*., **40**, 4973–4985.

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., 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.

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.

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

3. CMAQ SYSTEM REQUIREMENTS AND INSTALLATION

This chapter provides recommended hardware configurations and software requirements for installing and running CMAQ. The hardware configurations in particular are subject to change with each new release of CMAQ and with the development of new computing technologies. The installation instructions in this chapter guide the user through obtaining the CMAQ source code and installing it on your system. Brief instructions for running the CMAQ benchmarkcase 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 7.

3.1 System Recommendations

All of the CMAQ programs are written in Fortran (except for M3BLD, written in C) 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. In light of the dynamic nature of the computer hardware world, the specifications listed in this section are current recommendations, not requirements. While there are minimum levels of processing power and disk capacity needed for running CMAQ, there is no single system on which CMAQ is optimized to run. The flexibility of the modeling system enables users to optimize CMAQ for most current hardware configurations.

CMAQ is distributed and supported for executing on Linux operating systems with the Intel or Portland Group Fortran (PGF) compilers. CMAQ can be ported to most computers running Linux. Documented configurations include the SGI Altix, Red Hat Enterprise, Fedora, Mandrake, MacOSX, and Suse operating systems. In addition to the Intel and PGF compilers, CMAQ has been built with Sun and gfortran compilers. Information about these ports and up-to-date hardware recommendations are available through the CMAS Center web site (http://www.cmascenter.org). The hardware recommendations provided below are the minimum level required to run CMAQ. The software recommendations may be considered as "requirements," as all of the necessary source code libraries and utilities needed for running CMAQ are listed.

3.1.1 Hardware

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

- PC with a single 1.0 GHz processor with a Linux operating system OS
- 1 GB RAM
- 10 GB hard drive storage (Note: the benchmark simulation requires 3 GB of free storage capacity).

Below are two examples of optimal hardware configurations for running CMAQ on multiple processors in a production environment:

Optimal CMAQ Hardware Solution #1

- 4 dual-CPU 2.8 GHz Xeon IBM BladeCenter rack-mounted nodes with Red Hat Enterprise Linux OS
- 2 GB RAM per node
- Gigabit Ethernet network
- 1.5 TB hard drive storage
- Super DLT 110 GB tapes for system backups
- Uninterruptible power supply (UPS)

Optimal CMAQ Hardware Solution #2

- 8 dual-CPU 2.5 GHz AMD Athlon MP 2000+ PCs with Ubuntu Linux O/S
- 2.0 GB RAM per PC
- Gigabit Ethernet network
- 80 GB system storage
- 10 TB IDE-SCSI RAID 5 array
- UPS

3.1.2 Software

CMAQ requires all of the programs listed in Table 3-1; this list includes the programs distributed with CMAQ. Table 3-2 lists additional utility software that is not required for running CMAQ, but is useful for model diagnostics and evaluation. Note that MPICH needs to be installed by the system administrator because this program is specific to the computing platform.

Table 3-1. Software required for running CMAQ

Software	Description	Source
CMAQ Programs		
M3BLD	Models-3 program builder for source code management and code compilation	Contained in the standard CMAQ distribution available at http://www.cmascenter.org
JPROC	Photolysis rate preprocessor	
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	
PROCAN	Process analysis preprocessor for setting up CMAQ to generate integrated reaction rates or integrated process rates	
Compilers		
IFORT	Intel Fortran 90 compiler	http://www.intel.com
PGF90	Portland Group Fortran 90 compiler	http://www.pgroup.com/
GCC	Gnu C compiler	http://gcc.gnu.org/
Code libraries		
STENEX	CMAQ stencil exchange library for parallel job management	Contained in the standard CMAQ distribution available at http://www.cmascenter.org
PARIO	CMAQ parallel input/output management library	
MPICH	Library for the message passing interface; used for multiprocessor CMAQ simulations	http://www-unix.mcs.anl.gov/mpi/mpich1/
netCDF	Network Common Data Form library for controlling CMAQ file formats	http://my.unidata.ucar.edu/content/software/netcdf/index.html
I/O API	Input/Output Application Programming Interface for controlling internal and external communications	http://www.baronams.com/products/ioapi/
Source code management		
CVS	Concurrent Versions System for managing the distributed archive of the CMAQ source code	http://ximbiot.com/cvs/cvshome/

Table 3-2. Optional support software for CMAQ

Software	Description	Source
Evaluation and visualization 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/
I/O API Tools	Postprocessing tools for manipulating data in the I/O API/netCDF format	http://www.baronams.com/products/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		
PGDBG	Portland Group Fortran 90 debugger	http://www.pgroup.com/
PGPROF	Portland Group Fortran 90 code profiler	http://www.pgroup.com/

3.2 Installing CMAQ Source Code

After installing CVS, the I/O API and netCDF libraries, and Fortran and C compilers on the hardware system, users should then download the CMAQ source code, scripts, and benchmark data files from the CMAS Center web site (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.

3.2.1 Distribution contents

The following files and archives compose the CMAQ distribution:

- AEROSOL_NOTES – description of updates to the CMAQ aerosol routines.
- BIDIRECTIONAL_HG_EXCHANGE – description of the atmospheric-surface exchange mechanism for mercury that is implemented in CMAQ.
- CHEMISTRY_NOTES – details about the chemistry updates in the release version of CMAQ.
- CVS_NETCDF – text file explaining the CVS configuration management system and the netCDF data system and how to set them up.

- EVALUATION_TOOLS – text file explaining how to obtain, compile, and apply CMAQ data analysis utility programs.
- HAZARDOUS_AIR_POLLUTANTS – description of the implementation of HAPs modeling in CMAQ.
- INLINE_EMISSIONS_DEPV_NOTES – description of how to incorporate the calculation of biogenic emissions and elevated-point-source plume rise into CMAQ simulations.
- INLINE_RUN_SCRIPT – example run script for a CCTM simulation that uses in-line emissions processing.
- IOAPI – explanation of how to get the I/O API system and set up the libraries
- MCIP_TO_HYSPLIT – description of the utility program for converting MCIP data into HYSPLIT format
- MULTIPOLLUTANT_MODEL – description of the multipollutant configuration of CMAQ for simulating mercury, air toxics, ozone, PM, and acid deposition.
- PARALLEL_NOTES – comments related to running the CMAQ CCTM in Linux MPICH clusters
- PROCESS_ANALYSIS_NOTES – description of the CMAQ process analysis option
- README – description of the CMAQ installation procedures (alternative to this installation chapter).
- RELEASE_NOTES – list of the changes since the last release
- RESTART_FILE – description of the restart file that was first introduced in CMAQ version 4.6
- SMOKE – description of the SMOKE emissions system
- MODELS.CMAQv4.7.1.tar.gz – gzipped tar file (~7.2 MB) containing model, tools, and libraries source code CVS archives
- DATA.CMAQv4.7.1.tar.gz – gzipped tar file (~52 MB) containing the required datasets necessary to run the benchmark case.
- SCRIPTS.CMAQv4.7.1.tar.gz – gzipped tar file (~16 KB) containing C-shell scripts to build and execute the CMAQ models
- DATA_REF.CMAQv4.7.1.tar.gz – gzipped tar file (~217 MB) containing reference data to compare with datasets produced by the tutorial on a Linux workstation

The CMAQ installation scripts are configured for a Linux system with either the Portland Group or Intel Fortran compiler.

3.2.2 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 the `M3HOME` directory installs the CMAQ source code, scripts, and benchmark data files in a directory structure recognized by the default run and build scripts. The `M3HOME` directory is the base location of the CMAQ installation for a specific application. Under `M3HOME`, the *scripts* directory contains the build and run scripts, the *models* directory contains the CVS archive of model source code, 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:

```
M3LIB     =    M3HOME/lib
M3DATA    =    M3HOME/data
M3MODEL   =    M3HOME/models
```

The CMAQ scripts require users to select only the location of the `M3HOME` directory; the other CMAQ directories are relative to `M3HOME`. 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.

3.2.3 Configuring your system for compiling CMAQ

Before installing any components of CMAQ, users must install CVS, netCDF, and I/O API. Compiler flag consistency between the netCDF and I/O API is critical for building library files compatible with CMAQ. Table 3-3 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 3-3. NetCDF and I/O API compilation options for CMAQ

Library type	Intel Fortran	PGI Fortran

netCDF	`CC = gcc`
	`CPPFLAGS = ` ``-DNDEBUG –DpgiFortran``
	`CFLAGS = ` ``-g –O``
	`FC = ifort`
	`F77 = ifort`
	`FFLAGS = ` ``-g –O2 –mp –recursive``
	`CXX = g++`
		`CC = gcc`
		`CPPFLAGS = ` ``-DNDEBUG –DpgiFortran``
		`CFLAGS = -O`
		`FC = pgf90`
		`FFLAGS = ` ``-O –w``
		`CXX = g++`

I/O API 32-bit	`BIN = Linux2_x86ifc`	`BIN = Linux2_x86pg_pgcc_nomp`
I/O API 64-bit	`BIN = Linux2_x86ifc`	`BIN = Linux2_x86_64pg_pgcc_nomp`

To install CMAQ (with examples using a C-shell environment, a Red Hat Linux system, and the Portland Group Fortran compiler):

1. Set the environment variable `M3HOME` to alias the directory where CMAQ will be installed. For example, if you installed the CMAQ source code in the directory /home/user/CMAQ, set the `M3HOME` environment variable using the following command:

   ```
   setenv M3HOME /home/user/CMAQ
   ```

2. Set environment variables for the `M3DATA`, `M3LIB`, and `M3MODEL` directory paths needed by CMAQ:

   ```
   setenv M3DATA $M3HOME/data
   setenv M3LIB $M3HOME/lib
   setenv M3MODEL $M3HOME/models
   ```

3. Create the `M3LIB` directory

   ```
   mkdir $M3LIB
   ```

4. Download the CMAQ tar files and unpack them in the `M3HOME` directory:

   ```
   tar xvzf SCRIPTS.CMAQv4.7.1.tar.gz
   tar xvzf MODELS.CMAQv4.7.1.tar.gz
   tar xvzf DATA.CMAQv4.7.1.tar.gz
   tar xvzf DATA_REF.CMAQv4.7.1.tar.gz
   tar xvzf DOCS.CMAQv4.7.1.tar.gz
   ```

5. The CMAQ scripts assume that the netCDF and I/O API are already installed and compiled on the hardware system. The precompiled libraries can be copied into `$M3LIB`, or symbolic links to each library (i.e., *.a) file can be created in the appropriate locations in `$M3LIB`. The example below is for Portland Group Fortran compiled libraries on a 64-bit operating system.

   To copy a precompiled version of the netCDF library, use a variation of the following commands:

   ```
   mkdir –p $M3LIB/netCDF/Linux2_x86_64pg
   cd $M3LIB/netCDF/Linux2_x86_64pg
   cp /home/user/netcdf-3.6.1/lib/libnetcdf.a .
   ```

   or to create a symbolic link:

   ```
   mkdir –p $M3LIB/netCDF/Linux2_x86_64pg
   cd $M3LIB/netCDF/Linux2_x86_64pg
   ln –s /home/user/netcdf-3.6.1/lib/libnetcdf.a libnetcdf.a
   ```

   Similarly for the I/O API, either copy a precompiled version of the library to the `$M3LIB/ioapi/Linux2_x86_64pg` directory or create a symbolic link to the library *.a file. You also need to copy or link the contents of the I/O API fixed source INCLUDE file directory to the `$M3LIB/ioapi/fixed_src` directory.

   A list of the alternative library directory names expected by the CMAQ compile and run scripts for 32-bit and 64-bit operating systems using the Intel and Portland Group compilers include:

- Portland Group 32-bit Linux: `Linux2_x86pg`
- Portland Group 64-bit Linux: `Linux2_x86_64pg`
- Intel 32-bit Linux: `Linux2_x86ifort`
- Intel 64-bit Linux: `Linux2_x86_64ifort`

Other operating systems and compiler combinations are possible but may require manual configuration of the CMAQ scripts to point to the correct directory paths for the netCDF and I/O API libraries.

After unpacking all of the CMAQ tar files under the `$M3HOME` directory, setting the CMAQ environment variables, and installing the I/O API and netCDF libraries in the `$M3LIB` directory, the CMAQ executables can then be compiled. Before proceeding with the compilation steps, check to make sure CVS is available on your system by typing the following command:

```
which cvs
```

If the message “cvs: Command not found.” is returned, you must install CVS on your system before you can continue with the CMAQ installation.

3.3 Compiling CMAQ for the Benchmark Test Case Simulation

For all CMAQ programs (other than MCIP), the program M3BLD is used to compile the source code into executables. The first step in the compilation of CMAQ is to compile M3BLD. Then compile the program libraries STENEX and PARIO before moving on to compiling the rest of the CMAQ programs. *For all of the CMAQ programs and libraries, the directory paths for the Fortran and C compilers in the build scripts have to be changed to reflect the correct locations on your system*. The scripts for compiling all of the CMAQ programs (with the exception of M3BLD) contain compiler flags for both the Portland Group and Intel Fortran compilers. In addition to pointing these scripts to the correct location of the compiler on your system, you must also uncomment (remove ‘#’ at the beginning of the line) the compiler flag setting for the Fortran compiler that you will use to create CMAQ executables. CMAQ may be run in singale processor (serial) or in parallel on multiple processors. For parallel compilations, some of the programs also require directory paths to MPICH INCLUDE files. Program-specific compilation instructions are provided below. These compilation instructions are for building executables for simulating the benchmark data sets distributed with CMAQ. Other than the paths to the compilers and libraries and the compilation flags, the build scripts used in this section will not need to be changed to create executables. Additional information about the configuration options for the various CMAQ programs is provided in Chapters 2 and 5.

1. Use the following commands to compile M3BLD:

```
cd $M3HOME/scripts/build
bldit.m3bld
```

2. Next create the stencil exchange (STENEX) libraries for serial and parallel processing. Verify on line 37 of the file bldit.se.Linux that the MPICH INCLUDE file directory path is correct for the parallel version of the library. Use the following commands to compile the STENEX libraries:

   ```
   cd $M3HOME/scripts/stenex
   bldit.se_noop.Linux
   bldit.se.Linux
   ```

3. For parallel CCTM operation, create the parallel input/output (PARIO) library:

   ```
   cd $M3HOME/scripts/pario
   bldit.pario.Linux
   ```

Now create the model executables for JPROC, ICON, BCON, MCIP, and CCTM.

4. For the benchmark case, confirm that JPROC is configured to produce photolysis rates for the mechanism labeled "cb05cl_ae5_aq". Use the following command to compile JPROC:

   ```
   cd $M3HOME/scripts/jproc
   bldit.jproc
   ```

5. 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 Chapters 2 and 6. Use the following commands to compile ICON and BCON:

   ```
   cd $M3HOME/scripts/icon
   bldit.icon
   cd $M3HOME/scripts/bcon
   bldit.bcon
   ```

6. MCIP is compiled using a Fortran Makefile instead of M3BLD. To create the MCIP executable, set the compiler, compiler flags, and netCDF and I/O API library paths in the Makefile distributed with MCIP. The I/O API include files needed to compile MCIP are different from those needed by the other CMAQ components. The other CMAQ programs use the include files in the $M3LIB/ioapi/fixed_src directory. For MCIP, you must set the I/O API include file path in the Makefile to the directory on your system that contains the I/O API source code. The I/O API fixed source directory containing the include files needed by the rest of the CMAQ programs is contained within the base I/O API source code directory. This base source code directory contains the version of the I/O API include files needed to compile MCIP.

   Once you have configured the MCIP Makefile for your system, use the following commands to compile MCIP:

   ```
   cd $M3HOME/scripts/mcip/src
   ```

```
make
```

7. 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 Chapters 2 and 6. The CCTM build script is configured by default to create a single-processor executable for the benchmark simulation. For multiprocessor applications, CMAQ uses the MPICH message passing interface (MPI) to manage communication between processors in a clustered multiprocessor computing environment. Before compiling CCTM for parallel execution, you must specify the location of the MPICH directory on your system in the CCTM build script. For single-processor systems, configure the 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 $M3HOME/scripts/cctm
bldit.cctm
```

Although not used for the benchmark simulation, the PROCAN processor can also be compiled using M3BLD.

3.4 Running the CMAQ Benchmark 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. Other than that there are no dependencies among the other CMAQ programs, so they can be run in any order to create input data for CCTM. If all of the programs were compiled with Intel Fortran on a Linux 64-bit operating system, the benchmark simulation will execute without any modifications to the run scripts. There will be slight modifications to the run scripts if the CMAQ programs were either compiled on 32-bit operating systems and/or they were compiled with other brands of Fortran. These modifications will entail changing the name of the program executables to reflect the configuration of your computing system. For example, by default the ICON run script will look for the 64-bit Intel Fortran compiled executable named `ICON_e1a_Linux2_x86_64ifort`. If you compiled the program on a 32-bit operating system using the Portland Group compiler, change the EXEC variable in the `run.icon` file to `ICON_${CFG}_Linux2_x86pg`.

CCTM is configured by default to run in single-processor mode. For single-processor computing, set NPROCS to 1 and NPCOL_NPROW to "1 1". Running in multiprocessor mode requires building an executable that includes routines from the MPICH library and setting the number of processors to allocate to the simulation and the location of the MPI initialization command (mpirun) on your system. Configuring the CCTM run script for parallel processing requires selecting the number of processors to use for the simulation by setting the NPROCS environment variable, and choosing the domain decomposition configuration by setting the variable NPCOL_NPROW. The number of processors 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”. Most clustered multiprocessor systems require a command to start the MPI run-time environment. The CCTM run script is configured by default to use the *mpirun* command. Consult your system administrator to find out how to invoke MPI when running multiprocessor applications.

3.5 Benchmarking CMAQ

After completing the CMAQ benchmark case, the CCTM output files can be compared with the reference datasets provided in the CMAQ distribution using PAVE or VERDI. A plot of percent difference between the output and the benchmark data is the easiest method. If the benchmark case is run on a system similar to that used to create the reference data (Red Hat Linux with the Portland Group or Intel Fortran compiler), the results should differ by no more than 0.01% for every model species. Changing the optimization of the compiler or compiling on other operating systems with different compilers can lead to larger differences between the benchmark results and the reference datasets. The CCTM benchmark targets for operating systems/compilers other than the Red Hat Linux/Portland Group combination are differences less than 1% for every model species. Differences greater than this require a revew of the installation. Remember it is necessary that the same compilers be used for all programs.

4. 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 (MPICH) is only required for multiple-processor applications of the CCTM. Brief descriptions of these three libraries are provided in this chapter. For additional information, including how to compile and configure these libraries, refer to the documentation associated with each library.

4.1 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 http://www.baronams.com/products/ioapi/.. 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.

The CMAQ input and output files use a hybrid Network Common Data Form (netCDF)-I/O API file format. The netCDF is described in detail in Section 4.2. 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. In addition to providing A further 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 performing 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 Chapter 9 and the I/O API documentation.

4.1.1 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 *YYYYDAY* (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. This will be discussed further in terms of the CMAQ programs in Chapter 5. The remainder of this section explains some of the rudimentary details of programming in an environment using I/O API data files.

4.1.2 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 4-1). 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 4-2. GRDDED3 and BNDARY3 are the most prevalent file types in a CMAQ simulation. Magic number is an indicator associated with the files type.

Table 4-1. 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 4-2. Possible Data Type Structures in I/O API Files

File Type	Magic Number	Data Type	Description
CUSTOM3	-1	Custom	User-dimensioned array of REAL*4s that the system reads/writes reliably
DCTNRY3	0	Dictionary	Data type stores and retrieves parts of an FDESC.EXT file description
GRDDED3	1	Gridded	Dimension as REAL*4 ARRAY (NCOLS, NROWS, NLAYS, NVARS)
BNDARY3	2	Boundary	Dimension as REAL*4 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

File Type	Magic Number	Data Type	Description
			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

4.1.3 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:	CHARACTER*(*) FNAME:	file name for query
	INTEGER FSTATUS:	see possible values in Table 4.3
	CHARACTER*(*) PGNAME:	name of calling program

This function 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 4-3.

Table 4-3. Possible values for OPEN(3) FSTATUS

FSTATUS	Value	Description
FSREAD3	1	for READ-ONLY 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
```

4.1.4 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 4-4.

Table 4-4. 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.

