REPORT NO. DOT-TSC-OST-73-24

THE TRANSPORTATION AIR POLLUTION STUDIES (TAPS) SYSTEM

David S. Prerau
Paul J. Downey

MARCH 1974

INTERIM REPORT

Prepared for
DEPARTMENT OF TRANSPORTATION
OFFICE OF THE SECRETARY
Office of the Assistant Secretary for
Systems Development and Technology
Washington DC 20590

NOTICE

This document is disseminated under the sponsorship of the Department of Transportation in the interest of information exchange. The United States Government assumes no liability for its contents or use thereof.

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
DOT-TSC-OST-73-24		

4. Title and Subtitle	5. Report Date
THE TRANSPORTATION AIR POLLUTION STUDIES (TAPS) SYSTEM	March 1974
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
David S. Prerau, Paul J. Downey	DOT-TSC-OST-73-24
9. Performing Organization Name and Address Department of Transportation Transportation Systems Center Kendall Square Cambridge MA 02142	10. Work Unit No. R-3539/OS-322
	11. Contract or Grant No.
12. Sponsoring Agency Name and Address Department of Transportation, Office of the Secretary, Office of the Assistant Secretary for Systems Development and Technology Washington DC 20590	13. Type of Report and Period Covered Interim Report June 1972 - May 1973
	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

This report describes the Transportation Air Pollution Studies (TAPS) Data Base and the Software System which has been developed in association with it.

The TAPS Data Base will be used to store the transportation air pollution data (including emissions, meteorological and other data) which are required for the TSC model validation program. The TAPS System is a package of computer programs for storing, manipulating and retrieving data. The system also contains routines for analyzing the performance of dispersion models as well as programs to generate both tabular and graphical output.

Users guides for both the storage and retrieval programs of the TAPS System are included as well as examples of how these programs might be used. The report also contains complete listings for the TAPS System.

17. Key Words	18. Distribution Statement
Air Pollution, Computer Mode, TAPS, Information Storage and Retrieval, Statistical Testing, Data Base	DOCUMENT IS AVAILABLE TO THE PUBLIC THROUGH THE NATIONAL TECHNICAL INFORMATION SERVICE, SPRINGFIELD, VIRGINIA 22151.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	124	

Form DOT F 1700.7 (8-69)

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT)

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

The responsibility of the Department of Transportation to carefully and systematically consider the environmental effects of its actions and to aid and guide state and local agencies in this area has been made clear by recent legislation, including the National Environmental Policy Act of 1969, the Airport and Airways Development Act, the Mass Transportation Act, the Federal-Aid Highway Act, and the Clean Air Amendments of 1970. These Acts, as well as other legislative mandates, require transportation planners, technologists, and systems developers to consider the environmental effects of all transportation-related actions. This necessitates the development of analytic techniques to determine the environmental impact of transportation systems. A vital part of this is the development of computer models for the estimation of the dispersion of transportation-generated air pollution. There is an immediate need by the Department of Transportation and by state and local agencies for such models.

With this end in mind the Transportation Systems Center (TSC) initiated a program aimed at the constructing and testing of computer dispersion models and the analytic techniques used in such models.

This program encompassed the following steps:

1. State-of-the-Art
2. Model Acquisition
3. Data Acquisition
4. Computer Systems for Model Testing
5. Data Base Design

In response to a questionnaire sent to developers of computer model programs for air pollution analysis, a report entitled, "COMPUTER MODELING OF TRANSPORTATION-GENERATED AIR POLLUTION, A State-of-the-Art Survey," Report No. DOT-TSC-OST-72-20 by Eugene M. Darling, Jr., was published in June 1972. This report is

concerned with the types of computer models available, the pollutants measured, the model input and output, and model validation. Since then, TSC has taken steps to acquire models as well as data to validate and test these models. Each model developer has been invited to submit a copy of his programs to TSC for testing and validation. The Center expects to acquire a sufficient amount of data to test computer models under a wide variety of meteorological and geographic conditions.

The Transportation Air Pollution Studies (TAPS) System has been designed to test and validate transportation air pollution models. It consists of a set of software routines (FORMAN, DARES, SMOG, DIMOTE) and a data base. FORMAN and DARES were written to allow manipulation of data going into and coming out of the data base. SMOG and DIMOTE are used in the analysis of the model results. This report is concerned with the design of the data base and the use of the TAPS system.

2. THE TAPS DATA BASE

The TAPS Data Base was designed as a standard storage structure for highway and airport air pollution data. It is from this data base that air pollution model programs are tested, evaluated and analyzed.

The acquisition of highway air pollution data is not a simple undertaking and is by no means inexpensive. Once the site has been selected, receptors must be strategically positioned for each pollutant to be measured; traffic counts and traffic flows must be obtained; wind speeds, wind directions and background concentrations of pollutants must be tabulated. Data gathered at different times and dates vary greatly and should be treated as separate cases. The data collected at an airport are similar to the highway data but much more extensive. Not only are most of the highway-type data pertinent, but additional data such as time vs. operating mode, arrivals and departures, runway lengths, heating plant emissions, and plane type are also required.

The data base must be designed to accommodate all data collected. The data themselves present problems. There obviously are no standard units or format for air pollution data collection at either highways or airports. Since the air pollution data may be received from many different agencies, companies, etc., the integrity of the data has to be maintained while the units and format must be easily recognizable.

There are additional factors that must be considered in data base design. For example, the data base should be designed to minimize redundancy. Also, since it is to be expected that the data will, in general, be entered into the data base once but will be retrieved several times, the data base must be designed to emphasize ease of retrieval rather than ease of storage.

With the aforementioned criteria in mind, the Transportation Air Pollution Studies (TAPS) Data Base was designed. Each dataset in the Data Base is stored in a standard form, the TAPS Standard

Format. To minimize the redundancy inherent in repeating static site information for several data-sets measured at the same site, it was decided that the primary division of the Data Base would be by the site of data-set measurements. Thus all cases of data-sets measured on the same site are grouped together. For each case, i.e. for each data-set, the names and the values of each of the parameters measured must be indicated. Therefore, in the TAPS Data Base, it was decided that there would be associated with each case a set of parameter pointers (one pointer for each parameter that the data-set might have measured), and each pointer would indicate where the corresponding data were stored.

Based upon the above design decisions, the TAPS Data Base was designed as a four-level structure. Level I contains the site and site description data. Level II contains case pointers. Level III contains parameter definition pointers for both highways and airports. Level IV is the actual data. See Figure 2-1 for a general view of the four-level structure.

2.1 LEVEL I, THE SITE LEVEL

Level I of the Data Base is defined as the Site Level. The purpose of this level is to define and describe a location at which air pollution measurements were taken. This level of the Data Base can be thought of as a summary level. That is, the data represent a summation of all cases at this site entered in the data base. This summary also applies to meteorological conditions and pollutants sampled. A brief description of the data that comprise Level I is presented in Table 2-1. An example of a Level I entry is shown in Table 2-2.

2.2 LEVEL II, THE CASE LEVEL

Level II of the Data Base is called the Case Level. This level contains fixed length records of case pointers (disk pointers) to Level III and is updated as new cases are entered into the Data Base. Each site, as it is defined in Level I, is assigned a fixed length record (50 words) in Level II. As more cases are added for

