Final Report

Speed Improvements for EPS3

Prepared for
Jim MacKay
TCEQ
12100 Park 35 Circle
Austin, TX 78753

Prepared by
Gary Wilson
Jeremiah Johnson
ENVIRON International Corporation
773 San Marin Drive
Suite 2115
Novato, CA 94998

July 2010

TABLE OF CONTENTS

TABLES

FIGURES

APPENDIX

1.0 INTRODUCTION

EPS3 is the foundation of emissions modeling for the TCEQ. Every part of the emissions inventory, including point, mobile, non-road, area, and biogenic sources, is processed through at least part of the EPS3 system. The large meteorological and photochemical models used by the TCEQ have required the purchase of many multi-core servers. These servers also process the emissions inputs. Computers with multi-core central processing units can process data in parallel rather than sequentially. EPS3 lacks this capability. The purpose of this work order was to evaluate and implement parallel data processing algorithms into EPS3 where appropriate. In addition, other methodologies for improving the performance of the EPS3 modules were evaluated and incorporated into the system.

Although parallel processing methods proved not to be effective, other methods were utilized and ENVIRON has accomplished the goal of significantly improving the performance of the EPS3 system. The speedup in the EPS3 emissions processing will provide benefits of rapid control strategy evaluation and quicker turnaround when inventories are being developed and quality assured. Faster processing will be especially beneficial for the extremely large data sets encompassing tens of thousands of Texas point sources and thousands of roadway links in urban emissions modeling required by State Implementation Plan (SIP) modeling.

2.0 METHODS

2.1 Enhancements to PIGEMS Module

The PIGEMS module, which selects sources for treatment using Plume-in-Grid (PiG), can be a time-consuming step in processing point source emissions through EPS3. PIGEMS also prepares the CAMx-ready format point source emissions file. There are two reasons why this module is time-consuming. First it must process a tremendous amount of data, effectively the entire point source inventory. Secondly, because the point source inventory may contain day-specific emissions estimates, the PIGEMS module must be exercised for each day of the modeling episode. There is no way of getting around day-specific application of PIGEMS, however the amount of data to be processed uniquely for each day could be greatly reduced.

Currently, the input files read by PIGEMS must contain the full complement of point sources each time it is run. This is because PIGEMS must write all point sources to the final CAMx-ready point source emissions file. Consequently, PIGEMS is reading and processing many sources that do not change from one day to another or from one scenario to another, which is inefficient. A more efficient approach would be to modify PIGEMS to allow groups of sources to be processed incrementally.

The modified version of PIGEMS optionally reads a CAMx-ready point source emissions file that may or may not include sources selected for PiG. This point source file should include all sources that do not change from day-to-day (the static sources). PIGEMS then reads emissions for additional time varying point sources and performs calculations to select sources for PiG treatment. Finally, PIGEMS merges both sets of sources into a single CAMx-ready point source emissions file, creating a complete inventory file.

Point source emissions processing using the modified PIGEMS must be performed in stages. First, all sources that are unchanged in future scenarios are processed normally by PIGEMS. In most cases this will be the bulk of the point source inventory. By excluding sources with day-specific data from this first set, the processing can be done for a “representative day” further reducing the amount of processing needed. The first output file produced by PIGEMS would then be used as a “static” file in further processing. In the second stage, the user must supply to PIGEMS the static file plus additional emission files containing the sources that were excluded from the static file.

Because the bulk of the sources in the test bed provided by the TCEQ staff were processed by the static sources step, this method proved to be highly effective when processing the test bed. As shown in Table 2-1 the processing time of PIGEMS on the testbed dataset was reduced by a factor of 4. It was verified that the modified version produces an output file that is identical to the original version.

Table 2-1. Timing of the PIGEMS module.

Testbed dataset (16 simulation days)		
Version	**Total time (min)**	**Min/day**
Original	25	1.6
Modified	6	0.4

2.2 Profiling the Code Using PGPROF

PGPROF (Portland Group PROFiler) is a graphical performance profiler that calculates and displays runtime performance statistics. The program can profile parallelized MPI programs and multi-threaded OpenMP programs. PGPROF is included in all of the various PGI Fortran compiler packages available for Linux. Once the Fortran program has been compiled with the appropriate profile flags, it can be run normally on any application. At completion a text file with performance statistics is generated. The PGPROF interface then uses this text file to display the statistics data.

PGPROF was used to analyze individual sections of the source code for the various EPS3 components to determine where the majority of time was being spent. Figure 2-1 shows a typical display when the program is fed a PGPROF statistics file. The top table lists execution time in seconds (column labeled "Time"), and the number of times each routine is called (column labeled "Count"). From this table, we can determine that 99% of the execution time is spent in two routines, prclbs and totlbs. The bottom table gives the execution time and number of times called for the selected function, for each parallel thread that was utilized during the execution of the program.

Once an individual function is selected, PGPROF will display the execution time and number of iterations for each source code line in the selected routine. An example of this display is shown in Figure 2-2. This kind of analysis is particularly useful in determining sections of the code that can be effectively parallelized.