4.1.5 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: CHARACTER*(*) FNAME	file name for query
CHARACTER*(*) VNAME	variable name (or ALLVAR3 (='ALL'))
INTEGER JDATE	date, formatted YYYYDDD
INTEGER JTIME	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
```

4.1.6 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 4-5. Further information about how to use the utilities are available in the I/O API documentation.

Table 4-5. 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

Utility	Description
VERTOT	compute vertical-column totals of variables in a file
UTMTOOL	coordinate conversions and grid-related computations for lat/lon, Lambert, and UTM

4.2 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 http://www.unidata.ucar.edu/software/netcdf. Compilation and configuration information for the netCDF is available through the Unidata website.

4.3 Message Passing Interface Library (MPICH)

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. MPICH is a freely available, portable implementation of MPI that is available from Argonne National Laboratoryn (ANL). MPICH is used in CMAQ to control how the CCTM is run on parallel computing systems. MPICH can be configured to run the CCTM in parallel on either shared memory systems or on a workstation cluster. Users should download the latest code for MPICH from http://www.mcs.anl.gov/research/projects/mpi/mpich1. Compilation and configuration information for MPICH is available through the ANL website.

4.4 References

Coats, C., 2005: The EDSS/Models-3 I/O API. [Available online at http://www.baronams.com/products/ioapi.]

Unidata, 2009: NetCDF. [Available online at http://www.unidata.ucar.edu/software/netcdf]

5. CMAQ PROGRAMS AND LIBRARIES

5.1 Overview

The core CMAQ programs that are needed to perform a basic air quality model simulation are MCIP, ICON, BCON, JPROC, and CCTM. The relationships among these programs are depicted within the green box in Figure 5-1. 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, BCON, and JPROC. 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 Figure 5-1 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.

The **meteorological model,** such as MM5 or WRF-ARW, generates gridded meteorology for input to both CMAQ and the emissions model.

The **emissions model** is required to convert annual, county-level emissions estimates to gridded, hourly emissions formatted for CMAQ. The SMOKE and CONCEPT emissions models are currently available for preparing emissions data for CMAQ.

With the release of CMAQv4.7, there are two new **in-line** options for emissions: the user can incorporate the processing of biogenic emissions, or of point-source plume rise, or both, directly in a CCTM simulation. Previous versions of CMAQ required that biogenic emissions and point-source plume rise were provided from input files as pre-calculated hourly values. There are several advantages of incorporating these processes directly in a CCTM simulation: (1) the emissions are meteorologically modulated at the synchronization (chemistry) time step rather than being linearly time-interpolated within each simulation hour; (2) disk space may be saved, because a 3-D emissions file is no longer needed for elevated point sources; and (3) CMAQ can more easily be coupled with a meteorological model, enabling direct emissions modulation by the underlying, freshly computed meteorological variables. In-line emissions are an option in CMAQv4.7.1, the traditional approaches of computing biogenic and 3-d point source emissions offline are still available. See the CMAQv4.7 release notes for details on in-line emissions processing.

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.

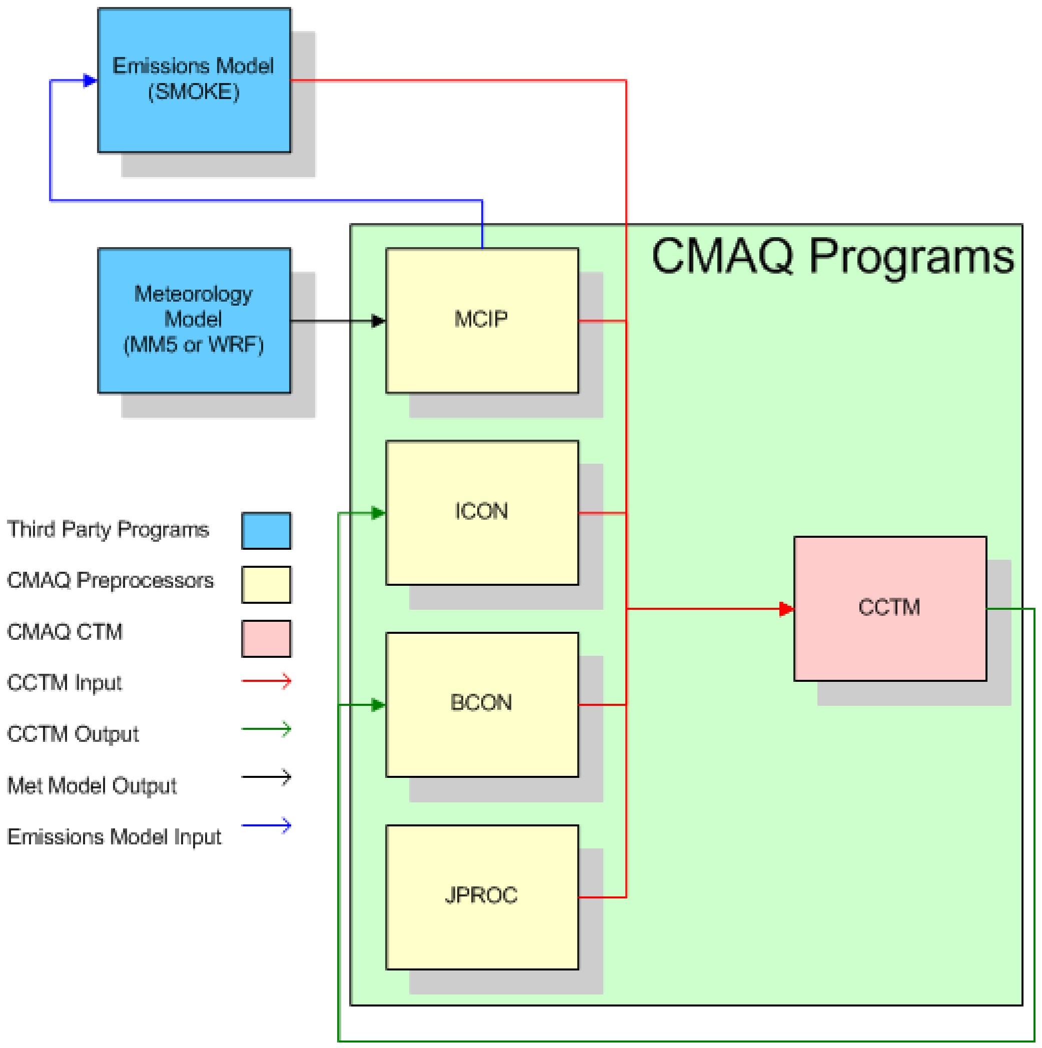


Figure 5-1.CMAQ core programs

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.

JPROC converts physical information about photoreactive molecules into clear-sky photolysis rate look-up tables for input to CCTM. CMAQv4.7 includes a feature for calculating photolysis rates **in-line** in the CCTM. The in-line photolysis approach allows photolysis rates to be adjusted by simulated gas and aerosol concentrations rather than by climatological values in the offline approach.

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 at a time granularity set by the user.

In addition to the core programs shown in Figure 5-1, the CMAQ distribution package also includes utilities and libraries for utilizing some of the special features in CMAQ and for setting up CCTM for multiprocessor computing. CMAQ includes the **PROCAN** utility for preparing process analysis simulations, and **CHEMMECH** for editing existing and preparing new chemical mechanisms for CMAQ. The stencil exchange code library (**STENEX**) is a module that CCTM uses to control the communication between processors in a multiprocessor computing environment. Similarly, CCTM uses the parallel I/O (**PARIO**) code library to synchronize the reading and writing of information among multiple processors.

In the remaining sections of this chapter, we provide detailed descriptions of these programs, utilities, and libraries, in alphabetical order. Information about the third-party libraries used by CMAQ—such as I/O API, netCDF, and MPICH—is available in Chapter 4.

When viewing the tables that list each program's input and output files, recall that the various file formats shown are described in Table 4-2.

5.2 BCON

5.2.1 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 MM5 or 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 *ModInpt* variable. This variable determines the input module to use when creating a BCON executable.

5.2.2 Files, configuration, and environment variables

Figure 5-2 shows the input and output files and configuration options for BCON. A distinction is made between the options that are invoked at compilation time 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 *Mechanism* variable in the BCON compile script configures the program to use a specific set of mechanism INCLUDE files to build an executable. Separate BCON executables must be prepared for different mechanism configurations.

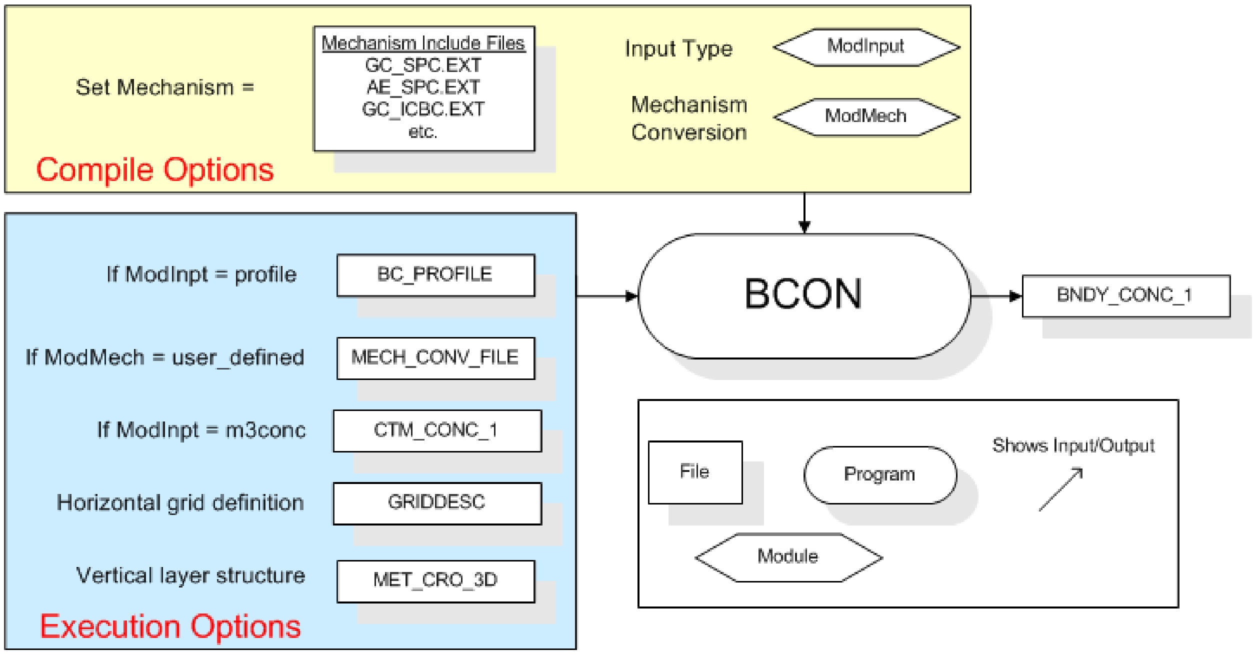


Figure 5-2. BCON input and output files

At execution the user provides a data file of chemical conditions that BCON converts to BCs on a predefined model grid. Through the specification of the *ModInpt* variable, 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. If the input file is not in the same chemical speciation as the simulation for which the user is creating BCs, it is necessary to specify a chemical conversion option by setting the *ModMech* variable at compilation. The default conversion profiles in BCON are set up to convert from RADM2 speciation to either CB05 or SAPRC-99 chemistry. It is possible to create a custom conversion file (MECH_CONV_FILE) and input this file to BCON by setting the *ModMech* variable to "user_defined" at compilation.

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.

5.2.2.1 BCON input files

Table 5-1. 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"
MECH_CONV_FILE	ASCII	Mapping factors for converting between chemical mechanisms; this file is created by the user; used only when **BCON** is compiled with the ModMech configuration set to "user_defined"
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
MET_CRO_3D	GRDDED3	3-D cross-point meteorology file for defining the vertical layer structure of the model grid; this file is output by MCIP

5.2.2.2 BCON compilation options

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.

- `Opt:[default: verbose]`

  Defines the action to be taken by the program **M3BLD** when extracting source code from CVS and compiling an executable.

  - *compile_all:* force compile, even if all the object files are current
  - *clean_up:* remove all source files upon successful compilation
  - *no_compile:* do everything except compile
  - *no_link:* do everything except link

- *one_step:* compile and link in one step
- *parse_only:* check configuration file syntax
- *show_only:* show requested commands but do not execute them
- *verbose:* show requested commands as they are executed

- `MakeOpt`

  Uncomment to build a Makefile to compile the executable.

- `ModInpt:[default: 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

- `ModMech:[default: module radm2_to_cb05]`

  Defines whether the input boundary conditions data need to be converted from one chemical mechanism to another.

  - *mc_noop*: do not perform any mechanism conversion; used when extracting boundary conditions from a CCTM CONC file for a nested simulation or when the input profiles are already formatted for the correct mechanism
  - *user_defined*: input the file defined by the MECH_CONV_FILE variable; used for custom mechanism conversions
  - *radm2_to_cb05*: convert the input boundary conditions profiles from RADM2 to CB05 speciation
  - *radm2_to_saprc99*: convert the input boundary conditions profiles from RADM2 to SAPRC-99 speciation

- `Mechanism: [default: cb05cl_ae5_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 $M3MODEL/include/release directory. Examples include:

  - *cb05ae5_aq*: CB05 gas-phase mechanism, fifth-generation CMAQ aerosol mechanism with sea salt, aqueous/cloud chemistry
  - *cb05cl_ae5_aq*: CB05 gas-phase mechanism, fifth-generation CMAQ aerosol mechanism with sea salt, aqueous/cloud chemistry, and active chlorine

- *cb05cltx_ae5_aq*: CB05 gas-phase mechanism, fifth-generation CMAQ aerosol mechanism with sea salt, aqueous/cloud chemistry, active chlorine, and air toxics
- *cb05cltxhg_ae5_aq*: CB05 gas-phase mechanism, fifth-generation CMAQ aerosol mechanism with sea salt, aqueous/cloud chemistry, active Chlorine, air toxics, and mercury; this is for the multipollutant model (see CMAQv4.7 release notes)
- *saprc99_ae5_aq*: SAPRC-99 gas-phase mechanism, fifth-generation CMAQ aerosol mechanism with sea salt, aqueous/cloud chemistry
- *saprc99tx3_ae5*: SAPRC-99 gas-phase mechanism with air toxics and fifth-generation CMAQ aerosol mechanism

5.2.2.3 BCON compilation

First, it is assumed that you have already installed and compiled the I/O API and netCDF libraries (see Section 3.2.3).

Section 3.3 provides an overview of how to install and compile the CMAQ programs for the benchmark simulation. Follow the steps outlined in Section 3.3 (summarized below) to compile new versions of **BCON**:

- If you have not already done so, compile **M3BLD**, the CMAQ source code and compilation management program. This needs to be done only once—the first time CMAQ is installed.
- If needed, configure the **BCON** build script to use the available I/O API and netCDF libraries.
- Configure the **BCON** build script for your application (using the options discussed in Section 5.2.2.2)
- Invoke the build script to create an executable:

```
./bldit.bcon
```

5.2.2.4 BCON execution options

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

- `EXEC: [default: BCON_${CFG}_ Linux2_x86_64pg]`

  Executable to use for the simulation.

- `NPCOL_NPROW: [default: 1 1]`

Domain decomposition for parallel mode. BCON is normally run in a single-processor environment, so this setting should always be “1 1”.

- `GRIDDESC: [default: ../GRIDDESC1]`

  Grid description file for setting the horizontal grid definition.

- `GRID_NAME: [default:M_36_2001]`

  Name of the grid definition contained in the GRIDDESC file that specifies the horizontal grid for the current application of the model.

- `LAYER_FILE:[default:$M3DATA/mcip3/M_36_2001/METCRO3D_010722]`

  Name and location of a MET_CRO_3D file for specifying the vertical layer structure for the current application of the model.

- `OUTDIR: [default: $M3DATA/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**. Also optionally looks for the variable MECH_CONV_FILE to point to a user-defined mechanism conversion file.
  - *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.

5.2.2.5 BCON output files

Table 5-2. 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 `$M3DATA/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.

5.3 CCTM

5.3.1 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 configura-tion 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.

Starting in version 4.7 the 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 the CCTM, such as emissions processing with SMOKE, with algorithms internal to the 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 the CCTM. When the 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 the CCTM includes all of the photolysis rate data internal to the in-line instrumented version of the model.

5.3.2 Files, configuration, and environment variables

Figure 5-3 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 the 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 the CCTM must also be selected during the compilation step for the CCTM. Separate CCTM executables must be prepared for different mechanism and science configurations. During the execution step, or when the 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.

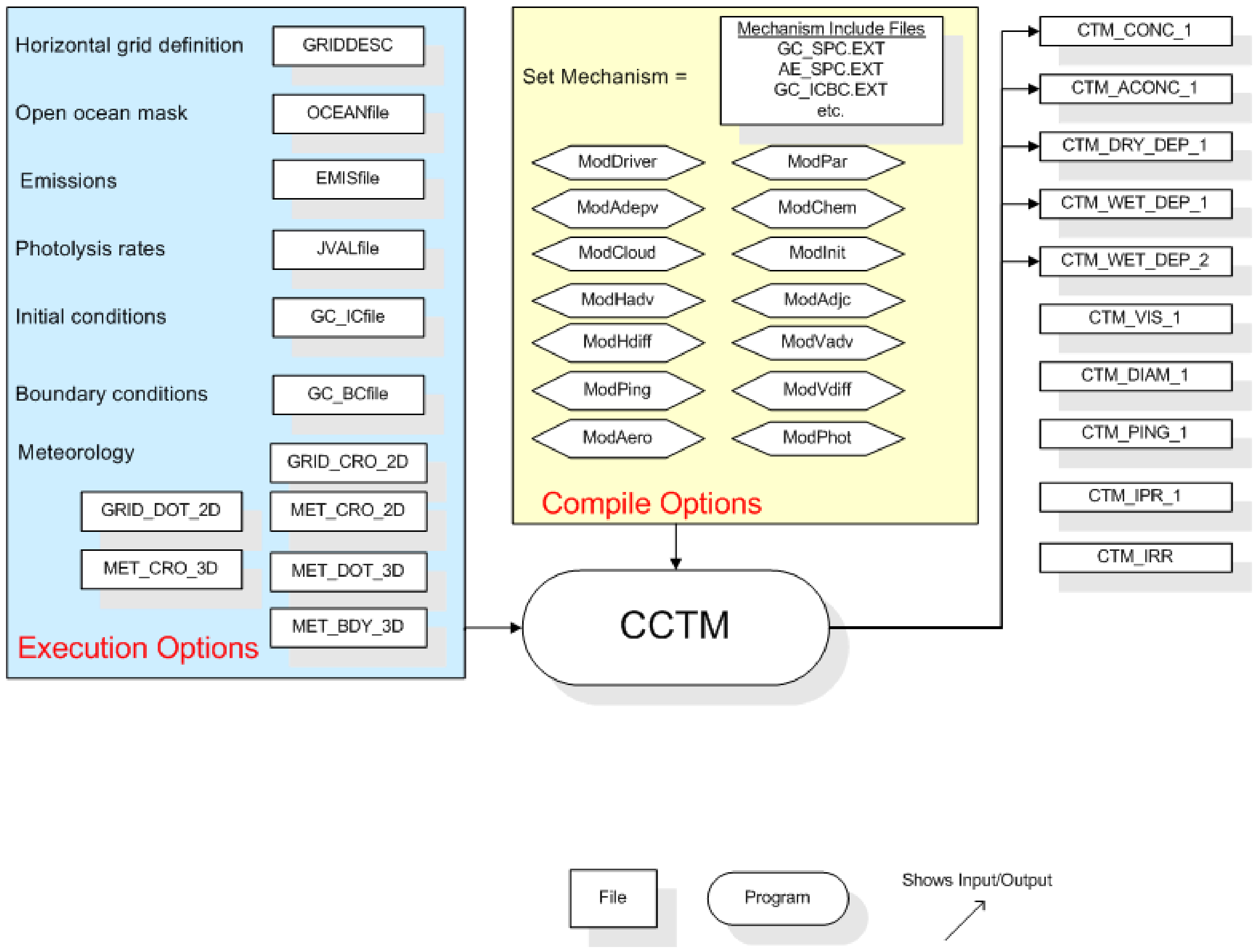


Figure 5-3. CCTM input and output files

5.3.2.1 CCTM input files

Table 5-3. 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
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]_1	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]_1	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
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

5.3.2.2 CCTM compilation options

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.

- `Opt: [default: verbose]`

  Defines the action to be taken by the program **M3BLD** when extracting source code from CVS and compiling an executable.

  - *compile_all:* force compile, even if all the object files are current
  - *clean_up:* remove all source files upon successful compilation
  - *no_compile:* do everything except compile
  - *no_link:* do everything except link
  - *one_step:* compile and link in one step
  - *parse_only:* check configuration file syntax
  - *show_only:* show requested commands but does not execute them
  - *verbose:* show requested commands as they are executed

- `MakeOpt`

Uncomment to build a Makefile to compile the executable.

- `ParOpt`

  Uncomment to build an executable for running on multiple processors. Invoking this command requires the availability of a parallel STENEX library file, a PARIO library file, and the MPI library/INCLUDE files.

- `ModDriver: [default: ctm_yamo]`

  The CCTM generalized -coordinate driver module.

  - *ctm*: use air-density-based scheme for mass-conserving advection
  - *ctm_yamo*: use Yamartino scheme for mass-conserving advection

- `ModInit: [default: init_yamo]`

  The CCTM time-step initialization module.

  - *init*: use air-density-based scheme for mass-conserving advection
  - *init_yamo*: use Yamartino scheme for mass-conserving advection

- `ModAdjc: [default: // Yamartino option]`

  Mass conservation error adjustment scheme. Corrects for mass inconsistencies arising from how the input meteorology treats density and wind fields. This adjustment is required only if the air-density-based scheme for mass-conserving advection is selected.

  - *adjcon_noop*: deactivate the air-density-based conservation adjustment scheme
  - *denrate*: adjust the vertical advection error term from the meteorological model by air density from CCTM
  - //: comment out when using Yamartino scheme for mass-conserving advection

- `ModCpl: [default: gencoor]`

  Unit conversion and concentration coupling module. The only option is for the CMAQ generalized coordinate (*gencoor*).

- `ModHadv: [default: hyamo]`

  Horizontal advection module.

  - *hadv_noop*: deactivate horizontal advection

  - *hppm*: use the unmodified Piecewise Parabolic Method to calculate horizontal advection
  - *hyamo*: use the global mass-conserving scheme to calculate horizontal advection

- `ModVadv: [default: vyamo]`

  Vertical advection module.

  - *vadv_noop*: deactivate vertical advection
  - *vppm*: use the unmodified Piecewise Parabolic Method to calculate vertical advection
  - *vyamo*: use the global mass-conserving scheme to calculate vertical advection

- `ModHdiff: [default: multiscale]`

  Horizontal diffusion module.

  - *hdiff_noop*: deactivate horizontal diffusion
  - *multiscale*: use diffusion coefficient based on local wind deformation

- `ModVdiff: [default: acm2_inline]`

  Vertical diffusion module.

  - *vdiff_noop*: deactivate vertical diffusion
  - *eddy*: calculate vertical diffusion using eddy diffusivity theory
  - *eddy_txhg*: use eddy vertical diffusion instrumented for the mercury and air toxics CMAQ model
  - *acm2*: calculate vertical diffusion using the Asymmetric Convective Model version 2 (ACM2)
  - *acm2_inline*: use ACM2 vertical diffusion instrumented for in-line calculation of emissions
  - *acm2_inline_txhg*: use ACM2 vertical diffusion with in-line calculation of emissions and instrumented for the mercury and air toxics CMAQ model; this is for the multipollutant model (see CMAQv4.7 release notes)
  - *acm2_inline_ txhgsim*: use ACM2 vertical diffusion with in-line calculation of emissions and instrumented for the mercury and air toxics CMAQ model, with bidirectional exchange of mercury; this is for the multipollutant model (see CMAQv4.7 release notes)

- `ModPhot: [default: phot]`

  Photolysis calculation module.

  - *phot_noop*: deactivate photolysis rate calculations
  - *phot*: calculate photolysis rates
  - *phot_sat*:
  - *phot_inline*:

- `ModChem: [default: ebi_cb05cl_ae5]`

  Gas-phase chemistry solver module.

  - *chem_noop*: deactivate gas-phase chemistry
  - *smvgear*: use the SMVGEAR chemistry solver
  - *ros3*: use the Rosenbrock chemistry solver
  - *ebi_cb05cl_ae5*: use the Euler Backward Iterative solver optimized for the Carbon Bond-05 mechanism with chlorine and extended aerosols
  - *ebi_cb05cltxhg_ae5*: use the Euler Backward Iterative solver optimized for the Carbon Bond-05 mechanism with air toxics, mercury, and chlorine with extended aerosols; this is for the multipollutant model (see CMAQv4.7 release notes)
  - *ebi_saprc99_ae5*: use the Euler Backward Iterative solver optimized for the SAPRC-99 mechanism with extended aerosols
  - *ebi_saprc99tx3*: use the Euler Backward Iterative solver optimized for the SAPRC-99 mechanism with air toxics

- `ModAero: [default: aero5]`

  CMAQ aerosol module.

  - *aero_noop*: deactivate aerosol chemistry
  - *aero4*: use the fourth-generation modal CMAQ aerosol model with extensions for sea salt emissions and thermodynamics
  - *aero5*: use the fifth-generation modal CMAQ aerosol model with extensions for sea salt emissions and thermodynamics and a new formulation for secondary organic aerosol

  - *aero5_txhg*: use aero5 with air toxics and mercury chemistry; this is for the multipollutant model (see CMAQv4.7 release notes)

- `ModAdepv: [default: aero_depv2]`

  CMAQ aerosol deposition velocity module.

  - *aero_depv_noop*: deactivate aerosol deposition
  - *aero_depv2*: use second-generation CMAQ aerosol deposition velocity routine

- `ModCloud: [default: cloud_acm_ae5]`

  CMAQ cloud module for modeling the impacts of clouds on deposition, mixing, photolysis, and aqueous chemistry.

  - *cloud_ noop*: deactivate clouds in modeling
  - *cloud_acm_ae5*: use ACM cloud processor that uses the ACM methodology to compute convective mixing
  - *cloud_acm_ae5_tx*: use ACM cloud processor that uses the ACM methodology to compute convective mixing including air toxics
  - *cloud_acm_ae5_txhg*: use ACM cloud processor that uses the ACM methodology to compute convective mixing including air toxics and mercury.
  - *cloud_acm_ae5_tshgsim*: use ACM cloud processor that uses the ACM to compute convective mixing including air toxics, mercury, and chlorine emissions from the sea and surf zone; this version is for the multipolutant model with bidirectional exchange of mercury (see CMAQv4.7 release notes)

- `ModPa: [default: pa]`

  Process analysis module.

  - *pa*: only configuration option at the module level; to turn process analysis on/off in CMAQ, use the PAOpt variable (see below).

- `ModUtil: [default: util]`

  CMAQ utility modules.

  - *util*: only configuration option at the module level

- `Mechanism: [default: cb05_ae5_aq]`

Specifies the gas-phase, aerosol, and aqueous-phase chemical mechanisms to use for modeling air quality. The choices for the Mechanism variable are the mechanism directory names under the `$M3MODEL/include/release` directory. Examples include:

- *cb05ae5_aq*: CB05 gas-phase mechanism, fifth-generation CMAQ aerosol mechanism with sea salt, aqueous/cloud chemistry
- *cb05cl_ae5_aq*: CB05 gas-phase mechanism, fifth-generation CMAQ aerosol mechanism with sea salt, aqueous/cloud chemistry, and active chlorine
- *cb05cltx_ae5_aq*: CB05 gas-phase mechanism, fifth-generation CMAQ aerosol mechanism with sea salt, aqueous/cloud chemistry, active chlorine, and air toxics
- *cb05cltxhg_ae5_aq*: CB05 gas-phase mechanism, fifth-generation CMAQ aerosol mechanism with sea salt, aqueous/cloud chemistry, active chlorine, air toxics, and mercury; this is for the multipollutant model (see CMAQv4.7 release notes)
- *saprc99_ae5_aq*: SAPRC-99 gas-phase mechanism, fifth-generation CMAQ aerosol mechanism with sea salt, aqueous/cloud chemistry
- *saprc99tx3_ae5*: SAPRC-99 gas-phase mechanism with air toxics and fifth-generation CMAQ aerosol mechanism

- `Tracer [default trac0]`

  Specifies tracer species to use from the emissions, IC, or BC input files. Invoking inert tracer species in CMAQ requires defining the tracers using INCLUDE files and compiling CCTM with these files. The setting for this module corresponds to the directory name in the `$M3MODEL/include/release` directory that contains the INCLUDE files for the tracer configuration to implement in CCTM. The default setting is to not use tracers in CCTM.

- `PABase = $GlobInc`

  Specifies the base directory location of the process analysis include files to use when compiling the CCTM.

- `PAOpt: [default: pa_noop]`

  Specifies the process analysis configuration to use for CMAQ. The choices for the PAOpt variable are the available directories for process analysis INCLUDE files under the `$M3MODEL/include/release` directory.

5.3.2.3 CCTM compilation

First, it is assumed that you have already installed and compiled the I/O API, netCDF, and MPICH libraries (see Section 3.2.3), or that these are already available from a previous CMAQ compilation.

Section 3.3 provides an overview of how to install and compile the CMAQ programs for the benchmark simulation. Follow the steps outlined in Section 3.3 (summarized below) to compile new versions of **CCTM**:

- If you have not already done so, compile **M3BLD**, the CMAQ source code and compilation management program. This needs to be done only once—the first time CMAQ is installed.
- If needed, configure the **CCTM** build script to use the available I/O API, netCDF, and MPICH libraries.
- If you have not already done so, compile the STENEX and PARIO libraries.
- Configure the **CCTM** build script for your application (using the options discussed in Section 5.3.2.2)
- Invoke the build script to create an executable:

