United States
Environmental Protection
Agency

Office of Air Quality
Planning and Standards
Research Triangle Park, NC 27711

EPA-453/R-94-058B
July 1994

AIR

EPA Toxic Modeling System Short-Term (TOXST)

User's Guide: Volume II

TOXIC MODELING SYSTEM SHORT-TERM (TOXST)
USER'S GUIDE

VOLUME II
CASE STUDY AND SOURCE CODE

July 30, 1994

Prepared for:

U.S. Environmental Protection Agency
Office of Air Quality Planning and Standards
Research Triangle Park, NC 27711

DISCLAIMER

This document has been reviewed by the Office of Air Quality Planning and Standards, U.S. Environmental Protection Agency, and approved for publication. Approval does not signify that the contents necessarily reflect the view and policies of the U.S. Environmental Protection Agency, nor does mention of trade names or commercial products constitute endorsement or recommendation for use.

ACKNOWLEDGMENTS

This document was originally based on the document, Integrated Toxic Expected Exceedance (INTOXX) Model User's Guide, written by T. E. Stoeckenius and J. R. Pehling of Systems Applications International, San Rafael, California. That work was funded by the U. S. Environmental Protection Agency (EPA) under Contract 68D90006. A revision was performed in 1992 by J.S. Dunn of Research Triangle Institute, Research Triangle Park, NC, which was also funded by the EPA under Contract 2D1482NALX. During the revision made by Research Triangle Institute the INTOXX model was renamed TOXST. Subsequent modifications have been the result of efforts in 1993 by T. Grosch, D. Hlinka, and D. Sullivan of Sullivan Environmental Consulting, Alexandria, VA under EPA contract 68D20189. The development of this modeling system has been supervised by Dr. David E. Guinnup of the EPA.

Appendix A

Case Study

A.1. Case Study

A case study was conducted to demonstrate the application of the TOXST model at a representative chemical manufacturing facility (Sullivan, Grosch and Hlinka, 1993). The process area was one small part of a major organic chemical manufacturing facility located somewhere in the U.S.. To protect the confidentiality of the facility, the specific name of the plant and specific components of the process area will not be described. In addition, receptor coordinates have been changed.

A.2. Pollutants / Health Thresholds

The actual emissions from this facility's process area include four common chemicals. For the purposes of this case study, the following pollutants were evaluated: ethyl acetate, acetaldehyde, acetic acid, and styrene. As an example of state requirements, the Maryland acceptable thresholds will be used, which set a compliance level for one-hour concentrations as $1/100^{th}$ of the short-term emission limits (STELS), and $1/100^{th}$ of the 8-hour Threshold Limit Values (TLV) for 8-hour averages.

Table A-1 summarizes the Maryland acceptable thresholds, and the ratio of the TLV for acetaldehyde over the other chemical's TLVs (used subsequently for pre-processing emissions). Since a wide range of factors can be used to convert STELS and TLVs to state ambient guideline values, the exceedance displays shown in this report are shown relative to factors of $1/100^{th}$ and $1/200^{th}$ of the applicable STEL or TLV value.

A.3. Facility Process Area Emissions

The selected process area represents a typical industrial application of TOXST. This operation is comprised of two major units: (1) a dedicated unit that produces intermediate products that supply the main production unit, and (2) a production unit that is used to produce a variety of products on a campaign basis. Figure A-1 shows a simplified sketch of the two units relative to the entire plant and the plant boundaries.

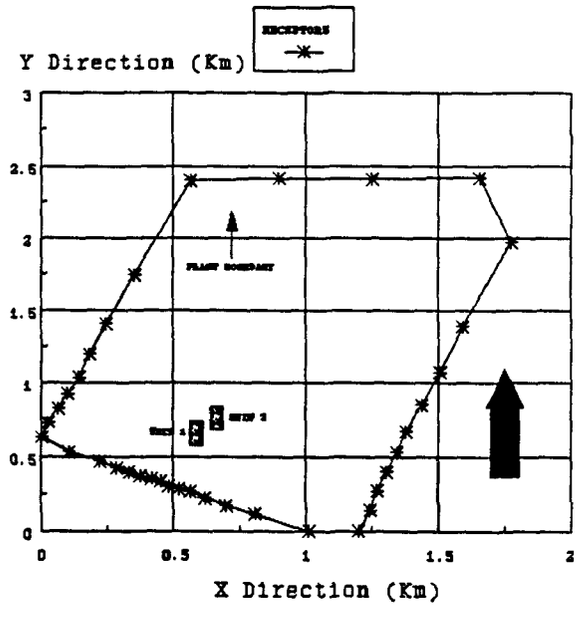
TABLE A-1

TLVs, STELs, 8-Hour and 1-Hour Concentration Thresholds For Each Pollutant

Pollutant	TLV ($\mu g/m^3$)	STEL ($\mu g/m^3$)	8-Hour Thresholds ($\mu g/m^3$)	1-Hour Thresholds ($\mu g/m^3$)	8-Hour Acetaldehyde Threshold/ Pollutant Threshold	1-Hour Acetaldehyde Threshold/ Pollutant Threshold
ETHYL ACETATE	1,400,000	N/A	14,000	N/A	0.13	N/A
ACETALDE HYDE	180,000	270,000	1,800	2,700	1.00	1.00
ACETIC ACID	25,000	N/A	250	N/A	7.20	N/A
STYRENE	215,000	425,000	2,150	4,250	0.84	0.64

FIGURE A-1

LOCATION OF PROCESS UNITS AND MODEL RECEPTORS ALONG PLANT BOUNDARY



This figure also shows the locations of the 36 boundary receptors that were used in the modeling analysis.

Both units are typically operated on a three-month on and one-month off cycle[1], with nearly continuous emissions for some sources, and also batch emissions associated with both units. The dedicated unit (Unit 1) that supplies the intermediate products always produces the same intermediates for use in Unit 2. The operation of the production unit (Unit 2), on the other hand, is varied to produce the range of chemical products that are manufactured by this process area.

The operation of this process area produces emissions from the following major categories: (1) production equipment vents, (2) breathing and working loss from chemical storage tanks, (3) emissions during cleaning of tanks (since many tanks are not dedicated to the storage of only one chemical), and (4) infrequent chemical spills during transfer operations. Some simplifications were made to minimize the time required for the engineering staff of the case study plant to define emissions data. Data were gathered based on a two-day trip to the plant. For a study to support a permit, it would be projected that 1-2 man-weeks of engineering support would be needed to gather the additional information required to define emissions data beyond the resolution typically required for a routine ISCST2 model application study. Each of the four source categories is described below:

Production Vents - Both Units 1 and 2 contain production vents that emit a variety of chemicals, some on a continuous basis, and others on an infrequent basis, such as operating for a one week campaign, 4-8 times per year. Sources that emit through the vents include stills associated with both units. Refer to Table A-2 for a listing of the operational characteristics of each vent modeled in this study, including: (1) the annual frequency of releases, (2) the duration of operation each batch cycle, and (3) emission rates for all four pollutants.

Breathing and Working Losses from Storage Tanks - Data were available for chemical storage for Unit 2 only, therefore, this case study was defined to exclude storage emissions from Unit 1. Data were available to describe the number of storage cycles per year for each storage tank. Refer to Table A-2 for a summary of the number of storage cycles and the emissions from each tank. It was assumed for this example that all breathing losses occur during the daytime (when tanks are relatively warm), and that working losses occur throughout the diurnal cycle.

---

[1] For simplicity, it was assumed that the continuous sources in Unit 1 operate three months on and one month off, and that intermittent sources could "turn on" at any time. For a regulatory application, however, the modeling could be refined by doing four separate model runs, i.e. one model run to represent each three month period for the continuous sources, and one to represent the intermittent sources. The results of each model run could then be merged during post processing to estimate exceedances.

Tank Cleaning - Tanks are steam cleaned for approximately four hours whenever a different chemical will be stored. It was assumed that 3 gallons (approximately 11.4 liters) of residual chemical on the sides of the tanks and the chemical potentially left in the bottom would be emitted to the atmosphere.[2] Emissions during the first hour are relatively low because the temperature within the tanks will not reach maximum temperature. Most emissions would be released during the second hour when the cleaning process reaches full temperature. The following was assumed:

Hour	% Emissions Released
1	10
2	80
3	10
4	0

Spills - While spills during transfer operations are rare, this source also was considered as part of the case study. To support an actual facility, the operational records could be used, or for a new facility data representative of a similar facility could be used, to define the probability of a range of spills. **For illustrative purposes only**, the following were assumed:

- Two very small spills (5 gallons each) every year.

- A small spill (10 gallons) every year.

- The duration of emissions for each spill was assumed to be 2 hours.

- It was conservatively assumed that a 3-foot diameter pool was formed during the 5 gallon spills, and a 5-foot diameter pool during the 10 gallon spill.

- It was assumed that 50 percent of the spill volatilized during the period specified (USEPA, 1989).

- It was assumed in this illustration that acetaldehyde was spilled in each of the two spill scenarios.

A.3.1 Composite Emissions

For the composite chemical model runs, it was assumed that the health effects were interactive (which is consistent with the eye and respiratory irritation effects for each of these four chemicals). The emissions for each pollutant were pre-processed to show emissions relative to

---

[2] The storage capacity of the tanks ranged from 1 X 10^6 to 2 X 10^6 liters each.

acetaldehyde, consistent with Section 2.3.4 of the TOXST User's Guide. In particular, the composite annual (tons/year) emission rates relative to acetaldehyde emissions were calculated first, based on the procedure described in Section 2.3.4 of the text[3]. The annual pollutant emission rates for each source were factored by the appropriate 8-hour and 1-hour acceptable threshold ratios shown in Table A-1, and then were factored to represent each emission category (start-up, routine operation, and tank cleaning). The factors for each emission category were supplied by engineers at the example plant, and are not displayed in this guide.

The resulting composite emission rates presented in Tables A-3 and A-4 were used for the modeling runs of the composite chemicals. The processed emission rates are substantially higher for the 8-hour averaging period, as shown in Table A-4, since there are more chemicals with 8-hr TLVs than those with applicable STELS.

---

[3] For the interactive mode, the procedure to develop a composite emission rate is based on consideration of the emission rate and the threshold for each species (See Section 2.3.4 of TOXST User's Guide). A frequency distribution of combined concentrations then can be displayed, since multiple thresholds can be used in TOXX if the emissions are preprocessed in this manner.

TABLE A-2

Hours of Operation and Emission Rates For Sources of Example Facility

Source	Number of Times Source On Per Year	Hours of Operation Each Cycle	Ethyl Acetate Emission Rate (T/Y)	Acetalde-hyde Emission Rate (T/Y)	Acetic Acid Emission Rate (T/Y)	Styrene Emission Rate (T/Y)
PROCESS UNIT 1						
SRC 1	3	2184	22.680	4.860	0.000	0.000
SRC 2	3	2184	1.422	0.450	0.010	0.000
SRC 3	30	144	9.564	13.042	0.289	0.000
SRC 4	34	144	2.203	1.714	0.000	0.000
PROCESS UNIT 2						
SRC 5	3	2184	0.269	0.000	0.000	0.000
SRC 6	27	195	15.334	0.000	0.000	0.000
SRC 7	21	264	0.089	0.000	0.000	0.000
SRC 8	24	267	1.120	0.000	0.000	0.000
SRC 9	12	267	27.771	0.000	0.000	0.000
SRC 10	2	120	2.447	0.000	0.000	0.000
SRC 11	12	264	7.505	0.000	0.000	0.000
SRC 12	12	264	0.872	0.000	0.000	0.000
UNIT 2 STORAGE AREA						
SRC 13	4	66	0.105	0.000	0.051	0.063
SRC 14	4	66	0.105	0.000	0.051	0.063
SRC 15	4	66	0.105	0.000	0.051	0.063
SPILLS						
SRC 16 (5 Gallon)	2	2	0.000	0.016	0.000	0.000
SRC 17 (10 Gallon)	1	2	0.000	0.016	0.000	0.000

TABLE A-3

Hours of Operation and Emission Rates
For Sources of 1-Hour Composite Emissions

Source	Number of Times Source On Per Year	Hours of Operation Each Cycle	Start Up Emission Rate (G/S)	Routine Emission Rate (G/S)	Max. Tank Cleaning Emission Rate (G/S)	Yearly Emission Rate (T/Y)
PROCESS UNIT 1						
SRC 1	3	2184	0.189	0.189	N/A	4.86
SRC 2	3	2184	0.018	0.018	N/A	0.45
SRC 3	30	144	0.761	0.761	N/A	13.042
SRC 4	34	144	0.088	0.088	N/A	1.714
PROCESS UNIT 2						
SRC 5	3	2184	0.0	0.0	N/A	0.0
SRC 6	27	195	0.0	0.0	0.0	0.0
SRC 7	21	264	0.0	0.0	N/A	0.0
SRC 8	24	267	0.0	0.0	0.0	0.0
SRC 9	12	267	0.0	0.0	0.0	0.0
SRC 10	2	120	0.0	0.0	N/A	0.0
SRC 11	12	264	0.0	0.0	N/A	0.0
SRC 12	12	264	0.0	0.0	N/A	0.0
UNIT 2 STORAGE AREA						
SRC 13	4	66	0.238	0.004	0.870	0.04
SRC 14	4	66	0.238	0.004	0.870	0.04
SRC 15	4	66	0.238	0.004	0.870	0.04
SPILLS						
SRC 16 (5 Gallon)	2	2	1.5531	0.5177	N/A	0.016
SRC 17 (10 Gallon)	1	2	3.1062	1.0354	N/A	0.016

N/A = Not Applicable

TABLE A-4

Hours of Operation and Emission Rates For Sources of 8-Hour Composite Emissions

Source	Number of Times Source On Per Year	Hours of Operation Each Cycle	Start Up Emission Rate (G/S)	Routine Emission Rate (G/S)	Max. Tank Cleaning Emission Rate (G/S)	Yearly Emission Rate (T/Y)
PROCESS UNIT 1						
SRC 1	3	2184	0.304	0.304	N/A	7.808
SRC 2	3	2184	0.027	0.027	N/A	0.707
SRC 3	30	144	0.955	0.955	N/A	16.366
SRC 4	34	144	0.103	0.103	N/A	2.0
PROCESS UNIT 2						
SRC 5	3	2184	0.001	0.001	N/A	0.035
SRC 6	27	195	0.115	0.092	0.762	1.993
SRC 7	21	264	0.0011	0.001	N/A	0.012
SRC 8	24	267	0.044	0.040	0.431	1.06
SRC 9	12	267	0.356	0.285	0.431	3.61
SRC 10	2	120	0.334	0.334	N/A	0.318
SRC 11	12	264	0.081	0.078	N/A	0.976
SRC 12	12	264	0.010	0.009	N/A	0.113
UNIT 2 STORAGE AREA						
SRC 13	4	66	2.555	0.017	9.356	0.434
SRC 14	4	66	2.555	0.017	9.356	0.434
SRC 15	4	66	2.555	0.017	9.356	0.434
SPILLS						
SRC 16 (5 Gallon)	2	2	1.5531	0.5177	N/A	0.016
SRC 17 (10 Gallon)	1	2	3.1062	1.0354	N/A	0.016

N/A = Not Applicable

A.4. Model Input Files

Appendix A.1 documents the input files used for the ISCST2 and TOXST model runs. A total of 17 source groups were modeled in both ISCST2 and TOXST. Five years of hourly meteorological data (1984-1985 and 1987-1989) were used to support the analysis, however, only the 1984 ISCST2 input and output files are shown in the Appendix A example. The TOXX results are based only on 1984 and 1985.

A.5. Results of Model Runs

The results in this section are shown for the four chemical (composite) screening-level and TOXST model runs. The following summaries are based on assuming additive effects for the four chemicals, using pre-processing of emission rates as shown in Section 2.3.4 of the text.

A.5.1 **Screening-Level Model Runs**

Typically, a conservative, screening-level analysis should be done prior to using the TOXST model, since more extensive emissions and dispersion modeling analysis are necessary to support TOXST applications. TOXST should then be used only if a source fails the screen. Table A-5 compares the number of exceedances for the base case using the tiered modeling approach developed by EPA (USEPA, 1992). Maximum 1-hour composite emission rates were used in the initial screen. Each of the two screening approaches shows predicted maximum 1-hour concentrations to be above the guideline values.

A.5.2 **TOXST Model Runs**

Since the EPA Tier 1 and Tier 2 screening approaches (see Section A.5.1) show concentrations of the composite emissions that exceed the threshold concentration for the 1-hour averaging period, more detailed modeling analysis through TOXST was justified in order to demonstrate compliance.

TABLE A-5

Two-Tier Modeling Approach Results
For The Case Study Composite Emission Sources

	Source	Modeled Source Height (m)	Modeled Source Width (m)	Closest Distance to Fence (m)	Max 1 hr Emission Rate (G/S)	Tier 1# Normalized Conc. ($\mu g/m^3$)	Tier 1 Predicted Conc. ($\mu g/m^3$)	Tier 2## Normalized Conc ($\mu g/m^3$)	Tier 2 Predicted Conc. ($\mu g/m^3$)
PROCESS UNIT 1	SRC 1	1.0	61.2	200	0.189	5,590	1,057	3,617	684
	SRC 2	1.0	61.2	200	0.018	5,590	101	3,617	65
	SRC 3	1.0	61.2	200	0.761	5,590	4,254*	3,617	2,753*
	SRC 4	1.0	61.2	200	0.088	5,590	492	3,617	318
PROCESS UNIT 2	SRC 5	1.0	45.8	300	0.000	5,590	0	2,691	0
	SRC 6	1.0	45.8	300	0.000	5,590	0	2,691	0
	SRC 7	1.0	45.8	300	0.000	5,590	0	2,691	0
	SRC 8	1.0	45.8	300	0.000	5,590	0	2,691	0
	SRC 9	1.0	45.8	300	0.000	5,590	0	2,691	0
	SRC 10	1.0	45.8	300	0.000	5,590	0	2,691	0
	SRC 11	1.0	45.8	300	0.000	5,590	0	2,691	0
	SRC 12	1.0	45.8	300	0.000	5,590	0	2,691	0
UNIT 2 STORAGE	SRC 13	1.0	30.0	300	0.870	5,590	4,863*	3,226	2,807*
	SRC 14	1.0	30.0	300	0.870	5,590	4,863*	3,226	2,807*
	SRC 15	1.0	30.0	300	0.870	5,590	4,863*	3,226	2,807*
SPILLS	SRC 16 (5G)	1.0	0.9	300	1.0351	8,070	8,356*	4,900	5,073*
	SRC 17 (10G)	1.0	1.5	300	2.071	8,070	16711*	4,857	10,058*
	Total with out Spills						20,493*		12,239*
	Total with spills						45,560*		27,371*

\# From Table 2 (page 16) of USEPA, 1992.
From SCREEN 1.1 Model Runs.
* Exceeds 1-Hour Acetaldehyde Threshold Concentration of 2,700 $\mu g/m^3$.

The number of exceedances of the 1- and 8-hour thresholds for the composite emissions are shown for each boundary receptor in Tables A-6 through A-9, based on four modeling approaches:

I. Maximum emissions rates / ISCST2 only (Table A-6).

II. Average emission rates / ISCST2 only (Table A-7).

III. Previous TOXST (no emissions distributions or batch treatments (see Table A-8). The previous version of TOXST could not simulate batch operations or multiple hourly averaging. For this case study, emission rates were set at the maximum emission rate for each batch and non-batch source to simulate the treatment in the previous version of TOXST. As a means of demonstrating the benefit of the new features in TOXST relative to the preceding version of the model, these model runs were made using the new version of TOXST, but simulating the previous version by maintaining peak emissions rates throughout each batch and non-batch operation. This assumption is quite conservative for the 8-hour treatments.

IV. Current version of TOXST (Table A-9).

Figures A-2 and A-3 graphically summarize the data in Tables A-6 through A-9, based on the most affected receptors. As shown, TOXST provides the flexibility to reduce the degree of conservatism through using emission distributions, batch operations, and the multiple-hour features.

TABLE A-6

Average Exceedances per Year
Across All Receptors* and All Years
For Maximum Composite Emission Rates (ISCST2)

	Averaging Period Thresholds			
	1-Hour	1-Hour	8-hour	8-hour
SOURCES	1,350 $\mu g/m^3$	2,700 $\mu g/m^3$	900 $\mu g/m^3$	1,800 $\mu g/m^3$
1	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00
5	-	-	0.00	0.00
6	-	-	0.00	0.00
7	-	-	0.00	0.00
8	-	-	0.00	0.00
9	-	-	0.00	0.00
10	-	-	0.00	0.00
11	-	-	0.00	0.00
12	-	-	0.00	0.00
13	0.00	0.00	26.80	3.40
14	0.00	0.00	25.20	2.20
15	0.00	0.00	22.60	2.60
16	48.00	21.00	1.00	0.00
17	84.80	48.00	14.40	1.00
ALL SOURCES COMBINED	179.40	97.20	137.60	76.20

* Worst-Case Receptor Selected

TABLE A-7

Average Exceedances per Year
Across All Receptors* and All Years
For Average Composite Emission Rates (ISCST2)

	Averaging Period Thresholds			
	1-Hour	1-Hour	8-hour	8-hour
SOURCES	1,350 $\mu g/m^3$	2,700 $\mu g/m^3$	900 $\mu g/m^3$	1,800 $\mu g/m^3$
1	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00
5	-	-	0.00	0.00
6	-	-	0.00	0.00
7	-	-	0.00	0.00
8	-	-	0.00	0.00
9	-	-	0.00	0.00
10	-	-	0.00	0.00
11	-	-	0.00	0.00
12	-	-	0.00	0.00
13	0.00	0.00	0.00	0.00
14	0.00	0.00	0.00	0.00
15	0.00	0.00	0.00	0.00
16	0.00	0.00	0.00	0.00
17	0.00	0.00	0.00	0.00
ALL SOURCES COMBINED	0.00	0.00	0.00	0.00

* Worst-Case Receptor Selected

TABLE A-8

Average Exceedances per Year
Across All Receptors* and All Years
For Maximum Composite Emission Rates (Old TOXST)

	Averaging Period Thresholds			
	1-Hour	1-Hour	8-hour	8-hour
SOURCES	1,350 $\mu g/m^3$	2,700 $\mu g/m^3$	900 $\mu g/m^3$	1,800 $\mu g/m^3$
1	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00
5	-	-	0.00	0.00
6	-	-	0.00	0.00
7	-	-	0.00	0.00
8	-	-	0.00	0.00
9	-	-	0.00	0.00
10	-	-	0.00	0.00
11	-	-	0.00	0.00
12	-	-	0.00	0.00
13	0.00	0.00	0.88	0.10
14	0.00	0.00	0.80	0.07
15	0.00	0.00	0.72	0.09
16	0.03	0.01	0.00	0.00
17	0.03	0.02	0.00	0.00
ALL SOURCES COMBINED	0.48	0.03	3.12	0.49

* Worst-Case Receptor Selected

TABLE A-9

Average Exceedances per Year
Across All Receptors* and All Years
For Maximum Composite Emission Rates (Revised TOXST)

	Averaging Period Thresholds			
	1-Hour	1-Hour	8-hour	8-hour
SOURCES	1,350 $\mu g/m^3$	2,700 $\mu g/m^3$	900 $\mu g/m^3$	1,800 $\mu g/m^3$
1	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00
5	-	-	0.00	0.00
6	-	-	0.00	0.00
7	-	-	0.00	0.00
8	-	-	0.00	0.00
9	-	-	0.00	0.00
10	-	-	0.00	0.00
11	-	-	0.00	0.00
12	-	-	0.00	0.00
13	0.00	0.00	0.06	0.00
14	0.00	0.00	0.06	0.00
15	0.00	0.00	0.06	0.00
16	0.02	0.01	0.00	0.00
17	0.01	0.01	0.00	0.00
ALL SOURCES COMBINED	0.03	0.01	0.23	0.01

* Worst-Case Receptor Selected

FIGURE A-2

COMPARISON OF PREDICTED NUMBER OF EXCEEDANCES
FOR MAXIMUM AND AVERAGE EMISSION RATE SCENARIOS
FOR 8-HOUR AVERAGING PERIOD

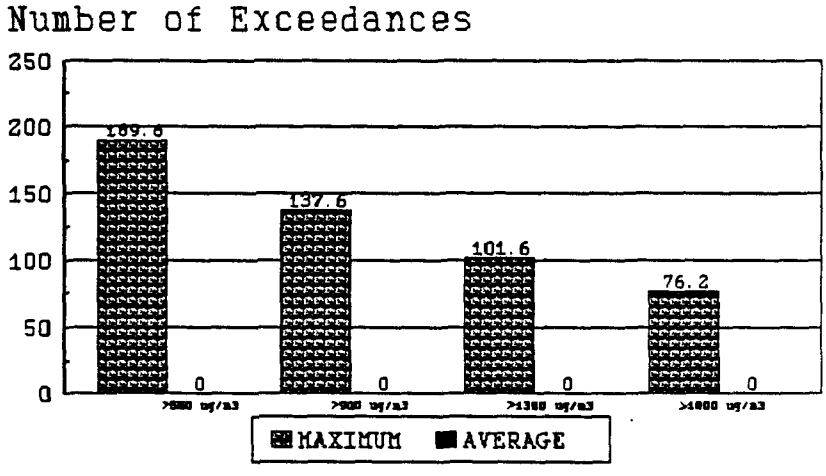
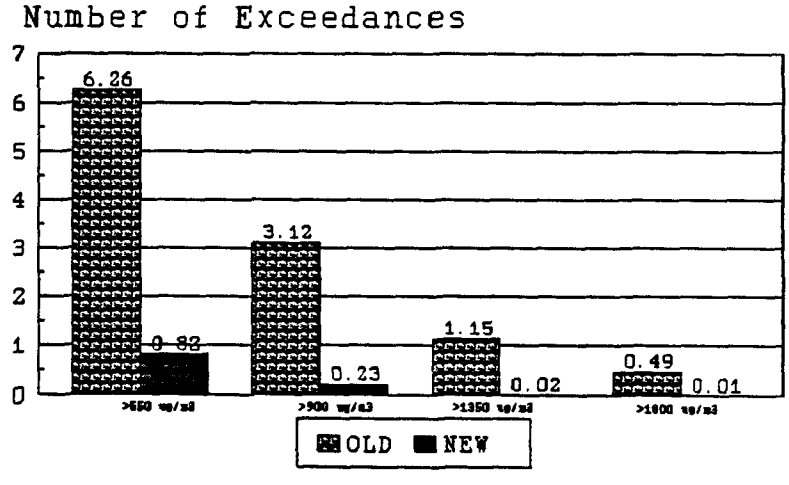


FIGURE A-3

COMPARISON OF PREDICTED NUMBER OF EXCEEDANCES
FOR OLD (PREVIOUS) AND NEW (REFINED) TOXST
FOR 8-HOUR AVERAGING PERIOD



A.6. Hypothetical Scenarios to Highlight Specific TOXST Features

The results that were presented in Sections A.1 through A.5 were based on the actual operations of the case study facility, with appropriate simplifications. Results in this section are presented for hypothetical scenarios in order to more fully demonstrate the expanded features of TOXST. For these hypothetical cases, 1-hour averaging periods were used for the first example shown below, and 8-hour averaging is used for the subsequent two examples:

1. Assuming no tank cleaning or transfer during nighttime periods.

2. Assuming ramp-down emissions (rather than essentially uniform emissions prior to cleaning of tanks) for the three largest batch operations in Unit #2, i.e. sources 6 and 9.

3.Assuming a hypothetical operation with two mutually exclusive groups of sources, i.e. 6 and 9.

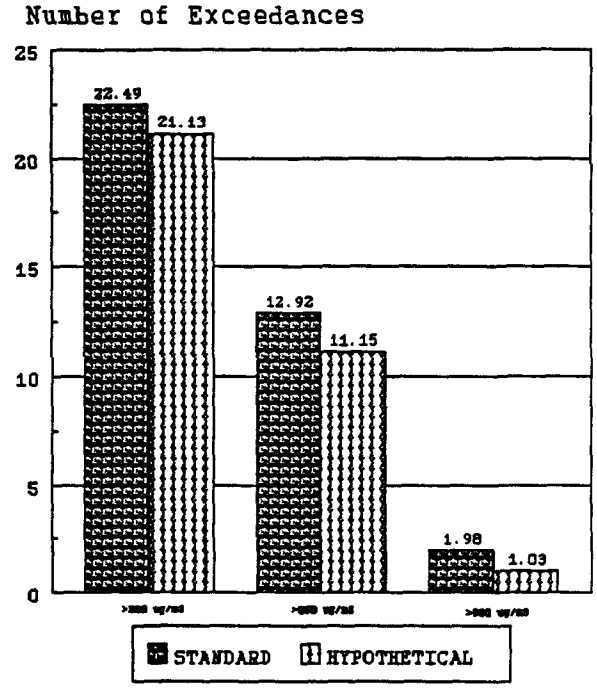
A.6.1 No Tank Cleaning or Transfer During Nighttime Periods

The case study did not provide the opportunity to fully display the feature of TOXST to account for different emissions on a daytime and nighttime basis. This hypothetical example better displays the diurnal features of TOXST by using a potential mitigation option of restricting certain operations to the daytime periods. Figure A-4 compares the standard case (with full diurnal coverage) to the exceedances that would occur if there was no tank cleaning or transfers during the nighttime periods (when adverse dispersion / transport conditions more typically occur). This scenario should not be interpreted to suggest that such restrictions in operation would necessarily be feasible at the case study facility.

The results of this hypothetical run, graphically shown in Figure A-4, indicate that the number of exceedances of the 900 $\mu g/m^3$ threshold are essentially reduced by a factor of two if it is assumed that tank cleaning and tank transfer operations are performed only during the daylight hours.

FIGURE A-4

COMPARISON OF MODEL RESULTS FOR HYPOTHETICAL SCENARIO
OF NO TANK CLEANING OR TRANSFER EMISSIONS
DURING NIGHTTIME HOURS TO STANDARD MODEL RUN
FOR 1-HOUR AVERAGING PERIOD



6.2 Ramp-Down for Major Batch Operations In Unit #2

The objective of this hypothetical scenario is to more fully display the enhanced benefits of the new version of TOXST when there are large difference in emission rates throughout batch operations (which was not the case for the process operations at this facility). An additional model run was made where emissions at four sources were assumed to sharply ramp down after initiation of the batch operation. The following provides the differing emission rates and durations for the sources chosen for this scenario as compared to the standard model run.

EMISSION RATES AND DURATIONS FOR SOURCES 1-2
USED IN THE 8-HOUR RAMP-DOWN SCENARIO

	SOURCE 1	SOURCE 2
STANDARD RUN		
1-2184 HOURS	0.882 G/S	0.055 G/S
RAMP-DOWN SCENARIO		
HOURS 1-10	15.200 G/S	1.350 G/S
HOURS 11-60	6.080 G/S	0.540 G/S
HOURS 61-560	0.304 G/S	0.027 G/S
HOURS 561-1184	0.061 G/S	0.006 G/S
HOURS 1185-2184	0.018 G/S	0.002 G/S

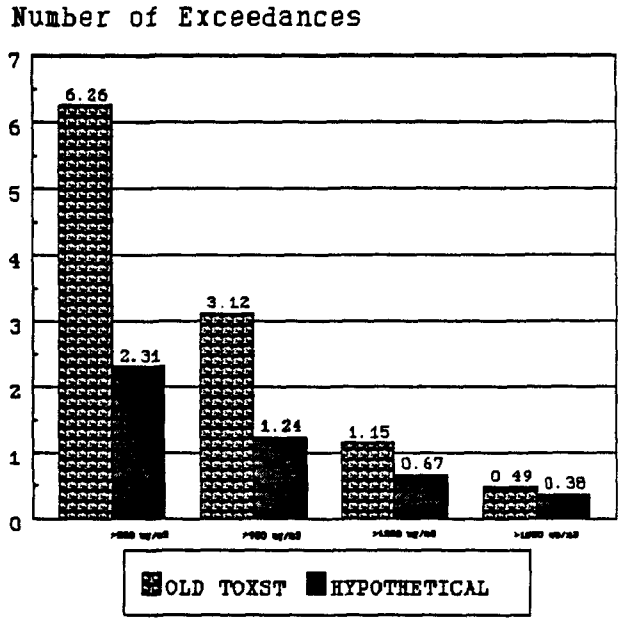
EMISSION RATES AND DURATIONS FOR SOURCES 3-4
USED IN THE 8-HOUR RAMP-DOWN SCENARIO

	SOURCE 3	SOURCE 4
STANDARD RUN		
1-144 HOURS	0.558 G/S	0.113 G/S
RAMP-DOWN SCENARIO		
HOURS 1-5	19.100 G/S	2.060 G/S
HOURS 6-25	1.190 G/S	0.206 G/S
HOURS 26-60	0.095 G/S	0.011 G/S
HOURS 61-100	0.010 G/S	0.001 G/S
HOURS 101-144	0.003 G/S	0.0002 G/S

The results of the hypothetical scenario for the 8-hour ramp-down emissions in comparison to the more simplified treatment of assuming uniform emissions rates during the batch operations are presented in Figure A-5. The results indicate that TOXST estimates of the number of exceedances are nearly two-to-three times lower than the more simplified approach of using constant, conservative emission rates during a batch operation.

FIGURE A-5

COMPARISON OF MODEL RESULTS FOR SCENARIO
OF RAMP-DOWN EMISSIONS
TO OLD TOXST MODEL RUN
FOR 8-HOUR AVERAGING PERIOD



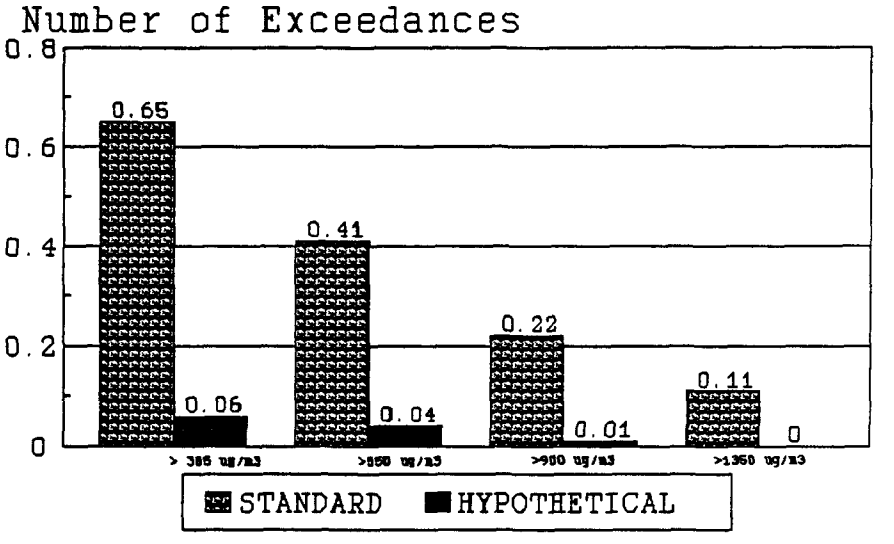
A.6.3 **Mutual Exclusion Scenario**

TOXST is structured to provide a mutual exclusion option, where if this feature is selected, only one source can be in operation at any given time. Since the case study facility is not operated in a manner in which this model option could be displayed, a simplified scenario was established to show this feature. Three of the source groups were selected from Unit #2 as a demonstration of this feature, i.e. 3601, 3603 and 3604. As an illustration, it was assumed that these were the only sources associated with this operation.

The results of the comparison of the hypothetical model run using the mutual exclusion option to the standard model run without mutual exclusion are presented in Figure A-6.
A much lower number of exceedances is predicted when the mutual exclusion option is used, even though in this particular hypothetical example the impacts were all shown to be below the thresholds.

FIGURE A-6

COMPARISON OF MODEL RESULTS FOR SCENARIO
OF MUTUAL EXCLUSION OPTION FOR SELECTED SOURCES
TO STANDARD MODEL RUN
FOR 8-HOUR AVERAGING PERIOD



A.7. Sensitivity to Number of Simulations

Limited testing was done based on a set of hypothetical sources to evaluate the sensitivity of the model output to the number of simulations. For this example, five sources, each with a 10 g/s release were defined. Each source had the same source dimensions and were located in the same geographic location. Each source, when "turned on" operated for 1 hour, however, the probability of each source "turning on" varied as is shown in the table on the next page.

Figures A-7 and A-8 display the percent error as a function of frequency of "turning on" per year and the number of simulations. "Percent error" for these purposes is defined

Source #	Number of Times Per Year Source Turns On
1	5
2	10
3	100

4	500
5	1999

relative to the "true" simulation which uses the maximum number of simulation years (1999). Based on these illustrations, sources that emit 5-25 percent of the time could be represented by 100 simulations per year with the difference between 100 and 1999 simulations being less than 10 percent. For less frequent emitters, i.e. those that emit less than 5 percent per year, a minimum of 200 simulations would be recommended based on these comparisons. For sources that emit on a very infrequent basis, such as one time per year or less, it would be expected, based on these initial tests, that more simulations would be needed to approach an accurate representation. For example, perhaps a minimum of 500 simulations should be used in that case. It would be recommended for applications with low probability sources that testing be performed to confirm that a sufficient number of simulations were done to reasonably approach the average exceedances that would be computed for a large number of simulations, e.g. 1,999.

FIGURE A-7

SENSITIVITY TO NUMBER OF SIMULATIONS
1-HOUR AVERAGING

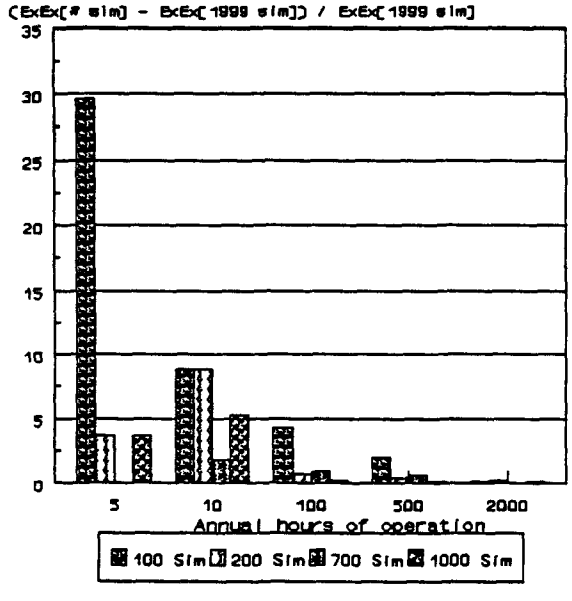
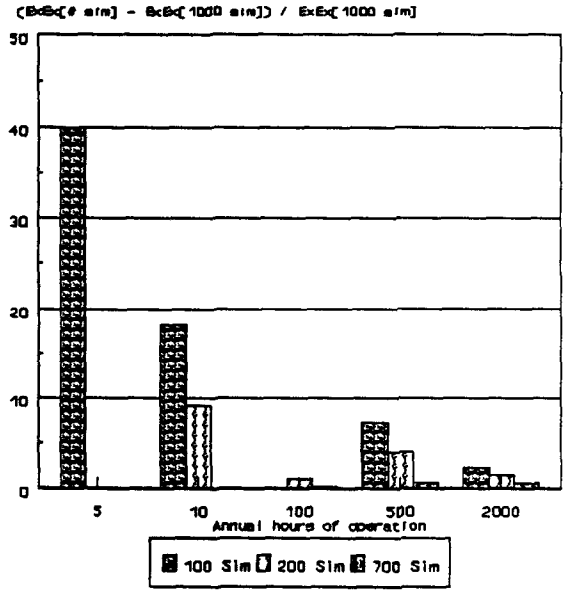


FIGURE A-8

SENSITIVITY TO NUMBER OF SIMULATIONS
8-HOUR AVERAGING



A.8. Conclusions

This case study shows that the more conservative approaches of using Tier 1 or 2, or worst

case emissions with a full year of meteorological data, show exceedances of the TLV/200 and TLV/100 for the mix of pollutants evaluated. TOXST was shown to substantially reduce the degree of conservatism relative to more simplistic treatments.

The greatest benefits of the enhanced version of TOXST for this case study were shown to be for the multiple-hour scenarios (e.g. 8-hour averaging), and for sources that have highly variable emission rates, e.g. batch operations, and sources that have significant differences in operating characteristics on a daytime and nighttime basis.

For example, the simulation of the previous version of TOXST showed 3.12 exceedances per year of the TLV / 200, while the current version showed 0.23.

The following additional observations can be made based on the case study:

- It is important to recognize that in many cases it would not be necessary to use the full features of the TOXST in order to determine compliance of a facility with a specific short-term ambient standard. The first steps should be to use the Tier 1 and 2 screening approaches, and then only if necessary, evaluate compliance based on assuming maximum emission rates along with a full year (or preferably five years) of meteorological data. If compliance is demonstrated on this basis, there again should be no reason to refine the emission terms. If further analysis is needed, however, it would be recommended that the full features of the enhanced version of TOXST be used where appropriate to more accurately represent the operating characteristics of the sources. Whenever the emissions variability can be characterized through engineering analysis and TOXST, more accurate estimates would be expected for peak exposures.

- A common question concerning the use of TOXST relates to the additional effort required to compile the model inputs that are needed for the analysis. There are two separate considerations. First, the ISCST2 modeling analysis that is needed to support TOXST is comparable to the analysis that would be required to support a routine (non-stochastic) assessment of peak exposures. Second, the emissions data needed to support TOXST are more involved than for a non-stochastic analysis. This is the area where additional analysis is required. The data that serve as the basis for the existing emissions inventory would need to be reevaluated to characterize the probabilities and emission rates of the batch operations and the emission distributions for non-batch sources. Based on this case study as an example, it would appear that approximately two-weeks of engineering support would be needed to compile source data of suitable resolution to support

an actual permit preparation for a process area of the magnitude of the case study.

- TOXST is sensitive to the number of simulations if sources are being modeled that emit on a very infrequent basis, e.g. less than five times per year. Based on preliminary testing, it would be recommended that a minimum of 200 simulations be used for sources that emit less than 25 percent of the year, and at least 500 simulations for sources than emit less than 5 times per year. Fewer simulations could be used if it could be demonstrated that the results are not adversely affected.

********** APPENDIX A Example Run **********

Example Input and Output Files for ISCST2 and TOXST
Using the 1-Hour Composite Emissions Run As An Example

********** Example ISCST2 Input File **********

```
CO STARTING
CO TITLEONE CASE STUDY / NORMALIZED EMISSIONS - 1984
CO MODELOPT CONC RURAL
CO AVERTIME 1
CO POLLUTID COMPOSITE
CO TERRHGTS FLAT
CO RUNORNOT RUN
CO FINISHED

SO STARTING
SO LOCATION SRC1   VOLUME   46375  818675
SO LOCATION SRC2   VOLUME   46375  818675
SO LOCATION SRC3   VOLUME   46375  818675
SO LOCATION SRC4   VOLUME   46375  818675
SO LOCATION SRC5   VOLUME   46450  818750
SO LOCATION SRC6A  VOLUME   46450  818750
SO LOCATION SRC6B  VOLUME   46450  818750
SO LOCATION SRC7   VOLUME   46450  818750
SO LOCATION SRC8   VOLUME   46450  818750
SO LOCATION SRC9   VOLUME   46450  818750
SO LOCATION SRC10  VOLUME   46450  818750
SO LOCATION SRC11  VOLUME   46450  818750
SO LOCATION SRC12  VOLUME   46450  818750
SO LOCATION SRC13  VOLUME   46465  818750
SO LOCATION SRC14  VOLUME   46465  818760
SO LOCATION SRC15  VOLUME   46465  818770
SO LOCATION SRC16    AREA   46465  818750
SO LOCATION SRC17    AREA   46465  818750
SO SRCPARAM SRC1     1.00 5.33 4.96 14.24
SO SRCPARAM SRC2     1.00 5.33 4.96 14.24
SO SRCPARAM SRC3     1.00 5.33 4.96 14.24
SO SRCPARAM SRC4     1.00 5.33 4.96 14.24
SO SRCPARAM SRC5     1.00 5.33 4.96 10.66
SO SRCPARAM SRC6A    0.56 5.33 4.96 10.66
SO SRCPARAM SRC6B    0.44 5.33 4.96 10.66
SO SRCPARAM SRC7     1.00 5.33 4.96 10.66
SO SRCPARAM SRC8     1.00 5.33 4.96 10.66
SO SRCPARAM SRC9     1.00 5.33 4.96 10.66
SO SRCPARAM SRC10    1.00 5.33 4.96 10.66
SO SRCPARAM SRC11    1.00 5.33 4.96 10.66
SO SRCPARAM SRC12    1.00 5.33 4.96 10.66
SO SRCPARAM SRC13    1.00 6.40 5.95 6.98
SO SRCPARAM SRC14    1.00 6.40 5.95 6.98
SO SRCPARAM SRC15    1.00 6.40 5.95 6.98
SO SRCPARAM SRC16      1.20 1.00 0.91
SO SRCPARAM SRC17      0.43 1.00 1.52
** SOURCE GROUPS
```

```
SO SRCGROUP SRC1  SRC1
SO SRCGROUP SRC2  SRC2
SO SRCGROUP SRC3  SRC3
SO SRCGROUP SRC4  SRC4
SO SRCGROUP SRC5  SRC5
SO SRCGROUP SRC6  SRC6A SRC6B
SO SRCGROUP SRC7  SRC7
SO SRCGROUP SRC8  SRC8
SO SRCGROUP SRC9  SRC9
SO SRCGROUP SRC10 SRC10
SO SRCGROUP SRC11 SRC11
SO SRCGROUP SRC12 SRC12
SO SRCGROUP SRC13 SRC13
SO SRCGROUP SRC14 SRC14
SO SRCGROUP SRC15 SRC15
SO SRCGROUP SRC16 SRC16
SO SRCGROUP SRC17 SRC17
SO FINISHED

RE STARTING
RE BOUNDARY SRC1 1910
RE BOUNDARY SRC1 2000
RE BOUNDARY SRC1 2180
RE BOUNDARY SRC1 1880
RE BOUNDARY SRC1 1340
RE BOUNDARY SRC1 1085
RE BOUNDARY SRC1 930
RE BOUNDARY SRC1 825
RE BOUNDARY SRC1 780
RE BOUNDARY SRC1 750
RE BOUNDARY SRC1 750
RE BOUNDARY SRC1 780
RE BOUNDARY SRC1 825
RE BOUNDARY SRC1 690
RE BOUNDARY SRC1 480
RE BOUNDARY SRC1 380
RE BOUNDARY SRC1 310
RE BOUNDARY SRC1 260
RE BOUNDARY SRC1 250
RE BOUNDARY SRC1 240
RE BOUNDARY SRC1 225
RE BOUNDARY SRC1 230
RE BOUNDARY SRC1 250
RE BOUNDARY SRC1 270
RE BOUNDARY SRC1 300
RE BOUNDARY SRC1 350
RE BOUNDARY SRC1 460
RE BOUNDARY SRC1 575
RE BOUNDARY SRC1 575
RE BOUNDARY SRC1 580
```

RE BOUNDARY SRC1 610
RE BOUNDARY SRC1 660
RE BOUNDARY SRC1 770
RE BOUNDARY SRC1 935
RE BOUNDARY SRC1 1230
RE BOUNDARY SRC1 1870
RE FINISHED

ME STARTING
ME INPUTFIL CMA84.BIN UNFORM
ME ANEMHGHT 6.1 METERS
ME SURFDATA 05879 1984
ME UAIRDATA 05879 1984
ME FINISHED

OU STARTING
OU RECTABLE 1 FIRST
OU TOXXFILE 1 2.07E-6 ISC8H84.TOX
OU FINISHED

********** Example ISCST2 Output File **********

```
CO STARTING
CO TITLEONE CASE STUDY / NORMALIZED EMISSIONS - 1984
CO MODELOPT CONC RURAL
CO AVERTIME 1
CO POLLUTID COMPOSITE
CO TERRHGTS FLAT
CO RUNORNOT RUN
CO FINISHED

SO STARTING
SO LOCATION SRC1   VOLUME   46375  818675
SO LOCATION SRC2   VOLUME   46375  818675
SO LOCATION SRC3   VOLUME   46375  818675
SO LOCATION SRC4   VOLUME   46375  818675
SO LOCATION SRC5   VOLUME   46450  818750
SO LOCATION SRC6A  VOLUME   46450  818750
SO LOCATION SRC6B  VOLUME   46450  818750
SO LOCATION SRC7   VOLUME   46450  818750
SO LOCATION SRC8   VOLUME   46450  818750
SO LOCATION SRC9   VOLUME   46450  818750
SO LOCATION SRC10  VOLUME   46450  818750
SO LOCATION SRC11  VOLUME   46450  818750
SO LOCATION SRC12  VOLUME   46450  818750
SO LOCATION SRC13  VOLUME   46465  818750
SO LOCATION SRC14  VOLUME   46465  818760
SO LOCATION SRC15  VOLUME   46465  818770
SO LOCATION SRC16    AREA   46465  818750
SO LOCATION SRC17    AREA   46465  818750
SO SRCPARAM SRC1     1.00 5.33 4.96 14.24
SO SRCPARAM SRC2     1.00 5.33 4.96 14.24
SO SRCPARAM SRC3     1.00 5.33 4.96 14.24
SO SRCPARAM SRC4     1.00 5.33 4.96 14.24
SO SRCPARAM SRC5     1.00 5.33 4.96 10.66
SO SRCPARAM SRC6A    0.56 5.33 4.96 10.66
SO SRCPARAM SRC6B    0.44 5.33 4.96 10.66
SO SRCPARAM SRC7     1.00 5.33 4.96 10.66
SO SRCPARAM SRC8     1.00 5.33 4.96 10.66
SO SRCPARAM SRC9     1.00 5.33 4.96 10.66
SO SRCPARAM SRC10    1.00 5.33 4.96 10.66
SO SRCPARAM SRC11    1.00 5.33 4.96 10.66
SO SRCPARAM SRC12    1.00 5.33 4.96 10.66
SO SRCPARAM SRC13    1.00 6.40 5.95 6.98
SO SRCPARAM SRC14    1.00 6.40 5.95 6.98
SO SRCPARAM SRC15    1.00 6.40 5.95 6.98
SO SRCPARAM SRC16      1.20 1.00 0.91
SO SRCPARAM SRC17      0.43 1.00 1.52
** SOURCE GROUPS
SO SRCGROUP SRC1  SRC1
SO SRCGROUP SRC2  SRC2
SO SRCGROUP SRC3  SRC3
SO SRCGROUP SRC4  SRC4
SO SRCGROUP SRC5  SRC5
SO SRCGROUP SRC6  SRC6A SRC6B
SO SRCGROUP SRC7  SRC7
SO SRCGROUP SRC8  SRC8
SO SRCGROUP SRC9  SRC9
SO SRCGROUP SRC10 SRC10
SO SRCGROUP SRC11 SRC11
SO SRCGROUP SRC12 SRC12
SO SRCGROUP SRC13 SRC13
SO SRCGROUP SRC14 SRC14
SO SRCGROUP SRC15 SRC15
SO SRCGROUP SRC16 SRC16
SO SRCGROUP SRC17 SRC17
SO FINISHED
```

```
RE STARTING
RE BOUNDARY SRC1 1910
RE BOUNDARY SRC1 2000
RE BOUNDARY SRC1 2180
RE BOUNDARY SRC1 1880
RE BOUNDARY SRC1 1340
RE BOUNDARY SRC1 1085
RE BOUNDARY SRC1 930
RE BOUNDARY SRC1 825
RE BOUNDARY SRC1 780
RE BOUNDARY SRC1 750
RE BOUNDARY SRC1 750
RE BOUNDARY SRC1 780
RE BOUNDARY SRC1 825
RE BOUNDARY SRC1 690
RE BOUNDARY SRC1 480
RE BOUNDARY SRC1 380
RE BOUNDARY SRC1 310
RE BOUNDARY SRC1 260
RE BOUNDARY SRC1 250
RE BOUNDARY SRC1 240
RE BOUNDARY SRC1 225
RE BOUNDARY SRC1 230
RE BOUNDARY SRC1 250
RE BOUNDARY SRC1 270
RE BOUNDARY SRC1 300
RE BOUNDARY SRC1 350
RE BOUNDARY SRC1 460
RE BOUNDARY SRC1 575
RE BOUNDARY SRC1 575
RE BOUNDARY SRC1 580
RE BOUNDARY SRC1 610
RE BOUNDARY SRC1 660
RE BOUNDARY SRC1 770
RE BOUNDARY SRC1 935
RE BOUNDARY SRC1 1230
RE BOUNDARY SRC1 1870
RE FINISHED

ME STARTING
ME INPUTFIL CMA84.BIN UNFORM
ME ANEMHGHT 6.1 METERS
ME SURFDATA 05879 1984
ME UAIRDATA 05879 1984
ME FINISHED

OU STARTING
OU RECTABLE 1 FIRST
OU TOXXFILE 1 2.07E-6 ISC8H84.TOX
OU FINISHED

**********************************
*** SETUP Finishes Successfully ***
**********************************
```

*** MODELING OPTIONS USED: CONC RURAL FLAT

*** MODEL SETUP OPTIONS SUMMARY ***

**Model Is Setup For Calculation of Average CONCentration Values.

**Model Uses RURAL Dispersion.

**Model Uses User-Specified Options:
1. Final Plume Rise.
2. Stack-tip Downwash.
3. Buoyancy-induced Dispersion.
4. Calms Processing Routine.
5. Not Use Missing Data Processing Routine.
6. Default Wind Profile Exponents.
7. Default Vertical Potential Temperature Gradients.