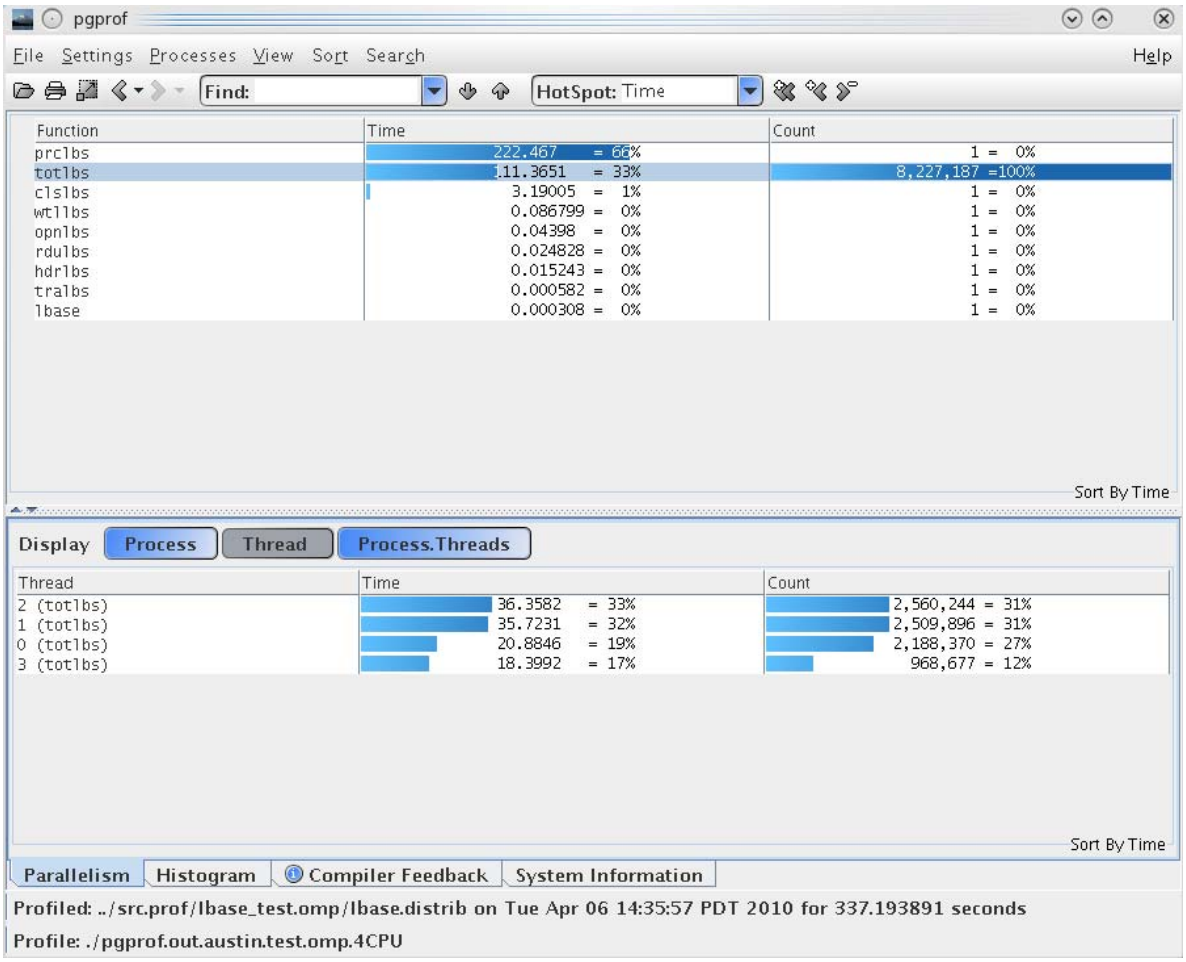

Figure 2-1. Example of PGPROF analysis of entire module.

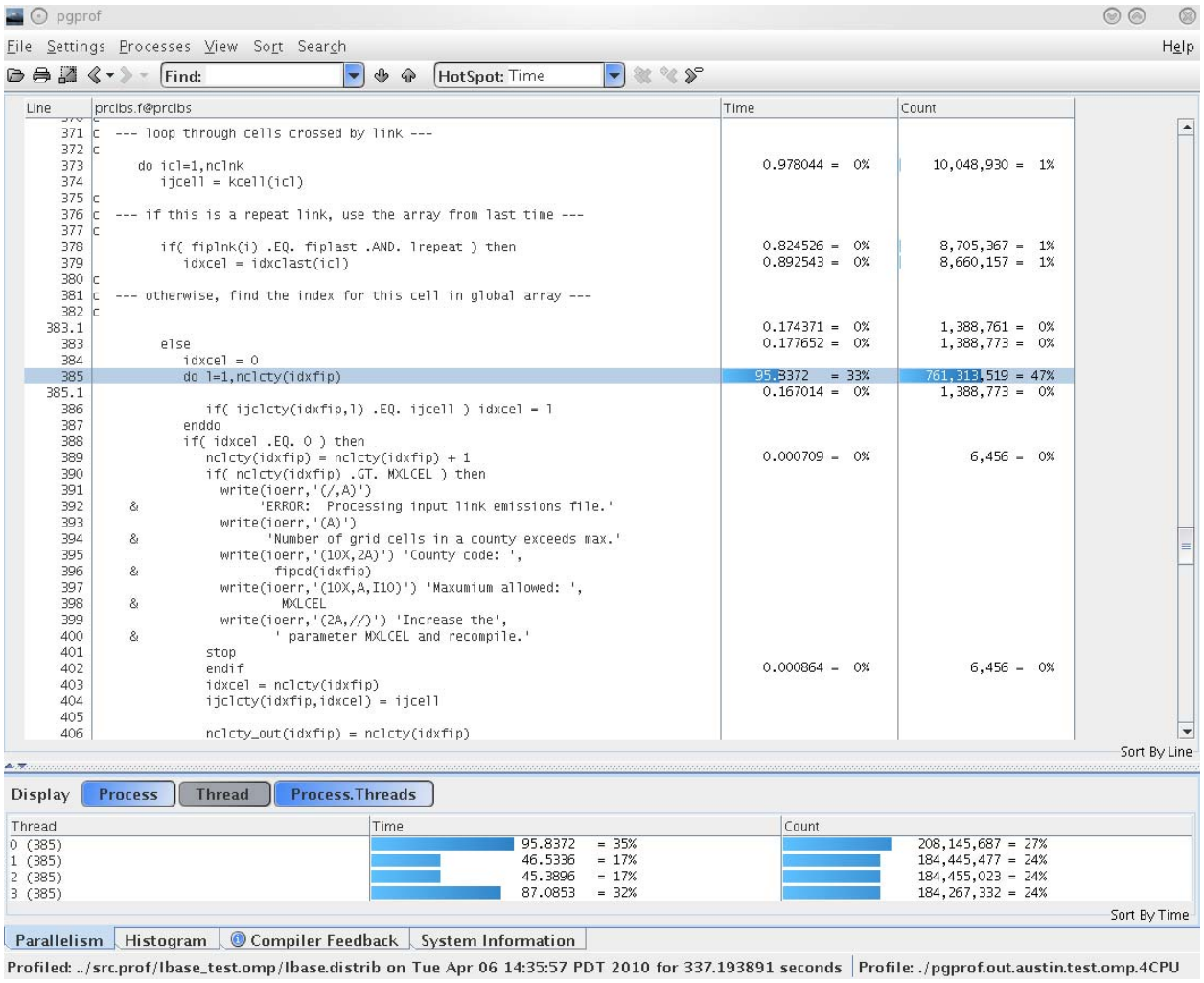

Figure 2-2. Example of PGPROF analysis of single subroutine.

2.3 Evaluation of Parallel Processing Strategies

Based on the PGPROF analysis, two parallelization strategies were investigated. Unfortunately, neither strategy proved effective in reducing the runtime for the targeted module. In fact, in some cases the runtimes actually increased when running with multiple processors.

The first method attempted is the classic approach of using OMP directives to parallelize a loop. Using the PGPROF output, it was possible to identify the loops that required the most time in each of the EPS3 modules GRDEM and LBASE. These modules were targeted by TCEQ staff as two of the most time-consuming programs. The code was modified by adding OMP directives to parallelize these loops and the modules were run using the testbed dataset provided by TCEQ staff. Although a number of configurations of multiple processor applications were tried (ranging from 2 processors to 8 processors), in no case did the parallelized version perform better than the single processor version. In some cases, it actually performed worse. Although this seems counter-intuitive, a quick review of the code provides a possible explanation. In each case the targeted loop contained a large multi-dimensional data structure. Although we were not able to confirm this, we suspect that parallelization requires the program to cache memory more often, leading to latency in memory access. Attempts were made to restructure the data structures

involved (including using dynamically allocated arrays), with the hope that memory access and runtime performance would improve. These attempts did not prove to be fruitful. Because the targeted loops were by far the most time-consuming parts of the code and they could not be effectively parallelized, this approach was not further investigated.

The second approach attempted was not a standard OMP approach. This involved reading large amounts of data and then parallelizing the processing of this data. Most of the EPS3 modules process one record of the input file at a time. A record is read and processed. The results are either written to the output file or stored until the end of the processing. ENVIRON proposed that the time required to process the entire file could be improved by reading a block of data at a time and then parallelizing the entire processing step. It was determined that the LBASE module is the best candidate for this approach because LBASE input files are typically the largest of any of the EPS3 entry level modules. The LBASE module was modified to read a number of records at a time and the data stored in arrays instead of scalars. An additional loop was added to the processing routine (prclbs.f) and this loop was parallelized using OMP directives. Although there was some improvement in the runtime with this modified version, it was not significant and did not scale well to larger number of processors. Tests were performed which adjusted the number of records read at a time. No significant improvement was reached. Because this approach was not effective at improving the LBASE module, and this was clearly the best candidate for this method, this approach was not further investigated.