  ```
  ./bldit.cctm
  ```

5.3.2.4 CCTM execution options

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

- `EXEC [default: CCTM_${APPL}_$CFG ]`

  Executable to use for the simulation.

- `NPCOL_NPROW [default: 1 1]`

  Domain decomposition for parallel mode; recommended configuration is for the number of columns to be larger than the number of rows. For example, if running with 8 processors, the recommended setting is “4 2”.

- `NPROCS [default: 1]`

  Number of processors for parallel execution; equal to the product of NPCOL x NPROW.

- `STDATE:`

Simulation start date in Julian format (YYYYDDD).

- `STIME`

  Simulation start time (HHMMSS).

- `NSTEPS [default: 240000]`

  Number of simulation time steps (HHMMSS).

- `TSTEP [default: 010000]`

  Simulation output time step interval (HHMMSS).

- `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.

- `IOAPI_LOG_WRITE [default: F]`

  [T|F]; set to “T” to turn on excess WRITE3 logging by the I/O API.

- `FL_ERR_STOP [default: F]`

  [T|F|Y|N]; set to “T” or “Y” to configure the program to exit if inconsistent headers are found in the input files.

- `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

- `OUTDIR [default: $M3DATA/cctm]`

  CCTM output file directory location.

- `CTM_APPL [default: $APPL]`

  CCTM log file naming extension.

- `GRIDDESC [default: ../GRIDDESC1]`

Grid description file for setting the horizontal grid definition.

- `GRID_NAME [default: M_36_2001]`

  Name of the grid definition contained in the GRIDDESC file that specifies the horizontal grid for the current application of the model.

- `AVG_CONC_SPCS [default: “O3 NO CO NO2 ASO4I ASO4J NH3”]`

  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: “1 1”]`

  Vertical model layer range for integral average concentrations; this variable sets the lower and upper layers over which to calculate integral average concentrations.

- `CONC_SPCS [default: if not defined, all CONC-file species]`

  Model species to be written to the CCTM CONC file.

- `CONC_BLEV_ELEV [default: if not defined, 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.

- `CTM_MAXSYNC [default: 720]`

  Maximum synchronization time step in seconds.

- `CTM_MINSYNC [default: 60]`

  Minimum synchronization time step in seconds.

- `CTM_AERDIAG [default: Y]`

  Generate an instantaneous hourly aerosol diagnostic file with the geometric mean diameters and the geometric standard deviations for the lognormal aerosol modes. Set to Y or T to turn on, N or F to turn off.

- `CTM_SSEMDIAG [default: Y]`

Generate the calculated sea salt emissions in a diagnostic netCDF output file. Set to Y or T to turn on, N or F to turn off.

- `CTM_ILDEPV [default: Y]`

  Calculate in-line deposition velocities. Set to Y or T to turn on, N or F to turn off.

- `CTM_SFC_HONO [default: Y]`

  Calculate surface HONO interactions. This option is ignored if CTM_ILDEPV is turned off. Set to Y or T to turn on, N or F to turn off.

- `CTM_DEPV_FILE [default: N]`

  Generate an hourly diagnostic file for the in-line deposition velocity calculations. This option is ignored if CTM_ILDEPV is turned off. Set to Y or T to turn on, N or F to turn off.

- `CTM_BIOGEMIS [default: N]`

  Calculate biogenic emissions. Set to Y or T to turn on, N or F to turn off.

  If the option to calculate in-line biogenic emissions is activated (i.e., CTM_BIOGEMIS is set to Y or T), the following variables must be set.

  - `GSPRO`

    Directory path and file name for input speciation profiles.

  - `B3GRD`

    Directory path and file name for grid-normalized biogenic emissions input file.

  - `BIOSW_YN [default: Y]`

    Use the frost dates switch file to determine whether to use winter or summer biogenic emissions. Set to Y or T to turn on, N or F to turn off.

  - `BIOSEASON`

    Directory path and file name for the frost date switch file.

  - `SUMMER_YN [default: Y]`

Use summer season normalized biogenic emissions. This variable is ignored if BIOSW_YN is set to Y. Set to Y or T to turn on, N or F to turn off.

- `BIOG_SPRO`

  Profile ID for speciating biogenic VOCs. This profile ID must be present in the GSPRO file.

- `B3GTS_DIAG [default: N]`

  Generate the calculated biogenic emissions (mass units) in a diagnostic netCDF output file. Set to Y or T to turn on, N or F to turn off.

- `B3GTS_S`

  Directory path and file name for the diagnostic output biogenic emissions. This variable is ignored if B3GTS_DIAG is turned off.

- `INITIAL_RUN`

  Set to Y or T if this is the first time that biogenic NO soil emissions will be calculated. If there is a previously created file, set to N or F.

- `SOILINP`

  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.

- `CTM_PT3DEMIS [default: N]`

  Calculate plume rise for elevated point sources. Set to Y or T to turn on, N or F to turn off.

  The CMAQv4.7 release notes file INLINE_EMISSIONS_DEPV_NOTES.txt describes additional settings that are required for calculating elevated-point-source emissions with CCTM. If the option to calculate in-line plume rise is activated (i.e., CTM_PT3DEMIS is set to Y or T), the following variables must be set.

  - `NPTGRPS [default: 1]`

    The number of input point-source 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.

- `MERGE_DATES`

  Directory path and file name of a file that lists the Gregorian dates for creating merged emissions files.

- `LAPY_STTIME [HHMMSS]`

  Start time for calculating elevated-point-source emissions.

- `LAPY_NSTEPS [HHHHHH]`

  Number of time steps for calculating elevated-point-source emissions.

- `CTM_EMLAYS [##]`

  Number of emissions layers for calculating elevated-point-source emissions.

- `ASKfile`

  Directory path and location for the output surface media mercury concentrations. This variable is used only when CCTM is compiled to calculate the bidirectional surface exchange for mercury.

- `OCEANpath [default: $M3DATA/emis/2001]`
- `OCEANfile [default: us36_surf40x44.ncf]`

  Directory path and file name for the input sea salt emissions file.

- `EMISpath [default: $M3DATA/emis/2001]`
- `EMISfile [default: emis3d.10010722,US36_40x44.ncf]`

  Directory path and file name for the input emissions file.

- `TR_EMpath [default: commented out]`
- `TR_Emfile [default: commented out]`

  Directory path and file name for the input tracer emissions file.

- `[GC|AE|NR|TR]_ICpath [default: $M3DATA/icon]`
- `GC_ICfile [default:ICON_e1a_${cfg}_cb05cl_M_36_2001_profile]`

  Directory path and file name for the input initial conditions file.

- `$MD_ICpath`
- `$MD_ICfile`

  Directory path and file name for the input initial conditions for the bidirectional surface exchange model. These variables are used only when CCTM is compiled to calculate the bidirectional surface exchange for mercury.

- `[GC|AE|NR|TR]_BCpath [default: $M3DATA/bcon]`
- `GC_BCfile [default:BCON_e1a_${cfg}_cb05cl_M_36_2001_profile]`

  Directory path and file name for the input boundary conditions file.

- `METpath [default: $M3DATA/mcip3/M_36_2001]`

  Directory path for the input meteorology files.

- `JVALpath [default: $M3DATA/jproc/${cfg}]`
- `JVALfile [default: JTABLE_${STDATE}`

  Directory path and file name for input clear-sky photolysis rate files.

5.3.2.5 CCTM output files

Table 5-4. CCTM output files

File Name	Format	Description
CTM_CONC_1	GRDDED3	Name and location of hourly 3-D instantaneous gas- and aerosol-phase pollutant estimates
CTM_CGRID_1	GRDDED3	Name and location of simulation-ending 3-D full CGRID (gas- and aerosol-phase pollutants) concentrations for use as a restart file
CTM_ACONC_1	GRDDED3	Name and location of hourly 2-D or 3-D integral average gas- and aerosol-phase pollutant estimates

File Name	Format	Description
CTM_DRY_DEP_1	GRDDED3	Name and location of hourly 3-D gas- and aerosol-phase dry deposition estimates
CTM_WET_DEP_1	GRDDED3	Name and location of hourly 3-D gas- and aerosol-phase wet deposition estimates
CTM_SSEMIS_1	GRDDED3	Name and location of hourly 2-D sea salt emissions; set the variable CTM_SSEMDIAG to "T" in CCTM to run script to write this additional file
CTM_VIS_1	GRDDED3	Name and location of hourly 3-D visibility metrics
CTM_DIAM_1	GRDDED3	Name and location of hourly 3-D aerosol diagnostics; dp and sigmas for Aitken and accumulation mode aerosol species; set the variable CTM_AERODIAG to "T" in CCTM run script to write this additional file
CTM_IPR_[1-3]	GRDDED3	Name and location of hourly 2-D or 3-D integrated process rate files; multiple files written when CCTM is configured to track a large number of process/pollutant combinations
CTM_IRR_[1-3]	GRDDED3	Name and location of hourly 2-D or 3-D integrated reaction rate files; multiple files written when CCTM is configured to track a large number of reaction/pollutant combinations
CTM_RJ_[1-2]	GRDDED3	Name and location of hourly photolysis diagnostic output file; multiple files written when there are a large number of photolytic reactions in a chemical mechanism

The default location of the CCTM output files is the `$M3DATA/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.

5.4 CHEMMECH

5.4.1 Description

The program CHEMMECH generates mechanism INCLUDE files for all chemical mechanism-dependent CMAQ programs. Using an ASCII mechanism definition file as input, a combination of the Fortran program CHEMMECH and the Python script include_generator.py (created by Chao-Jung Chien at University of California, Riverside) creates all of the Fortran INCLUDE files that define the gas-phase chemical mechanisms for the CMAQ programs.

To implement new mechanisms created by CHEMMECH in the CMAQ programs, manually move the output INCLUDE files from CHEMMECH to the `$M3MODEL/include/release/{NewMechanism}` directory within the CMAQ installation directories, where "NewMechanism" corresponds to the name of the new mechanism created with CHEMMECH. To invoke this new mechanism, set the *Mechanism* variable in the build script for the applicable CMAQ program to the name of the mechanism directory that you created. For example, if CHEMMECH was used to create INCLUDE files for a mechanism that is called "cb05_2006", create a directory `$M3MODEL/include/release/c05_2006`, move the INCLUDE files created by CHEMMECH to this directory, and compile the CMAQ programs for this new mechanism by setting the *Mechanism* variable within the build scripts to "cb05_2006".

CHEMMECH can be run in either mechanism processor (MP) or species processor (SP) mode. The MP mode generates new mechanism include files for CMAQ. The SP mode generates a new species table include file for CMAQ. If an existing mechanism (e.g. CB05) is modified by the user and no new species are added to the mechanism, CHEMMECH only needs to be run in MP mode. When new species are added or a new mechanism is being processed for CMAQ, CHEMMECH needs to also be run in SP model.

5.4.2 Files, configuration, and environment variables

Figure 5-3 shows the input and output files and configuration options for CHEMMECH. The full set of mechanism INCLUDE 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 RXDT.EXT and RXCM.EXT INCLUDE files are then input to the Python script include_generator.py, along with the Species_Table.csv file, to create the full set of INCLUDE files needed to compile CMAQ with a new chemical mechanism configuration. The Species_Table.csv file defines the atmospheric processes that each species undergoes in the model, and is generated by the Excel spreadsheet Species_Table.xls that is distributed with CHEMMECH.

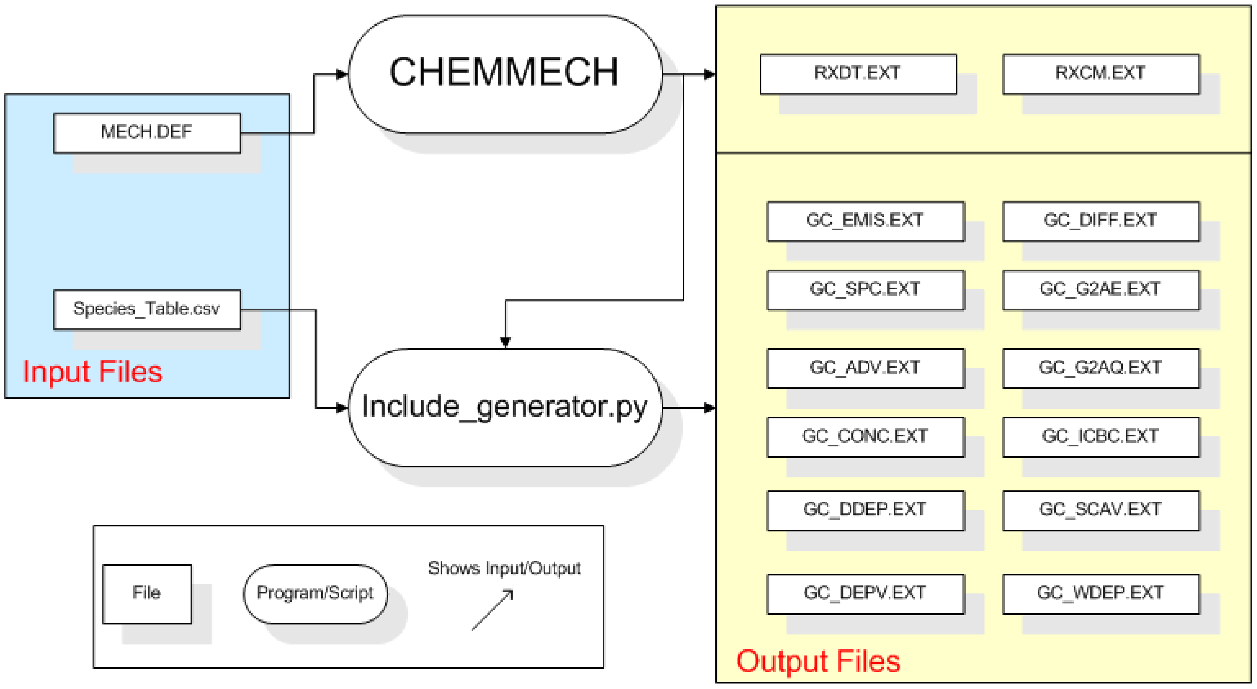


Figure 5-4. CHEMMECH input and output files

To implement a new mechanism in CMAQ, edit a MECH.DEF file that is associated with a base mechanism and provide this new MECH.DEF files as input to CHEMMECH. Edit the default run script MP.saprc99.csh to point to the new MECH.DEF file. This script will call both the CHEMMECH executable and the include_generator.py script to create the full set of mechanism INCLUDE files required by the CMAQ programs. There are no horizontal grid, vertical layer, or temporal settings for CHEMMECH.

5.4.2.1 CHEMMECH input files

Table 5-5. 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
SPST (Species_Table.csv)	ASCII/CSV	Species table; comma-delimited file generated from an Excel spreadsheet that lists the atmospheric processes that affect each model species

5.4.2.2 CHEMMECH compilation options

The configuration options listed here are set during compilation of the CHEMMECH executable. The compiler-specific options, such as the compiler names and flags, are set in the make.reader script distributed with the program. The directory paths of the CHEMMECH source code and working directory are set in the makeit script.

- `FC:[default: pgf90]`

  Fortran compiler path and name

- `FFLAGS:`

  Fortran compilation flags

- `CC:[default: cc]`

  C compiler path and name

- `C_FLAGS:`

  C compilation flags

- `BASE: [default: $cwd]`

  Working directory for compiling **CHEMMECH**

- `SRC: [default: ../]`

  Location of **CHEMMECH** source code

5.4.2.3 CHEMMECH compilation

To compile the **CHEMMECH** program, run the following command in the directory that contains the scripts:

```
./makeit MP
```

For the compilation to be successful, you must have the Portland Group Fortran and Gnu C compilers added to your path. To port **CHEMMECH** to other compilers, change the compiler names, locations, and flags in the make.reader script.

5.4.2.4 CHEMMECH execution options

The environment variables listed here are invoked during execution of the program and are set in the **CHEMMECH** run script. The default run script is called MP.saprc99.csh.

- `BASE: [default: $cwd]`

  Working directory path

- `Xpath: [default: $BASE]`

  Executable directory path

- `Mpath: [default: ../mech]`

  Directory path to the MECH.DEF file

- `Opath: [default: ../exts]`

  Output file directory path

- `APPL:`

  Simulation identifier

- `EXEC: [default: CHEMMECH]`

  Executable name

- `MCFL: [default: mech.def.saprc99]`

Mechanism definition file name

- `EXSPCS: [default: $Opath/SPECIES.ext]`

  Output file name when **CHEMMECH** is run in SP mode

- `EXRXDT: [default: $Opath/RXDT.EXT]`

  Name of output mechanism data INCLUDE file

- `EXRXCM: [default: $Opath/RXCM.EXT]`

  Name of output mechanism common INCLUDE file

5.4.2.5 CHEMMECH output files

Table 5-6. 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
GC_ADV.EXT	ASCII	File listing the gas-phase model species that are transported by advection
GC_CONC.EXT	ASCII	File listing the gas-phase model species to write to the CCTM output CONC file
GC_DDEP.EXT	ASCII	File listing the gas-phase model species that dry deposit
GC_DEPV.EXT	ASCII	File listing the gas-phase model species for which to calculate deposition velocities
GC_DIFF.EXT	ASCII	File listing the gas-phase model species that are transported by diffusion
GC_EMIS.EXT	ASCII	File listing the emitted gas-phase model species.
GC_G2AE.EXT	ASCII	File listing the gas-phase model species that react to form aerosols
GC_G2AQ.EXT	ASCII	File listing the gas-phase model species that have aqueous chemistry pathways
GC_ICBC.EXT	ASCII	File listing the gas-phase model species that require initial and boundary conditions
GC_SCAV.EXT	ASCII	File listing the gas-phase species that are scavenged by cloud water
GC_SPC.EXT	ASCII	File listing all of the gas-phase model species
GC_WDEP.EXT	ASCII	File listing the gas-phase model species that wet deposit

The default location of the CHEMMECH output files is the `exts` directory, controlled by the Opath variable in the run script. 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 $M3MODEL/include/release directory. Point the CMAQ build scripts to this new directory through the “Mechanism” variable.

5.5 ICON

5.5.1 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 *ModInpt* variable. This variable determines the input module to use when creating an ICON executable.

5.5.2 Files, configuration, and environment variables

Figure 5-4 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 determine the gas-phase chemistry and aerosol mechanism for which to calculate chemical ICs. Setting the *Mechanism* variable in the ICON compile script configures the program to use a specific set of mechanism INCLUDE files to build an executable. Separate ICON executables are required for each mechanism configuration.

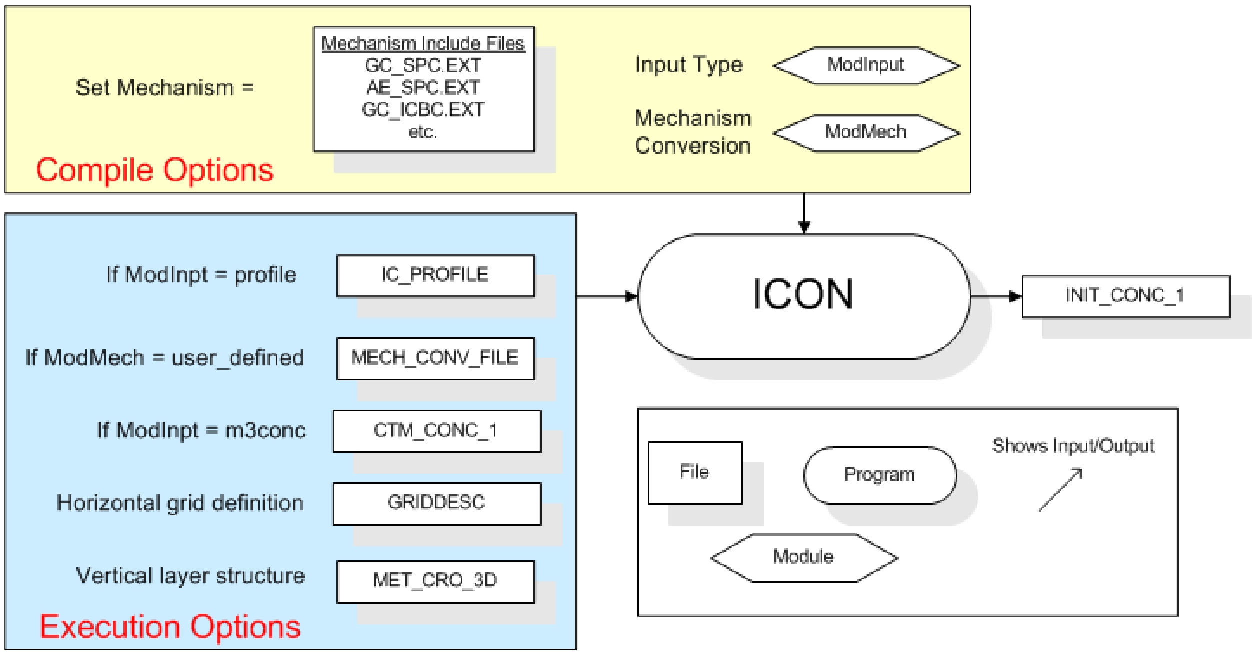


Figure 5-5. ICON input and output files

At execution the user provides a data file of chemical conditions that ICON converts to ICs on a predefined model grid. Through the specification of the *ModInpt* variable, 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. If the input file is not in the same chemical speciation as the simulation for which the user is creating ICs, it is necessary to specify a chemical conversion option by setting the *ModMech* variable at compilation. The default conversion profiles in ICON are set up to convert from RADM2 speciation to CB05 or SAPRC-99 chemistry. It is possible to create a custom conversion file (MECH_CONV_FILE) and input this file to ICON by setting the *ModMech* variable to "user_defined" at compilation.

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.

5.5.2.1 ICON input files

Table 5-7. 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"

MECH_CONV_FILE	ASCII	Mapping factors for converting between chemical mechanisms; this file is created by the user; used only when **ICON** is compiled with the ModMech configuration set to "user_defined"
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
MET_CRO_3D	GRDDED3	3-D cross-point meteorology file for defining the vertical layer structure of the model grid; this file is output by MCIP

5.5.2.2 ICON compilation options

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.

- `Opt: [default: verbose]`

  Defines the action to be taken by the program **M3BLD** when extracting source code from CVS and compiling an executable.

  - *compile_all:* force compile, even if all the object files are current
  - *clean_up:* remove all source files upon successful compilation
  - *no_compile:* do everything except compile
  - *no_link:* do everything except link
  - *one_step:* compile and link in one step
  - *parse_only:* check configuration file syntax
  - *show_only:* show requested commands but do not execute them
  - *verbose:* show requested commands as they are executed

- `MakeOpt`

  Uncomment to build a Makefile to compile the executable.

- `ModInpt: [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

- `ModMech: [default: module radm2_to_cb05 ]`

  Defines whether the input initial conditions data need to be converted from one chemical mechanism to another.

  - *mc_noop*: do not perform any mechanism conversion; used when extracting initial conditions from a CCTM CONC file for a nested simulation or when the input profiles are already formatted for the correct mechanism
  - *user_defined*: input the file defined by the MECH_CONV_FILE variable; used for custom mechanism conversions
  - *radm2_to_cb05*: convert the input initial conditions profiles from RADM2 to CB05 speciation
  - *radm2_to_saprc99*: convert the input initial conditions profiles from RADM2 to SAPRC-99 speciation

- `Mechanism: [default: cb05cl_ae5_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 $M3MODEL/include/release directory. Examples include:

  - *cb05ae5_aq*: CB05 gas-phase mechanism, fifth-generation CMAQ aerosol mechanism with sea salt, aqueous/cloud chemistry
  - *cb05cl_ae5_aq*: CB05 gas-phase mechanism, fifth-generation CMAQ aerosol mechanism with sea salt, aqueous/cloud chemistry, and active chlorine
  - *cb05cltx_ae5_aq*: CB05 gas-phase mechanism, fifth-generation CMAQ aerosol mechanism with sea salt, aqueous/cloud chemistry, active chlorine, and air toxics
  - *cb05cltxhg_ae5_aq*: CB05 gas-phase mechanism, fifth-generation CMAQ aerosol mechanism with sea salt, aqueous/cloud chemistry, active Chlorine, air toxics, and mercury; this is for the multipollutant model (see CMAQv4.7 release notes)

- *saprc99_ae5_aq*: SAPRC-99 gas-phase mechanism, fifth-generation CMAQ aerosol mechanism with sea salt, aqueous/cloud chemistry
- *saprc99tx3_ae5*: SAPRC-99 gas-phase mechanism with air toxics and fifth-generation CMAQ aerosol mechanism

5.5.2.3 ICON compilation

First, it is assumed that you have already installed and compiled the I/O API and netCDF libraries (see Section 3.2.3), or that these are already available from a previous CMAQ compilation.

Section 3.3 provides an overview of how to install and compile the CMAQ programs for the tutorial simulation. Follow the steps outlined in Section 3.3 (summarized below) to compile new versions of **ICON**:

- If you have not already done so, compile **M3BLD**, the CMAQ source code and compilation management program. This needs to be done only once—the first time CMAQ is installed.
- If needed, configure the **ICON** build script to use the available I/O API and netCDF libraries.
- If you have not already done so, compile the STENEX library.
- Configure the **ICON** build script for your application (using the options discussed in Section 5.5.2.2)
- Invoke the build script to create an executable:

```
./bldit.icon
```

5.5.2.4 ICON execution options

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

- `EXEC: [default: ICON_${CFG}_Linux2_x86_64_pg]`

  Executable to use for the simulation.

- `NPCOL_NPROW: [default: 1 1]`

  Domain decomposition for parallel mode; ICON is normally run in a single-processor environment, so this setting should always be “1 1”.

- `GRIDDESC: [default: ../GRIDDESC1]`

Grid description file for setting the horizontal grid definition.

- `GRID_NAME: [default: [M_36_2001]`

  Name of the grid definition contained in the GRIDDESC file that specifies the horizontal grid for the current application of the model.

- `LAYER_FILE:[default: $m3DATA/mcip3/M_36_2001/metcro3d_010722]`

  Name and location of a MET_CRO_3D file for specifying the vertical layer structure for the current application of the model

- `OUTDIR: [default: $M3DATA/icon]`

  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**. Also optionally looks for the variable MECH_CONV_FILE to point to a user-defined mechanism conversion file.
  - *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:[default: 2001203]`

  Sets the Julian date to use in naming the **ICON** output file for nested runs.

- `SDATE: [default: ${DATE}]`

  Julian start date for extracting initial conditions from a CCTM CONC file for a nested simulation. If SDATE is not set, ICON will use the first hour of the CTM_CONC_1 file.

- `STIME: [default: 000000]`

  Start time for extracting initial conditions from a CCTM CONC file for a nested simulation. If SDATE is not set, ICON will use the first hour of the CTM_CONC_1 file.

5.5.2.5 ICON output files

Table 5-8. 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 `$M3DATA/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.

5.6 JPROC

5.6.1 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.

5.6.2 Files, configuration, and environment variables

Figure 5-5 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.

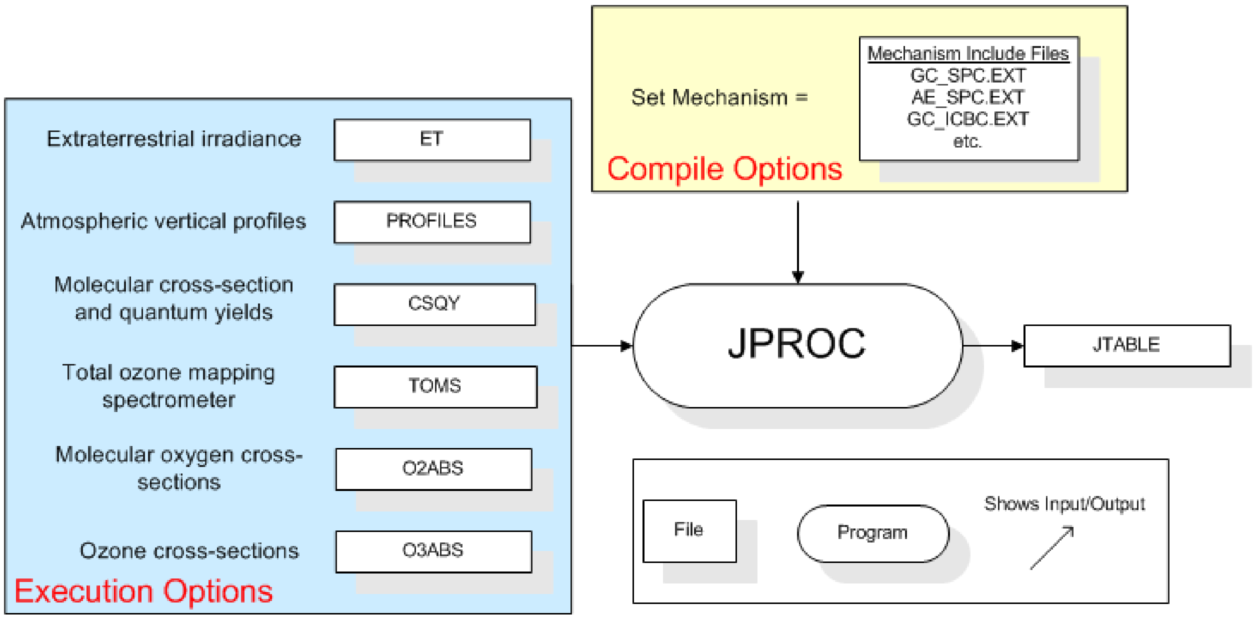


Figure 5-6. JPROC input and output files

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 CB05 and SAPRC-99 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.

5.6.2.1 JPROC input files

Table 5-9. 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

5.6.2.2 JPROC compilation options

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.

- `Opt: [default: verbose]`

  Defines the action to be taken by the program **M3BLD** when extracting source code from CVS and compiling an executable.

  - *compile_all:* force compile, even if all the object files are current
  - *clean_up:* remove all source files upon successful compilation
  - *no_compile:* do everything except compile
  - *no_link:* do everything except link
  - *one_step:* compile and link in one step
  - *parse_only:* check configuration file syntax
  - *show_only:* show requested commands but do not execute them
  - *verbose:* show requested commands as they are executed

- `MakeOpt`

  Uncomment to build a Makefile to compile the executable.

- `Mechanism:`

  Specifies the gas-phase, aerosol, and aqueous-phase chemical mechanisms for which to create photolysis rates. The choices for the *Mechanism* variable are the mechanism directory names under the $M3MODEL/include/release directory. Examples include:

  - *cb05ae5_aq*: CB05 gas-phase mechanism, fifth-generation CMAQ aerosol mechanism with sea salt, aqueous/cloud chemistry

  - *cb05cl_ae5_aq*: CB05 gas-phase mechanism, fifth-generation CMAQ aerosol mechanism with sea salt, aqueous/cloud chemistry, and active chlorine

  - *cb05cltx_ae5_aq*: CB05 gas-phase mechanism, fifth-generation CMAQ aerosol mechanism with sea salt, aqueous/cloud chemistry, active chlorine, and air toxics

  - *cb05cltxhg_ae5_aq*: CB05 gas-phase mechanism, fifth-generation CMAQ aerosol mechanism with sea salt, aqueous/cloud chemistry, active Chlorine, air toxics, and mercury; this is for the multipollutant model (see CMAQv4.7 release notes)

- *saprc99_ae5_aq*: SAPRC-99 gas-phase mechanism, fifth-generation CMAQ aerosol mechanism with sea salt, aqueous/cloud chemistry
- *saprc99tx3_ae5*: SAPRC-99 gas-phase mechanism with air toxics and fifth-generation CMAQ aerosol mechanism

5.6.2.3 JPROC compilation

First, it is assumed that you have already installed and compiled the I/O API and netCDF libraries (see Section 3.2.3), or that these are already available from a previous CMAQ compilation.

Section 3.3 provides an overview of how to install and compile the CMAQ programs for the tutorial simulation. Follow the steps outlined in Section 3.3 (summarized below) to compile new versions of **JPROC**:

- If you have not already done so, compile **M3BLD**, the CMAQ source code and compilation management program. This needs to be done only once—the first time CMAQ is installed.
- If needed, configure the **JPROC** build script to use the available I/O API and netCDF libraries.
- Configure the **JPROC** build script for your application (using the options discussed in Section 5.6.2.2)
- Invoke the build script to create an executable:

```
./bldit.jproc
```

5.6.2.4 JPROC execution options

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

- `EXEC: [default: JPROC_${CFG}]`

  Executable to use for the simulation

5.6.2.5 JPROC output files

Table 5-10. 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 `$M3DATA/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.

5.7 MCIP

5.7.1 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 also calculated by the current version of MCIP; however, because CMAQv4.7 can now calculate all deposition velocities, MCIPv3.4 will be the last version of MCIP to calculate those velocities. In addition, the user has the option of either passing through the planetary boundary layer (PBL) and radiation fields that are calculated by the meteorological model (recommended) or recalculating those fields with MCIP. The MCIP run script contains Fortran namelist variables that control these configuration settings.

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 Section 5.7.2.4, "MCIP execution options." 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.

5.7.2 Files, configuration, and environment variables

Figure 5-6 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.

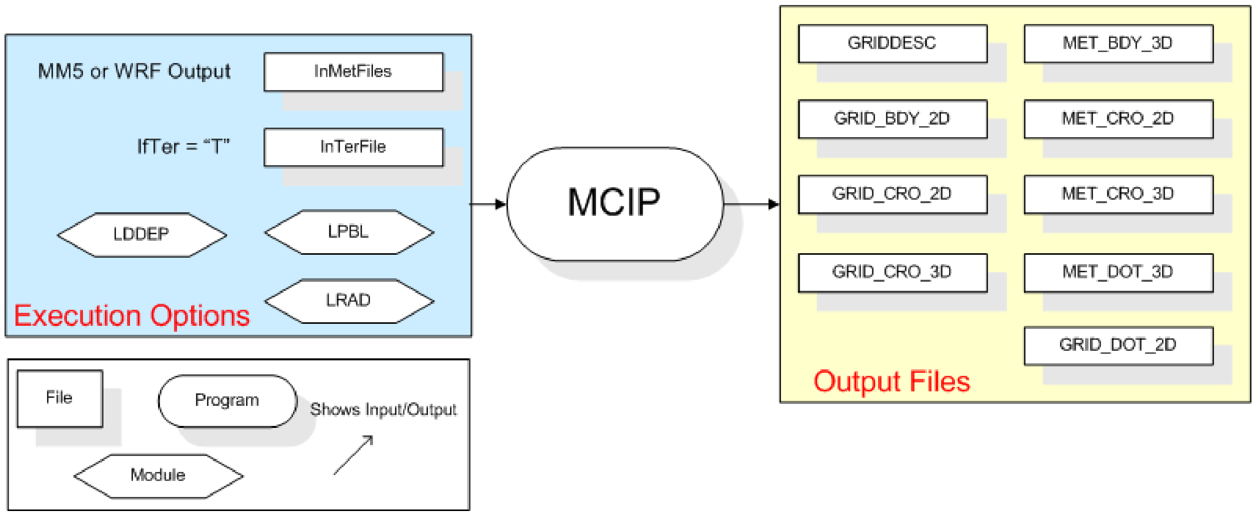


Figure 5-7. MCIP input and output files

5.7.2.1 MCIP input files

Table 5-11. 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.

5.7.2.2 MCIP compilation options

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.

5.7.2.3 MCIP compilation

First, it is assumed that you have already installed and compiled the I/O API and netCDF libraries (see Section 3.2.3), or that these are already available from a previous CMAQ compilation.

Section 3.3 provides an overview of how to install and compile the CMAQ programs for the tutorial simulation. Follow the steps outlined in Section 3.3 (summarized below) to compile new versions of **MCIP**:

- Configure the Makefile for the current operating system/compiler combination. Comment out the configuration that does not apply to the current system. Uncomment the configuration that is closest to that of the current system and make the necessary changes to point to the compiler path, I/O API location, and netCDF locations on the current system.

- Invoke the Makefile to create an executable by typing the following command in the directory that contains the Makefile and **MCIP** source code:

  ```
  ./make
  ```

5.7.2.4 MCIP execution options

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

- `APPL:`

  Application name; scenario ID for file naming

- `CoordName:`

  Coordinate system name of the MCIP output grid that is written to the GRIDDESC file

- `GridName:`

  Model grid name of the MCIP output grid that is written to the GRIDDESC file

- `DataPath:`

  Input/output data directory path

- `InMetDir:`

  Path of the input data directory containing the MM5 or WRF-ARW output data files

- `InTerDir:`

  Path of the input data directory containing the MM5 TERRAIN file; not compatible with WRF-ARW.

- `OutDir: [default: $M3DATA/mcip]`

  Path of the **MCIP** output data directory

- `ProgDir:[default: $cwd]`

  Working directory containing the **MCIP** executable

- `WorkDir:`

  Temporary working directory for Fortran links and the namelist file

- `InMetFiles:`

  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

- `IfTer:[default: T]`

  Binary flag indicating the availability of an input MM5 TERRAIN file; options include T (true) or F (false)

- `InTerFile:`

  Name and location of input MM5 TERRAIN file

- `LPBL: [default: 1]`

  Sets the source of the PBL values in the **MCIP** output files. The setting of this variable determines whether to use PBL values from the input meteorology file or to recalculate those values within **MCIP**.

  - *1*: use PBL values from the input meteorology
  - *2*: recalculate PBL values within **MCIP** using PBL similarity
  - *3*: recalculate PBL values within **MCIP** using surface layer similarity

- `LRAD:[default: 1]`

  Sets the source of the radiation fields in the **MCIP** output files. The setting of this variable determines whether to use the radiation fields from the input meteorology or to recalculate those fields within **MCIP**.

  - *1*: use radiation fields from the input meteorology
  - *2*: recalculate radiation fields using the **MCIP** version 1 algorithm

- `LDDEP [default: 1]:`

  Sets the dry deposition routine. The setting of this variable determines which dry deposition routine is used for calculating deposition velocities.

  - *1*: use the RADM (Wesely) dry deposition routine (Wesely, 1989)
  - *2*: use the Models-3 (Pleim) dry deposition routine (Pleim et al., 2001)
  - *3*: Use the Models-3 dry deposition routine and include chlorine
  - *4*: Use the Models-3 dry deposition routine and include chlorine and mercury

- `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

5.7.2.5 MCIP output files

Table 5-12. 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
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 `$M3DATA/mcip3/$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.

5.8 PARIO

5.8.1 Description

The parallel input/output (**PARIO)** library contains modules for controlling the model input and output in parallel multiprocessor computing environments. In addition to a series of CMAQ-specific routines, it contains special implementations of several of the I/O API modules for multiprocessor computing environments.

The **PARIO** library is necessary only when compiling CCTM for parallel multiprocessor environments; single-processor versions of CCTM and the other CMAQ programs do not use the **PARIO** library.

5.8.2 Files, configuration, and environment variables

5.8.2.1 PARIO input files

PARIO does not require any input files.

5.8.2.2 PARIO compilation options

Other than configuring the build script for the current system (i.e., compiler and library locations), **PARIO** does not require any configuration at compilation.

5.8.2.3 PARIO compilation

First, it is assumed that you have already installed and compiled the I/O API, netCDF, and MPICH libraries (see Section 3.2.3), or that these are already available from a previous CMAQ compilation.

Section 3.3 provides an overview of how to install and compile the CMAQ programs for the tutorial simulation. Follow the steps outlined in Section 3.3 (summarized below) to compile new versions of **PARIO**:

- If you have not already done so, compile **M3BLD**, the CMAQ source code and compilation management program. This needs to be done only once—the first time CMAQ is installed.
- If needed, configure the **PARIO** build script to use the available I/O API and MPICH libraries.
- Invoke the build script to create an executable:

```
./bldit.pario
```

5.8.2.4 PARIO execution options

Because **PARIO** is not a program, it does not have an associated run script.

5.8.2.5 PARIO output files

Successful compilation of **PARIO** will produce the library file libpario.a. along with several module files in the `$M3LIB/pario/$OS` directory.

5.9 PROCAN

5.9.1 Description

Process analysis is a diagnostic tool that captures model-generated data not routinely output by models and provides quantitative information on the contributions of individual physical and chemical processes to model predictions. This quantitative information can then be used to form a picture of how the model obtains its predictions. In the CMAQ modeling system, two types of process analysis data can be captured during a CCTM simulation: integrated process rates (IPRs) and integrated reaction rates (IRRs). IPRs give the contributions of individual physical processes and the net effect of chemical reactions to the overall model concentrations. For example, IPR analysis can be used to determine the quantitative contribution of horizontal transport, vertical transport, chemistry, and emissions to the predicted hourly concentration of ozone in a particular grid cell. IRRs give the contributions of individual chemical reactions to the net effects of chemical reaction on species concentrations. As an example, IRR can be used to calculate the mass throughput for a particular reaction sequence in a photochemical mechanism for a particular grid cell and time period. Because the amount of IPR and IRR data that can be obtained may be large and the analysis of such data can be fairly complex, the user is advised to read Chapter 16 in Byun and Ching (1999) before attempting to use this tool. The discussion here focuses primarily on the procedures that must be followed to capture process analysis data, rather than on what data should be captured and how they might be analyzed.

5.9.2 Files, configuration, and environment variables

The program **PROCAN** creates a set of three output INCLUDE files needed to instrument CCTM to produce process analysis output (See Figure 5-7). This program reads and interprets instructions from a command file and then generates three Fortran INCLUDE files used by CCTM to produce the process analysis outputs that were requested in the commands. The process analysis commands themselves are formatted according to a simple set of rules and a free-form format. Nevertheless, each command has a special syntax that must be followed, and each command makes use of special keywords and/or operators that have specific meaning to **PROCAN**. The commands are of three major types: global commands, IPR commands, and IRR commands. The discussion begins first, however, with a description of some general rules for configuring **PROCAN**.

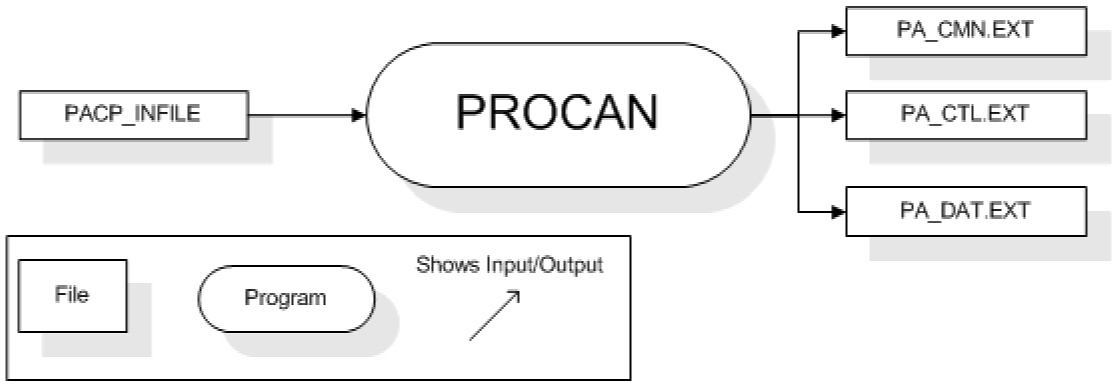


Figure 5-8. PROCAN input and output files

5.9.2.1 PROCAN input files

Table 5-13. PROCAN input files

File Name	Format	Description
PACP_INFILE	ASCII	PROCAN command file specifies configuration options for the program

5.9.2.2 PROCAN configuration

PROCAN configuration is implemented through the command file PACP_INFILE. The free-form format of the **PROCAN** command file is similar to that used by the general mechanism processor, CHEMMECH. In general, white spaces are ignored and line wrap is allowed (i.e., commands can be continued on a subsequent line after a hard return). The free-form format also allows embedded comments and makes use of special symbols to indicate the type of input data.

Special rules for naming species, entering labels, and specifying numerical values, such as stoichiometric coefficients and rate constant parameters, are also used. Each major component of the command file is discussed below.

- *Comments.* All lines that have an exclamation point in column 1 are treated as comment lines and are ignored by **PROCAN**. Any text enclosed in braces ({ }) or parentheses (( )) is also treated as comment and ignored by **PROCAN**.
- *Species names*. **PROCAN** recognizes two types of species names: model species and user-defined process analysis species. "Model species" refer to species names in the mechanism species tables.These names must be spelled exactly as they appear in that table. For user-defined species names, the following special rules have been established:
    - The process analysis species names must not contain any blanks and can be up to 16 characters long.
    - The name must begin with an alphabetic character but may contain any alphanumeric character (i.e., "A-Z", "a-z", and "0-9") or the characters ":" and "_" after the first position.
    - The name is case sensitive. Thus, NO2 and no2 would represent two different species.
    - A name can have embedded comments but cannot span two lines.
- *Label names*. For some of the IRR commands, reaction labels appearing in the chemical mechanism reaction list input file can be referenced. These labels would normally be spelled exactly as they appear in the chemical mechanism reaction list input file, except embedded comments and their delimiters should be omitted. However, any embedded blanks in those label names should be omitted, and the label name should contain no more that 16 nonblank characters.
- *Numbers.* Numerical inputs in the command file can be either integer (e.g., 5), floating point (e.g., 5.0), or exponential (e.g., 5.0E+00). With the exponential format, the "E" may be either upper or lowercase; a positive exponent will be assumed if the sign of the exponent is missing.
- *Command line terminator.* Terminate input command lines with a semicolon.

5.9.2.3 PROCAN commands

This section describes the individual process analysis commands that are used to construct a **PROCAN** command file. In the description of these commands, the following conventions are used: bold type is used for **PROCAN** keywords, and normal type for user-supplied inputs. Alternative inputs are separated by vertical bars (|), and optional inputs are enclosed in braces ({}). The **PROCAN** commands fall into three general categories:

1. *Process Analysis Global Commands*. These commands (Table 5-14) include general specifications that are applicable throughout the configuration.
2. *Integrated Process Rate Command*. This command (Tables 5-15 and 5-16) is specific to the configuration of the integrated process rates. There is only one command for IPRs, and it controls the specific IPRs that are output. Note that one command can cause many IPR outputs to be generated. For example, if one species or family is specified in a

command but no process codes are specified, then one IPR will be generated for each science process for that species. Similarly, if the keyword ALL is used for the species name and no process code is specified, 12 IPRs will be generated for every model species. This would generate an output file that would be approximately 12 times as large as the corresponding concentration file. Also, the impact on the CCTM memory requirements would be substantial, since adding a single IPR output has roughly the same effect as adding a model species. Thus, caution should be exercised when formulating the commands to request IPR outputs. The IPR outputs are written to an I/O API output file in exactly the same format and with the same number of time steps as the concentration output file. Since the I/O API currently has a limit of 120 output variables in a file, multiple files will be output if this limit is exceeded.

3. *Integrated Reaction Rate Commands.* These commands (Table 5-17) are specific to the configuration of the IRRs. The same considerations regarding file size and memory usage detailed in item 2 above should be considered when using these commands.

Table 5-14. Process analysis global commands

Command	**Description**
DEFINE FAMILY familyname = {c_1*****}$species_1$ {+ {c_2*****}$species_2$ + ...}**;**	The DEFINE FAMILY command is used to define a group of species as members of a family. The user-specified "familyname" must be unique, and can be referenced in subsequent commands. The c_i are numerical coefficients that default to 1 if not specified; "$species_i$" represents the model species names.
ENDPA;	The ENDPA command signifies the end of the command input in the PROCAN command file.

Table 5-15. Integrated process rate output commands

Command	**Description**
IPR_OUTPUT species|familyname|**ALL =** {$pcode_1$ + $pcode_2$ + ...}**;**	The IPR_OUTPUT command defines specific IPR outputs to be generated during a CMAQ CTM simulation. A model species name, family name, or the keyword "ALL" must follow the IPR_OUTPUT keyword. The keyword ALL refers to all model species. IPRs are generated for the selected species or family, and they are controlled by the specified values of $pcode_i$, where $pcode_i$ corresponds to one of the process codes listed below. If no process codes are specified, IPRs will be generated for every science process (i.e., the first 12 codes shown in Table 5-16). The output variables that are generated are named either species_$pcode_i$ or familyname_$pcode_i$.

Table 5-16. Integrated process rates process codes

Process	**Definition**

Yamo	PPM	
—	**XADV**	Advection in the E-W direction for the PPM scheme
—	**YADV**	Advection in the N-S direction for the PPM scheme
ZADV	**ZADV**	Vertical advection
HADV	**ADV2**	Total horizontal advection (XADV+YADV)
MADV	**ADV3**	Total advection (XADV+YADV+ZADV)
—	**TADV**	Total advection for the PPM scheme (ADV3+ADJC)
—	**ADJC**	Mass adjustment for the PPM scheme
HDIF	**HDIF**	Horizontal diffusion
VDIF	**VDIF**	Vertical diffusion
EMIS	**EMIS**	Emissions
DDEP	**DDEP**	Dry deposition
CHEM	**CHEM**	Chemistry
AERO	**AERO**	Aerosols
CLDS	**CLDS**	Cloud processes and aqueous chemistry
TDIF	**TDIF**	Total diffusion (HDIF+VDIF)
TRNM	**TRAN**	Total transport (advection + diffusion)

Table 5-17. Integrated reaction rate output commands

Command	Description
IRRTYPE = FULL|PARTIAL|NONE;	The **IRRTYPE** command defines the type of IRR analysis. With the type set to FULL, IRRs for each individual reaction will be calculated and written to the IRR output file, and all other IRR commands will be ignored. **IRRTYPE** set to PARTIAL indicates that the IRR commands following this command are to be processed to produce user-defined IRR outputs. Type set to NONE causes all IRR commands to be ignored and no IRR output to be generated. If the command is omitted, type PARTIAL is assumed.
DEFINE CYCLE cyclename = species$_1$**;**	The **DEFINE CYCLE** command is used to compute the net of all chemical production and loss of a species. Thus, this quantity is computed by summing the IRRs for all reactions in which a species is consumed, and then subtracting that sum from the sum of the IRRs for all reactions in which the species is produced. The "cyclename" is a user-defined name that must be unique, and can be referenced in subsequent IRR_OUTPUT

	commands.
DEFINE RXNSUM sumname = {c_1*****}<rxlabl$_1$> { ± {c_2*****} <rxlabl$_2$> ± ...}**;**	The **DEFINE RXNSUM** command is used to compute a linear combination of the IRRs for individual reactions that can then be referenced in a subsequent IRR_OUTPUT command; "sumname" is user-defined and must be unique. The linear combination of IRRs is defined according to the expressions following the equal sign that specify which reaction's IRRs to sum. The "rxlabl$_i$" is the reaction label that is used in the generalized mechanism. The "c_i" are optional numerical coefficients that default to 1 if not specified.
IRR_OUTPUT irrname = {c_1*****}op$_1$|cyclname{qual$_1$}| sumname{qual$_1$}| <rxlabl$_1$>{ ± {c_2*****}op$_2$ | cyclname{qual$_2$}| sumname{qual$_2$}| <rxlabl$_2$> + ...}**;**	The **IRR_OUTPUT** command defines a specific IRR output to be generated during a CMAQ simulation. It is constructed by specifying a linear combination of IRR operators, IRR global definitions, or IRRs for specified reactions. Each individual term in the combination must include either one of the five IRR operators (i.e., op$_i$), a cycle name, a reaction sum name, or a reaction label enclosed in "greater than" and "less than" signs. The optional qualifiers (qual$_i$) for cycle name or reaction sum name can be either POSONLY or NEGONLY. With these qualifiers, the defined quantity is included as a term only when it is positive or negative, respectively. If the name is not qualified, the quantity is included regardless of sign. The numerical coefficients for each term (c_i) are assumed to be 1 unless they are explicitly included. The irrname that is supplied by the user will be assigned as the variable name in the I/O API IRR output file.
DESCRIPTION = 'description'**;**	The **DESCRIPTION** command is provided to allow the user to specify a long description of the output variable that will be included on the I/O API IRR output name. If a description is not specified for an IRR_OUTPUT variable, the irrname (or short name) will be used in the output file. If the description command is used, it should be located immediately following the IRR_OUTPUT command to which it applies.
PROD [species$_1$**]** {**FROM[**species$_2$**]** {**AND**|**OR [**species$_3$**]** }}	The **PROD** operator is used to compute the total production of a species by summing the IRRs of all reactions in which species$_1$ appears as a

	product. The optional qualifiers FROM and AND/OR restrict the sum to include only those reactions in which $species_2$ and/or $species_3$ are reactants. The "$species_i$" can be any gas-phase mechanism species or a family of gas-phase species; "$species_2$" or "$species_3$" may also be the keyword HV to restrict the selection to photolytic reactions.
NETP [$species_1$**]** {**FROM[**$species_2$**]** {**AND**|**OR [**$species_3$**]** }}	The **NETP** operator is very similar to the **PROD** operator, as it also is used to compute the production of a species. Whereas the **PROD** operator includes every reaction in which $species_1$ occurs as a product, the **NETP** operator includes only those reactions in which the net production of $species_1$ is greater than zero. Thus, if $species_1$ or any member of the species family appears as both a reactant and a product with equal stoichiometry in a reaction, the **PROD** operator will include it, but the **NETP** operator will not. This operator is useful for getting the net production of a family, for example.
LOSS[$species_1$**]** {**AND**|**OR [**$species_2$**]** }	The **LOSS** operator is used to compute the total loss of a species by summing the IRRs of all reactions in which $species_1$ appears as a reactant. The optional qualifier AND restricts the sum to include only those reactions in which both $species_1$ and $species_2$ are reactants. Similarly, the OR qualifier includes all reactions in which either "$species_1$" or "$species_2$" appears as a reactant, where "$species_1$" or "$species_2$" can be any gas-phase species in the mechanism, a family name that includes only gas-phase mechanism species, or the keyword HV to restrict the selection of reactions to those that are photolytic.
NETL[$species_1$**]** {**AND**|**OR [**$species_2$**]** }}	The **NETL** operator is very similar to the **LOSS** operator, as it also is used to compute the loss of a species. However, it includes only those reactions in which there is a net loss of "$species_1$" and/or "$species_2$". Thus, if $species_1$ or any member of the species family appears as both a reactant and a product with equal stoichiometry in reaction, the **NETL** operator will not include it in summing the loss of that species, whereas the **LOSS** operator will include the IRR for that reaction.
NET[$species_1$**]**	The **NET** operator gives the net of the production

	and the loss of a species for all reactions in which "species$_1$" appears either as reactant or a product; "species$_1$" may be any gas-phase, mechanism species or any family consisting wholly of gas-phase mechanism species.

5.9.2.4 PROCAN execution options

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

- `EXEC: [default: PACP_${CFG}]`

  Executable to use for the simulation

- `PACP_INFILE: [default: $M3DATA/procan/pacp.inp]`

  PROCAN control file for setting process analysis configuration

5.10 STENEX

5.10.1 Description

The stencil exchange (**STENEX**) library contains modules for controlling the communication between processors in parallel multiprocessor computing environments. A “noop” version of the library is required for single-processor versions of CCTM and ICON; the parallel version is required when compiling for parallel multiprocessor versions of CCTM.

5.10.2 Files, configuration, and environment variables

5.10.2.1 STENEX input files

STENEX does not require any input files.

5.10.2.2 STENEX compilation options

Other than configuring the build script for your system (i.e., compiler and library locations), **STENEX** does not require any configuration at compilation.

5.10.2.3 STENEX compilation

First, it is assumed that you have already installed and compiled the I/O API, netCDF, and MPICH libraries (see Section 3.2.3), or that these are already available from a previous CMAQ compilation.

Section 3.3 provides an overview of how to install and compile the CMAQ programs for the tutorial simulation. Follow the steps outlined in Section 3.3 (summarized below) to compile new versions of **STENEX**:

- If you have not already done so, compile **M3BLD**, the CMAQ source code and compilation management program. This needs to be done only once—the first time CMAQ is installed.
- If needed, configure the **STENEX** build script to use the available I/O API and MPICH libraries.
- Invoke the single-processor build script to create serial executables:

  ```
  ./bldit.se_noop
  ```

- Invoke the multiprocessor build script to create parallel executables:

  ```
  ./bldit.se.Linux
  ```

5.10.2.4 STENEX execution options

Because **STENEX** is not a program, it does not have an associated run script.

5.10.2.5 STENEX output files

Successful compilation of **STENEX** will produce the library files libsef90_noop.a for serial compilations and libse_snl.a for parallel compilations, along with several module files in the `$M3LIB/stenex/$OS` directory.

5.11 References

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.

6. CMAQ FILES

The input files for CMAQv4.7 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 CMAQv4.7 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.

6.1 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 6-1 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 6-1; 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 CMAQv4.7 release notes for additional file information.

Table 6-1. CMAQ Input File Information

File Name	File Type	Time-Dependence	Spatial Dimensions	Source
General				
GRIDDESC (horizontal domain definition)	ASCII	n/a	n/a	user/MCIP
ICON				
IC_PROFILE (initial conditions vertical profiles)	ASCII	Annual	n/a	user
CTM_CONC_1 (CCTM concentration files)	GRDDED3	Hourly	X*Y*Z	CCTM
MECH_CONV_FILE (mechanism conversion file)	ASCII	n/a	n/a	user
MET_CRO_3D (3-D meteorological cross-point fields)	GRDDED3	Hourly	X*Y*Z	MCIP
BCON				
BC_PROFILE (boundary conditions vertical profiles)	ASCII	Annual	n/a	user

CTM_CONC_1 (CCTM concentration files)	GRDDED3	Hourly	X*Y*Z	CCTM
MECH_CONV_FILE (mechanism conversion file)	ASCII	n/a	n/a	user
MET_CRO_3D (3-D meteorological cross-point fields)	GRDDED3	Hourly	X*Y*Z	MCIP
JPROC				
ET (extraterrestrial irradiance)	ASCII	Annual	n/a	user
PROFILES (default atmospheric profiles)	ASCII	Annual	n/a	user
O2ABS (O2 absorption)	ASCII	Annual	n/a	user
O3ABS (O3 absorption)	ASCII	Annual	n/a	user
TOMS (total ozone mapping spectrometer data)	ASCII	varies	n/a	user
CSQY (absorption cross section and quantum yields)	ASCII	Annual	n/a	User
MCIP				
InMetFiles (list of MM5 or WRF-ARW output files)	Binary or netCDF	typically hourly, but sometimes sub-hourly	X*Y*Z	MM5 or WRF-ARW
InTerFile (MM5 terrain file)	Binary	n/a	X*Y	MM5
CCTM				
INIT_CONC_1 (initial conditions)	GRDDED3	Time-invariant	X*Y*Z	ICON/CCTM
BNDY_CONC_1 (boundary conditions)	BNDARY3	Hourly	[2(X+1)+2(Y+1)]*Z	BCON
XJ_DATA (photolysis rates look-up table)	ASCII	Daily	n/a	JPROC
EMIS_1 (Emissions)	GRDDED3	Hourly	X*Y*Z	SMOKE
OCEAN_1 (sea salt mask)	GRDDED3	Time-invariant	X*Y	Spatial Allocator
GSPRO (speciation profiles)	ASCII	Time-invariant	N/a	User
B3GRD (grid-normalized biogenic emissions)	GRDDED3	Time-invariant	X*Y	SMOKE
BIOSEASON (freeze dates)	GRDDED3	Time-invariant	X*Y	Metscan
STK_GRPS_## (stack groups)	GRDDED3	Time-invariant	X*Y	SMOKE
STK_EMIS_## (point-source emissions)	GRDDED3	Hourly	X*Y	SMOKE
GRID_CRO_2D (2-D grid cross-point fields)	GRDDED3	Time-invariant	X*Y	MCIP
GRID_DOT_2D (2-D grid dot-point fields)	GRDDED3	Time-invariant	(X+1)*(Y+1)	MCIP
MET_BDY_3D (3-D meteorological boundary input)	BNDARY3	Hourly	PERIM*Z	MCIP
MET_CRO_2D (2-D meteorological cross-point fields)	GRDDED3	Hourly	X*Y	MCIP
MET_CRO_3D (3-D meteorological cross-point fields)	GRDDED3	Hourly	X*Y*Z	MCIP

MET_DOT_3D (3-D meteorological dot-point fields)	GRDDED3	Hourly	(X+1)*(Y+1)*Z	MCIP

6.1.1 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 is an I/O API-related 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 6-2) 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 6-3) 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 6-2. Coordinate system 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)
	B	P_ALP	Double	First map projection descriptive parameter (dependent on projection type)
	C	P_BET	Double	Second map projection descriptive parameter (dependent on projection type)
	D	P_GAM	Double	Third map projection descriptive parameter (dependent on

				projection type)
	E	XCENT	Double	Longitude for coordinate system center
	F	YCENT	Double	Latitude for coordinate system center

Table 6-3. 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
3	A	COORD-NAME	String	Name of the coordinate description in the previous section (required); single quote delimited
	B	XORIG	Double	X-coordinate for lower-left (southwest) corner of the grid with respect to (XCENT,YCENT) (dependent on projection type)
	C	YORIG	Double	Y-coordinate for lower-left (southwest) corner of the grid with respect to (XCENT,YCENT) (dependent on projection type)
	D	XCELL	Double	X-coordinate grid cell size (dependent on projection type)
	E	YCELL	Double	Y-coordinate grid cell size (dependent on projection type)
	F	NCOLS	Int	Number of horizontal grid columns (dependent on projection type)
	G	NROWS	Int	Number of horizontal grid rows (dependent on projection type)
	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.

6.1.2 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 Section 5.5 on ICON). 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 Section 6.1.5.

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 6-4. The header of the profiles file is list-formatted, while the data section of the file is fixed format.

Table 6-4. 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)
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
```

6.1.3 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 Section 6.1.5.

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 µg 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 6-5. The header of the profiles file is list-formatted, while the data section of the file is fixed format.

Table 6-5. 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)
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)
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
```

6.1.4 MECH_CONV_FILE: Mechanism conversion file

Used by: ICON, BCON

MECH_CONV_FILE is the logical name of the ASCII input file that contains photochemical mechanism conversion rules. This file contains expressions that are used to convert ICs and BCs from one mechanism form to another. The default IC and BC profiles distributed with CMAQ contain RADM2 species. The ICON and BCON processors contain routines for converting from RADM2 to CB05. If desired, these routines can be overridden by supplying a MECH_CONV_FILE at run time.

A detailed description of the MECH_CONV_FILE format is provided in Table 6-6. The file is list-formatted.

Table 6-6. MECH_CONV_FILE format description

Line	Column	Name	Type	Description
1	1-16	SPP1_OUT	String	Output mechanism species name for first species (required)
	17	"="	Char	"=" delimits the left-hand and right-hand side of the conversion equation (required)
	18	SPP1_IN	String	Input mechanism species name for first species; coefficients on right-hand side of equation may be either integer, real, or exponential format (required)
1	n	...	...	...
1	n+1	";"	Char	";" conversion rule terminator; allows rules to span multiple lines (required)
2	1-16	SPP2_OUT	String	Output mechanism species name for second species (required)
	17	"="	Char	"=" delimits the left-hand and right-hand side of the conversion equation (required)
	18	SPP2_IN	String	Input mechanism species name for second species; coefficients on right-hand side of equation may be either integer, real, or exponential format (required)
2	n	...	...	...
2	n+1	";"	Char	";" conversion rule terminator; allows rules to span multiple lines (required)
...	...	...	...	...
X	1-16	SPPX_OUT	String	Output mechanism species name for Xth species (required)
	17	"="	Char	"=" delimits the left-hand and right-hand side of the conversion equation (required)
	18	SPPX_IN	String	Input mechanism species name for Xth species; coefficients on right-hand side of equation may be either integer, real, or exponential format (required)
X	n	....	...	...
X	n+1	";"	Char	";" conversion rule terminator; allows rules to span multiple lines (required)
X+1	1	"END;"	String	"END;" file terminator (required)

A sample of the entries in a MECH_CONV_FILE file is shown below.