**Model Assumes Receptors on FLAT Terrain.

**Model Assumes No FLAGPOLE Receptor Heights.

**Model Calculates 1 Short Term Average(s) of: 1-HR

**This Run Includes: 18 Source(s); 17 Source Group(s); and 36 Receptor(s)

**The Model Assumes A Pollutant Type of: COMPOSIT

**Model Set To Continue RUNning After the Setup Testing.

**Output Options Selected:
Model Outputs Tables of Highest Short Term Values by Receptor (RECTABLE Keyword)
Model Outputs External File(s) of Values for Input to TOXX Model (TOXXFILE Keyword)

**NOTE: The Following Flags May Appear Following CONC Values: c for Calm Hours
m for Missing Hours
b for Both Calm and Missing Hours

**Misc. Inputs: Anem. Hgt. (m) = 6.10 ; Decay Coef. = .0000 ; Rot. Angle = .0
Emission Units = GRAMS/SEC ; Emission Rate Unit Factor = .10000E+07
Output Units = MICROGRAMS/M**3

**Input Runstream File: ISC8H84.INP ; **Output Print File: ISC8H84.OUT

*** MODELING OPTIONS USED: CONC RURAL FLAT

*** VOLUME SOURCE DATA ***

SOURCE ID	NUMBER PART. CATS.	EMISSION RATE (GRAMS/SEC)	X (METERS)	Y (METERS)	BASE ELEV. (METERS)	RELEASE HEIGHT (METERS)	INIT. SY (METERS)	INIT. SZ (METERS)	EMISSION RATE SCALAR VARY BY
SRC1	0	.10000E+01	46375.0	818675.0	.0	5.33	4.96	14.24	
SRC2	0	.10000E+01	46375.0	818675.0	.0	5.33	4.96	14.24	
SRC3	0	.10000E+01	46375.0	818675.0	.0	5.33	4.96	14.24	
SRC4	0	.10000E+01	46375.0	818675.0	.0	5.33	4.96	14.24	
SRC5	0	.10000E+01	46450.0	818750.0	.0	5.33	4.96	10.66	
SRC6A	0	.56000E+00	46450.0	818750.0	.0	5.33	4.96	10.66	
SRC6B	0	.44000E+00	46450.0	818750.0	.0	5.33	4.96	10.66	
SRC7	0	.10000E+01	46450.0	818750.0	.0	5.33	4.96	10.66	
SRC8	0	.10000E+01	46450.0	818750.0	.0	5.33	4.96	10.66	
SRC9	0	.10000E+01	46450.0	818750.0	.0	5.33	4.96	10.66	
SRC10	0	.10000E+01	46450.0	818750.0	.0	5.33	4.96	10.66	
SRC11	0	.10000E+01	46450.0	818750.0	.0	5.33	4.96	10.66	
SRC12	0	.10000E+01	46450.0	818750.0	.0	5.33	4.96	10.66	
SRC13	0	.10000E+01	46465.0	818750.0	.0	6.40	5.95	6.98	
SRC14	0	.10000E+01	46465.0	818760.0	.0	6.40	5.95	6.98	
SRC15	0	.10000E+01	46465.0	818770.0	.0	6.40	5.95	6.98	

*** MODELING OPTIONS USED: CONC RURAL FLAT

*** AREA SOURCE DATA ***

SOURCE ID	NUMBER PART. CATS.	EMISSION RATE (GRAMS/SEC /METER**2)	COORD (SW CORNER) X (METERS)	Y (METERS)	BASE ELEV. (METERS)	RELEASE HEIGHT (METERS)	WIDTH OF AREA (METERS)	EMISSION RATE SCALAR VARY BY
SRC16	0	.12000E+01	46465.0	818750.0	.0	1.00	.91	
SRC17	0	.43000E+00	46465.0	818750.0	.0	1.00	1.52	

*** MODELING OPTIONS USED: CONC RURAL FLAT

*** SOURCE IDs DEFINING SOURCE GROUPS ***

GROUP ID	SOURCE IDs
SRC1	SRC1 ,
SRC2	SRC2 ,
SRC3	SRC3 ,
SRC4	SRC4 ,
SRC5	SRC5 ,
SRC6	SRC6A , SRC6B ,
SRC7	SRC7 ,
SRC8	SRC8 ,
SRC9	SRC9 ,
SRC10	SRC10 ,
SRC11	SRC11 ,
SRC12	SRC12 ,
SRC13	SRC13 ,
SRC14	SRC14 ,
SRC15	SRC15 ,
SRC16	SRC16 ,
SRC17	SRC17 ,

*** MODELING OPTIONS USED: CONC RURAL FLAT

*** BOUNDARY RECEPTOR LOCATIONS ***
(DISCRETE RECEPTORS AT 10 DEGREE SECTORS)

BOUNDARY RECEPTORS FOR SOURCE ID: SRC1
OF SOURCE TYPE: VOLUME; WITH ORIGIN AT (46375.00, 818675.00, .00)

SEC.	XCOORD	YCOORD	ZELEV	ZFLAG	SEC.	XCOORD	YCOORD	ZELEV	ZFLAG	SEC.	XCOORD	YCOORD	ZELEV	ZFLAG
1	46706.70,	820556.00,	.00,	.0	2	47059.00,	820555.00,	.00,	.0	3	47465.00,	820563.00,	.00,	.0
4	47583.40,	820115.00,	.00,	.0	5	47401.50,	819536.00,	.00,	.0	6	47314.60,	819218.00,	.00,	.0
7	47248.90,	818993.00,	.00,	.0	8	47187.50,	818818.00,	.00,	.0	9	47155.00,	818675.00,	.00,	.0
10	47113.60,	818545.00,	.00,	.0	11	47079.80,	818419.00,	.00,	.0	12	47050.50,	818285.00,	.00,	.0
13	47007.00,	818145.00,	.00,	.0	14	46818.50,	818147.00,	.00,	.0	15	46615.00,	818259.00,	.00,	.0
16	46505.00,	818318.00,	.00,	.0	17	46428.80,	818370.00,	.00,	.0	18	46375.00,	818415.00,	.00,	.0
19	46331.60,	818429.00,	.00,	.0	20	46292.90,	818450.00,	.00,	.0	21	46262.50,	818480.00,	.00,	.0
22	46227.20,	818499.00,	.00,	.0	23	46183.50,	818514.00,	.00,	.0	24	46141.20,	818540.00,	.00,	.0
25	46093.10,	818573.00,	.00,	.0	26	46030.30,	818614.00,	.00,	.0	27	45915.00,	818675.00,	.00,	.0
28	45808.70,	818775.00,	.00,	.0	29	45834.70,	818872.00,	.00,	.0	30	45872.70,	818965.00,	.00,	.0
31	45907.70,	819067.00,	.00,	.0	32	45950.80,	819181.00,	.00,	.0	33	45990.00,	819342.00,	.00,	.0
34	46055.20,	819554.00,	.00,	.0	35	46161.40,	819886.00,	.00,	.0	36	46375.00,	820545.00,	.00,	.0

*** MODELING OPTIONS USED: CONC RURAL FLAT

*** METEOROLOGICAL DAYS SELECTED FOR PROCESSING ***
(1=YES; 0=NO)

```
1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1
```

NOTE: METEOROLOGICAL DATA ACTUALLY PROCESSED WILL ALSO DEPEND ON WHAT IS INCLUDED IN THE DATA FILE.

*** UPPER BOUND OF FIRST THROUGH FIFTH WIND SPEED CATEGORIES ***
(METERS/SEC)

1.54, 3.09, 5.14, 8.23, 10.80,

*** WIND PROFILE EXPONENTS ***

STABILITY	WIND SPEED CATEGORY					
CATEGORY	1	2	3	4	5	6
A	.70000E-01	.70000E-01	.70000E-01	.70000E-01	.70000E-01	.70000E-01
B	.70000E-01	.70000E-01	.70000E-01	.70000E-01	.70000E-01	.70000E-01
C	.10000E+00	.10000E+00	.10000E+00	.10000E+00	.10000E+00	.10000E+00
D	.15000E+00	.15000E+00	.15000E+00	.15000E+00	.15000E+00	.15000E+00
E	.35000E+00	.35000E+00	.35000E+00	.35000E+00	.35000E+00	.35000E+00
F	.55000E+00	.55000E+00	.55000E+00	.55000E+00	.55000E+00	.55000E+00

*** VERTICAL POTENTIAL TEMPERATURE GRADIENTS ***
(DEGREES KELVIN PER METER)

STABILITY	WIND SPEED CATEGORY					
CATEGORY	1	2	3	4	5	6
A	.00000E+00	.00000E+00	.00000E+00	.00000E+00	.00000E+00	.00000E+00
B	.00000E+00	.00000E+00	.00000E+00	.00000E+00	.00000E+00	.00000E+00
C	.00000E+00	.00000E+00	.00000E+00	.00000E+00	.00000E+00	.00000E+00
D	.00000E+00	.00000E+00	.00000E+00	.00000E+00	.00000E+00	.00000E+00
E	.20000E-01	.20000E-01	.20000E-01	.20000E-01	.20000E-01	.20000E-01
F	.35000E-01	.35000E-01	.35000E-01	.35000E-01	.35000E-01	.35000E-01

*** MODELING OPTIONS USED: CONC RURAL FLAT

*** THE FIRST 24 HOURS OF METEOROLOGICAL DATA ***

FILE: CMA84.BIN FORMAT: UNFORM
SURFACE STATION NO.: 05879 UPPER AIR STATION NO.: 05879
NAME: UNKNOWN NAME: UNKNOWN
YEAR: 1984 YEAR: 1984

YEAR	MONTH	DAY	HOUR	FLOW VECTOR	SPEED (M/S)	TEMP (K)	STAB CLASS	MIXING HEIGHT (M) RURAL	MIXING HEIGHT (M) URBAN
84	1	1	1	181.0	1.00	268.7	7	645.0	113.0
84	1	1	2	178.0	1.54	268.1	7	645.0	113.0
84	1	1	3	184.0	1.00	267.6	7	645.0	113.0
84	1	1	4	183.0	1.00	267.6	7	645.0	113.0
84	1	1	5	183.0	1.00	267.6	7	645.0	113.0
84	1	1	6	222.0	2.06	267.6	6	645.0	113.0
84	1	1	7	235.0	2.06	266.5	6	645.0	113.0
84	1	1	8	243.0	2.06	268.1	5	89.4	186.7
84	1	1	9	237.0	2.06	273.7	4	182.0	263.1
84	1	1	10	271.0	4.63	277.0	3	274.6	339.5
84	1	1	11	304.0	3.09	280.4	3	367.2	415.9
84	1	1	12	296.0	3.09	282.0	2	459.8	492.2
84	1	1	13	313.0	2.06	284.3	2	552.4	568.6
84	1	1	14	329.0	2.06	284.8	3	645.0	645.0
84	1	1	15	332.0	2.06	285.9	3	645.0	645.0
84	1	1	16	324.0	4.12	285.9	4	645.0	645.0
84	1	1	17	291.0	2.57	284.3	4	645.0	645.0
84	1	1	18	297.0	2.57	281.5	5	637.0	584.1
84	1	1	19	304.0	1.00	279.8	6	627.7	513.4
84	1	1	20	277.0	3.60	278.7	5	618.4	442.7
84	1	1	21	260.0	2.06	275.9	6	609.1	372.0
84	1	1	22	192.0	1.54	274.3	7	599.8	301.4
84	1	1	23	190.0	1.00	274.3	7	590.5	230.7
84	1	1	24	160.0	2.57	272.6	6	581.2	160.0

*** NOTES: STABILITY CLASS 1=A, 2=B, 3=C, 4=D, 5=E AND 6=F.
FLOW VECTOR IS DIRECTION TOWARD WHICH WIND IS BLOWING.

*** MODELING OPTIONS USED: CONC RURAL FLAT

*** THE 1ST HIGHEST 1-HR AVERAGE CONCENTRATION VALUES FOR SOURCE GROUP: SRC1 ***
INCLUDING SOURCE(S): SRC1 ,

** CONC OF COMPOSIT IN MICROGRAMS/M**3 **

BOUNDARY RECEPTOR NETWORK OF SOURCE ID: SRC1
OF SOURCE TYPE: VOLUME; WITH ORIGIN AT (46375.00, 818675.00, .00)

(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)	(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)
1	46706.70,	820556.00,	104.53900	(84082024)	2	47059.00,	820555.00,	98.72540	(84020104)
3	47465.00,	820563.00,	91.23414	(84041720)	4	47583.40,	820115.00,	79.99493	(84042320)
5	47401.50,	819536.00,	271.28060	(84012802)	6	47314.60,	819218.00,	228.90210	(84072120)
7	47248.90,	818993.00,	263.28920	(84042222)	8	47187.50,	818818.00,	271.08520	(84010620)
9	47155.00,	818675.00,	288.14010	(84033021)	10	47113.60,	818545.00,	225.65000	(84013102)
11	47079.80,	818419.00,	225.39970	(84012701)	12	47050.50,	818285.00,	317.49330	(84040901)
13	47007.00,	818145.00,	224.51960	(84091302)	14	46818.50,	818147.00,	360.08810	(84110305)
15	46615.00,	818259.00,	510.39940	(84060103)	16	46505.00,	818318.00,	627.00920	(84110601)
17	46428.80,	818370.00,	740.52660	(84072522)	18	46375.00,	818415.00,	1240.82600	(84010101)
19	46331.60,	818429.00,	871.30550	(84060305)	20	46292.90,	818450.00,	854.84180	(84010802)
21	46262.50,	818480.00,	939.40330	(84060922)	22	46227.20,	818499.00,	924.70780	(84010204)
23	46183.50,	818514.00,	871.32480	(84021504)	24	46141.20,	818540.00,	780.60380	(84020205)
25	46093.10,	818573.00,	718.88410	(84033104)	26	46030.30,	818614.00,	671.46290	(84060505)
27	45915.00,	818675.00,	530.52370	(84021407)	28	45808.70,	818775.00,	395.46380	(84021623)
29	45834.70,	818872.00,	396.61470	(84070104)	30	245872.70,	818965.00,	392.67590	(84060503)
31	45907.70,	819067.00,	407.02590	(84071622)	32	45950.80,	819181.00,	376.54370	(84091122)
33	45990.00,	819342.00,	241.36350	(84063021)	34	46055.20,	819554.00,	235.93010	(84062921)
35	46161.40,	819886.00,	177.54030	(84052502)	36	46375.00,	820545.00,	91.84793	(84082721)

*** MODELING OPTIONS USED: CONC RURAL FLAT

*** THE 1ST HIGHEST 1-HR AVERAGE CONCENTRATION VALUES FOR SOURCE GROUP: SRC2 ***
INCLUDING SOURCE(S): SRC2 ,

** CONC OF COMPOSIT IN MICROGRAMS/M**3 **

BOUNDARY RECEPTOR NETWORK OF SOURCE ID: SRC1
OF SOURCE TYPE: VOLUME; WITH ORIGIN AT (46375.00, 818675.00, .00)

(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)	(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)
1	46706.70,	820556.00,	104.53900	(84082024)	2	47059.00,	820555.00,	98.72540	(84020104)
3	47465.00,	820563.00,	91.23414	(84041720)	4	47583.40,	820115.00,	79.99493	(84042320)
5	47401.50,	819536.00,	271.28060	(84012802)	6	47314.60,	819218.00,	228.90210	(84072120)
7	47248.90,	818993.00,	263.28920	(84042222)	8	47187.50,	818818.00,	271.08520	(84010620)
9	47155.00,	818675.00,	288.14010	(84033021)	10	47113.60,	818545.00,	225.65000	(84013102)
11	47079.80,	818419.00,	225.39970	(84012701)	12	47050.50,	818285.00,	317.49330	(84040901)
13	47007.00,	818145.00,	224.51960	(84091302)	14	46818.50,	818147.00,	360.08810	(84110305)
15	46615.00,	818259.00,	510.39940	(84060103)	16	46505.00,	818318.00,	627.00920	(84110601)
17	46428.80,	818370.00,	740.52660	(84072522)	18	46375.00,	818415.00,	1240.82600	(84010101)
19	46331.60,	818429.00,	871.30550	(84060305)	20	46292.90,	818450.00,	854.84180	(84010802)
21	46262.50,	818480.00,	939.40330	(84060922)	22	46227.20,	818499.00,	924.70780	(84010204)
23	46183.50,	818514.00,	871.32480	(84021504)	24	46141.20,	818540.00,	780.60380	(84020205)
25	46093.10,	818573.00,	718.88410	(84033104)	26	46030.30,	818614.00,	671.46290	(84060505)
27	45915.00,	818675.00,	530.52370	(84021407)	28	45808.70,	818775.00,	395.46380	(84021623)
29	45834.70,	818872.00,	396.61470	(84070104)	30	45872.70,	818965.00,	392.67590	(84060503)
31	45907.70,	819067.00,	407.02590	(84071622)	32	45950.80,	819181.00,	376.54370	(84091122)
33	45990.00,	819342.00,	241.36350	(84063021)	34	46055.20,	819554.00,	235.93010	(84062921)
35	46161.40,	819886.00,	177.54030	(84052502)	36	46375.00,	820545.00,	91.84793	(84082721)

*** MODELING OPTIONS USED: CONC RURAL FLAT

*** THE 1ST HIGHEST 1-HR AVERAGE CONCENTRATION VALUES FOR SOURCE GROUP: SRC3 ***
INCLUDING SOURCE(S): SRC3 ,

** CONC OF COMPOSIT IN MICROGRAMS/M**3 **

BOUNDARY RECEPTOR NETWORK OF SOURCE ID: SRC1
OF SOURCE TYPE: VOLUME; WITH ORIGIN AT (46375.00, 818675.00, .00)

(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)	(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)
1	46706.70,	820556.00,	104.53900	(84082024)	2	47059.00,	820555.00,	98.72540	(84020104)
3	47465.00,	820563.00,	91.23414	(84041720)	4	47583.40,	820115.00,	79.99493	(84042320)
5	47401.50,	819536.00,	271.28060	(84012802)	6	47314.60,	819218.00,	228.90210	(84072120)
7	47248.90,	818993.00,	263.28920	(84042222)	8	47187.50,	818818.00,	271.08520	(84010620)
9	47155.00,	818675.00,	288.14010	(84033021)	10	47113.60,	818545.00,	225.65000	(84013102)
11	47079.80,	818419.00,	225.39970	(84012701)	12	47050.50,	818285.00,	317.49330	(84040901)
13	47007.00,	818145.00,	224.51960	(84091302)	14	46818.50,	818147.00,	360.08810	(84110305)
15	46615.00,	818259.00,	510.39940	(84060103)	16	46505.00,	818318.00,	627.00920	(84110601)
17	46428.80,	818370.00,	740.52660	(84072522)	18	46375.00,	818415.00,	1240.82600	(84010101)
19	46331.60,	818429.00,	871.30550	(84060305)	20	46292.90,	818450.00,	854.84180	(84010802)
21	46262.50,	818480.00,	939.40330	(84060922)	22	46227.20,	818499.00,	924.70780	(84010204)
23	46183.50,	818514.00,	871.32480	(84021504)	24	46141.20,	818540.00,	780.60380	(84020205)
25	46093.10,	818573.00,	718.88410	(84033104)	26	46030.30,	818614.00,	671.46290	(84060505)
27	45915.00,	818675.00,	530.52370	(84021407)	28	45808.70,	818775.00,	395.46380	(84021623)
29	45834.70,	818872.00,	396.61470	(84070104)	30	45872.70,	818965.00,	392.67590	(84060503)
31	45907.70,	819067.00,	407.02590	(84071622)	32	45950.80,	819181.00,	376.54370	(84091122)
33	45990.00,	819342.00,	241.36350	(84063021)	34	46055.20,	819554.00,	235.93010	(84062921)
35	46161.40,	819886.00,	177.54030	(84052502)	36	46375.00,	820545.00,	91.84793	(84082721)

*** MODELING OPTIONS USED: CONC RURAL FLAT

*** THE 1ST HIGHEST 1-HR AVERAGE CONCENTRATION VALUES FOR SOURCE GROUP: SRC4 ***
INCLUDING SOURCE(S): SRC4 ,

** CONC OF COMPOSIT IN MICROGRAMS/M**3 **

BOUNDARY RECEPTOR NETWORK OF SOURCE ID: SRC1
OF SOURCE TYPE: VOLUME; WITH ORIGIN AT (46375.00, 818675.00, .00)

(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)	(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)
1	46706.70,	820556.00,	104.53900	(84082024)	2	47059.00,	820555.00,	98.72540	(84020104)
3	47465.00,	820563.00,	91.23414	(84041720)	4	47583.40,	820115.00,	79.99493	(84042320)
5	47401.50,	819536.00,	271.28060	(84012802)	6	47314.60,	819218.00,	228.90210	(84072120)
7	47248.90,	818993.00,	263.28920	(84042222)	8	47187.50,	818818.00,	271.08520	(84010620)
9	47155.00,	818675.00,	288.14010	(84033021)	10	47113.60,	818545.00,	225.65000	(84013102)
11	47079.80,	818419.00,	225.39970	(84012701)	12	47050.50,	818285.00,	317.49330	(84040901)
13	47007.00,	818145.00,	224.51960	(84091302)	14	46818.50,	818147.00,	360.08810	(84110305)
15	46615.00,	818259.00,	510.39940	(84060103)	16	46505.00,	818318.00,	627.00920	(84110601)
17	46428.80,	818370.00,	740.52660	(84072522)	18	46375.00,	818415.00,	1240.82600	(84010101)
19	46331.60,	818429.00,	871.30550	(84060305)	20	46292.90,	818450.00,	854.84180	(84010802)
21	46262.50,	818480.00,	939.40330	(84060922)	22	46227.20,	818499.00,	924.70780	(84010204)
23	46183.50,	818514.00,	871.32480	(84021504)	24	46141.20,	818540.00,	780.60380	(84020205)
25	46093.10,	818573.00,	718.88410	(84033104)	26	46030.30,	818614.00,	671.46290	(84060505)
27	45915.00,	818675.00,	530.52370	(84021407)	28	45808.70,	818775.00,	395.46380	(84021623)
29	45834.70,	818872.00,	396.61470	(84070104)	30	45872.70,	818965.00,	392.67590	(84060503)
31	45907.70,	819067.00,	407.02590	(84071622)	32	45950.80,	819181.00,	376.54370	(84091122)
33	45990.00,	819342.00,	241.36350	(84063021)	34	46055.20,	819554.00,	235.93010	(84062921)
35	46161.40,	819886.00,	177.54030	(84052502)	36	46375.00,	820545.00,	91.84793	(84082721)

*** MODELING OPTIONS USED: CONC RURAL FLAT

*** THE 1ST HIGHEST 1-HR AVERAGE CONCENTRATION VALUES FOR SOURCE GROUP: SRC5 ***
INCLUDING SOURCE(S): SRC5 ,

** CONC OF COMPOSIT IN MICROGRAMS/M**3 **

BOUNDARY RECEPTOR NETWORK OF SOURCE ID: SRC1
OF SOURCE TYPE: VOLUME; WITH ORIGIN AT (46375.00, 818675.00, .00)

(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)	(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)
1	46706.70,	820556.00,	122.31550	(84091123)	2	47059.00,	820555.00,	127.12700	(84020104)
3	47465.00,	820563.00,	114.84610	(84041720)	4	47583.40,	820115.00,	99.33892	(84042320)
5	47401.50,	819536.00,	313.85100	(84012802)	6	47314.60,	819218.00,	269.65330	(84012904)
7	47248.90,	818993.00,	310.23070	(84012804)	8	47187.50,	818818.00,	382.18620	(84041421)
9	47155.00,	818675.00,	372.73990	(84012707)	10	47113.60,	818545.00,	411.44610	(84020401)
11	47079.80,	818419.00,	400.03780	(84032505)	12	47050.50,	818285.00,	372.57550	(84020406)
13	47007.00,	818145.00,	284.25650	(84060303)	14	46818.50,	818147.00,	392.08220	(84072203)
15	46615.00,	818259.00,	550.69840	(84112404)	16	46505.00,	818318.00,	656.27900	(84090301)
17	46428.80,	818370.00,	833.72860	(84010101)	18	46375.00,	818415.00,	796.93990	(84010122)
19	46331.60,	818429.00,	788.61630	(84010802)	20	46292.90,	818450.00,	816.96710	(84090623)
21	46262.50,	818480.00,	846.25050	(84060105)	22	46227.20,	818499.00,	822.78040	(84072901)
23	46183.50,	818514.00,	778.93790	(84090802)	24	46141.20,	818540.00,	756.00820	(84030306)
25	46093.10,	818573.00,	705.65370	(84072205)	26	46030.30,	818614.00,	611.90270	(84071624)
27	45915.00,	818675.00,	491.66350	(84080603)	28	45808.70,	818775.00,	447.09960	(84072802)
29	45834.70,	818872.00,	458.20240	(84021623)	30	45872.70,	818965.00,	455.14840	(84070104)
31	45907.70,	819067.00,	440.12910	(84020123)	32	45950.80,	819181.00,	413.56510	(84071622)
33	45990.00,	819342.00,	284.63270	(84051103)	34	46055.20,	819554.00,	217.84990	(84030319)
35	46161.40,	819886.00,	213.49990	(84071923)	36	46375.00,	820545.00,	136.22000	(84051301)

*** MODELING OPTIONS USED: CONC RURAL FLAT

*** THE 1ST HIGHEST 1-HR AVERAGE CONCENTRATION VALUES FOR SOURCE GROUP: SRC6 ***
INCLUDING SOURCE(S): SRC6A , SRC6B ,

** CONC OF COMPOSIT IN MICROGRAMS/M**3 **

BOUNDARY RECEPTOR NETWORK OF SOURCE ID: SRC1
OF SOURCE TYPE: VOLUME; WITH ORIGIN AT (46375.00, 818675.00, .00)

(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)	(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)
1	46706.70,	820556.00,	122.31550	(84091123)	2	47059.00,	820555.00,	127.12700	(84020104)
3	47465.00,	820563.00,	114.84610	(84041720)	4	47583.40,	820115.00,	99.33892	(84042320)
5	47401.50,	819536.00,	313.85100	(84012802)	6	47314.60,	819218.00,	269.65330	(84012904)
7	47248.90,	818993.00,	310.23070	(84012804)	8	47187.50,	818818.00,	382.18620	(84041421)
9	47155.00,	818675.00,	372.73990	(84012707)	10	47113.60,	818545.00,	411.44610	(84020401)
11	47079.80,	818419.00,	400.03780	(84032505)	12	47050.50,	818285.00,	372.57550	(84020406)
13	47007.00,	818145.00,	284.25650	(84060303)	14	46818.50,	818147.00,	392.08220	(84072203)
15	46615.00,	818259.00,	550.69840	(84112404)	16	46505.00,	818318.00,	656.27900	(84090301)
17	46428.80,	818370.00,	833.72860	(84010101)	18	46375.00,	818415.00,	796.93990	(84010122)
19	46331.60,	818429.00,	788.61630	(84010802)	20	46292.90,	818450.00,	816.96700	(84090623)
21	46262.50,	818480.00,	846.25050	(84060105)	22	46227.20,	818499.00,	822.78040	(84072901)
23	46183.50,	818514.00,	778.93790	(84090802)	24	46141.20,	818540.00,	756.00830	(84030306)
25	46093.10,	818573.00,	705.65380	(84072205)	26	46030.30,	818614.00,	611.90270	(84071624)
27	45915.00,	818675.00,	491.66340	(84080603)	28	45808.70,	818775.00,	447.09970	(84072802)
29	45834.70,	818872.00,	458.20240	(84021623)	30	45872.70,	818965.00,	455.14840	(84070104)
31	45907.70,	819067.00,	440.12910	(84020123)	32	45950.80,	819181.00,	413.56510	(84071622)
33	45990.00,	819342.00,	284.63270	(84051103)	34	46055.20,	819554.00,	217.84990	(84030319)
35	46161.40,	819886.00,	213.49990	(84071923)	36	46375.00,	820545.00,	136.22000	(84051301)

*** MODELING OPTIONS USED: CONC RURAL FLAT

*** THE 1ST HIGHEST 1-HR AVERAGE CONCENTRATION VALUES FOR SOURCE GROUP: SRC7 ***
INCLUDING SOURCE(S): SRC7 ,

** CONC OF COMPOSIT IN MICROGRAMS/M**3 **

BOUNDARY RECEPTOR NETWORK OF SOURCE ID: SRC1
OF SOURCE TYPE: VOLUME; WITH ORIGIN AT (46375.00, 818675.00, .00)

(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)	(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)
1	46706.70,	820556.00,	122.31550	(84091123)	2	47059.00,	820555.00,	127.12700	(84020104)
3	47465.00,	820563.00,	114.84610	(84041720)	4	47583.40,	820115.00,	99.33892	(84042320)
5	47401.50,	819536.00,	313.85100	(84012802)	6	47314.60,	819218.00,	269.65330	(84012904)
7	47248.90,	818993.00,	310.23070	(84012804)	8	47187.50,	818818.00,	382.18620	(84041421)
9	47155.00,	818675.00,	372.73990	(84012707)	10	47113.60,	818545.00,	411.44610	(84020401)
11	47079.80,	818419.00,	400.03780	(84032505)	12	47050.50,	818285.00,	372.57550	(84020406)
13	47007.00,	818145.00,	284.25650	(84060303)	14	46818.50,	818147.00,	392.08220	(84072203)
15	46615.00,	818259.00,	550.69840	(84112404)	16	46505.00,	818318.00,	656.27900	(84090301)
17	46428.80,	818370.00,	833.72860	(84010101)	18	46375.00,	818415.00,	796.93990	(84010122)
19	46331.60,	818429.00,	788.61630	(84010802)	20	46292.90,	818450.00,	816.96710	(84090623)
21	46262.50,	818480.00,	846.25050	(84060105)	22	46227.20,	818499.00,	822.78040	(84072901)
23	46183.50,	818514.00,	778.93790	(84090802)	24	46141.20,	818540.00,	756.00820	(84030306)
25	46093.10,	818573.00,	705.65370	(84072205)	26	46030.30,	818614.00,	611.90270	(84071624)
27	45915.00,	818675.00,	491.66350	(84080603)	28	45808.70,	818775.00,	447.09960	(84072802)
29	45834.70,	818872.00,	458.20240	(84021623)	30	45872.70,	818965.00,	455.14840	(84070104)
31	45907.70,	819067.00,	440.12910	(84020123)	32	45950.80,	819181.00,	413.56510	(84071622)
33	45990.00,	819342.00,	284.63270	(84051103)	34	46055.20,	819554.00,	217.84990	(84030319)
35	46161.40,	819886.00,	213.49990	(84071923)	36	46375.00,	820545.00,	136.22000	(84051301)

*** MODELING OPTIONS USED: CONC RURAL FLAT

*** THE 1ST HIGHEST 1-HR AVERAGE CONCENTRATION VALUES FOR SOURCE GROUP: SRC8 ***
INCLUDING SOURCE(S): SRC8 ,

** CONC OF COMPOSIT IN MICROGRAMS/M**3 **

BOUNDARY RECEPTOR NETWORK OF SOURCE ID: SRC1
OF SOURCE TYPE: VOLUME; WITH ORIGIN AT (46375.00, 818675.00, .00)

(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)	(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)
1	46706.70,	820556.00,	122.31550	(84091123)	2	47059.00,	820555.00,	127.12700	(84020104)
3	47465.00,	820563.00,	114.84610	(84041720)	4	47583.40,	820115.00,	99.33892	(84042320)
5	47401.50,	819536.00,	313.85100	(84012802)	6	47314.60,	819218.00,	269.65330	(84012904)
7	47248.90,	818993.00,	310.23070	(84012804)	8	47187.50,	818818.00,	382.18620	(84041421)
9	47155.00,	818675.00,	372.73990	(84012707)	10	47113.60,	818545.00,	411.44610	(84020401)
11	47079.80,	818419.00,	400.03780	(84032505)	12	47050.50,	818285.00,	372.57550	(84020406)
13	47007.00,	818145.00,	284.25650	(84060303)	14	46818.50,	818147.00,	392.08220	(84072203)
15	46615.00,	818259.00,	550.69840	(84112404)	16	46505.00,	818318.00,	656.27900	(84090301)
17	46428.80,	818370.00,	833.72860	(84010101)	18	46375.00,	818415.00,	796.93990	(84010122)
19	46331.60,	818429.00,	788.61630	(84010802)	20	46292.90,	818450.00,	816.96710	(84090623)
21	46262.50,	818480.00,	846.25050	(84060105)	22	46227.20,	818499.00,	822.78040	(84072901)
23	46183.50,	818514.00,	778.93790	(84090802)	24	46141.20,	818540.00,	756.00820	(84030306)
25	46093.10,	818573.00,	705.65370	(84072205)	26	46030.30,	818614.00,	611.90270	(84071624)
27	45915.00,	818675.00,	491.66350	(84080603)	28	45808.70,	818775.00,	447.09960	(84072802)
29	45834.70,	818872.00,	458.20240	(84021623)	30	45872.70,	818965.00,	455.14840	(84070104)
31	45907.70,	819067.00,	440.12910	(84020123)	32	45950.80,	819181.00,	413.56510	(84071622)
33	45990.00,	819342.00,	284.63270	(84051103)	34	46055.20,	819554.00,	217.84990	(84030319)
35	46161.40,	819886.00,	213.49990	(84071923)	36	46375.00,	820545.00,	136.22000	(84051301)

*** MODELING OPTIONS USED: CONC RURAL FLAT

*** THE 1ST HIGHEST 1-HR AVERAGE CONCENTRATION VALUES FOR SOURCE GROUP: SRC9 ***
INCLUDING SOURCE(S): SRC9 ,

** CONC OF COMPOSIT IN MICROGRAMS/M**3 **

BOUNDARY RECEPTOR NETWORK OF SOURCE ID: SRC1
OF SOURCE TYPE: VOLUME; WITH ORIGIN AT (46375.00, 818675.00, .00)

(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)	(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)
1	46706.70,	820556.00,	122.31550	(84091123)	2	47059.00,	820555.00,	127.12700	(84020104)
3	47465.00,	820563.00,	114.84610	(84041720)	4	47583.40,	820115.00,	99.33892	(84042320)
5	47401.50,	819536.00,	313.85100	(84012802)	6	47314.60,	819218.00,	269.65330	(84012904)
7	47248.90,	818993.00,	310.23070	(84012804)	8	47187.50,	818818.00,	382.18620	(84041421)
9	47155.00,	818675.00,	372.73990	(84012707)	10	47113.60,	818545.00,	411.44610	(84020401)
11	47079.80,	818419.00,	400.03780	(84032505)	12	47050.50,	818285.00,	372.57550	(84020406)
13	47007.00,	818145.00,	284.25650	(84060303)	14	46818.50,	818147.00,	392.08220	(84072203)
15	46615.00,	818259.00,	550.69840	(84112404)	16	46505.00,	818318.00,	656.27900	(84090301)
17	46428.80,	818370.00,	833.72860	(84010101)	18	46375.00,	818415.00,	796.93990	(84010122)
19	46331.60,	818429.00,	788.61630	(84010802)	20	46292.90,	818450.00,	816.96710	(84090623)
21	46262.50,	818480.00,	846.25050	(84060105)	22	46227.20,	818499.00,	822.78040	(84072901)
23	46183.50,	818514.00,	778.93790	(84090802)	24	46141.20,	818540.00,	756.00820	(84030306)
25	46093.10,	818573.00,	705.65370	(84072205)	26	46030.30,	818614.00,	611.90270	(84071624)
27	45915.00,	818675.00,	491.66350	(84080603)	28	45808.70,	818775.00,	447.09960	(84072802)
29	45834.70,	818872.00,	458.20240	(84021623)	30	45872.70,	818965.00,	455.14840	(84070104)
31	45907.70,	819067.00,	440.12910	(84020123)	32	45950.80,	819181.00,	413.56510	(84071622)
33	45990.00,	819342.00,	284.63270	(84051103)	34	46055.20,	819554.00,	217.84990	(84030319)
35	46161.40,	819886.00,	213.49990	(84071923)	36	46375.00,	820545.00,	136.22000	(84051301)

*** MODELING OPTIONS USED: CONC RURAL FLAT

*** THE 1ST HIGHEST 1-HR AVERAGE CONCENTRATION VALUES FOR SOURCE GROUP: SRC10 ***
INCLUDING SOURCE(S): SRC10 ,

** CONC OF COMPOSIT IN MICROGRAMS/M**3 **

BOUNDARY RECEPTOR NETWORK OF SOURCE ID: SRC1
OF SOURCE TYPE: VOLUME; WITH ORIGIN AT (46375.00, 818675.00, .00)

(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)	(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)
1	46706.70,	820556.00,	122.31550	(84091123)	2	47059.00,	820555.00,	127.12700	(84020104)
3	47465.00,	820563.00,	114.84610	(84041720)	4	47583.40,	820115.00,	99.33892	(84042320)
5	47401.50,	819536.00,	313.85100	(84012802)	6	47314.60,	819218.00,	269.65330	(84012904)
7	47248.90,	818993.00,	310.23070	(84012804)	8	47187.50,	818818.00,	382.18620	(84041421)
9	47155.00,	818675.00,	372.73990	(84012707)	10	47113.60,	818545.00,	411.44610	(84020401)
11	47079.80,	818419.00,	400.03780	(84032505)	12	47050.50,	818285.00,	372.57550	(84020406)
13	47007.00,	818145.00,	284.25650	(84060303)	14	46818.50,	818147.00,	392.08220	(84072203)
15	46615.00,	818259.00,	550.69840	(84112404)	16	46505.00,	818318.00,	656.27900	(84090301)
17	46428.80,	818370.00,	833.72860	(84010101)	18	46375.00,	818415.00,	796.93990	(84010122)
19	46331.60,	818429.00,	788.61630	(84010802)	20	46292.90,	818450.00,	816.96710	(84090623)
21	46262.50,	818480.00,	846.25050	(84060105)	22	46227.20,	818499.00,	822.78040	(84072901)
23	46183.50,	818514.00,	778.93790	(84090802)	24	46141.20,	818540.00,	756.00820	(84030306)
25	46093.10,	818573.00,	705.65370	(84072205)	26	46030.30,	818614.00,	611.90270	(84071624)
27	45915.00,	818675.00,	491.66350	(84080603)	28	45808.70,	818775.00,	447.09960	(84072802)
29	45834.70,	818872.00,	458.20240	(84021623)	30	45872.70,	818965.00,	455.14840	(84070104)
31	45907.70,	819067.00,	440.12910	(84020123)	32	45950.80,	819181.00,	413.56510	(84071622)
33	45990.00,	819342.00,	284.63270	(84051103)	34	46055.20,	819554.00,	217.84990	(84030319)
35	46161.40,	819886.00,	213.49990	(84071923)	36	46375.00,	820545.00,	136.22000	(84051301)

*** MODELING OPTIONS USED: CONC RURAL FLAT

*** THE 1ST HIGHEST 1-HR AVERAGE CONCENTRATION VALUES FOR SOURCE GROUP: SRC11 ***
INCLUDING SOURCE(S): SRC11 ,

** CONC OF COMPOSIT IN MICROGRAMS/M**3 **

BOUNDARY RECEPTOR NETWORK OF SOURCE ID: SRC1
OF SOURCE TYPE: VOLUME; WITH ORIGIN AT (46375.00, 818675.00, .00)

(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)	(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)
1	46706.70,	820556.00,	122.31550	(84091123)	2	47059.00,	820555.00,	127.12700	(84020104)
3	47465.00,	820563.00,	114.84610	(84041720)	4	47583.40,	820115.00,	99.33892	(84042320)
5	47401.50,	819536.00,	313.85100	(84012802)	6	47314.60,	819218.00,	269.65330	(84012904)
7	47248.90,	818993.00,	310.23070	(84012804)	8	47187.50,	818818.00,	382.18620	(84041421)
9	47155.00,	818675.00,	372.73990	(84012707)	10	47113.60,	818545.00,	411.44610	(84020401)
11	47079.80,	818419.00,	400.03780	(84032505)	12	47050.50,	818285.00,	372.57550	(84020406)
13	47007.00,	818145.00,	284.25650	(84060303)	14	46818.50,	818147.00,	392.08220	(84072203)
15	46615.00,	818259.00,	550.69840	(84112404)	16	46505.00,	818318.00,	656.27900	(84090301)
17	46428.80,	818370.00,	833.72860	(84010101)	18	46375.00,	818415.00,	796.93990	(84010122)
19	46331.60,	818429.00,	788.61630	(84010802)	20	46292.90,	818450.00,	816.96710	(84090623)
21	46262.50,	818480.00,	846.25050	(84060105)	22	46227.20,	818499.00,	822.78040	(84072901)
23	46183.50,	818514.00,	778.93790	(84090802)	24	46141.20,	818540.00,	756.00820	(84030306)
25	46093.10,	818573.00,	705.65370	(84072205)	26	46030.30,	818614.00,	611.90270	(84071624)
27	45915.00,	818675.00,	491.66350	(84080603)	28	45808.70,	818775.00,	447.09960	(84072802)
29	45834.70,	818872.00,	458.20240	(84021623)	30	45872.70,	818965.00,	455.14840	(84070104)
31	45907.70,	819067.00,	440.12910	(84020123)	32	45950.80,	819181.00,	413.56510	(84071622)
33	45990.00,	819342.00,	284.63270	(84051103)	34	46055.20,	819554.00,	217.84990	(84030319)
35	46161.40,	819886.00,	213.49990	(84071923)	36	46375.00,	820545.00,	136.22000	(84051301)

*** MODELING OPTIONS USED: CONC RURAL FLAT

*** THE 1ST HIGHEST 1-HR AVERAGE CONCENTRATION VALUES FOR SOURCE GROUP: SRC12 ***
INCLUDING SOURCE(S): SRC12 ,

** CONC OF COMPOSIT IN MICROGRAMS/M**3 **

BOUNDARY RECEPTOR NETWORK OF SOURCE ID: SRC1
OF SOURCE TYPE: VOLUME; WITH ORIGIN AT (46375.00, 818675.00, .00)

(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)	(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)
1	46706.70,	820556.00,	122.31550	(84091123)	2	47059.00,	820555.00,	127.12700	(84020104)
3	47465.00,	820563.00,	114.84610	(84041720)	4	47583.40,	820115.00,	99.33892	(84042320)
5	47401.50,	819536.00,	313.85100	(84012802)	6	47314.60,	819218.00,	269.65330	(84012904)
7	47248.90,	818993.00,	310.23070	(84012804)	8	47187.50,	818818.00,	382.18620	(84041421)
9	47155.00,	818675.00,	372.73990	(84012707)	10	47113.60,	818545.00,	411.44610	(84020401)
11	47079.80,	818419.00,	400.03780	(84032505)	12	47050.50,	818285.00,	372.57550	(84020406)
13	47007.00,	818145.00,	284.25650	(84060303)	14	46818.50,	818147.00,	392.08220	(84072203)
15	46615.00,	818259.00,	550.69840	(84112404)	16	46505.00,	818318.00,	656.27900	(84090301)
17	46428.80,	818370.00,	833.72860	(84010101)	18	46375.00,	818415.00,	796.93990	(84010122)
19	46331.60,	818429.00,	788.61630	(84010802)	20	46292.90,	818450.00,	816.96710	(84090623)
21	46262.50,	818480.00,	846.25050	(84060105)	22	46227.20,	818499.00,	822.78040	(84072901)
23	46183.50,	818514.00,	778.93790	(84090802)	24	46141.20,	818540.00,	756.00820	(84030306)
25	46093.10,	818573.00,	705.65370	(84072205)	26	46030.30,	818614.00,	611.90270	(84071624)
27	45915.00,	818675.00,	491.66350	(84080603)	28	45808.70,	818775.00,	447.09960	(84072802)
29	45834.70,	818872.00,	458.20240	(84021623)	30	45872.70,	818965.00,	455.14840	(84070104)
31	45907.70,	819067.00,	440.12910	(84020123)	32	45950.80,	819181.00,	413.56510	(84071622)
33	45990.00,	819342.00,	284.63270	(84051103)	34	46055.20,	819554.00,	217.84990	(84030319)
35	46161.40,	819886.00,	213.49990	(84071923)	36	46375.00,	820545.00,	136.22000	(84051301)

*** MODELING OPTIONS USED: CONC RURAL FLAT

*** THE 1ST HIGHEST 1-HR AVERAGE CONCENTRATION VALUES FOR SOURCE GROUP: SRC13 ***
INCLUDING SOURCE(S): SRC13 ,

** CONC OF COMPOSIT IN MICROGRAMS/M**3 **

BOUNDARY RECEPTOR NETWORK OF SOURCE ID: SRC1
OF SOURCE TYPE: VOLUME; WITH ORIGIN AT (46375.00, 818675.00, .00)