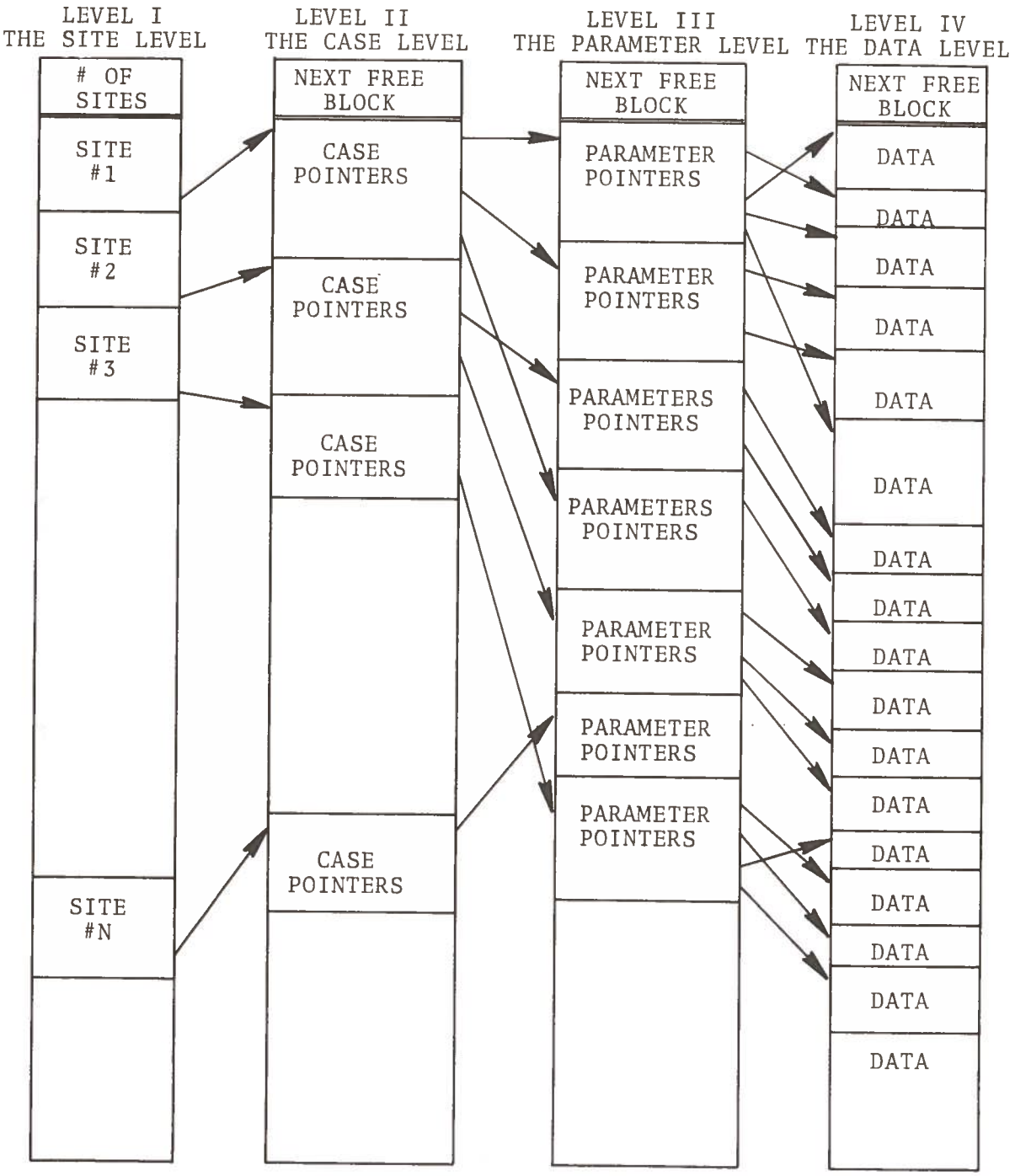

Figure 2-1 Data Base Structure

TABLE 2-1 LEVEL I DATA

WORD 1	"HWY" for a highway; "APT" for an airport
WORD 2-9	32 alphanumeric characters identifying the location of the site.
WORD 10	Wind Speed 0-5 mph
11	" " 5-10 mph
12	" " 10-20 mph
13	" " >20 mph
WORD 14	Stability Class A
15	" " B
16	" " C
17	" " D
18	" " E
19	" " F
WORD 20	Pollutant Indicator: Carbon Monoxide
21	" " Sulphur Dioxide
22	" " Particulates
23	" " Nitrogen Oxides
24	" " Lead
25	" " Hydrocarbons (Total)
26	" " Hydrocarbons (Reactive)
27	" " Aldehydes
WORD 28	Road or Airport Type
WORD 29	Level II Pointer
WORD 30	Site Description Pointer
WORD 31	Number of Case at Site
WORD 32-35	Unused

An example of a LEVEL I entry can be seen in Figure 2-1.

TABLE 2-2 SAMPLE SITE ENTRY*

1.	HWY
2.	1-93
3.	STON
4.	EHAM
5.	MASS.
6.	
7.	
8.	
9.	
10.	6
11.	15
12.	2
13.	0
14.	1
15.	1
16.	6
17.	12
18.	0
19.	3
20.	12
21.	2
22.	7
23.	0
24.	1
25.	0
26.	6
27.	
28.	2
29.	30571
30.	1172
31.	23
32-25.	0

*All numbers are decimal

this site, the record is filled one word per case. An overflow word accompanies each record. When overflow occurs, the overflow word is set with a pointer to the next record where case pointers are stored for this site. Word 1 of Level II contains a pointer to the first free record in Level II. This pointer is used when new sites are added to the data base, or an overflow occurs in Level II. A general view of Level II is shown in Figure 2.2. An example of Level II is seen in Figure 2.3.

2.3 LEVEL III, THE PARAMETER LEVEL

Level III of the Standard Data Base is referred to as the Parameter Level. The purpose of this level is to indicate the existence, or non-existence, of specified parameters in each data-set stored in the Standard Data Base. Three types of parameters are specified in this level: meteorological, emission, and general. It was found that these three categories were extensive enough to classify all input data for air pollution model programs surveyed by TSC.

For each case, 153 consecutive words is reserved on the disk. This is divided into three Blocks - one for each parameter type. Block 1 is for Meteorological data, Block 2 is for Emission data, and Block 3 contains General data. Each Block can contain up to 50 Items (data definition words), plus an overflow word for record continuation. A pointer to Block 1 is stored is Level II.

Note that only disk address pointers are stored in Level III, not actual input data. Associated with each Item in each of the three Blocks is a data definition. If that particular type of data is defined for this case, a pointer is stored in that word. The pointer indicates the address where the data can be found on Level IV. Since air pollution model programs designed for airports require different inputs than roadway model programs, Level III definitions will vary, depending on whether a case pertains to a roadway or an airport site. The storage method and linking concept will be exactly the same. Table 2-3 lists sample Level III entries and Figure 2.4 shows the Level III storage structure.