```
NO2 = NO2;
  FORM = HCHO + 1.0 * GLY;
  END;
```

6.1.5 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.

6.1.6 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 a header that describes the applicable photolysis reaction. This header includes (1) the reaction name/ID; (2) comment lines describing the reaction, the stoichiometry, and the data source (note that comment lines are preceded with a "!"); (3) the location on the wavelength interval that the data represent (beginning, centered, ending, or point); and (4) a multiplier (FAC) that is applied to the photolysis rate calculation. The data section of the CSQY file lists the wavelength of the incoming solar radiation (nm), the absorption cross section (cm), and the quantum yield as columns, with each row corresponding to a specific wavelength interval. 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 6-7.

Table 6-7. CSQY format description

Line	Column	Name	Type	Description
1	A	Reaction ID	String	Text name identifying the CSQY data this name is cross-referenced in the chemical mechanism description and INCLUDE files (required)
2	A	Comments	String	Preceded by "!", comment lines describe the reaction, list the stoichiometry, 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 is listed in real or exponential format (required)
n+3	A	Wavelength	Int or Real	Wavelength corresponding to CSQY data; units = nm (required)
	B	Absorption Cross-Section	Real or Exp	Measurement of the cross-section of a molecule's spherical receiving surface for actinic flux; units = cm^2 $molecule^{-1}$ (required)
	C	Quantum Yield	Real	Ratio of the number of molecules reacting via a specific pathway to the number of molecules absorbing photons in that wavelength interval; units = molecules $photon^{-1}$ (required)
n+4	A	Wavelength	Int	Wavelength corresponding to CSQY data; units = nm (required)
	B	Absorption Cross-Section	Real or Exp	Measurement of the cross-section of a molecule's spherical receiving surface for actinic flux; units = cm^2 $molecule^{-1}$ (required)
	C	Quantum Yield	Real	Ratio of the number of molecules reacting via a specific pathway to the number of molecules absorbing photons in that wavelength interval; units = molecules $photon^{-1}$ (required)
n+X	...	...	...	...

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

```
ALD_CBIV88
! Acetaldehyde Photolysis (ALD)
! CH3CHO + hv (+2O2)-> CH3OO + HO2 + CO
! Taken from Gery et al. (1988); CSQY from Baulch et al. 5 (1984).
! format: wl, abs_cs, qy
Centered
! With FAC, units are (cm^2/molecule)
FAC=1.0E-20
280 4.50 0.580
281 4.54 0.575
282 4.58 0.570
```

6.1.7 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 (Section 6.1.6). 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 6-8.

Table 6-8 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)
n+X	...	...	...	...

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_irrad
Beginning
! With FAC, units are (photons cm-2 s-1)
FAC=1.0
```

```
185.185 3.620E+11
186.916 4.730E+11
```

6.1.8 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 O_3 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 (19×3×4) data blocks, with each block representing one of the three variables (O_3, 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 (O_3, air temperature, air density), latitude, and altitude. For example, the first block in the PROFILES file contains spring O_3 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 O_3 concentration profiles but at the latitude band ranging from 80°N to 70°N. The next 17 data blocks complete the spring O_3 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 19×3 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. A detailed description of the file format is provided in Table 6-10.

Table 6-10. 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)

	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)
	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)

6.1.9 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 6-11. The files are fixed format.

Table 6-11 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)
...	...	...	...	...

6.1.10 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 O_2 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 Section 6.1.6.

6.1.11 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 O_3 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 Section 6.1.6.

6.1.12 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 at http://www.mmm.ucar.edu/mm5. For information about the format of the WRF output files, visit the WRF-ARW homepage at http://www.mmm.ucar.edu/wrf/users.

6.1.13 InTerFile: MM5 terrain file

Used by: MCIP

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

6.1.14 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 6-1 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 6-2 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.

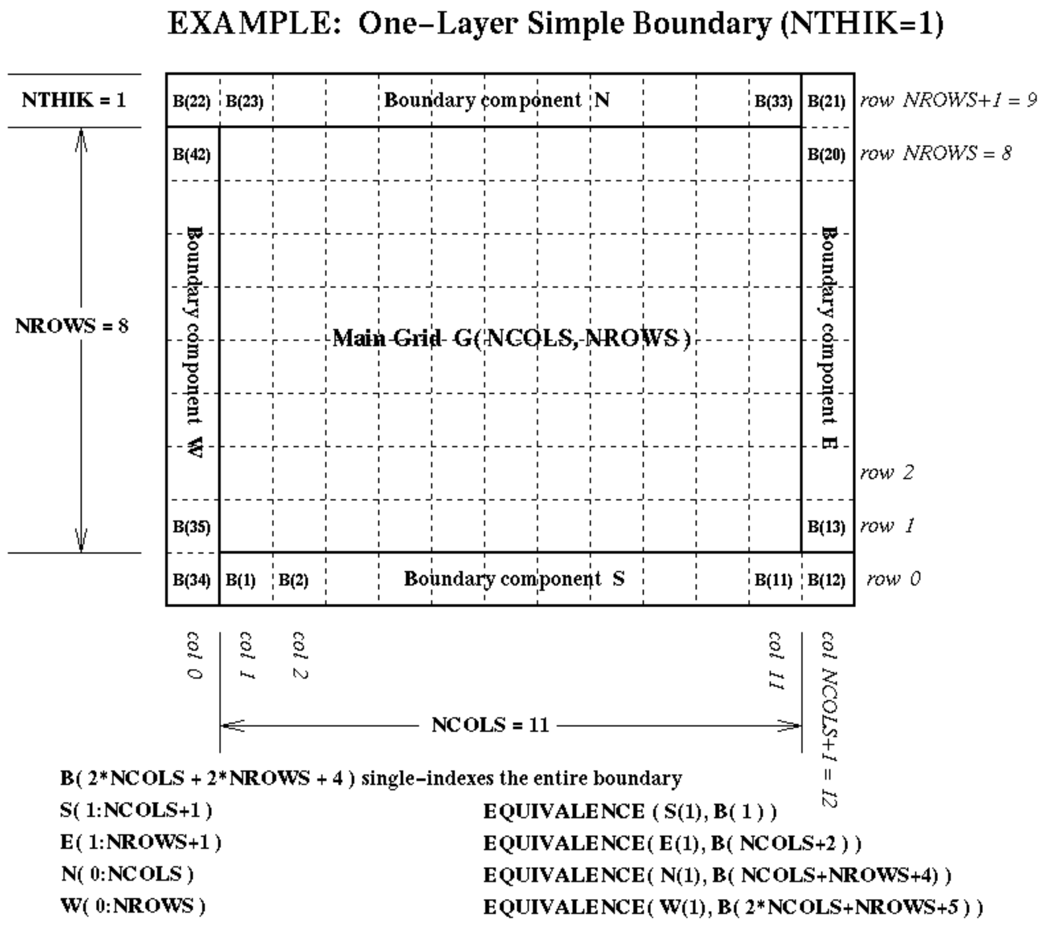


Figure 6-1. Illustration of CMAQ boundary condition file

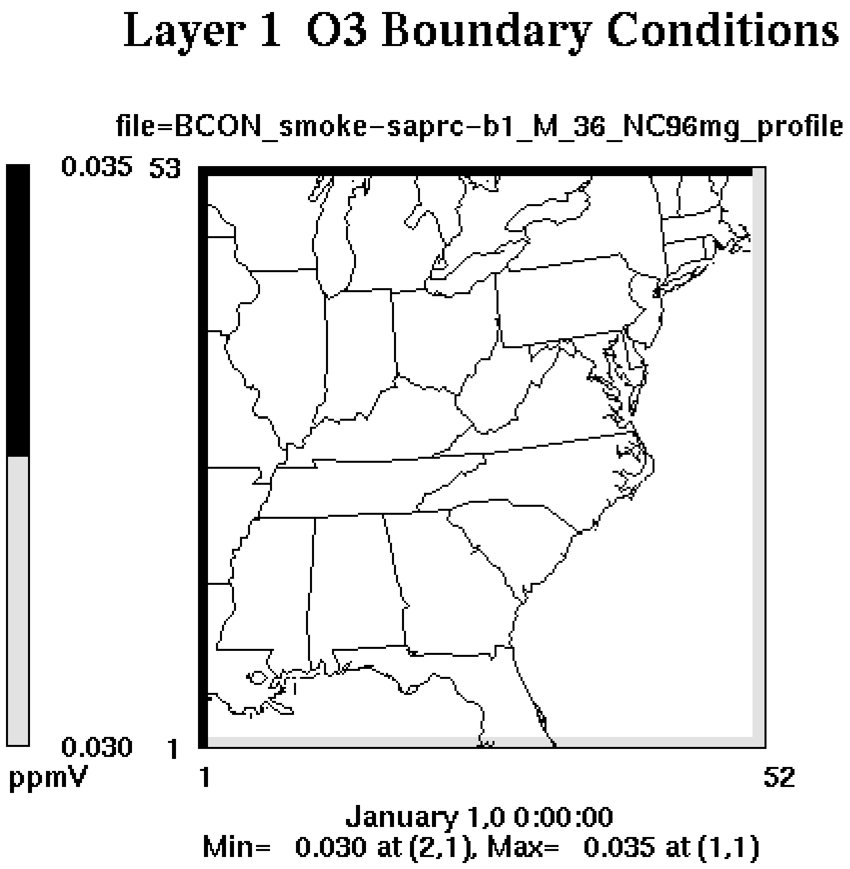

Layer 1 O3 Boundary Conditions
file=BCON_smoke-saprc-b1_M_36_NC96mg_profile
0.035
53
0.030
1
ppmV
1
52
January 1,0 0:00:00
Min= 0.030 at (2,1), Max= 0.035 at (1,1)

Figure 6-2. Graphical example of a CMAQ gridded boundary conditions file

6.1.15 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 6-3 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).

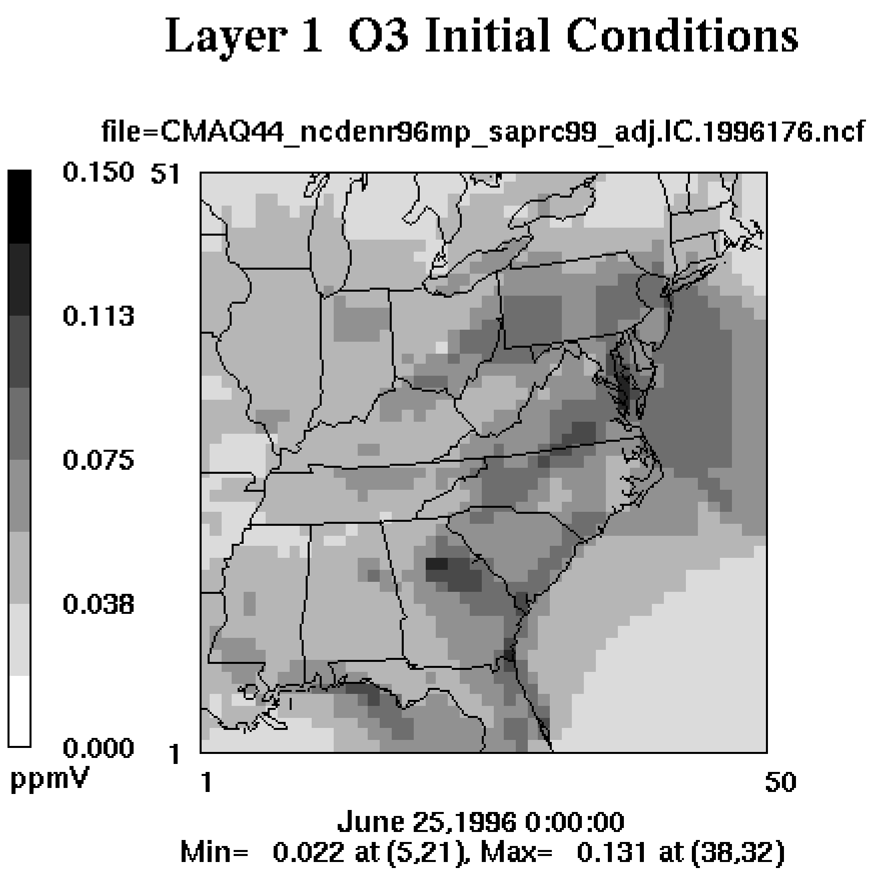

Layer 1 O3 Initial Conditions
file=CMAQ44_ncdenr96mp_saprc99_adj.IC.1996176.ncf
0.150
0.113
0.075
0.038
0.000
ppmV
51
1
1
50
June 25,1996 0:00:00
Min= 0.022 at (5,21), Max= 0.131 at (38,32)

Figure 6-3. Graphical example of a CMAQ gridded initial conditions file

6.1.16 JTABLE: Photolysis rates look-up table

Used by: CCTM

Each of the gas-phase mechanisms in CMAQ (CB05 and SAPRC-99) 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 6-12. The files are list-formatted.

Table 6-12. 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)
	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)
...	...	...	...	...
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
```

6.1.17 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 i   in the CMAQv4.7 release notes.

6.1.18 OCEAN_1: Sea salt mask

Used by: CCTM

The CMAQ aerosol model AERO5 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, it will expect the OCEAN_1 file as input.

6.1.19 GSPRO: Speciation profiles

Used by: CCTM

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 (http://www.smoke-model.org/version2.5/html/ch08s05s02.html).

6.1.20 B3GRD: Gridded, normalized biogenic emissions

Used by: CCTM

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 (http://www.smoke-model.org/version2.5/html/ch06s11.html).

6.1.21 BIOSEASON: Freeze dates

Used by: CCTM

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 (http://www.smoke-model.org/version2.5/html/ch05s11.html).

6.1.22 STK_GRPS_##: Stack groups

Used by: CCTM – inline 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 (http://www.smoke-model.org/version2.5/html/ch06s03.html).

6.1.23 STK_EMIS_##: Point source emissions

Used by: CCTM – inline 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 (http://www.smoke-model.org/version2.5/html/ch06s03.html).

6.1.24 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 (m^2 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)

6.1.25 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 (m^2 m^{-2})

6.1.26 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 [ρ 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})

6.1.27 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 (m^3 m^{-3})
SOIM2:	volumetric soil moisture in top m (m^3 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 SO_2 (m s^{-1})
VD_SULF:	deposition velocities for SO_4 (m s^{-1})
VD_NO2:	deposition velocities for NO_2 (m s^{-1})
VD_NO:	deposition velocities for NO (m s^{-1})
VD_O3:	deposition velocities for O_3 (m s^{-1})
VD_HNO3:	deposition velocities for HNO_3 (m s^{-1})
VD_H2O2:	deposition velocities for H_2O_2 (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 NH_3 (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 N_2O_5 (m s^{-1})
VD_NO3:	deposition velocities for NO_3 (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})

6.1.28 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.

6.1.29 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]

6.2 Basic 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 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 UNIXsyntax.

6.2.1 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 BCON simulation, use the following command:

```
run.bcon >& bcon_e1a.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 UNIXredirect command (“>”), as shown in the example above.

6.2.2 CONC: 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 species concentration INCLUDE files (CONC.EXT) within the mechanism INCLUDE directories list the species that are written to the CONC files for each mechanism configuration. The GC_CONC.EXT file lists the gas-phase species, the AE_CONC.EXT file lists the aerosol species, and the NR_CONC lists the nonreactive (inert) species written to the CONC file. Species can be removed from the CONC.EXT files to reduce the number of species that are written to, and thus the size of, the CONC file.

6.2.3 CGRID: 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.

6.2.4 ACONC: CCTM hourly average concentration file

The 2-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.

6.2.5 DRYDEP: 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 INCLUDE files within the mechanism INCLUDE directories. Dry deposition INCLUDE files exist for gas-phase species (GC_DDEP.EXT), aerosol species (AE_DDEP.EXT), and inert model species (NR_DDEP.EXT). Species can be removed from the DDEP.EXT files to adjust the number of species that undergo the dry deposition process and are written to the DRYDEP output file.

6.2.6 WETDEP: 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 INCLUDE files within the mechanism INCLUDE directories. Wet deposition INCLUDE files exist for gas-phase species (GC_WDEP.EXT), aerosol species (AE_WDEP.EXT), and inert model species (NR_WDEP.EXT). Species can be removed from the WDEP.EXT files to adjust the number of species that undergo the wet deposition process and are written to the WETDEP output file.

6.2.7 AEROVIS: CCTM hourly instantaneous visibility metrics

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

6.3 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.

6.3.1 AERODIAM: Instantaneous hourly aerosol diameter file

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

6.3.2 WETDEP2: CCTM cloud diagnostics file

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 INCLUDE files within the mechanism INCLUDE directories. Wet deposition INCLUDE files exist for gas-phase species (GC_WDEP.EXT), aerosol species (AE_WDEP.EXT), and inert model species (NR_WDEP.EXT). Species can be removed from the WDEP.EXT files to adjust the number of species that undergo the wet deposition process. These extra species are written to the WETDEP2 output file.

6.3.3 SSEMIS: 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.

6.3.4 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.

6.3.5 RJ: 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.

6.3.6 PA: Process analysis output – integrated process rate file

The 3-D CCTM integrated process rate file (PA) contains hourly concentrations of selected model output species in terms of the model process that contributed to the concentration in each grid cell at each hour. For each grid cell in the process analysis domain (which is most likely a subset of the full modeling domain), the PA file shows the hourly change in species concentration that is due to the major model processes, such as horizontal and vertical advection, chemistry, and wet deposition. The process analysis preprocessor, PROCAN (Section 2.2.6), is used to select the process analysis domain, the model species for which to capture process analysis information, and the model processes to track during the process analysis.

6.3.7 IRR: 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 process analysis preprocessor, PROCAN (Section 2.2.6), 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.

7. DEFINING GRIDS, LAYERS, DOMAINS, AND CHEMISTRY

CMAQ is a three-dimensional Eulerian air quality model. In this type of model, the extent of the area of interest (or domain) and the characteristics of the three-dimensional computational grid structure that represent this domain must be specified. The domain is divided into individual three-dimensional grid cells. A horizontal grid specification sets the *x* and *y* dimensions of each grid cell for the entire domain. All grid cells in a specified CMAQ domain have the same horizontal resolution. The vertical resolution of each grid cell depends on the vertical layer specification; the physical vertical extent of individual grid cells can vary in space and time. Mathematical algorithms describing atmospheric transport govern the flow of material into and out of individual grid cells. Mathematical algorithms describing chemical reactions and aerosol dynamics govern the production and loss of material contained in the grid cells.

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.

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 8 to learn how to develop new model executables for running a CMAQ simulation.

7.1 Supported CMAQ Coordinate Systems

The choice of horizontal coordinate system, or map projection, for CMAQ is governed by the input meteorological model fields and stored in the input file headers by MCIP. The grid resolution and the maximum horizontal and vertical extent are determined by the meteorological configuration that is input to MCIP. MM5 and WRF-ARW support the Lambert conformal, polar stereographic, and Mercator projections, which can be directly passed to CMAQ.

Generally, the same meteorological model fields that are used to drive CMAQ are also used to generate emissions for CMAQ.

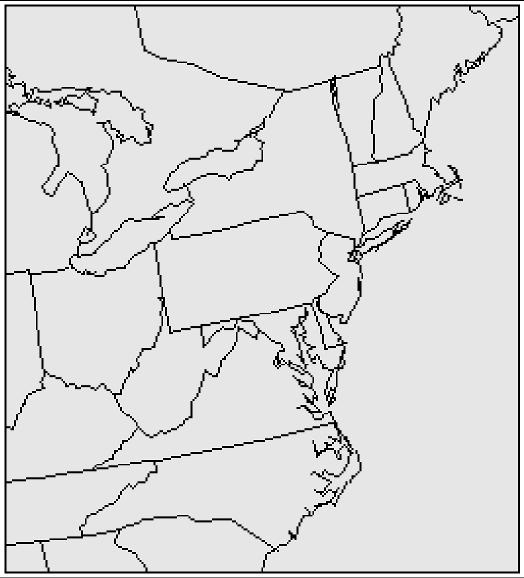
7.2 Horizontal Grids

The extent of the horizontal grid used in CMAQ is limited by the size of the domain of the input meteorology. MCIP, CCTM, and the I/O API utilities are all capable of windowing subsets of meteorology data. Choosing the appropriate horizontal grid scale and extent for the CCTM simulation is largely dependent on the issues to be addressed through the modeling. However, practical consideration should also be paid to the relationship between grid size, output file size, and execution times.

Prior to CMAQ version 4.3, Fortran INCLUDE files defined the horizontal configuration. The COORD.EXT file contained data statements for defining the horizontal coordinate and grid to be used for CMAQ. CMAQ used these INCLUDE files at compilation to create executables that were hard-wired to a specific grid configuration. Starting with CMAQ version 4.3, CMAQ developers transitioned to a dynamic horizontal grid configuration that could be defined at execution, forgoing the need to recompile the program each time a user wanted to simulate a new grid. Horizontal grids are now configured in CMAQ through the GRIDDESC file (Section 6.1.1). CMAQ horizontal grids are selected in the "horizontal grid definition" section of the CMAQ run scripts 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 can be selected either from predefined grids contained in the distributed GRIDDESC file or by adding/modifying horizontal grids in the GRIDDESC file.

7.2.1 Using predefined horizontal grids

CMAQv4.7 is distributed with a GRIDDESC file that contains a definition for a grid covering the northeastern U.S. named M_36_2001 that uses a coordinate definition named RPO_GRID. A picture of the grid and the grid definition, in the GRIDDESC format, is shown in Figure 7-1.



```
' '

'RPO_GRID'

2 33.00 45.00 -97.00 -97.00 40.00

' '

'M_36_2001'

'RPO_GRID' 936000.00 -576000.000
36000.00 36000.00 40 44 1

' '
```

Figure 7-1. CMAQ tutorial grid

7.2.2 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 file format documentation in Chapter 6 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. MM5 or WRF-ARW outputs can be run through MCIP to generate a GRIDDESC file that can be input directly to both CMAQ and SMOKE. 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.

7.2.3 Further information on horizontal grids

Meteorological Grids

- The lateral boundaries of the meteorological model fields (~5 cells on each horizontal side) are typically not used in the air quality simulation. Therefore, the meteorological model domain is often oversized to cover an area that is somewhat larger than is used in CMAQ .
- Horizontal grid spacing for the meteorology nested grids often has a 3:1 ratio, although other ratios have been employed.

C MAQ Grids

- A CMAQ grid uses the same horizontal resolution as its parent meteorology grid.
- Minimum grid dimensions of 30 rows and 30 columns are recommended.
- 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.

7.3 Vertical Layers

The vertical structure of CMAQ is inherited from the model used to prepare the meteorological information. Both MM5 and WRF-ARW use 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 MM5 and WRF-ARW are nonhydrostatic models, the vertical coordinate is time varying.

7.3.1 Vertical layer resolution

Resolving the surface boundary layer requires high resolution near the surface for meteorological simulations. To determine mass exchange between the boundary layer and free troposphere, good resolution near the boundary layer top is 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.

7.3.2 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.

7.4 Chemical Mechanism

The CMAQ modeling system accounts for chemistry in three phases: a gas phase, aerosols (solid or liquid), and an aqueous phase. The CMAQ modeling system's existing modules for gas-phase chemistry are the 2005 update to the Carbon Bond mechanism (CB05), and the Statewide Air Pollution Research Center-99 (SAPRC-99) gas-phase mechanism. 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. This procedure is described in Sections 5.4 and 7.4.2.