2.4 Improvements to the MRGUAM Module

The EPS3 module MRGUAM is used to merge together several emission files that are in gridded UAM format. The MRGUAM program can become a significant bottleneck in the generation of a complete modeling inventory if many component files must be merged together. Intuitively, it would seem that this is not because MRGUAM is computationally intensive. Rather it is because it must be utilized multiple times, once for each modeling episode day, resulting in a significant amount of I/O. MRGUAM is a simple program; it reads a series of files, adds the numbers together and writes out a single file. After analysis using PGPROF, it was discovered that the MRGUAM module was not using the majority of its time during the I/O step. Instead, a significant amount of time was taken in tracking the processed emissions for purposes of producing output tables. An analysis of the pertinent source code showed that this data tracking was being performed in a very inefficient way. The code was originally written to utilize existing EPS3 data structures and library subroutines. This was done for convenience and ease of programming. The code was restructured by moving the lines that accumulate the emissions into the routine that performs the I/O, and customizing the data structures to better match the MRGUAM processing. This lead to a considerable improvement in the processing time. Table 2-2 shows an example of runtimes for the two versions of MRGUAM. Because the modifications affected only the emissions tables, the output files produced by the new version are identical. It was also verified using several applications that the numbers in the output tables produced by the two versions are identical.

Table 2-2. Timing of the MRGUAM module.

Domain size: 92 x 113 Number of species: 36		
Number of files merged	**Original (secs)**	**Modified (secs)**
5	14	4
10	28	7
20	54	14
30	81	22
Average time/file merged	2.7	0.7

2.5 Incorporating Hash Tables

After further investigation applying PGPROF to the EPS3 modules a discovery was made that lead to a speedup approach that would prove to be applicable not only to the modules involved, but to the system in general. Looking at the testbed dataset provided by the TCEQ staff for use with the CNTLEM module, the PGPROF output showed that most of the runtime was spent in routines that search for a match in a lookup table. The lookup table involved is particularly large. Also, the lookup tables in EPS3 are primarily of character data types and although Fortran is the language of choice for floating point calculations, it is notoriously slow when dealing with character strings. It seemed obvious that performance benefits could be gained by creating more efficient searching routines. The first attempt at this was to put a hash table into the routines that search the control factors file in CNTLEM. A hash table consists of 2 data structures: an initial pointer array and a mapping array; as well as a numeric function called a hash function. When the input data is read and stored normally, the hash table is populated in the following way:

1. Apply a hash function to the input data. (for example, sum the ASCII codes of the characters in the source category code)
2. Use the hash function value as a location in the initial pointer array and determine if this pointer has been set.
3. If it has not been set, then initialize it to be the location of this data item in the data array.
4. If it has been set, then look for a match in the data array at the location indicated by the initial pointer value.
5. If no match is found, then look in the mapping array for the location of the next data item that hashed to this same hash value.
6. Continue looking in this manner until either a match is found or the mapping array points to an empty location in the data array.
7. In the case of an empty location, update this location so that it points to the current data item.

The lookup for data is performed using a similar process, with the difference being the action taken if no match is found in the existing hash table. The efficiency of a lookup using a hash table depends on the hash function applied. An ideal hash function will produce a different hash value for each possible data item. The most wasteful hash function will produce the same hash value for each data item. Most applications are somewhere in between, balancing the higher memory requirements required by a hash table that needs a great deal of array space (an ideal hash function) with the inefficiency introduced by having to perform multiple lookups (the most

wasteful hash function). The goal is to find a hash function with a reasonable number of possible values that evenly distributes the number times each distinct value will be generated.

After some experimentation, a reasonably well-distributed hash function was created for the CNTLEM module. This was a function of the FIPS code/source category code pair that is the driver for the lookup of the control factors file. The performance improvement gained using this implementation on the CNTLEM testbed was startling. Part of the reason for this performance is because the CNTLEM module performs a lookup of the control factors data structure at two times during its execution. In addition to searching the lookup table when trying to find a match of the current data record for purposes of applying controls, it also must make a series of lookups when reading the control factors file itself. This is done to identify when the control factors file contains duplicate identification information and inform the user accordingly. Table 2-3 shows the speedup gained in CNTLEM using the testbed dataset.

Table 2-3. Timing of the CNTLEM Module.

Version	Time
Original	2 hours, 5mins
Modified	24 secs

After the success of the hash table approach, hash tables were implemented at various places within the EPS3 system, with varying success. Some were unique to a given module, such as the hash table added to the "emissions by surrogate" lookup table in the GRDEM module. Others were system wide, such as the lookup table for storing totals by source category code. The greatest success was with the GRDEM module, when it is run with the option to generate the surrogate report. Table 2-4 shows this performance increase.

Table 2-4. Timing of the GRDEM Module.

Area Sources Generating A Surrogate Report	
Version	**Time**
Original	4 hours 20 min
Modified	10 min

3.0 RESULTS

Although the parallelization of the EPS3 system was not successful, significant improvements were made to the system, greatly reducing the time needed to produce model-ready emissions files. This is especially true for longer simulations such as an annual PM simulation. Timing numbers have already been presented here, but these were for individual modules using testbed datasets. A more realistic timing comparison can be performed by looking at a production dataset and measuring the performance improvements from the beginning of the application to the completion of the entire set of inventory files. After receiving the modified source code, the TCEQ staff did some of these comparisons and provided the results for use in this report. Table 3-1 shows some of the timing comparisons performed by the staff at TCEQ.

Table 3-1. Timing for complete inventory processing.

Domain / Run ID	Original (mins)	Modified (mins)
4km / b11a_9co.diesel_NOxCor	28	1
4km / b11a_no9co.diesel_NOxCor	40	1
12km / b11a.diesel_NOxCor	135	3
36km / b11a.diesel_NOxCor	90	2

Appendix A
PIGEMS Section from EPS3 User's Guide

3.13 PIGEMS

The PIGEMS (**P**lume-**I**n-**G**rid **EM**ission**S**) module serves two functions. First, it serves as the exit stage for the EPS3 system, producing an elevated emissions file for the CAMx photochemical model. Second, it provides a sophisticated methodology for flagging sources for the Plume in Grid (PiG) treatment within CAMx. Ultimately, the decision to simulate a point source plume with PiG is based on the emission rate of the source. Any source that emits at a rate greater than a user-specified threshold value will be flagged for the PiG treatment. The complexity of the PIGEMS module stems from two design elements:

1) flexibility in how threshold values can be assigned, and
2) ability to combine similar co-located sources into a single plume

> ➢ **Special Note**
> Although the execution of PIGEMS can be complicated, it can also be easily run to format elevated point source stack lists and emission files prepared by PSTPNT for input into the CAMx model. In this form, the PIGEMS module serves as tool for merging elevated point source files and reformatting for use with CAMx. See the "Running the PIGEMS Module" later in this section.

Flexible Emissions Threshold

It is often advantageous to apply a different threshold value for PiG sources based on criteria such as geographical location. For example, the sources contained in a small region of interest within the modeling domain could be selected for PiG using a lower threshold, producing a finer network of PiG sources in that area, without unnecessarily generating a multitude of PiG sources in the surrounding region. The PIGEMS module supports variable threshold values by allowing users to assign a threshold code. Associated with each threshold code is a pollutant name and an emissions value. Any source that emits the specified pollutant above the threshold rate will be flagged as a PiG source. Threshold codes are defined in the /THRESHOLD VALUE/ packet of the USERIN file found under the discussion of the PIGEMS inputs. Each source in the emissions inventory is then assigned to a threshold code. This assignment can be done in two ways, by county or by grid cell. The assignment of sources by grid cell is done via a "mask" file.