(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)	(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)
1	46706.70,	820556.00,	135.71290	(84070405)	2	47059.00,	820555.00,	134.30710	(84120723)
3	47465.00,	820563.00,	120.74040	(84041720)	4	47583.40,	820115.00,	118.94690	(84012824)
5	47401.50,	819536.00,	348.76090	(84012802)	6	47314.60,	819218.00,	307.44080	(84012904)
7	47248.90,	818993.00,	325.53170	(84012804)	8	47187.50,	818818.00,	424.28390	(84041421)
9	47155.00,	818675.00,	427.59890	(84012707)	10	47113.60,	818545.00,	449.49670	(84020401)
11	47079.80,	818419.00,	444.36570	(84032505)	12	47050.50,	818285.00,	408.48860	(84020406)
13	47007.00,	818145.00,	354.59600	(84110607)	14	46818.50,	818147.00,	442.95700	(84060103)
15	46615.00,	818259.00,	569.06410	(84022501)	16	46505.00,	818318.00,	717.54940	(84053102)
17	46428.80,	818370.00,	802.10320	(84072005)	18	46375.00,	818415.00,	837.67260	(84010407)
19	46331.60,	818429.00,	871.02180	(84072503)	20	46292.90,	818450.00,	882.47030	(84060922)
21	46262.50,	818480.00,	904.12740	(84073105)	22	46227.20,	818499.00,	868.95400	(84100403)
23	46183.50,	818514.00,	840.27790	(84021504)	24	46141.20,	818540.00,	803.77340	(84051206)
25	46093.10,	818573.00,	745.63960	(84072205)	26	46030.30,	818614.00,	683.27390	(84071624)
27	45915.00,	818675.00,	502.31330	(84080603)	28	45808.70,	818775.00,	478.83640	(84072802)
29	45834.70,	818872.00,	491.79390	(84021623)	30	45872.70,	818965.00,	451.11170	(84070104)
31	45907.70,	819067.00,	476.51460	(84060503)	32	45950.80,	819181.00,	468.81220	(84071622)
33	45990.00,	819342.00,	355.64140	(84091122)	34	46055.20,	819554.00,	251.95750	(84030319)
35	46161.40,	819886.00,	243.99460	(84071923)	36	46375.00,	820545.00,	144.79180	(84051223)

*** MODELING OPTIONS USED: CONC RURAL FLAT

*** THE 1ST HIGHEST 1-HR AVERAGE CONCENTRATION VALUES FOR SOURCE GROUP: SRC14 ***
INCLUDING SOURCE(S): SRC14 ,

** CONC OF COMPOSIT IN MICROGRAMS/M**3 **

BOUNDARY RECEPTOR NETWORK OF SOURCE ID: SRC1
OF SOURCE TYPE: VOLUME; WITH ORIGIN AT (46375.00, 818675.00, .00)

(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)	(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)
1	46706.70,	820556.00,	135.73690	(84070405)	2	47059.00,	820555.00,	134.28650	(84120723)
3	47465.00,	820563.00,	121.63410	(84041720)	4	47583.40,	820115.00,	111.05690	(84012824)
5	47401.50,	819536.00,	346.45590	(84012802)	6	47314.60,	819218.00,	295.30550	(84012904)
7	47248.90,	818993.00,	365.53810	(84012804)	8	47187.50,	818818.00,	425.03450	(84041421)
9	47155.00,	818675.00,	450.98730	(84012707)	10	47113.60,	818545.00,	391.86900	(84020401)
11	47079.80,	818419.00,	411.72290	(84040901)	12	47050.50,	818285.00,	373.13290	(84020406)
13	47007.00,	818145.00,	370.10280	(84110607)	14	46818.50,	818147.00,	441.96490	(84060103)
15	46615.00,	818259.00,	580.83450	(84022501)	16	46505.00,	818318.00,	704.74180	(84053102)
17	46428.80,	818370.00,	788.45820	(84072005)	18	46375.00,	818415.00,	843.24810	(84020724)
19	46331.60,	818429.00,	861.04700	(84072004)	20	46292.90,	818450.00,	865.71150	(84033101)
21	46262.50,	818480.00,	886.09030	(84071501)	22	46227.20,	818499.00,	864.33920	(84072901)
23	46183.50,	818514.00,	826.74730	(84090802)	24	46141.20,	818540.00,	791.75290	(84030306)
25	46093.10,	818573.00,	746.73440	(84101124)	26	46030.30,	818614.00,	674.27970	(84033104)
27	45915.00,	818675.00,	561.53930	(84080603)	28	45808.70,	818775.00,	476.77170	(84012822)
29	45834.70,	818872.00,	457.99500	(84021623)	30	45872.70,	818965.00,	450.48300	(84050405)
31	45907.70,	819067.00,	488.04440	(84060503)	32	45950.80,	819181.00,	453.09120	(84071622)
33	45990.00,	819342.00,	392.66440	(84091122)	34	46055.20,	819554.00,	252.47640	(84030319)
35	46161.40,	819886.00,	246.34160	(84071923)	36	46375.00,	820545.00,	145.89650	(84051223)

*** MODELING OPTIONS USED: CONC RURAL FLAT

*** THE 1ST HIGHEST 1-HR AVERAGE CONCENTRATION VALUES FOR SOURCE GROUP: SRC15 ***
INCLUDING SOURCE(S): SRC15 ,

** CONC OF COMPOSIT IN MICROGRAMS/M**3 **

BOUNDARY RECEPTOR NETWORK OF SOURCE ID: SRC1
OF SOURCE TYPE: VOLUME; WITH ORIGIN AT (46375.00, 818675.00, .00)

(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)	(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)
1	46706.70,	820556.00,	135.69680	(84070405)	2	47059.00,	820555.00,	133.94760	(84120723)
3	47465.00,	820563.00,	121.94590	(84041720)	4	47583.40,	820115.00,	105.11380	(84042320)
5	47401.50,	819536.00,	334.20830	(84012802)	6	47314.60,	819218.00,	268.27160	(84012904)
7	47248.90,	818993.00,	377.15330	(84012804)	8	47187.50,	818818.00,	381.26220	(84041421)
9	47155.00,	818675.00,	449.08250	(84013102)	10	47113.60,	818545.00,	306.45900	(84020401)
11	47079.80,	818419.00,	439.75160	(84040901)	12	47050.50,	818285.00,	320.30460	(84020406)
13	47007.00,	818145.00,	370.96730	(84110607)	14	46818.50,	818147.00,	428.56850	(84060103)
15	46615.00,	818259.00,	583.98460	(84022501)	16	46505.00,	818318.00,	691.06950	(84053102)
17	46428.80,	818370.00,	773.48790	(84072005)	18	46375.00,	818415.00,	839.82740	(84020724)
19	46331.60,	818429.00,	835.84830	(84010802)	20	46292.90,	818450.00,	844.70920	(84090623)
21	46262.50,	818480.00,	868.76740	(84060105)	22	46227.20,	818499.00,	850.58840	(84020807)
23	46183.50,	818514.00,	811.30700	(84051205)	24	46141.20,	818540.00,	775.92880	(84030106)
25	46093.10,	818573.00,	741.95730	(84021503)	26	46030.30,	818614.00,	656.15620	(84033104)
27	45915.00,	818675.00,	560.11240	(84060505)	28	45808.70,	818775.00,	473.30940	(84021407)
29	45834.70,	818872.00,	441.63270	(84061721)	30	45872.70,	818965.00,	497.66780	(84050405)
31	45907.70,	819067.00,	492.82930	(84020704)	32	45950.80,	819181.00,	464.53400	(84103022)
33	45990.00,	819342.00,	415.99130	(84091122)	34	46055.20,	819554.00,	248.71090	(84030319)
35	46161.40,	819886.00,	247.83530	(84071923)	36	46375.00,	820545.00,	147.00560	(84051223)

*** MODELING OPTIONS USED: CONC RURAL FLAT

*** THE 1ST HIGHEST 1-HR AVERAGE CONCENTRATION VALUES FOR SOURCE GROUP: SRC16 ***
INCLUDING SOURCE(S): SRC16 ,

** CONC OF COMPOSIT IN MICROGRAMS/M**3 **

BOUNDARY RECEPTOR NETWORK OF SOURCE ID: SRC1
OF SOURCE TYPE: VOLUME; WITH ORIGIN AT (46375.00, 818675.00, .00)

(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)	(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)
1	46706.70,	820556.00,	162.51520	(84070405)	2	47059.00,	820555.00,	159.91720	(84120723)
3	47465.00,	820563.00,	141.12930	(84041720)	4	47583.40,	820115.00,	139.45220	(84012824)
5	47401.50,	819536.00,	476.14350	(84012802)	6	47314.60,	819218.00,	452.42730	(84012904)
7	47248.90,	818993.00,	493.55110	(84012804)	8	47187.50,	818818.00,	710.17000	(84041421)
9	47155.00,	818675.00,	722.84230	(84012707)	10	47113.60,	818545.00,	765.64270	(84020401)
11	47079.80,	818419.00,	752.52390	(84032505)	12	47050.50,	818285.00,	665.57980	(84020406)
13	47007.00,	818145.00,	555.25160	(84110607)	14	46818.50,	818147.00,	755.30100	(84060103)
15	46615.00,	818259.00,	1164.03600	(84022501)	16	46505.00,	818318.00,	1702.61900	(84053102)
17	46428.80,	818370.00,	2063.70800	(84072005)	18	46375.00,	818415.00,	2274.18000	(84010407)
19	46331.60,	818429.00,	2435.99000	(84072503)	20	46292.90,	818450.00,	2491.92500	(84060922)
21	46262.50,	818480.00,	2611.02700	(84073105)	22	46227.20,	818499.00,	2414.79200	(84100403)
23	46183.50,	818514.00,	2265.40300	(84021504)	24	46141.20,	818540.00,	2083.57300	(84051206)
25	46093.10,	818573.00,	1823.26300	(84072205)	26	46030.30,	818614.00,	1549.87900	(84071624)
27	45915.00,	818675.00,	952.57060	(84080603)	28	45808.70,	818775.00,	848.41670	(84072802)
29	45834.70,	818872.00,	881.74790	(84021623)	30	45872.70,	818965.00,	784.88650	(84070104)
31	45907.70,	819067.00,	850.62950	(84060503)	32	45950.80,	819181.00,	819.49790	(84071622)
33	45990.00,	819342.00,	560.63700	(84091122)	34	46055.20,	819554.00,	382.31200	(84030319)
35	46161.40,	819886.00,	336.93160	(84071923)	36	46375.00,	820545.00,	174.99690	(84051223)

*** MODELING OPTIONS USED: CONC RURAL FLAT

*** THE 1ST HIGHEST 1-HR AVERAGE CONCENTRATION VALUES FOR SOURCE GROUP: SRC17 ***
INCLUDING SOURCE(S): SRC17 ,

** CONC OF COMPOSIT IN MICROGRAMS/M**3 **

BOUNDARY RECEPTOR NETWORK OF SOURCE ID: SRC1
OF SOURCE TYPE: VOLUME; WITH ORIGIN AT (46375.00, 818675.00, .00)

(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)	(SEC.)	X-COORD	Y-COORD	CONC	(YYMMDDHH)
1	46706.70,	820556.00,	162.77410	(84070405)	2	47059.00,	820555.00,	159.99520	(84120723)
3	47465.00,	820563.00,	141.11660	(84041720)	4	47583.40,	820115.00,	139.61020	(84012824)
5	47401.50,	819536.00,	476.29880	(84012802)	6	47314.60,	819218.00,	452.54260	(84012904)
7	47248.90,	818993.00,	495.32100	(84012804)	8	47187.50,	818818.00,	711.64400	(84041421)
9	47155.00,	818675.00,	726.26790	(84012707)	10	47113.60,	818545.00,	762.54020	(84020401)
11	47079.80,	818419.00,	750.34090	(84032505)	12	47050.50,	818285.00,	662.94760	(84020406)
13	47007.00,	818145.00,	558.28760	(84110607)	14	46818.50,	818147.00,	756.84340	(84060103)
15	46615.00,	818259.00,	1173.52100	(84022501)	16	46505.00,	818318.00,	1704.05300	(84053102)
17	46428.80,	818370.00,	2067.51400	(84062606)	18	46375.00,	818415.00,	2287.15000	(84010407)
19	46331.60,	818429.00,	2437.62500	(84072503)	20	46292.90,	818450.00,	2488.65900	(84060922)
21	46262.50,	818480.00,	2605.74900	(84073105)	22	46227.20,	818499.00,	2411.86400	(84100403)
23	46183.50,	818514.00,	2260.39000	(84021504)	24	46141.20,	818540.00,	2079.25800	(84051206)
25	46093.10,	818573.00,	1822.05900	(84072205)	26	46030.30,	818614.00,	1544.66800	(84071624)
27	45915.00,	818675.00,	956.91630	(84080603)	28	45808.70,	818775.00,	848.04900	(84072802)
29	45834.70,	818872.00,	880.15740	(84021623)	30	45872.70,	818965.00,	777.50970	(84070104)
31	45907.70,	819067.00,	853.94820	(84060503)	32	45950.80,	819181.00,	818.19780	(84071622)
33	45990.00,	819342.00,	565.55270	(84091122)	34	46055.20,	819554.00,	382.11300	(84030319)
35	46161.40,	819886.00,	336.95000	(84071923)	36	46375.00,	820545.00,	175.05630	(84051223)

*** MODELING OPTIONS USED: CONC RURAL FLAT

*** THE SUMMARY OF HIGHEST 1-HR RESULTS ***

** CONC OF COMPOSIT IN MICROGRAMS/M**3 **

GROUP ID	AVERAGE CONC	DATE (YYMMDDHH)	RECEPTOR (XR, YR, ZELEV, ZFLAG)	OF TYPE	NETWORK GRID-ID
SRC1 HIGH 1ST HIGH VALUE IS	1240.82600	ON 84010101: AT (	46375.00, 818415.00, .00, .00)	BD	
SRC2 HIGH 1ST HIGH VALUE IS	1240.82600	ON 84010101: AT (	46375.00, 818415.00, .00, .00)	BD	
SRC3 HIGH 1ST HIGH VALUE IS	1240.82600	ON 84010101: AT (	46375.00, 818415.00, .00, .00)	BD	
SRC4 HIGH 1ST HIGH VALUE IS	1240.82600	ON 84010101: AT (	46375.00, 818415.00, .00, .00)	BD	
SRC5 HIGH 1ST HIGH VALUE IS	846.25050	ON 84060105: AT (	46262.50, 818480.00, .00, .00)	BD	
SRC6 HIGH 1ST HIGH VALUE IS	846.25050	ON 84060105: AT (	46262.50, 818480.00, .00, .00)	BD	
SRC7 HIGH 1ST HIGH VALUE IS	846.25050	ON 84060105: AT (	46262.50, 818480.00, .00, .00)	BD	
SRC8 HIGH 1ST HIGH VALUE IS	846.25050	ON 84060105: AT (	46262.50, 818480.00, .00, .00)	BD	
SRC9 HIGH 1ST HIGH VALUE IS	846.25050	ON 84060105: AT (	46262.50, 818480.00, .00, .00)	BD	
SRC10 HIGH 1ST HIGH VALUE IS	846.25050	ON 84060105: AT (	46262.50, 818480.00, .00, .00)	BD	
SRC11 HIGH 1ST HIGH VALUE IS	846.25050	ON 84060105: AT (	46262.50, 818480.00, .00, .00)	BD	
SRC12 HIGH 1ST HIGH VALUE IS	846.25050	ON 84060105: AT (	46262.50, 818480.00, .00, .00)	BD	
SRC13 HIGH 1ST HIGH VALUE IS	904.12740	ON 84073105: AT (	46262.50, 818480.00, .00, .00)	BD	
SRC14 HIGH 1ST HIGH VALUE IS	886.09030	ON 84071501: AT (	46262.50, 818480.00, .00, .00)	BD	
SRC15 HIGH 1ST HIGH VALUE IS	868.76740	ON 84060105: AT (	46262.50, 818480.00, .00, .00)	BD	
SRC16 HIGH 1ST HIGH VALUE IS	2611.02700	ON 84073105: AT (	46262.50, 818480.00, .00, .00)	BD	
SRC17 HIGH 1ST HIGH VALUE IS	2605.74900	ON 84073105: AT (	46262.50, 818480.00, .00, .00)	BD	

*** RECEPTOR TYPES: GC = GRIDCART
GP = GRIDPOLR
DC = DISCCART
DP = DISCPOLR
BD = BOUNDARY

*** MODELING OPTIONS USED: CONC RURAL FLAT

*** Message Summary For ISC2 Model Execution ***

--------- Summary of Total Messages --------

A Total of 0 Fatal Error Message(s)
A Total of 0 Warning Message(s)
A Total of 902 Informational Message(s)

A Total of 902 Calm Hours Identified

******** FATAL ERROR MESSAGES ********
*** NONE ***

******** WARNING MESSAGES ********
*** NONE ***

****
*** ISCST2 Finishes Successfully ***
****

********** Example TOXST Input File **********

```
CASE STUDY OPERATIONS - 8-HOUR - 1984-1985  DIST
6, 385., 550., 900., 1350., 1800., 2700.
1, 0.
THRESHOLD #1
THRESHOLD #2
THRESHOLD #3
THRESHOLD #4
THRESHOLD #5
THRESHOLD #6
COMPOSITE
200, 17, 8, 17
18, 0, 0, 1
2, 1984, 1985
ISC8H84.OUT
SOURCE 1          (371)
1, 1, 1.0000,  1,   2
1.0,  1.0
 2184,    736
0.304,   0.000
0.304,   0.000
SOURCE 2          (374)
2, 2, 1.0000,  1,   2
1.0,  1.0
 2184,    736
0.027,   0.000
0.027,   0.000
SOURCE 3          (378)
3, 3, 0.0034,  1,   1
1.0
   144
0.955
0.955
SOURCE 4          (403)
4, 4, 0.0039,  1,   1
1.0
   144
0.103
0.103
SOURCE 5          (414)
5, 5, 1.0000,  1,   2
1.0,  1.0
 2184,    736
0.001,   0.000
0.001,   0.000
SOURCE 6          (417-418)
6, 6, 0.0031,  1,   5
```

1.0, 1.0, 1.0, 1.0, 1.0
2, 190, 1, 1, 1
0.115, 0.092, 0.095, 0.762, 0.095
0.115, 0.092, 0.095, 0.762, 0.095
SOURCE 7 (419)
7, 7, 0.0024, 1, 2
1.0, 1.0
2, 262
0.0011, 0.0010
0.0011, 0.0010
SOURCE 8 (421)
8, 8, 0.0027, 1, 5
1.0, 1.0, 1.0, 1.0, 1.0
2, 262, 1, 1, 1
0.044, 0.040, 0.054, 0.431, 0.054
0.044, 0.040, 0.054, 0.431, 0.054
SOURCE 9 (425)
9, 9, 0.0014, 1, 5
1.0, 1.0, 1.0, 1.0, 1.0
2, 262, 1, 1, 1
0.356, 0.285, 0.054, 0.431, 0.054
0.356, 0.285, 0.054, 0.431, 0.054
SOURCE 10 (428)
10, 10, 0.0002, 1, 1
1.0
120
0.334
0.334
SOURCE 11 (430)
11, 11, 0.0140, 1, 2
1.0, 1.0
2, 262
0.081, 0.078
0.081, 0.078
SOURCE 12 (431)
12, 12, 0.0140, 1, 2
1.0, 1.0
2, 262
0.010, 0.009
0.010, 0.009
SOURCE 13 (3601)
13, 13, 0.0050, 1, 6
1.0, 1.0, 1.0, 1.0, 1.0, 1.0
3, 48, 12, 1, 1, 1
2.555, 0.034, 0.645, 1.170, 9.356, 1.170
2.555, 0.000, 0.645, 1.170, 9.356, 1.170
SOURCE 14 (3603)
14, 14, 0.0050, 1, 6
1.0, 1.0, 1.0, 1.0, 1.0, 1.0
3, 48, 12, 1, 1, 1

2.555, 0.034, 0.645, 1.170, 9.356, 1.170
2.555, 0.000, 0.645, 1.170, 9.356, 1.170
SOURCE 15 (3604)
15, 15, 0.0050, 1, 6
1.0, 1.0, 1.0, 1.0, 1.0, 1.0
3, 48, 12, 1, 1, 1
2.555, 0.034, 0.645, 1.170, 9.356, 1.170
2.555, 0.000, 0.645, 1.170, 9.356, 1.170
SOURCE 16 (5 GALLON SPILL)
16, 16, .0002283, 1, 2
1.0, 1.0
1, 1
1.5531, 0.5177
1.5531, 0.5177
SOURCE 17 (10 GALLON SPILL)
17, 17, .0001142, 1, 2
1.0, 1.0
1, 1
3.1062, 1.0354
3.1062, 1.0354

********** Example TOXST Output File **********

1

********** INTERMITTENT RELEASE TOXICS EXPECTED EXCEEDANCES SYSTEM **********

TOXX, September 1993 Version
(IBM PC Compatible Version)
Sullivan Environmental Consulting, Inc.

****

Date and time of run: 10- 6-1993 17:34

---------- User Input Information ----------

Title of run	: CASE STUDY OPERATIONS - 8-HOUR - 1984-1985 DIST
# simulations per meteorological year	: 200
# emission sources	: 17
# receptors printed per row	: 18
Averaging period (hours)	: 8
Number of source group batch operations	: 17
Only source groups combined output Yes = 1	: 1
Emissions mutually exclusive? Yes = 1	: 0
Representative ISCST2 output file name	: ISC8H84.OUT
Metyears of requested dispersion results	: 1984 1985
	Output only expected exceedances averaged over all meteorological years.
	Find expected exceedances of multiple thresholds.

Pollutant and Threshold Information:

Poll. Id #:	Pollutant Name:	Background (ug/m3):	Thrs. Id #:	Health Effect Name:	Threshold (ug/m3):
1	COMPOSITE	.00	1	THRESHOLD #1	385.00
			2	THRESHOLD #2	550.00
			3	THRESHOLD #3	900.00
			4	THRESHOLD #4	1350.00
			5	THRESHOLD #5	1800.00
			6	THRESHOLD #6	2700.00

Emission Sources Information:

Source#	Group#	Source Name	Release Probability	Max Release Duration (hr)	Pollutant Name	Max Emiss Rate (g/s)	Batch Yes = 1	Number Em Cats
1	1	SOURCE 1	1.00000000	2920	COMPOSITE	.304	1	2
Dist Prob: >		1.00000 1.00000						
Dist Valid (hrs): >		2184 736						
AM Emiss Rate:		.304 .000						
PM Emiss Rate:		.304 .000						
2	2	SOURCE 2	1.00000000	2920	COMPOSITE	.027	1	2
Dist Prob: >		1.00000 1.00000						
Dist Valid (hrs): >		2184 736						
AM Emiss Rate:		.027 .000						
PM Emiss Rate:		.027 .000						
3	3	SOURCE 3	.00340000	144	COMPOSITE	.955	1	1
Dist Prob: >		1.00000						
Dist Valid (hrs): >		144						
AM Emiss Rate:		.955						
PM Emiss Rate:		.955						
4	4	SOURCE 4	.00390000	144	COMPOSITE	.103	1	1
Dist Prob: >		1.00000						
Dist Valid (hrs): >		144						
AM Emiss Rate:		.103						
PM Emiss Rate:		.103						
5	5	SOURCE 5	1.00000000	2920	COMPOSITE	.001	1	2

Dist Prob: >		1.00000	1.00000							
Dist Valid (hrs): >		2184	736							
AM Emiss Rate:		.001	.000							
PM Emiss Rate:		.001	.000							
6	6	SOURCE 6		.00310000		195	COMPOSITE	.762	1	5
Dist Prob: >		1.00000	1.00000	1.00000	1.00000	1.00000				
Dist Valid (hrs): >		2	190	1	1	1				
AM Emiss Rate:		.115	.092	.095	.762	.095				
PM Emiss Rate:		.115	.092	.095	.762	.095				
7	7	SOURCE 7		.00240000		264	COMPOSITE	.001	1	2
Dist Prob: >		1.00000	1.00000							
Dist Valid (hrs): >		2	262							
AM Emiss Rate:		.001	.001							
PM Emiss Rate:		.001	.001							
8	8	SOURCE 8		.00270000		267	COMPOSITE	.431	1	5
Dist Prob: >		1.00000	1.00000	1.00000	1.00000	1.00000				
Dist Valid (hrs): >		2	262	1	1	1				
AM Emiss Rate:		.044	.040	.054	.431	.054				
PM Emiss Rate:		.044	.040	.054	.431	.054				
9	9	SOURCE 9		.00140000		267	COMPOSITE	.431	1	5
Dist Prob: >		1.00000	1.00000	1.00000	1.00000	1.00000				
Dist Valid (hrs): >		2	262	1	1	1				
AM Emiss Rate:		.356	.285	.054	.431	.054				
PM Emiss Rate:		.356	.285	.054	.431	.054				
10	10	SOURCE 10		.00020000		120	COMPOSITE	.334	1	1
Dist Prob: >		1.00000								
Dist Valid (hrs): >		120								
AM Emiss Rate:		.334								
PM Emiss Rate:		.334								
11	11	SOURCE 11		.01400000		264	COMPOSITE	.081	1	2
Dist Prob: >		1.00000	1.00000							
Dist Valid (hrs): >		2	262							
AM Emiss Rate:		.081	.078							
PM Emiss Rate:		.081	.078							
12	12	SOURCE 12		.01400000		264	COMPOSITE	.010	1	2
Dist Prob: >		1.00000	1.00000							
Dist Valid (hrs): >		2	262							
AM Emiss Rate:		.010	.009							
PM Emiss Rate:		.010	.009							
13	13	SOURCE 13		.00500000		66	COMPOSITE	9.356	1	6
Dist Prob: >		1.00000	1.00000	1.00000	1.00000	1.00000	1.00000			
Dist Valid (hrs): >		3	48	12	1	1	1			
AM Emiss Rate:		2.555	.034	.645	1.170	9.356	1.170			
PM Emiss Rate:		2.555	.000	.645	1.170	9.356	1.170			
14	14	SOURCE 14		.00500000		66	COMPOSITE	9.356	1	6
Dist Prob: >		1.00000	1.00000	1.00000	1.00000	1.00000	1.00000			
Dist Valid (hrs): >		3	48	12	1	1	1			
AM Emiss Rate:		2.555	.034	.645	1.170	9.356	1.170			
PM Emiss Rate:		2.555	.000	.645	1.170	9.356	1.170			
15	15	SOURCE 15		.00500000		66	COMPOSITE	9.356	1	6
Dist Prob: >		1.00000	1.00000	1.00000	1.00000	1.00000	1.00000			
Dist Valid (hrs): >		3	48	12	1	1	1			
AM Emiss Rate:		2.555	.034	.645	1.170	9.356	1.170			
PM Emiss Rate:		2.555	.000	.645	1.170	9.356	1.170			
16	16	SOURCE 16		.00022830		2	COMPOSITE	1.553	1	2
Dist Prob: >		1.00000	1.00000							
Dist Valid (hrs): >		1	1							
AM Emiss Rate:		1.553	.518							
PM Emiss Rate:		1.553	.518							
17	17	SOURCE 17		.00011420		2	COMPOSITE	3.106	1	2
Dist Prob: >		1.00000	1.00000							
Dist Valid (hrs): >		1	1							
AM Emiss Rate:		3.106	1.035							
PM Emiss Rate:		3.106	1.035							

---------- Dispersion Model Results Data Information ----------

Datafile Headers:

Title of dispersion model result data	: CASE STUDY / NORMALIZED EMISSIONS - 1984
Year modeled	: 1984
# sources	: 17
# receptors	: 36
# averaging periods modeled in ISCST2	: 8784
Exex output by recptr? (0=no,1=pol,2=rect)	: 0
# radials (polar) or rows (rect)	: 0
Cut-off value used in ISCST2 (g/m3)	: 2.0700E-06

Title of dispersion model result data	: CMA CASE STUDY / NORMALIZED EMISSIONS - 1985
Year modeled	: 1985
# sources	: 17
# receptors	: 36
# averaging periods modeled in ISCST2	: 8760
Exex output by recptr? (0=no,1=pol,2=rect)	: 0
# radials (polar) or rows (rect)	: 0
Cut-off value used in ISCST2 (g/m3)	: 2.0700E-06

1

---------- Messages from Subroutine Setup (may be blank) ----------

1

---------- Dispersion Model Results (DMR) Read/Screened ----------

DMR File #	Year Modeled	Total # Data in File	# Saved for Calculation
1	1984	424711	316897
2	1985	423657	314449
	Totals:	848368	631346

Lowest value any chi/q may take and still cause exceedance of a threshold is (X/Q)min = 1.064495E-05 s/m3.

TABLE 1

Average Exceedances of Pollutant: COMPOSITE
Above threshold number 1 (> 385.00 ug/m3)
All source groups combined.
Year: 1984-1985

Y-Coordinate (m)	X-Coordinate (m)																
	45808.7	45834.7	45872.7	45907.7	45915.0	45950.8	45990.0	46030.3	46055.2	46093.1	46141.2	46161.4	46183.5	46227.2	46262.5	46292.9	46331.6	46375.0
820563.0																		
820556.0																		
820555.0																		
820545.0																		.0
820115.0																		
819886.0												.0						
819554.0									.0									
819536.0																		
819342.0							.0											
819218.0																		
819181.0						.1												
819067.0				.1														
818993.0																		
818965.0			.1															
818872.0		.1																
818818.0																		
818775.0	.2																	
818675.0					.2													
818614.0								.3										
818573.0											.7							
818545.0																		
818540.0																		
818514.0													1.6					

818499.0	1.8
818480.0	1.3
818450.0	.9
818429.0	.7
818419.0	
818415.0	.6
818370.0	
818318.0	
818285.0	
818259.0	
818147.0	
818145.0	

TABLE 1 (continued)

Average Exceedances of Pollutant: COMPOSITE
Above threshold number 1 (> 385.00 ug/m3)
All source groups combined.
Year: 1984-1985

Y-Coordinate (m)	X-Coordinate (m)															
	46428.8	46505.0	46615.0	46706.7	46818.5	47007.0	47050.5	47059.0	47079.8	47113.6	47155.0	47187.5	47248.9	47314.6	47401.5	47465.0	47583.4
820563.0																.0	
820556.0				.0													
820555.0								.0									
820545.0																	
820115.0																	.0
819886.0																	
819554.0																	
819536.0															.0		
819342.0																	
819218.0														.0			
819181.0																	
819067.0																	
818993.0													.0				
818965.0																	
818872.0																	
818818.0												.1					
818775.0																	
818675.0											.0						
818614.0																	
818573.0																	
818545.0										.0							
818540.0																	

818514.0									
818499.0									
818480.0									
818450.0									
818429.0									
818419.0									.0
818415.0									
818370.0	.6								
818318.0		.3							
818285.0							.0		
818259.0			.1						
818147.0					.1				
818145.0						.0			

Number receptors with at least one exceedance: 4
Total exceedances in above grid: 11.47990
Average exceedances per receptor in above grid: .31889 exceedances per receptor.

TABLE 2

Average Exceedances of Pollutant: COMPOSITE
Above threshold number 2 (> 550.00 ug/m3)
All source groups combined.
Year: 1984-1985

Y-Coordinate (m)	X-Coordinate (m)																
	45808.7	45834.7	45872.7	45907.7	45915.0	45950.8	45990.0	46030.3	46055.2	46093.1	46141.2	46161.4	46183.5	46227.2	46262.5	46292.9	46331.6	46375.0
820563.0																		
820556.0																		
820555.0																		
820545.0																		.0
820115.0																		
819886.0												.0						
819554.0									.0									
819536.0																		
819342.0							.0											
819218.0																		
819181.0						.0												
819067.0				.0														
818993.0																		
818965.0			.0															
818872.0		.0																
818818.0																		
818775.0	.1																	
818675.0					.0													
818614.0								.1										
818573.0											.3							
818545.0																		
818540.0																		

818514.0	.7
818499.0	.7
818480.0	.6
818450.0	.5
818429.0	.3
818419.0	
818415.0	.3
818370.0	
818318.0	
818285.0	
818259.0	
818147.0	
818145.0	

TABLE 2 (continued)

Average Exceedances of Pollutant: COMPOSITE
Above threshold number 2 (> 550.00 ug/m3)
All source groups combined.
Year: 1984-1985

Y-Coordinate (m)	X-Coordinate (m)															
	46428.8	46505.0	46615.0	46706.7	46818.5	47007.0	47050.5	47059.0	47079.8	47113.6	47155.0	47187.5	47248.9	47314.6	47401.5	47465.0	47583.4
820563.0																.0	
820556.0				.0													
820555.0								.0									
820545.0																	
820115.0																	.0
819886.0																	
819554.0																	
819536.0															.0		
819342.0																	
819218.0														.0			
819181.0																	
819067.0																	
818993.0													.0				
818965.0																	
818872.0																	
818818.0												.0					
818775.0																	
818675.0											.0						
818614.0																	
818573.0																	
818545.0										.0							
818540.0																	

818514.0									
818499.0									
818480.0									
818450.0									
818429.0									
818419.0									.0
818415.0									
818370.0	.4								
818318.0		.1							
818285.0							.0		
818259.0			.0						
818147.0					.0				
818145.0						.0			

Number receptors with at least one exceedance: 0
Total exceedances in above grid: 4.94975
Average exceedances per receptor in above grid: .13749 exceedances per receptor.

TABLE 3

Average Exceedances of Pollutant: COMPOSITE
Above threshold number 3 (> 900.00 ug/m3)
All source groups combined.
Year: 1984-1985

Y-Coordinate (m)	X-Coordinate (m)															
	45808.7	45834.7	45872.7	45907.7	45915.0	45950.8	45990.0	46030.3	46055.2	46093.1	46141.2	46161.4	46183.5	46227.2	46262.5	46292.9	46331.6	46375.0
820563.0																		
820556.0																		
820555.0																		
820545.0																		.0
820115.0																		
819886.0												.0						
819554.0									.0									
819536.0																		
819342.0							.0											
819218.0																		
819181.0						.0												
819067.0				.0														
818993.0																		
818965.0			.0															
818872.0		.0																
818818.0																		
818775.0	.0																	
818675.0					.0													
818614.0								.0										
818573.0										.1								
818545.0																		
818540.0											1							

818514.0	.2
818499.0	.2
818480.0	.2
818450.0	.1
818429.0	.1
818419.0	
818415.0	.1
818370.0	
818318.0	
818285.0	
818259.0	
818147.0	
818145.0	

TABLE 3 (continued)

Average Exceedances of Pollutant: COMPOSITE
Above threshold number 3 (> 900.00 ug/m3)
All source groups combined.
Year: 1984-1985

Y-Coordinate (m)	X-Coordinate (m)															
	46428.8	46505.0	46615.0	46706.7	46818.5	47007.0	47050.5	47059.0	47079.8	47113.6	47155.0	47187.5	47248.9	47314.6	47401.5	47465.0	47583.4
820563.0																.0	
820556.0				.0													
820555.0								.0									
820545.0																	
820115.0																	.0
819886.0																	
819554.0																	
819536.0															.0		
819342.0																	
819218.0														.0			
819181.0																	
819067.0																	
818993.0													.0				
818965.0																	
818872.0																	
818818.0												.0					
818775.0																	
818675.0											.0						
818614.0																	
818573.0																	
818545.0										.0							
818540.0																	

818514.0									
818499.0									
818480.0									
818450.0									
818429.0									
818419.0									.0
818415.0									
818370.0	.1								
818318.0		.0							
818285.0							.0		
818259.0			.0						
818147.0					.0				
818145.0						.0			

Number receptors with at least one exceedance: 0
Total exceedances in above grid: 1.12814
Average exceedances per receptor in above grid: .03134 exceedances per receptor.

TABLE 4

Average Exceedances of Pollutant: COMPOSITE
Above threshold number 4 (> 1350.00 ug/m3)
All source groups combined.
Year: 1984-1985

Y-Coordinate (m)	X-Coordinate (m)																
	45808.7	45834.7	45872.7	45907.7	45915.0	45950.8	45990.0	46030.3	46055.2	46093.1	46141.2	46161.4	46183.5	46227.2	46262.5	46292.9	46331.6	46375.0
820563.0																		
820556.0																		
820555.0																		
820545.0																		.0
820115.0																		
819886.0												.0						
819554.0									.0									
819536.0																		
819342.0							.0											
819218.0																		
819181.0						.0												
819067.0				.0														
818993.0																		
818965.0			.0															
818872.0		.0																
818818.0																		
818775.0	.0																	
818675.0					.0													
818614.0								.0										
818573.0											.0							
818545.0																		
818540.0																		

818514.0	.0
818499.0	.0
818480.0	.0
818450.0	.0
818429.0	.0
818419.0	
818415.0	.0
818370.0	
818318.0	
818285.0	
818259.0	
818147.0	
818145.0	

TABLE 4 (continued)

Average Exceedances of Pollutant: COMPOSITE
Above threshold number 4 (> 1350.00 ug/m3)
All source groups combined.
Year: 1984-1985

Y-Coordinate (m)	X-Coordinate (m)															
	46428.8	46505.0	46615.0	46706.7	46818.5	47007.0	47050.5	47059.0	47079.8	47113.6	47155.0	47187.5	47248.9	47314.6	47401.5	47465.0	47583.4
820563.0																.0	
820556.0				.0													
820555.0								.0									
820545.0																	
820115.0																	.0
819886.0																	
819554.0																	
819536.0															.0		
819342.0																	
819218.0														.0			
819181.0																	
819067.0																	
818993.0													.0				
818965.0																	
818872.0																	
818818.0												.0					
818775.0																	
818675.0											.0						
818614.0																	
818573.0																	
818545.0										.0							
818540.0																	

818514.0									
818499.0									
818480.0									
818450.0									
818429.0									
818419.0									.0
818415.0									
818370.0	.0								
818318.0		.0							
818285.0							.0		
818259.0			.0						
818147.0					.0				
818145.0						.0			

Number receptors with at least one exceedance: 0
Total exceedances in above grid: .13065
Average exceedances per receptor in above grid: .00363 exceedances per receptor.

TABLE 5

Average Exceedances of Pollutant: COMPOSITE
Above threshold number 5 (> 1800.00 ug/m3)
All source groups combined.
Year: 1984-1985

Y-Coordinate (m)	X-Coordinate (m)																
	45808.7	45834.7	45872.7	45907.7	45915.0	45950.8	45990.0	46030.3	46055.2	46093.1	46141.2	46161.4	46183.5	46227.2	46262.5	46292.9	46331.6	46375.0
820563.0																		
820556.0																		
820555.0																		
820545.0																		.0
820115.0																		
819886.0												.0						
819554.0									.0									
819536.0																		
819342.0							.0											
819218.0																		
819181.0						.0												
819067.0				.0														
818993.0																		
818965.0			.0															
818872.0		.0																
818818.0																		
818775.0	.0																	
818675.0					.0													
818614.0								.0										
818573.0										.0								
818545.0																		
818540.0											.0							

818514.0	.0
818499.0	.0
818480.0	.0
818450.0	.0
818429.0	.0
818419.0	
818415.0	.0
818370.0	
818318.0	
818285.0	
818259.0	
818147.0	
818145.0	

TABLE 5 (continued)

Average Exceedances of Pollutant: COMPOSITE
Above threshold number 5 (> 1800.00 ug/m3)
All source groups combined.
Year: 1984-1985

Y-Coordinate (m)	X-Coordinate (m)															
	46428.8	46505.0	46615.0	46706.7	46818.5	47007.0	47050.5	47059.0	47079.8	47113.6	47155.0	47187.5	47248.9	47314.6	47401.5	47465.0	47583.4
820563.0																.0	
820556.0				.0													
820555.0								.0									
820545.0																	
820115.0																	.0
819886.0																	
819554.0																	
819536.0															.0		
819342.0																	
819218.0														.0			
819181.0																	
819067.0																	
818993.0													.0				
818965.0																	
818872.0																	
818818.0												.0					
818775.0																	
818675.0											.0						
818614.0																	
818573.0																	
818545.0										.0							
818540.0																	

818514.0							
818499.0							
818480.0							
818450.0							
818429.0							
818419.0							.0
818415.0							
818370.0	.0						
818318.0		.0					
818285.0						.0	
818259.0			.0				
818147.0				.0			
818145.0					.0		

Number receptors with at least one exceedance: 0
Total exceedances in above grid: .05025
Average exceedances per receptor in above grid: .00140 exceedances per receptor.

TABLE 6

Average Exceedances of Pollutant: COMPOSITE
Above threshold number 6 (> 2700.00 ug/m3)
All source groups combined.
Year: 1984-1985

Y-Coordinate (m)	X-Coordinate (m)															
	45808.7	45834.7	45872.7	45907.7	45915.0	45950.8	45990.0	46030.3	46055.2	46093.1	46141.2	46161.4	46183.5	46227.2	46262.5	46292.9	46331.6	46375.0
820563.0																		
820556.0																		
820555.0																		
820545.0																		.0
820115.0																		
819886.0												.0						
819554.0									.0									
819536.0																		
819342.0							.0											
819218.0																		
819181.0						.0												
819067.0				.0														
818993.0																		
818965.0			.0															
818872.0		.0																
818818.0																		
818775.0	.0																	
818675.0					.0													
818614.0								.0										
818573.0										.0								
818545.0																		
818540.0																		

818514.0	.0
818499.0	.0
818480.0	.0
818450.0	.0
818429.0	.0
818419.0	
818415.0	.0
818370.0	
818318.0	
818285.0	
818259.0	
818147.0	
818145.0	

TABLE 6 (continued)

Average Exceedances of Pollutant: COMPOSITE
Above threshold number 6 (> 2700.00 ug/m3)
All source groups combined.
Year: 1984-1985

Y-Coordinate (m)	X-Coordinate (m)															
	46428.8	46505.0	46615.0	46706.7	46818.5	47007.0	47050.5	47059.0	47079.8	47113.6	47155.0	47187.5	47248.9	47314.6	47401.5	47465.0	47583.4
820563.0																.0	
820556.0				.0													
820555.0								.0									
820545.0																	
820115.0																	.0
819886.0																	
819554.0																	
819536.0															.0		
819342.0																	
819218.0														.0			
819181.0																	
819067.0																	
818993.0													.0				
818965.0																	
818872.0																	
818818.0												.0					
818775.0																	
818675.0											.0						
818614.0																	
818573.0																	
818545.0										.0							
818540.0																	

818514.0									
818499.0									
818480.0									
818450.0									
818429.0									
818419.0									.0
818415.0									
818370.0	.0								
818318.0		.0							
818285.0							.0		
818259.0			.0						
818147.0					.0				
818145.0						.0			

Number receptors with at least one exceedance: 0
Total exceedances in above grid: .01005
Average exceedances per receptor in above grid: .00028 exceedances per receptor.

TABLE 7

Number of exceedances within each HET category. (Years: 1984-1985)
All sources combined

Rec #	X-Coord	Y-Coord	(ug/m^3) 0.0- 385.0	385.0- 550.0	550.0- 900.0	900.0- 1350.0	1350.0- 1800.0	1800.0- 2700.0	>2700.0
1	46706.70	820556.00	1095.00	.00	.00	.00	.00	.00	.00
2	47059.00	820555.00	1095.00	.00	.00	.00	.00	.00	.00
3	47465.00	820563.00	1095.00	.00	.00	.00	.00	.00	.00
4	47583.40	820115.00	1095.00	.00	.00	.00	.00	.00	.00
5	47401.50	819536.00	1094.99	.01	.00	.00	.00	.00	.00
6	47314.60	819218.00	1094.99	.01	.00	.00	.00	.00	.00
7	47248.90	818993.00	1094.98	.02	.01	.00	.00	.00	.00
8	47187.50	818818.00	1094.94	.06	.00	.00	.00	.00	.00
9	47155.00	818675.00	1094.97	.02	.00	.00	.00	.00	.00
10	47113.60	818545.00	1094.98	.02	.00	.00	.00	.00	.00
11	47079.80	818419.00	1094.96	.02	.02	.00	.00	.00	.00
12	47050.50	818285.00	1094.98	.02	.00	.00	.00	.00	.00
13	47007.00	818145.00	1094.98	.02	.00	.00	.00	.00	.00
14	46818.50	818147.00	1094.95	.04	.02	.00	.00	.00	.00
15	46615.00	818259.00	1094.91	.06	.01	.01	.01	.00	.00
16	46505.00	818318.00	1094.66	.19	.14	.01	.00	.00	.00
17	46428.80	818370.00	1094.38	.25	.26	.10	.01	.00	.00
18	46375.00	818415.00	1094.38	.35	.21	.07	.00	.00	.00
19	46331.60	818429.00	1094.27	.43	.23	.08	.00	.00	.00
20	46292.90	818450.00	1094.11	.39	.36	.12	.01	.01	.00
21	46262.50	818480.00	1093.70	.69	.45	.14	.01	.01	.01
22	46227.20	818499.00	1093.17	1.09	.54	.18	.02	.00	.00
23	46183.50	818514.00	1093.44	.89	.49	.16	.02	.01	.00
24	46141.20	818540.00	1093.57	.75	.58	.10	.00	.00	.00
25	46093.10	818573.00	1094.31	.38	.26	.04	.01	.01	.00
26	46030.30	818614.00	1094.70	.19	.11	.00	.00	.00	.00
27	45915.00	818675.00	1094.77	.18	.05	.00	.00	.00	.00
28	45808.70	818775.00	1094.84	.11	.05	.00	.00	.00	.00
29	45834.70	818872.00	1094.90	.08	.01	.00	.00	.00	.00
30	45872.70	818965.00	1094.89	.09	.02	.00	.00	.00	.00
31	45907.70	819067.00	1094.91	.08	.01	.00	.00	.00	.00
32	45950.80	819181.00	1094.91	.09	.01	.00	.00	.00	.00
33	45990.00	819342.00	1094.98	.01	.01	.00	.00	.00	.00
34	46055.20	819554.00	1095.00	.00	.00	.00	.00	.00	.00
35	46161.40	819886.00	1095.00	.00	.00	.00	.00	.00	.00
36	46375.00	820545.00	1095.00	.00	.00	.00	.00	.00	.00

Date and time of end: 10- 7-1993 13:57

REFERENCES

CMA, 1992: Chemical Manufacturers Association, Demonstration Version - Draft - Toxic Modeling System Short-Term (TOXST) User' Guide. Washington, DC, October, 1992.

USEPA, 1989: Hazardous Waste Treatment, Storage, and Disposal Facilities (TSDF) - Air Emissions Models. U.S. Environmental Protection Agency, EPA-450/3-87-026, Office of Air Quality Planning and Standards, Research Triangle Park, NC.

USEPA, 1992: A Tiered Modeling Approach for Assessing the Risks Due to Sources of Hazardous Air Pollutants. D. E. Guinnup, U.S. Environmental Protection Agency, EPA-450/4-92-001, Office of Air Quality Planning and Standards, Research Triangle Park, NC.

Whitmyre, G.K., L.B. Wakefield, D.A. Sullivan, and D.J. Hlinka, 1992: Evaluation of Modeling and Risk Assessment Methods for Peak Releases of Air Toxics. Prepared for Air Dispersion Work Group of the Chemical Manufacturers Association.

Sullivan, David A., Thomas G. Grosch, and Dennis J. Hlinka, "Case Study of TOXST Application," prepared by Sullivan Environmental for the Chemical Manufacturer's Association, June 1993.