Gas-phase chemical mechanisms are defined in CMAQ as a series of Fortran INCLUDE files. Located in subdirectories of the `$M3MODEL/include/release` directory (each corresponding to a mechanism name), these INCLUDE files define the species, source, reaction parameters, and atmospheric processes (e.g., diffusion, deposition, advection) of the various mechanism species. 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.

7.4.1 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

`$M3MODEL/include/release`. Table 7-1 lists the available chemical mechanisms in CMAQv4.7 and what is included with each mechanism. Set the *Mechanism* variable in the CMAQ build script to the Mechanism ID in Table 7-1 to select a particular mechanism. Detailed descriptions of some of these mechanisms can be found in Byun and Ching (1999).

Table 7-1. CMAQ chemical mechanisms

Mechanism ID	cb05	saprc 99	CMAQ Aerosols		Aqueous Chemistry	Additional species
			4rd gen.	5th gen.		
cb05cl_ae4_aq	x		x		x	Cl
cb05cl_ae5_aq	x			x	x	Cl
cb05cltx_ae5_aq	x			x	x	Cl, air toxics
cb05txhg_ae5_aq	x			x	x	Hg,, air toxics, Hg
saprc99_ae4_aq		x	x		x	
saprc99_ae5_aq		x		x	x	
saprc99tx3_ae_aq		x		x		air toxics

7.4.2 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 INCLUDE files. CHEMMECH translates an ASCII mechanism listing to the INCLUDE files required by CMAQ. Like all of the CMAQ preprocessors, CHEMMECH is a Fortran program that must be compiled by the user 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 INCLUDE files. See Section 5.4 for a description of CHEMMECH.

To modify an existing mechanism, copy the mech.def file that is contained in one of the existing mechanism INCLUDE file directories to a new directory and modify the mechanism accordingly. Provide this modified mechanism definition file to CHEMMECH as input to produce the mechanism INCLUDE files needed to compile CMAQ. Put these mechanism INCLUDE files in a new directory under the `$M3MODEL/include/release` directory. 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.

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 INCLUDE files for CMAQ. Move the resulting INCLUDE files to a new directory under `$M3MODEL/include` release. 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. See Section 5.4 for additional details about the CHEMMECH program.

7.4.3 Further information on chemical mechanisms

- The versions of each chemical mechanism that include both aerosols and aqueous chemistry represent the most comprehensive modeling approach.
- 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 the default EBI solver cannot be used for the new mechanism. The Rosenbrock and SMVGEAR solvers are the only mechanism-independent choices of chemistry solvers with 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 INCLUDE files. If species are added to the domain through the emissions files, the GC_EMIS.EXT INCLUDE file must contain these new species. If the new species will also enter the domain as initial and boundary conditions, check the GC_ICBC.EXT file for the presence of these species.

8. DEVELOPING NEW CMAQ SIMULATIONS

Second-generation air quality modeling systems, such as the Regional Oxidant Model, were composed of a single air quality model and the associated input data processors. Changes in the extent of the modeling domain, its horizontal resolution, the chemistry mechanism, etc., required extensive coding changes in the air quality model and several associated input processors. Each executable program was then compiled and tested to ensure that errors were not introduced during the code modifications. Users had no control over the extent of the model domain and no flexibility to use an alternate chemistry mechanism from an existing model.

The design of the CMAQ modeling system overcomes the inflexibility of second-generation systems through the M3BLD model builder program, which builds models and processors according to user specifications. For application users of CMAQ, M3BLD 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 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 M3BLD to check out working versions 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 (see Chapter 10).

The purpose of this chapter is to demonstrate how to build the executables for the CMAQ programs beyond running the benchmark case. Before proceeding with this chapter, review Chapter 3 for an overview of the system requirements for CMAQ. In general, there are three major steps in compiling CMAQ executables:

Install the CMAQ libraries (netCDF and I/O API)

Build M3BLD

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.

8.1 General Introduction to Model Building

Before using CMAQ for operational modeling in a new computing environment, it is recommended that the model be exercised using the benchmark dataset that is distributed with the model. Chapter 3 describes how to configure a minimal hardware system for CMAQ and how to install CMAQ to run the benchmark simulation. After benchmarking CMAQ, it 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. 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 CMAQ libraries (STENEX and PARIO) and the non-CMAQ libraries (netCDF, I/O API, MPICH) can be linked to for multiple configurations and applications of the model. Likewise, the CMAQ model builder, M3BLD, can 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 programs can be reused. MCIP needs to be compiled only once on a user's system and then reused for all applications of the model, unless new source code, including libraries, becomes available.

8.2 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. The selection of episodes in CMAQ, however, is completely determined by the availability of input meteorology and emissions data. 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.

8.2.1 Defining a new horizontal grid

The grid-dependent CMAQ programs (CCTM, ICON, BCON) use the GRIDDESC grid description file (Section 6.1.1) 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 7.

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

8.2.2 Defining a new vertical layer structure

The CMAQ programs that produce 3-D output (CCTM, ICON, BCON) use the MET_CRO_3D file (Section 6.1.21) 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. See Section 5.7 for details on configuring MCIP.

8.2.3 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 generally 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, 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
- Create photolysis look-up tables (JTABLE) with JPROC for each day to be modeled
- 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

8.2.4 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; however, a module can be included in each processor to convert from the RADM2 to the CB05 or SAPRC-99 mechanisms. This option would be necessary, for example, when the CB05 mechanism is to be used in the model but the tabulated tropospheric vertical profiles that are used as inputs to ICON and BCON were constructed using RADM2 mechanism species. 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 remain time independent (i.e., the same boundary conditions are used for each day of a multiday case) for the outermost, coarse-grid 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 8-1 for an example 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.

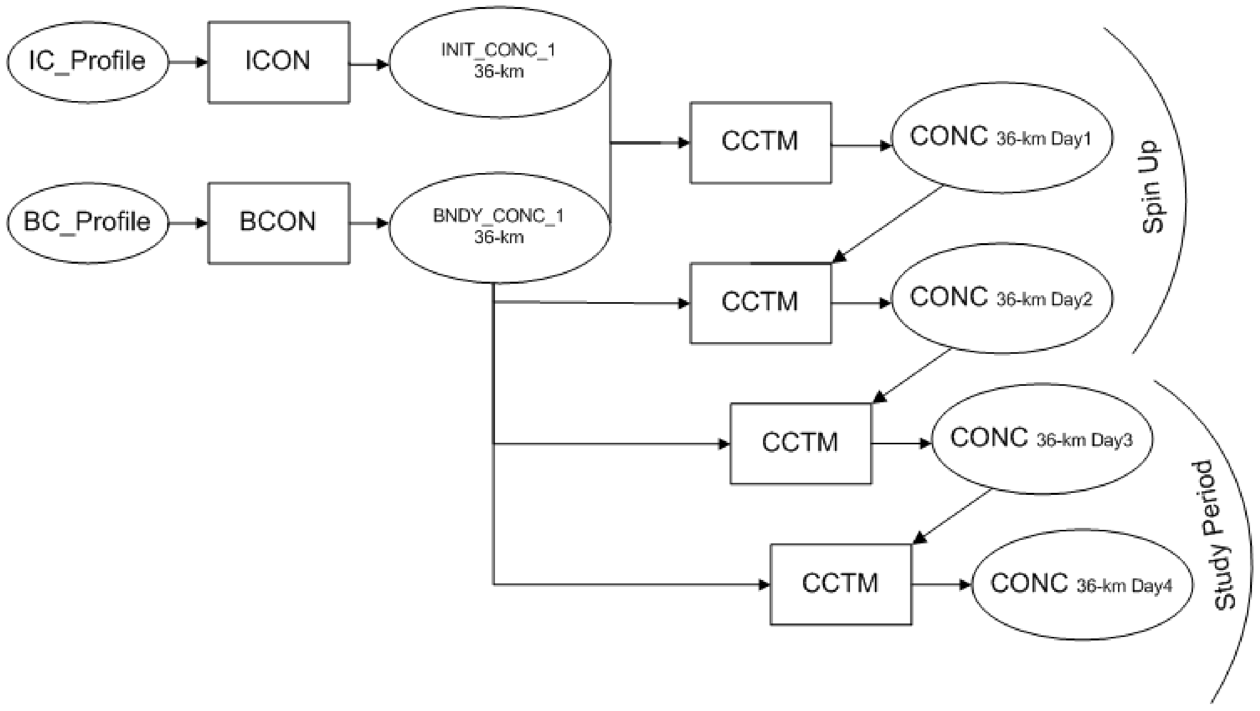


Figure 8-1. 36-km four-day modeling period IC/BC schematic

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 8-2 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.

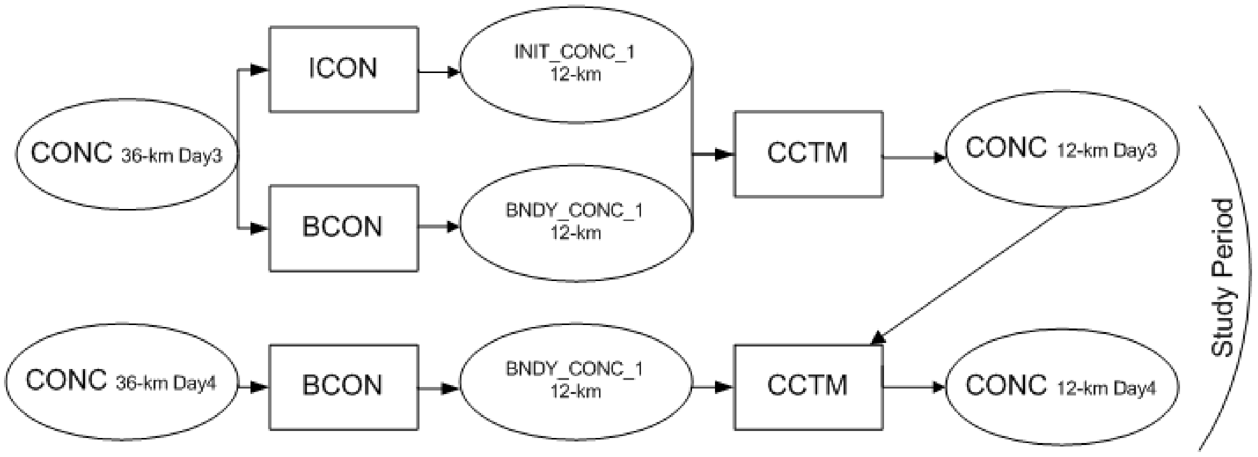


Figure 8-2. 12-km nested two-day modeling period IC/BC schematic

8.2.5 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 obvious file name for a CMAQ version 4.7 CCTM concentration file would be CCTMv47.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. Linux_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 output files that already exist. If a CMAQ program does not run successfully for this reason, the incomplete or erroneous output files must be manually deleted, or else the *DISP* variable should be invoked in the CMAQ run scripts (Chapter 5) to dispose of the output files before attempting to rerun the program. Alternatively, if the names of the output files are not correctly configured, an error will occur immediately when CMAQ is run. Check the last few lines of the standard error from the CMAQ program to determine whether the problem was due to incorrect specifications of input/output files.

8.2.6 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. JPROC science options are limited by the choice of chemical mechanism. MCIP output is affected by the source of input meteorological model output (i.e., WRF-ARW or MM5), the option to recalculate some of the meteorology variables, and the method used to compute dry deposition velocities for various species. CCTM science configuration is the most complex of all the CMAQ programs because it contains a 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 Section 5.3.2.2.

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.
- For configurations that do not use the Yamartino model driver, the Yamartino options cannot be used for the initialization module and the advection routines, and *the density-based concentration adjustment scheme must be used.*
- Ensure that there is consistency in the selection of the chemistry mechanism and the aerosol module; for example, the *aero4*aerosol module cannot be used with a chemistry mechanism that is tagged “ae5.
- Ensure that there is consistency in the selection of the chemistry mechanism with the cloud module; a chemistry mechanism that contains linkages to aqueous chemistry cannot be used when the *noop* option for the cloud module is selected.
- 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-99-based 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.

8.3 References

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

9. 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.

9.1 Source Code Management

9.1.1 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.

9.1.2 CVS explained

There are many configuration management tools, both free and commercially available. We chose The Concurrent Versions System (CVS) mainly because of its versatility. CVS controls the concurrent editing of sources by several users working on releases built from a hierarchical set of directories. CVS uses the Revision Control System (RCS) as the base system. Other reasons that CVS was an attractive choice include the following:

- It works on virtually all UNIX and Linux platforms and on many PCs.
- It is publicly available and free.

The CVS wiki states that "CVS uses a client-server architecture: a server stores the current version(s) of a project and its history, and clients connect to the server in order to 'check out' a complete copy of the project, work on this copy and then later 'check in' their changes. Typically, the client and server connect over a LAN or over the Internet, but client and server may both run on the same machine if CVS has the task of keeping track of the version history of a project with only local developers. The server software normally runs on UNIX and Linux.

"Several developers may work on the same project concurrently, each one editing files within their own 'working copy' of the project, and sending (or checking in) their modifications to the server. To avoid the possibility of people stepping on each other's toes, the server will only accept changes made to the most recent version of a file. Developers are therefore expected to keep their working copy up-to-date by incorporating other people's changes on a regular basis. This task is mostly handled automatically by the CVS client, requiring manual intervention only when a conflict arises between a checked-in modification and the yet-unchecked local version of a file." Thus, CVS adds power and features that are attractive for the CMAQ system.

9.1.3 The CVS repository

The CVS repository structure, i.e., the UNIX directory hierarchy, follows the class/module organization discussed in Young (1999). The repository is actually divided into many repositories, one for each generic model. This division makes it easier to maintain the class/module organization that is important for the model-building operation described in Chapter 8. CVS allows for the use of a "modules" file,[2] which enables a user to easily check out or extract a complete CMAQ module. For example, a user might check out a module to make code modifications. Complete modules are checked out during the CMAQ model building operation.

2 The CVS *modules* file has no intrinsic relationship with the CMAQ classes/module design implementation.

The following shows a small portion of a symbolic CVS UNIX directory tree that represents the current structure for CCTM:

```
+-> CCTM
    +-> CVSROOT  (CVS administrative files)
    +-> src
        +-> adjcon
        |   +-> adjcon_noop --> RCS files
        |   +-> denrate --> RCS files
        +-> aero
        |   +-> aero_noop --> RCS files
        |   +-> aero4 --> RCS files
        |   +-> aero5 --> RCS files
        |   +-> aero5_txhg --> RCS files
        +-> aero_depv
        |   +-> aero_depv_noop --> RCS files
        |   +-> aero_depv2 --> RCS files
        +-> chem
        |   +-> chem_noop --> RCS files
        |   +-> smvgear --> RCS files
        |   +-> ros3 --> RCS files
        |   +-> ebi_cb05cltx_ae5 --> RCS files
        |   +-> ebi_cb05cltxhg_ae5 --> RCS files
        |   +-> ebi_saprc99 --> RCS files
        |   +-> ebi_saprc99 --> RCS files
```

The symbolic tree is shown relative to the subdirectory in the repository named for the CCTM model. Similar trees exist for each of the generic models. The RCS files are the revision control history files that contain the change tables to reconstruct the actual source code according to a specific revision identifier. The tree closely follows the organization of classes and modules for CCTM and contains alternate modules within the classes. In particular, most classes contain a "no-operation" (noop) module that allows a user to essentially turn off that particular science process modeling. This is useful, for example, in debugging, where rapid turnaround is important, and a computationally demanding module that is not needed can be bypassed.

9.2 Guidelines for Developing New CMAQ Source Code

9.2.1 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.

9.2.2 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, M3BLD, 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 M3BLD to extract the codes for that module from the model code repository for compilation.

9.2.3 Thin Interface

As mentioned in Section 9.2.2, 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.

9.2.4 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.

9.2.5 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.

9.2.6 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

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

( 1)  C-----------------------------------------------------------------------
( 1)  C Function:
( 1)  C Preconditions:
( 1)  C Subroutines and Functions Called:
( 1)  C Revision History:
( 1)  C References:
      C-----------------------------------------------------------------------

( 2)         USE AERO_EMIS           ! inherits GRID_CONF

( 2)         USE SUBST_MODULES              ! stenex
      !      USE SUBST_GLOBAL_SUM_MODULE    ! stenex

( 3)         IMPLICIT NONE

      !      INCLUDE SUBST_HGRD_ID   ! horizontal dimensioning parameters
      !      INCLUDE SUBST_VGRD_ID   ! vertical dimensioning parameters
( 4)         INCLUDE SUBST_RXCMMN    ! model mechanism name

( 4)         INCLUDE SUBST_GC_SPC    ! gas chemistry species table
( 4)         INCLUDE SUBST_GC_EMIS   ! gas chem emis surrogate names and map table
( 4)         INCLUDE SUBST_GC_DEPV   ! gas chem dep vel surrogate names and map table
( 4)         INCLUDE SUBST_GC_DDEP   ! gas chem dry dep species and map table
             INCLUDE SUBST_GC_DIFF   ! gas chem diffusion species and map table

( 4)         INCLUDE SUBST_AE_SPC    ! aerosol species table
      !      INCLUDE SUBST_AE_EMIS   ! aerosol emis surrogate names and map table
( 4)         INCLUDE SUBST_AE_DEPV   ! aerosol dep vel surrogate names and map table
( 4)         INCLUDE SUBST_AE_DDEP   ! aerosol dry dep species and map table
( 4)         INCLUDE SUBST_AE_DIFF   ! aerosol diffusion species and map table

( 4)         INCLUDE SUBST_NR_SPC    ! non-reactive species table
( 4)         INCLUDE SUBST_NR_EMIS   ! non-react emis surrogate names and map table
( 4)         INCLUDE SUBST_NR_DEPV   ! non-react dep vel surrogate names and map table
( 4)         INCLUDE SUBST_NR_DDEP   ! non-react dry dep species and map table
( 4)         INCLUDE SUBST_NR_DIFF   ! non-react diffusion species and map table

( 4)         INCLUDE SUBST_TR_SPC    ! tracer species table
( 4)         INCLUDE SUBST_TR_EMIS   ! tracer emis surrogate names and map table
( 4)         INCLUDE SUBST_TR_DEPV   ! tracer dep vel surrogate names and map table
( 4)         INCLUDE SUBST_TR_DDEP   ! tracer dry dep species and map table
( 4)         INCLUDE SUBST_TR_DIFF   ! tracer diffusion species and map table

      !      INCLUDE SUBST_EMLYRS_ID ! emissions layers parameter
( 5)  #ifdef emis_chem
( 5)        INCLUDE SUBST_EMPR_CH   ! emissions processing in chem
( 5)  #else
( 5)        INCLUDE SUBST_EMPR_VD   ! emissions processing in vdif
( 5)  #endif

( 6)        INCLUDE SUBST_PACTL_ID  ! PA control parameters
( 6)        INCLUDE SUBST_CONST     ! constants
( 6)        INCLUDE SUBST_FILES_ID  ! file name parameters
( 6)        INCLUDE SUBST_IOPARMS   ! I/O parameters definitions
      #include      SUBST_IODECL    # I/O definitions and declarations
     !       INCLUDE SUBST_COORD_ID ! coordinate and domain definitions (req IOPARMS)

( 7)        CHARACTER( 120 ) :: XMSG = ' '

( 8)  C Arguments:

      !      REAL         CGRID( NCOLS,NROWS,NLAYS,* )  !  concentrations
      !      REAL      :: CGRID( :,:,:,: )              !  concentrations
( 8)        REAL, POINTER :: CGRID( :,:,:,: )              !  concentrations
( 8)        INTEGER      JDATE        ! current model date, coded YYYYDDD
( 8)        INTEGER      JTIME        ! current model time, coded HHMMSS
( 8)        INTEGER      TSTEP( 2 )   ! time step vector (HHMMSS)
                                 ! TSTEP(1) = local output step
                                 ! TSTEP(2) = sciproc sync. step (chem)

( 9)  C Parameters:

( 9)  C explicit, THETA = 0, implicit, THETA = 1
( 9)        REAL, PARAMETER :: THETA = 0.5,  ! Semi-implicit (Crank-Nicolson)
( 9)     &                  THBAR = 1.0 - THETA
( 9)        REAL THRAT  ! THBAR/THETA
( 9)        INTEGER, PARAMETER :: N_SPC_DDEP = N_GC_DDEP
( 9)     &                                 + N_AE_DDEP
( 9)     &                                 + N_NR_DDEP
( 9)     &                                 + N_TR_DDEP
( 9)        <  >

( 9)  C number of species on the PM emissions input file. Set in OPEMIS
( 9)  C the value changes with the type of emissions file.
( 9)        INTEGER, SAVE :: NAESPCEMIS

( 9)        REAL, PARAMETER :: M2PHA = 1.0E+04       ! 1 hectare = 1.0e4 m**2
( 9)        REAL, PARAMETER :: CMLMR = 1.0E+06       ! ppmV/Molar Mixing Ratio
( 9)        REAL, PARAMETER :: CNVTD = M2PHA / CMLMR / MWAIR ! combined ddep
                                                    ! conversion factor
( 9)        REAL, PARAMETER :: GPKG = 1.0E+03        ! g/Kg
( 9)        REAL, PARAMETER :: MGPG = 1.0E+06        ! micro-g/g

(10)  C External Functions not previously declared in IODECL3.EXT:

(10)        INTEGER, EXTERNAL :: SECSDIFF, SEC2TIME, TIME2SEC
(10)        LOGICAL, EXTERNAL :: ENVYN

(11)  C File variables:
(11)        <  >

(12)  C Local Variables:

(12)        CHARACTER( 16 ), SAVE :: PNAME = 'VDIFFIM'
(12)        <  >
(12)        REAL, ALLOCATABLE, SAVE :: VDEMIS( :,:,:,: ) ! total emissions array
(12)        <  >
```

```
(13)        INTERFACE
(13)           SUBROUTINE RDMET( MDATE, MTIME, RDEPVHT, RJACM, RVJACMF, RRHOJ, DENS1 )
(13)              IMPLICIT NONE
(13)              INTEGER, INTENT( IN )       :: MDATE, MTIME
(13)              REAL, INTENT( OUT )         :: RDEPVHT( :,: )
(13)              REAL, INTENT( OUT )         :: RJACM  ( :,:,: )
(13)              REAL, INTENT( OUT )         :: RVJACMF( :,:,: )
(13)              REAL, INTENT( OUT )         :: RRHOJ  ( :,:,: )
(13)              REAL, INTENT( OUT )         :: DENS1  ( :,: )
(13)           END SUBROUTINE RDMET
(13)           SUBROUTINE RDDEPV ( MDATE, MTIME, MSTEP, CGRID, DEPV )
(13)              IMPLICIT NONE
(13)              INTEGER, INTENT( IN )       :: MDATE, MTIME, MSTEP
(13)              REAL, POINTER               :: CGRID( :,:,:,: )
(13)              REAL, INTENT( OUT )         :: DEPV( :,:,: )
(13)           END SUBROUTINE RDDEPV
(13)        <  >
      END INTERFACE

     C-----------------------------------------------------------------------

(14)        IF ( FIRSTIME ) THEN
(14)           FIRSTIME = .FALSE.
(14)           LOGDEV = INIT3()

(14) C for emissions (from COORD.EXT) .......................................