Combining Sources Using Co-location

The simplest methodology by which to identify a source for the PiG treatment is to compare the emission rate of the source to the threshold value. However, this methodology would exclude clusters of similar, closely located sources from which individual plumes would merge to form a

single plume such that the combined emission rate exceeds the threshold. For example, a facility could contain two stacks, standing tens of meters apart, each emitting at a rate that is two-thirds of the threshold value. Treated individually, neither of these stacks would be flagged for PiG. But combined as a single source, the pair easily exceeds the threshold. The process of combining multiple sources that are in close proximity to each other is defined as "co-location." The PIGEMS module determines co-located sources using the following steps:

1. Determine if the source is "significant" (defined as one-fourth of the threshold value). Sources that emit below this level are not considered.

2. Find all sources that are within a user-specified co-location distance. These are considered close enough to merge into a single common plume immediately after release. In order to avoid including a large number of small sources in a single cluster, sources that are deemed to be "insignificant" (less than 2.5% of the threshold) are ignored.

3. Compare the aggregate emission rate for sources in this cluster against the threshold value.

4. If the aggregate emission rate for the cluster exceeds the threshold value, a new source is created with the combined emission rate of the cluster, and this source is flagged for PiG treatment. The stack parameters of the new source will be an "average" of the stack parameters of all of the sources in the cluster. See the description of the stack parameter averaging presented later in this section.

5. Eliminate the sources in the cluster from consideration for co-location of other sources. This prevents a source from being included in multiple clusters, and thus potentially multi-counting that source's emissions.

The co-location of sources cannot be performed for multiple pollutants. Although it is possible to flag individual sources for PiG treatment according to either a NOx threshold value or a different VOC threshold value, the PIGEMS module can only co-locate multiple sources for PiG based on one pollutant. The name of the pollutant that is to be used as the co-location pollutant, as well as the co-location distance, is supplied in the /THRESHOLD VALUE/ packet of the USERIN file.

Combining a User-Defined Array of Sources

Co-locating sources is performed automatically and is determined strictly by proximity. However, there may be instances where it would be necessary to combine sources that would not be clustered using the co-location algorithm. For example, it may make sense to cluster a large array of small sources, such as in an oil or chemical processing facility, that are spread out over a fairly large area but act photochemically as a single plume from the facility. For this reason, the PIGEMS module allows for user-specified clusters. The user must provide newly created stack identification information (county code, facility ID and stack ID) for the new source, as well as the existing stack identification information for all of the sources that are to be combined to create this new source. Any existing stacks can be combined into a new source, although this only makes sense from a photochemical modeling standpoint for a few specific cases. Note that this feature should be used only after careful consideration of the possible impacts.

There is an option to specifically supply stack parameters and location coordinates of the newly created source, or have the PIGEMS module automatically average the values from the combined sources to create the stack parameters of the newly created source. To have the stack parameters and location coordinates automatically calculated, simply enter a missing value (-9) in the field for that parameter in the /COMBINE STACKS/ packet of the USERIN file. The /COMBINE STACKS/ packet is described in detail later in this section.

The Master Stack List

In many instances, point sources are processed using day-specific emissions in which some sources operate on a regular schedule (e.g., weekday/weekend), and others are used only occasionally (e.g., peaking units, etc.). If separate inventory processing streams are used for each day of a simulation period, this can lead to different point source lists day-to-day as various point sources shut down and others start up. In order to ensure that all point sources that will emit above the user-defined thresholds at some period in the simulation will receive the PiG treatment, it is necessary to have a single cohesive list of stacks for the inventory. This list, called the master stack list, must include any stack that emits at any time during the modeling period. For example, an auxiliary unit that emits at a rate that exceeds the threshold (and so should be flagged for PiG) may be active only during a weekend period. If PiG selection is applied only to the weekday emissions inventory files, this source would be skipped. The simple solution is to run PiGEMS for each day in the episode. However, this could lead to a series of independent PiG point source emissions files, each having a different list and order of stacks. While CAMx can accept such inconsistent PiG inputs (even if OSAT is invoked), these inconsistencies can lead to difficulties with database management.

The PIGEMS module handles this complication by first creating and then utilizing a master stack list. The first application of the PIGEMS module to an inventory should be performed to create a master stack list over all simulation days. When creating the master stack list, the PIGEMS module will also identify all of the sources that should be flagged for PiG treatment and put those sources at the top of the list. It will also identify any clusters via co-location, or user-defined stack combination, and create records in the master stack list for the newly created stacks. The master stack list is an ASCII file and can be edited to override the automatic assumptions made by the PIGEMS module. Once the master stack list is created, the PIGEMS module can be run again to create an emissions inventory file for each specified day of the inventory. The emissions inventory file will contain the entire complement of sources in the master stack list, including sources that do not emit on the given day.

The Static Sources File

Typically, the PIGEMS module must process a tremendous amount of data, effectively the entire point source inventory. Also, because the point source inventory may contain day-specific emissions estimates, the PIGEMS module must be exercised for each day of the modeling episode. This can lead to a time-consuming procedure and in some instances a good deal of this processing can be avoided. The sources that do not vary by day, which are refererred to as “static sources”, need only be processed once for the entire modeling episode. PIGEMS supports this through a mechanism called the “static sources file”. PIGEMS optionally reads a CAMx-ready point source emissions file that may or may not include sources selected for PiG. This point

source file should include all sources that do not change from day-to-day (the static sources). PIGEMS then reads emissions for additional time varying point sources and performs calculations to select sources for PiG treatment. Finally, PIGEMS merges both sets of sources into a single CAMx-ready point source emissions file, creating a complete inventory file.

To utilize the static sources file, the PIGEMS module must be run in stages. First, all sources that are unchanged in future scenarios are processed normally by PIGEMS. In most cases this will be the bulk of the point source inventory. The first stage can also be used to process point sources that have a "representative day", for example a Weekday/Weekend. In this case, PIGEMS should be run two separate times, generating a static sources file for each representative day. The output file produced in this first stage would then be used as a "static" file in further processing. In the second stage, the user must supply to PIGEMS the static sources file plus additional emission files containing the sources that were excluded from the static file processing (the day-specific sources).

RUNNING THE PIGEMS MODULE

Because of its flexibility and dual purpose nature – creating a master stack list or creating a model ready emissions file – the PIGEMS module job stream can be a bit daunting to configure. Some files are always needed; others should only be supplied under certain conditions. What follows are a few examples of PIGEMS run scripts, demonstrating how the module should be configured for a few of the typical scenarios.

Running PIGEMS When No PiG Treatment Is Needed

When no PiG is being performed by the photochemical model, the PIGEMS module acts as a simple reformatting program, with the added ability to merge files generated by separate EPS3 streams. Because of their consistent format, a stack definition file output by PSTPNT can be used as a master stack list input to the PIGEMS module. To ensure that all stacks are included in the inventory it is necessary to concatenate each of the stack definition files into a single stack definition file.

Exhibit 3-53 shows a sample job script for running the PIGEMS module when no PiG treatment is needed. In this case, it is not necessary to include any of the PIGEMS specific packets of the USERIN file. If these packets are included, they will be ignored. The output from this run of the PIGEMS module is a model-ready emissions file that contains all of the sources but represents one day of the episode.

```
#!/bin/csh
#
#   --- Sample job script for running the
#       EPS3 module PIGEMS
#
set EXEC = /models/eps3/src
#
set SCENARIO = test_problem
#
foreach DAY (wkd sat sun)
#
rm -f ../msg/msg.pigems.$SCENARIO.$DAY
rm -f ../emiss/ptsrce.$SCENARIO.$DAY.bin
#
echo "------------------------------------------------"
echo "    Running PIGEMS for $SCENARIO - point sources"
echo "         Day - $DAY"
echo "------------------------------------------------"

$EXEC/pigems/pigems << IEOF
USERIN file        :../inputs/userin.$SCENARIO.$DAY
Master Stack List  : ../emiss/stkhdr.pts.$SCENARIO.$DAY
No Stack def. file :/END/
Stack emiss files  : ../emiss/elvems.pts.$SCENARIO.$DAY
                     /END/
Threshold Mask file:
Static sources file:
Message file       :../msg/msg.pigems.$SCENARIO.$DAY
Output Master Stack:
Output Emissions   :../emiss/ptsrce.$SCENARIO.$DAY.bin
IEOF
end
```

Exhibit 3-53. Example of a PIGEMS job script, when no PiG treatment is needed.