Appendix B

TOXX Model Source Code

```
c    /*************************************************************/
c    /*                                                           */
c    /*   TOXX (TOxics eXpected eXceedances system for            */
c    /*         intermittent sources)                             */
c    /*                                                           */
c    /*   Host is IBM PC Compatible running MS-DOS.               */
c    /*   Compiler is Microsoft FORTRAN Version 5.1.              */
c    /*                                                           */
c    /*   Developed under contract for the                        */
c    /*   U.S. Environmental Protection Agency                    */
c    /*   Project manager, David E. Guinnup                       */
c    /*                                                           */
c    /*   Thomas G. Grosch                                        */
c    /*   September 1993                                          */
c    /*                                                           */
c    /*   Based on TOXX Version 1.12, which was derived from      */
c    /*   TOXX Version 1.00, by Jamie Pehling.                    */
c    /*                                                           */
c    /*************************************************************/

c==== TOXX Program Main Documentation ==================================
c
c Development log:
c ~~~~~~~~~~~~~~~
c
c Please write clean & consistent code!!!
c
cSE
cSE
c 930930 Thomas Grosch
cSE      TOXX Enhanced Model
cSE
cSE    New comment lines added to this demonstration model are
cSE    indicated by cSE.  New code and changes made to the existing
cSE    code are preceded and followed by cSE.
cSE
cSE    1.    Code broken into two .FOR files.  TOXX.FOR and TOXX2.FOR.
cSE    2.    Re-format of output tables.
cSE    3.    Addition of Batch operations option
cSE    3.    Addition of Mutually Exclusive option
cSE    3.    Addition of multiple hourly averaging option.
cSE    4.    Addition of AM and PM emission rates option.
cSE
cSE    This program is in need of a new random generator to provide
cSE    random numbers for emissions that occur less frequently than
cSE    once every three years (i.e. fuel/chemical spills)

c  10/5/92
c      Revision of TOXX.FOR to accommodate:
c  1.  the complete re-coding of the Industrial Source Complex Short Term
c      computer model (previously ISCST, now ISCST2 of 9/29/92) from which
c      TOXX obtains its input of dispersion concentrations, and
c  2.  the initiatives set forth in "A Tiered Modeling Approach For Assessing
c      The Risks Due To Sources Of Hazardous Air Pollutants"
c      (EPA-450/4-92-001)
c      D.E. Guinnup, March 1992
c
c
c 901115 Jamie Pehling
```

```
c        TOXX Version 1.12
c
c        Certain page banners so long that they wrapped on output
c        printout have been shortened.  Warning message file output
c        has been adjusted so that first column of output string is
c        always blank.  (This adjustment was not made for errors.)
c        Later (11/29/90), I fixed a format bug in the labeling of
c        the max # exex of a single additive threshold.
c
c
c 900914 Jamie Pehling
c        TOXX Version 1.11
c
c        Elimination of variables pertaining to the original, severely
c        deficient output handling scheme.  Output technique altered
c        so that page breaks are fixed and each exex table header
c        begins on a new page.
c
c
c 900911 Jamie Pehling
c        TOXX Version 1.10
c
c        Modification of TOXX 1.00 to reduce array storage requirement
c        for source group on/off switches.  This involved the following
c        modifications:
c
c        --elimination of variable nRand
c        --elimination of error condition #25
c        --creation of array tOUAvP
c        --minor restructuring of subroutine CaExEx
c        --complete recoding of subroutine OnOff
c
c
c 900904 Jamie Pehling
c        TOXX Version 1.00
c
c        Conversion of TOXXS Version 1.01 to IBM PC Compatible.
c
c
c 900101 Jamie Pehling
c        TOXXS Version 1.01
c
c        Conversion of TOXXS Version 1.0 to run on EPA NCC IBM 3090.
c
c
c 890811 Jamie Pehling
c        TOXXS Version 1.0
c
c        Conversion of ExExS Version 3.0 to handle more sources and
c        source groups, plus intermittent emissions.  (The current
c        program is no longer suited to variable SO2 emissions stemming
c        from variation in coal sulfur content. It is intended for
c        simulating continuous and batch emissions of hazardous air
c        pollutants)
c
c
c 890208 Thomas Stocking
c        ExExS Version 3.0
c        (comments paraphrased by Jamie Pehling)
c
```

```
c        Bug fixed in Model5 subroutine. Error was to pass exex array
c        to metyearly output subroutine, so that cumulative results
c        instead of metyearly results were printed. Array that should
c        have been passed is exexyr. Only the metyearly results appear
c        to have been affected.
c
c        (At some point a reference to the zeroeth element of an
c        array was also found and eliminated.)
c
c
c 860722 Paulus Irpan
c        ExExS Version 2.0
c
c        Modify & add paging scheme (memory <-> disk file) for the conc-
c        data buffers, using a tempfile.
c
c        The runfile size is reduced significantly, yet the program now
c        can handle infinite #conc-data where disk space is the limit!
c        The program maintains only a page of conc-data in memory, and
c        pages in & out to the tempfile (the save-data file) as needed.
c        This paging, using tempfile, is done only when the #conc-data
c        exceed the page/buffer size.
c
c        The run-time of this version is not much more than that of V1.0
c        (in-memory version) when few or no paging of data to tempfile
c        is needed!
c
c
c 860618 Paulus Irpan, Till E. Stoeckenius
c        ExExS Version 1.0
c
c        Initial design & coding of the program.
c
c        Based on the major concepts of EXEX,FASTXX,FXMINI programs by
c        Martin J. Hillyer, John P. Nordin, Till E. Stoeckenius at SAI.
c
c-----------------------------------------------------------------------
c
c TOXX Glossary:
c ~~~~~~~~~~~~~
c
c Throughout the TOXX files, the following terms and phrases
c are used interchangeably:
c
c averaging period = time between acquisition of adjacent DMR data pairs
c                  = period
c
c buffer = internal array containing screened dispersion model results
c        = any of the arrays iPers, iRecs, and concs
c
c buffered = stored in memory (as opposed to disk or other mass storage)
c
c data page = set of acMxBf DMR data pairs
c
c data pair = an encoded id datum giving period, receptor, and source
c             information, plus a corresponding chi/q value
c           = data = data point
c
c DMR = dispersion model results
c
```

```
c effhet = effective health effect threshold (g/m3)
c        = concentration of a particular pollutant required for a
c          certain health effect to appear, minus background
c          concentration of that pollutant
c
c exex = expected exceedance(s)
c
c g = grams
c
c group = TOXX source group = sg
c       = a set of emission sources which turn on and off at the
c         same time, and which typically have identical emission rates
c         for all pollutants
c
c het = health effect threshold
c     = concentration of a particular pollutant required for a certain
c       health effect to appear
c     = threshold
c
c impacts = concentrations at receptors resulting from toxic release
c           by emission sources
c
c m3 = cubic meters of atmosphere
c
c meteorological year = one calendar year's worth of DMR
c                     = year of dispersion data
c                     = calendar year
c                     = metyear = year
c
c normalized concentrations = chi/q values = DMR concentrations
c
c prescreened data = DMR pairs whose chi/q values passed screening by
c                    the preprocessor and are used as input to TOXX
c
c s = seconds
c
c screened data = DMR pairs whose chi/q values passed screening by
c                 TOXX and are retained for exex calculation
c
c simulation year = simulation = sampling = simyear
c
c TOXST = TOxics eXpected exceedances Short Term system
c
c ug = micrograms
c
c-----------------------------------------------------------------------
c
c Subprogram flow diagram:
c ~~~~~~~~~~~~~~~~~~~~~~~
c
c The following diagram gives an approximate sense of control flow
c among the various subroutines and functions of TOXX.  Not every
c path is followed on every execution, and some subprograms are called
c more than once.  Descriptions of what each subprogram does may be
c found at the start of the code for each subprogram.
c
c prog TOXX  ====> subr Error
c            ====> subr GetInp ====> subr Error
c            ====> subr ChkDat ====> func leap
c                              ====> subr Error
```

```
c            ====> subr SetUp  ====> subr SortTI
c                              ====> subr Error
c                              ====> subr Warng  ====> subr Error
c            ====> subr GetDta ====> subr Data1
c                              ====> subr Data2
c            ====> subr DoExEx ====> subr Header
cSE                            ====> subr CaExEx ====> subr BatPr
c                                                      ====> subr RandNo
c                                                        ====> subr Taus15
c                              ====> subr WrExEx ====> subr WrTab
cSE                                                    ====> subr WrTab1
cSE                                                    ====> subr Tabline
c
c-----------------------------------------------------------------------
c
c Identifier conventions:
c ~~~~~~~~~~~~~~~~~~~~~~
c
c Maximum identifier length: 6 characters.
c
c
c Upper and lower case letter usage:
c
c constants    : all upper case
c
c functions,   : first char lower case, all others lower case, except for
c variables      any character corresponding to the first appearance of
c                a new word in the identifier; ex: for a variable called
c                "actual maximum bufferable", the identifier is "acMxBf"
c
c subroutines,: for subprograms coded specifically for TOXX, the
c programs       convention is the same as for variables, except first
c                char is uppercase; for subprograms native to Microsoft
c                FORTRAN, the first char is not uppercase
c
c
c Integer variable naming conventions:
c
c Most integer variables containing some total # of things begin with
c "n": nSrc, nYear.  Integer variables which serve as indexes, loop
c counters, etc., typically begin with "i": iRec, iGrp.
c
c
c Calling program/subroutine variable naming conventions:
c
c Variables are named so as to convey the most information on their
c contents that is possible in 6 characters.  In subprograms,
c however, actual arguments are USUALLY assigned to formal arguments
c with names different from those used in the calling program,
c especially when subroutines make unusual use of arguments that differs
c substantially
c from usage in the calling program.  Usually the identifier for a
c formal argument in a subroutine or function, when different from that
c of the corresponding actual argument, is a shortened form of the
c actual argument's identifier in the calling routine (iRec --> iR, for
c example), although sometimes a type change is used (FUPRT --> fuPrt).
c
c Variables extracted directly from common blocks via an include file
c have the same identifiers used throughout (except for subroutine
c CaExEx; see documentation in CaExEx header).
```

```
c
c-----------------------------------------------------------------------
c
c User input file map:
c ~~~~~~~~~~~~~~~~~~
c
c Format is list-directed/free-format.
c
c Record   Variable  Descriptions
c
c    1     title     Title of run (A80).
c
c    2     nHET      Number of toxic pollutant health effect thresholds
c                    whose exex are to be predicted.
c          hET ()    Numeric values of health effect thresholds (ug/m3).
c                    In general, reducing these values will increase
c                    run-time, since more chi/q data will pass the TOXX
c                    screening procedure in the Data* subroutines.
c                    nPol=1 always, each het present corresponds to
c                    a different health effect of the same pollutant.
c
c    3     nPol      Number of distinct chemical pollutants present
c                    in emissions of the sources under analysis.
cSE                  Must be 1.
c          bkgd      Background concentrations for pollutant in
c                    area under study (ug/m3).
c
c                    Note that the record numbers occupied by hName and
c                    pName data depend on nHET and nPol.  The numbers
c                    given below reflect the current maximum allowed
c                    value (MHET = 6).
c
c    4-9   hName ()  Health effect threshold names (A20).
c
c    10-15 pName ()  Pollutant names (A20).
c
c    16    nSim      # simulations to be performed for each metyear.
c          nSrc      # TOXX emission sources.
cSE        iAvPer    Averaging period (hours).  Restricted to 1, 2, 3,
cSE                  4, 6, 8, 12, and 24 hours.
cSE        nBat      # of emission sources that are batch operations.
c
cSE  17    MRecPr    # of X-coordinates to print horizontally across
cSE                  output tables.
c          lYrly     Flag whether to calc & print exex for each metyear.
c                    0 = no output by individual metyears, just for all
c                        metyears together
c                    1 = output exex for each metyear
cSE        MEEm      0 = emissions are not mutually exclusive.
cSE                  1 = mutually exclusive emissions are desired.
cSE        IGP       0 = print output tables for each source group and
cSE                      all source groups combined.
cSE                  1 = only print output tables for all source groups
cSE                      combined.
c
c    18    nYear     # meteorological years (# DMR datafiles).
c          iYear ()  The calendar years for which DMR data have been
c                    produced, in the read-in order of the DMR files.
c
cSE  19    ISCOuFn   name of ISCST2 output file name.
```

```
cSE
c                       Records 20-25 are repeated for ea. of nSrc sources.
c
c     20     sName ()   Source name (A20)
c
c     21     iSrc ()    Source #, does not have to be ordered with respect
c                       to source #s of other sources.
cSE          iGrp ()    Source group # of which source is a member.  All
cSE                     sources belonging to the same TOXX source group
cSE                     must follow one another.  TOXX source group numbers
cSE                     must increase in number as they are added to the
cSE                     input file.
c
c                       The values of the following three variables are the
c                       same for all members of a source group.  Hence some
c                       information is repeated when there is more than one
c                       source within any group.
c
c            probOn ()  Probability of source turning on, when off.
cSE          Batch ()   Switch indicating if this source is a batch
cSE                     process.  1 = yes; 0 = no.
cSE          IMAX ()    Number of emission distributions for this source.
cSE                     Maximum of 10.
cSE
cSE   22     dist (,)   Probability associated emission distribution
cSE                     will be selected for a non-batch process.
cSE   23     BatOn (,)  Time (hours) associated emission distribution
cSE                     is valid for.
cSE   24 BatRate(,,,)   Emission Rate for source for AM hours
cSE   25 BatRate(,,,)   Emission Rate for source for PM hoursce group
c
c-----------------------------------------------------------------------
c
c Prescreened Dispersion Model Results Datafile Map:
c ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
c
c These files do not contain the output of the dispersion model
c used to simulate atmospheric effects.  Rather, they contain an
c altered version of the DMR output, namely the output of the
c ExEx preprocessor.  The preprocessor performs a coarse prescreening
c of the DMR data, in which all chi/q values unlikely to produce an
c exceedance are dropped.
c
c The format of this data depends on the data I/O statements/routines
c you choose to use: standard FORTRAN formatted or binary I/O. TOXX
c expects binary I/O, but commented-out code exists to enable a
c formatted read if desired.  The ORGANIZATION of data should be
c identical to that shown below in either case, however.
c
c Record  Variable  Description
c
c     1     dTitle    Title of DMR data file.
c
c     2     iYr       The calendar year to which the modeled data
c                     corresponds.
c           nS        # sources, must be >= nSrc.
c
c                     The value of each of the following variables should
c                     be the same in all DMR files.
c
```

```
c          nR          # receptors.
c          nP          # averaging periods in metyear iYr; must be the
c                      number of iAvPer's in iYr.
c          iTab        Output table type (options are similar to those
c                      of lTab).
c          nX          # receptors per ring or row in physical receptor
c                      array
c          iDum ()     NIDUM integers not used by TOXX.  Change NIDUM in
c                      insert file if they are used in future.
c
c     3    pCutOf      Chi/Q cut value used in prescreening of data for
c                      metyear iYr.
c          rDum ()     NRDUM reals not used by TOXX.  Change NRDUM in
c                      insert file if they are used in the future.
c
c                      Records 4-5 are repeated as many times as needed.
c                      Each record pair must contain NPAIR data (value set
c                      in the Data* subroutines).  The last 2 records must
c                      be padded with 0's, so that number of data on each
c                      line of the file is the same.  TOXX expects data
c                      for a given period and receptor to be contiguous in
c                      the DMR files, although it may be in any order with
c                      respect to source #.
c
c     4    id ()       Identification data.  Each datum contains period,
c                      source, and receptor information.  Given a result
c                      for averaging period # pppp, for emissions from
c                      TOXX source # ss, taken at receptor # rrr, the
c                      value of the integer id datum would be ppppssrrr.
c                      Hence the respective maximum "coding" #s of periods,
c                      sources, and receptors are 9999, 99, and 999
c                      (Constant parameters MCPER, MCSRC, and MCREC are
c                      defined equal to these numbers, and serve as "hard"
c                      upper limits to "soft" constants MPER, MSRC, and
c                      MREC, respectively.  To adjust MCPER, MCSRC, and
c                      MCREC would require some nontrivial recoding.)
c
c     5    conc ()     Chi/Q normalized concentrations ( units of
c                      ( (g/m3) / (g/s) ), or (s/m3) ) corresponding
c                      to the ID codes in the previous record.
c
c-----------------------------------------------------------------------
c
c Program parameters:
c ~~~~~~~~~~~~~~~~~~
c
c These are constants in source code which set various limits.
c They can be changed to suit needs, as memory permits.
c Their values are set in the source code insert file TOXX.INS.
c
c     IDMXBF    Ideal max # DMR data that are allowed in memory.
c
c     MCPER,    Upper limits on MPER, MSRC, and MREC which exist because
c     MCREC,    of the coding of the id part of DMR data pairs.
c     MCSRC
c
c     MGRP      Max source group #; since source group numbers are
c               assumed to be consecutive positive integers, places
c               an effective limit on the # of source groups.
c
```

```
c    MHET      Max # of health effect thresholds for which exex may be
c              calculated, and maximum # distinct chemical pollutants
c              that can be processed by TOXX.
c
c    MPER      Max possible # of averaging periods in a year,
c              equivalent to the number of averaging periods of the
c              shortest duration (currently 1 hour) in a leap year.
c              Used to dimension the arrays of random numbers which
c              determine whether a source group's members are on or
c              off for each averaging period in a given metyear.
c
c    MREC      Max # receptors.
c
c    MSIM      Max # samplings in a Monte Carlo simulation for one
c              metyear (put another way, the max # of simyears per
c              metyear); limited to prevent runaways.
c
c    MSRC      Max # TOXX sources.
c
c    MYEAR     Max # metyears (and files) of DMR data.
c
c    NIDUM     # of integral dummy parameters currently available (i.e.,
c              currently unused) in headers of DMR files.
c
c    NRDUM     # of real dummy parameters currently available (i.e.,
c              currently unused) in headers of DMR files.
c
c-----------------------------------------------------------------------
c
c Memory Paging Scheme:
c ~~~~~~~~~~~~~~~~~~~~
c
c TOXX uses a virtual memory system coded by Paulus Irpan for ExExS
c Version 2.0 to remove any core limit on the number of DMR data that
c may be handled.
c
c Data are read from the preprocessed DMR datafiles on disk into
c two arrays, iDRA and cOQRA, NPAIR data at
c a time.  Chi/q values are accumulated for each source group and
c every period and receptor.  When all normalized concentration data
c for a given period and receptor has been accumulated, the total
c concentration is compared to the value of the variable cutoff.  If
c the concentration arising from at least one source group is greater
c than the value of cutoff, the concentrations for each source group
c are stored in the buffers, iPers, iRecs, and concs.  (This procedure
c is called "screening" the data, and is independent of the "pre-
c screening" of data performed by the precessor, which is far less
c analytical.)
c
c When the iPers, iRecs, and concs buffers become "full" (a condition
c defined conservatively by Irpan and not by the machine), the arrays
c are emptied to a temporary file on disk.  New raw data are read into
c the buffer arrays, then new screened data is put into iPers, iRecs,
c and concs, and the cycle proceeds as before. Indexes to the data
c stored on disk and in memory are maintained so that all subroutines
c can access the DMR data as needed.
c
c Concise descriptions of the variables used in memory paging appear
c in the insert file, TOXX.INS.  Verbose descriptions of certain
c variables appear below:
```

```
c
c IDMXBF: Ideal maximum number of DMR data pairs that may be bufferd.
c         IDMXBF is limited (by programmer choice) to the max
c         I/O record size in the version of FORTRAN being used,
c         divided by the size of one DMR data pair (max I/O record
c         size (bytes, or whatever unit) / (size of one element
c         iPers () + iRecs () + concs (), in same units), quantity minus
c         1.  In Prime FORTRAN 77,  the maximum I/O record size is
c         32 kbytes (1kbyte= 1024 bytes).  In order to
c         conserve memory, iPers and iRecs have been made arrays
c         of 2-byte integers.  The amount of memory occupied by
c         a single DMR data pair is then 2x 2 bytes (for one element
c         each of iPers and iRecs) + 4 bytes (for one element of
c         concs) = 8 bytes.  Hence IDMXBF = (32768 bytes/ (8 bytes/
c         data pair)) - 1 = 4095.  This is only an ideal max # in
c         that DMR data are saved in groups of nSG data pairs, one
c         for each source group.  If the number of source groups
c         divides IDMXBF with no remainder, then IDMXBF is the
c         actual max # of DMR data pairs that can be stored.  In
c         general, however, variable acMxBf is the actual max
c         #.
c
c nBuf   : Number of DMR data pairs which successfully passed screening
c         and are currently buffered in memory.
c
c nPage : Total number of data pages currently in use.  Within the
c         Data* subroutine, typically includes only full pages that have
c         been saved to disk.  After the last metyear has been pro-
c         cessed, however, nPage is updated to include the possibly
c         partially full final page, which is held in memory AND
c         saved to disk if there are too many data overall to fit in
c         the buffer arrays.  In short, nPage = 1 if all data can fit
c         in memory, in which case no data is paged to disk; else nPage
c         = number of pages saved to tempfile on disk (of which the last
c         page also sits in memory).
c
c iSavYr: Numerical indexes to ultimate DMR data of each metyear to
c         have passed the screening.  The index is merely by save
c         sequence, and carries no information itself about whether
c         a metyear's data is on disk or in memory.
c
c ioPage: Index of the data page currently held in memory.
c
c-----------------------------------------------------------------------
c
c Compiler- or computer-dependent Codes Sections:
c ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
c
c Certain items of the program may require modification when run on
c different machines or using different compilers.  These items are
c listed below. Some, but not all, are delimited in the code by 'c!'
c comment lines.
c
c    - I/O fileunits (parameterized in the main program).
c    - The directory in which the include file is stored.
c    - Date/time routines.
c    - The random # generator routines RandNo and Taus15, particularly
c      the binary and exclusive or function calls.
c    - Lines needed for formatted DMR I/O; as program stands now,
c      all this I/O is binary.
```

```
c     - Other code delimited by the comment symbol 'c!'.
c
c end main documentation
c


c==== program TOXX ===================================================
c
c Estimate the probability of expected exceedances of multiple hets
c from intermittent release of multiple toxics, using Monte Carlo
c simulation.
c
c--------------------------------------------------------------------
c
      program TOXX
c
c--------------------------------------------------------------------
c
c Declarations, etc.
c
c     Insert file.
c!
      include 'toxx.ins'
c!
c     Description of local variables.
c
c     integer
c     parameter (
c     .    FUHEM=10,          ! HEM input fileunit
c     .    FUINP=11,          ! user input fileunit
c     .    FUPRT=12,          ! print fileunit (output file)
c     .    FUTMP=13,          ! temp fileunit on disk containing saved
c     .                       ! DMR data
c     .    FUDAT=14           ! 1st of DMR data fileunits
c                             ! (nYear integers beginning at FUDAT and
c                             ! ending at FUDAT+nYear-1 must not be
c                             ! used as fileunits; they are already
c                             ! used by the program as the fileunits
c                             ! for DMR files.)
c     .)
c     character
c     .    fn*64              ! variable for user input of user
c                             ! input, DMR, and print filenames
c                             ! (pathnames)
c     integer
c     .    iFu,               ! index to current DMR data file
c     .    hour,              ! hour at run-time
c     .    min,               ! minute
c     .    sec,               ! second
c     .    sec100,            ! hundredth second (!)
c     .    year,              ! year at run-time
c     .    month,             ! month
c     .    day                ! day
c
      integer
     .    iFu, hour, min, sec, sec100, year, month, day,
     .    FUHEM,FUINP,FUPRT,FUTMP,FUDAT
      parameter
     .    (FUHEM=10,FUINP=11,FUPRT=12,FUTMP=13,FUDAT=14)
      character
```

```
     .    fn*64
c
c-----------------------------------------------------------------------
c
c Initial input of parameters.
c
c     Print out a small program banner.
c
      print '(A)', ' ---------------------------------------------'
      print '(A)', ' | TOXX ISCST2 TOXXFILE Compatible Version)  |'
      print '(A)', ' ---------------------------------------------'

c
c     Get filenames from user, and open files for user input params,
c     DMR data, and output.
c
      print '(/A)', ' Enter TOXX input parameter filename. >>'
      read '(A)', fn
      open (FUINP, file=fn, status='OLD')

      print '(/A)', ' Enter # of meteorological years. >>'
      read *, nYear

      print '(/A)',
     .    ' Enter chi/q TOXXFILE name (1 per metyear, 1 per line). >>'
      do 100 iFu = FUDAT, FUDAT+nYear-1
                  read '(A)', fn
                  open (iFu, file=fn, form='UNFORMATTED', status='OLD')
  100 continue

      print '(/A)', ' Enter TOXX text output filespec. >>'
      read '(A)', fn
      open (FUPRT, file=fn)

      print '(/A)', ' Enter TOXX HEM output filespec. >>'
      read '(A)', fn
      open (FUHEM, file=fn)

cSE
c
c-----------------------------------------------------------------------
c
c Output header.
c
c     Write banner messages and date-time of run.
c     Change or comment out the date-time section for other compilers.
c
      print *
      write (FUPRT,'(1H1///(3A))')
     .  ' ****************************************  TOXST, TOXICS ',
     .  'EXPECTED EXCEEDANCES SYSTEM, SHORT-TERM  ********************',
     .  '*********',
     .  ' ',' ',' ',
     .  '                                              TOXX  September',
     .  ' 1993 Version',' ',
     .  '                                                (IBM PC Compa',
     .  'tible Version)',' ',
     .  '                                                             ',
     .  '                    ',' ';
     .  ' ',' ',' ',
```

```
     .  ' ************************************************************',
     .  '************************************************************',
     .  '**********'
c!
c!    Output run date and time, using Microsoft date and time
c!    subroutines.
c!
      call getDat (year, month, day)
      call getTim (hour, min, sec, sec100)
      write (FUPRT, 1000)
     .    'Date and time of run: ', month, day, year, hour, min
 1000 format (/43X, A22, I2, '-', I2, '-', I4, '  ', I2, ':', I2)
c
c-----------------------------------------------------------------------
c
c Initial error check.
c
c     Perform error check on the maximum allowed values for # periods,
c     # sources, and # receptors.  These are set in TOXX.INS, but are
c     constrained by the technique used to code id information for chi/q
c     values (see map of DMR datafiles above).
c
      if (MPER .gt. MCPER) then
                  call Error (fuPrt, 1, MPER, MCPER, 0., 0.)
      elseif (MSRC .gt. MCSRC) then
                  call Error (fuPrt, 2, MSRC, MCSRC, 0., 0.)
      elseif (MREC .gt. MCREC) then
                  call Error (fuPrt, 3, MREC, MCREC, 0., 0.)
      endif
c
c-----------------------------------------------------------------------
c
c Code for exex run.
c
c     Input user parameters.
c
      call GetInp (FUINP, FUPRT)
c
c     Input DMR datafile headers and do some simple checks.
c
      call ChkDat (FUDAT, FUPRT, nP)
c
c     Prepare some other params needed for the run.
c
      call SetUp (FUPRT, FUTMP)
c
c     Read and screen DMR datafiles.
c
      call GetDta (FUDAT, FUPRT, FUTMP)
c
c     Computation and output of exex.
c
      call DoExEx (FUPRT, FUTMP, FUHEM, nP)
c
c     End output file and program.
c
cSE
      call getDat (year, month, day)
      call getTim (hour, min, sec, sec100)
      write (FUPRT, 1001)
```

```
     .    'Date and time of end: ', month, day, year, hour, min
 1001 format (/43X, A22, I2, '-', I2, '-', I4, '  ', I2, ':', I2.2)
cSE
      endfile (FUPRT)
      stop 'Ok. TOXX run finished normally.'
      end
c
c     end program TOXX
c


c==== subroutine GetInp ============================================
c
c Get user input params, set defaults, and prepare other params.
c Echo input info to printfile.
c
c Subroutine GetInp is called by program TOXX.
c
c-----------------------------------------------------------------------
c
      subroutine GetInp (fuInp, fuPrt)
c
c-----------------------------------------------------------------------
c
c Declarations, etc.
c
c     Description of passed parameters.
c
c     integer
c    .    fuInp, fuPrt    ! I/O fileunits
c
      integer
     .    fuInp, fuPrt
c
c     Insert file.
c!
      include 'toxx.ins'
c!
c     Description of local variables.
c
c     integer
c    .    i, j,           ! handy indexes
c
      integer
     .    i, j

cSE   character*20
cSE  .    tday            ! contains line header for output file.

      character*20
     .    tday
c
c-----------------------------------------------------------------------
c
c GetInp output header.
c
      print '(1X,A)', '>      GetInp'
      write (fuPrt,'(///3A/)')
     .  ' ------------------------------------------- User In',
     .  'put Information ----------------------------------------',
```

```
     .  '---------'
c
c-----------------------------------------------------------------------
c
c Handling of general run params.
c
c     Get general params from user input file.
c
      read (fuInp,'(A80)') title
      read (fuInp,*) nHET, (hET (i), i=1, nHET)
cSE   nPol can only equal 1 in this version.
      read (fuInp,*) nPol, bkgd
      nPol=1  ! hardwire # poll. to 1 incase user did not specify 1.
      do 30 i = 1, nHET
   30     read (fuInp,'(A20)') hName (i)
      do 40 i = 1, nPol
   40     read (fuInp,'(A20)') pName (i)
cSE
      read (fuInp,*) nSim, nSrc, iAvPer, nBat
cSE   No longer read in switch for lAdd or lTab in this version.
cSE   Added in the flag for source group combined output iGP
      read (fuInp,*) mRecPR, lYrly, MEEm, iGP
      lAdd=0  ! hardwire lAdd to 0.  New output tables don't need this.
      lTab=0  ! hardwire lTab to 0.
cSE
      read (fuInp,*) nYear, (iYear(i), i=1, nYear)

cSE   read in the name of the ISCST2 output file which contains receptor
cSE   location data.

      read (fuInp,'(A30)') ISCOuFn

cSE   Open the ISCST2 output file.
      open (1,file=ISCOuFn,form='formatted',status='old')

c
c     Echo these user inputs.
c
cSE
      write (fuPrt,'(4X, A46, A80/ 4X, A46, I4/ 4X, A46, 2X, I2/
     .       4X, A46, 2X, I2/ 4X, A46, 2X, I2/ 4X, A46, 2X, I2/
     .       4X, A46, 2X, I2/ 4X, A46, 2X, I2/ 4X, A46, 2X, A30/
     .       4X, A46, 6(I4, 3X))')
     .    'Title of run                                  : ', title,
     .    '# simulations per meteorological year         : ', nSim,
     .    '# emission sources                            : ', nSrc,
     .    '# receptors printed per row                   : ', mRecPR,
     .    'Averaging period (hours)                      : ', iAvPer,
     .    'Number of source group batch operations       : ', nBat,
     .    'Output only source groups combined? Yes = 1   : ', iGP,
     .    'Emissions mutually exclusive?  Yes = 1        : ', MEEm,
     .    'Representative ISCST2 output file name        : ', ISCOuFn,
     .    'Metyears of requested dispersion results      : ',
     .    (iYear(i), i=1, nYear)
cSE
c
c     Flag user options selected.
c
c
      if (lYrly .eq. 0) then
```

```
                write (fuPrt,'(50X, A, A)')
     .        'Output only expected exceedances averaged over all mete',
     .        'orological years.'
      elseif (lYrly .eq. 1) then
                write (fuPrt,'(50X, A, A)')
     .        'Output expected exceedances yearly and averaged over al',
     .        'l meteorological years.'
      endif
c
cSE   Modified to represent only 1 pollutant.
      if (nHET .eq. 1) then
                write (fuPrt, '(50X, A)')
     .        'Find expected exceedances of one threshold.'
      else
                write (fuPrt, '(50X, A)')
     .        'Find expected exceedances of multiple thresholds.'
      endif
c
c     Print title for pollutant and threshold information.
c
      write (fuPrt, '(//4X, A)')
     .    'Pollutant and Threshold Information:'
      write (fuPrt, '(/ 4X, A, A, 4X, A, A)')
     .    '---------------------- Pollutant Data ----------------------',
     .    '-',
     .    '--------------- Health Effect Threshold Data --------------',
     .    '-'
      write (fuPrt, '(4X, A11, 5X, A15, 10X, A19, 4X, A11, 5X, A19,
     .    7X, A18/)')
     .    'Poll. Id #:',
     .    'Pollutant Name:',
     .    'Background (ug/m3):',
     .    'Thrs. Id #:',
     .    'Health Effect Name:',
     .    'Threshold (ug/m3):'
c
c     Print pollutant and threshold information, side by side.
c     Allow for the case of one pollutant with multiple thresholds.
c
      do 50 i = 1, nHET
                if ( (i .eq. 1) .or. (nHET .eq. nPol) ) then
                    write (fuPrt, '(4X, I1, 15X, A20, 11X, F7.2, 10X, I1,
     .            15X, A20, 11X, F7.2)')
     .            i, pName (i), bkgd, i, hName (i), hET (i)
                  else
                    write (fuPrt, '(68X, I1, 15X, A20, 11X, F7.2)')
     .            i, hName (i), hET (i)
                  endif
   50 continue
c
c     Error check the general inputs that have been read so far.
c
      if (nHET .gt. MHET) call Error (fuPrt, 4, nHET, MHET, 0., 0.)
      if (nPol .gt. MHET) call Error (fuPrt, 5, nPol, MHET, 0., 0.)
      if ( (nPol .gt. 1) .and. (nHET .ne. nPol) )
     .    call Error (fuPrt, 6, nPol, nHET, 0., 0.)
c
      if (nSim .gt. MSIM) call Error (fuPrt, 7, nSim, MSIM, 0., 0.)
      if (nSrc .gt. MSRC) call Error (fuPrt, 8, nSrc, MSRC, 0., 0.)
      if (iAvPer.gt.24)
```

```
     .    call Error (fuPrt, 9, iAvPer, 0, 0., 0.)
c
      if (lYrly.ne.0 .and. lYrly.ne.1)
     .    call Error (fuPrt, 13, lYrly, 0, 0., 0.)
      if (nYear .gt. MYEAR)
     .    call Error (fuPrt, 14, nYear, MYEAR, 0., 0.)
c
c-----------------------------------------------------------------------
c
c Handling of emission sources information.
c
      write (fuPrt,'(//4X, A)') 'Emission Sources Information:'
c
c     Read the emission sources information.
c
cSE
      do 100 i = 1, nSrc
          read (fuInp,'(A)') sName(i)
          read (fuInp,*) iSrc(i), iGrp(i), probOn(i), Batch(iGrp(i)),
     .       iMax(iGrp(i))
                 timeOn(i) = 0
                 K=iGrp(i)
                 do 16 L=1,nPol
                     rate(K,L)=0.
   16     continue
cSE

cSE       Read in the probabilities that each emission distribution
cSE       will be selected for non-batch processes.

          read(fuInp,*) (dist(K,EmDist), EmDist=1,iMax(K))

cSE       Read in the amount of time each emission category will be valid.

                   read(fuInp,*) (BatOn(K,EmDist), EmDist=1,iMax(K))

cSE       Determine the amount of time a batch process will be on
cSE       once it is turned on for each source group.

          do 24 EmDist = 1,iMax(K)
   24         timeOn(K) = timeOn(K) + BatOn(K,EmDist)

cSE       When sun = 1, AM emission rates apply; when sun = 2, PM
cSE       emission rates apply.
cSE       AM hours: when it is usually light outside.

cSE
          do 22 sun = 1, 2  ! loop on AM and PM
              do 23 iPol = 1, nPol  ! loop on the 1 pollutant this ver.

cSE               Read in the emission rates.

                      read(fuInp,*) (BatRate(K,iPol,EmDist,sun),
     .                EmDist=1,iMax(K))
                  do 18 EmDist=1,iMax(K) ! loop on emission distributions.
cSE                   Determine max. emission rate for this source.
                          if(BatRate(K,iPol,EmDist,sun) .gt.
     .                    rate(K,iPol)) then

rate(K,iPol)=BatRate(K,iPol,EmDist,sun)
```

```
                              end if
   18              continue
   23            continue
   22        continue

  100 continue
cSE

c
c     Echo the sources information.
c
cSE
      write (fuPrt, '(T42, A, T56, A, T90, A, T105, A, T115, A)')
     .    'Release', 'Max Release', 'Max Emiss', 'Batch','Number'
      write (fuPrt, '(T5,A,T15,A,T24,A,T42,A,T56,A,T72,A,T90,A,T105,
     .    A,T115,A)')
     .    'Source#', 'Group#', 'Source Name',
     .    'Probability', 'Duration (hr)',
     .    'Pollutant Name', 'Rate (g/s)', 'Yes = 1', 'Em Cats'
      do 10, i = 1, nSrc
                  write (fuPrt,'(T7, I2, T17, I2, T24, A20,T45, F10.8, T59,
I4,
     .           T72, A20, T90, F8.3, T105, I2, T118, I2)')
     .         iSrc (i), iGrp (i), sName (i), probOn (i), timeOn (i),
     .         pName (1), rate (i,1), Batch (iGrp(i)), iMax (iGrp(i))

          write(fuPrt,'(T5,A12,T25,10f8.5)') 'Dist Prob: >',(dist(i,
     .         EmDist),EmDist=1,iMax(iGrp(i)))
          write(fuPrt,'(T5,A19,T25,10I8)') 'Dist Valid (hrs): >',
     .         (BatOn(i,EmDist),EmDist=1,iMax(iGrp(i)))
          do 15 j=1,nPol
              do 15 is=1,2
                  if(is .eq. 1) then
                  tday='AM Emiss Rate:'
               else
                  tday='PM Emiss Rate:'
               end if
               write(fuPrt, '(T5,A15,T25,10F8.3)') tday,(BatRate(i,
     .             j,k,is),k=1,iMax(iGrp(i)))
   15     continue
   10 continue
cSE

      write (fuPrt, '(A4)') '    '
      write (fuPrt, '(A4)') '    '
c
      return
      end
c     end subroutine GetInp



c==== subroutine ChkDat ============================================
c
c Get and check headers of the DMR datafiles.  Echo headers.
c
c Subroutine ChkDat is called by program TOXX.
c
c-------------------------------------------------------------------
c
```

```
      subroutine ChkDat (fuDat, fuPrt, nP)
c
c-----------------------------------------------------------------------
c
c Declarations, etc.
c
c     Description of passed parameters.
c
c     integer
c    .    fuDat, fuPrt  ! DMR data & print fileunits
c
      integer
     .    fuDat, fuPrt
c
c     Insert file.
c!
      include 'toxx.ins'
c!
c     Description of local variables.
c
c     character
c    .    dTitle*80     ! title of datafile
c     integer
c    .    iYr,          ! metyear modeled
c    .    nR0,          ! # receptors from previous DMR file
c    .    nPer,         ! # of periods in current metyear
c    .    nS, nR, nP,   ! # sources,receptors,periods
c    .    iTab,         ! output table type
c    .    nX,           ! # receptors per ring or row
c    .    iDum (NIDUM), ! NIDUM integers output by preprocessor,
c    .                  ! for future use
c    .    iFl,          ! index to current DMR data file
c    .    iFu,          ! current DMR data fileunit
c    .    i             ! handy index
c     real
c    .    pCutOf,       ! chi/q cut value used in data prescreen-
c                       ! ing in preprocessor, for current
c                       ! metyear's data
c    .    rDum (NRDUM)  ! NRDUM reals output by preprocessor, for
c                       ! future use
c
      character
     .    dTitle*80
      integer
     .    iYr, nR0, nPer, nS, nR, nP, iTab, nX, iFl, iFu, NYTOX

      real
     .    pCutOf
c
c     Functions.
c
c     integer
c    .    leap          ! function to determine if iYr is leap
c
      integer
     .    leap
c
c-----------------------------------------------------------------------
c
c ChkDat output header.
```

```
c
      print '(1X,A)', '>>     ChkDat'
      write (fuPrt,'(1H1///3A)')
     .  ' ---------------------------------------- Dispersion Model',
     .  ' Results Data Information  ---------------------------------',
     .  '---------'
      write (fuPrt,'(//4X, A)') 'Datafile Headers:'
c
c-----------------------------------------------------------------------
c
c Initialization.
c
      nR0 = -1
      pMxCut = -1.
c
c-----------------------------------------------------------------------
c
c Loop for each metyear (each dispersion model results file).
c
      do 100 iFl = 1, nYear
c
c         Fileunit for this datafile is in a unit incremental sequence
c         starting from the 1st fileunit fuDat.
c
                  iFu = fuDat + iFl - 1
c
c         Read the header data in binary form.
crti  Revisions for ISCST2 of 9/10/92 are noted with crti in cols 1-4:
crti  Read datafile created by ISCST2 using the TOXXFILE option.
crti   The header information remains almost the same except:
crti     NIDUM, number of dummy integer variables, has been changed from
crti       4 in old SISC.FOR to 3
crti     the integer variable NYTOX in ISCST2 has been included.  This
crti       is the number of y-coordinates (or directions) in the receptor grid
crti                                        (or radials    )
crti  Per ISCST2 documentation, iTab represents:
crti    iTab = 1 => one polar grid
crti    iTab = 2 => one Cartesian grid
crti    iTab = 0 => more than one grid and/or discrete and/or boundary
receptors
crti
crti  Dummy integer and real variable arrays idum and rdum apparently serve
crti    no purpose and therefore are not read.
crti
crti  The rest of the TOXXFILE structure is consistent with the binary file
crti    created by (old) SISC.
crti
                  read (iFu) dTitle
crti      read (iFu) iYr, nS, nR, nP, iTab, nX, (iDum (i), i=1, NIDUM)
C  ITAB = 1 => POLAR GRID
C         2 => CARTESIAN GRID
C         0 => DISCRETE OR MIXED GRIDS
C
                  read (iFu) iYr, nS, nR, nP, iTab, nXTOX, NYTOX
crti  nxtox = no. of x-coordinates (or distances i.e. polar rings)
crti  nytox = no. of y-coordinates (or directions i.e. polar radials)
C  NX = # RECEPTORS PER RING OR ROW (LINE 401, ABOVE)
crti      read (iFu) pCutOf, (rDum (i), i=1, NRDUM)
                  read (iFu) pCutOf
                nX = NXTOX
```

```
c
c     The following fragment should be uncommented, and the preceding
c     3 read statements commented, for formatted I/O.
c
c!         read (iFu,'(A)') dTitle
c!         read (iFu,*) iYr, nS, nR, nP, iTab, nX, (iDum(i), i=1, NIDUM)
c!         read (iFu,*) pCutOf, (rDum (i),i=1, NRDUM)
c
c         ------------------------------------------------------------
c
c         Echo this header info.
crti
crti  if(itab .eq. 1) then
crti    nx = nytox
crti    nytox = nxtox
crti  end if
crti

crti
                  write (fuPrt,'(/4X, A46, A80/ 6(4X, A46, I4/), 4X, A46,
3X,
     .        1PE10.4)')
     .        'Title of dispersion model result data       : ', dTitle,
     .        'Year modeled                                : ', iYr,
     .        '# sources                                   : ', nS,
     .        '# receptors                                 : ', nR,
     .        '# averaging periods modeled in ISCST2       : ', nP,
     .        'Exex output by recptr?  (0=no,1=pol,2=rect) : ', iTab,
     .        '# radials (polar) or rows (rect)            : ', NYTOX,
     .        'Cut-off value used in ISCST2 (g/m3)         : ', pCutOf

c
c         ------------------------------------------------------------
c
c         Error check this header info.
c
                  if (nS .lt. nSrc)
     .        call Error (fuPrt,15,iYr, nS, float(nSrc), 0.)
                  if (nR .gt. MREC)
     .        call Error (fuPrt,16,iYr, nR, float(MREC), 0.)
                  if (nR .ne. nR0) then
                      if (nR0 .eq. -1) then
                        nR0 = nR
                      else
                        call Error (fuPrt, 17, iYr, nR, float (nR0), 0.)
                      endif
                  endif
c
cSE       Setting defrt to the cutoff value used in the preprocessor
                  defrt = pCutOf
cSE       Replaced the   24/iAvPer   with   24   because iAvPer does
cSE       not have to be 1 anymore

cSE          nPer = 24/iAvPer * (365 + leap(iYr))
                  nPer = 24 * (365 + leap(iYr))
                  if (nPer .gt. MPER)
     .        call Error (fuPrt, 18, iYr, nPer, float (MPER), 0.)
c
                  if (iYr .ne. iYear(iFl))
     .        call Error (fuPrt, 20, iYr, iFl, float (iYear(iFl)), 0.)
```

```
c
c         Find the max of prescreening chi/q values used by
c         preprocessor, over all DMR datafiles.
c
                    if (pMxCut .lt. pCutOf) pMxCut = pCutOf
c
  100 continue
c
c End loop over metyears.
c
c-----------------------------------------------------------------------
c
c Assign # receptors information.
c
      nRec = nR
      nXRec = nX
c
c     If user requests receptors table and wants to use the table style
c     from the datafiles, set the table flag.
c

crti itab, from ISCST2, = 0 => discrete or mixed grids.
      if (iTab .eq. 0) lTab = 9
c
      return
      end
c
c     end subroutine ChkDat
c

c==== function leap ===================================================
c
c Determine if a year is leap or not.  Returns 1 if the year is leap,
c 0 otherwise.
c
c (Written by Paulus Irpan, 860417.)
c
c Function leap is called by subroutine ChkDat.
c
c-----------------------------------------------------------------------
c
      integer function leap (iYear)
c
c-----------------------------------------------------------------------
c
c Declarations, etc.
c
c     Description of passed parameters.
c
c     integer
c    .    iYear            ! the International/Gregorian year (yyyy)
c
      integer
     .    iYear
c
c-----------------------------------------------------------------------
c
c Test for leap.
c
c     Input parameter iYear must be in international year form, 'yyyy'.
```

```
c
      if ( (mod(iYear,4).eq.0 .and. mod(iYear,100).ne.0) .or.
     .     (mod(iYear,400).eq.0) ) then
                leap = 1
      else
                leap = 0
      endif
c
      return
      end
c
c     end function leap
c

c==== subroutine SetUp ==========================================
c
c Prepare input and other parameters for later use.  Perform extensive
c error checking of input to TOXX program.
c
c Subroutine SetUp is called by program TOXX.
c
c----------------------------------------------------------------------
c
      subroutine SetUp (fuPrt, fuTmp)
c
c----------------------------------------------------------------------
c
c Declarations, etc.
c
c     Description of passed parameters.
c
c     integer
c    .    fuPrt,            ! print fileunit
c    .    fuTmp             ! temp fileunit on disk for saved DMR data
c
      integer
     .    fuPrt, fuTmp
c
c     Insert file.
c!
      include 'toxx.ins'
c!
c     Description of local variables.
c
c     character
c    .    tFile (MSRC)*98   ! temporary internal array for sorting
c     integer
c    .    i, j, k,          ! handy integer indexes
c    .    index (MSRC),     ! index to sorted array of source group #s
c    .    iG0,              ! previous source group #
c    .    recLen            ! record length used for DMR data I/O
c     real
c    .    rateSm,           ! sum of rates for one pollutant over
c                           ! all source groups
c    .    minCut,           ! running minimum value of array oneCut
c    .    oneCut (MHET)     ! array of cutoff values for each
c                           ! individual het

      character
     .    tFile (MSRC)*98
```

```
      integer
     .    i, j, k, index (MSRC), iG0, recLen

      real
     .    rateSm, minCut, oneCut (MHET)
c
c  Functions.
c
c     real
c    .    addCut            ! function to calculate cutoff for
c                           ! additive model
c
c
c-----------------------------------------------------------------------
c
c Initial processing and sorting.
c
      print '(1X,A)', '>>>    SetUp'
c
c     Output a header to the print fileunit.
c
      write (fuPrt,'(1H1///3A)')
     .  ' ---------------------------------------  Messages from Subr',
     .  'outine Setup (may be blank)  -------------------------------',
     .  '--------'
c
c
c     Sort sources info by source group #.
c
      call SortTI (nSrc, iGrp, index)
c
c     Array index now contains a tag list of the source information
c     sorted by group #.  Write out the source information in
c     sorted order to array tFile and read it back in.
c
      do 100 i = 1, nSrc
                  j = index(i)
                    write (tFile(i),'(A20,2I2,1PE20.10E1,I4,6(1PE10.5E1))')
     .         sName(j), iSrc(j), iGrp(j), probOn(j), timeOn(j),
     .         (rate (j,k), k=1,nPol)
  100 continue
                do 110 i = 1, nSrc
                   read (tFile(i),'(A20,2I2,1PE20.10E1,I4,6(1PE10.5E1))')
     .         sName(i), iSrc(i), iGrp(i), probOn(i), timeOn(i),
     .         (rate (j,k), k=1,nPol)
  110 continue

c
c-----------------------------------------------------------------------
c
c Indexing of source information by source group.
c
c     Find and store indexes to the start of each source groups'
c     info in the sorted source arrays.  Store indexes in array
c     iSG.  The kth element of iSG contains the index, in the
c     source arrays, of the start of the data for source group
c     k.  (If some source group with source group number 1
c     does not have any source data, iSG (1) = 0.  This will only
c     happen for unused source group #s, i.e., those greater than
c     the maximum source group number used in the user input file
```

```
c     and less than or equal to MGRP.)
c
c     Check that the source group #s rise in increments of 1.
c
      do 120 i = 1, MSRC
c         All indexes are set to zero.
              iSG(i) = 0
  120 continue
c
c     Set the index of the first source group to 1.
      iSG (iGrp (1)) = 1
c     Assign iG0 as the previous source group # (which is the current
c     one at this position in the program.
      iG0 = iGrp (1)
      do 130 i = 2, nSrc
c     If the current source group id # <> to the previous source group
c     #, then determine if the current source group id# <> current
c     source group #.  If it <> then Call the Error subroutine.
                  if ( iGrp (i) .ne. iG0 ) then
                      if ( iGrp (i) .ne. (iG0+1) ) then
                        call Error (fuPrt, 23, iGrp (i), iSrc (i),
                            float (iG0), 0.)
                      else
                        iSG ( iGrp (i) ) = i
                        iG0 = iGrp (i)
                      endif
                  endif
  130 continue
c
c Find the highest source group # used and error check.
c
      nSG = iGrp (nSrc)
      if (nSG .gt. MGRP) call Error (fuPrt, 24, nSG, MGRP, 0., 0.)
c
c-----------------------------------------------------------------------
c
c Toxics release probability, duration, and emission rates.
c
c     Check turn-on probabilities, toxics release durations, and actual
c     emission rates for positive definiteness; warning if any value is
c     zero, error if any value is negative.
c
c     Also check that probabilities are less than or equal to 1.0 (error
c     if not), and that durations are integral multiples of iAvPer
c     (warning if not).
c
c     ---------------------------------------------------------------
c
c     Test of first elements of source arrays.
c
      if (probOn (1) .gt. 1.0)
     .    call Error (fuPrt, 26, iSrc (1), iGrp (1), probOn (1), 0.)
c
      if (probOn (1) .le. 0.) then
                  if (probOn (1) .lt. 0) then
                      call Error (fuPrt, 27, iSrc (1), iGrp (1), probOn (1),
0.)
                  else
                      call Warng (fuPrt, 1, iSrc (1), iGrp (1), probOn (1),
0.)
```

```
                    endif
      endif
c
      if (timeOn (1) .le. 0) then
                    if (timeOn (1) .lt. 0) then
                        call Error (fuPrt, 28, iSrc (1), iGrp (1),
                        float (timeOn (1)), 0.)
                    else
                        call Warng (fuPrt, 2, iSrc (1), iGrp (1),
     .                    float (timeOn (1)), 0.)
                    endif
      endif
c
      do 20 j = 1, nPol
                    if (rate (1,j) .le. 0.) then
                        if (rate (1,j) .lt. 0) then
                          call Error (fuPrt, 29, iSrc (1), iGrp (1), float
(j),
     .                         rate (1,j))
                        else
                          call Warng (fuPrt, 3, iSrc (1), iGrp (1), float (j),
     .                         rate (1,j))
                        endif
                    endif
   20 continue
c
c
c     ----------------------------------------------------------------
c
c     Test remaining elements of source arrays.
c
      do 150 i = 2, nSrc
                    if (iGrp (i) .ne. iGrp (i-1)) then
c
c             If current source group does not equal previous, run the
c             slew of tests used on the first elements above.
c
              if (probOn (i) .gt. 1.0)
     .            call Error (fuPrt, 26, iSrc (i), iGrp (i), probOn (i),
     .                 0.)
                        if (probOn (i) .le. 0.) then
                          if (probOn (i) .lt. 0) then
                              call Error (fuPrt, 27, iSrc (i), iGrp (i),
     .                            probOn (i), 0.)
                          else
                              call Warng (fuPrt, 1, iSrc (i), iGrp (i),
     .                            probOn(i), 0.)
                          endif
                        endif
c
                        if (timeOn (i) .le. 0) then
                          if (timeOn (i) .lt. 0) then
                              call Error (fuPrt, 28, iSrc (i), iGrp (i),
     .                            float (timeOn (i)), 0.)
                          else
                              call Warng (fuPrt, 2, iSrc (i), iGrp (i),
     .                            float (timeOn (i)), 0.)
                          endif
                        endif
c
```

```
                        do 30 j = 1, nPol
                          if (rate (i,j) .le. 0.) then
                              if (rate (i,j) .lt. 0) then
                                        call Error (fuPrt, 29, iSrc (i),
iGrp (i),
     .                                  float (j), rate(i,j))
                              else
                                        call Warng (fuPrt, 3, iSrc (i), iGrp
(i),
     .                                  float (j), rate(i,j))
                              endif
                            endif
   30         continue
c

                   else
c
c             If current group number is the same as previous, check
c             current toxics release probabilities, durations, and
c             emission rates for consistency with previous source's.
c
                        if ( probOn (i) .ne. probOn (i-1) )
     .             call Error (fuPrt, 30, iSrc (i), iGrp (i),
     .                        probOn (i), float (iSrc (i-1)) )
                        if ( timeOn (i) .ne. timeOn (i-1) )
     .             call Error (fuPrt, 31, iSrc (i), iGrp (i),
     .                        float (timeOn (i)), float (iSrc (i-1)) )
                        do 60 j = 1, nPol
                          if ( rate (i, j) .ne. rate (i-1, j) )
     .                 call Error (fuPrt, 32, iSrc (i), j,
     .                           rate (i,j), float (iSrc (i-1)) )
   60         continue
                   endif
  150   continue
c
c       ------------------------------------------------------------
c
c       End loop over sources arrays.
c
c-------------------------------------------------------------------
c
c Effective health effect thresholds (i.e., hets corrected for bkgd).
c
c     Compute effective health effect threshold concentrations and
c     convert units to g/m3 for compatibility with the chi/q (s/m3) and
c     rate (g/s) data.
c
c     If there is only one pollutant to be analyzed (in which case there
c     may be up to MHET hets but only one bkgd), allow for calculation
c     of effective thresholds using the sole background concentration.
c
      do 10, j = 1, nHET
                   if (nPol .eq. 1) then
                       k = 1
                   else
                       k = j
                   endif
                   effHET (j) = ( hET (j) - bkgd ) * 1.0E-6
                   if ( effHET (j) .le. 0 ) then
c
```

```
c              Check that each element of effHET () is positive.
c              Nonpositive values would cause ALL DMR data to pass
c              screening in Data*.
c
                        if ( effHET (j) .lt. 0 ) then
                          call Error (fuPrt, 33, j, 0, hET (j), bkgd)
                        else
                          call Warng (fuPrt, 5, j, 0, hET (j), bkgd)
                        endif
                      endif
   10 continue
c
c-----------------------------------------------------------------------
c
c Logical number of hets and TOXX screening cutoffs.
c
c
c         No additive model is used, so exceedances of each threshold
c         are individually noted.  The logical number of hets is
c         simply the total number.
c
                    logiNH = nHET
c
c         Compute variable oneCut (), for each het (in both single and
c         multipollutant, nonadditive (lAdd = 0) cases).  Find the
c         minimum of oneCut over all thresholds, cutoff.
c
                    minCut = 6.0e+23
                    do 70 j = 1, logiNH
                        if (nPol .eq. 1) then
                          k = 1
                        else
                          k = j
                        endif
                        rateSm = 0.0
                        do 80 i = 1, nSG
   80               rateSm = rateSm + rate (iSG(i), k)
                        oneCut (j) = effHET (k) / rateSm
                        if ( oneCut (j) .lt. minCut ) minCut = oneCut (j)
   70     continue
                    cutoff = minCut

c
c     Check if cutoff is lower than the largest chi/q value used in
c     prescreening the data (by the preprocessor).
c

      if (cutoff .lt. pMxCut)
     .    call Error (fuPrt, 34, 0, 0, cutoff, pMxCut)


c
c-----------------------------------------------------------------------
c
c Miscellaneous calculations and checks.
c
c     ------------------------------------------------------------------
c
c     Conversion of units of array timeOn from hours to iAvPer.
c
```

```
c     The exex calculator CaExEx works more efficiently in units of the
c     averaging period.  The conversion is performed here to get it out
c     of the way once and for all.  If a timeOn value is less than
c     one averaging period, set the corresponding tOUAvP value to unity.
c
      do 160 i = 1, nSrc
                  tOUAvP (i) = timeOn (i) / iAvPer
                  if (tOUAvP (i) .eq. 0) tOUAvP (i) = 1
  160 continue
c
c     ----------------------------------------------------------------
c
c     Actual maximum data bufferable check.
c
c     Calc acMxBf, the ceiling integral # DMR data pairs that can be
c     stored in memory in the DMR data buffer arrays iPers, iRecs, and
c     concs.  Calculate the size of the I/O record needed to write
c     acMxBf data pairs to disk at one time; store in recLen.  Prepare
c     tempfile on disk for storage of DMR data pairs in excess of acMxBf
c     in number.
c!
c!    (Note on the second line of code below: In general, recLen =
c!    (size of one element of each of iPers, iRecs, and concs, in units
c!    used to find IDMXBF) * acMxBf.
c!
c!    Here the units are bytes. Each element of iPers and iRecs is 2
c!    bytes, and each of concs is 4 bytes (set in insert file; if
c!    element sizes in this file are changed, so too must the definition
c!    of recLen
c!
      acMxBf = (IDMXBF/nSG) * nSG
      recLen = (2 + 2 + 4) * acMxBf
      open (fuTmp, status='SCRATCH', access='DIRECT', form='UNFORMATTED'
     .      , recl=recLen)
c
      return
      end
c
c     end subroutine SetUp


c==== subroutine SortTI ============================================
c
c Sort input vector array of n integer data, in ascending order, by a
c tag insertion sort.  No sorted array is actually output by subroutine
c SortTI; result is an array of subscripts of the data in the unsorted
c array, in the order in which they would appear in the sorted array.
c
c (Explicitly, this is the meaning of the output array tagLst:
c
c    INDEXES OF ARRAY IND <==> indexes of a hypothetical, sorted
c                              version of input array arr
c
c    ELEMENTS OF ARRAY IND contain the indexes of elements in the
c                          hypothetical, sorted version of input
c                          array arr IN THE ORIGINAL, UNSORTED ARRAY ARR
c
c Put one more way: tagLst is sorted tag list for original array arr.
c
c Put one last way: the value of the ith element of the hypothetical,
```

```
c                   sorted version of array arr is arr (tagLst (i))
c
c Method is best for small and almost sorted arrays.  Worst case is
c for inversely sorted arrays.
c
c Subroutine written by Paulus Irpan, SAI, 840830.)
c
c Subroutine SortTI is called by subroutine SetUp.
c
c-----------------------------------------------------------------------
c
      subroutine SortTI (n, arr, tagLst)
c
c-----------------------------------------------------------------------
c
c Declarations, etc.
c
c     Description of passed parameters.
c
c     integer
c    .    n,               ! # elements in array to be tag sorted
c    .    arr(n),          ! the array to be tag sorted
c    .    tagLst(n)        ! sorted tag list of the original array
c
      integer
     .    n, arr(n), tagLst(n)
c
c     Description of local variables.
c