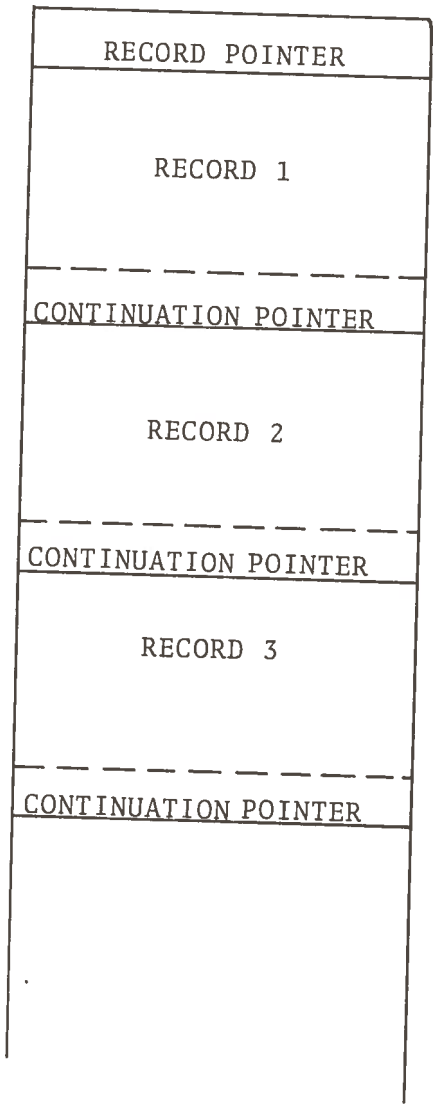

Figure 2-2 Level II Structure

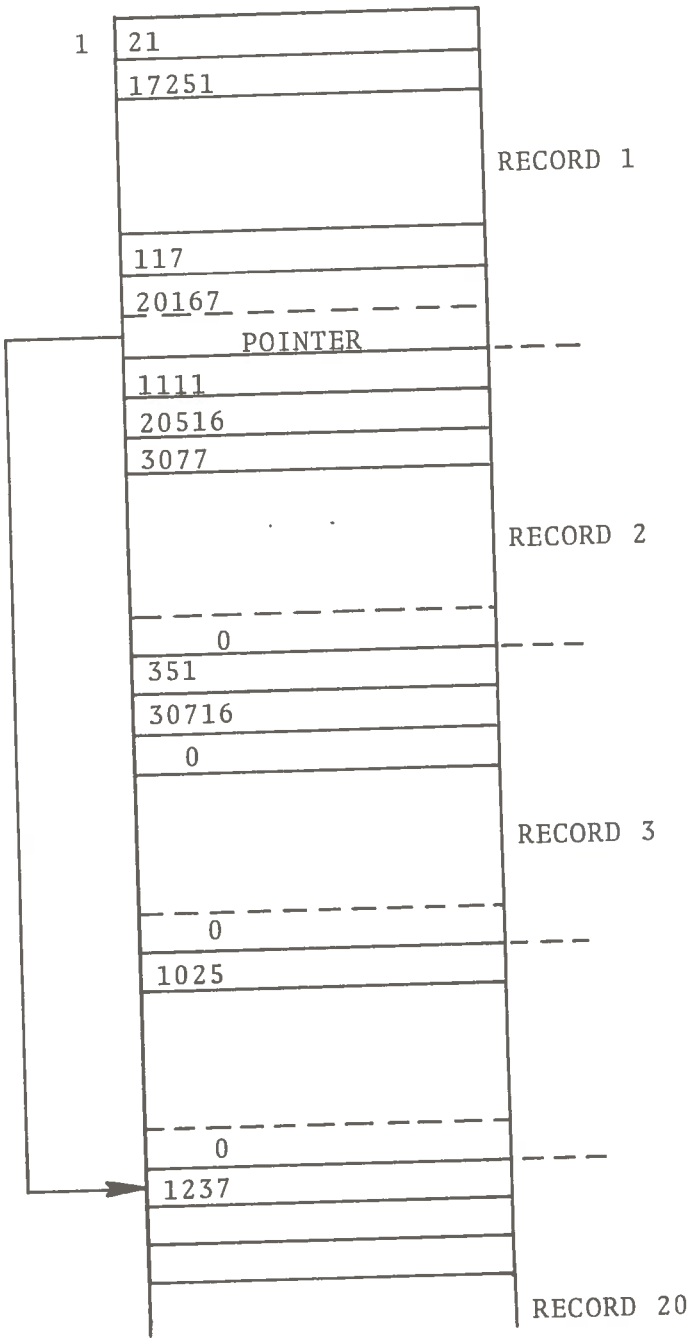

Figure 2-3 Level II Example

TABLE 2-3-a LEVEL III - EMISSION BLOCK, HIGHWAY

1. Source strength
2. Ground level concentration
3. Vehicle speed
4. Type of vehicle
5. Average number of vehicles of each type per period
6. Number of cars
7. Estimated traffic diurnal variation
8. Percentage of traffic in grid
9. Roadway and background surface fluxes
10. Average emission rate
11. Emission rate/unit area
12. Grid description
13. Emission by grid block
14. Exhaust initial conditions: velocity
15. Exhaust initial conditions: temperature
16. Exhaust initial conditions: pollutants
17. Correction for cold start factors
18. Distribution of freeway and surface street mileage
19. Temporal distribution for freeway and surface streets
20. Rate of emission of species from elevated sources
21. Correction for variations of emission rates with auto speed
22. Rate of production of species through chemical reaction

23-50. Currently blank

TABLE 2-3-b LEVEL III - EMISSION BLOCK, AIRPORT

1. Number of sources
2. Source locations
3. Source extensions
4. Source heights
5. Source emission rates
6. Source emission time schedule
7. Diameter of source
8. Time coordinates of source
9. Emission factors for aircraft class
10. Emission factors for modes of operation
11. Emission factors for surroundings
12. Emission factors for heating fuels
13. Emission factors for fuel storage areas
14. Emission rate per unit area
15. Heat capacity
16. Vertical velocity
17. Ground level concentration
18. Crosswind distance from source
19. Alongside distance from source
20. Grid description
21. Distance travelled in grid
22. Emission by grid block
23. Aircraft speed
24. Aircraft mix
25. Number of aircraft in time period

TABLE 2-3-b LEVEL III - EMISSION BLOCK, AIRPORT (CONTINUED)

26. Aircraft arrival and departure
27. Volume flux for each species of pollution
28. Pollutant emission flux for point source
29. Background surface fluxes
30. Exhaust initial conditions: velocity
31. Exhaust initial conditions: temperature
32. Exhaust initial conditions: pollutants
33. Rate of production of species thru chemical reaction

34-50. Currently blank

TABLE 2-3-c LEVEL III - METEOROLOGICAL BLOCK, HIGHWAY

1. Cloud cover
2. Cloud fraction
3. Cloud turbulence intensity
4. Cloud albedo
5. Stack height
6. Ceiling height
7. Inversion height
8. Mixing height
9. Distance of mixing height
10. Diffusion coefficient
11. Downwind diffusion coefficient
12. Crosswind diffusion coefficient
13. Vertical eddy diffusivity
14. Horizontal eddy diffusivity
15. Turbulence intensity
16. Cloud turbulence intensity
17. Stability class
18. Plume constant
19. Richardson's number
20. Radiosonde data pair
21. Ambient air temperature
22. Temperature differences for sea and lake breeze effects.
23. Average Rural temperatures
24. Average temperature lapse rate
25. Temperatures
26. Air pressure

TABLE 2-3-c LEVEL III - METEOROLOGICAL BLOCK, HIGHWAY (CONTINUED)

27. Average rural pressure
28. Atmospheric surface pressure
29. Relative humidity
30. Dew Point
31. Sun elevation
32. Solar radiation
33. Mean wind direction
34. Wind variability
35. Wind speed
36. Wind vectors
37. Horizontal winds
38. Wind class frequency
39. Wind variation with altitude
40. Wind shear
41. Wind rose
42-50. Currently blank

TABLE 2-3-d LEVEL III - METEOROLOGICAL BLOCK, AIRPORT

1. Cloud cover
2. Cloud fraction
3. Cloud turbulence intensity
4. Cloud albedo
5. Stack height
6. Ceiling height
7. Inversion height
8. Mixing height
9. Distance of mixing height
10. Diffusion coefficient
11. Downwind diffusion coefficient
12. Crosswind diffusion coefficient
13. Vertical eddy diffusivity
14. Horizontal eddy diffusivity
15. Turbulence intensity
16. Cloud turbulence intensity
17. Stability class
18. Plume constant
19. Richardson's number
20. Radiosonde data pair
21. Ambient air temperature
22. Temperature differences for sea and lake breeze effects.
23. Average rural temperatures
24. Average temperature lapse rate
25. Temperatures
26. Air pressure

TABLE 2-3-d LEVEL III - METEOROLOGICAL BLOCK, AIRPORT (CONTINUED)

27. Average rural pressure
28. Atmospheric surface pressure
29. Relative humidity
30. Dew point
31. Sun elevation
32. Solar radiation
33. Mean wind direction
34. Wind variability
35. Wind speed
36. Wind vectors
37. Horizontal winds
38. Wind class frequency
39. Wind variation with altitude
40. Wind shear
41. Wind rose
42-50. Currently blank

TABLE 2-3-e LEVEL III - GENERAL BLOCK

1. Road elevation
2. Road cross-section
3. Road orientation
4. Receptor coordinates
5. Receptor calibration factors
6. Receptor background concentrations
7. Chemical rate constants
8. Half-life of pollutants in atmosphere
9. Reaction rate for each mass species
10. Vegetation absorption coefficient
11. Diurnal scaling factor
12. Seasonal scaling factor
13. Day
14. Month
15. Year
16. Sampling interval
17. Accuracy of solution
18. Number of point sources
19. Type of model
20. City information
21. Traffic code
22. Terrain
23. Topographic grid height
24. Isopleth concentration desired
25. Ground boundary conditions
26. Starting time

TABLE 2-3-e LEVEL III - GENERAL BLOCK (CONTINUED)

27. Ending time

28-50. Currently blank

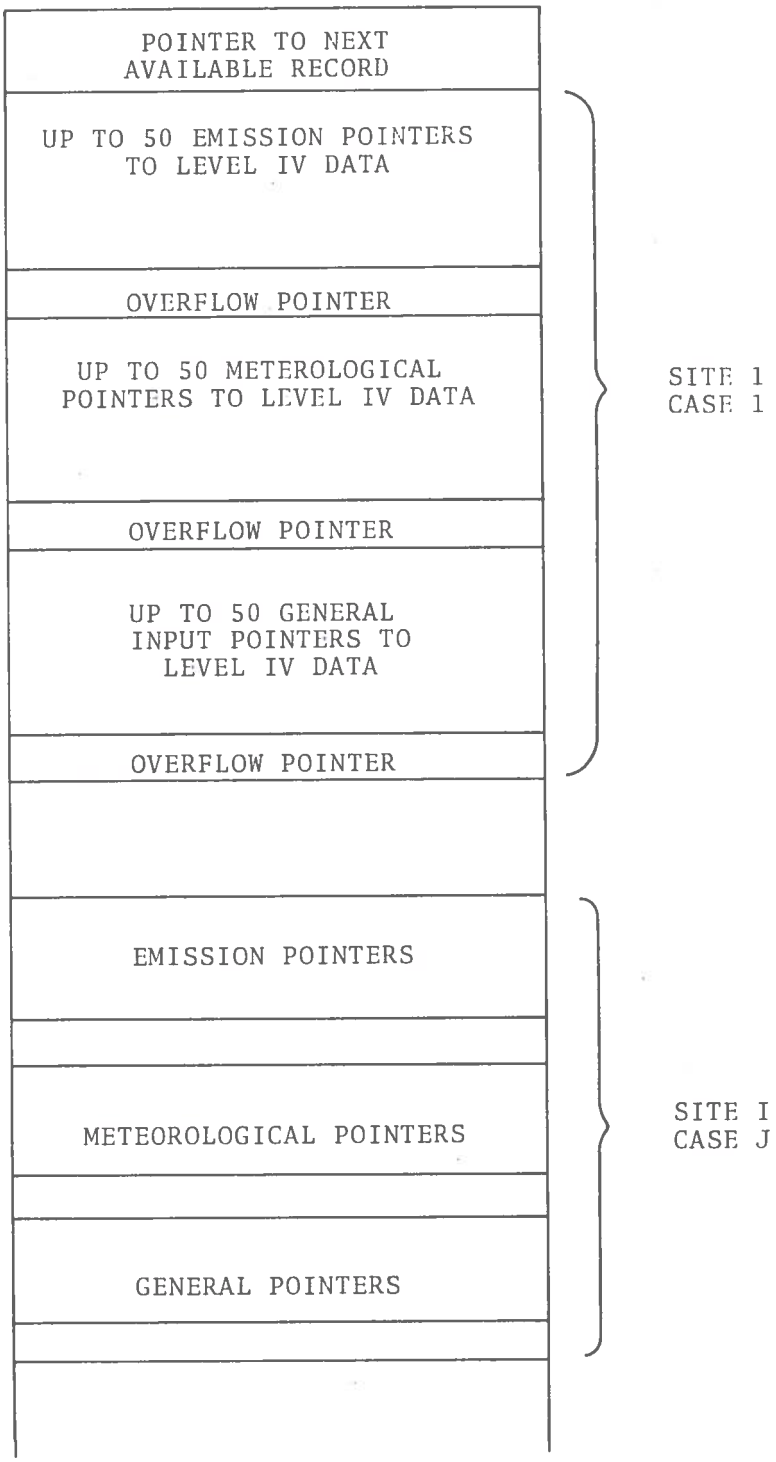

Figure 2-4 Level III Storage Structure

2.4 LEVEL IV, THE DATA LEVEL

Level IV of the data base is defined as the Data Level. It is on this level of the four level structure that the actual data are stored. All data are stored as floating point numbers.