Running PIGEMS to Create the Master Stack List

To create the master stack list, both the full complement of stack definition files and the full complement of stack emissions files must be supplied to the PIGEMS module. All of the stack definition files, including all days, must be supplied so that the master stack list includes each stack that emits at any time during the modeling episode. The PIGEMS module ignores repeated instances of stacks, creating a stack list that contains a single entry for each stack. All of the stack emissions files, including all days, must be supplied so that the PIGEMS module can determine if any stack emits at a rate that exceeds the threshold at any time during the modeling episode. Obviously, a valid filename must be supplied for the output master stack list, but it is important that the filename for the input master stack list is left blank. Exhibit 3-54 shows an example of a job script configured for the PIGEMS module to **create** a master stack list.

```
#!/bin/csh
#
#  Script to run PIGEMS to create master stack list
#
#==============================================================================
#         Run PIGEMS to Create the Master Stack List
#==============================================================================
#
rm -f ../msg/msg.pigems.ext_reg.nei99v3.master_stacklist
rm -f ../emiss/master.stacklist.ext_reg_nei99v3
#
../../src/pigems/pigems.pts << ieof
USERIN file         :../inputs/userin.ext_reg.990813
Master Stack List   :
No Stack def. files:../emiss/stkdef.pt.TX.egu_99po.990813
                    :../emiss/stkdef.pt.TX.egu_99po.990814
                    :../emiss/stkdef.pt.TX.egu_99po.990815
                    :../emiss/stkdef.pt.TX.egu_99po.990816
                    :../emiss/stkdef.pt.TX.egu_99po.990817
                    :../emiss/stkdef.pt.TX.egu_99po.990818
                    :../emiss/stkdef.pt.TX.egu_99po.990819
                    :../emiss/stkdef.pt.TX.negu_99po.sat
                    :../emiss/stkdef.pt.TX.negu_99po.sun
                    :../emiss/stkdef.pt.TX.negu_99po.wkd
                    :../emiss/stkdef.ext_reg.others_nei99v3.sat
                    :../emiss/stkdef.ext_reg.others_nei99v3.sun
                    :../emiss/stkdef.ext_reg.others_nei99v3.wkd
/END/
Stack emiss files   :../emiss/stkemis.pt.TX.egu_99po.990813
                    :../emiss/stkemis.pt.TX.egu_99po.990814
                    :../emiss/stkemis.pt.TX.egu_99po.990815
                    :../emiss/stkemis.pt.TX.egu_99po.990816
                    :../emiss/stkemis.pt.TX.egu_99po.990817
                    :../emiss/stkemis.pt.TX.egu_99po.990818
                    :../emiss/stkemis.pt.TX.egu_99po.990819
                    :../emiss/stkemis.pt.TX.negu_99po.sat
                    :../emiss/stkemis.pt.TX.negu_99po.sun
                    :../emiss/stkemis.pt.TX.negu_99po.wkd
                    :../emiss/stkemis.ext_reg.others_nei99v3.sat
                    :../emiss/stkemis.ext_reg.others_nei99v3.sun
                    :../emiss/stkemis.ext_reg.others_nei99v3.wkd
                     /END/
Threshold Mask file:../inputs/pigems.maskfile.4km
Static sources file:
Message file        :../msg/msg.pigems.ext_reg.nei99v3.master_stacklist
Output Master Stack:../emiss/master.stacklist.ext_reg_nei99v3
Output Emissions    :
ieof
```

Exhibit 3-54. Example of a PIGEMS job script. This is used to create a master stack list.

Running PIGEMS to Create a Model Ready Emissions File

Once the master stack list is created, the PIGEMS module can be run to create model-ready emissions files. A PIGEMS run must be made for each day of the modeling episode. In this configuration, the master stack list created in the previous run is used as the input stack list and only the stack emissions files for the day of interest are included. The lines containing the output master stack list and the threshold map file are ignored, as are the PIGEMS specific packets of the USERIN file. Exhibit 3-55 shows a job script for creating model-ready emissions files using the PIGEMS module. Notice that this script has a loop that walks through each day of the inventory.

```
#!/bin/csh
#
#  Script to run PIGEMS to create model-ready files
#
# --- loop over all days in the episode ---
#
foreach f (13.wkd 14.sat 15.sun 16.wkd 17.wkd 18.wkd)
#
#===============================================================================
#         Run PIGEMS to Create Model-Ready File
#===============================================================================
#
set cal = $f:r
set rep = $f:e
#
rm -f ../msg/msg.pigems.PiG.ext_reg.nei99v3.9908$cal
rm -f ../emiss/ptsrce.PiG.ext_reg_nei99v3.9908$cal.bin
#
echo '---------------------------------'
echo "       Doing Aug $cal"
echo '---------------------------------'
#
../../src/pigems/pigems.pts << ieof
USERIN file         :../inputs/userin.ext_reg.9908$cal
Master Stack List   :../emiss/master.stacklist.ext_reg_nei99v3
No Stack def. files:/END/
Stack emiss files   :../emiss/stkemis.pt.TX.egu_99po.9908$cal
                    :../emiss/stkemis.pt.TX.negu_99po.$rep
                    :../emiss/stkemis.ext_reg.others_nei99v3.$rep
                     /END/
Threshold Mask file:
Static sources file:
Message file        :../msg/msg.pigems.PiG.ext_reg.nei99v3.9908$cal
Output Master Stack:
Output Emissions    :../emiss/ptsrce.PiG.ext_reg_nei99v3.9908$cal.bin
ieof
end
```

Exhibit 3-55. Example of a PIGEMS job script, to create the PiGged model ready files for each day in the inventory.

Running PIGEMS to Insert Additional Sources

Sometimes it is necessary to insert new sources into an inventory after all of the main processing has been completed. For example, on certain days of the episode in which a considerable number of wildfires added to the air quality problem, it may be necessary to insert the wildfire emissions, which have been processed as point sources. Or perhaps a future year control strategy calls for a new unit, which was not included in the standard processing, to be included in the inventory. The PIGEMS module allows for additional sources to be inserted when creating a model-ready emissions file. The new sources will be added to the list of sources and included in the inventory, but will not be considered for PiG treatment. The sources can be flagged for PiG treatment by manually changing the stack diameter field on the stack definition record to a negative value. To insert additional sources in the inventory, merely supply the PIGEMS module with an input master stack list and any number of stack definition files. Of course, the stack emissions files for the additional sources must also be supplied. Exhibit 3-56 shows an example of how to configure the PIGEMS module to insert additional sources for one day of the episode. In this example, a file containing wildfire emissions modeled as point sources is inserted.

```
#!/bin/csh
#
#  Script to run PIGEMS to create model-ready file
#  On this day there were wild-fires (which should
#  not be PiGged)
#
#=============================================================================
#         Run PIGEMS to Create PiGged PTSRCE file for Aug 15 (Sunday)
#=============================================================================
#
rm -f ../msg/msg.pigems.PiG.ext_reg.nei99v3.990815
rm -f ../emiss/ptsrce.PiG.ext_reg_nei99v3.990815.bin
#
../../src/pigems/pigems.pts << ieof
USERIN file         :../inputs/userin.ext_reg.990815
Master Stack List   :../emiss/master.stacklist.ext_reg_nei99v3
Additonal Stack def:../emiss/stkdef.fires.990815
                     /END/
Stack emiss files   :../emiss/stkemis.pt.TX.egu_99po.990815
                    :../emiss/stkemis.pt.TX.negu_99po.sun
                    :../emiss/stkemis.ext_reg.others_nei99v3.sun
                    :../emiss/stkemis.fires.990815
                     /END/
Threshold Mask file:
Static sources file:
Message file        :../msg/msg.pigems.PiG.ext_reg.nei99v3.990815
Output Master Stack:
Output Emissions    :../emiss/ptsrce.PiG.ext_reg_nei99v3.990815.bin
ieof
```

Exhibit 3-56. Example of a PIGEMS job script, to add additional sources to the inventory.