c     integer
c    .    i,               ! index of current element in unsorted
c                          ! array, which is to be inserted in
c                          ! tagLst; also, a do-loop variable
c    .    j,               ! variable index used to put i in its
c                          ! proper position in tagLst
c    .    value            ! value of current element of unsorted
c                          ! array
c
      integer
     .    i, j, value
c
c-----------------------------------------------------------------------
c
c Sort by tag insertion method.
c
c     Start with 1 element presumed sorted (n_th data element).
c
      tagLst(n) = n
c
c     Then insertion sort the rest, changing the indexes only.
c
      do 100 i = n-1, 1, -1
                  if (arr(i) .le. arr(tagLst(i+1))) then
                      tagLst(i) = i
                  else
                      value = arr(i)
                      j = i + 1
  110             tagLst(j-1) = tagLst(j)
                      j = j + 1
                      if (j.le.n .and. arr(tagLst(j)).lt.value)
```

```
     .            go to 110
                       tagLst(j-1) = i
                  endif
  100 continue
c
      return
      end
c
c     end subroutine SortTI
c

c==== subroutine GetDta ==========================================
c
c Manage reading and screening of all DMR files.
c
c Subroutine GetDta is called by program TOXX.
c
c-----------------------------------------------------------------
c
      subroutine GetDta (fuDat, fuPrt, fuTmp)
c
c-----------------------------------------------------------------
c
c Declarations/insertions.
c
c     Description of passed parameters.
c
c     integer
c     .    fuDat,           ! first DMR data fileunit
c     .    fuPrt,           ! print fileunit
c     .    fuTmp            ! temp fileunit holding excess
c                           ! screened DMR data
c
      integer
     .    fuDat, fuPrt, fuTmp
c
c     Insert file.
c!
      include 'toxx.ins'
c!
c     Description of local variables.
c
c     integer
c     .    iFu,             ! DMR data fileunit currently being screened
c     .    iFl,             ! index of same
c     .    i                ! handy index
c     integer
c     .    nData            ! total # DMR data pairs available for
c                           ! all metyears combined, before screening
c                           ! in TOXX
c
      integer
     .    iFu, iFl, i
      integer
     .    nData
c
c-----------------------------------------------------------------
c
c Initialization.
c
```

```
      print '(1X,A)', '>>>>   GetDta'
c
c     Reset data paging variables.  There are so far no screened DMR
c     data pairs in memory (nBuf), no screened DMR data pairs saved
c     to disk or held in memory (nSave), and no data pages in memory
c     or saved to disk (nPage).
c
      nBuf = 0
      nSave = 0
      nPage = 0
c
c-----------------------------------------------------------------------
c
c Loop over DMR files (years of meteorological data).
c
c     Read and screen DMR datafiles.  Due to the large volume of data
c     and iterations, separate data processing for the single and
c     multisource cases.
c
      do 100 iFl = 1, nYear
                  iFu = fuDat + iFl - 1
                  print '(8X, A, I2, A)',
     .        'Message from GetDta: Now screening datafile #', iFl, '.'
c
c         Extract data from datafiles.
c
                  if (nSG .eq. 1) then
                      call Data2 (iFu, fuTmp, iFl)
                  else
                      call Data1 (iFu, fuTmp, iFl)
                  endif
c
c         Remember index of last DMR data stored for this metyear.
c
                  iSavYr(iFl) = nSave + nBuf
  100 continue
c
c     Also remember the data page currently in memory (a.k.a. the old
c     page), of which there will also be a copy on disk if it was
c     necessary to save excess data to disk.
c
      ioPage = nPage
c
c-----------------------------------------------------------------------
c
c Output of data statistics.
c
      write (fuPrt,'(1H1/ A, A, A)')
     .    '-----------------------------------------  ',
     .    'Dispersion Model Results (DMR) Read/Screened',
     .    '  ----------------------------------------'
c
c     Write # screened data by metyear, with totals.
c
      write (fuPrt,'(/4X, A10, 5X, A12, 5X, A20, 5X, A23)')
     .    'DMR File #', 'Year Modeled', 'Total # Data in File',
     .    '# Saved for Calculation'
      nData = 0
      do 200 i = 1, nYear
                  nData = nData + nDatYr(i)
```

```
                  if (i .eq. 1) then
                      write (fuPrt, '(8X, I1, 14X, I4, 15X, I8, 18X, I8)')
     .            i, iYear(i), nDatYr(i), iSavYr(i)
                  else
                      write (fuPrt, '(8X, I1, 14X, I4, 15X, I8, 18X, I8)')
     .            i, iYear(i), nDatYr(i), iSavYr(i)-iSavYr(i-1)
                  endif
  200 continue
      write (fuPrt, '(/19X, A7, 15X, I9, 18X, I8)')
     .    'Totals:' , nData, nSave
c
c     Write the chi/q cutoff value.
c
      write (fuPrt,'(/4X, A, A, 1PE12.6, A)')
     .  'Lowest value any chi/q may take and still cause exceedance',
     .  ' of a threshold is (X/Q)min = ', cutoff, ' s/m3.'
c
      return
      end
c
c     end subroutine GetDta
c

c==== subroutine Data1 ==========================================
c
c Get data for the multiple source group case.  Read and screen 1
c meteorological year of DMR data.  Save only the DMR data pairs which
c may cause exceedance of effective thresholds.
c
c Subroutine Data1 is called by subroutine GetDta.
c
c
c-----------------------------------------------------------------------
c
      subroutine Data1 (fu, fuTmp, iYr)
c
c-----------------------------------------------------------------------
c
c Declarations, etc.
c
c     Description of passed parameters.
c
c     integer
c     .    fu,         ! current DMR data fileunit
c     .    fuTmp,      ! temp fileunit for DMR data
c                      ! saved to disk
c     .    iYr         ! index of current DMR datafile
c
      integer
     .    fu, fuTmp, iYr
c
c     Insert file.
c!
      include 'toxx.ins'
c!
c     Description of local variables.
c
c     integer
c     .    NPAIR
c     parameter
```

```
c     .     (NPAIR=100)       ! # id/conc data pairs per record in
c                             ! the DMR datafiles, binary I/O
c
      integer
     .    NPAIR
      parameter
     .    (NPAIR=100)
c
c     The following fragment should be uncommented, and the preceding
c     parameter statement commented, for formatted I/O.
c
c!      parameter
c!    .      (NPAIR=10)        ! Alternative: same as above, for
c!                             ! formatted I/O
c
c     integer
c     .    idRA (NPAIR),      ! id data read-in array for one record
c     .    oneId,             ! id part of current data pair
c     .    iPair,             ! index of the current data pair
c     .    iP, iS, iR,        ! period, source, receptor # decoded from
c                             ! oneId
c     .    iP0, iR0,          ! previous period, receptor #
c     .    i, j               ! handy indexes
c     real
c     .    totCOQ (MGRP),     ! total chi over q from all sources
c                             ! (for current receptor and period,
c                             ! indexed by source group #)
c     .    cOQRA (NPAIR)      ! chi over q read-in array for one
c                             ! record (s/m3)
c
      integer
     .    idRA (NPAIR), oneId, iPair, iP, iS, iR, iP0, iR0, i, j
      real
     .    totCOQ (MGRP), cOQRA (NPAIR)
c
c-----------------------------------------------------------------------
c
c Initialization.
c
c     No data pairs are buffered yet.  Set value of previous record
c     variables to -1.
c
      do 10 i = 1, nSG
   10     totCOQ (i) = 0.
      nDatYr(iYr) = 0
      iP0 = -1
      iR0 = -1
c
c-----------------------------------------------------------------------
c
c Loop over DMR data on disk.
c
c     Read 1 record of NPAIR pairs of id and chi/q data at a time.
c     Return to this point each time screening of an entire record is
c     complete.  If FORTRAN detects an end of file, branch to label
c     199.
c
  100 continue
c
      read (fu, end=199) (idRA (i), i = 1, NPAIR)
```

```
      read (fu, end=199) (cOQRA (i), i = 1, NPAIR)
c!
c!    The following fragment should be uncommented, and the preceding
c!    2 read statements commented, for formatted I/O.
c!
c!      read (fu, *, end=199) (idRA (i), i = 1, NPAIR)
c!      read (fu, *, end=199) (cOQRA (i), i = 1, NPAIR)
c!
      nDatYr (iYr) = nDatYr (iYr) + NPAIR
c
c-----------------------------------------------------------------------
c
c Loop over elements of current record.
c
      do 110 iPair = 1, NPAIR
c
c         ------------------------------------------------------------
c
c         Initial processing for current record.
c
c         Decode the id part of data into a period, source, and
c         receptor # (ppppssrrr = pppp ss rrr).
c
                  oneId = idRA (iPair)
                  iP = oneId/100000
                  iS = oneId/1000 - (oneId/100000)*100
                  iR = oneId - (oneId/1000)*1000
c
c         ------------------------------------------------------------
c
c         Screening and saving of DMR data.
c
c         If the current period and receptor (whose data has NOT yet
c         been accumulated) is different from the previous ones, all
c         chi/q data for the PREVIOUS period and receptor has been
c         accumulated.
c
c         (The following condition can be false at most MSRC
c         consecutive times.)
c
                  if (iP.ne.iP0 .or. iR.ne.iR0) then
c
c             Check accumulated chi/q value of each source group.  If
c             none are greater than or equal to cutoff, none can cause
c             exceedance, in which case DMR data for period #iP0,
c             receptor #iR0 will not be buffered.
c
                     do 210 i = 1, nSG
                       if (totCOQ(i) .ge. cutoff) go to 220
  210             continue
                     go to 229
c
c             The accumulated chi/q for the previous period and
c             receptor is to be saved.  If number of data in buffers
c             is already within nSG of acMxBf, write current data
c             page (i.e., save buffers' contents) to disk to make
c             room.  (This is because when data is buffered, nBuf
c             is incremented by nSG).  Update indexing variables.
c
c             (Since acMxBf is in fact an integral multiple of nSG,
```

```
c             the condition for saving to disk could just as well be
c             (nBuf .eq. acMxBf).  As it stands, the condition will
c             only be true when nBuf = acMxBf.)
c
  220         continue
                        if ( (nBuf + nSG) .gt. acMxBf ) then
                          nPage = nPage + 1
                          write (fuTmp,rec=nPage) (iPers(i),i=1,acMxBf),
     .                                            (iRecs(i),i=1,acMxBf),
     .                                            (concs(i),i=1,acMxBf)
                          nSave = nSave + nBuf
                          nBuf = 0
                        endif
c
c             Continuously increment pointer variable to next available
c             element of buffers, nBuf, then put period iP0, receptor
c             iR0, and the accumulated chi/q values into buffers by
c             source group.
c
                        do 230 j = 1, nSG
                          nBuf = nBuf + 1
                          iPers(nBuf) = iP0
                          iRecs(nBuf) = iR0
                          concs(nBuf) = totCOQ(j)
  230         continue
  229         continue
c
c             Whether data has been saved or not, update previous
c             period and receptor variables to reflect new iP and
c             iR.  Also reset array containing total chi/q value,
c             totCOQ.
c
                        iP0 = iP
                        iR0 = iR
                        do 240 i = 1, nSG
                          totCOQ(i) = 0.
  240         continue
c
                    endif
c
c       ------------------------------------------------------------
c
c       Check for zero value which indicates end of data.
c
c       (The number of data is not always an integral multiple of
c       NPAIR, and the last record may be padded with 0's.  If the
c       current id datum is 0, branch to label 190.)
c
                        if (oneId .eq. 0) go to 190
c
c       ------------------------------------------------------------
c
c       Accumulate the chi/q value for period #iP0, receptor #iR0,
c       and source #iS, into array totCOQ.  Place value into the
c       element of array totCOQ () whose index corresponds to
c       that of source #iS's group number in the source info arrays.
c
c       (It is possible for there to be more sources in the DMR data
c       than are desired in the TOXX run.  If source #iS is not found
c       in the iSrc array, do not accumulate the current chi/q value.)
```

```
c
                  do 310 i = 1, nSrc
                     if (iS .eq. iSrc(i)) then
                       totCOQ(iGrp(i)) = totCOQ(iGrp(i)) + cOQRA(iPair)
                       go to 319
                     endif
  310     continue
  319     continue
c
c         ------------------------------------------------------------
c
c         End loop on elements of current record.
c
  110 continue
c
c-----------------------------------------------------------------------
c
c End of loop processing.
c
c     Read next record of data.
c
      go to 100
c
c     Label 190 is branched to if end of data was reached during the
c     loop over elements of the current record (a 0 was found in oneId).
c     Correct the variable containing the number of data in the current
c     DMR file to reflect this fact.
c
  190 nDatYr(iYr) = nDatYr(iYr) - NPAIR + iPair-1
c
c     Label 199 is branched to if FORTRAN detects an end of file while
c     reading from the DMR file on disk (i.e., the DMR file was an
c     integral multiple of 100 data long, and was therefore unpadded.)
c
  199 continue
c
c-----------------------------------------------------------------------
c
c Processing for last metyear.
c
c     If the last meteorological year of DMR data has just been
c     screened, the following special processing is required.
c
c     If more than acMxBf data have passed screening (nSave > acMxBf),
c     save the last page of data, which is the one currently in
c     memory, to disk.  (This page will contain less than acMxBf data,
c     unless the last storage of data to the buffers filled them
c     exactly.)
c
c     Note that if nSave is less than or equal to acMxBf, no saves to
c     disk have been made, and none are made now.
c
c     nSave is updated whether or not a save to disk is performed
c     here, and nBuf is set to 0, because of the way iSavYr (iYr) is
c     calculated upon returning to the GetDta subroutine.  The
c     calculation in GetDta works fine for every metyear except the
c     last, necessitating this correction.
c
c     nPage is also incremented whether or not a save to disk is done
c     here.  If no save is performed, it means that the screened data
```

```
c     do  not fill the buffers, in which case nPage = 0 until now  and
c     must  be  updated to reflect the single page of  screened  data,
c     residing entirely in memory.  If a save is performed, of  course
c     nPage must be incremented to reflect the last page of data saved
c     to disk.
c
      if (iYr .eq. nYear) then
                  nSave = nSave + nBuf
                  nBuf = 0
                  nPage = nPage + 1
                  if (nSave .gt. acMxBf) then
                      write (fuTmp,rec=nPage) (iPers(i),i=1,acMxBf),
     .                                        (iRecs(i),i=1,acMxBf),
     .                                        (concs(i),i=1,acMxBf)
                  endif
      endif
c
      return
      end
c
c     end subroutine Data1
c

c==== subroutine Data2 ==========================================================
c
c Get data for the single source group case.  Read and screen 1
c meteorological year of DMR data.  Save only the DMR data pairs which
c may cause exceedance of effective thresholds.
c
c Subroutine Data2 is called by subroutine GetDta.
c
c----------------------------------------------------------------------------
c
      subroutine Data2 (fu, fuTmp, iYr)
c
c----------------------------------------------------------------------------
c
c Declarations, etc.
c
c     Description of passed parameters.
c
c     integer
c     .    fu,              ! current DMR data fileunit
c     .    fuTmp,           ! temp fileunit for DMR data
c                           ! saved to disk
c     .    iYr              ! index of current DMR datafile
c
      integer
     .    fu, fuTmp, iYr
c
c     Insert file.
c!
      include 'toxx.ins'
c!
c     Description of local variables.
c
c     integer
c     .    NPAIR
c     parameter
c     .    (NPAIR=100)      ! # id/conc data pairs per record in
```

```
c                           ! the DMR datafiles, binary I/O
c
      integer
     .    NPAIR
      parameter
     .    (NPAIR=100)
c
c     The following fragment should be uncommented, and the preceding
c     parameter statement commented, for formatted I/O.
c
c!      parameter
c!    .      (NPAIR=10)        ! Alternative: same as above, for
c!                             ! formatted I/O
c
c     integer
c    .    idRA (NPAIR),      ! id data read-in array for one record
c    .    oneId,             ! id part of current data pair
c    .    iPair,             ! index of the current data pair
c    .    iP, iS, iR,        ! period, source, receptor # decoded from
c                            ! oneId
c    .    iP0, iR0,          ! previous period, receptor #
c    .    i                  ! handy index
c     real
c    .    totCOQ,            ! total chi over q from all sources
c                            ! (for current receptor and period,
c                            ! indexed by source group #)
c    .    cOQRA (NPAIR)      ! chi over q read-in array for one
c                            ! record (s/m3)
c
      integer
     .    idRA (NPAIR), oneId, iPair, iP, iS, iR, iP0, iR0, i
      real
     .    totCOQ, cOQRA (NPAIR)
c
c-----------------------------------------------------------------------
c
c Initialization.
c
c     No data pairs are buffered yet.  Set value of previous record
c     variables to -1.
c
      totCOQ = 0.
      nDatYr(iYr) = 0
      iP0 = -1
      iR0 = -1
c
c-----------------------------------------------------------------------
c
c Loop over DMR data on disk.
c
c     Read 1 record of NPAIR pairs of id and chi/q data at a time.
c     Return to this point each time screening of an entire record is
c     complete.  If FORTRAN detects an end of file, branch to label
c     199.
c
  100 continue
c
      read (fu, end=199) (idRA (i), i = 1, NPAIR)
      read (fu, end=199) (cOQRA (i), i = 1, NPAIR)
```

```
c!
c!    The following fragment should be uncommented, and the preceding
c!    2 read statements commented, for formatted I/O.
c!
c!      read (fu, *, end=199) (idRA (i), i = 1, NPAIR)
c!      read (fu, *, end=199) (cOQRA (i), i = 1, NPAIR)
c!
      nDatYr (iYr) = nDatYr (iYr) + NPAIR
c
c-----------------------------------------------------------------------
c
c Loop over elements of current record.
c
      do 110 iPair = 1, NPAIR
c
c         ------------------------------------------------------------
c
c         Initial processing for current record.
c
c         Decode the id part of data into a period, source, and
c         receptor # (ppppssrrr = pppp ss rrr).
c
                  oneId = idRA (iPair)
                  iP = oneId/100000
                  iS = oneId/1000 - (oneId/100000)*100
                  iR = oneId - (oneId/1000)*1000
c
c         ------------------------------------------------------------
c
c         Screening and saving of DMR data.
c
c         If the current period and receptor (whose data has NOT yet
c         been accumulated) is different from the previous ones, all
c         chi/q data for the PREVIOUS period and receptor has been
c         accumulated.
c
c         (The following condition can be false at most MSRC
c         consecutive times.)
c
                  if ( (iP .ne. iP0) .or. (iR .ne. iR0) ) then
c
c             Check accumulated chi/q. If it is not greater than or
c             equal to cutoff, it cannot cause exceedance, in which case
c             DMR data for period #iP0, receptor #iR0 will not be
c             buffered.
c
                     if ( totCOQ .ge. cutoff .and. iP0 .ne. -1 .and. iR0
.ne.
     .         -1) then
c
c                 The accumulated chi/q for the previous period and
c                 receptor is to be saved.  If number of data in buffers
c                 is already greater than or equal to acMxBf, write
c                 current data page (i.e., save buffers' contents) to
c                 disk to make room.  Update indexing variables.
c
                        if ( nBuf .ge. acMxBf ) then
                           nPage = nPage + 1
                           write (fuTmp,rec=nPage) (iPers(i),i=1,acMxBf),
     .                                             (iRecs(i),i=1,acMxBf),
```

```
                                                (concs(i),i=1,acMxBf)
                              nSave = nSave + nBuf
                              nBuf = 0
                            endif
c
c                   Increment pointer variable to next available element
c                   of buffers, nBuf, then put period iP0, receptor iR0,
c                   and the accumulated chi/q value into buffers.
c
                              nBuf = nBuf + 1
                              iPers (nBuf) = iP0
                              iRecs (nBuf) = iR0
                              concs (nBuf) = totCOQ
                            endif
c
c                 Whether data has been saved or not, update previous
c                 period and receptor variables to reflect new iP and
c                 iR.  Also reset variable containing total chi/q value,
c                 totCOQ.
c
                              iP0 = iP
                              iR0 = iR
                              totCOQ = 0.
c
                        endif
c
c           ------------------------------------------------------------
c
c           Check for zero value which indicates end of data.
c
c           (The number of data is not always an integral multiple of
c           NPAIR, and the last record may be padded with 0's.  If the
c           current id datum is 0, branch to label 190.)
c
                        if (oneId .eq. 0) go to 190
c
c           ------------------------------------------------------------
c
c           Accumulate the chi/q value for period #iP0, receptor #iR0,
c           and source #iS, into variable totCOQ.
c
c           (It is possible for there to be more sources in the DMR data
c           than are desired in the TOXX run.  If source #iS is not found
c           in the iSrc array, do not accumulate the current chi/q value.)
c
                        do 210 i = 1, nSrc
                            if (iS .eq. iSrc(i)) then
                              totCOQ = totCOQ + cOQRA(iPair)
                              go to 219
                            endif
  210                   continue
  219                   continue
c
c           ------------------------------------------------------------
c
c           End loop on elements of current record.
c
  110 continue
c
c-------------------------------------------------------------------
```

```
c
c End of loop processing.
c
c     Read next record of data.
c
      go to 100
c
c     Label 190 is branched to if end of data was reached during the
c     loop over elements of the current record (a 0 was found in oneId).
c     Correct the variable containing the number of data in the current
c     DMR file to reflect this fact.
c
  190 nDatYr(iYr) = nDatYr(iYr) - NPAIR + iPair-1
c
c     Label 199 is branched to if FORTRAN detects an end of file while
c     reading from the DMR file on disk (i.e., the DMR file was an
c     integral multiple of 100 data long, and was therefore unpadded.)
c
  199 continue
c
c-----------------------------------------------------------------------
c
c Processing for last metyear.
c
c     (See the end of subroutine Data1 for documentation of the
c     following fragment.)
c
      if (iYr .eq. nYear) then
                  nSave = nSave + nBuf
                  nBuf = 0
                  nPage = nPage + 1
                  if (nSave .gt. acMxBf) then
                      write (fuTmp,rec=nPage) (iPers(i),i=1,acMxBf),
                                              (iRecs(i),i=1,acMxBf),
                                              (concs(i),i=1,acMxBf)
                  endif
      endif
c
      return
      end
c
c     end subroutine Data2
c

c==== subroutine DoExEx ==============================================
c
c Manage the calculation of the expected exceedances of effective health
c effect thresholds, and the resultant output of this calculation, for
c any number of source groups.
c
c Subroutine DoExEx is called by program TOXX.
c
c-----------------------------------------------------------------------
c
      subroutine DoExEx (fuPrt, fuTmp, fuHEM, nP)
c
c-----------------------------------------------------------------------
c
c Declarations, etc.
c
```

```
c     Description of passed parameters.
c
c     integer
c    .    fuPrt,                    ! print fileunit
c    .    fuTmp,                    ! temp fileunit for DMR data
c                                   ! saved to disk
c    .    fuHEM                     ! HEM output fileunit
c
      integer
     .    fuPrt, fuTmp, fuHEM
c
c     Insert file.
c!
      include 'toxx.ins'
c!
c     Description of local variables.
c
c     integer
c    .    iG,                       ! index of current source group
c    .    iH,                       ! index of current het
c    .    iR,                       ! index of current receptor
c    .    iYr                       ! index of current metyear
c     real
c    .    xXRHGY (MREC,MHET,MGRP),  ! exex by receptor, by het, by
c                                   ! source group for one metyear
c    .    xXRHG (MREC,MHET,MGRP),   ! exex by receptor, by het, by
c                                   ! group, summed over all metyears
c    .    xXRHY (MREC,MHET),        ! exex by receptor, by het, summed
c                                   ! over all groups, for one metyear
c    .    xXRH (MREC,MHET),         ! exex by receptor, by het, summed
c                                   ! over all groups and metyears
c    .    xMaxHG (MHET,MGRP),       ! exex max by het, by group, inde-
c                                   ! pendent of receptor
c    .    xMaxH (MHET)              ! overall exex max, by het, inde-
c                                   ! pendent of group and receptor
c
      integer
     .    iG, iYr

      real
     .    ICODE1(MGRP,MHET,MREC),rHMC(MREC,MHET)
c
c-----------------------------------------------------------------------
c
c Initialization.
c
      print '(1X,A)', '>>>>>  DoExEx'
c
c     Check whether any data passed screening in the Data* subroutines.
c
      if (nSave .eq. 0) then
                  write (fuPrt,'(//A)')
     .         ' No data passed screening, thus no expected exceedances.'
                  return
      endif
c
c-----------------------------------------------------------------------
c
c Loop over years of meteorological data.
c
```

```
c     Calculate exex by metyear, and output by metyear if desired.
c     Maintain running sum of exex for all metyears handled so far.
c
      do 200 iYr = 1, nYear
c
c         ------------------------------------------------------------
c
c         Calculate exex and the maxima for current metyear.  CaExEx
c         must be passed the actual, rather than logical, # of hets.
c
                    call CaExEx (fuTmp, iYr, MREC, MHET, nRec, nHET, nSG,
     .         ICODE1, rHMC)
c
c
c         ------------------------------------------------------------
c
c         Metyearly output.
c
c         Output results by metyear if desired, but not when there is
c         only one metyear.
c
c
c             Output results by individual source group, unless there
c             is only one source group, in which case output for the
c             single source group will appear as the output for all
c             source groups combined.
c
                         do 400 iG = 1, nSG
                    call WrExEx (fuPrt, fuHEM, iYear(iYr),iYear
     .                         (nYear),
     .                         iGrp(iSG(iG)), lAdd, MGRP, MREC,
     .                         nXRec, MHET, nHET, logiNH,iYr,nP,
     .                         ICODE1,rHMC)
  400             continue
c
c       ------------------------------------------------------------
c
c       End loop over years of meteorological data.
c
  200 continue
c
c-----------------------------------------------------------------------
c
c Overall exex, all metyears combined.
c
c     Calc exex by receptor, by group, averaged over all metyears. Store
c     in array xXRHG.  Find the maximum average exex for each het, each
c     source group, independent of the receptor.
c

      return
      end
c
c     end subroutine DoExEx
c

c==== subroutine CaExEx ==============================================
c
c Calculate exex using Monte Carlo simulation, for one metyear.
c
```

```
c In order to maintain the convention of placing the insert file AFTER
c the declaration of passed parameters, mR, mH, nR, nH, nG are passed
c for array dimensioning purposes, although they in fact equal insert
c file variables MREC, MHET, nRec, nHET, and nSG, and are held in com-
c mon.  The names of the insert file variables are not used to avoid
c having local dummy parameters masquerading as insert file variables;
c this would be deceptive and confusing.  The names mR, mH, nR, nH, and
c nG are used throughout CaExEx for consistency.
c
c (CaExEx arrays have been specially named with suffix "MC" (Monte
c Carlo) to distinguish them clearly from DoExEx arrays with similar
c but different contents.)
c
cSE This version of subroutine CaExEx was written for Sept. 30 1993 version
cSE of TOXX.  The calls to subroutine RandNo have been incorporated into a
cSE new subroutine, BatPr.  The subroutine OnOff from the original version
cSE of INTOXX has been incorporated into this new subroutine.
c
c Subroutine CaExEx is called by subroutine DoExEx.
c
c-----------------------------------------------------------------------
c
      subroutine CaExEx (fuTemp, yr, mR, mH, nR, nH, nG, ICODE1, rHMC)
c
c-----------------------------------------------------------------------
c
c Declarations, etc.
c
c     Description of passed parameters.
c
c     integer
c    .    fuTemp,            ! temp fileunit for DMR data
c                            ! saved to disk
c    .    yr,                ! index of current metyear
c    .    mR, mH,            ! max # recs, max # hets
c    .    nR, nH, nG         ! # receptors, # hets, and # groups
c
c     (In the following four array names, the 'MC' identifies each
c     variable as belonging to CaExEx.)
c
c     real
c    .rHGMC (mR,mH,nG),      ! array of exex by receptor, by het,
c                            ! by source group, summed over simyears;
c                            ! avgd. over simyears at end
c    .rHMC (mR,nH),          ! array of exex by receptor, by het,
c                            ! summed over source groups and
c                            ! simyears; avgd. over simyears at end
c    .mxHGMC (mH,nG),        ! simyear-averaged max exex by het, by
c                            ! source group, any receptor
c    .mxHMC (nH)             ! simyear averaged max exex by het, any
c                            ! source group or receptor
c
      integer
     .    fuTemp, yr, mR, nR, nH, nG


cSE
      parameter (NPAIR = 100)

cSE
```

```
c
c     (In the following four array names, the 'MC' identifies each
c     variable as belonging to CaExEx.)
c
                    real
     .         rHMC (mR,nH)
c
c     Insert file.
c!
      include 'toxx.ins'
c!
c     Description of local variables.
c
c     integer
cSE   .     compl,              ! indicates that an averaging period has
cSE                             ! been completed when it equals YES.  Will
cSE                             ! only equal YES when the last receptor for
cSE                             ! for the final hour of the averaging period
cSE                             ! has been processed
c     .     iR,                 ! current receptor
c     .     iH,                 ! index of current het
c     .     iG,                 ! current source group
c     .     iP,                 ! current period
c     .     iPol,               ! current pollutant
c     .     iSim,               ! index of current simulation
c     .     i,                  ! handy index
c     .     iBSav,              ! location of first datum for current
c                               ! metyear in screened DMR data
c     .     iESav,              ! location of last  datum for current
c                               ! metyear in screened DMR data
c     .     iBPage,             ! page containing first datum
c     .     iEPage,             ! page containing last datum
c     .     iPage,              ! index of current page
c     .     iBDat,              ! location of first datum for current
c                               ! metyear on current page
c     .     iEDat,              ! location of last datum for current
c                               ! metyear on current page
c     .     iDat,               ! index of current datum on current page
cSE   .     Recp,               ! identical to iR, but is needed when multiple
cSE                             ! hour averaging is desired.  If an averaging
cSE                             ! period is completed when before all skipped
cSE                             ! periods have been processed for a receptor,
cSE                             ! iR will become the last receptors when
cSE                             ! calculating the exceedances.  Therefore it is
cSE                             ! necessary to have a variable that represents
cSE                             ! the current receptor that will be passed to
cSE                             ! subroutine BatPr
cSE   .     scndskp             ! similar to iSkip, except this is assign
cSE   .                         ! the value of iSkip - iPriod for use when
cSE                             ! an averaging period has been completed prior
cSE                             ! using up all the hours in iSkip
cSE   .     sendit,             ! indicates that an averaging period has been
cSE                             ! completed prior to processing all skipped
cSE                             ! periods.  When this equals YES, the program
cSE                             ! skips over the process of getting the data
cSE                             ! next in line for iP, iR, and concs()
cSE   .     SK,                 ! sent into CaExEx as YES from subroutine
cSE                             ! BatPr indicating that an averaging
cSE                             ! period been completed prior to processing all
cSE                             ! skipped periods.  Becomes NO when the last
```

```
cSE                             ! receptor has been processed through CaExEx
cSE   .    TK                   ! when SK changes from YES to NO, TK becomes YES
cSE                             ! to indicate that the last receptor still needs
cSE                             ! to be processed.  When this becomes YES sendit
cSE                             ! also becomes NO.
c
c     real
c    . impSSG (MHET,MGRP),      ! concentration impacts of single source
c                               ! groups, by het, by group
c    . impASG (MHET),           ! concentration impacts of all source
c                               ! source groups, by het
cSE  . ON, YES,                 ! when a variable equals 1.0, it is either
cSE                             ! ON or YES, respectively
cSE  . OFF, NO                  ! when a variable equals 0.0, it is either
cSE                             ! OFF or NO, respectively.

      real
     .    ICODE1(MGRP,MHET,MREC)

      integer
     .    iR, iH, iG, iP, iPol, iSim, i, iBSav, iESav, iBPage, iEPage,
     .    iPage,iBDat,iEDat,iDat, compl, Recp, SK, TK, sendit, numGps,
     .    scndskp,BGNiR,LASTiR

      real
     .    impSSG (MHET,MGRP), impASG (MHET), ON, OFF, YES, NO

      logical lastR

      parameter
     .    (ON = 1.0, OFF = 0.0, YES = 1.0, NO = 0.0)
c
c-----------------------------------------------------------------------
c
c Initialization.
c
      print '(1X,A)', '>>>>>> CaExEx'
c

cSE   Zero out array rHMC for cumulative years.

      do 119 iH = 1, nH
                  do 120 iR = 1, nR
                      rHMC (iR,iH) = 0.
  120     continue
  119 continue

cSE   Initialize scndskp to the number of hours in an averaging period

      scndskp = iAvPer

c
c-----------------------------------------------------------------------
c
c Prepare indexes to needed DMR data.
c
c     Calculate the locations of the first and last data for this
c     metyear within the screened DMR data for all metyrs.
c
      if (yr .eq. 1) then
```

```
                    iBSav = 1
      else
                    iBSav = iSavYr (yr-1) + 1
      endif
      iESav = iSavYr(yr)
c
c     Determine which data pages contain datum #s iBSav and iESav
c
      iBPage = (iBSav + acMxBf-1) / acMxBf
      iEPage = (iESav + acMxBf-1) / acMxBf

c
c-----------------------------------------------------------------------
c
c Monte Carlo simulation loop.
c
c     ------------------------------------------------------------------
c
c     Loop over simyears.
c
      do 200 iSim = 1, nSim

      write (*,201) iSim
  201 format ('+','***** Simulation: ',i5.5)

cSE   Set iP to 0.
      iP = 0

c
c         Start-of-simyear initialization.  No previous period, no time
c         elapsed so far for any source group, all source groups off.
c
c         ------------------------------------------------------------
c
c         Loop over pages of data for current metyear (#yr).
c


                    do 400 iPage = iBPage, iEPage
c
c             Page in data from tempfile if needed.
c
                      if (iPage .ne. ioPage) then
                        read (fuTemp,rec=iPage) (iPers(i),i=1,acMxBf),
                                                (iRecs(i),i=1,acMxBf),
                                                (concs(i),i=1,acMxBf)
                        ioPage = iPage
                      endif
c
c             If the page # of the data currently in memory is between
c             iBPage and iEPage, uninclusive, then the start and end of
c             the data on the current page for metyear #yr are at the
c             beginning and end, respectively, of the DMR data buffers.
c
                      iBDat = 1
                      iEDat = acMxBf
c
c             If, on the other hand, the current data page # = iBPage or
c             iEPage, then there is a possibility that some of the data
c             on the current page does not belong to metyear #yr.  In
```

```
c             this case a special calculation is necessary to determine
c             where on the current page to start or stop reading
c             screened DMR data for metyear #yr.
c
                      if (iPage.eq.iBPage) iBDat = iBSav - (iPage-1)*acMxBf
                      if (iPage.eq.iEPage) iEDat = iESav - (iPage-1)*acMxBf
c
c             ------------------------------------------------------------
c
c             Loop over data on current page (#iPage).
c
                      do 410 iDat = iBDat, iEDat, nG

cSE
cSE           If the averaging period for TOXST is > 1, we have to
cSE           loop through all receptors even if there are receptors
cSE           that did not have an exceedance for this hour.

  413             if (iAvPer.gt.1) then
                          if (iP.eq.0) then
                                  lastR = .FALSE.
                                  iP = iPers(iDat)
                                  iR = iRecs(iDat)
                                  LASTiR = iR
                                  BGNiR = 1
                          else if (iP.eq.iPers(iDat)) then
                                  iR = iRecs(iDat)
                                  BGNiR = LASTiR + 1
                                  LASTiR = iR
                          else if (iP.lt.iPers(iDat)) then
                                  BGNiR = LASTiR+1
                                  LASTiR = nR
                                  lastR = .TRUE.
                          end if
                      else
                          iP = iPers(iDat)
                          iR = iRecs(iDat)
                          BGNiR = iR
                          LASTiR = iR
                      end if
cSE

cSE   Assign concen(,) to the concentrations in the temp. file.  These
cSE   are passed to BatPr through TOXX.INS.

                      do 414 I = BGNiR, LASTiR

                          do 499 iG = 1, nSG
                                  if (I .eq. LASTiR .and. .not. lastR)
then
                                      concen(iG,I) = concs(iDat + iG -
1)
                                  else
                                      concen(iG,I) = defrt
                                  end if
  499                 continue

cSE   Assign variable Recp to iR.  Later on in this subroutine, after
cSE   the call to BatPr, iR is used in other loops
cSE   which when finished result in iR assigned to the last receptor.
```

```
cSE   This is needed when the iAvPer > 1.  If an averaging period does
cSE   not have enough hours to form a complete averaging period, the
cSE   program goes to statement 412 to finish up the remaining receptors
cSE   for the current hour.

                        Recp = I

cSE   If mutually exclusive emissions are desired, set the number of
cSE   groups to 1.  Processing of all source groups will take place
cSE   in subroutine BatPr.

  412             if (MEEm .eq. YES) then
                          numGps = 1
                      else
                          numGps = nSG
                      end if

                      do 411 iG = 1, numGps
                          call BatPr (iG, iP, Recp, impSSG, compl, SK,
     .                        scndskp)

cSE   iR must be assigned to the original iR which was assigned to Recp
cSE   prior to statement 412.

                          iR = Recp

  411             continue

cSE   If, as determined from BatPr that an averaging period has not
cSE   been completed (compl = NO), the program goes to statement
cSE   410 if the iAvPer > 1.

                      if (compl .ne. YES .and. iAvPer .gt. 1) go to 414

cSE   This next "go to" takes care of the last receptor.

                      if (iR .eq. nR .and. SK .eq. NO) then
                          compl = NO
                          if (TK .eq. YES) then
                                      TK = NO
                                      go to 451
                          end if
                      end if
                          if (sendit .eq. YES .and. (SK .eq. YES .or. TK .eq.
     .                YES) .and. scndskp .lt. iAvPer) go to 451

c
c                 ------------------------------------------------
c
c                 Calculation of exex.
c
c
c                     Zero out the array impASG, which accumulates
c                     impacts for all source groups.  It must be reset
c                     for each receptor-period combination.
c
                          do 424 iH = 1, nH
  424                   impASG (iH) = 0.
c
                          do 425 iG = 1, nG
```

```
                                        do 425 iH = 1, nH
                                            impASG (iH) = impASG (iH)
     .                                    + impSSG (iH,iG)
  425                continue

cSE   iC:     Assigned to the current threshold.

                              iInc=2

                                        iPol = 1
                                        do 440 iH = 1, nH
                                            do 441 iG = 1, nG
                                                                  if
(impSSG(iH,iG) .ge. effHET(iH))
     .                           then

  445                               if ((impSSG(iH,iG).lt.effHET(iInc)
     .                              .and. iH.ne.nH) .or. iInc-1.eq.nH)
     .                              then
                                                                  iC =
iInc-1
                                                                  iInc=2
                                                                      else

iInc=iInc+1
                                                                  goto 445
                                                                      end
if

ICODE1(iG,iC,iR)=ICODE1(iG,iC,iR)
     .                                  + 1
                                                                   endif
  441                           continue
                                        iInc=2
                                        if (impASG(iH).ge.effHET(iH))
then
  436                               if (impASG(iH).lt.effHET(iInc)
     .                                 .and. iH.ne.nH .or. iInc-1.eq.nH)
     .                              then
                                                                       iC =
iInc-1

iInc=2
                                                                   else

iInc=iInc+1
                                                                       go
to 436
                                                                  end if

rHMC(iR,iC)=rHMC(iR,iC)+1
                                        end if
  440                       continue

c                 End calculation of exex.
c
c                 ------------------------------------------------
c

cSE   Once there has been a completed averaging period, set impSSG(,)
```

```
cSE   to 0.

                                do 450 iG = 1, nG
                                       do 450 iPol = 1, nPol
                                           impSSG(iPol,iG) = 0.
  450                  continue

cSE   If SK = YES, an averaging period has been completed prior to looping
cSE   through all the skipped periods.  The last skipped period is the
cSE   period with the hit (concentration over the cutoff value) so we
cSE   must use that period.  SK will only be set to YES for a period
cSE   when the loop in BatPr is on the last receptor.
cSE   When this occurs, iR and Recp are set to the first receptor,
cSE   sendit is set to YES and the program is diverted back to statement
cSE   412.  This by-passes the process of getting a new iP and iR because
cSE   we still need to finish up all of the skipped periods.

  451             if (SK .eq. YES) then
                                if (Recp .eq. nR) then
                                       iR = 1
                                       Recp = iR
                                       sendit = YES
                                else

cSE   When we are in this part and iR is set to the last receptor,
cSE   we have to set SK and sendit back to NO.  TK is set to YES
cSE   because we still have this last receptor to go through and
cSE   something needs to indicate this.

                                       iR = iR + 1
                                       Recp = iR
                                       if (iR .eq. nR) then
                                           SK = NO
                                           TK = YES
                                           sendit = NO
                                       end if
                                   end if
                                   go to 412
                               end if
  414              continue
                           if ((I-1).eq.nR.and.iAvPer.gt.1) then
                               iP = 0
                               if (lastR) go to 413
                           end if
  410          continue
c
c              End loop over data on current page (#iPage).
c
c              ------------------------------------------------------
c
  400      continue
c
c          End loop over data for current metyear (#yr).
c
c          ------------------------------------------------------
c
  200 continue
c
c      End Monte-Carlo simulations loop.
c
```

```
c-----------------------------------------------------------------------
c
      return
      end
c
c     end subroutine CaExEx
c-----

cSE== subroutine BatPr ==========================================
cSE
cSE Determine emission rates and ON/OFF switches for the passed source
cSE group and receptor.
cSE
cSE Subroutine BatPr was written for Sept. 30, 1993 version of TOXX.
cSE Subroutine OnOff, originally called from subroutine CaExEx,
cSE has been incorporated into this subroutine.
cSE
cSE Subroutine BatPr is called by subroutine CaExEx.
cSe
cSE-----------------------------------------------------------------------
cSE

      subroutine BatPr (iG, iP, iR, impSSG, compl, SK, scndskp)
cSE
cSE-----------------------------------------------------------------------
cSE Insert file
cSE!
      include 'toxx.ins'
cSE!
cSE
cSE Declarations, etc.
cSE
cSE   Description of passed parameters.
cSE
cSE   integer
cSE   .    iG,                ! current source group
cSE   .    iP,                ! current period (in hours)
cSE   .    iR,                ! current receptor
cSE   .    compl,             ! indicates that an averaging period has
cSE                           ! been completed when it equals YES.  Will
cSE                           ! only equal YES when the last receptor for
cSE                           ! for the final hour of the averaging period
cSE                           ! has been processed
cSE   .    scndskp            ! similar to iSkip, except this is assign
cSE   .                       ! the value of iSkip - iPriod for use when
cSE                           ! an averaging period has been completed prior
cSE                           ! using up all the hours in iSkip
cSE   .    SK,                ! when sent into CaExEx as YES from subroutine
cSE                           ! BatPr indicates that an averaging
cSE                           ! period been completed prior to processing all
cSE                           ! skipped periods.  Becomes NO when the last
cSE                           ! receptor has been processed through CaExEx
cSE   . leftovr (MGRP,MREC)   ! When multiple averaging periods are desired,
cSE                           ! this indicates the number of periods that are
cSE                           ! left over from the skipped periods. (hours
cSE                           ! that did not fit into an averaging period).
cSE                           ! For example, say the averaging period is 3
cSE                           ! hours, and that the number of skipped periods
cSE                           ! is 11 hours, 11 / 3 is 3.6667.  3 x 3 = 9,
cSE                           ! therefore 9 single hours fit evenly into three
```

```
cSE                          ! 3 hour averaging periods.  .6667 x 3 = 2 which
cSE                          ! is what leftovr(,) is equal to.  Because all
cSE                          ! source groups will have the same amount of
cSE                          ! periods stored in the temporary files when
cSE                          ! multiple hour averaging is desired, we need
cSE                          ! only identify the value of leftovr(,) for the
cSE                          ! first source group and first receptor

cSE   real
cSE  . impSSG (MHET,MGRP),   ! concentration impacts of single source
cSE                          ! groups, by het (poll), by group
cSE  . tot (MGRP,MHET,MREC)  ! used when averaging period is greater than
cSE                          ! 1 to hold accumulated concentrations for
cSE                          ! a partial averaging period (i.e. 2 hours of
cSE                          ! a three hour averaging period) for each
cSE                          ! source group, pollutant, and receptor.
cSE  . usert                 ! the value of the concentration to be used
cSE                          ! for the current period in this subroutine.
cSE                          ! If the current period is the period that has
cSE                          ! the hit (i.e., the last skipped period),
cSE                          ! usert will equal conc, otherwise it will
cSE                          ! equal the default (i.e., the cutoff value
cSE                          ! used in ISCST2.)

cSE   passed parameters
      integer
     .    compl, iG, iP, iR, iSrG, scndskp, SK, leftovr(MGRP,MREC)
      real
     .    tot(MGRP,1,MREC), impSSG(MHET,MGRP), rand(2), usert

cSE Description of local parameters.
cSE
cSE   integer
cSE  .    count,         ! variable to indicate the # of source groups
cSE                      ! that have passed through this routine for the
cSE                      ! current period
cSE  .    firstR,        ! when 1 hour averaging periods are desired,
cSE                      ! all receptors are not put into the temporary
cSE                      ! file, therefore it is necessary to have some-
cSE                      ! thing indicate which receptor was the first
cSE                      ! to enter this routine for a particular period
cSE                      ! so that a new ON/OFF switch will not be
cSE                      ! determined for subsequent receptors of the
cSE                      ! same period
cSE  .    hr,            ! represents what hour the current period
cSE                      ! is on a 24 hour clock.  Used in
cSE                      ! determining if emissions should be AM or PM
cSE                      ! rates
cSE  .    imark,         ! used only when mutually exclusive emissions
cSE  .                   ! are desired.  It is set to YES when an averaging
cSE  .                   ! period is completed and the source group that is
cSE  .                   ! on is not equal to the last source group,
cSE  .                   ! i.e., nSG
cSE  .    iPrvPr,        ! indicates what the previous period was for the
cSE                      ! first time that a new period enters this routine
cSE  .    iPol,          ! indicates the current pollutant
cSE  .    iPriod,        ! indicates the current skipped period
cSE  .    iSkip,         ! calculated number of skipped periods
cSE  .    iStart,        ! indicator of where iPriod should start.
cSE                      ! Normally, this will be 1, however when an
```

```
cSE                          ! averaging period has been completed prior
cSE                          ! processing all skipped periods, the period
cSE                          ! that contains the hit (when iPriod = iSkip)
cSE                          ! has not been processed yet.  The next time
cSE                          ! we go through this loop again for the same
cSE                          ! receptor, iStart must be equal to the next
cSE                          ! period.  See also JK
cSE  .    j,                 ! variable assigned to Edist(jg) if source
cSE  .                       ! group is not batch
cSE  .    jg,                ! identical to iG when mutually exclusive
cSE                          ! emissions are not desired.  When mutually
cSE                          ! exclusive emissions are desired jg equals one
cSE                          ! when it is passed to this subroutine but is
cSE                          ! assigned to a randomly chosen source group
cSE  .    JK,                ! set to whatever iPriod is equal to when an
cSE                          ! averaging period is completed but all skipped
cSE                          ! periods have not been processed.  When this
cSE                          ! subroutine is entered again to complete the
cSE                          ! processing of the remaining skipped periods,
cSE                          ! iStart will be set to JK + 1 so that the
cSE                          ! processing can continue right after where it
cSE                          ! left off
cSE  .    loop1,             ! indicator that this subroutine has been entered
cSE                          ! already and the code following the "if"
cSE                          ! statement containing this variable will not be
cSE                          ! executed again if it equals 1
cSE  .    Prd,               ! the current period in hours of the year.
cSE  .    PrvPd,             ! the previous period.  This variable remains
cSE                          ! as the previous period until a new period from
cSE                          ! the temporary file is encountered.  This is un-
cSE                          ! like iPrvPr which becomes the current period while
cSE                          ! processing the first receptor.
cSE  .BatOn(MGRP,MCAT)       ! the number of hours that an emission category
cSE                          ! stays on once on
cSE  .condit(MGRP)           ! indicates if the condition of the current
cSE                          ! source group is on
cSE  .Edist(MGRP)            ! index of the current emission category
cSE  .FOn(MGRP)              ! when this is YES it indicates the this is
cSE                          ! the first hour that the current source
cSE                          ! group is ON since it was OFF
cSE  .MK(MGRP)               ! if tera(iG) becomes YES then this will
cSE                          ! also become YES.  However, it does not
cSE                          ! become NO when tera(iG) becomes NO.
cSE                          ! Indicates the same thing as tera()
cSE  .nxtiR(Prd)             ! when mutually exclusive emissions are desired,
cSE                          ! it is assigned to jg to represent the source
cSE                          ! group that is ON for a particular period
cSE  .onTime(MGRP)           ! the amount of time in hours a group has
cSE                          ! been on for a particular emission category
cSE  .PrvLftO(MGRP,MREC)     ! the number of periods left over when the
cSE                          ! previous period entered this routine.  There
cSE                          ! is a value for each source group and each
cSE                          ! receptor
cSE  .state(MGRP,24)         ! switch indicating if the source group is
cSE                          ! on for each hour of a day
cSE  .tera(MGRP)             ! when tera() equals YES, an averaging period
cSE                          ! was completed prior to processing all the
cSE                          ! skipped periods.

cSE   real
```

```
cSE   . ON, YES,           ! when a variable equals 1.0, it is either
cSE                        ! ON or YES, respectively
cSE   . OFF, NO            ! when a variable equals 0.0, it is either
cSE                        ! OFF or NO, respectively.
cSE   . tmpy,              ! used when a source group is non-batch to
cSE                        ! keep track of the preprocessed probabilities
cSE                        ! of an emission category being used
cSE   .BatRate(MGRP,MHET,! emission rate for the source group, pollutant,
cSE   .        MCAT,2),    ! emission category, and period of day (AM or PM)
cSE   .dist(MGRP,MCAT),    ! probability of an emission category being used.
cSE                        ! These values are read in for all sources but are
cSE                        ! only used if the source group is non-batch
cSE   .randnum(2),         ! random number controlling which emission category
cSE                        ! will be used if mutually exclusive emissions are
cSE                        ! desired
cSE   .rndNos(MPER),       ! random numbers controlling whether a source
cSE                        ! group turns on or not, by period
cSE   .PdBRt(MGRP,         ! emission rate for each source group,
cSE   .      MHET,24),     ! and pollutant for each hour of the day

cSE    local parameters

       integer
      .    count, firstR, hr, imark, iPrvPr, iPol, iPriod,
      .    iSkip, iStart, j, jg, JK, loop1, Prd, PrvPd, x, ip, ifstsrc
       integer
      .    PrvLftO(MGRP,MREC), state(MGRP,24),
      .    onTime(MGRP), condit(MGRP),
      .    MK(MGRP), FOn(MGRP), tera(MGRP), nxtiR(MPER), hold(MGRP)
       real
      .    rndNos(MPER), PdBRt(MGRP,1,24), randnum(2),
      .    tmpy, ON, OFF, YES, NO, srange, Pick

       parameter (YES = 1, NO = 0, ON = 1, OFF = 0)

       save loop1, count, iPrvPr, PrvPd, PrvLftO, iSkip, state, JK

cSE    Assign "jg" to the incoming source group.  Note: if emissions
cSE    are mutually exclusive, "jg" at this point will always be 1.

       jg = iG

cSE    Since jg = 1 at this point, count must equal the total number
cSE    of source groups for proper decision making later if MEEm are
cSE    desired.

       if (MEEm .eq. YES) count = nSG

cSE    Determine how many periods will be left over for the current
cSE    TOXX source group and receptor.
cSE    When MEEm = YES, assign all leftovr() to leftovr(1) for all
cSE    source groups.  jg will = 1.

       leftovr(jg,iR) = mod(iP,iAvPer)
       if (MEEm .eq. YES .and. nSG .gt. 1) then
                     do 2 x = 2, nSG
                         leftovr(x,iR) = leftovr(jg,iR)
     2       continue
                     x = 0
       end if
```

```
cSE   If this is the first source group and the period is not equal to the
cSE   previous period, assign PrvPd to previous period, iPrvPr.

      if (jg .eq. 1 .and. iP .ne. iPrvPr) PrvPd = iPrvPr

cSE   If the period is the same as the previous period and all source
cSE   groups have entered this subroutine once, and tera(jg) and MK(jg)
cSE   are both NO (meaning that we are not currently involved with
cSE   averaging periods that have not been completed) then we do not
cSE   want to determine if this period is either ON or OFF because we have
cSE   already done so during this routines first entry when the current
cSE   period was not equal to the previous period.  Go to statement 63 to
cSE   by-pass the ON/OFF switching code.

      if (iP .eq. iPrvPr .and. count .eq. nSG .and. tera(jg) .eq. NO
     .    .and. MK(jg) .eq. NO) goto 63

cSE   If there were skipped periods that were leftover, assign iStart,
cSE   which is the beginning of the loop that increments the skipped
cSE   periods to the period just after the last period, to the next
cSE   period.  JK is an indicator of the number of periods that were
cSE   looped through.  We skip over the ON/OFF coding again.

      if (tera(jg) .eq. YES .or. MK(jg) .eq. YES) then
                    MK(jg) = YES
                    iStart = JK + 1
                    goto 64
      end if

cSE   If count is equal to the number of source groups, reset
cSE   the counter to 0 and start incrementing each time a different
cSE   source group goes through.  This is necessary when MEEm = NO.

      if (count .eq. nSG) count = 0
      count = count + 1

cSE   If the averaging period is equal to 1, set firstR to iR.

      if (iAvPer .eq. 1) firstR = iR

cSE   The previous period indicator iPrvPr gets set to the current period.

      iPrvPr = iP

cSE   Determine how many periods have been skipped since the last "hit".
cSE   We use PrvPd because iPrvPr is now equal to the current period.

      iSkip = iP - PrvPd

cSE   When this subroutine is first entered, loop1 equals NO and the
cSE   nested code will be executed.

      if (loop1 .eq. NO) then

cSE   Read in emission rates and ON TIME periods for each group.
cSE   Initialize rand(1).

          rand(1) = -1.0

cSE   Assign loop1 to YES so that we don't have to go through section
```

```
cSE   of the subroutine again.

                  loop1 = YES
      end if

cSE   Determine random numbers for all skipped periods if MEEm = NO.
cSE   If MEEm = YES, a different random number will be generated to
cSE   determine which source group will turn on.
cSE   Due to the peculiarities of the random number generator
cSE   RandNo, the number of random numbers requested from it must be
cSE   even.  Make sure iSkip is even and, if not, pass iSkip + 1
cSE   instead of iSkip.  In this case iSkip will still be used
cSE   below; however, one random number returned by RandNo will not
cSE   be used.

      if (MEEm .eq. NO) then
                  if (mod(iSkip,2) .eq. 0) then
                      call RandNo (iSkip,rndNos)
                  else
                      call RandNo (iSkip+1,rndNos)
                  end if
      end if

   63 iStart = 1

cSE   Loop through all skipped periods.

   64 do 30 iPriod = iStart, iSkip

cSE   Determine the current period.  Prd will be a number from
cSE   1 to 8784.

                  Prd = PrvPd + iPriod

cSE   Determine what the period would be on a 24 hour clock.

                  hr = mod(Prd,24)
                  if (hr .eq. 0) hr = 24

cSE   Determine which set of emission rates to use, AM or PM.
cSE   You can think of AM as hours where it is usually light outside.

                  if (hr .le. 6 .or. hr .ge. 19) then
                      sun = 2
                  else
                      sun = 1
                  end if

cSE   If the averaging period is 1 hour, and iR is the first receptor