Before describing the storage structure and defining the type of data stored, a brief comment on the input data themselves may be in order. As far as can be ascertained, no two air pollution model programs use the same input data.

Furthermore, not only do models use different types of input data, but they also generally require different units for the input that they have in common. For example, for pollutant concentration, one model may use parts per million (ppm); another, grams per cubic meter; and still another, tons. Each unit will be designated in the Data Base by a specific integer which will be called the Units Number. There will be a Units Number stored in the Data Base corresponding to each stored data value or set of values.

Another problem is the form in which data are accumulated. One data collector may record traffic by the number of cars per hour, another by cars per lane, another by cars per direction, etc. Each form will be designated in the Data Base by a specific integer which will be called the Form Number. There will be a Form Number stored in the Data Base corresponding to each stored data value or set of values. The "usual" form of each type of data will be designated as Form No. 0.

In order to handle these two problems (i.e., the units and the form of the data), Level IV had to be made flexible while at the same time maintaining the integrity of the data.

In order to achieve these characteristics, a set of data headers was designed to precede each data entry in Level IV. The following header types were found to adequately handle the present data requirements:

SINGLE NUMBER

VECTOR

MIXED VECTOR

MATRIX - 2-DIMENSIONAL

COLUMN MATRIX - 2 DIMENSIONAL

GRID - 2 DIMENSIONAL

POINT VALUES - 2 DIMENSIONAL

MATRIX - 3 DIMENSIONAL

GRID - 3 DIMENSIONAL

POINT VALUES - 3 DIMENSIONAL

NON-UNIFORM GRID - 2 DIMENSIONAL

NON-UNIFORM GRID - 3 DIMENSIONAL

By use of these data headers, data manipulation programs will be able to convert data from their stored form and units to the form and units needed by an air pollution model porgram. In this way one data set can be tested on several models. The data headers are discussed in detail below.

2.4.1 Single Number Header

This header has the following form:

S

FORM NO.

UNITS NO.

VALUE

The character "S" signifies that the value stored is a single number. The second word is the form number. The third word is the units number. The fourth word contains the value itself.

EXAMPLE:

S

2

10

4534.2

The number 4534.2 is stored in form number 2 (e.g., 8 lane traffic average) and units 10 (e.g., vehicles per hour)

2.4.2 Vector Header

This header has the following form:

V
FORM NO.
UNITS NO.
NO. OF ELEMENTS
$VALUE_1$
$VALUE_2$
$VALUE_3$
$VALUE_n$

The character "V" signifies that the data is stored in a vector. The second word is the form number; the third word, the units number, the fourth word contains a count of the number of values in the vector; the next n words contain the values themselves.

EXAMPLE:

V
1
0
8
2500
1500
1700
1200
3000
2500
2100
2000

Eight values for cars per hour per lane are stored in this vector. The second word is a 1 indicating this form. The third word is a zero meaning, the data is dimensionless. The fourth word is 8, indicating the length of the vector.

2.4.3 Mixed Vector Header

This header has the following form:

VM

FORM NO.

NO. OF ELEMENTS

$UNITS_1$

$UNITS_2$

$UNITS_3$

$UNITS_n$

$VALUE_1$

$VALUE_2$

$VALUE_3$

$VALUE_n$

The characters "VM" signify that the data stored are a mixed vector (mixed in the sense that several different units have been used); the second word contains the form; the third word, the number of elements; the next n words contain the units for the succeeding n words, which are the n values.

EXAMPLE:

VM

0

6

1

2

2

1

1

1

16

40

19

10

35

50

A mixed vector of 0 form is six words in length. The next six words are the units (1,2,2,1,1,1) for the last six values.

2.4.4 Matrix Header: 2-Dimensional

This header has the following form:

M2

FORM NO.

UNITS NO.

NO. OF ROWS

NO. OF COLS.

VALUE 1,1

VALUE 2,1

.

.

.

VALUE N,M

The second and third words contain the form number and units, respectively; the fourth and fifth words contain the number of rows and number of columns in the matrix; the next N x M words contain the data.

EXAMPLE:

M2
0
3
4
3
8.2
7.5
10.6
50.0
22.5
30.6
17.9
34.0
25.7
26.8
19.0
6.1

A 2 dimensional matrix is stored with 0 form and a unit number 3 (e.g., parts per cubic centimeter); the matrix a 4 x 3. The last 12 words are the values.

2.4.5 Column Matrix Header: 2-Dimensional

This header has the following form:

C2

FORM NO.

NO. OF ROWS

NO. OF COLS.

UNITS FOR COL_1

UNITS FOR COL_2

UNITS FOR COL_3

.

.

.

.

UNITS FOR COL_m

VALUE 1,1

VALUE 2,1

VALUE 3,1

.

.

.

VALUE N,M

The characters "C2" signify that the data stored are a column matrix; the second word is the form; words three and four indicate the number of rows and columns; the next m words indicate the units for each column in the matrix; the next N x M words are values of the matrix elements.

EXAMPLE:

C2

0

2

5

1

2

4

5

3

7.4

10.0

1000.7

1021.0

51.1

106.2

75.1

54.7

2000.480

3126.721

A column matrix is stored with 0 form. The matrix is two rows by five columns. The units for each column are the next five words (1,2,4,5,3). The last ten words are the ten matrix values.

2.4.6 Grid Header: 2-Dimensional

This header has the following form:

G2

FORM NO.

UNITS FOR X and Y

NO. OF X's

NO. OF Y's

DELTA X

DELTA Y

X_o

Y_o

NO. OF VALUES PER POINT

$UNITS_1$

$UNITS_2$

.

.

.

$UNITS_n$

VALUE $(1,1)_1$

VALUE $(1,1)_2$

VALUE $(1,1)_3$

.

.

.

VALUE $(1,1)_N$

VALUE $(2,1)_1$

.

.

.

.

The characters "G2" indicate that these data are stored in a 2-dimensional grid; the second word contains the form number; the third word, the units number for the X's and Y's; the fourth and fifth words contain the number of X's and Y's; delta X and delta Y are the grid increments; X_o and Y_o are the initial X and Y; the next word contains the number of values stored at each point in the grid; the next n words contain the units for each value recorded at the point; finally, the data values recorded at each point in the grid are entered.

EXAMPLE:

G2

0

2

2

4

2

2
20
10
3
1
3
2
6
6
5
12
14
10
17
50
18
6
8
12
20
22
20
14
13
12
18
25

27

20

19

18

A 2-dimensional grid of form 0 is stored with X and Y units of type 2. This will be a 2 x 4 grid with a delta X and delta Y of 2. The initial X will be 20 and the initial Y will be 10. There are three values stored for each grid point. The units for these three values are 1,3,2. The next 24 numbers are values recorded at each grid point.

2.4.7 Point Value Header: 2-Dimensional

This header has the following form:

P2

FORM NO.

UNITS FOR X AND Y

NO. OF POINTS

NO. OF VALUES/PT.

$UNITS_1$

$UNITS_2$

.

.

.

$UNITS_n$

X_1

Y_1

$VALUE_1$ (1)

$VALUE_2$ (1)

.

.

.

.

$VALUE_N$ (1)

X_2

X_2

$VALUE_1$ (2)

$VALUE_2$ (2)

.

.

.

$VALUE_N$ (2)

.

.

.

$VALUE_N$ (M)

The characters "P2" indicate that the data are stored by a point value method; next are the form number and the type of units for X and Y; the following two words contain the number of points (X,Y) and the number of values recorded at each point; the next n words contain the units for each of the recorded values; then come X and Y, followed by the n values recorded at this point. Then, the X and Y for the next point are followed by its values, and so on.

EXAMPLE:

P2

0

1

4

3

1

```
 3
 2
50.7
14.3
 7.5
52.6
14.1
20.2
17.5
 9.6
35.7
15.4
30.1
17.9
12.1
35.9
14.9
17.1
16.2
20.1
30.1
15.6
```

A point value array of form 0 is shown with units of type 1 for X and Y. There are 4 points and there are 3 point values stored for each point. The units for each of the 3 point values are 1,3,2. The next 2 words are the X and Y values followed by the 3 corresponding point values. Each of the remaining 3 points is followed by its associated point values.