Input Files

The PIGEMS program anticipates five input files: the USERIN file, the master stack list, stack definition files, stack emissions files, and an optimal threshold mask file. The input master stack list is blank if the PIGEMS run is to create the list. The stack definition files are optional if an input master stack list is provided.

USERIN File

The PIGEMS module reads four of the standard USERIN packets and an additional three packets specific to PIGEMS.

/DATE/ This packet (Table 2-6) provides the modeling episode information.

/REGION/ This packet (Table 2-7) provides the region definition and is used to convert source locations to grid cell.

/CRITERIA POLLUTANT/ PIGEMS reads this packet (Table 2-9) to determine which of the input pollutants to include in the modeling inventory. .

/SPECIES LIST/ This packet (Table 2-11) provides the air quality model species list.

All of the PIGEMS-specific packets are optional, although failing to supply either the /THRESHOLD VALUE/ or the /THRESHOLD MAP/ packets will result in the inability of the PIGEMS module to flag any sources for PiG treatment. Therefore, when processing an inventory for a photochemical modeling application in which the Plume-in-Grid sub-model will not be used, the THRESHOLD packets are not required.

/THRESHOLD VALUE/ This packet contains the threshold values that will be used to determine if a source is to be flagged for PiG. The first record of this packet contains the co-location information: specifically, the co-location distance and the co-location pollutant (Table 3-34). The co-location distance is the distance (in meters) that the PIGEMS module will use to determine if a group of stacks forms a cluster. The emissions from stacks that are within this distance of one another will be aggregated to determine if the combination exceeds the threshold. Because a different threshold value can be supplied for each pollutant, the PIGEMS module also needs to be provided the name of the pollutant that is to be used for purposes of co-location. This is referred to as the co-location pollutant. Only threshold values associated with this pollutant will be used when determining the co-located sources.

The remaining records of the /THRESHOLD VALUE/ packet contain the threshold codes and the associated threshold value (in tons per day), as well as the name of the pollutant to which the threshold applies. The threshold code must be an integer between 1 and 99. The same threshold code could be used for multiple pollutants. If a source is mapped to this threshold code then it will be flagged for PiG treatment if either of the threshold values is exceeded. For example, suppose there is a record with a threshold code of 1 that has a NOx threshold of 50 tons per day. Also suppose there is a record with threshold code 1 but with a VOC threshold of 10 tons

per day. If a source (or a cluster of sources) emits over 50 tons per day of NOx *or* 10 tons per day of VOC it will be flagged for PiG. Table 3-34 shows the format of the /THRESHOLD VALUE/ packet. Exhibit 3-57 shows a sample /THRESHOLD VALUE/ packet.

Table 3-34. The /THRESHOLD VALUE/ packet.

Variable	Columns	Description
	1 – 20	Ignored, used for notational purposes.
Colodist	21 – 30	Co-location distance (m)
Colocpol	31 – 40	Pollutant for which co-location is based.
ithrshcd( )	21 – 30	Numeric code for threshold (1-99)
Thrshspc( )	31 – 40	Criteria pollutant code for which the threshold applies.
thrshval( )	41 – 50	Threshold value for corresponding code and pollutant.

```
/THRESHOLD VALUE/
coloc dist, poll    : 250.        NOX
id, specie, val     : 1           NOX       20.
id, specie, val     : 2           NOX       50.
/END/
```

Exhibit 3-57. Sample /THRESHOLD VALUE/ packet. This packet could be used to apply a more stringent threshold to an area of interest.

/THRESHOLD MAP/ This packet is used to assign a threshold code to sources based on the county in which they are contained. Each record defines one mapping, by defining a state/county FIPS code and the threshold code to which that county is to be assigned. The threshold code must be listed in the /THRESHOLD VALUE/ packet where the actual threshold value will be defined. The regular EPS3 best-match criteria will be applied to the FIPS codes for purposes of matching sources. For example, a code of 06067 will be a better match than a code of 06000. Because all sources should have some threshold code mapping, it is important to have a record in this packet with the "catch-all" FIPS code of 00000. If this is not included some sources may not have a threshold code mapping and will not be considered for PiG treatment. Table 3-35 shows the format of the /THRESHOLD MAP/ packet, while Exhibit 3-58 shows an example.

Table 3-35. The /THRESHOLD MAP/ packet.

Variable	Columns	Description
	1 – 20	Ignored, used for notational purposes.
thrshcty( )	21 – 30	County code for which threshold applies.
ithrshmap( )	31 – 40	Numeric code for threshold value (corresponds to code specified in /THRESHOLD VALUE/ packet).

```
/THRESHOLD MAP/
All counties        : 00000     2
Texas counties      : 48000     1
/END/
```

Exhibit 3-58. Sample /THRESHOLD MAP/ packet. This packet could be used to apply a more stringent threshold to an area of interest.

/COMBINE STACKS/ This packet allows any user-defined set of stacks to be combined into a single stack. This facilitates the clustering of sources that would not normally be combined using the automated co-location algorithm. This packet is comprised of a set of records, each set defining the new stack's characteristics as well as identifying the existing stacks that will be combined. The format of the packet records (Table 3-36) is the same as the records in the stacks definition file produced by the PSTPNT module and the master stack list processed by the PIGEMS module. This means populating a /COMBINE STACKS/ record can be done by extracting the records of the stacks to be combined from the existing stack definition file and then creating a new stack definition record for the newly created source. Note that the new stack definition record is similar to the existing stack definition records but has an additional field for determining if the newly created stack should be considered for PiG treatment. It may be useful to combine stacks without applying the PiG treatment and the /COMBINE STACKS/ allows for this.

Because the record identifying the new source must have unique stack identification information, it is important to come up with a coding scheme that will yield a unique stack ID. For example, the string NEWSTK0001 could be used for the new stacks, incrementing the numeric portion for each new stack. The county code and facility identification codes should be taken from the sources being combined. The keyword at the beginning of the record should be the word "NEW" for the new stack definition record, and a blank string for those records listing the sources to be combined. When combining stacks defined in the /COMBINE STACKS/ packet, the PIGEMS module will either use the stack parameters listed on the new stack record or, alternatively, "average" the stack parameters of the stacks to be combined. To force the use of the averaging scheme the stack parameters on the new stack record should be set to missing value (-9.0). When "averaging" stack parameters of combined stacks, the PIGEMS module will follow this algorithm:

1. Average the coordinates,
2. Average the volumetric flow rates,
3. Calculate the square root of the sum of the squares of the diameter,
4. Calculate the maximum stack height,
5. Calculate the average of the exit temperatures weighted by the volumetric flow rate,
6. Calculate the gas exit velocity from the newly calculated volumetric flow rate and diameter.