cSE   encountered for this period, go to 65 and determine if it is ON
cSE   or OFF.  This has to be in here because when iAvPer = 1 all
cSE   receptors are not output to the temporary files.  For example
cSE   when there is multiple hour averaging, all receptors are available
cSE   in the temp. file for whatever period that there was a "hit".  With
cSE   1 hour averaging, we don't have to be concerned with receptors that
cSE   do not exceed the cutoff and iR may not start off with the first
cSE   receptor.

cSE   If the current period is the same as iPrvPr and all source
cSE   groups that are batch have entered this routine once during
```

```
cSE   this period and it is not the first receptor then we want to
cSE   skipped over the ON/OFF processing because we already know
cSE   if it is ON or OFF for this period and group.
cSE   The ".or. MEEm .eq. YES" code is included because count may
cSE   be equal to 1 when MEEm = YES.

                  if (iAvPer .eq. 1 .and. iR .eq. firstR) then
                      goto 65
                  else if (iP .eq. iPrvPr .and. ((MEEm .eq. NO .and.
     .        count .eq. nSG) .or. MEEm .eq. YES) .and. iR
     .    .gt. 1) then
                      goto 69
                  end if

cSE       Set nxtiR(Prd) to jg so that it is not zero.
                  nxtiR(Prd) = jg

cSE   FOn(jg) is indicator that this is the first hour that it is on.

cSE   The following section applies only to Batch processes.

   65     if (condit(jg) .eq. ON .and. Batch(jg) .eq. YES) then
                    state(jg,hr) = ON
                    if (FOn(jg) .eq. NO) onTime(jg) = onTime(jg) + 1
                    FOn(jg) = NO
   66       if (onTime(jg) .gt. BatOn(jg,Edist(jg)) .and.
     .      Edist(jg) .lt. iMax(jg)) then
                      Edist(jg) = Edist(jg) + 1
                      onTime(jg) = 1
                      if (BatOn(jg,Edist(jg)) .eq. 0) goto 66
                    else if (onTime(jg) .gt. BatOn(jg,Edist(jg))) then
                      condit(jg) = OFF
                      state(jg,hr) = OFF
                    end if

cSE   The following section applies only for non-Batch processes.

                  else if (condit(jg) .eq. ON .and. Batch(jg) .eq. NO) then
                      state(jg,hr) = ON
                      if (FOn(jg) .eq. NO) onTime(jg) = onTime(jg) + 1
                      FOn(jg) = NO
                      if (onTime(jg) .gt. BatOn(jg,Edist(jg))) then
                        condit(jg) = OFF
                        state(jg,hr) = OFF
                      end if
                  end if

cSE   The following determines if the processes should turn on if
cSE   it is OFF.

                  if (condit(jg) .eq. OFF) then
                      state(jg,hr) = OFF

cSE   IF MEEm = YES, select a random number to determine which
cSE   source group should turn on.

                    if (MEEm .eq. YES) then
                      if (rand(1) .eq. -1.0) then
                          call RandNo (2,rand)
                      end if
```

```
cSE             We first need to choose a random number to indicate
cSE             which source groups probability of turning on will
cSE             be compared to a random number chosen to determine
cSE             if the source group will turn on.  This will prevent
cSE             any bias in selecting source groups in numerical order.

cSE             Here we choose a random number with the Microsoft
cSE             intrinsic function RANDOM.

                      do ipos = 1, nSG
                         hold(ipos) = 0
                      end do

                      ifstsrc = 0
  68            call RANDOM (Pick)

cSE             Now we will start comparing the random number with
cSE             the first source groups range over all source groups.
cSE             If the random number chosen (i.e., iPick) is less than
cSE             then upper limit of the current source group range, than
cSE             we will choose this source groups probability to be
cSE             compared with another random number.

                      do 67 isource = 1, nSG
                         do ipos = 1, ifstsrc
                                    if (isource .eq. hold(ipos)) goto 67
                         end do
                         srange = float(isource)/float(nSG)
                         if (Pick .le. srange) then
                                    iSrG = isource
                                    ifstsrc = ifstsrc + 1
                                    hold(ifstsrc) = isource
                                    exit
                         end if
  67            continue

                      if (probOn(iSG(iSrG)) .lt. rand(1)) then

cSE             If the probability for the chosen source group
cSE             is less than the random number and we have looped
cSE             through all of the source groups, reset rand(1)
cSE             to -1.0 to indicate that there are no source groups
cSE             on.

                         iSrg = 0
                         if (ifstsrc .eq. nSG) then
                                    rand(1) = -1.0
                         else
                                    goto 68
                         end if
                      end if

cSE   Assign "jg" to the source group that was selected.
cSE   Put the value of "jg' into array nxtiR() to represent the
cSE   source group that is ON for this particular period.

                      jg = iSrG
                      nxtiR(Prd) = jg

                   end if
```

```
cSE   Execute the next statement if mutually exclusive emissions
cSE   are not desired or if they are desired and the probability
cSE   of one of the source groups turning on is greater than the
cSE   random number selected. (i.e., rand(1))

                    if ((MEEm .eq. NO .and. (probOn(iSG(jg))) .ge.
     .         rndNos(iPriod)) .or. rand(1) .ne. -1.0) then
                      iSrG = 0
                      rand(1) = -1.0
                      condit(jg) = ON
                      FOn(jg) = YES
                      Edist(jg) = 1
                      onTime(jg) = 1

cSE   For non-Batch processes, determine which emission category
cSE   is to be used by comparing a random number to the probability
cSE   that each emission category will be used.

                      if (Batch(jg) .eq. NO) then
                          Edist(jg) = 0
                    tmpy = 0.0

cSE   We only need one random number, but the random generator needs
cSE   two.

                           call RandNo (2, randnum)

                           do 40 j = 1, iMax(jg)

cSE   When an emission category has been selected, "exit" this do loop

                                        if (Edist(jg) .gt. 0) exit
                                        tmpy = tmpy + dist(jg,j)

cSE   If the random number selected is less than the cumulative
cSE   probability that an emission category will be used assign
cSE   Edist() to that category.

                                        if (randnum(1) .le. tmpy) Edist(jg) =
j
   40                 continue

cSE   The following is only executed for Batch processes.  It checks
cSE   to make sure that the time on (which will be 1 hour) does not
cSE   exceed the total amount of time allowed on.  In other words, it
cSE   makes sure that it will be on for at least 1 hour.

                  else if (onTime(jg) .gt. timeOn(jg)) then
                           condit(jg) = OFF
                           state(jg,hr) = OFF
                        end if
                      end if

cSE   If it is determined that the source is ON, go to statement 65.

                      if (condit(jg) .eq. ON) goto 65
                  end if

cSE   If we are at the last skipped period use the concentration from
cSE   the temporary file if there is one for the current receptor,
```

```
cSE   otherwise use the default value which is equal to the cutoff
cSE   used in ISCST2.

   69     if (MEEm .eq. YES .and. nxtiR(prd) .eq. 0) then
                  x = 0
                  goto 71
               end if

               if (iPriod .eq. iSkip .and. concen(jg,iR).gt.0.0) then
                  usert = concen(jg,iR)
               else
                  usert = defrt
               end if


               if (MEEm .eq. YES) x = 0

cSE   If emissions are mutually exclusive, only one source group will
cSE   be operating at a time, however, we still need to track what the
cSE   other source groups (if any.)  "x" is used to track which source
cSE   is currently being processed.  When "x" equals a source group
cSE   that is OFF (all but one source group), the "state" for that
cSE   source group and hour will be set to OFF (zero).  When it is
cSE   multiplied by the emission rate, it will produce no value.
cSE   When "x" equals the source group that is ON, "jg" is assigned
cSE   to the source group that is ON for this period (which is really
cSE   the same as "x".


   71     if (MEEm .eq. YES) then
                  x = x + 1
                  if (x .eq. nxtiR(Prd) .and. nxtiR(Prd) .ne. 0) then
                    jg = nxtiR(Prd)
                  else
                    jg = x
                    state(jg,hr) = OFF
                  end if
               end if

cSE   Process for each pollutant.
cSE   nPol = 1 in Sept. 30, 1993 version.

               do 70 iPol = 1, nPol

cSE   PdBRt() is set to the emission rate when iR is the first
cSE   receptor.  This way each group, pollutant and HOUR has an
cSE   emission rate assigned to it.  If iAvPer is 1 then each iR
cSE   gets assigned.

                  if (iR .eq. 1.and.Edist(jg).gt.0)
     .         PdBRt(jg,iPol,hr) = BatRate(jg,iPol,Edist(jg),sun)

                  if (iAvPer .eq. 1.and.Edist(jg).gt.0)
     .         PdBRt(jg,iPol,hr) = BatRate(jg,iPol,Edist(jg),sun)

cSE   If the previous averaging period resulted in leftover periods
cSE   (periods that were not used in the completed averaging period),
cSE   the number of periods for each group and receptor was assigned
cSE   to PrvLftO(jg,iR).  The first "if ... then" statement of the
cSE   following series of "if... then ... else" statements uses those
```

```
cSE   leftover periods in the next averaging period.

  72          if (PrvLftO(jg,iR) .gt. 0) then

cSE   For the first pollutant, increment PrvLftO(jg,iR) by 1 so that
cSE   it now represents the number of periods combined to form a
cSE   possible complete averaging period.  For example, if the averaging
cSE   period desired is 3 (i.e., iAvPer = 3) and during the last period
cSE   which had a hit we had 2 periods left over, we add 1 to it to
cSE   give us 3 hours in THIS (the current) period.  We only want to
cSE   add it during the loop of the first pollutant.

                          if (iPol .eq. 1) PrvLftO(jg,iR) = PrvLftO(jg,iR) + 1

cSE   If we now have enough hours to form a complete averaging period
cSE   process impSSG(iPol,iR) for this pollutant and this receptor.
cSE   First add tot(jg,iPol,iR) to the current emission and switch
cSE   data.  tot(jg,iPol,iR) is the accumulated concentration from the
cSE   left over periods for the current source group, pollutant, and
cSE   receptor.  usert, PdBRt(jg,iPol,hr) and state(jg,hr) have all been
cSE   updated for the current situation.
cSE   Divide impSSG(,) by the number of hours that make up an averaging
cSE   period (i.e., iAvPer) to produce the average impact for the single
cSE   source group.
cSE   Reset tot(,,) to zero.
cSE   Set "compl" to YES to indicate that an averaging period has
cSE   been completed for this period.

                          if (PrvLftO(jg,iR) .eq. iAvPer) then
                              impSSG(iPol,jg) = tot(jg,iPol,iR) + usert *
                     PdBRt(jg,iPol,hr) * state(jg,hr)
                              impSSG(iPol,jg) = impSSG(iPol,jg) / iAvPer
                              tot(jg,iPol,iR) = 0.0
                              compl = YES

cSE   When the last pollutant has been processed, check to see if
cSE   this is the last receptor.  If it is, we need to assign JK
cSE   to the period we got up to when a complete averaging period
cSE   was formed.  JK will be used the next time this subroutine
cSE   is entered.

cSE   If iPriod is less than the number of periods skipped (which means
cSE   that an averaging period has been completed prior to using up
cSE   all the skipped periods) set tera(jg) and SK to YES, other-
cSE   wise set tera(jg) to NO so that iStart will not be adjusted
cSE   by JK on the first time around the next period.

                              if (iPol .eq. nPol) then
                                        if (iR .eq. nRec) then

cSE   If MEEm = YES, tera(1) must also be assigned to YES because
cSE   when tera() is used to make the decision to skip the ON/OFF
cSE   processing, it sees "jg" only as 1.  We must make sure that
cSE   it is set properly.
cSE   When an averaging period is completed before using up all
cSE   the hours from 1 though iSkip, scndskp is assigned to the
cSE   difference between iSkip and the last iPriod used to complete
cSE   the completed averaging period (i.e., scndskp = iSkip - iPriod).
cSE   As always, the program will process all receptors and determine
cSE   if there are any exceedances for the completed averaging period.
```

```
cSE   It will then start on the next averaging period beginning with
cSE   the next available hour.  Without scndskp, the program would
cSE   check to make sure that there was enough hours in iSkip to
cSE   complete an averaging period.  This would be true even on the
cSE   second time around in some cases because iSkip does not change.
cSE   "scndskp"  has to be used instead of "iSkip"

                                            if (iPriod .lt. iSkip) then
                                                                   if (MEEm
.eq. YES) tera(1) = YES
                                                                   JK =
iPriod
                                                                   tera(jg)
= YES
                                                                   SK = YES
                                                                   scndskp
= iSkip - iPriod
                                            else
                                                                   tera(jg)
= NO
                                                                   scndskp
= iAvPer
                                            end if
                                        end if

cSE   The previously left over periods have now been used; set
cSE   PrvLftO(jg,iR) back to 0 and return to CaExEx to determine
cSE   if the averaging period for this source group and receptor
cSE   exceeds the thresholds.  Return without processing the
cSE   remaining skipped periods if MEEm = NO.  The remaining periods
cSE   will be taken care of later.
cSE   If MEEm = YES, go to statement 70 to continue processing the
cSE   remaining source groups before we return to subroutine CaExEx.
cSE   imark is set to YES to indicate at the end of this loop that
cSE   we must return to CaExEx.

                                        PrvLftO(jg,iR) = 0
                                        if (MEEm .eq. YES .and. x .ne. nSG)
then
                                            imark = YES
                                            goto 70
                                        else
                                            return
                                        end if
                                end if

cSE   If we still have not completed an averaging period, add the
cSE   current calculated concentration to the accumulated concentration.

                            else
                                tot(jg,iPol,iR) = tot(jg,iPol,iR) + usert *
                        PdBRt(jg,iPol,hr) * state(jg,hr)
                            end if

cSE   This is the second in a series of four "if ... then ... else"
cSE   statements.  If there were no previously left over periods AND
cSE   the current period (iPriod) is greater than the difference
cSE   between the number of skipped periods and the number of
cSE   hours in the desired averaging period AND then number of
cSE   skipped periods minus one less then the current period
```

```
      is less than or equal to the number of hours in an averaging
      period AND there are no left over periods, then we do the
      following.
      Multiply the concentration (usert) by the emission
cSE   rate for this source group, pollutant, and hour, and by
cSE   the switch indicating if this source group is ON or OFF.
cSE   If it is OFF impSSG(,,) will be 0.
cSE   When we encounter the last skipped period, calculate the
cSE   average impact.

                  else if (PrvLftO(jg,iR) .eq. 0 .and. iPriod .gt.
     .        (iSkip-iAvPer) .and. (iSkip-(iPriod-1) .le.
     .        iAvper) .and. leftovr(jg,iR) .eq. 0 .and. iSkip .ge.
     .        iAvPer .and. scndskp .ge. iAvPer) then
                      impSSG(iPol,jg) = impSSG(iPol,jg) + usert *
     .              PdBrt(jg,iPol,hr) * state(jg,hr)
                      if (iPriod .eq. iSkip) then
                          impSSG(iPol,jg) = impSSG(iPol,jg) / iAvPer
                          compl = YES
                          if (iR .eq. nRec) scndskp = iAvPer
                      end if

cSE   The following is the third of a series four "if ... then ...
cSE   else" statements.  The relevant code will be executed when
cSE   there are no periods left over from the previous period that
cSE   had a "hit" AND when the number of skipped periods is less
cSE   than the averaging period (this occurs more often with larger
cSE   averaging periods such as a 24 hour averaging period.)
cSE   Add to tot(jg,iPol,iR) the calculated impact for this
cSE   period.
cSE   If it is the last skipped period AND the last pollutant,
cSE   we can set PrvLftO(jg,iR) to iSkip so that the next time
cSE   around we know that there are skipped periods which did
cSE   not get incorporated into an averaging period.

                  else if (PrvLftO(jg,iR) .eq. 0 .and. (iSkip .lt.
     .        iAvPer .or. scndskp .lt. iAvPer)) then
                      tot(jg,iPol,iR) = tot(jg,iPol,iR) + usert *
     .            PdBRt(jg,iPol,hr) * state(jg,hr)
                      if (iPriod .eq. iSkip .and. iPol .eq. nPol) then
                          if (scndskp .lt. iAvPer) then
                                  PrvLftO(jg,iR) = scndskp
                          else
                                  PrvLftO(jg,iR) = iSkip
                          end if
                      end if

cSE   This is the last of four "if ... then ... else" statements.
cSE   If none of the other situations occur, the following code is
cSE   executed.
cSE   We are only concerned with number of periods that are left
cSE   over.  In other words the periods that did not fit into
cSE   an averaging period.  The period (iPriod) that is equal to
cSE   iSkip contains the concentration where a "hit" occurred, so
cSE   we must save it in tot(,,) to be use in the next round.
cSE   Set PrvLftO(,) to the current number of periods left over.

                  else if (iPriod .gt. iSkip-leftovr(jg,iR) .or. iSkip
     .        .lt. iAvPer) then
                      tot(jg,iPol,iR) = tot(jg,iPol,iR) + usert *
```

```
     .            PdBRt(jg,iPol,hr) * state(jg,hr)
                        if (iPol.eq.nPol.and.iPriod.eq.iSkip)
     .                PrvLftO(jg,iR)=leftovr(jg,iR)
                        end if
  70     continue

cSE   If MEEm = YES and
cSE   "x" is not equal to the last source group, go back to
cSE   statement 71 to work on the next source group.

                    if (MEEm .eq. YES .and. x .ne. nSG) goto 71

cSE   If MEEm = YES and imark was set to YES, reset imark to NO,
cSE   and return to CaExEx with out processing the remaining
cSE   skipped periods.  This will be taken care of later (after
cSE   all receptors have been processed for this period.)

                    if (MEEm .eq. YES .and. imark .eq. YES) then
                        imark = NO
                        return
                    end if

cSE   If the last period (iPriod) is equal to iSkip, then there is
cSE   no skipped periods that were missed so we can set tera(jg) to
cSE   NO.  Also set MK(jg) to NO if this is the last receptor so
cSE   that on the next entrance to this subroutine, iStart does
cSE   not begin where JK left off.

                    if (iPriod .eq. iSkip) then
                        if (MEEm .eq. YES) tera(1) = NO
                        tera(jg) = NO
                        if (iR .eq. nRec) then
                          if (MEEm .eq. YES) MK(1) = NO
                          MK(jg) = NO
                        end if
                    end if
cSE       Reset jg to the source group that is on for this period if a
cSE       source group is on.  Otherwise set it to the first source group.
cSE       This is only done when MEEm = YES.

                    if (MEEm .eq. YES) then
                        if (nxtiR(Prd) .ne. 0) then
                          jg = nxtiR(Prd)
                        else
                          jg = 1
                        end if
                    end if

  30  continue
      return
      end
cSE   end of subroutine BatPr

c==== subroutine RandNo ============================================
c
c Generate random numbers for purposes of simulating intermittent toxics
c emissions.
c
c (Written by John P. Nordin and Gary W. Lundberg, SAI.)
c
```

```
c Subroutine RandNo is called by subroutine OnOff.
c
c-----------------------------------------------------------------------
c
      subroutine RandNo (nRands, rands)
c
c-----------------------------------------------------------------------
c
c Declarations, etc.
c
c     Description of passed parameters.
c
c     integer
c    .    nRands           ! number of random numbers to generate;
c                          ! must be even
c     real
c    .    rands (nRands)   ! vector of real numbers; after subroutine
c                          ! execution, contains uniformly
c                          ! distributed numbers between 0 and 1.
c
      integer
     .    nRands
      real
     .    rands (nRands)
c
c     Local variables.
c
      integer
     .    i, j,
     .    kk,
     .    m,
     .    kount,
     .    remdr
c
c     Initializations.
c
      integer r(128)
      save r
      data r /31215,1416,28516,13922,1804,37,854,30144,-26368,119*0/
c
c-----------------------------------------------------------------------
c
c Create the random numbers.
c
      m = nrands/128
      kount  =  m*128
      remdr = nrands - kount
      if (m .lt. 1) goto 33
      do 20 i = 1,m
                  kk = 128*(i-1)
                  call Taus15(r)
                  do 25 j = 1,128
                      rands (kk+j) = (1.+float(r(j))) / 32768.
   25     continue
   20 continue
c
   33 call Taus15(r)
      do 30 i = 1,remdr
                  rands(kount+i) = (1.+float(r(i)))/32768.
   30 continue
```

```
c
      return
      end
c
c     end subroutine RandNo
c

c==== subroutine Taus15 ==========================================
c
c Help subroutine RandNo generate random numbers.
c
c (Written by John P. Nordin and Gary W. Lundberg, SAI.)
c
c Subroutine Taus15 is called by subroutine RandNo.
c
c-----------------------------------------------------------------
c
      subroutine Taus15 (s)
c
c-----------------------------------------------------------------
c
c Declarations, etc.
c
c     Description of passed parameters.
c
c     integer
c    .    s(128)            ! vector of integers
c
      integer
     .    s(128)
c
c     Description of local variables.
c
c     integer
c    .    i, nb, nn         ! variables used by Taus15
c
      integer
     .    i, nb, nn
c
c-----------------------------------------------------------------
c
c Do it.
c
c     s(9) is negative only on first call to Taus15.
c
      if (s(9) .lt. 0) goto 30
c
c     Main recurrence overwrites s(i+128) on s(i). Two loops and a
c     statement are used so that no subscript exceeds 128.
c
      do 10 i = 1,16
                  s(i) = ieor (s(i), ieor (s(i+1), s(i+112)) )
   10 continue
      do 20 i = 17,127
                  s(i) = ieor(s(i),ieor(s(i+1),s(i-16)))
   20 continue
      s(128) = ieor(s(128),ieor(s(1),s(112)))
      return
c
c      Initializing recurrence (used only on first call):
```

```
c      If s is set on first call of Taus15 to:
c      31215, 1416,28516,13922, 1804,   37,  854,30144, -26368,119*0;
c      then after first call s(1) to s(30) should be:
c      31215, 1416,28516,13922, 1804,   37,  854,30144,27699,  824,
c       3046,28554, 3460,  388, 9346,29289,12977,  439,14752,24447,
c      21860,20839,21806,22474,16047, 4012,14559, 6241,26571,  873
c
c      s(9) should be set to -26368 on machines using two's complement
c      representation (prime, ibm360/370) and -26367 on those
c      which uses one's complement notation (cdc7600, univac).
c      When first installing on a new machine check that the result
c      of the first call is as listed above.
c
   30 nn = s(9)
      do 40 i=9,128
                  nb = s(i-8)/128
                  s(i) = iand (32767,ieor (s(i-1),nn+nb))
                  nn = s(i-8)*256
   40 continue
c
      return
      end
c
c     end subroutine Taus15
c

c==== subroutine WrExEx ==========================================
c
c Write out results of exex calculation to printfile, either in
c tabular form.


c Subroutine WrExEx is called by subroutine DoExEx.
c
c-----------------------------------------------------------------------
c
      subroutine WrExEx (prtFil, HEMFil, yr, lyr, grp, lA, mG, mR,
     .                   nXR, mH, nH, lNH, iYr, nP, ICODE1, rHMC)
c
c-----------------------------------------------------------------------
c
c Declarations, etc.
c
c     Description of passed parameters.
c
c     integer
c     .    prtFil,       ! print fileunit
c     .    HEMFil,       ! HEM output fileunit
c     .    yr,           ! index of current metyear (yr = 0 -->
c                        ! do summary for all metyears)
c     .    grp,          ! current source group # (grp = 0 -->
c                        ! do summary for all source groups)
c     .    lT,           ! flag to make tabular output or not,
c                        ! and which kind of table to prepare
c     .    lA,           ! switch determining whether additive model used
c     .    mR,           ! max # receptors
c     .    nXR,          ! # receptors per ring or row
c     .    mH,           ! max # hets
c     .    nH,           ! # hets
c     .    lNH           ! logical # hets
```

```
c
c     (In the following three array names, the 'Wr' identifies each
c     variable as belonging to WrExEx.)
c
c     real
c    .   rHWr (mR,mH), ! exex array by receptor, by het, either
c                      ! for a single group and/or year or for
c                      ! all groups and/or years combined
c    .   mxHWr (nH)    ! max exex array by het, same comment as
c                      ! for rHWr
c
      integer
     .    prtFil, HEMFil, yr, grp, lA, mG,
     .    mR, nXR, mH, nH, lNH, nP

      real
     .    ICODE1(mG,mH,mR), rHMC(mR,mH)

c
c     (In the following three array names, the 'Wr' identifies each
c     variable as belonging to WrExEx.)
c
c
c     Description of local variables.
c
c     integer
c    .    iH,           ! index of current het
c     logical
c    .    noExEx        ! true if all elements of mxHWr are 0,
c                       ! false otherwise
c
      logical
     .    noExEx
c
c-----------------------------------------------------------------------
c
      noExEx = .FALSE.
               call WrTab1 (prtFil,grp,nP,yr,iYr,ICODE1,rHMC)

c
      return
      end
c
c     end subroutine WrExEx
c


cSE== subroutine WrTab1 ==============================================
cSE   called from subroutine WREXEX

      subroutine WrTab1 (prt,iGW,nP,yr,iYr,xXRCGY,xXTab6)

cSE   Called from subroutine WrExEx

c     integer
c    .    iH,                      ! index of current het
c    .    iR,                      ! index of current receptor
c    .    iYr                      ! current year
c     real
cSE  .    xXRCGY (MREC,MHET)       ! exex by receptor, by conc. cut range,
```

```
cSE                                     ! by source group for one metyear.

cSE   .    nRGrd                        ! the number of receptors modeled that
cSE                                     ! are in the grid.  The remaining
cSE                                     ! receptors will be discrete.

      include
     .    'TOXX.INS'

      parameter (NPAIR=100)
      integer
     .   prt,TabNum

      integer
     .    indexX(MREC), indexY(MREC)

      integer
     .    irow, icolm
      integer
     .    nRTab, nRGrd, totRec, RStrt, iGw
      integer
     .    iR, Yr

      integer
     .    nRRA

      real
     .    xXRCGY(MGRP,MHET,MREC),xXTab8(MGRP,MHET,MREC),
     .    xXTab11(MGRP,MHET,MREC),
     .    xXTab1(MREC,MHET),xXTab6(MREC,MHET),xXTab12(MREC,MHET),
     .    xXTab7(MREC,MHET),xXTab9(MREC,MHET),xXTab10(MREC,MHET),
     .    coordX(MREC), coordY(MREC), UTMX(MREC), UTMY(MREC),
     .    eH(MHET),Range(10),Direct(36)

      character AreaType*9
      character key*150, fmtstrg*30,LINE*9,LINE2*9
      character*1 FFEED*1,SWCrn,SECrn


      logical loop2

cSE   if loop2 is .true. then loop through to statement 64
      if (loop2) goto 64
      loop2=.TRUE.

      FFEED = char(12)
      SWCrn = char(200)
      SECrn = char(188)
      write (LINE(:),'(9(A1))') (char(196),i=1,9)
      write (LINE2(:),'(9(A1))') (char(205),i=1,9)


cSE   READ IN RECEPTOR LOCATION DATA.

cSE   Set marker iDiscrt to 1 indicating that there was not a grid system
cSE   modeled.  If the scan of the ISCST2 output file indicates that
cSE   a grid system was modeled, it will be changed to zero.  Can only
cSE   model one grid system with ISCST2 when using it with TOXX.

      if (iYr.eq.1) then  ! only do it for the first year modeled.
```

```
cSE   Set iDiscrt to 1 to indicate that no grid system has been
cSE   identified yet.

      iDiscrt=1

   05 read (1,'(a150)',end=64) key
      if (key(:19).eq.' BOUNDARY RECEPTORS') then
                    AreaType='BOUNDARY'
                    goto 10
      else if (key(50:77).eq.'DISCRETE CARTESIAN RECEPTORS') then
                    AreaType='DISCART'
                    goto 20
      else if (key(47:70).eq.'ORIGIN FOR POLAR NETWORK') then
                    AreaType='GRIDPOLAR'
                    goto 30
      else if (key(47:67).eq.'X-COORDINATES OF GRID') then
                    AreaType='GRIDCART'
                    goto 40
      else
                    goto 05    !  NO MATCH.  Continue looking.
      end if
      goto 05

C************* BOUNDARY RECEPTORS SECTION ****************************
   10 iDiscrt = 0  ! indicates a grid system was modeled.
      read (1,*)
      read (1,*)
      do iR = 1, 36, 3
                    read (1,111) UTMX(iR),UTMY(iR),UTMX(iR+1),UTMY(iR+1),
     .         UTMX(iR+2),UTMY(iR+2)
      end do
      nRGrd=36  !  FOR BOUNDARY RECEPTORS, ISCST2 MODELS FOR 36 RECEPTORS
      goto 05
  111 format (t6,f10.2,t18,f10.2,t51,f10.2,t63,f10.2,t96,f10.2,
     .    t108,f10.2)

C************** DISCRETE CARTESIAN RECEPTORS SECTION ******************
   20 do i=1,3
                    read (1,*)
      end do
   21 read (1,'(1x,a150)') key
      iDB=6  ! iDB is the first character to look at in the key variable
      iDE=14 ! iDE is the last character to look at in the key variable
      if (key(7:12).eq.'ISCST2') then  ! when key = ISCST2, stop reading
                    goto 24
      else if (key(iDB:iDE).ne.'          ') then
                    iR=iR+1
                    read (key(iDB:iDE),'(f9.1)') UTMX(iR)
                    iDB=iDB+11
                    iDE=iDE+11
                    read (key(iDB:iDE),'(f9.1)') UTMY(iR)
                    iDB=iDB+44
                    iDE=iDE+44
                    if (key(iDB:iDE).ne.'          ') then
                        iR=iR+1
                        read (key(iDB:iDE),'(f9.1)') UTMX(iR)
                        iDB=iDB+11
                        iDE=iDE+11
                        read (key(iDB:iDE),'(f9.1)') UTMY(iR)
                    end if
```

```
      end if
      goto 21
cSE   Set the number of discrete receptors to the number of receptors
cSE   iR is equal to.
   24 nRDisc=nRDisc+iR-nRGrd
      goto 05

C************ POLAR GRID RECEPTORS SECTION ****************************
   30 iDiscrt = 0  ! indicates a grid system was modeled.
      read (1,'(t41,f10.2,t65,f10.2)') XOrig,YOrig
      do i=1,4
                  read (1,*)
      end do
      read (1,'(1x,a150)') key

c     iDB == index of the range or direction beginning position
c     iDE == index of the range or direction ending position
c     iDRN == index of the range number
c     iDNN == index of the direction number

      iDB=5
      iDE=14
      iDRN=1
   31 if (key(iDB:iDE).ne.'           ') then
                  read (key(iDB:iDE),'(f10.1)') Range(iDRN)
                  iDRN=iDRN+1
                  iDB=iDB+11
                  iDE=iDE+11
                  goto 31
      end if

      iDRN=iDRN-1

      do i=1,4
                  read (1,*)
      end do
      iDDN=1
   32 read (1,'(1x,a150)') key
      iDB=5
      iDE=14

      if (key(7:12).ne.'ISCST2') then ! INDICATES THERE ARE NO MORE RADIALS
   33     if (key(iDB:iDE).ne.'           ') then
                     read (key(iDB:iDE),'(f10.1)') Direct(iDDN)
                     iDDN=iDDN+1
                     iDB=iDB+11
                     iDE=iDE+11
                     goto 33
                  else
                     goto 32
                  end if
      end if

      iDDN=iDDN-1

c     Calculate the number of receptors in the polar grid.
      nRGrd = iDRN*iDDN

      iR=0
      do i=1,iDDN    !  LOOP THROUGH THE RADIALS
```

```
c         Convert degrees to radians
                  Direct(i)= (2*3.1415927)/(360.0/Direct(i))
                  do j=1,iDRN     !  LOOP THROUGH THE RANGES
                     iR=iR+1
                     UTMX(iR)=Range(j) * sin(Direct(i))   ! CONVERT TO CART.
COORD
                     if (ABS(UTMX(iR)).lt.0.001) UTMX(iR)=0.0
                     UTMY(iR)=Range(j) * cos(Direct(i))   ! CONVERT TO CART.
COORD
                     if (ABS(UTMY(iR)).lt.0.001) UTMY(iR)=0.0
                  end do
      end do
      goto 05

C********** CARTESIAN GRID RECEPTORS SECTION **************************
   40 iDiscrt = 0  ! indicates a grid system was modeled.

      do i=1,2  ! skip over 6 lines of ISCST2 output file.
                  read (1,*)
      end do
      read (1,'(1x,a150)') key

c     iDB == index of the range or direction beginning position
c     iDE == index of the range or direction ending position
c     iDRN == index of the range number
c     iDNN == index of the direction number

      iDB=5
      iDE=14
      iDRN=1
   41 if (key(iDB:iDE).ne.'          ') then
                  read (key(iDB:iDE),'(f10.1)') Range(iDRN)
                  iDRN=iDRN+1
                  iDB=iDB+11
                  iDE=iDE+11
                  goto 41
      end if

      iDRN=iDRN-1
      do i=1,4
                  read (1,*)
      end do
      iDDN=1
   42 read (1,'(1x,a150)') key
      iDB=5
      iDE=14

      if (key(7:12).ne.'ISCST2') then ! INDICATES THERE ARE NO MORE RADIALS
   43    if (key(iDB:iDE).ne.'          ') then
                     read (key(iDB:iDE),'(f10.1)') Direct(iDDN)
                     iDDN=iDDN+1
                     iDB=iDB+11
                     iDE=iDE+11
                     goto 43
                  else
                     goto 42
                  end if
      end if

      iDDn=iDDN-1
```

```
c     Calculate the number of receptors in the polar grid.
      nRGrd = iDRN*iDDN

      iR=0
      do i=1,iDDN     !  LOOP THROUGH THE Y COORDINATES
                    do j=1,iDRN     !  LOOP THROUGH THE X COORDINATES
                        iR=iR+1
                        UTMX(iR)=Range(j)
                        if (ABS(UTMX(iR)).lt.0.001) UTMX(iR)=0.0
                        UTMY(iR)=Direct(i)
                        if (ABS(UTMY(iR)).lt.0.001) UTMY(iR)=0.0
                    end do
      end do
      goto 05
c*************************************************************************

      end if  ! end if for    if (iYr.eq.1) then

   64 continue    ! STANDALONE CONTINUE

      close (1)

cSE   Reset the counter for the new source group.

cSE   nRGrd is the number of receptors in the receptor grid.     This number
cSE   does not include the discrete receptors.

c     If there is only one source group modeled, iGW is passed over as
c     zero.  We want to let iGW indicate that it is the one and only
c     source group.

      iGG=iGW
      if (iGW.eq.0) iGW = 1

   17 nRTab = nRGrd
      totRec = nRGrd + nRDisc
      RStrt = 1

cSE   Loop through all the source groups.  iGW is an indicator of the
cSE   current source group.  This will be used in comparison with the
cSE   decoded source group number.

   50 continue  ! standalone continue

cSE   Sort the x- and y- receptor coordinates only once and only if
cSE   a grid system was modeled.
cSE   IGW will be zero when only one source group is modeled.

      if (iGW.eq.1.or.iGW.eq.0.and.iDiscrt.eq.0.and.iYr.eq.1) then
                    call SortTi2 (nRTab,UTMX,indexX)
                    call SortTi2 (nRTab,UTMY,indexY)
                    irow = 0
          icolm = 0

cSE       call xyTab to determine where the x- and y- coordinates go
cSE       on the output table.

                    call xyTab (UTMX,UTMY,coordX,coordY,RStrt,nRTab,indexX,
     .          indexY,irow,icolm)
```

```
c         Area of modeling domain converted into square kilometers.

          if (AreaType.eq.'DISCART'.or.AreaType.eq.'GRIDCART') then
                        GArea = ABS(((coordX(icolm)-coordX(1)) *
     .             (coordY(irow)-coordY(1)))/1000.0**2)
                      else if
(AreaType.eq.'GRIDPOLAR'.or.AreaType.eq.'DISCPOLAR')
     .    then
                        GArea = (3.1415927 * (Range(iDRN)-XOrig)**2)
     .             /1000.0**2
                      end if

                      do iH=1,logiNH
                          eH(iH)=effHET(iH)*10**6
                      end do
       end if

cSE    Zero out the array that contains the maximum number of exex
cSE    for each threshold and array that contains the number of exex
cSE    for each receptor and threshold, for this group.

   75 do 110 iH = 1, logiNH
                        if (lYrly.eq.1) then
                          do iR = RStrt,nRTab
                              xXTab1(iR,iH) = 0.
                              if (iGW.eq.1) then
                                          xXTab10(iR,iH) = 0.
                                          xXTab12(iR,iH) = 0.
                              end if
                          end do
                        end if
  110 continue

       do 210 iH = 1, logiNH
                      do 210 iR = RStrt, nRTab
                          xXRCGY(iGW,iH,iR) = xXRCGY(iGW,iH,iR) / nSim
                          xXTab6(iR,iH) = xXTab6(iR,iH) / nSim
  210 continue

       do 500 iH = 1, logiNH
c
cSE         Fill array to be printed out in Table.

                      do 517 iR = RStrt, nRTab
                          xXTab9(iR,iH)=xXTab9(iR,iH) + xXTab6(iR,iH) ! all
grps, all yrs

xXTab11(iGW,iH,iR)=xXTab11(iGW,iH,iR)+xXRCGY(iGW,iH,iR)
                        xXTab12(iR,iH)=xXTab12(iR,iH)+xXTab6(iR,iH)

                        if (iDiscrt.eq.1) then
                          do iInc=iH,logiNH
                              xXTab1(iR,iH) = xXTab1(iR,iH) +
     .                    xXRCGY(iGW,iInc,iR)
                              xXTab7(iR,iH) = xXTab7(iR,iH) +
     .                    xXTab6(iR,iInc) ! all grp all yrs
                              xXTab10(iR,iH) = xXTab10(iR,iH) +
     .                    xXTab6(iR,iInc) ! all grp one year
c                     xXTab8 () for single group, all years.
                              xXTab8(iGW,iH,iR) = xXTab8(iGW,
```

```
     .                    iH,iR) + xXRCGY(iGW,iInc,iR)
                          end do

                        else

cSE   loop on output table rows.
                          do 519 Y = 1, irow
                            if (UTMY(iR).gt.(coordY(Y)+0.02).or.UTMY(iR)
     .                   .lt.(coordY(Y)-0.02)) then
                                goto 519
                            end if
                            do 520 X = 1, icolm
                                if (UTMX(iR).gt.(coordX(X)-0.02).and.
     .                      UTMX(iR).lt.(coordX(X)+0.02)) then
                                        do iInc=iH,loginH
                                            xXTab1(iR,iH) = xXTab1(iR,iH) +
     .                          xXRCGY(iGW,iInc,iR)
                                            xXTab7(iR,iH) = xXTab7(iR,iH) +
     .                          xXTab6(iR,iInc) ! all grp all yrs
                                            xXTab10(iR,iH) = xXTab10(iR,iH)
+
     .                          xXTab6(iR,iInc) ! all grp one year
c                           xXTab8 () for single group, all years.
                                            xXTab8(iGW,iH,iR) = xXTab8(iGW,
     .                          iH,iR) + xXRCGY(iGW,iInc,iR)
                                        end do
                                        EXIT
                                end if
  520               continue
  519           continue
                          end if
  517       continue

  500 continue


c*****This 500 series section prints out table for each source group
c*****for each year as they are processed if the user requests output from
c*****each year modeled and there are more than one source groups
c*****and more than one year modeled.

c       If the user wants output for each year and there is more than
c       one year modeled, do the 500 and 600 series.

        if (lYrly.eq.1.and.nYear.gt.1) then


c******************** 500 SERIES ************************************
        if (iDiscrt.ne.1.and.iGP.eq.0) then

                    call WrTab (500,irow,icolm,istprt,nColCR,iGW,
     .          RStrt,nRTab,UTMY,UTMX,coordY,coordX,eH,xXTab1,
     .          xXRCGY,xXTab7,xXTab11,fmtstrg,GArea,nP,prt,yr,TabNum)

        else !  print out for discrete receptors.

            if (iGP.eq.0) then

                    write (prt,'(a1)') FFEED
                    TabNum=TabNum+1
```

```
                    write (prt,'(//25X,A6,I2//)') 'TABLE ',TabNum
                    write (prt,9102) sName(iGW),yr
                    write (prt,*)
          write (prt,9110) (eH(iH),iH=1,logiNH)
                    do iR=RStrt,nRTab
                        do iH=1,logiNH
                          xXTab1(iR,iH)=xXTab1(iR,iH)
                        end do
                        write (prt,9100)
iR,UTMX(iR),UTMY(iR),(xXTab1(iR,iH),iH=
     .             1,logiNH)
          end do

          end if

      end if

c********************** 600 SERIES ******************************
c*****This section will print out a table for each year for all source
c*****source groups combined when the current source group is equal
c*****to the last source group modeled and only if there is a total of
c*****more than one source group modeled.

      if (iGW.eq.nSG.and.nSG.gt.1) then

                    if (iDiscrt.ne.1) then
                        call WrTab (600,irow,icolm,istprt,nColCR,iGW,
     .         RStrt,nRTab,UTMY,UTMX,coordY,coordX,eH,xXTab10,
     .         xXRCGY,xXTab12,xXTab11,fmtstrg,GArea,nP,prt,yr,TabNum)
                    else
                        write (prt,'(a1)') FFEED
                        TabNum=TabNum+1
                        write (prt,'(//25X,A6,I2//)') 'TABLE ',TabNum
                        write (prt,9104) yr
                        write (prt,*)
              write (prt,9110) (eH(iH),iH=1,logiNH)
                        write (prt,*)
                        do iR = RStrt, nRTab
                          do iH=1,logiNH
                              xXTab10(iR,iH)=xXTab10(iR,iH)
                          end do
                          write (prt,9100)
iR,UTMX(iR),UTMY(iR),(xXTab10(iR,iH),
     .                  iH=1,logiNH)
                          end do
                    end if

      end if

      end if  ! end if for        if (lYearly.eq.1) then

c     Begin working on discrete receptors if there are any and they
c     haven't already be processed.
      if (nRec.gt.nRTab) then
                    RStrt = nRTab+1
                    nRTab = nRec
                    iDiscrt=1
                    goto 75
      else
                    iDiscrt=0
```

```
      end if

c     If the source group index iGW is equal to the total number of
c     source groups modeled, then we have loop though all the source
c     groups for the current year and can output all source groups
c     combined data for the current year.

      if (iGW.eq.nSG.and.iYr.eq.nYear) then

                  nRTab = nRGrd
                  RStrt = 1
                  iDiscrt = 0

   76     if (nSG.gt.1.and.iGP.eq.0) then

                     if (iDiscrt.ne.1) then
                       call WrTab (700,irow,icolm,istprt,nColCR,iGW,
     .                RStrt,nRTab,UTMY,UTMX,coordY,coordX,eH,xXTab1,
     .                xXTab8,xXTab7,xXTab11,fmtstrg,GArea,nP,prt,yr,
     .                TabNum)
             else
                 if (iGP.eq.0) then
                     do iGW2 = 1, nSG
                         write (prt,'(a1)') FFEED
                         TabNum=TabNum+1
                         write (prt,'(//25X,A6,I2//)') 'TABLE ',TabNum
                         if (nYear.gt.1) then
                                 write (prt,9101)
sName(iGW2),iYear(1),
     .                    iYear(nYear)
                         else
                                 write (prt,9102) sName(iGW2),yr
                         end if
                         write (prt,*)
                   write (prt,9110) (eH(iH),iH=1,logiNH)
                         write (prt,*)
                         do iR = RStrt, nRTab
                                 do iH=1,logiNH

xXTab8(iGW2,iH,iR)=xXTab8(iGW2,iH,iR)/
     .                           nYear
                                 end do
                                 write (prt,9100)
iR,UTMX(iR),UTMY(iR),
     .                     (xXTab8(iGW2,iH,iR),iH=1,logiNH)
                         end do
                 end do
                 end if
                     end if
          end if  !  only do the above 700 series if nSG > 1.
          if (iDiscrt.ne.1) then
                     call WrTab (800,irow,icolm,istprt,nColCR,iGW,
     .            RStrt,nRTab,UTMY,UTMX,coordY,coordX,eH,xXTab7,
     .            xXRCGY,xXTab9,xXTab11,fmtstrg,GArea,nP,prt,yr,TabNum)
               else
                     write (prt,'(a1)') FFEED
                     TabNum=TabNum+1
                     write (prt,'(//25X,A6,I2//)') 'TABLE ',TabNum
                     if (nYear.gt.1) then
                       write (prt,9103) iYear(1),iYear(nYear)
```

```
                        else
                          write (prt,9104) yr
                        end if
                        write (prt,*)

                        write (prt,9110) (eH(iH),iH=1,logiNH)
                        write (prt,*)
                        do iR = RStrt, nRTab
                          do iH=1,logiNH
                              xXTab7(iR,iH)=xXTab7(iR,iH)/nYear
                          end do
                          write (prt,9100) iR,UTMX(iR),UTMY(iR),(xXTab7(
     .                  iR,iH),iH=1,logiNH)
                          end do
                      end if

 9101     format (t5,'Expected Exceedances for discrete receptors ',
     .            'above thresholds:',/
     .            t5,'Source: ',a20,/
     .            t5,'Years: ',i4.4,'-',i4.4)
 9102     format (t5,'Expected Exceedances for discrete receptors ',
     .            'above thresholds:',/
     .            t5,'Source: ',a20,/
     .            t5,'Year: ',i4.4)
 9103     format (t5,'Expected Exceedances for discrete receptors ',
     .            'above thresholds:',/
     .            t5,'All sources combined: ',/
     .            t5,'Years: ',i4.4,'-',i4.4)
 9104     format (t5,'Expected Exceedances for discrete receptors ',
     .            'above thresholds:',/
     .            t5,'All sources combined: ',/
     .            t5,'Year: ',i4.4)
 9110     format (t6,'Rec #',5x,'X-Coord',8x,'Y-Coord',t49,
     .        6('>',f6.1,3x))

                      if (nRec.gt.nRTab) then
                          RStrt = nRTab+1
                          nRTab = nRec
                          iDiscrt=1
                          goto 76
                      else
                          iDiscrt=0
                      end if

      end if

c      DEALLOCATE (indexX,indexY,STAT=error)
      return
 9100 format (t5,i3,5x,f10.2,5x,f10.2,10x,6(f8.2,2x))
      end
cSE== end subroutine WrTab1 ============================================


cSE== subroutine WrTab ================================================
cSE   subroutine WrTab called by subroutine WrTab1

      subroutine WrTab (LNNum,irow,icolm,istprt,nColCR,iGW,
     .    RStrt,nRTab,UTMY,UTMX,coordY,coordX,eH,TabA,TabAA,
     .    TabBB,TabCC,fmtstrg,GArea,nP,prt,yr,TabNum)
```

```
      include
     .    'TOXX.INS'

      integer
     .    LNNum,iH,prtAlr,istprt,nColCR,icolm,RStrt,nRTab,
     .    X,Y,iR1,ibInc,prt,Yr,iStrt,iEnd,TabNum

      real
     .    TabA(MREC,MHET),TabAA(MGRP,MHET,MREC),TabCC(MGRP,MHET,MREC),
     .    TabBB(MREC,MHET),UTMX(MREC),UTMY(MREC),coordX(MREC),
     .    coordY(MREC),eH(MHET),GArea

cNOTE  buff is dimensioned to 20 + 1, the 1 is for the coordY variable
cNOTE that is printed out.

      character
     .    buff(20+1)*13,fmtstrg*30,fmtstrg3*30,FFEED*1,LSpc*1,RSpc*1,
     .    fmtstrg4*120

      FFEED = char(12)

      iTotGrp = 1
      if (LNNum.eq.700) then
                  iStrt=1
                  iEnd=nSG
      else
                  iStrt=iGW
                  iEnd=iGW
      end if

      do 10 iG2=iStrt,iEnd

      do 516 iH  = 1, logiNH  ! loop on HET's

          iTablchk=0
          do 530 iR= RStrt, nRTab
              if (LNNum.ne.700) then
                  if (TabA(iR,iH) .gt. 0.) then
                      iTablchk=1
                      exit
                  end if
              else
                  if (TabAA(iG2,iH,iR) .gt. 0.) then
                      iTablchk=1
                      exit
                  end if
              end if
  530     continue

          if (iTablchk .eq. 0) then
              write (prt,9105) iG2, iH
              goto 516
          end if

                    TabB = 0.0

                    prtAlr=0  ! set # of receptors PRinTed ALReady to 0
                    istprt=0  ! set the Index of the STart PRinT to 0
                    nColCR=0  ! set the Number of COLumns before Carriage
Return 0

      •
```

```
                    iRxX=0

   78     if (prtAlr.lt.icolm) then

                        istprt=prtAlr+1

                        if ((icolm-prtAlr).lt.MRECPR) then
                          nColCR=icolm
                          prtAlr=prtAlr+(icolm-prtAlr)
                        else
                          nColCR=nColCR+MRECPR
                          prtAlr=prtAlr+MRECPR
                        end if

                        call WrTHdr1
(LNNum,Yr,iH,coordX,prt,istprt,nColCR,eH,iGW,
     .             iG2,fmtstrg,icolm,TabNum)

                        do 591 Y = irow, 1, -1  ! loop on the rows in the
table
                          call TABLINE
(10,istprt,nColCR,icolm,fmtstrg,prt,LSpc,
     .                RSpc)
                          do ibInc=1,MRECPR+1
                            buff(ibInc) = '          '
                          end do
                          iR1=RStrt-1
  599             iR1=iR1+1

c                 The following if statement is basically the same
c                 as saying if UTMY = coordY.  The +- 0.02 takes
c                 care of any trig. calc. differences.

                          if (
     .                   (UTMY(iR1).gt.(coordY(Y)+0.02)
     .                   .or.UTMY(iR1).lt.(coordY(Y)-0.02))
     .                   ) goto 599
                          ibInc=1
                          write (buff(ibInc),'(2X,F9.1,2X)') coordY(Y)
                          do 593 X=istprt,nColCr
                            ibInc=ibInc+1
                            do 590 iR = RStrt, nRTab
                                    if
((UTMX(iR).gt.(coordX(X)-0.02).and.
     .                          UTMX(iR).lt.(coordX(X)+0.02)) .and.
     .                          (UTMY(iR).gt.(coordY(Y)-0.02)
     .                          .and.UTMY(iR).lt.(coordY(Y)+0.02)))
     .                      then
                                        if
(LNNum.eq.500.or.LNNum.eq.600.or.
     .                             LNNum.eq.800) then
                                                          if
(LNNum.eq.800)
     .                                TabA(iR,iH)=TabA(iR,iH)/nYear

TabB=TabB+TabA(iR,iH)
                                                          write
(buff(ibInc),'(2X,F7.1)')
     .                                TabA(iR,iH)
                                                          if
```

```
(TabA(iR,iH).ge.1.0) iRxX=iRxX+1
                                                    else if (LNNum.eq.700) then
                                          TabB=TabB+TabAA(iG2,iH,iR)
                                                                          write
(buff(ibInc),'(2X,F7.1)')
     .                                       TabAA(iG2,iH,iR)
                                                                          if
(TabAA(iG2,iH,iR).ge.1.0) iRxX=
     .                                       iRxX+1
                                                    end if
                                                    EXIT
                                                end if
  590                 continue
  593               continue
                          write (fmtstrg3(:),'(a8,i2,a11)')
     .                   '(a1,a13,',ibInc-1,'(a9),1x,a1)'
                          write (prt,fmtstrg3) LSpc,(buff(i),i=1,ibInc),RSpc
  591           continue
                          call TABLINE (20,istprt,nColCR,icolm,fmtstrg,prt,LSpc,
     .                RSpc)
                          goto 78
                        end if

            if (GArea.ne.0.) write
     .          (prt,'(//T5,A,F10.5,A)') 'Total area of grid: ',GArea,
     .          ' sq. km.'

                      write (prt,'(T5,A,A,I3)') 'Number receptors with at least
',
     .          'one exceedance: ',iRxX
                          write (prt,1510) TabB
              if(GArea .ne. 0.) then
                TabB = TabB/GArea
              end if
              if (GArea.ne.0.) write (prt,1517) TabB
              if(nRTab .eq. 0) then
                          TabB = 0.
                        else
                          TabB = TabB/nRTab
              end if
              if (GArea.ne.0.) then
                  TabB = TabB * GArea
                  write (prt,1520) TabB
              else
                  write (prt,1520) TabB  ! when BOUNDARY file
              end if
  516 continue  !  loop on HET's

 1510 format (T5,'Total exceedances in above grid: ',F10.5)
 1517 format (T5,'Average exceedance in above grid: ',F10.5,' per sq.'
     .    ' km.')
 1520 format (T5,'Average exceedances per receptor in above grid: '
     .    ,F10.5, ' exceedances per receptor.')


c     Write out frequency table.

      NoTab=0
      do 518 iR = RStrt, nRTab
          if (NoTab.eq.1) goto 518
```

```
      if (iR.eq.1) then
          do 523 iR2=RStrt,nRTab
              do 524 iH=1,logiNH
                 if (LNNum.eq.500.and.TabAA(iGW,iH,iR2).gt.0..and.
     .                iGP.eq.0) then
                     NoTab=0
                     goto 777
                 else if (LNNum.eq.700.and.TabCC(iG2,iH,iR2).gt.0.
     .                .and.iGP.eq.0)
     .           then
                     NoTab=0
                     goto 777
                 else if ((LNNum.eq.600.or.LNNum.eq.800).and.
     .               TabBB(iR2,iH).gt.0.) then
                     NoTab=0
                     goto 777
                 end if
524           continue
523       continue
          NoTab=1
          goto 518

777       write (prt,'(a1)') FFEED
          TabNum=TabNum+1
          write (prt,'(//25X,A6,I2//)') 'TABLE ',TabNum
          if (LNNum.eq.500.or.LNNum.eq.600.or.nYear.eq.1) then
              write (prt,'(//a,a,I4,a)')
     .            ' Number of exceedances within each HET ',
     .            'category.  (Year: ',yr,')'
          else
              write (prt,'(//a,a,i4,a1,i4,a)') ' Number of exceed',
     .            'ances within each HET category.  (Years: ',
     .            iYear(1),'-',iYear(nYear),')'
          end if

          if (LNNum.eq.500) then
              write (prt,'(a,a20/)') ' Source: ',sName(iGW)
          else if (LNNum.eq.700) then
              write (prt,'(a,a20/)') ' Source: ',sName(iG2)
          else
              write (prt,'(a/)') ' All sources combined'
          end if

          write (prt,'(t53,a)') '(ug/m^3)'

          if (logiNH.eq.1) then
              inHET=1
              write (fmtstrg4(:),'(a120)') '(5x,''Rec #'',
     .        5x,''X-Coord'',8x,''Y-Coord'',t53,''0.0-'',3x,
     .        ''>'',f6.1)'
              write (prt,fmtstrg4) eH(inHET)
          else
              inHET=logiNH-1
              write (fmtstrg4(:),'(a80,i1,a23)') '(5x,''Rec #'',
     .        5x,''X-Coord'',8x,''Y-Coord'',t53,''0.0-'',3x,',
     .        inHET,'(f6.1,''-'',3x),''>'',f6.1)'
              write (prt,fmtstrg4) (eH(iH),iH=1,(inHET)),
     .            eH(logiNH)
          end if
```

```
            write (prt,1516) (eH(iH),iH=1,logiNH)
            write (prt,*)

        end if  ! for if (iR.eq.1)

        BeloweH = 0.

        do iH=1,logiNH
            if (LNNum.eq.500) then
                BeloweH = BeloweH + TabAA(iGW,iH,iR)   ! xXRCGY
            else if (LNNum.eq.700) then
                TabCC(iG2,iH,iR) = TabCC(iG2,iH,iR) /nYear
                BeloweH = BeloweH + TabCC(iG2,iH,iR)   ! xXRCGY
            else if (LNNum.eq.600) then
                BeloweH = BeloweH + TabBB(iR,iH)   ! xXTab6
            else if (LNNum.eq.800) then
                TabBB(iR,iH) = TabBB(iR,iH) / nYear
                BeloweH = BeloweH + TabBB(iR,iH)   ! xXTab9
            end if
        end do
        BeloweH = nP/iAvPer - BeloweH

        if (LNNum.eq.500) then
            write (prt,1530) iR,UTMX(iR),UTMY(iR),BeloweH,
     .           (TabAA(iGW,iH,iR),iH=1,logiNH)
        else if (LNNum.eq.700) then
            write (prt,1530) iR,UTMX(iR),UTMY(iR),BeloweH,
     .           (TabCC(iG2,iH,iR),iH=1,logiNH)
        else if (LNNum.eq.600.or.LNNum.eq.800) then
            write (prt,1530) iR,UTMX(iR),UTMY(iR),BeloweH,
     .           (TabBB(iR,iH),iH=1,logiNH)
        end if
        BeloweH = 0.
  518 continue
   10 continue  !  loop on iG2

 9105 format (//t5,'No Expected Exceedances for source group ',I1,
     .        ', threshold ',I1,'.')
 1500 format (2X,F9.1,2X,32(2X,F7.1))
 1516 format (t50,6(f6.1,4x))
 1530 format (T5,I3,5X,F10.2,5X,F10.2,10X,7(f8.2,2X))

      end
cSE== end WrTab subroutine ==============================================

cSE== subroutine WrTHdr1 ================================================

      subroutine WrTHdr1 (LNNum, yr, iH, coordX, prt, istprt, nColCR,
     .    eH, iGW, iG2, fmtstrg, icolm,TabNum)

      include
     .   'TOXX.INS'

      integer LNNum,TabNum,yr,prt,iclm,istprt,nColCR

      real coordX(MREC), eH(MHET)

      character FFEED*1,VERTL*1,LINE*9,LINE2*6,fmtstrg*30,fmtstrg2*30,
     .    srcnme*20
```

```
      FFEED = char(12)
      VERTL = char(186)
      NWCnr = char(201)
      NECnr = char(187)
      SWCnr = char(200)
      SECnr = char(188)
      YArw = char(31)
      XArw = char(16)
      write (LINE(:),'(9(A1))') (char(205),i=1,9)
      write (LINE2(:),'(6(A1))') (char(205),i=1,6)

      write (prt,'(a1)') FFEED

cSE   Increment the Table number by one for the next Table if we
c     are finished printing out the previous table.

      if (istprt.eq.1) TabNum = Tabnum+1

      if (istprt.eq.1) then
                  write (prt,'(//25X,A6,I2)') 'TABLE ',TabNum
      else
                  write (prt,'(//25X,A6,I2,A)') 'TABLE ',TabNum,
     .        ' (continued)'
      end if

      if (LNNum.eq.600.or.LNNum.eq.800) then
                  srcnme=' '
                  if (LNNum.eq.800.and.nYear.gt.1) then
                      assign 1109 to ifmt
                  else
                      assign 1110 to ifmt
                  end if
      else if (LNNum.eq.700) then
                  srcnme=sname(iG2)
                  if (nYear.gt.1) then
                      assign 1112 to ifmt
                  else
                      assign 1111 to ifmt
                  end if
      else  ! when LNNum = 500
                  srcnme=sname(iGW)
                  assign 1111 to ifmt
      end if

                  if (nYear.eq.1.or.LNNum.eq.500.or.LNNum.eq.600) then
                      write (prt,ifmt) pName(1), iH, eH(iH), srcnme, yr
                  else
                      write (prt,ifmt) pName(1), iH, eH(iH), srcnme,
iYear(1),
     .            iYear(nYear)
                  end if

 1109 format (//5X,'Average Exceedances of Pollutant: ',A20,
     .    /5X,'Above threshold number ',I1,' ( > ' ,f7.2,' ug/m3)',
     .    /5X,'All source groups combined.',A20,
     .    /5X,'Year: ',I4.4,'-',I4.4)

 1110 format (//5X,'Average Exceedances of Pollutant: ',A20,
     .    /5X,'Above threshold number ',I1,' ( > ' ,f7.2,' ug/m3)',
     .    /5X,'All source groups combined.',A20,
```

```
     .    /5X,'Year: ',I4)
 1111 format (//5X,'Average Exceedances of Pollutant: ',A20,
     .    /5X,'Above threshold number ',I1,' ( > ' ,f7.2,' ug/m3)',
     .    /5X,'For source group: ',A20,
     .    /5X,'Year: ',I4)
 1112 format (//5X,'Average Exceedances of Pollutant: ',A20,
     .    /5X,'Above threshold number ',I1,' ( > ' ,f7.2,' ug/m3)',
     .    /5X,'For source group: ',A20,
     .    /5X,'Years: ',I4.4,'-',I4.4)

      write (prt,1210)
 1210 format (//1x,' Y-Coordinate (m)',35X,' X-Coordinate (m)')
      iHORZL1=nColCR-istprt+2
      iHORZL2=nColCr-istprt+1
      iHORZL3=nColCr-istprt+3
      if (istprt.eq.1.and.nColCR.ne.icolm) then
                    write (fmtstrg(:),'(a4,I2,A8)') '(A1,',iHORZL1,'(A9),A5)'
                    write (prt,fmtstrg(:)) NWCnr,(LINE,i=istprt,nColCR+1),LINE2
                    write (fmtstrg2(:),'(a8,i2,a10)') '(a1,13x,',iHORZL2,
     .         '(1x,f8.1))'
                    write (prt,fmtstrg2)
VERTL,(coordX(iclm),iclm=istprt,nColCR)
      else if (nColCR.eq.icolm.and.istprt.ne.1) then
                    write (fmtstrg(:),'(a1,I2,A11)') '(',iHORZL1,'(A9),A6,A1)'
                    write (prt,fmtstrg(:))
(LINE,i=istprt,nColCR+1),LINE2,NECnr
                    write (fmtstrg2(:),'(a5,i2,a19)') '(14x,',iHORZL2,
     .         '(1x,f8.1),1x,a1)'
                    write (prt,fmtstrg2)
(coordX(iclm),iclm=istprt,nColCR),VERTL
      else if (istprt.eq.1.and.nColCr.eq.icolm) then
                    write (fmtstrg(:),'(A7,I2,A8)')
'(A1,A5,',iHORZL1,'(A9),A1)'
                    write (prt,fmtstrg(:))
NWCnr,(LINE,i=istprt,nColCR+2),NECnr
                    write (fmtstrg2(:),'(a8,i2,a16)') '(a1,13x,',iHORZL2,
     .         '(1x,f8.1),1x,a1)'
                    write (prt,fmtstrg2)
VERTL,(coordX(iclm),iclm=istprt,nColCR),
     .         VERTL
      else
          write (fmtstrg(:),'(A1,I2,A5)') '(',iHORZL3,'(A9))'
          write (prt,fmtstrg(:)) (LINE,i=istprt,nColCR+2)
                    write (fmtstrg2(:),'(a4,i2,a10)') '(14x,',iHORZL2,
     .         '(1x,f8.1))'
                    write (prt,fmtstrg2) (coordX(iclm),iclm=istprt,nColCR)
      end if

      return
      end
cSE== end subroutine ================================================


cSE== SUBROUTINE TABLINE ============================================

      subroutine TABLINE (LNNum,istprt,nColCR,icolm,fmtstrg,prt,LSpc,
     .    RSpc)

      integer
     .    LNNum,iVBL,istprt,nColCR,icolm,prt
```

```
      character
     .    fmtstrg*30,LINE*9,LINE2*9,LftM,RgtM

      character*1
     .    SWCrn,SECrn,LSpc,RSpc

      LftM = char(199)
      RgtM = char(182)

      if (LNNum.eq.10) then
                  assign 10 to iVBL
      else if (LNNum.eq.20) then
                  assign 20 to iVBL
      end if

      goto iVBL (10,20)

 10   continue  ! print out lines in center of table
      write (LINE(:),'(9(A1))') (char(196),i=1,9)

      if (istprt.eq.1.and.nColCR.ne.icolm) then
                  write (prt,fmtstrg) LftM,(LINE,i=istprt,
     .        nColCR+1),LINE
                  LSpc = char(186)
                  RSpc = ' '
      else if (nColCr.eq.icolm.and.istprt.ne.1) then
                  write (prt,fmtstrg) (LINE,i=istprt,nColCR+1),
     .        LINE,RgtM
                  LSpc = ' '
                  RSpc = char(186)
      else if (istprt.eq.1.and.nColCR.eq.icolm) then
                  write (prt,fmtstrg) LftM,(LINE,i=istprt,
     .        nColCR+2),RgtM
                  LSpc = char(186)