2.4.8 Matrix Header: 3-Dimensional

This header has the following form:

M3

FORM NO.

UNITS NO.

NO. OF ROWS

NO. OF COLS.

NO. OF LEVELS

VALUE 1,1,1

VALUE 2,1,1

.

.

.

VALUE L,M,N

The characters "M3" indicate that the data stored is a 3-dimensional matrix. The form and units numbers are the second and third words; the next 3 words are the number of rows, columns, and levels (or third dimension) of the matrix; the following (L x M x N) words are the elements of the matrix.

EXAMPLE:

M3

0

12

4

2

3

100.71

71.5

```
  12.6
 200.5
  17.7
  16.2
  51.0
  19.7
  43.2
  37.5
  22.3
  11.6
 300.2
  66.7
   8.1
 151.2
  77.6
  81.5
  55.5
 250.1
 220.6
   9.1
 130.6
  75.0
```

A 3-dimensional matrix of form 0 and units 12 is stored in this array. The matrix has the dimensions 4 x 2 x 3. The last 24 words contain the elements of the matrix.

2.4.9 Grid Header: 3-Dimensional

This header has the following form:

G3

FORM NO.

UNITS FOR X,Y,Z

NO. OF X's

NO. OF Y's

NO. OF Z's

DELTA X

DELTA Y

DELTA Z

X_o

Y_o

Z_o

NO. OF VALUES PER PT.

UNITS OF $VALUE_1$

UNITS OF $VALUE_2$

.

.

.

UNITS OF $VALUE_N$

VALUE $(1,1,1)_1$

VALUE $(1,1,1)_2$

.

.

.

VALUE $(1,1,1)_N$

VALUE $(2,1,1)_1$

.

.

.

.

VALUE $(Nx, Ny, Nz)_N$

This header is similar to the 2-dimensional grid header discussed in Section 2.4.6

EXAMPLE:

```
G3
 0
 6
 3
 2
 2
  .05
 1.0
 0.5
10.0
15.0
10.0
 2
10
12
50.0
17.5
101.6
90.0
 7.2
45.1
30.2
```

63.7

7.1

18.2

10.5

70.6

51.5

9.6

12.4

112.4

44.4

89.7

91.2

175.0

160.2

29.1

22.7

90.4

A 3-dimensional grid contains data with a form 0 and units of type 6 for X,Y, and Z. The grid is 3 x 2 x 2 with delta X, delta Y, and delta Z of .05, 1.0, and 0.5 respectively. The initial X,Y and Z are 10.0, 15.0, and 10.0. There are 2 stored values per point. These unit types are 10 and 12 respectively. The last 24 words are the data elements recorded.

2.4.10 Point Value Header: 3-Dimensional

This header has the following form:

P3

FORM NO.

UNIT FOR X,Y,Z

NO. OF POINTS

NO. OF VALUES PER POINT

$UNITS_1$

$UNITS_2$

.

.

.

$UNITS_n$

X_1

Y_1

Z_1

VALUE 1 (1)

VALUE 2 (1)

.

.

.

VALUE N (1)

X_2

Y_2

Z_2

VALUE 1 (2)

.

.

.

VALUE N (2)

.

.

.

.

.

The characters "P3" indicate that a 3-dimensional point value array is stored; next are the form number and units number; the following words contain the number of points and the number of recorded values per point respectively. The next n words contain the units for their associated recorded values. Then each X,Y and Z point is followed by its n values.

EXAMPLE:

```
P3
 0
 5
 4
 3
10
 2
 6
10.4
 5.1
42.6
100.7
74.6
85.9
15.7
25.9
13.1
74.1
17.2
```

9.8

114.1

74.9

48.2

14.2

14.4

24.4

34.4

122.1

98.6

111.1

24.1

34.1

A point value array is stored with form 0 and units type 5. There are four points and three recorded values per point. The units for the three recorded values are 10, 2, and 6. Following each of the point coordinates are the three recorded values at this point.

2.4.11 Non-Uniform Grid Header: 2-Dimensional

This header has the following form:

N2

FORM NO.

NO. OF GRIDS

POINTER TO GRID 1

POINTER TO GRID 2

.

.

.

POINTER TO GRID N

The characters "N2" indicate that the data stored is a non-uniform grid of two dimensions; the second and third words contain the form number; the next n words are pointers to disk addresses where a 2-dimensional grid is stored. See Section 2.4.6 for an explanation of 2-dimensional grids.

EXAMPLE:

```
N2
0
6
10157
20635
1045
35627
47025
21772
```

A non-uniform grid of 2-dimensions is stored in an array of form 0. There are 6 two-dimensional uniform grids that comprise the non-uniform grid. The next six words contain the disk addresses where these grids are stored.

2.4.12 Non-Uniform Grid Header: 3-Dimensional

This header has the following form:

```
N3
FORM NO.
NO. OF GRIDS
POINTER TO GRID 1
POINTER TO GRID 2
.
.
.
```

POINTER TO GRID N

The characters "N3" indicate that a non-uniform 3-dimensional grid is stored in this array; the next word contains the form number; then comes the number of grids; finally there is a pointer corresponding to the disk address of each component uniform grid. See Section 2.4.9 for an explanation of 3-dimensional grids.

EXAMPLE:

```
    N3
     0
     5
 21507
  4507
 75066
 33244
 41257
```

A non-uniform grid of three-dimensions is stored in an array of form 0. There are five 3-dimensional grids that comprise the non-uniform grid. The next five words contain the disk addresses where these grids are stored.

2.5 SITE DESCRIPTION

In Level I there exists a pointer to the site description. This description applies both to the physical structure where the air pollution sampling was performed and also to the location of the receptors used in the sampling. The description is defined initially when the site is created. If another portion of this site is later entered or if new receptors are added to measure pollution concentrations, the information may be easily added later (see FORMAN).

For example, Figure 2.5 lists the elements that comprise the highway site description form number and units are as previously defined in 2.4. Number of sections indicates the number of highway

sections where data sampling has been done. The number of receptors is similarly defined. Next come the end points of the first section. The next three words are the number of lanes, the highway width, and the center strip width. The road type (at-grade, cut, fill, or viaduct) followed by its parameters (if any) are next. Words 5-14 are repeated for each section as needed. Finally, the receptor coordinates are entered. Figure 2.6 shows an example of a two section highway site.

WORD	1	FORM NO.
	2	UNITS
	3	NUMBER OF SECTIONS
	4	NUMBER OF RECEPTORS
	5	1ST END POINT X
	6	1ST END POINT Y
	7	2ND END POINT X
	8	2ND END POINT Y
	9	NUMBER OF LANES
	10	HIGHWAY WIDTH
	11	CENTER STRIP WIDTH
	*12	ROAD TYPE
	*13	ROAD TYPE PARAMETER 1
	*14	ROAD TYPE PARAMETER 2
	15	X1
	16	X1
	17	Z1
	18	X2
	19	Y2
	20	Z2
	.	.
	.	.
	.	.
	.	XN
	.	YN
	.	ZN

*

ROAD TYPE	AT GRADE	CUT	FILL	VIADUCT
13	0	WIDTH OF CUT	HEIGHT OF FILL	HEIGHT OF ROAD
14	0	DEPTH OF CUT	0	0

Figure 2-5 Highway Site Description

WORD	1	0
	2	3
	3	2
	4	5
	5	0.0
	6	500.0
	7	0
	8	-500.0
	9	8
	10	140.0
	11	30.0
	12	1
	13	0
	14	0
	15	0.0
	16	500.0
	17	0.0
	18	1000.00
	19	4
	20	105.0
	21	12.0
	22	1
	23	0
	24	0
	25	80.0
	26	-200.0
	27	1.0
	28	80.0
	29	100.0
	30	1.0
	31	80.0
	32	300.0
	33	1.0
	34	80.0
	35	600.0
	36	1.0
	37	80.0
	38	900.0
	39	1.0

Figure 2-6. Two-Section Highway Site

3. THE TAPS SYSTEM

The Transportation Air Pollution Studies (TAPS) System is a set of computer routines which allows transportation-source air pollution data to be stored in the TAPS Data Base and then used to validate and evaluate transportation-source air pollution dispersion models. Specifically, the TAPS System allows (1) data to be stored in the TAPS Data Base, and retrieved from it, (2) dispersion model programs to be run using the retrieved data as input, and (3) the resulting output of the model program to be compared with measured values and with the results of other model programs to produce an evaluation of the model.

3.1 SYSTEM DESCRIPTION

The TAPS System consists of four parts: FORMAN (The FORmat MANipulator), DARES (The DAta REtrieval System), SMOG (The Standard Model Output Generator), and DIMOTE (The DIspersion MOdel TEster). Briefly, FORMAN stores incoming data in the TAPS Data Base; DARES retrieves data from the TAPS Data Base in a form acceptable as input to the model program being evaluated; SMOG transforms the model program's output to a standard form; and DIMOTE evaluates the model, using a variety of statistical tests to compare the program's output with the results obtained by other model programs and also with the measured values.