(14)           IF ( GDTYP_GD .EQ. LATGRD3 ) THEN
(14)               DX1 = DG2M * XCELL_GD ! in m.
(14)               DX2 = DG2M * YCELL_GD
(14)      &           * COS( PI180*( YORIG_GD + YCELL_GD * FLOAT( GL_NROWS/2 ))) ! in m.
(14)               ELSE
(14)               DX1 = XCELL_GD        ! in m.
(14)               DX2 = YCELL_GD        ! in m.
(14)               END IF

(14) C create global maps

(14)           CALL VDIFF_MAP ( DF2EM, DF2DV, DD2DV, DEPV_MAP, DIFF_MAP, DDEP_SPC,
(14)      &                     DV2DF )

(14) C set vertical layer definitions from COORD.EXT

(15)           ALLOCATE ( RDX3F( NLAYS ), STAT = ALLOCSTAT )
(15)           ALLOCATE ( RDX3M( NLAYS ), STAT = ALLOCSTAT )
(15)           IF ( ALLOCSTAT .NE. 0 ) THEN

(15)               XMSG = 'Failure allocating RDX3F or RDX3M'
(15)               CALL M3EXIT( PNAME, JDATE, JTIME, XMSG, XSTAT1 )
(15)               END IF

(14)       < other calculations that need to be performed only the first time >

               END IF          !  if Firstime

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

     C read & interpolate met data
```

```
(17)        CALL RDMET ( MDATE, MTIME, RDEPVHT, RJACM, RVJACMF, RRHOJ, DENS1 )

     C read & interpolate deposition velocities

           < perform other operations >

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

(19)        CALL EDYINTB ( EDDYV, DT, JDATE, JTIME, TSTEP( 2 ) )

            < Perform other operations to set up for tridiagonal solver >

(20)        DO 345 R = 1, MY_NROWS
(20)           DO 344 C = 1, MY_NCOLS
            <  Perform operations >

(21)               DO 301 N = 1, NSTEPS( C,R )

            < Perform operations >

(21) 301            CONTINUE      !  end time steps loop

            < Update concentration and deposition arrays >

(20) 344         CONTINUE         !  end loop on col C
(20) 345      CONTINUE            !  end loop on row R
      < Perform other operations >

     C If last call this hour:  write accumulated depositions:

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

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

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

(22)               END DO
```

```
(18)                        EMIS_PA( C,R,L,S ) = VDEMIS( S,L,C,R )
(18)                        END DO
(18)                  END DO
(18)                     END DO
(18)            END DO
(18)              CALL PA_UPDATE_EMIS ( 'VDIF', EMIS_PA, JDATE, JTIME, TSTEP )
(18)              END IF

(19)           CALL EDYINTB ( EDDYV, DT, JDATE, JTIME, TSTEP( 2 ) )

               < Perform other operations to set up for tridiagonal solver >

(20)           DO 345 R = 1, MY_NROWS
(20)              DO 344 C = 1, MY_NCOLS
               <  Perform operations >

(21)                  DO 301 N = 1, NSTEPS( C,R )

               < Perform operations >

(21) 301               CONTINUE      !  end time steps loop

               < Update concentration and deposition arrays >

(20) 344            CONTINUE         !  end loop on col C
(20) 345         CONTINUE            !  end loop on row R
               < Perform other operations >
     C If last call this hour:  write accumulated depositions:

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

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

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

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

(18)              IF ( LIPR ) THEN
     !              DO V = 1, N_SPC_DDEP
(18)                  DO V = 1, N_SPC_DEPV
(18)                      DO R = 1, MY_NROWS
(18)                         DO C = 1, MY_NCOLS
(18)                            DDEP_PA( C,R,V ) = DDEP( V,C,R )
(18)                            END DO
(18)                         END DO
```

```
(18)                  END DO
(18)               CALL PA_UPDATE_DDEP ( 'VDIF', DDEP_PA, JDATE, JTIME, TSTEP )
(18)               END IF

      C re-set dry deposition array to zero

             DDEP = 0.0

             END IF

(23)      RETURN
(23)      END
```

Footnotes:

(1) Header comments - Highly recommended for internal documentation.
(2) USE <module name> 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

9.3 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, create a working directory for the model download.

- Download the appropriate tar file SCRIPTS.tar.gz from the CMAS web site (www.cmascenter.org) for the chosen platform. Users must register before proceeding with the download steps.

- Untar the file using the command:

  ```
  > tar xvfz SCRIPTS.tar.gz
  ```

  This will expand a directory labeled *scripts* that contains all the scripts necessary to compile and run CMAQ.

- Either install the CMAQ source code and libraries (Chapter 3) or create links to the CMAQ models and libraries as follows:

  ```
  > ln –s <models directory> models
  > ln –s <lib directory> lib
  ```

- In the `scripts/cctm` subdirectory, modify a file called `bldit.cctm` as follows:

  uncomment the line "`set MakeOpt`" by removing the leading '#' character.

- Execute the `bldit.cctm` script. This creates a Makefile as well as a configuration file in the subdirectory `scripts/cctm/BLD_e3a`, where the model code has been copied.

- The Makefile can be modified to compile and link the new module by specifying <full path name>.o for the object file that needs to be linked in. It is essential that a source file with the corresponding name (with extension ".F") reside in the same directory as the specified path name for the object file.

- Issue the "make" command to compile the source code into an executable.

  ```
  > make –f Makefile
  ```

9.4 Guidelines to Writing Shell Scripts for CMAQ

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

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

```
#! /bin/csh –f

# ======================= CCTMv4.7 Run Script ===================== #
# Usage: run.cctm >&! cctm_e3a.log &                                #
# The following environment variables must be set for this script to  #
# execute properly:                                                 #
#   setenv M3DATA =  input/output data directory                    #
# To report problems or request help with this script/program:      #
#             http://www.cmascenter.org/html/help.html              #
# ================================================================= #

#> Check that M3DATA is set:
 if ( ! -e $M3DATA ) then
    echo "   $M3DATA path does not exist"
    exit 1
    endif
 echo " "; echo " Input data path, M3DATA set to $M3DATA"; echo " "

 set APPL     = e3a
 set CFG      = e3a
#set CFG      = $APPL
 set EXEC     = CCTM_$CFG       # ctm version

#> horizontal domain decomposition
#setenv NPCOL_NPROW "1 1"; set NPROCS   = 1 # single processor setting
 setenv NPCOL_NPROW "4 2"; set NPROCS   = 8

#> for Scyld Beowulf ...
#setenv NP $NPROCS
#setenv BEOWULF_JOB_MAP -1:-1:0:0:1:1:2:2:3:3:4:4
```

```
#echo " task-processor map `beomap`"

#> Set the working directory:
 set BASE     = $cwd
 cd $BASE; date; cat $BASE/cfg.$CFG; echo "    "; set echo

#> timestep run parameters

 set STDATE   = 2001203       # 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 log file [ default = unit 6 ]; uncomment to write standard output to a
log
#setenv LOGFILE $BASE/$APPL.log

#> turn off excess WRITE3 logging
 setenv IOAPI_LOG_WRITE F

#> max sync time step (sec) (default is 720)
#setenv CTM_MAXSYNC 300

#> aerosol diagnostic file [ T | Y | F | N ] (default is F|N)
#setenv CTM_AERDIAG Y

#> sea-salt emissions diagnostic file [ T | Y | F | N ] (default is F|N)
#setenv CTM_SSEMDIAG Y

#> stop on inconsistent input file [ T | Y | F | N ]
 setenv FL_ERR_STOP F

#> remove existing output files?
#set DISP = delete
#set DISP = update
 set DISP = keep

#> output files and directories
 set OUTDIR   = $M3DATA/cctm
 if ( ! -d "$OUTDIR" ) mkdir -p $OUTDIR
 set CONCfile  = $EXEC"CONC".$APPL                    # CTM_CONC_1
 set ACONCfile = $EXEC"ACONC".${APPL}                 # CTM_ACONC_1
 set CGRIDfile = $EXEC"CGRID".${APPL}                 # CTM_CGRID_1
 set DD1file   = $EXEC"DRYDEP".$APPL                  # CTM_DRY_DEP_1
 set WD1file   = $EXEC"WETDEP1".$APPL                 # CTM_WET_DEP_1
 set WD2file   = $EXEC"WETDEP2".$APPL                 # CTM_WET_DEP_2
 set SS1file   = $EXEC"SSEMIS1".$APPL                 # CTM_SSEMIS_1
 set AV1file   = $EXEC"AEROVIS".$APPL                 # CTM_VIS_1
 set AD1file   = $EXEC"AERODIAM".$APPL                # CTM_DIAM_1
 set PA1file   = $EXEC"PA_1".$APPL                    # CTM_IPR_1
 set PA2file   = $EXEC"PA_2".$APPL                    # CTM_IPR_2
 set PA3file   = $EXEC"PA_3".$APPL                    # CTM_IPR_3
 set IRR1file  = $EXEC"IRR_1".$APPL                   # CTM_IRR_1
 set IRR2file  = $EXEC"IRR_2".$APPL                   # CTM_IRR_2
 set IRR3file  = $EXEC"IRR_3".$APPL                   # CTM_IRR_3
 set RJ1file   = $EXEC"RJ_1".$APPL                    # CTM_RJ_1
 set RJ2file   = $EXEC"RJ_2".$APPL                    # CTM_RJ_2
```

```
#> set ancillary log file name extensions
 setenv CTM_APPL $APPL

#> set floor file (neg concs)
 setenv FLOOR_FILE $BASE/FLOOR_${APPL}

#> horizontal grid defn; check GRIDDESC file for GRID_NAME options
 setenv GRIDDESC ../GRIDDESC1
 setenv GRID_NAME M_36_2001

#> species for standard conc
#setenv CONC_SPCS "O3 NO ANO3I ANO3J NO2 FORM ISOP ANH4J ASO4I ASO4J"

#> layer range for standard conc
#setenv CONC_BLEV_ELEV " 1 4"

#> species for integral average conc
 setenv AVG_CONC_SPCS "O3 NO CO NO2 ASO4I ASO4J NH3"
#setenv AVG_CONC_SPCS "ALL"

#> layer range for integral average conc
 setenv ACONC_BLEV_ELEV " 1 1"

#> input files and directories

 set OCEANpath = $M3DATA/emis/2001
 set OCEANfile = us36_surf.40x44.ncf

 set EMISpath  = $M3DATA/emis/2001
 set EMISfile  = emis3d.20010722.US36_40X44.ncf

#set TR_EMpath =
#set TR_EMfile =

#set GC_ICpath = $OUTDIR
#set GC_ICfile = CCTM_e3aCGRID.d1b
 set GC_ICpath = $M3DATA/icon
 set GC_ICfile = ICON_cb05cl_M_36_2001_profile

 set GC_BCpath = $M3DATA/bcon
 set GC_BCfile = BCON_cb05cl_M_36_2001_profile

 set METpath   = $M3DATA/mcip3/M_36_2001
 set extn      = 010722
 set GC2file   = GRIDCRO2D_${extn}
 set GD2file   = GRIDDOT2D_${extn}
 set MC2file   = METCRO2D_${extn}
 set MD3file   = METDOT3D_${extn}
 set MC3file   = METCRO3D_${extn}
 set MB3file   = METBDY3D_${extn}

 set TR_DVpath = $METpath
 set TR_DVfile = $MC2file

#> 7-level photolysis data w/ file header
```

```
 set JVALpath  = $M3DATA/jproc
 set JVALfile  = JTABLE_${STDATE}

 set AE_ICpath = $GC_ICpath
 set NR_ICpath = $GC_ICpath
 set TR_ICpath = $GC_ICpath
 set AE_ICfile = $GC_ICfile
 set NR_ICfile = $GC_ICfile
 set TR_ICfile = $GC_ICfile

 set AE_BCpath = $GC_BCpath
 set NR_BCpath = $GC_BCpath
 set TR_BCpath = $GC_BCpath
 set AE_BCfile = $GC_BCfile
 set NR_BCfile = $GC_BCfile
 set TR_BCfile = $GC_BCfile

#> input and output files and directories (boilerplate)
 source in_out.q
 if ( $status ) exit 1

#> for the run control ...

 setenv CTM_STDATE      $STDATE
 setenv CTM_STTIME      $STTIME
 setenv CTM_RUNLEN      $NSTEPS
 setenv CTM_TSTEP       $TSTEP
 setenv CTM_PROGNAME    $EXEC

#> look for existing log files

 set test = `ls CTM_LOG_???.${APPL}`
 if ( "$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

#> - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

 env

 ls -l $BASE/$EXEC; size $BASE/$EXEC

#> Executable call for single PE, uncomment to invoke
# time  $BASE/$EXEC

#> Executable call for multiple PE, set location of MPIRUN script
 set MPIRUN = /share/linux/bin/mpich-ch_p4/bin/mpirun
 set TASKMAP = $BASE/machines8
```

```
cat $TASKMAP
time $MPIRUN -v -machinefile $TASKMAP -np $NPROCS $BASE/$EXEC

date
exit
```

9.5 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 http://www.epa.gov/asmdnerl/CMAQ/CMAQscienceDoc.html].
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 (see Section 9.2.5).
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.

9.6 References

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.

10. ANALYSES 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, the Atmospheric Model Evaluation Tool (AMET), netCDF Operators (NCO), Python-based ioapiTools, 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.

Almost all of the CMAQ input and output files use the I/O API file format, which is a modified version of the netCDF format. If the user has already built the netCDF library (http://www.unidata.ucar.edu/software/netcdf/index.html) 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.

This chapter presents a brief overview of each set of tools mentioned above. At the beginning of each subsection below is a table listing helpful information for users who wish to use the tools.

10.1 CMAQ Utility Tools

Latest Version	Version 4.7, released on 12/09/2008
Main website	http://www.cmascenter.org
Support	http://bugz.unc.edu

Several utility tools (Fortran-based) are provided along with the CMAQ code/scripts distribution. These are included in the MODELS.tar.gz file, and located in the `$M3MODEL/TOOLS` source code directory. 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 various CMAQ utility tools are described below.

a) combine

This utility combines fields from a set of I/O API input files to create an output file. The file assigned to the environment variable SPECIES_DEF defines the new species variables and how they are constructed.

b) sitecmp

This utility, “site compare,” generates a CSV (comma-separated values) file that compares CMAQ-generated concentrations with an observed dataset. The various environment variables required by sitecmp are:

- TABLE_TYPE: dataset type (AQS, IMPROVE, CASTNET, STN, NADP, MDN, SEARCH)
- M3_FILE_n: I/O API input files that contain modeled species data (maximum of 12 files)
- SITE_FILE: site file containing site ID, longitude, latitude (tab-delimited)
- IN_TABLE: observed data (comma-delimited with header)
- OUT_TABLE: output data table with columns of observed and modeled values

c) rd_airs

This utility reads the raw AQS data and writes hourly values in one-day-per-record format. The various environment variables required by rd_airs are:

- INFILE: AIRS data file downloaded from web site http://www.epa.gov/ttn/airs/airsaqs/detaildata/downloadaqsdata.htm
- SITEFILE: list of AIRS site codes with latitude and longitude values
- OUTFILE: output data file (hourly values in 24/record format)
- STATES: list of states to process (default is all states)
- YEARS: list of years to process (default is all years)
- PARAMETER: code of species to process (default is 44201 OZONE)
- CHECKUNITS: switch to check for valid units

d) airs2ext

This utility reads the output from the rd_airs program and writes hourly or daily values in the sitecmp CASTNET input format (CSV). The various environment variables required by airs2ext are:

- INFILE: data file generated with the rd_airs program
- OUTFILE: output data file used with sitecmp (hourly or daily values)
- SPECIES: name of species for header line
- STEP: time step of output (DAY or HOUR)
- START_DATE: starting date to window data (YYYYDDD format)

- END_DATE: end date of window data (YYYYDDD format)

e) cast2ext

This utility reads the CASTNET hourly values downloaded from www.epa.gov/castnet and generates an input file for the sitecmp program. The various environment variables required by cast2ext are:

- INFILE: hourly CASTNET data file
- OUTFILE: output file in format to use with sitecmp

10.2 M3tools

Main website	http://www.baronams.com/products/ioapi
Download	http://www.baronams.com/products/ioapi/ioapi-3.0.tar.gz
Answers to FAQ	http://www.baronams.com/products/ioapi/ERRORS.html
Latest User's Manual	http://www.baronams.com/products/ioapi/AA.html#tools

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 O_3 data from 1-h O_3), 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

10.3 Package for Analyses and Visualization of Environmental Data (PAVE)

Latest Version	Version 2.3 released on October 18, 2004
Main website	http://paved.sourceforge.net
Download	http://paved.sourceforge.net/#Downloads
Latest User's Manual	http://paved.sourceforge.net/pave_doc/Pave.html

Answers to FAQ	http://paved.sourceforge.net/pave_doc/Pave.FAQ.html
Support website	http://bugz.unc.edu

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).

10.4 Visualization Environment for Rich Data Interpretation (VERDI)

Latest Version	Version 1.1 released on May 22, 2009
Main website	http://www.verdi-tool.org
Download	http://www.verdi-tool.org
Latest User's Manual	http://www.verdi-tool.org
Answers to FAQ	http://www.verdi-tool.org
Support website	http://bugz.unc.edu

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.

10.5 Atmospheric Model Evaluation Tool (AMET)

Latest Version	Version 1.1 released on May 31, 2008
Main website	http://www.cmascenter.org
Download	http://www.cmascenter.org
Latest User's Manual	http://www.cmascenter.org
Training/Workshop	http://www.cmascenter.org
Answers to FAQ	N/A
Support website	http://bugz.unc.edu

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, and CMAQ, 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).

10.6 netCDF Operators (NCO)

Latest version	Version 3.9.9 released on July 24, 2009
Main website	http://nco.sourceforge.net/
Download	http://nco.sourceforge.net/#Download
Latest User's Manual	http://nco.sourceforge.net/#RTFM
Answers to FAQ	http://nco.sourceforge.net/#FAQ
Support website	http://nco.sourceforge.net/nco.html#help

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.

10.7 Python ioapiTools

Main website	http://www-pcmdi.llnl.gov/software-portal/Members/azubrow/ioapiTools/index_html
Download	http://www-pcmdi.llnl.gov/softwareportal/Members/azubrow/ioapiTools/download-source-file
Latest User's Manual	http://www-pcmdi.llnl.gov/software-portal/Members/azubrow/ioapiTools/index_html
Support info	http://www-pcmdi.llnl.gov/software-portal/Members/azubrow/ioapiTools/problems-file

The *ioapiTools* package was developed to include I/O API data within the Climate Data Analyses Tools (CDAT) framework. The *ioapiTools* module provides functions for extracting and manipulating data (individual variables) and writing to either an I/O API file or a climate and forecast compliant (CF) netCDF file with I/O API metadata.

The *ioapiTools* module is a contributed package to CDAT and is built on top of the Climate Data Management System (cdms) module. The key object in *ioapiTools*, *iovar,* is a daughter of the *cdms* transient variable. In other words, an *iovar* object has all the capabilities of a *cdms* variable plus some extra methods and attributes. So, one can use the methods and attributes that the user may already be familiar with from *cdms* variables, as well as the new methods and attributes. The user needs to install *pyIoapi*, a low-level python interface to the I/O API library, as well as *ioapiTools*, the high-level python module that integrates I/O API data into *cdms*.

After installing CDAT and the ioapiTools, the user should use a python interpreter (*python*, *idle*, *ipython*, etc.) when using the various packages.

10.8 Integrated Data Viewer (IDV)

Main website	http://www.unidata.ucar.edu/software/idv/
Download	http://www.unidata.ucar.edu/software/idv/docs/userguide/Starting.html
Latest User's Manual	http://www.unidata.ucar.edu/software/idv/docs/userguide/
Training/Workshop	http://www.unidata.ucar.edu/software/idv/docs/workshop/
Answers to FAQ	http://www.unidata.ucar.edu/software/idv/docs/userguide/Faq.html
Support info	http://www.unidata.ucar.edu/software/idv/docs/userguide/Support.html

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.

More recently, two features have been added to IDV that make it attractive for use by the air quality modeling community: (1) the capability to read I/O API netCDF formatted files, and (2) 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
```

10.9 NCAR Command Language (NCL)

Latest Version	Version 5.1.1 released on June 16, 2009
Main website	http://www.ncl.ucar.edu
Download	http://www.ncl.ucar.edu/Download/index.shtml
Latest User's Manual	http://www.ncl.ucar.edu/Document/index.shtml
Training/Workshop	http://www.ncl.ucar.edu/Training/index.shtml

Support info	http://www.ncl.ucar.edu/Support/ncl_talk.shtml

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.

11. CMAQ SUPPORT

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. The CMAS Center does not offer telephone or on-site support.

Additional CMAQ support can be obtained for a fee through support contracts with modeling contractors experienced in applying and developing CMAQ source code. Contact information for these contractors is available through the links page on the CMAS Center website.

11.1 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.

11.1.1 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
- Model 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

11.1.2 CMAS community

A primary focus of CMAS is to instill a sense of community among the users of environmental models. From the individual to the organizational level, the beneficiaries of CMAS include (but are not limited to):

- Government
- Industry
- Academia
- Research
- Consultants
- Stakeholder groups

By promoting cooperation within and between the various groups in the environmental modeling community, CMAS forms the foundation to enable the community to participate in the examination of issues and the subsequent development of strategies that meet societal challenges of environmental protection.

11.1.3 Why is CMAS needed?

Historically, advancements in air quality model technology could not easily be shared among modelers because of technical incompatibilities. By standardizing with open-source, advanced modeling systems, CMAS enables collaborative development and linking of models for meteorology, emissions, air quality, and environmental and health effects. CMAS builds on the pioneering work by EPA and other research organizations by developing advanced tools to assist users in building models, developing datasets, analyzing results, and understanding model behavior. Because the CMAS-supported models use a "modular" approach with well-defined communications between modules, developers can upgrade existing processes or add new ones, thus ensuring the rapid evolution of the technology to meet the changing needs of the environmental modeling community.

CMAQ was EPA's first tangible product to evolve out of the community modeling paradigm. With a framework for a community model in CMAQ, there is a need for centralized coordination of development and application efforts for mutual benefit to scientists, model developers, practitioners of modeling, and regulatory users of modeling results. CMAS and its accompanying center at the CEMPD facilitate the effort to draw the interest of the modeling community toward advancement through cooperation.

11.2 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.

11.2.1 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.

11.2.2 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.

11.2.3 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

11.2.4 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.

11.3 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
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

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 NO_x. 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 Models-3, 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 Models-3, 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 Models-3 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 Models-3 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 Models-3 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.

Models-3 components. The various subsystems within the Models-3 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.

Models-3. 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 Models-3 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 Models-3, speciation refers to using one of the chemical mechanisms available with Models-3 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 Models-3 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.