                  RSpc = char(186)
      else
         write (prt,fmtstrg) (LINE,i=istprt,nColCR+2)
                  LSpc = ' '
                  RSpc = ' '
      end if

      return


 20   continue  ! print out line at bottom of table

      SWCrn = char(200)
      SECrn = char(188)
      write (LINE2(:),'(9(A1))') (char(205),i=1,9)

      if (istprt.eq.1.and.nColCR.ne.icolm) then
                  write (prt,fmtstrg) SWCrn,(LINE2,i=istprt,
     .        nColCR+1),LINE2
      else if (nColCr.eq.icolm.and.istprt.ne.1) then
                  write (prt,fmtstrg) (LINE2,i=istprt,nColCR+1),
     .        LINE2,SECrn
      else if (istprt.eq.1.and.nColCR.eq.icolm) then
                  write (prt,fmtstrg) SWCrn,(LINE2,i=istprt,
     .        nColCR+2),SECrn
      else
```

```
          write (prt,fmtstrg) (LINE2,i=istprt,nColCR+2)
        end if

        return
        end
cSE== end subroutine ================================================

cSE== subroutine xyTab ==============================================

        subroutine xyTab (UTMX,UTMY,coordX,coordY,RStrt,nRTab,indexX,
       .                  indexY,irow,icolm)


        include
       .    'TOXX.INS'

        integer
       .    indexX(MREC),indexY(MREC),RStrt,nRTab,irow,icolm,iR

        real
       .    UTMX(MREC),UTMY(MREC),
       .    coordX(MREC),coordY(MREC)

cSE     Determine the x coordinates for the table.  The x
cSE     coordinates will be for all x and y values.

        do 100 iR = RStrt, nRTab
                    if (iR.ne.1) then
                        do iR2 = RStrt, (iR-1)
c                   if (iR.ne.iR2) then
                            if (UTMX(indexX(iR)).gt.(UTMX(indexX(iR2))-0.02)
       .                       .and.UTMX(indexX(iR)).lt.(UTMX(indexX(iR2))
       .                       +0.02)) then
                                    exit  ! exit from this loop
                               else if (iR2.eq.(iR-1)) then
                                    icolm = icolm + 1
                                    coordX(icolm) = UTMX(indexX(iR))
                                    exit
                               end if
c                   end if
                        end do
                        do iR2 = RStrt, (iR-1)
c                   if (iR.ne.iR2) then
                            if (UTMY(indexY(iR)).gt.(UTMY(indexY(iR2))-0.02)
       .                       .and.UTMY(indexY(iR)).lt.(UTMY(indexY(iR2))
       .                       +0.02)) then
                                    exit
                               else if (iR2.eq.(iR-1)) then
                                    irow = irow + 1
                                    coordY(irow) = UTMY(indexY(iR))
                                    exit
                               end if
c                   end if
                        end do
                    else
                        icolm = icolm + 1
                        coordX(icolm) = UTMX(indexX(iR))
                        irow = irow + 1
                        coordY(irow) = UTMY(indexY(iR))
                    end if
```

```
  100 continue
      return
      end
cSE== end subroutine xyTab ==========================================

c==== subroutine SortTI2 ============================================
c
c Sort input vector array of n integer data, in ascending order, by a
c tag insertion sort.  No sorted array is actually output by subroutine
c SortTI; result is an array of subscripts of the data in the unsorted
c array, in the order in which they would appear in the sorted array.
c
c (Explicitly, this is the meaning of the output array tagLst:
c
c    INDEXES OF ARRAY IND <==> indexes of a hypothetical, sorted
c                              version of input array arr
c
c    ELEMENTS OF ARRAY IND contain the indexes of elements in the
c                          hypothetical, sorted version of input
c                          array arr IN THE ORIGINAL, UNSORTED ARRAY ARR
c
c Put one more way: tagLst is sorted tag list for original array arr.
c
c Put one last way: the value of the ith element of the hypothetical,
c                   sorted version of array arr is arr (tagLst (i))
c
c Method is best for small and almost sorted arrays.  Worst case is
c for inversely sorted arrays.
c
c Subroutine written by Paulus Irpan, SAI, 840830.)
c
c Subroutine SortTI is called by subroutine SetUp.
c
c-----------------------------------------------------------------------
c
      subroutine SortTI2 (n, arr, tagLst)
c
c-----------------------------------------------------------------------
c
c Declarations, etc.
c
c     Description of passed parameters.
c
c     integer
c    .    n,               ! # elements in array to be tag sorted
c    .    arr(n),          ! the array to be tag sorted
c    .    tagLst(n)        ! sorted tag list of the original array
c
      integer
     .    n, tagLst(n)

      real arr(n)
c
c     Description of local variables.
c
c     integer
c    .    i,               ! index of current element in unsorted
c                          ! array, which is to be inserted in
c                          ! tagLst; also, a do-loop variable
c    .    j,               ! variable index used to put i in its
```

```
c                           ! proper position in tagLst
c     .     value             ! value of current element of unsorted
c                           ! array
c
      integer
     .     i, j

      real value
c
c-----------------------------------------------------------------------
c
c Sort by tag insertion method.
c
c     Start with 1 element presumed sorted (n_th data element).
c
      tagLst(n) = n
c
c     Then insertion sort the rest, changing the indexes only.
c
      do 100 i = n-1, 1, -1
                  if (arr(i) .le. arr(tagLst(i+1))) then
                      tagLst(i) = i
                  else
                      value = arr(i)
                      j = i + 1
  110         tagLst(j-1) = tagLst(j)
                      j = j + 1
                      if (j.le.n .and. arr(tagLst(j)).lt.value)
     .                goto 110
                      tagLst(j-1) = i
                  endif
  100 continue
c
      return
      end
c
c     end subroutine SortTI
c


c==== subroutine Warng ================================================
c
c Notify user of unusual situations which may cause problems but do not
c warrant termination of processing.
c
c Messages are output to the print fileunit and terminal.  Four numeric
c parameters, 2 integer and 2 real, may be passed for display and
c clarification purposes.
c
c Subroutine Warng is called by subroutine SetUp, whenever a nonfatal
c abnormality meriting user attention arises.
c
c-----------------------------------------------------------------------
c
      subroutine Warng (fu, wrCode, iParm1, iParm2, rParm1, rParm2)
c
c-----------------------------------------------------------------------
c
c Declarations, etc.
c
```

```
c     Description of passed parameters.
c
c     integer
c    .    fu,              ! message target fileunit
c    .    wrCode,          ! the warning code
c    .    iParm1,          ! first general purpose integer param
c    .    iParm2           ! second general purpose integer param
c     real
c    .    rParm1,          ! first general purpose real param
c    .    rParm2           ! second general purpose real param
c
      integer
     .    fu, wrCode, iParm1, iParm2
      real
     .    rParm1, rParm2
c
c-----------------------------------------------------------------------
c
c Branch to correct warning message.
c
      goto ( 10, 20, 30, 40, 50 ) wrCode
c
c     If warning code does not branch to any of the above line labels,
c     signal an error.
c
      call Error (fu, 35, wrCode, 0, 0., 0.)
c
c-----------------------------------------------------------------------
c
c Text of all warning messages.
c
c     ----------------------------------------------------------------
c
c     >> Warning 1: probOn for iSrc (i), iGrp (i), = 0.
c
   10 continue
c
c     Warning 1 screen output.
c
      print '(/A, F6.4, A, A, /A, I2, A, I2, A, /A, A, /A, A, /A)',
     .    ' Warning: Toxic release probability probOn (=', rParm1, ') ',
     .    'for TOXX',
     .    '          source #', iParm1,' (which is a member of group #',
     .    iParm2, ') is',
     .    '          identically zero.  This source will never become ',
     .    'activated,',
     .    '          and will therefore never emit.  This warning appl',
     .    'ies to all',
     .    '          sources in this group.'
c
c     Warning 1 file output.
c
      write (fu, '(/A, F6.4, A, A, /A, I2, A, I2, A, /A, A, /A, A,/A)')
     .   ' Warning: Toxic release probability probOn (=', rParm1, ') f',
     .    'or TOXX',
     .    '          source #', iParm1, ' (which is a member of group #',
     .    iParm2,') is',
     .    '          identically zero.  This source will never become a',
     .    'ctivated,',
     .    '          and will therefore never emit.  This warning appli',
```

```
     .    'ies to all',
     .    '          sources in this group.'
c
c     Warning 1 closing.
c
      write (fu, '(/A, I2, A, A)')
     .   ' TOXX run continuing: Group #',iParm2,' has 0 probability',
     .    ' of emission.'
      print '(/A, I2, A, A)',
     .   ' TOXX run continuing: Group #',iParm2,' has 0 probability',
     .    ' of emission.'
      write (fu, '(/A)') '    '
      print '(/A)', '    '
      return
c
c     ------------------------------------------------------------------
c
c     >> Warning 2: timeOn for iSrc (i), iGrp (i), = 0.
c
   20 continue
c
c     Warning 2 screen output.
c
      print '(/A, I2, A, /A, I2, A, I2, A, /A, A, /A, A, /A)',
     .    ' Warning: Toxic release duration timeOn (=', int (rParm1),
     .    ') for TOXX',
     .    '          source #', iParm1,' (which is a member of group #',
     .    iParm2, ') is',
     .    '          identically zero.  This source will never run onc',
     .    'e',
     .    '          activated, and will therefore never emit. This wa',
     .    'rning',
     .    '          applies to all sources in this group.'
c
c     Warning 2 file output.
c
      write (fu, '(/A, I2, A, /A, I2, A, I2, A, /A, A, /A, A, /A)')
     .   ' Warning: Toxic release duration timeOn (=', int (rParm1),
     .    ') for TOXX',
     .    '          source #', iParm1, ' (which is a member of group #',
     .    iParm2, ') is',
     .    '          identically zero.  This source will never run once',
     .    ' activated,',
     .    '          and will therefore never emit. This warning applie',
     .    's to all',
     .    '          sources in this group.'
c
c     Warning 2 closing.
c
      write (fu, '(/A, I2, A, A)')
     .   ' TOXX run continuing: Group #',iParm2,' has 0 duration of',
     .    ' emission.'
      print '(/A, I2, A, A)',
     .   ' TOXX run continuing: Group #',iParm2,' has 0 duration of',
     .    ' emission.'
      write (fu, '(/A)') '    '
      print '(/A)', '    '
      return
c
c     ------------------------------------------------------------------
```

```
c
c     >> Warning 3: rate for iSrc (i), pollutant j, = 0.
c
   30 continue
c
c     Warning 3 screen output.
c
      print '(/A, F6.4, A, I2, A, /A, I2, A, I1, A, /A, A, /A)',
     .  ' Warning: Emissions rate (=', rParm2, ') for TOXX source #',
     .    iParm1, ' (which is',
     .    '          a member of group #', iParm2, '), pollutant #',
     .    int(rParm1), ' is identically zero.',
     .    '          This source will never emit this pollutant. This',
     .    ' warning',
     .    '          applies to all sources in this group.'
c
c     Warning 3 file output.
c
      write (fu, '(/A, F6.4, A, I2, A, /A, I2, A, I1, A, /A, A, /A)')
     .  ' Warning: Emissions rate (=', rParm2, ') for TOXX source #',
     .    iParm1, ' (which is',
     .    '          a member of group #', iParm2, '), pollutant #',
     .    int(rParm1), ' is identically zero.',
     .    '          This source will never emit this pollutant. This',
     .    ' warning',
     .    '          applies to all sources in this group.'
c
c     Warning 3 closing.
c
      write (fu, '(/A, I2, A, A, I1, A)')
     .  ' TOXX run continuing: Group #', iParm2,' has 0 emissions r',
     .    'ate for pollutant #', int (rParm1), '.'
      print '(/A, I2, A, A, I1, A)',
     .   ' TOXX run continuing: Group #', iParm2,' has 0 emissions ',
     .    'rate for pollutant #', int (rParm1), '.'
      write (fu, '(/A)') '    '
      print '(/A)', '    '
      return
c
c     ------------------------------------------------------------------
c
c     >> Warning 4: mod (timeOn for iSrc (i) in iGrp (i) , iAvPer) ^= 0.
c
   40 continue
c
c     Warning 4 screen output.
c
      print '(/A, I4, A, A, I2, /A, I2, A, A, /A, I2, A, A, /A, A, /A,
     .    A, /A)',
     .    ' Warning: Toxic release duration (=', int(rParm1),' hrs) fo',
     .    'r source #', iParm1,
     .    '          in group #', iParm2, ' is not evenly',
     .    ' divided by the',
     .    '          averaging period (=', int (rParm2), ' hrs). Rele',
     .    'ase duration will',
     .    '          be truncated to the maximum # averaging periods l',
     .    'ess than duration,',
     .    '          but will be set to at least 1 averaging period. T',
     .    'his warning',
     .    '          applies to all sources in this group.'
```

```
c
c     Warning 4 file output.
c
      write (fu,'(/A, I4, A, A, I2, /A, I2, A, A, /A, I2, A, A, /A, A,
     .       /A, A, /A)')
     .   ' Warning: Toxic release duration (=', int(rParm1),' hrs) for',
     .    ' source #', iParm1,
     .    '         in group #', iParm2, ' is not evenly ',
     .    'divided by the',
     .    '         averaging period (=', int (rParm2), ' hrs).  Relea',
     .    'se duration will',
     .    '         be truncated to the maximum # averaging periods le',
     .    'ss than duration,',
     .    '         but will be set to at least 1 averaging period. Th',
     .    'is warning',
     .    '         applies to all sources in this group.'
c
c     Warning 4 closing.
c
      write (fu, '(/A, A, I2, A)')
     . ' TOXX run continuing: iAvPer does not divide timeOn for gro',
     .    'up #', iParm2, '.'
      print '(/A, A, I2, A)',
     .  ' TOXX run continuing: iAvPer does not divide timeOn for gr',
     .    'oup #', iParm2, '.'
      write (fu, '(/A)') '    '
      print '(/A)', '    '
      return
c
c     ------------------------------------------------------------------
c
c     >> Warning 5: effHET (j) = 0.
c
   50 continue
c
c     Warning 5 screen output.
c
      print '(/A, I1, A, F8.4, A, /A, F8.4, A, A, /A, I2, A, A, /A, A,
     .       /A, A)',
     .    ' Warning: Health effect threshold #', iParm1, ' (=', rParm1,
     .    ' ug/m3) and the',
     .    '          corresponding background (=', rParm2, ' ug/m3) ar',
     .    'e equal or nearly so,',
     .    '          causing effective threshold #', iParm1, ' to be z',
     .    'ero.  This will',
     .    '          allow every chi/q value to pass screening, making',
     .    ' the program run',
     .    '          a long time.  Consider termination and use of a l',
     .    'arger threshold.'
c
c     Warning 5 file output.
c
      write (fu, '(/A, I1, A, F8.4, A, /A, F8.4, A, A, /A, I1, A, A, /A,
     .            A, /A, A)')
     .   ' Warning: Health effect threshold #', iParm1, ' (=', rParm1,
     .    ' ug/m3) and the',
     .    '          corresponding background (=', rParm2, ' ug/m3) ar',
     .    'e equal or nearly so,',
     .    '          causing effective threshold #', iParm1, ' to be z',
     .    'ero.  This will',
```

```
     .    '          allow every chi/q value to pass screening, making',
     .    ' the program run',
     .    '          a long time.  Consider termination and use of a l',
     .    'arger threshold.'

c
c     Warning 5 closing.
c
      write (fu, '(/A, I1, A, A)')
     .  ' TOXX run continuing: Effective threshold #', iParm1, ' is ',
     .    'zero.'
      print '(/A, I1, A, A)',
     .   ' TOXX run continuing: Effective threshold #', iParm1,' is ',
     .    'zero.'
      write (fu, '(/A)') '    '
      print '(/A)', '    '
      return
c
      end
c
c     end subroutine Warng
c

c==== subroutine Error ==========================================
c
c Notify user of fatal abnormalities which demand termination of
c processing.
c
c Messages are output to the print fileunit and terminal.  Four numeric
c parameters, 2 integer and 2 real, are passed for display and
c clarification purposes.  Not all are used for every error message.
c
c Subroutine Error is called by program TOXX and subroutines GetInp,
c ChkDat, SetUp, and Warng, whenever a condition that forces termination
c of processing occurs.
c
c-----------------------------------------------------------------------
c
      subroutine Error (fu, erCode, iParm1, iParm2, rParm1, rParm2)
c
c-----------------------------------------------------------------------
c
c Declarations, etc.
c
c     Description of passed parameters.
c
c     integer
c    .    fu,              ! message target fileunit
c    .    erCode,          ! the error code
c    .    iParm1,          ! first general purpose integer param
c    .    iParm2           ! second general purpose integer param
c     real
c    .    rParm1,          ! first general purpose real param
c    .    rParm2           ! second general purpose real param
c
      integer
     .    fu, erCode, iParm1, iParm2
      real
     .    rParm1, rParm2
c
```

```
c-----------------------------------------------------------------------
c
c Branch to correct error message.
c
      goto ( 10, 20, 30, 40, 50, 60, 70, 80, 90, 100,
     .       110, 120, 130, 140, 150, 160, 170, 180, 190, 200,
     .       210, 220, 230, 240, 250, 260, 270, 280, 290, 300,
     .       310, 320, 330, 340, 350) erCode
c
c     If error code does not branch to any of the above line labels,
c     terminate with an error message.
c
c-----------------------------------------------------------------------
c
c     >> Error ?: Invalid error code.
c
c     Error ? screen output.
c
      print '(/A, I2, A, A, /A)',
     .    ' Error: The value of variable erCode (= ',erCode,') is not ',
     .    'valid.',
     .    '        No error condition matches this value.'
      print '(/A, A, /A, A, /A, A, /A, A, /A, A, /A)',
     .    ' Solution: This error is highly unusual, and probably refle',
     .    'cts a',
     .    ' typing error in the TOXX source code.  Try to trace to th',
     .    'e point',
     .    ' in the code which called the Error subroutine, and check t',
     .    'he value',
     .    ' of the second parameter passed to the Error subroutine. It',
     .    ' should',
     .    ' match the number of the most appropriate error message, gi',
     .    'ven the',
     .    ' context of the calling statement, in the subroutine Error.'
c
c     Error ? file output.
c
      write (fu, '(/A, I2, A, A, /A)')
     .    'Error: The value of variable erCode (= ',erCode,') is not ',
     .    'valid.',
     .    '       No error condition matches this value.'
      write (fu, '(/A, A, /A, A, /A, A, /A, A, /A, A, /A)')
     .    'Solution: This error is highly unusual, and probably reflec',
     .    'ts a',
     .    'typing error in the TOXX source code.  Try to trace to the',
     .    ' point',
     .    'in the code which called the Error subroutine, and check th',
     .    'e value',
     .    'of the second parameter passed to the Error subroutine. It ',
     .    'should',
     .    'match the number of the most appropriate error message, giv',
     .    'en the',
     .    'context of the calling statement, in the subroutine Error.'
c
c     Error ? closing.
c
      write (fu, '(/A)') 'TOXX run aborted: Invalid erCode.'
      print '(A)', '    '
      stop 'TOXX run aborted: Invalid erCode.'
c
```

```
c-----------------------------------------------------------------------
c
c Text of all error messages.
c
c     -----------------------------------------------------------------
c
c     >> Error 1: MPER > MCPER.
c
   10 continue
c
c     Error 1 screen output.
c
      print '(/A, I5, A, A, /A, I4, A)',
     .    ' Error: Max number of periods MPER (=', iParm1, ') exceeds ',
     .    'absolute',
     .    '        ceiling value MCPER (=',iParm2, ').'
      print '(/A, A, /A, A, /A, A)',
     .    ' Solution: Decrease MPER in insert file or alter data id co',
     .    'ding in',
     .    ' preprocessor and TOXX program to allow larger MCPER.  (Wa',
     .    'rning:',
     .    ' Latter solution involves code modification and is not reco',
     .    'mmended.)'
c
c     Error 1 file output.
c
      write (fu, '(/A, I5, A, A, /A, I4, A)')
     .    'Error: Max number of periods MPER (=', iParm1, ') exceeds ',
     .    'absolute',
     .    '       ceiling value MCPER (=',iParm2, ').'
      write (fu, '(/A, A, /A, A, /A, A)')
     .    'Solution: Decrease MPER in insert file or alter data id cod',
     .    'ing in',
     .    'preprocessor and TOXX program to allow larger MCPER.  (War',
     .    'ning:',
     .    'Latter solution involves code modification and is not recom',
     .    'mended.)'
c
c     Error 1 closing.
c
      write (fu, '(/A)') 'TOXX run aborted: MPER exceeds MCPER.'
      print '(A)', '    '
      stop 'TOXX run aborted: MPER exceeds MCPER.'
c
c-----------------------------------------------------------------------
c
c     >> Error 2: MSRC > MCSRC.
c
   20 continue
c
c     Error 2 screen output.
c
      print '(/A, I5, A, A, /A, I4, A)',
     .    ' Error: Max number of sources MSRC (=', iParm1, ') exceeds ',
     .    'absolute',
     .    '        ceiling value MCSRC (=',iParm2, ').'
      print '(/A, A, /A, A, /A, A)',
     .    ' Solution: Decrease MSRC in insert file or alter data id co',
     .    'ding in',
     .    ' preprocessor and TOXX program to allow larger MCSRC.  (Wa',
```

```
     .    'rning:',
     .    ' Latter solution involves code modification and is not reco',
     .    'mmended.)'
c
c     Error 2 file output.
c
      write (fu, '(/A, I5, A, A, /A, I4, A)')
     .    'Error: Max number of sources MSRC (=', iParm1, ') exceeds ',
     .    'ceiling value',
     .    '       MCSRC (=',iParm2, ').'
      write (fu, '(/A, A, /A, A, /A, A)')
     .    'Solution: Decrease MSRC in insert file or alter data id cod',
     .    'ing in',
     .    'preprocessor and TOXX program to allow larger MCSRC.  (Wa',
     .    'rning:',
     .    'Latter solution involves code modification and is not reco',
     .    'mmended.)'
c
c     Error 2 closing.
c
      write (fu, '(/A)') 'TOXX run aborted: MSRC exceeds MCSRC.'
      print '(A)', '    '
      stop 'TOXX run aborted: MSRC exceeds MCSRC.'
c
c-----------------------------------------------------------------------
c
c     >> Error 3: MREC > MCREC.
c
   30 continue
c
c     Error 3 screen output.
c
      print '(/A, I5, A, A, /A, I4, A)',
     .    ' Error: Max number of receptors MREC (=', iParm1, ') exceed',
     .    's absolute',
     .    '        ceiling value MCREC (=',iParm2, ').'
      print '(/A, A, /A, A, /A, A)',
     .    ' Solution: Decrease MREC in insert file or alter data id co',
     .    'ding in',
     .    ' preprocessor and TOXX program to allow larger MCREC.  (Wa',
     .    'rning:',
     .    ' Latter solution involves code modification and is not reco',
     .    'mmended.)'
c
c     Error 3 file output.
c
      write (fu, '(/A, I5, A, A, /A, I4, A)')
     .    'Error: Max number of recpetors MREC (=', iParm1, ') exceeds',
     .    ' ceiling value',
     .    '       MCREC (=',iParm2, ').'
      write (fu, '(/A, A, /A, A, /A, A)')
     .    'Solution: Decrease MREC in insert file or alter data id cod',
     .    'ing in',
     .    'preprocessor and TOXX program to allow larger MCREC.  (Wa',
     .    'rning:',
     .    'Latter solution involves code modification and is not reco',
     .    'mmended.)'
c
c     Error 3 closing.
c
```

```
      write (fu, '(/A)') 'TOXX run aborted: MREC exceeds MCREC.'
      print '(A)', '    '
      stop 'TOXX run aborted: MREC exceeds MCREC.'
c
c-----------------------------------------------------------------------
c
c     >> Error 4: nHET > MHET.
c
   40 continue
c
c     Error 4 screen output.
c
      print '(/A, I4, A, A, /A, I4, A)',
     .    ' Error: Number of thresholds nHET (=', iParm1, ') exceeds m',
     .    'ax value',
     .    '        MHET (=',iParm2, ').'
      print '(/A, A, /A)',
     .    ' Solution: Decrease nHET and # threshold data in user input',
     .    ' file',
     .    ' or increase value of parameter MHET in insert file.'
c
c     Error 4 file output.
c
      write (fu, '(/A, I4, A, A, /A, I4, A)')
     .    'Error: Number of thresholds nHET (=', iParm1, ') exceeds ma',
     .    'x value',
     .    '       MHET (=',iParm2, ').'
      write (fu, '(/A, A, /A)')
     .    'Solution: Decrease nHET and # threshold data in user input ',
     .    'file',
     .    'or increase value of parameter MHET in insert file.'
c
c     Error 4 closing.
c
      write (fu, '(/A)') 'TOXX run aborted: nHET exceeds MHET.'
      print '(A)', '    '
      stop 'TOXX run aborted: nHET exceeds MHET.'
c
c-----------------------------------------------------------------------
c
c     >> Error 5: nPol > MHET.
c
   50 continue
c
c     Error 5 screen output.
c
      print '(/A, I4, A, A, /A, I4, A)',
     .    ' Error: Number of pollutants nPol (=', iParm1, ') exceeds m',
     .    'ax value',
     .    '        MHET (=',iParm2, ').'
      print '(/A, A, /A)',
     .    ' Solution: Decrease nPol and # pollutant data in user input',
     .    ' file',
     .    ' or increase value of parameter MHET in insert file.'
c
c     Error 5 file output.
c
      write (fu, '(/A, I4, A, A, /A, I4, A)')
     .    'Error: Number of pollutants nPol (=', iParm1, ') exceeds ma',
     .    'x value',
```

```
     .    '       MHET (=',iParm2, ').'
      write (fu, '(/A, A, /A)')
     .    'Solution: Decrease nPol and # pollutant data in user input ',
     .    'file',
     .    'or increase value of parameter MHET in insert file.'
c
c     Error 5 closing.
c
      write (fu, '(/A)') 'TOXX run aborted: nPol exceeds MHET.'
      print '(A)', '    '
      stop 'TOXX run aborted: nPol exceeds MHET.'
c
c-----------------------------------------------------------------------
c
c     >> Error 6: nPol > 1 but nPol ^= nHET.
c
   60 continue
c
c     Error 6 screen output.
c
      print '(/A, I4, A, A, /A, I4, A)',
     .    ' Error: Number of pollutants nPol (=', iParm1, ') is greate',
     .    'r than 1',
     .    '       but does not equal nHET (=',iParm2, ').'
      print '(/A, A, /A)',
     .    ' Solution: If multiple pollutants are in fact desired, make',
     .    ' certain',
     .    ' that there is exactly one threshold for each pollutant.'
c
c     Error 6 file output.
c
      write (fu, '(/A, I4, A, A, /A, I4, A)')
     .    'Error: Number of pollutants nPol (=', iParm1, ') is greater',
     .    ' than 1',
     .    '       but does not equal nHET (=',iParm2, ').'
      write (fu, '(/A, A, /A)')
     .    'Solution: If multiple pollutants are in fact desired, make ',
     .    'certain',
     .    'that there is exactly one threshold for each pollutant.'
c
c     Error 6 closing.
c
      write (fu, '(/A)')
     .  'TOXX run aborted: nPol greater than 1 but not equal to nHET.'
      print '(A)', '    '
      stop
     .  'TOXX run aborted: nPol greater than 1 but not equal to nHET.'
c
c-----------------------------------------------------------------------
c
c     >> Error 7: nSim > MSIM.
c
   70 continue
c
c     Error 7 screen output.
c
      print '(/A, I5, A, /A, I4, A)',
     .    ' Error: Number of Monte Carlo simulations nSim (=', iParm1,
     .    ') exceeds max',
     .    '       value MSIM (=',iParm2, ').'
```

```
      print '(/A, A, /A)',
     .    ' Solution: Decrease nSim in user input file or increase val',
     .    'ue of',
     .    ' parameter MSIM in insert file.'
c
c     Error 7 file output.
c
      write (fu, '(/A, I5, A, /A, I4, A)')
     .    'Error: Number of Monte Carlo simulations nSim (=', iParm1,
     .    ') exceeds max',
     .    '       value MSIM (=',iParm2, ').'
      write (fu, '(/A, A, /A)')
     .    'Solution: Decrease nSim in user input file or increase valu',
     .    'e of',
     .    'parameter MSIM in insert file.'
c
c     Error 7 closing.
c
      write (fu, '(/A)') 'TOXX run aborted: nSim exceeds MSIM.'
      print '(A)', '    '
      stop 'TOXX run aborted: nSim exceeds MSIM.'
c
c-----------------------------------------------------------------------
c
c     >> Error 8: nSrc > MSRC.
c
   80 continue
c
c     Error 8 screen output.
c
      print '(/A, I4, A, A, /A, I4, A)',
     .   ' Error: Number of TOXX sources nSrc (=', iParm1, ') exceed',
     .    's max value',
     .    '        MSRC (=',iParm2, ').'
      print '(/A, A, /A)',
     .    ' Solution: Decrease nSrc and # source data lines in user in',
     .    'put file',
     .    ' or increase parameter MSRC in insert file.'
c
c     Error 8 file output.
c
      write (fu, '(/A, I4, A, A, /A, I4, A)')
     .   'Error: Number of TOXX sources nSrc (=', iParm1, ') exceeds',
     .    ' max value',
     .    '       MSRC (=',iParm2, ').'
      write (fu, '(/A, A, /A)')
     .    'Solution: Decrease nSrc and # source data lines in user inp',
     .    'ut file',
     .    'or increase parameter MSRC in insert file.'
c
c     Error 8 closing.
c
      write (fu, '(/A)') 'TOXX run aborted: nSrc exceeds MSRC.'
      print '(A)', '    '
      stop 'TOXX run aborted: nSrc exceeds MSRC.'
c
c-----------------------------------------------------------------------
c
c     >> Error 9: iAvPer ^= 1, 3, or 24.
c
```

```
   90 continue
c
c     Error 9 screen output.
c
      print '(/A, I4, A, /A)',
     .    ' Error: Duration in hours of averaging period iAvPer (=',
     .    iParm1,
     .    ') is not one',
     .    '        of the currently allowed values 1, 3, and 24.'
      print '(/A, A, /A, A, /A, A, /A)',
     .    ' Solution: Set iAvPer to one of the above values, or add a ',
     .    'new value',
     .    ' to those allowed.  (Warning: It is important that both the',
     .    ' preprocessor',
     .    ' and TOXX use the same averaging period, so that nPer and ',
     .    ' nP are',
     .    ' identical for every DMR file.)'
c
c     Error 9 file output.
c
      write (fu, '(/A, I4, A, /A)')
     .    'Error: Duration in hours of averaging period iAvPer (=',
     .    iParm1,
     .    ') is not one',
     .    '        of the currently allowed values 1, 3, and 24.'
      write (fu, '(/A, A, /A, A, /A, A, /A)')
     .    'Solution: Set iAvPer to one of the above values, or add a ',
     .    'new value',
     .    'to those allowed.  (Warning: It is important that both the',
     .    ' preprocessor',
     .    'and TOXX use the same averaging period, so that nPer and ',
     .    'nP are',
     .    'identical for every DMR file.)'
c
c     Error 9 closing.
c
      write (fu, '(/A)') 'TOXX run aborted: Invalid iAvPer.'
      print '(A)', '    '
      stop 'TOXX run aborted: Invalid iAvPer.'
c
c-----------------------------------------------------------------------
c
c     >> Error 10: lAdd ^= 0 or 1.
c
  100 continue
c
c     Error 10 screen output.
c
      print '(/A, I4, A, A, /A)',
     .    ' Error: Switch variable lAdd (=', iParm1, ') does not equal',
     .    ' either of the ',
     .    '        allowed values 0 or 1.'
      print '(/A, A, /A)',
     .    ' Solution: Set lAdd to one of the allowed values in user in',
     .    'put file.',
     .    ' (0=exex by indiv. threshold, 1=exex by additive model)'
c
c     Error 10 file output.
c
      write (fu, '(/A, I4, A, A, /A)')
```

```
     .    'Error: Switch variable lAdd (=', iParm1, ') does not equal ',
     .    'either of the ',
     .    '       allowed values 0 or 1.'
      write (fu, '(/A, A, /A)')
     .    'Solution: Set lAdd to one of the allowed values in user in',
     .    'put file.',
     .    '(0=exex by indiv. threshold, 1=exex by additive model)'
c
c     Error 10 closing.
c
      write (fu, '(/A)') 'TOXX run aborted: Invalid lAdd.'
      print '(A)', '    '
      stop 'TOXX run aborted: Invalid lAdd.'
c
c-----------------------------------------------------------------------
c
c     >> Error 11: lAdd = 1 and nPol = 1.
c
  110 continue
c
c     Error 11 screen output.
c
      print '(/A, I4, A, /A, I4, A)',
     .    ' Error: Additive calculation requested (lAdd = ', iParm1,
     .    ') with only',
     .    '        one pollutant (nPol = ', iParm2, ').'
      print '(/A, A, /A)',
     .    ' Solution: Assuming nPol=1 is desired, set lAdd to 0 in use',
     .    'r input file.',
     .    ' (0=exex by indiv. threshold, 1=exex by additive model)'
c
c     Error 11 file output.
c
      write (fu, '(/A, I4, A, /A, I4, A)')
     .    'Error: Additive calculation requested (lAdd = ', iParm1,
     .    ') with only',
     .    '       one pollutant (nPol = ', iParm2, ').'
      write (fu, '(/A, A, /A)')
     .    'Solution: Assuming nPol=1 is desired, set lAdd to 0 in use',
     .    'r input file.',
     .    '(0=exex by indiv. threshold, 1=exex by additive model)'
c
c     Error 11 closing.
c
      write (fu, '(/A)')
     .    'TOXX run aborted: lAdd equals 1 with nPol also equal to 1.'
      print '(A)', '    '
      stop 'TOXX run aborted:lAdd equals 1 with nPol also equal to 1.'
c
c-----------------------------------------------------------------------
c
c     >> Error 12: lTab ^= 0, 1, 2, or 9.
c
  120 continue
c
c     Error 12 screen output.
c
      print '(/A, I4, A, A, /A)',
     .    ' Error: Switch variable lTab (=', iParm1, ') does not equal',
     .    ' any of the ',
```

```
     .    '        allowed values 0, 1, 2, or 9.'
      print '(/A, A, /A, A, /A)',
     .    ' Solution: Set lTab to one of the allowed values in user in',
     .    'put file.',
     .    ' (0=no exex output by receptor, 1=output in polar table, 2=',
     .    'output in',
     .    ' Cartesian table, 9=output in form specified by DMR files).'
c
c     Error 12 file output.
c
      write (fu, '(/A, I4, A, A, /A)')
     .    'Error: Switch variable lTab (=', iParm1, ') does not equal',
     .    ' any of the ',
     .    '        allowed values 0, 1, 2, or 9.'
      write (fu, '(/A, A, /A, A, /A)')
     .    'Solution: Set lTab to one of the allowed values in user in',
     .    'put file.',
     .    '(0=no exex output by receptor, 1=output in polar table, 2=',
     .    'output in',
     .    'Cartesian table, 9=output in form specified by DMR files).'
c
c     Error 12 closing.
c
      write (fu, '(/A)') 'TOXX run aborted: Invalid lTab.'
      print '(A)', '    '
      stop 'TOXX run aborted: Invalid lTab.'

c
c-----------------------------------------------------------------------
c
c     >> Error 13: lYrly ^= 0 or 1.
c
  130 continue
c
c     Error 13 screen output.
c
      print '(/A, I2, A, A, /A)',
     .    ' Error: Switch variable lYrly (=', iParm1, ') does not equa',
     .    'l either of the ',
     .    '        allowed values 0 or 1.'
      print '(/A, A, /A, A, /A)',
     .    ' Solution: Set lYrly to one of the allowed values in user i',
     .    'nput file.',
     .    ' (0=output by individual years and all years combined, 1=ou',
     .    'tput only',
     .    ' by all years combined).'
c
c     Error 13 file output.
c
      write (fu, '(/A, I2, A, A, /A)')
     .    'Error: Switch variable lYrly (=', iParm1, ') does not equal',
     .    ' either of the ',
     .    '        allowed values 0 or 1.'
      write (fu, '(/A, A, /A, A, /A)')
     .    'Solution: Set lYrly to one of the allowed values in user in',
     .    'put file.',
     .    '(0=output by individual years and all years combined, 1=out',
     .    'put only',
     .    'by all years combined).'
c
```

```
c     Error 13 closing.
c
      write (fu, '(/A)') 'TOXX run aborted: Invalid lYrly.'
      print '(A)', '    '
      stop 'TOXX run aborted: Invalid lYrly.'
c
c-----------------------------------------------------------------------
c
c     >> Error 14: nYear > MYEAR.
c
  140 continue
c
c     Error 14 screen output.
c
      print '(/A, I2, A, /A, I1, A)',
     .    ' Error: Number of meteorological years nYear (=', iParm1,
     .    ') exceeds max',
     .    '        value MYEAR (=',iParm2, ').'
      print '(/A, A, /A)',
     .    ' Solution: Decrease nYear and # year data lines in user inp',
     .    'ut file,',
     .    ' or increase value of parameter MYEAR in insert file.'
c
c     Error 14 file output.
c
      write (fu, '(/A, I2, A, A, /A, I1, A)')
     .    'Error: Number of meteorological years nYear (=', iParm1, ')',
     .    ' exceeds max',
     .    '       value MYEAR (=',iParm2, ').'
      write (fu, '(/A, A, /A)')
     .    'Solution: Decrease nYear and # year data lines in user inpu',
     .    't file,',
     .    'or increase value of parameter MYEAR in insert file.'
c
c     Error 14 closing.
c
      write (fu, '(/A)') 'TOXX run aborted: nYear exceeds MYEAR.'
      print '(A)', '    '
      stop 'TOXX run aborted: nYear exceeds MYEAR.'
c
c-----------------------------------------------------------------------
c
c     >> Error 15: nS in DMR file for year iYr < nSrc.
c
  150 continue
c
c     Error 15 screen output.
c
      print '(/A, I2, A, A, I4, /A, I2, A, A)',
     .    ' Error: Number of sources nS (=', iParm2, ') from DMR file ',
     .    'for year ', iParm1,
     .    '        is less than number of sources nSrc (=', int(rParm1),
     .    ') from user input file.'
      print '(/A, A, /A, A, /A)',
     .    ' Solution: Decrease nSrc in user input file or use only DMR',
     .    ' files with',
     .    ' nS greater than or equal to nSrc.  TOXX may not use sourc',
     .    'es not present',
     .    ' in DMR files.'
c
```

```
c     Error 15 file output.
c
      write (fu, '(/A, I2, A, A, I4, /A, I2, A, A)')
     .    'Error: Number of sources nS (=', iParm2, ') from DMR file f',
     .    'or year ', iParm1,
     .    '       is less than number of sources nSrc (=', int (rParm1),
     .    ') from user input file.'
      write (fu, '(/A, A, /A, A, /A)')
     .    'Solution: Decrease nSrc in user input file or use only DMR ',
     .    'files with',
     .    'nS greater than or equal to nSrc.  TOXX may not use source',
     .    's not present',
     .    'in DMR files.'
c
c     Error 15 closing.
c
      write (fu, '(/A)')
     .   'TOXX run aborted: nS in a DMR file is less than nSrc.'
      print '(A)', '    '
      stop 'TOXX run aborted: nS in a DMR file is less than nSrc.'
c
c-----------------------------------------------------------------------
c
c     >> Error 16: nR in DMR file for year iYr > MREC.
c
  160 continue
c
c     Error 16 screen output.
c
      print '(/A, I3, A, A, I4, /A, I3, A)',
     .    ' Error: Number of receptors nR (=', iParm2, ') from DMR fil',
     .    'e for year ', iParm1,
     .    '         exceeds max value MREC (=', int (rParm1), ').'
      print '(/A)',
     .    ' Solution: Increase MREC in insert file.'
c
c     Error 16 file output.
c
      write (fu, '(/A, I3, A, A, I4, /A, I3, A, A, /A)')
     .    'Error: Number of receptors nR (=', iParm2, ') from DMR file',
     .    ' for year ', iParm1,
     .    '       exceeds max value MREC (=', int (rParm1), ').'
      write (fu, '(/A)')
     .    'Solution: Increase MREC in insert file.'
c
c     Error 16 closing.
c
      write (fu, '(/A)')
     .    'TOXX run aborted: nR in a DMR file exceeds MREC.'
      print '(A)', '    '
      stop 'TOXX run aborted: nR in a DMR file exceeds MREC.'
c
c-----------------------------------------------------------------------
c
c     >> Error 17: nR in DMR file for year iYr ^= nR for previous year.
c
  170 continue
c
c     Error 17 screen output.
c
```

```
      print '(/A, I3, A, A, I4, /A, I3, A)',
     .    ' Error: Number of receptors nR (=', iParm2, ') from DMR fil',
     .    'e for year ', iParm1,
     .    '        differs from that for previous year (nR0 = ',
     .    int (rParm1), ').'
      print '(/A)',
     .    ' Solution: Make certain nR is the same in all DMR files.'
c
c     Error 17 file output.
c
      write (fu, '(/A, I3, A, A, I4, /A, I3, A)')
     .    'Error: Number of receptors nR (=', iParm2, ') from DMR file',
     .    ' for year ', iParm1,
     .    '       differs from that for previous year (nR0 = ',
     .    int (rParm1), ').'
      write (fu, '(/A)')
     .    'Solution: Make certain nR is the same in all DMR files.'
c
c     Error 17 closing.
c
      write (fu, '(/A)')
     .   'TOXX run aborted: nR varies among DMR files.'
      print '(A)', '    '
      stop   'TOXX run aborted: nR varies among DMR files.'
c
c-----------------------------------------------------------------------
c
c     >> Error 18: nPer for year iYr > MPER.
c
  180 continue
c
c     Error 18 screen output.
c
      print '(/A, I4, A, A, I4, /A, I4, A)',
     .    ' Error: Number of periods nPer  (=', iParm2, ') calculated ',
     .    'for year ', iParm1,
     .    '        exceeds maximum value MPER (= ', int (rParm1), ').'
      print '(/A, A, /A, A)',
     .    ' Solution: This most likely results from an illegal value o',
     .    'f iAvPer.',
     .    ' Make certain iAvPer is not less than the minimum allowed',
     .    ' value of 1.'
c
c     Error 18 file output.
c
      write (fu, '(/A, I4, A, A, I4, /A, I4, A)')
     .    'Error: Number of periods nPer  (=', iParm2, ') calculated f',
     .    'or year ', iParm1,
     .    '       exceeds maximum value MPER (= ', int (rParm1), ').'
      write (fu, '(/A, A, /A, A)')
     .    'Solution: This most likely results from an illegal value of',
     .    ' iAvPer.',
     .    'Make certain iAvPer is not less than the minimum allowed va',
     .    'lue of 1.'
c
c     Error 18 closing.
c
      write (fu, '(/A)')
     .    'TOXX run aborted: nPer from a DMR file exceeds MPER.'
      print '(A)', '    '
```

```
      stop   'TOXX run aborted: nPer from a DMR file exceeds MPER.'
c
c-----------------------------------------------------------------------
c
c     >> Error 19: nP in DMR file for year iYr ^= nPer calc frm iAvPer,
c                  iYr.
c
  190 continue
c
c     Error 19 screen output.
c
      print '(/A, I4, A, A, I4, A, /A, I4, A, A, /A, I2, A)',
     .    ' Error: Number of periods nP (=', iParm2, ') from DMR file ',
     .    'for year iYr (=', iParm1, ')',
     .    '        differs from nPer (=', int (rParm1), ') calculated ',
     .    'for same year using',
     .    '        the user-input averaging period, iAvPer (=',
     .    int (rParm2), ').'
      print '(/A, A, /A, A, /A)',
     .    ' Solution: nPer is calculated as ((24/iAvPer) * Leap(iYr)).',
     .    ' Make certain',
     .    ' that the preprocessor used the same averaging period as TO',
     .    'XXS, and that',
     .    ' nP is equal to nPer in each DMR file.'
c
c     Error 19 file output.
c
      write (fu, '(/A, I4, A, A, I4, A, /A, I4, A, A, /A, I2, A)')
     .    'Error: Number of periods nP (=', iParm2, ') from DMR file f',
     .    'or year iYr (=', iParm1, ')',
     .    '       differs from nPer (=', int (rParm1), ') calculated ',
     .    'for same year using',
     .    '       the user-input averaging period, iAvPer (=',
     .    int (rParm2), ').'
      write (fu, '(/A, A, /A, A, /A)')
     .    'Solution: nPer is calculated as ((24/iAvPer) * Leap(iYr)).',
     .    '  Make certain',
     .    'that the preprocessor used the same averaging period as TO',
     .    'XXS, and that',
     .    'nP is equal to nPer in each DMR file.'
c
c     Error 19 closing.
c
      write (fu, '(/A, A)')
     .   'TOXX run aborted: nP in a DMR file ^= nPer calculated fr',
     .    'om same.'
      print '(A)', '     '
      stop 'TOXX run aborted: nP in a DMR file ^= nPer from same.'
c
c-----------------------------------------------------------------------
c
c     >> Error 20: iYr read from DMR file ^= corres. iYear from userinp.
c
  200 continue
c
c     Error 20 screen output.
c
      print '(/A, I4, A, I1, A, /A, I1, A, I4, A, /A)',
     .    ' Error: Year iYr (=', iParm1, ') read from DMR file #',
     .    iParm2, ' differs from',
```

```
     .    '        corresponding year iYear (', iParm2, ') (=',
     .    int(rParm1), ') read from',
     .    '        user input file.'
      print '(/A, A, /A)',
     .    ' Solution: Make certain iYr in each DMR file always equals ',
     .    'the',
     .    ' corresponding entry in array iYear.'
c
c     Error 20 file output.
c
      write (fu, '(/A, I4, A, I1, A, /A, I1, A, I4, A, /A)')
     .    'Error: Year iYr (=', iParm1, ') read from DMR file #',
     .    iParm2, ' differs from ',
     .    '       corresponding year iYear (', iParm2, ') (=',
     .    int(rParm1), ') read from',
     .    '       user input file.'
      write (fu, '(/A, A, /A)')
     .    'Solution: Make certain iYr in each DMR file always equals t',
     .    'he',
     .    'corresponding entry in array iYear.'
c
c     Error 20 closing.
c
      write (fu, '(/A)')
     .    'TOXX run aborted: iYr in a DMR file ^= iYear from user.'
      print '(A)', '    '
      stop 'TOXX run aborted: iYr in a DMR file ^= iYear from user.'
c
c-----------------------------------------------------------------------
c
c     >> Error 21: Output form from DMR file requested but iTab invalid.
c
  210 continue
c
c     Error 21 screen output.
c
      print '(/A, I1, A, /A, I4, A, I4, A)',
     .    ' Error: Output format from DMR file requested (lTab = ',
     .    int (rParm1), '), but iTab',
     .    '       value (=', iParm2, ') from DMR file for year ',
     .    iParm1, ' is invalid.'
      print '(/A, A, /A)',
     .    ' Solution: Assuming lTab=9 is desired, set iTab to the same',
     .    ' legal value',
     .    ' (0, 1, or 2) in every DMR file.'
c
c     Error 21 file output.
c
      write (fu, '(/A, I1, A, /A, I4, A, I4, A)')
     .    'Error: Output format from DMR file requested (lTab = ',
     .    int (rParm1), '), but iTab',
     .    '       value (=', iParm2, ') from DMR file for year ',
     .    iParm1, 'is invalid.'
      write (fu, '(/A, A)')
     .    'Solution: Assuming lTab=9 is desired, set iTab to the same ',
     .    'legal value',
     .    '(0, 1, or 2) in every DMR file.'
c
c     Error 21 closing.
c
```

```
      write (fu, '(/A)')
     .    'TOXX run aborted: Output option from DMR file invalid.'
      print '(A)', '    '
      stop 'TOXX run aborted: Output option from DMR file invalid.'
c
c-----------------------------------------------------------------------
c
c     >> Error 22: nX in DMR file for year iYr > MRECPR.
c
c     THIS ERROR CONDITION HAS BEEN ELIMINATED.
c
  220 continue
c
c-----------------------------------------------------------------------
c
c     >> Error 23: Group number differs from previous by > 1.
c
  230 continue
c
c     Error 23 screen output.
c
      print '(/A, I2, A, I2, A, /A, I2, A, A, /A, A)',
     .    ' Error: Group number (=', iParm1, ') for source #', iParm2,
     .    ' differs from',
     .    '        previous group number (=', int(rParm1), ') by more ',
     .    'than 1 after',
     .    '        source information sorted in ascending order by gro',
     .    'up number.'
      print '(/A, A, /A, A, /A)',
     .    ' Solution: Enforce restriction that group numbers must, aft',
     .    'er sorting',
     .    ' in ascending order, form a sequence of consecutive integer',
     .    's beginning',
     .    ' with 1.'
c
c     Error 23 file output.
c
      write (fu, '(/A, I2, A, I2, A, /A, I2, A, A, /A)')
     .    'Error: Group number (=', iParm1, ') for source #', iParm2,
     .    'differs from',
     .    '       previous group number (=', int(rParm1), ') by more t',
     .    'han 1 after',
     .   'source information sorted in ascending order by group number.'
      write (fu, '(/A, A, /A, A, /A)')
     .    'Solution: Enforce restriction that group numbers must, afte',
     .    'r sorting',
     .    'in ascending order, form a sequence of consecutive integers',
     .    ' beginning',
     .    'with 1.'
c
c     Error 23 closing.
c
      write (fu, '(/A)')
     .   'TOXX run aborted: Group numbers not consecutive.'
      print '(A)', '    '
      stop 'TOXX run aborted: Group numbers not consecutive.'
c
c-----------------------------------------------------------------------
c
c     >> Error 24: nSG > MGRP.
```

```
c
  240 continue
c
c     Error 24 screen output.
c
      print '(/A, I4, A, /A, I4, A)',
     .    ' Error: Calculated number of TOXX source groups nSG (=',
     .    iParm1, ') exceeds max value',
     .    '        MGRP (=', iParm2, ').'
      print '(/A, A, /A, A, /A)',
     .    ' Solution: Decrease number of source groups used in user in',
     .    'put file',
     .    ' or increase parameter MGRP in insert file.  (Warning: Incr',
     .    'easing MGRP',
     .    ' may make TOXX program too large to fit in memory.)'
c
c     Error 24 file output.
c
      write (fu, '(/A, I4, A, /A, I4, A)')
     .    ' Error: Calculated number of TOXX source groups nSG (=',
     .    iParm1, ') exceeds max value',
     .    '       MGRP (=', iParm2, ').'
      write (fu, '(/A, A, /A, A, /A)')
     .    'Solution: Decrease number of source groups used in user inp',
     .    'ut file',
     .    'or increase parameter MGRP in insert file.  (Warning: Incre',
     .    'asing MGRP',
     .    'may make TOXX program too large to fit in memory.)'
c
c     Error 24 closing.
c
      write (fu, '(/A)') 'TOXX run aborted: nSG exceeds MGRP.'
      print '(A)', '    '
      stop   'TOXX run aborted: nSG exceeds MGRP.'
c
c-----------------------------------------------------------------------
c
c     >> Error 25: nRand > MPER.
c
c     THIS ERROR CONDITION HAS BEEN ELIMINATED.
c
  250 continue
c
c-----------------------------------------------------------------------
c
c     >> Error 26: probOn for iSrc (i), iGrp (i) > 1.0.
c
  260 continue
c
c     Error 26 screen output.
c
      print '(/A, F7.4, A, A, /A, I2, A, I2, A, /A)',
     .    ' Error: Toxic release probability probOn (=', rParm1, ') fo',
     .    'r TOXX',
     .    '        source #', iParm1, ' (which is a member of group #',
     .    iParm2,') is not    ',
     .    '    equal to 1.0.  No probability may exceed 1.0.'
      print '(/A, A, /A)',
     .    ' Solution: Make certain the release probability for every s',
     .    'ource group',
```

```
     .    ' is a fraction between 0 and 1.0, inclusive.'
c
c     Error 26 file output.
c
      write (fu, '(/A, F7.4, A, A, /A, I2, A, I2, A, /A)')
     .    'Error: Toxic release probability probOn (=', rParm1, ') for',
     .    ' TOXX',
     .    '       source #', iParm1, ' (which is a member of group #',
     .    iParm2,') is not   ',
     .    '   equal to 1.0.  No probability may exceed 1.0.'
      write (fu, '(/A, A, /A)')
     .    'Solution: Make certain the release probability for every so',
     .    'urce group',
     .    ' is a fraction between 0 and 1.0, inclusive.'
c
c     Error 26 closing.
c
      write (fu, '(/A)')
     .    'TOXX run aborted: Probability for a source exceeds 1.0.'
      print '(A)', '    '
      stop 'TOXX run aborted: Probability for a source exceeds 1.0.'
c
c-----------------------------------------------------------------------
c
c     >> Error 27: probOn (i) for iSrc (i) (member of iGrp (i)) < 0.
c
  270 continue
c
c     Error 27 screen output.
c
      print '(/A, F7.4, A, A, /A, I2, A, I2, A, /A)',
     .    ' Error: Toxic release probability probOn (=', rParm1, ') fo',
     .    'r TOXX',
     .    '       source #', iParm1, ' (which is a member of group #',
     .    iParm2,') is less',
     .    '       than zero.'
      print '(/A, A, /A)',
     .    ' Solution: Make certain the release probability for every s',
     .    'ource group',
     .    ' is positive definite.'
c
c     Error 27 file output.
c
      write (fu, '(/A, F7.4, A, A, /A, I2, A, I2, A, /A)')
     .    'Error: Toxic release probability probOn (=', rParm1, ') for',
     .    ' TOXX',
     .    '       source #', iParm1, ' (which is a member of group #',
     .    iParm2,') is less',
     .    '       than zero.'
      write (fu, '(/A, A, /A)')
     .    'Solution: Make certain the release probability for every so',
     .    'urce group',
     .    'is positive definite.'
c
c     Error 27 closing.
c
      write (fu, '(/A)')
     .    'TOXX run aborted: Probability for a source is less than 0.'
      print '(A)', '    '
      stop 'TOXX run aborted: Probability for a source is less than 0.'
```

```
c
c-----------------------------------------------------------------------
c
c     >> Error 28: timeOn (i) for iSrc (i) (member of iGrp (i)) < 0 hrs.
c
  280 continue
c
c     Error 28 screen output.
c
      print '(/A, I4, A, A, /A, I2, A, I2, A, /A)',
     .    ' Error: Toxic release duration timeOn (=', int (rParm1), ' ',
     .    'hrs) for TOXX',
     .    '        source #', iParm1, ' (which is a member of group #',
     .    iParm2,') is',
     .    '        less than zero hours.'
      print '(/A, A, /A)',
     .    ' Solution: Make certain the release duration for every sour',
     .    'ce group',
     .    ' is positive definite.'
c
c     Error 28 file output.
c
      write (fu, '(/A, I4, A, A, /A, I2, A, I2, A, /A)')
     .    'Error: Toxic release duration timeOn (=', int (rParm1), ' h',
     .    'rs) for TOXX',
     .    '       source #', iParm1, ' (which is a member of group #',
     .    iParm2,') is',
     .    '       less than zero hours.'
      write (fu, '(/A, A, /A)')
     .    'Solution: Make certain the release duration for every sourc',
     .    'e group',
     .    'is positive definite.'
c
c     Error 28 closing.
c
      write (fu, '(/A)')
     .   'TOXX run aborted: Duration for a source is less than 0 hrs.'
      print '(A)', '    '
      stop
     . 'TOXX run aborted: Duration for a source is less than 0 hrs.'
c
c-----------------------------------------------------------------------
c
c     >> Error 29: rate (i,j) (iSrc (i), iGrp (i), pollutant j) < 0 g/s.
c
  290 continue
c
c     Error 29 screen output.
c
      print '(/A, F8.3, A, I2, A, /A, I2, A, I1, A, /A)',
     .    ' Error: Emissions rate (=', rParm2, ' g/s) for source #',
     .    iParm1, ' (which is',
     .    '        a member of group #', iParm2, '), pollutant #',
     .    int(rParm1), ' is less',
     .    '        than zero grams per second.'
      print '(/A, A, /A)',
     .    ' Solution: Make certain the emissions rate for every source',
     .    ' group',
     .    ' is positive definite.'
c
```

```
c     Error 29 file output.
c
      write (fu, '(/A, F8.3, A, I2, A, /A, I2, A, I1, A, /A)')
     .    'Error: Emissions rate (=', rParm2, ') for source #', iParm1,
     .    ' (which is',
     .    '       a member of group #', iParm2, '), pollutant #',
     .    int(rParm1), ' is less',
     .    '       than zero grams per second.'
      write (fu, '(/A, A, /A)')
     .    'Solution: Make certain the emissions rate for every source ',
     .    'group',
     .    'is positive definite.'
c
c     Error 29 closing.
c
      write (fu, '(/A)')
     .'TOXX run aborted: Emis. rate for one source, pollutant < 0 g/s'
      print '(A)', '    '
      stop
     .'TOXX run aborted: Emis. rate for one source, pollutant < 0 g/s'
c
c-----------------------------------------------------------------------
c
c     >> Error 30: probOn for iSrc (i), iGrp (i) ^= probOn (i-1) in same
c                  source group
c
  300 continue
c
c     Error 30 screen output.
c
      print '(/A, F6.4, A, A, I2, A, /A, I2, A, I2, A, /A)',
     .    ' Error: Toxic release probability (=', rParm1, ')  for sour',
     .    'ce #', iParm1, ',',
     .    '       group #', iParm2, ' differs from that of source #',
     .    int (rParm2), ', the prior',
     .    '       source in the same group.'
      print '(/A, A, /A)',
     .    ' Solution: Within a single source group, all toxic release ',
     .    'probabilities',
     .    ' must be the same.  Make certain this is the case.'
c
c     Error 30 file output.
c
      write (fu, '(/A, F6.4, A, A, I2, A, /A, I2, A, I2, A, /A)')
     .    'Error: Toxic release probability (=', rParm1, ')  for sour',
     .    'ce #', iParm1, ',',
     .    '       group #', iParm2, ' differs from that of source #',
     .    int (rParm2), ', the prior',
     .    '       source in the same group.'
      write (fu, '(/A, A, /A)')
     .    'Solution: Within a single source group, all toxic release p',
     .    'robabilities',
     .    'must be the same.  Make certain this is the case.'
c
c     Error 30 closing.
c
      write (fu, '(/A)')
     .   'TOXX run aborted: probOn varies within a source group.'
      print '(A)', '    '
      stop 'TOXX run aborted: probOn varies within a source group.'
```

```
c
c-----------------------------------------------------------------------
c
c     >> Error 31: timeOn for iSrc (i), iGrp (i) ^= timeOn (i-1) in same
c                  source group
c
  310 continue
c
c     Error 31 screen output.
c
      print '(/A, I2, A, A, I2, /A, I2, A, A, /A, I2, A, A)',
     .    ' Error: Toxic release duration (=', int (rParm1), ' hrs)  f',
     .    'or source #', iParm1,
     .    '         (which is a member of group #', iParm2, ') differs ',
     .    'from that ',
     .    '         of source #', int (rParm2), ', the prior source in ',
     .    'the same group.'
      print '(/A, A, /A)',
     .    ' Solution: Within a single source group, all toxic release ',
     .    'durations',
     .    ' must be the same.  Make certain this is the case.'
c
c     Error 31 file output.
c
      write (fu, '(/A, I2, A, A, I2, /A, I2, A, A, /A, I2, A, A)')
     .    'Error: Toxic release duration (=', int (rParm1), ' hrs)  fo',
     .    'r source #', iParm1,
     .    '        (which is a member of group #', iParm2, ') differs f',
     .    'rom that ',
     .    '       of source #', int (rParm2), ', the prior source in t',
     .    'he same group.'
      write (fu, '(/A, A, /A)')
     .    'Solution: Within a single source group, all toxic release d',
     .    'urations',
     .    'must be the same.  Make certain this is the case.'
c
c     Error 31 closing.
c
      write (fu, '(/A)')
     .   'TOXX run aborted: timeOn varies within a source group.'
      print '(A)', '    '
      stop 'TOXX run aborted: timeOn varies within a source group.'
c
c-----------------------------------------------------------------------
c
c     >> Error 32: rate for iSrc (i), pollutant j ^= rate for same pol-
c                  lutant, iSrc (i-1), same source group
c
  320 continue
c
c     Error 32 screen output.
c
      print '(/A, F8.4, A, I2, A, I2, /A, I2, A, A, /A)',
     .    ' Error: Emissions rate (=', rParm1, ' g/s)  for source #',
     .    iParm1, ', pollutant #', iParm2,
     .    '         differs from that of source #', int (rParm2), ', th',
     .    'e prior',
     .    '         source in the same group.'
      print '(/A, A, /A)',
     .    ' Solution: Within a single source group, all emissions rate',
```

```
     .    's for a given',
     .    ' pollutant must be the same.  Make certain this is the case.'
c
c     Error 32 file output.
c
      write (fu, '(/A, F8.4, A, I2, A, I2, /A, I2, A, A, /A)')
     .    'Error: Emissions rate (=', rParm1, ' g/s)  for source #',
     .    iParm1, ', pollutant #', iParm2,
     .    '       differs from that of source #', int (rParm2), ', th',
     .    'e prior',
     .    '       source in the same group.'
      write (fu, '(/A, A, /A)')
     .    ' Solution: Within a single source group, all emissions rate',
     .    's for a given',
     .    ' pollutant must be the same.  Make certain this is the case.'
c
c     Error 32 closing.
c
      write (fu, '(/A)')
     .   'TOXX run aborted: Rate for a pollutant varies w/i a group.'
      print '(A)', '    '
      stop 'TOXX run aborted: Rate for a pollutant varies w/i a group.'
c
c-----------------------------------------------------------------------
c
c     >> Error 33: hET (j) < bkgd (k), resulting in effHET (j) < 0.
c
  330 continue
c
c     Error 33 screen output.
c
      print '(/A, I1, A, F8.4, A, /A, F8.4, A, A, /A)',
     .    ' Error: Health effect threshold #', iParm1, ' (=', rParm1,
     .    ' ug/m3) is less than',
     .    '        corresponding background (=', rParm2, ' ug/m3), res',
     .    'ulting in an',
     .    '        effective threshold less than zero grams per m3.'
      print '(/A, A, /A)',
     .    ' Solution: Use only thresholds that are greater than the co',
     .    'rresponding',
     .    ' backgrounds.'
c
c     Error 33 file output.
c
      write (fu, '(/A, I1, A, F8.4, A, /A, F8.4, A, A, /A) ')
     .    'Error: Health effect threshold #', iParm1, ' (=', rParm1,
     .    ' ug/m3) is less than',
     .    '       corresponding background (=', rParm2, ' ug/m3), resu',
     .    'lting in an',
     .    '       effective threshold less than zero grams per m3.'
      write (fu, '(/A, A, /A)')
     .    'Solution: Use only thresholds that are greater than the cor',
     .    'responding',
     .    'backgrounds.'
c
c     Error 33 closing.
c
      write (fu, '(/A)')
     .   'TOXX run aborted: Health effect threshold < background.'
      print '(A)', '    '
```

```
      stop 'TOXX run aborted: Health effect threshold < background.'
c
c-----------------------------------------------------------------------
c
c     >> Error 34: cutoff < pMxCut.
c
  340 continue
c
c     Error 34 screen output.
c
      print '(/A, A, /A, 1PE10.4, A, A, /A, 1PE10.4, A, A, /A)',
     .    ' Error: Cutoff for normalized concentrations calculated in ',
     .    'TOXX program',
     .    '        (=', rParm1, ' s/m3) is less than maximum cutoff va',
     .    'lue used by',
     .    '        preprocessor pMxCut (=', rParm2, ' s/m3).  This may',
     .    ' result in TOXX',
     .    '        not detecting some exceedances.'
      print '(/A, A, /A, A, /A, A, /A, A)',
     .    ' Solution: The variable cutoff is used in the TOXX screeni',
     .    'ng of DMR',
     .    ' data.  It is formulated so that only DMR data that cannot ',
     .    'possibly',
     .    ' cause exceedance are dropped.  If TOXX cutoff is less tha',
     .    'n the cutoff',
     .    ' used by the preprocessor, the preprocessor cutoff should b',
     .    'e reduced.'
c
c     Error 34 file output.
c
      write (fu, '(/A, A, /A, 1PE10.4, A, A, /A, 1PE10.4, A, A, /A)')
     .    'Error: Cutoff for normalized concentrations calculated in T',
     .    'OXXS program',
     .    '       (=', rParm1, ' s/m3) is less than maximum cutoff val',
     .    'ue used by',
     .    '       preprocessor pMxCut (=', rParm2, ' s/m3).  This may ',
     .    'result in TOXX',
     .    '       not detecting some exceedances.'
      write (fu, '(/A, A, /A, A, /A, A, /A, A)')
     .    'Solution: The variable cutoff is used in the TOXX screenin',
     .    'g of DMR',
     .    'data.  It is formulated so that only DMR data that cannot p',
     .    'possibly',
     .    'cause exceedance are dropped.  If TOXX cutoff is less than',
     .    ' the cutoff',
     .    'used by the preprocessor, the preprocessor cutoff should be',
     .    ' reduced.'
c
c     Error 34 closing.
c
      write (fu, '(/A)')
     .    'TOXX run aborted: TOXX cutoff less than pMxCut.'
      print '(A)', '    '
      stop  'TOXX run aborted: TOXX cutoff less than pMxCut.'
c
c-----------------------------------------------------------------------
c
c     >> Error 35: Invalid warning code.
c
  350 continue
```

```
c
c     Error 35 screen output.
c
      print '(/A, I2, A, A, /A)',
     .    ' Error: The value of variable wrCode (= ',iParm1,') is not ',
     .    'valid.',
     .    '        No warning condition matches this value.'
      print '(/A, A, /A, A, /A, A, /A, A, /A, A, /A)',
     .    ' Solution: This error is highly unusual, and probably refle',
     .    'cts a',
     .    ' typing error in the TOXX source code.  Try to trace to th',
     .    'e point',
     .    ' in the code which called the Warng subroutine, and check t',
     .    'he value',
     .    ' of the second parameter passed to the Warng subroutine.  I',
     .    't should',
     .    ' match the number of the most appropriate warning message, ',
     .    'given the',
     .    ' context of the calling statement, in subroutine Warng.'
c
c     Error 35 file output.
c
      write (fu, '(/A, I2, A, A, /A)')
     .    'Error: The value of variable wrCode (= ',iParm1,') is not v',
     .    'alid.',
     .    '       No warning condition matches this value.'
      write (fu, '(/A, A, /A, A, /A, A, /A, A, /A, A, /A)')
     .    'Solution: This error is highly unusual, and probably reflec',
     .    'ts a',
     .    'typing error in the TOXX source code.  Try to trace to the',
     .    ' point',
     .    'in the code which called the Warng subroutine, and check th',
     .    'e value',
     .    'of the second parameter passed to the Warng subroutine.  It',
     .    ' should',
     .    'match the number of the most appropriate warning message, g',
     .    'iven the',
     .    'context of the calling statement, in subroutine Warng.'
c
c     Error 35 closing.
c
      write (fu, '(/A)') 'TOXX run aborted.'
      print '(A)', '    '
      stop  'TOXX run aborted.'
c
      end
c
c     end subroutine Error
c
```