The use of the TAPS System is illustrated in Figure 3-1. Its operation is as follows:

a. Incoming Data-Set

For each new transportation-source air pollution data-set received:

1. A set of FORMAN commands is written.
2. The FORMAN Processor uses the FORMAN commands and the new data-set as input, and inserts the data-set into the TAPS Data Base in standard form.

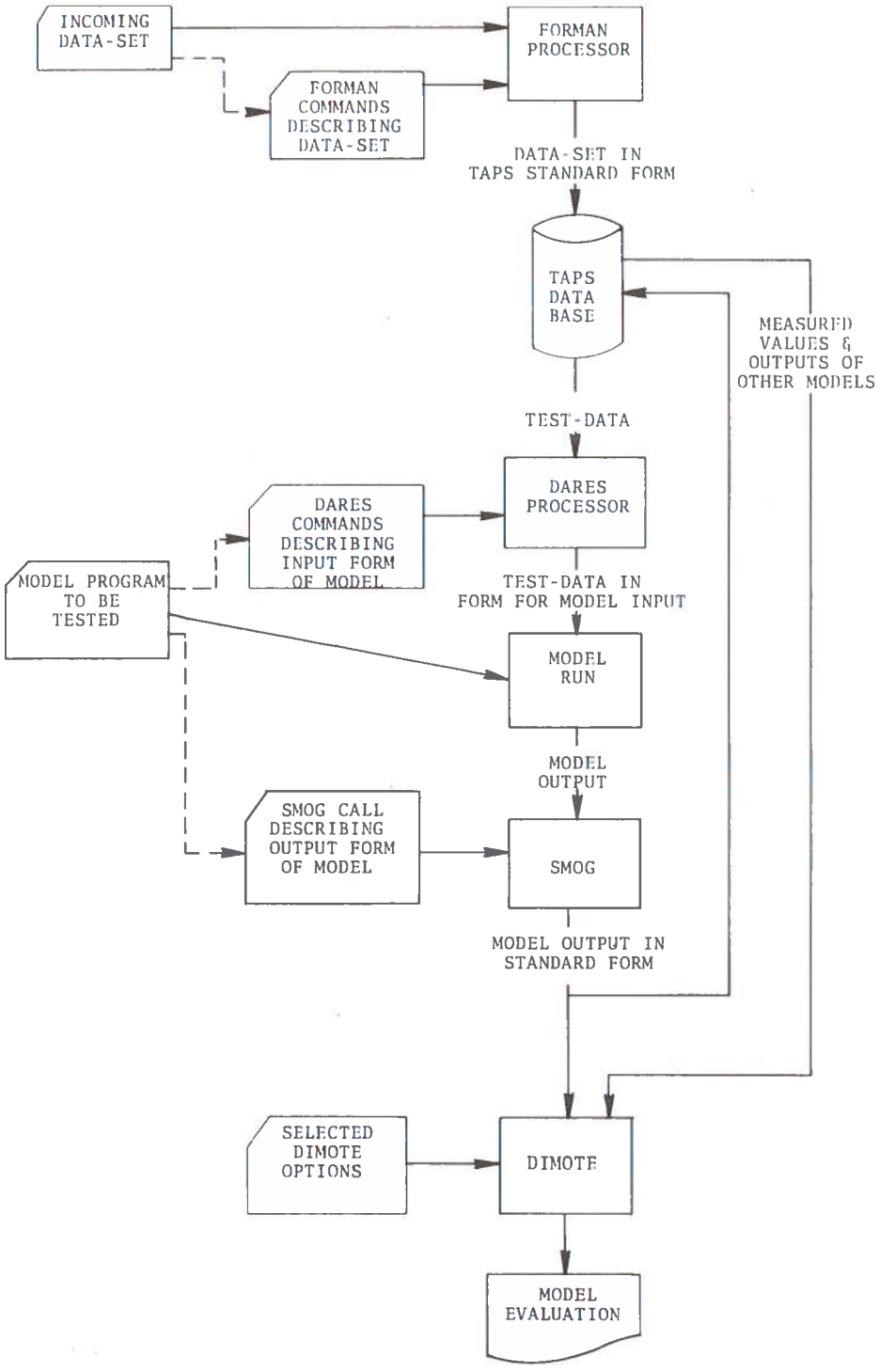

Figure 3-1 The TAPS System

b. Incoming Model Programs

For each new transportation-source air pollution dispersion model program received:

1. A set of DARES commands is written corresponding to the program's inputs.
2. A set of SMOG commands is written corresponding to the model program's output format.

c. Model Testing

To test a given model program using a selected data-set from the TAPS Data Base:

1. The DARES Processor is run using the model's DARES commands as input. It retrieves from the TAPS Data Base a set of test data (from the selected data-set) which is in the proper form for input to the model program.
2. The model is run using the test data produced by the DARES Processor as input.
3. The SMOG Routine is run using the model's SMOG commands as input. It converts the model program's output to a standard output form.
4. Selected DIMOTE Routines are run using the standard output produced by the SMOG Routine as input. DIMOTE compares the tested model's output with the values that were measured and with the outputs previously produced by other models for the same input data-set. A variety of statistical tests are employed.
5. Based on the results of the statistical tests, DIMOTE produces a set of graphs and tables which provide an evaluation of the model and a comparison with other models.

3.2 SYSTEM IMPLEMENTATION

The TAPS Data Base is disk-oriented, and thus a third-generation computer system should be used for the implementation of the TAPS System programs. Since most models to be tested were written in FORTRAN IV for the IBM 360 or 370 computers, it was decided to implement the TAPS System on an IBM 360/75, using FORTRAN IV. This allows the power of disk storage to be used by TAPS and also keeps to a minimum the task of converting models from one computer system to another. The listings of the TAPS System program are found in Chapter 7.

The individual TAPS routines are discussed in detail below.

3.3 FORMAN

The element of the TAPS System that deals with incoming data is the FORmat MANipulator, FORMAN. FORMAN's function is to take any data-set which has been acquired by the Transportation Systems Center, convert it into TAPS Standard Format, and store it in the TAPS Data Base.

The Center is receiving highway and airport air pollution data-sets from many sources. In most cases the data-sets were originally acquired for use in other projects and therefore are received in various formats which are tailored to the original application. Also, various media may be used to transmit the data to TSC, e.g., punch cards, magnetic tape, computer printouts, or even handwritten lists. FORMAN allows the user to conveniently extract the data-set from the medium in which it is received (in whatever format it is received) and store it in TAPS Standard Format in the Data Base. The process of converting the data-set from an arbitrary input form to the TAPS Standard Format is accomplished by the FORMAN Processor. A set of FORMAN Commands is written for each incoming data-set which describes the format of the data-set in terms of the TAPS Standard Format. This set of commands is input to the FORMAN Processor along with the data-set. The FORMAN Processor reads the Commands and enters the data-set into the TAPS Data Base in TAPS Standard Format as directed by the Commands.

By means of these Commands, the FORMAN Processor can:

a. Create a new Site entry in Level I.

b. Update a Site entry.

c. Create, for an already existing Site entry, a new Case entry in Level II and associated Parameter and Data entries in Levels III and IV.

d. Update an already existing case by adding or modifying its parameter and Data entries.

Some examples of FORMAN Commands will now be discussed. A complete description of FORMAN Commands is given in Section 4.1, the FORMAN User's Guide.

3.3.1 Site Creation (NSIT)

Creating a new Site entry is accomplished by a FORMAN "NSIT" command. The FORMAN Commands order is as follows:

a. The 'NSIT' instruction.

b. Two instructions containing data for the Site entry.

c. A set of instructions, each with one number, specifying the site description.

For example, the command shown in Figure 3-2. will create a Level I entry for Site 28. The 'HWY' on instruction 2 indicates that the site is a highway and this is followed by a field of up to 32 alphanumeric characters which identify the site. This is followed by four numbers indicating the breakdown of the cases by wind speed class (e.g., 3 cases of calm, 7 cases of low wind, etc.), and then by six numbers indicating the number of cases of each stability class (e.g., 0 cases of stability A, 6 cases of stability B, etc.)

The third instruction has the number of cases that measure each of eight pollutants (here all 41 cases measure only carbon monoxide). The "1" on the third card indicates an at-grade road, and the "46" is the number of cards in the site description. The cards following give details of the site description in the format

```
NSIT 28
HWY RTE 4, PRERAU FREEWAY, DOWNEY CALIF 3 7 22 9 0 6 18 11 5 1
41  0  0  0  0  0  0  0  1  46
0
13
1
11
-1000.
0.
1000.
0.
8
122.3
26.1
1
0
0
0
500
0
0
400
0
0
300
0
0
200
0
0
100
0
0
0
0
0
-100
0
0
-200
0
0
-300
0
0
-400
0
0
-500
0
```

Figure 3-2 Command: Level I Entry for Site 28

described in Section 2.1. This is an 8-lane, at-grade, east-west highway with 11 receptors placed 100 feet (assuming Units No. 13 is "feet") apart perpendicular to the highway. NSIT will print an error message if the new site being created is to have the same number as an already existing site.