This is also the averaging scheme that is used when combining sources that are automatically co-located. Exhibit 3-59 provides an example of the /COMBINE STACKS/ packet.

Table 3-36. The /COMBINE STACKS/ packet.

Variable	Columns	Description
The following refers to the new stack record. This record is followed by a series of existing stack records, one for each existing stack to combine.		
keyin	1 – 10	Keyword 'NEW' for new stack definition followed by keyword 'PIG' or 'NOPIG' indicator.
ctynew( )	12 – 21	County code for the new stack.
facnew( )	23 – 32	Facility ID for new stack
stidnew( )	34 – 43	Stack ID for new stack
hgtnew( )	45 – 54	Stack height for new stack (negative value indicates to calculate from combined stacks) (m)
dianew( )	56 – 65	Stack diameter for new stack (m)
tptnew( )	67 – 76	Stack gas exit temperature for new stack (K)
velnew( )	78 – 87	Stack gas exit velocity for new stack (m/s)
xlocnew( )	89 – 98	X-coordinate for new stack
ylocnew( )	100 - 109	Y-coordinate for new stack
The following refers to the existing stack record. These records are the same format as the stack definition file(output of PSTPNT) and the master stack list. There will typically be multiple instances of this record for each new stack record.		
	1 – 10	Blank indicates stack to be combined
ctycomb( )	12 – 21	County code for stack to combine to create new stack
faccomb( )	23 – 32	Facility ID for stack to combine to create new stack
stidcomb( )	34 – 43	Stack ID for stack to combine to create new stack

```
/COMBINE STACKS/
NEW PIG    01001      1          NEWSTK0001       -9.0      -9.0       -9.0       -9.0       -9.0     -9.0
           01001      1          001
           01001      1          002
           01001      1          003
NEW NOPIG  01073      7          NEW0000002      215.5      7.620       433.     90720.   1176037.   -596587.
           01073      7          001             213.4      7.620       433.     90720.   1176037.   -596587.
           01073      7          002             213.4      7.620       409.     73800.   1176037.   -596587.
           01073      7          004             213.4      7.620       411.     81360.   1176037.   -596587.
           01073      7          005             215.5      7.620       409.     80280.   1176053.   -596685.
           01073      7          005
/END/
```

Exhibit 3-59 Sample /COMBINE STACKS/ packet. (Notice that since only the first 3 fields of the records containing the stacks to combine are read, the later fields can be included, but they will be ignored.)

The Input Master Stack List

This file contains the list of all sources that will emit at any time during the episode period. The PiG sources will appear at the top of the file. The format of the master stack list is identical to the format of the stacks definition file (Table 3-37) produced by the PSTPNT module. Therefore, for an application in which the PiG treatment is not needed it is possible to supply the PSTPNT-produced stack definition file to the PIGEMS module as the master stack list. The result will be a model ready emissions file with no sources flagged for PiG. In the likely event that the EPS3 processing produced multiple PSTPNT stack definition files (by source category, for example) it is necessary to concatenate the stack definition files from the separate EPS3 streams. Since the PSTPNT stack definition file is an ASCII file with one record for each stack, this can be done using the Unix “cat” command, followed by the “sort –u” command to generate a unique list of stacks.

The PIGEMS module will typically be executed several times. The first run, which is significantly different from the standpoint of input file requirements than succeeding runs, is performed to create the master stack list. In this instance, the input master stack list filename should be left blank. This tells the PIGEMS module that a master stack list should be created, in which case an output master stack list should be supplied. If both an input master stack list file and an output master stack list file are supplied it is assumed that the input master stack list is valid and the PIGEMS module will ignore the line containing the name of the output master stack list. Subsequent runs of the PIGEMS module produce model-ready inventory files for each of the days of the modeling episode. In this instance, the master stack list file produced during the first execution of the PIGEMS module should be supplied as the input master stack list.

Table 3-37. Stack definition file format.

Variable	Columns	Description
Keywrd	1 – 10	Keyword; ORIGINAL
Ctytmp	12 – 21	County FIPS code
Factmp	23 – 32	Facility identifier
Stktmp	34 – 43	Stack identifier
Hgttmp	45 – 54	Stack height (m)
Diatmp	56 – 65	Stack diameter (m)
Tpttmp	67 – 76	Stack exit gas temperature (K)
Veltmp	78 – 87	Stack exit gas velocity (m/s)
Xloctmp	89 – 98	Stack X location
Yloctmp	100 – 109	Stack Y location

The Input Stack Definition Files

The PIGEMS module accepts an indeterminate number of PSTPNT-produced stack definition files. These must be supplied when creating a master stack list, but also can be used to add additional sources to an inventory. For example, a user may want to add a list of permitted, but not yet constructed sources, as a sensitivity run. When supplied both an input master stack list and additional input stack definition files, the PIGEMS module will simply add the sources contained in the stack definition files to the list of sources from the master stack list to include them in the inventory. Be aware that when adding sources to an inventory in this way, the additional sources will not be flagged for PiG treatment. When creating a master stack list, all of the stack definition files from the separate PSTPNT runs must be supplied, including the stack definition files for the different episode days and from different source category streams. The PIGEMS module will ignore the duplicates and create a stack list that contains all of the sources, apply the various criteria for application of PiG treatment, and write out a comprehensive master stack list. To supply multiple input stack definition filenames, simply list each file on a separate line and terminate the list with the keyword /END/ in place of a filename (Exhibit 3-54).

The Input Stack Emissions Files

Like the stack definition files, the stack emissions files (Table 3-38) are produced by the PSTPNT module. One stack emissions file will be produced for each EPS3 stream. Unlike the stack definition files, a set of stack emissions files must be supplied each time the PIGEMS module is utilized. When creating a master stack list, the PIGEMS module needs access to the entire complement of stack emissions files, including all days in the episode. This is necessary

for the PIGEMS module to determine if a source (or source cluster) exceeds the threshold on any day during the entire episode. The PIGEMS module merely calculates the maximum daily emissions for each source and uses this value to compare against the threshold value for purposes of flagging sources for PiG treatment. When running the PIGEMS module to produce a model-ready emissions file for point sources it is only necessary to supply the emissions files for that day of the episode.

Table 3-38. PSTPNT emissions output file format.

Variable	Columns	Description
ctytmp	1 – 10	County FIPS code
factmp	12 – 21	Facility identifier
stidmp	23 – 32	Stack identifier
hgttmp	34 – 43	Stack height (m)
diatmp	45 – 54	Stack diameter (m)
tpttmp	56 – 65	Stack exit gas temperature (K)
veltmp	67 – 76	Stack exit gas velocity (m/s)
xloctmp	78 – 87	Stack X location of (x,y) convention
yloctmp	89 – 98	Stack Y location of (x,y) convention
	100 – 105	Inventory date
	107 – 116	Criteria pollutant code
	118 – 127	Daily emissions estimates for criteria pollutant (tons/day)
spctmp	129 – 138	Speciated pollutant code
emsval( )	140 – 149	Emission estimates for speciated pollutant for hours 1 through 24
	151 – 160	(moles/hour)

During EPS3 processing it is often the case that a representative day inventory file is generated, rather than a specific-day inventory file. For example, it is typical to produce just a single file representing all weekdays in the summer season, rather than the dozens of files that would account for weekdays during this period. For this reason, the PIGEMS module does not perform any date matches when processing the emissions files. Therefore, it is critical to ensure that the list of stack emissions files supplied to the PIGEMS module represents a single day inventory that is complete. All of the emission records for a given stack will be added to the total emissions for the stack. To terminate the list of emissions files in the job script of the PIGEMS module, include the /END/ keyword in place of a filename.