TECHNICAL REPORT DATA

(Please read Instructions on reverse before completing)

1. REPORT NO.	2.	3. RECIPIENT'S ACCESSION NO.
EPA-453/R-94-058B		

4. TITLE AND SUBTITLE	5. REPORT DATE
Toxic Modeling System Short-Term (TOXST) User's Guide: Volume II	July 1994
	6. PERFORMING ORGANIZATION CODE
7. AUTHOR(S)	8. PERFORMING ORGANIZATION REPORT NO.
9. PERFORMING ORGANIZATION NAME AND ADDRESS Sullivan Environmental Consulting, Inc. 1900 Elkin St, Suite 200 Alexandria, VA 22308	10. PROGRAM ELEMENT NO.
	11. CONTRACT/GRANT NO. 68D20189
12. SPONSORING AGENCY NAME AND ADDRESS U.S. Environmental Protection Agency Office of Air Quality Planning and Standards Emission Standards Division Research Triangle Park, NC 27711	13. TYPE OF REPORT AND PERIOD COVERED
	14. SPONSORING AGENCY CODE

15. SUPPLEMENTARY NOTES
EPA project manager: David E. Guinnup

16. ABSTRACT

This document describes the Toxic Modeling System Short-Term (TOXST) and provides instructions on its implementation. TOXST is a personal-computer-based model that has been developed in conjunction with the release of the EPA's Industrial Source Complex Dispersion Model (EPA, 1992) and the publication of the EPA's "A Tiered Modeling Approach for Assessing the Risks Due to Sources of Hazardous Air Pollution," (EPA-450/4-92-001). The purpose of TOXST is to assist in the evaluation of acute health hazards that may result from short-term exposure to air pollutants. This document represents a revision of the earlier version of TOXST which incorporates additional options for simulating complex intermittent sources of industrial pollution.

17. KEY WORDS AND DOCUMENT ANALYSIS

a. DESCRIPTORS	b. IDENTIFIERS/OPEN ENDED TERMS	c. COSATI Field/Group
Air Pollution Atmospheric Dispersion Modeling Air Toxics Risk Assessment		

18. DISTRIBUTION STATEMENT	19. SECURITY CLASS *(Report)*	21. NO. OF PAGES
Release Unlimited	Unclassified	
	20. SECURITY CLASS *(Page)* Unclassified	22. PRICE

EPA Form 2220-1 (Rev. 4-77) PREVIOUS EDITION IS OBSOLETE