3.3.2 Updating Site (USIT)

The USIT Command will update a Level I Site entry. This allows a previously existing site entry to be partially changed, bypassing the error indication that would result if the NSIT Command was used. The use of the USIT Command will be similar to the NSIT Command as shown in Figure 3.2. USIT will be needed, for example, if a new case added to a site uses a receptor that had not previously been entered in the site description. This receptor will be added to the site description using the USIT Command.

3.3.3 Creating a New Case (NCAS)

The NCAS Command will create a new Case (with its associated Parameter pointer and Data) for a previously created Site. To use the NCAS Command, the order of the FORMAN input is as follows:

a. The 'NCAS' instruction.

b. A set of data cards from which the data will be extracted.

c. A 'FORMAN' instruction.

d. A set of Insert instructions indicating which data to extract from the data cards, and where to place them in the TAPS Data Base.

e. A Header instruction after each Insert instruction that references a Parameter (by Block and Item number) for the first time.

An example of the creation of a new case is shown in Figure 3-3. This NCAS command creates a new case (numbered Case 63) for Site 28.

CARD NO.		INSTRUCTION
1.		NCAS 28 63
2.	DATA	X=14.75 29.84 75.6
3.	DATA	3 14 3 29 1150 10* 3.0
4.	DATA	35.0 45.0 42.5 51.0 70.0 60.7
5.	DATA	35.0 33.1 17 38
6.		FORMAN
7.	INSERT INSTRUCTIONS	3 7 1* 1 1 1 (8X,F5.2)
8.	INSERT INSTRUCTIONS	3 5
9.	INSERT INSTRUCTIONS	2 21 1 1 1 1 (14X,F4.1)
10.	INSERT INSTRUCTIONS	0 25
11.	INSERT INSTRUCTIONS	2 17 1* 1 1 1 (11)
12.	INSERT INSTRUCTIONS	12 9
13.	INSERT INSTRUCTIONS	1 3 2* 6 1 8 (6(F4.1,2X))
14.	INSERT INSTRUCTIONS	3 17 8

Figure 3-3 An Example of NCAS

Following the data and the 'FORMAN' instruction are the Insert instructions. Consider the instruction cards 9 and 10:

```
2  21  1  1  1  1    (14X,F4.1)
   0  25
```

The "2 21" on card 9 indicates that these insertions will fill Block 2, Entry 21 (ambient air temperature) of Level III (of Site 28, Case 63) with data which are specified by the rest of card 9 and by card 10. The first "1" indicates that header 1 (Single Number) should be used. The next three 1's indicate the one value is being read and thus the first and last subscript is 1. The "(14X,F4.1)" indicates that the value is to be found on the data card (in this instance, card 2) by skipping 14 spaces and reading a number of FORTRAN Format notation F4.1 (thus 75.6 is read). The "0 25" on the card 10 indicates that the data are being stored in Form No. 0 (standard form) and the units are Units No. 25 (degrees Fahrenheit, for example). Thus, the final result is that for Site 28 Case 63, there is a pointer in Level III at Block 2, Entry 21 which points to a Level IV entry. The Level IV entry has a Single Number Header, and contains the number 75.6 which is in Form No. 0, Units No. 25.

As another example, consider the last pair of Insert Instructions in Figure 3-3, on cards 13 and 14:

```
1   3   2*   6   1   8   (6(F4.1,2X) )
    3  17  8
```

These insertions will fill block 1, Entry 3 (vehicle speed) with data using header 2 (Vector). The "*" means that data are read starting from the next data card (here the third data card, card 4). There are 6 data entries per card and the input is to be subscripted in the vector from 1 to 8 (thus two values must be read from the succeeding data card). The card format is (6(F4.1,2X)). Form No. 3 (speed by lane, say) and Units No. 17 (miles per hour, say) are used, and there are a total of 8 entries in the vector.

3.3.4 Updating a Case (UCAS)

The UCAS Command will update the Level III and Level IV entries for a specified case. This updating allows new data to be added to the previously created case. UCAS would be used, for example, if data for a specific case were to be found throughout the large data-set. NCAS would be used the first time the data for the case appeared, and UCAS would be used thereafter. The usage of UCAS is similar to that of NCAS as shown in Figure 3-3.

3.4 DARES

Data are retrieved from the TAPS Data Base by the DAta REtrieval System, DARES. The function of DARES is to retrieve the data of an air pollution data-set chosen from the Data Base and to prepare these data in a format which can be directly accepted as input by a given air pollution dispersion model being tested.

Since TSC will test each model program obtained using several of the data-sets stored in the TAPS Data Base, a method is needed to allow a model to be run using as input the data of any given data-set in the Data Base. Two alternative methods can be considered. The first would be to change each model's input commands so that the model program could read directly from the TAPS Data Base. The second would be to retrieve the data from the TAPS Data Base and construct, prior to the running of the model, a set of card images in the input format expected by the model program. This latter alternative was chosen for DARES for three reasons: (1) it is desirable to alter the model program being tested as little as possible, (2) a more accurate evaluation of the computer time used by the model can be found if all input retrievals are performed prior to model run, and (3) error conditions resulting from the lack of a match between the data requirements of the model and the data stored in the chosen data-set can be more easily handled utilizing software fixes or error message printouts.

The process of retrieving desired data from a chosen data-set in the TAPS Data Base and forming them into card images acceptable to the tested model program is accomplished by the DARES Processor. For each model program to be tested, a set of DARES Commands is written describing the expected input cards to the program. The DARES Commands describe the data which must be retrieved from the TAPS Data Base in terms of the standard position where such data is stored in the Data Base. In addition, the Commands indicate the position where such data should be placed on a card image and what other information should be included to form the card image in the format the program expects.

3.4.1 General DARES Format

The format of the input to the DARES Processor is as follows:

a. A set of Text Cards

b. A set of DARES Commands.

The Text Cards are a set of alphanumeric cards (or card-images) which contain all the information which should appear on the model program's input cards, except for the data to be retrieved. This includes such things as program titles, alphanumeric descriptions, and names of variables or vectors where the retrieved data will be stored by the model program. This information should be correctly formatted, with blanks or arbitrary symbols occupying the areas on the cards where the retrieved data should be placed. DARES can retrieve the data from the Data Base, and produce cards (or card-images) which overwrite the data in the correct fields of the Text Cards. DARES can also copy certain Text Cards directly, and create cards which have only data on them.

The use of the overwriting feature of DARES is especially convenient when a model program comes with a sample input deck. The sample input deck can be used as the Text Cards (since this deck is in correct form for input to the model program), and the data values for the Site and Case being used for model testing can be written over the sample input values on the deck. The use of

Text Cards also allows the values of any specified variables in the model program to be set by the DARES user, and not read from the Data Base.

The various DARES Commands allow the DARES Processor to:

a. Specify the Site and Case from which the data is to be retrieved.

b. Retrieve specified TAPS Data Base data and place them in a Data Buffer.

c. Create a model program input by copying a Text Card.

d. Create a model program input card from retrieved data.

e. Create a model program input card by overwriting data on a copy of a Text Card.

An example of a DARES Command set as input to the DARES Processor is shown in Figure 3.4. A typical set of model program input cards produced by the DARES Processor using this Command set is shown in Figure 3.5. This example will be discussed in the following sections to illustrate DARES. A complete description of DARES is given in Section 4.2, the DARES User's Guide.

3.4.2 Site and Case Specification (DARE, END)

Following the set of Text Cards, a DARE Command indicates the beginning of the DARES Commands. The "DARE" card is followed by a card which indicates from which Site and Case in the TAPS Data Base the data will be retrieved. In the example in Figure 3.4, the DARE Command indicates that the first five cards are Text Cards, and that the data will be retrieved from Site 13, Case 37.

An END Command indicates the end of DARES Commands for the specified Site and Case. If data from a second Site and Case were to be retrieved, a DARE Command would follow the first END Command, and another Site and Case would be indicated.

Note that any Text Cards are allowable until the first DARE Command is read by the DARES Processor. Thus the "END" on the fifth card is interpreted as a Text Card, and not a DARES END Command.

Text Cards

```
LOGAN INTERNATIONAL AIRPORT
BOSTON
H=              N=              RH=17.631 EM=
J=14
END
```

DARES Commands

```
DARE
13  37
CARDCOPY
CARDCOPY
RET 2  8        0  12 2     1   1
RETA1  1        0   0 0     1   1
RETA2 35
CARDOVERINT     2(2X,I10,3X,I10)
APNDOVERREAL    1(36X,F5.1)
CARDDATAREAL    4(4(F5.1,1X))
CARDDATAREAL    4(4(F5.1,1X))
CARDDATAREAL    1(F5.1)
RET 1 42        4  29 3   1   1
APNDDATAINT     1(6X,I4)
CARDCOPY
CARDCOPY
END
```

Figure 3-4 Sample DARES Command Set

```
LOGAN INTERNATIONAL AIRPORT
BOSTON
H=        2461 N=         28 RH=17.631 EM=29.3
427.7 384.2  93.1 192.0
135.5 183.3 199.2  82.1
941.4 2317
J=14
END
```

Figure 3-5 Typical Model Program Input Produced by DARES Command Set of 3-4

3.4.3 Data Retrieval (RET and RETA)

The RET and RETA Commands are used to retrieve data from the TAPS Data Base. The data is taken from the Site and Case specified by the last preceding DARE Command.