The Threshold Mask File

The optional threshold mask file is used to define the threshold value for sources based on the grid cell in which the source is located. The mask file is essentially a map of the modeling domain in which each modeling grid cell is assigned a number, called the threshold code, representing the threshold value to be applied to that cell. The origin of the modeling domain, defined in the /REGION/ packet of the USERIN file, appears as the lower left-hand corner of the mask file, so that when viewing the file in an editor it looks like the modeling domain, easting moving from left to right and northing moving from bottom to top. The threshold code, which is an integer, appears in a field of 3-characters.

Because political boundaries are more precise in identifying sources than modeling grid cells, any assignment in the /THRESHOLD MAP/ packet that is not the global county code (00000) will override the assignment in the mask file. For example, suppose a source in Sonoma County, CA is

assigned a threshold code of 1 using the threshold mask file. If the only match for Sonoma County in the /THRESHOLD MAP/ packet is the global county code 00000, the mask file will take priority. However, if the /THRESHOLD MAP/ packet contains a state FIPS code for California (06000) or the specific Sonoma County FIPS code (06097), the assignment from the /THRESHOLD MAP/ packet will be used. Exhibit 3-60 shows an example of the threshold mask file.

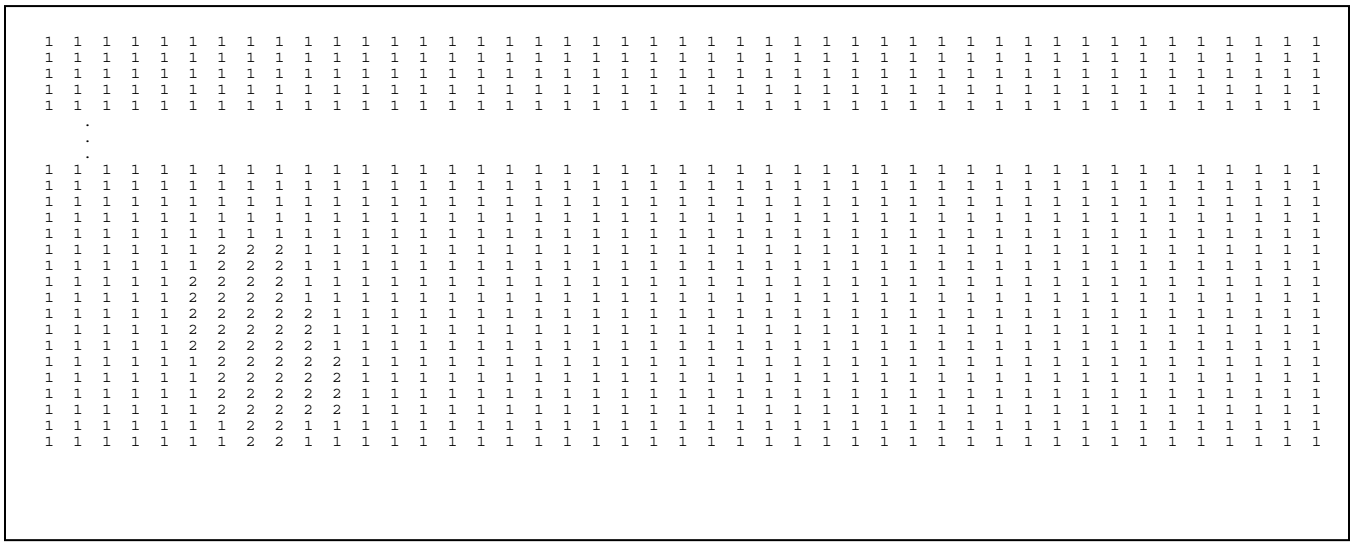

Exhibit 3-60. Sample Threshold Mask File.

The Static Sources File

As discussed above, this is an optional file designed to accelerate the procesing of large multi-day inventories that contain sources that have day-specific emission rates. This file is a CAMx-ready point source emissions file and in most cases will be generated by a previous run of the PIGEMS module. This file is designed to contain the sources in the inventory that do not vary by day. It is used as an initial file, to which additional sources will be appended.

Output Files

PIGEMS Message File

The PIGEMS module generates a standard EPS3 message file. This file contains a history of the run, including the date and time of the execution, a list of input and output files used and the user parameters specified in the USERIN file. The message file also contains one of two tables for QA purposes. The first table contains a count of the stacks in each input stack definition file and the second table contains the emissions totals for the stacks processed, also separated by input file. In each case, the information is reported by the following categories:

1. Total number of stacks.
2. Number of stacks that were flagged for PiG treatment.
3. Number of stacks identified for co-location.
4. Number of stacks combined using user definition.
5. Number of stacks in which the mask file determined the threshold code.

When running the PIGEMS module to produce a master stack list, only the table of stack counts will be produced. Alternatively, only the table of emissions totals will be produced when running the PIGEMS module to produce a model-ready emissions file. Exhibit 3-61 shows a PIGEMS message file.

```
                EPS3 PIGEMS module v. 1.2 Aug 2005      07/25/07 18:43:38

            Input Files
         USERIN file                    :../inputs/userin.test_problem.wkd
         Input master stack list file   :../emiss/stkhdr.pts.test_problem.wkd
         Input stack emissions files
         Emiss file 001                 :../emiss/elvems.pts.test_problem.wkd
         Threshold mask file            : Not provided.

            Output Files
         Using Input Master stack list : No output master stack list will be
created.
         Emissions file                 :../emiss/ptsrce.test_problem.wkd.bin

                   EPS3 PIGEMS module v. 1.2 Aug 2005      07/25/07 18:43:38

         File note                      :CAMx Modeling of Tx regional, 9/13/99
         Episode date (Calander)        :990913
         Episode date (Julian)          :99256
         Beginning hour                 :    0
         Ending hour                    :   24
         Grid origin (km)               : (  -108.000, -1584.000)
         UTM zone                       :    0
         Grid cell width (km)           : (    12.000,    12.000)
         Number of cells                : (  135,  138)

         Colocation dist (m)            :    0.00000


         Threshold Code Values read from USERIN file.
            Code   Species   Level (tons/day)


         Threshold Code Mappings read from USERIN file.
            FIPS Code   Threshold Code

              Table of Emissions Processed


      Input File          NOX          CO          VOC
 ------------------------------------------------------------
 Emiss file 001         2084.4503    852.5801    154.3051
 ------------------------------------------------------------
 Total                  2084.4502    852.5801    154.3051
```

Exhibit 3-61. Example of a PIGEMS message file.

Output Master Stack List File

This filename must be supplied when running the PIGEMS module to produce a master stack list. The PIGEMS module decides whether it should create a master stack list by looking at the filename for the input master stack list. If this filename is a blank string it is assumed that a master stack list is to be produced, in which case the line in the job script containing the output master stack list must contain a valid filename. However, if a valid filename is supplied for the input master stack list, it is assumed that a model-ready emissions file is to be produced and the line containing the output master stack list filename is ignored. The format of the output master stack list is the same as the stack definition files produced by the PSTPNT module.

Output Emissions File

This file is a binary model-ready point source emissions file containing a single day of hourly emissions for each of the stacks in the inventory. It is produced when running the PIGEMS module where an input master stack list is provided. When running the PIGEMS module to produce a master stack list, .i.e. the input master stack list file is blank and the output master stack list file is valid, this file is ignored.