A Data Buffer is used by RET and RETA to store retrieved data. The RET Command takes data from the Data Base and puts it in the Data Buffer starting at the top of the Buffer. The RETA Commands takes data from the Data Base and adds it to the Buffer, starting at the end of the data entered by the previous RET or RETA Command. Both Commands can retrieve all or part of a Data Base vector or matrix. When this is done, the Buffer is filled in the order retrieved.

In the example of Figure 3.4, the first RET Command:

RET 2 8 0 12 2 1 1

retrieves data from Block 2, Item 8 (of Site 13, Case 37). The form of the data to be retrieved from the TAPS Data Base is to be Form 0 and the Units are to be Units 12.

If the units of the data stored in the TAPS Data Base differ from the desired units, then, when the data are retrieved, they are converted to the desired units specified in the RET Command. Thus, in the example, if the Units Number stored in 23, which is, for example, meters, and if the Units No. 12 is feet, then the stored numbers are multiplied by 3.280840, the conversion from meters to feet. Similarly if the Form of the Data stored in the TAPS Data Base differs from the desired Form, then, when the data are retrieved, they are converted to the desired Form specified. This is done by a special Form Conversion subroutine. There must be such a routine for each specific conversion needed.

The RET Command specifies a Default Option which is exercised either if the data is missing from the Data Base, or if the specific form conversion subroutine is not found. In the example Command, the "2" represents the Default Option. This Default Option can indicate that the DARES run should be aborted, or that a special action should be taken (e.g., if a data value cannot be found, use 0.0).

The last two numbers of the RET Command indicate the subscripts of the first and last values of a stored vector which are to be retrieved. In the example command, the retrieval is for a single value and thus these two subscripts are "1".

Assuming that Site 13, Case 37, Block 2, Item 8 contains:

S
0
23
7.5E2

(i.e., a single number 7.5E2, in Form No. 0, Units No. 23), then the RET Command retrieves the number, converts it to Units 12 (by multiplying by 3.280840), and stores it at the top of the Data Buffer as 2.46063E3.

The RETA Command functions similarly to the RET Command except that it adds to the Data Buffer rather than starting at the top. The first RETA Command of the example of Figure 3-4 will retrieve data from Site 13, Case 37, Block 1, Item 1. This data will be added to the Data Buffer directly after the 2.46063E3 that was stored there by the preceding RET Command.

3.4.4 Text Card Copying (CARDCOPY)

The CARDCOPY Command is used to create a model program input card by copying a Text Card. The next Text Card after the one most recently used by a CARDCOPY or CARDOVER Command is used. (The first CARDCOPY or CARDOVER Command uses the first Text Card.)

The CARDCOPY Command is used if the model input requires titles or control cards. For example, the first CARDCOPY Command of the example in Figure 3-4 will create a copy of the first model input card as shown in Figure 3-5. The last CARDCOPY Command of the example is used to create a copy of the last Text Card (END) which is assumed to be a control card for the model program.

The CARDCOPY Command can also be used to set variables to values independent of the data in the Data Base. Thus, the next-to-last CARDCOPY Command of the example creates an input card of "J=14", setting the model program's value of J without reference to the Data Base.

3.4.5 Data Card Creation (CARDDATA, APNDDATA)

The CARDDATA Command creates a model program input card from data taken, in order, from the Data Buffer. The APNDDATA Command appends data to a model program input card that previously was created by a CARDDATA Command. Each command must specify (1) the mode, real or integer, in which the data will be placed on the card, (2) the number of data values to be placed on the card by the Command, and (3) the format of the placement of the data. Each command can specify data of only one mode. Therefore, if data of both modes are to be on the same model program input card, both a CARDDATA Command and an APNDDATA Command are used, one for each mode.

In the example of Figure 3-4, the last CARDDATA Command indicates that a card is to be created with one real value on it, and the format is to be F5.1. The following APNDDATA Command appends one integer mode value to the card in form 14 (with the 6X allowing room so the previous piece of data is not overwritten). The result is the third-from-last line of Figure 3-5.

3.4.6 Text Card Overwriting (CARDOVER, APNDOVER)

The CARDOVER Command is used to create a model program input card by copying a Text Card and then overwriting designated fields with data taken, in order, from the Data Buffer. The next Text Card after the one most recently used by a CARDCOPY or CARDOVER Command is used. (The first CARDCOPY or CARDOVER Command uses the first Text Card.)

The APNDOVER Command is used to take data, in order, from the Data Buffer and append them to a model program input card which was previously created by a CARDOVER Command. This is done by overwriting the data upon designated fields of the card.

Each of these commands must specify (1) the mode, real or integer, in which the data will be overwritten on the card, (2) the number of data values to be placed on the card by the Command, and (3) the format of the placement of the overwritten data. As with the CARDDATA and APNDDATA Commands, each command can specify data of only one mode. Therefore, if data of both modes is to be on the same model program input card, both a CARDOVER and an APNDOVER Command are used, one for each mode.

The use of the CARDOVER and APNDOVER Command is illustrated in the example of Figure 3-4. The CARDOVER Command takes two data values from the Data Buffer and overwrites them in integer mode on a copy of the next Text Card (in this case the third Text Card). Spaces are skipped in the overwriting to allow the data to be placed where desired. The resulting card is:

```
H=      2461 N=      28 RH=17.631 EM=
```

Note that the overwriting commands place the data where desired, and in the desired form. For example, the first value in the Buffer was 2.46063E3 as discussed in Section 3.4.3. CARDOVER converted this value to I10 format and thus used "2461" to overwrite. Note also that the overwriting Commands can be used to set variables to values independent of the data in the Data Base, as here RH is set to 17.631.

The APNDOVER Command of Figure 3-4 will take the card created by the CARDOVER Command and overwrite data upon it, this time in real mode. A single F5.1 value is overwritten, and the resulting card (as shown in Figure 3-5) is:

```
H=      2461 N=      28 RH=17.361 EM= 29.3
```

3.5 SMOG

Each model program produces its outputs in a form unique to the particular program. Therefore, in order to be able to analyze the model's results and to compare them with measured values and with the results of other models, the outputs of all the model programs must be put into a standard form. This function is accomplished by the Standard Model Output Generator, SMOG.

SMOG must be able to perform two operations:

a. Get the calculated values from the model program.

b. Store these values in Standard Form in the TAPS Data Base.

As with DARES, it is desirable that SMOG require as few changes to the original model program as possible - preferably none.

There are two ways a model program can produce its outputs: (1) it can produce a set of stored values which contain the results, or (2) it can calculate some values, print or otherwise use the values, and then destroy or overwrite them. SMOG must be able to handle model programs using either approach.

For models using the first approach, a call to SMOG can be made immediately after the termination of the model run and SMOG can get the calculated results directly from the stored values of the model program. This requires no change at all to the model program. For models using the second approach, slight changes must be made to the model program to allow for the storage of the calculated results. But even in this case, SMOG's major operation will take place after the model run, and the model will be essentially intact, and estimates of model run time will not be greatly affected.

Once SMOG has the information as to where the calculated results are stored, SMOG must get them, convert them from whatever form they are stored in to TAPS Standard Form, and place them in the TAPS Data Base, so they can be accessed by DIMOTE.

These operations are similar to those performed by FORMAN and similar techniques are used. Input commands to SMOG indicate: (1) the location where the calculated results are stored, (2) the format in which the results are stored, and (3) the model, and the site and case for which the results were obtained. The SMOG Processor will retrieve the results, and store them in Level IV of the TAPS Data Base, utilizing FORMAN methods. The SMOG Processor will also produce a Results Directory which will have an entry for each model number-site number-case number combination for which results have been obtained. The entry will contain a pointer to the location where those results are stored in Level IV. Using this Results Directory, DIMOTE will be able to find such items as: (1) the sites and cases which have been run for a given model, (2) the models that have been run for a site and case, (3) all the calculated results for a given model, (4) all the calculated results for a given site and case for various models, etc.

3.6 DIMOTE

The fourth and final part of the Transportation Air Pollution Studies Systems is the DIspersion MOdel TEster, DIMOTE. DIMOTE's function is to take the outputs of the model program runs (which have been stored in the TAPS Data Base by SMOG) and compare them with each other and with the measured values (which were stored with the other incoming data for each site and case by FORMAN). DIMOTE will use several statistical tests for these comparisons, and will then produce tables and graphs comparing and evaluating the different model programs. These results will be used in the production of reports on model program evaluation, which is the principal objective of this project.

DIMOTE will employ two major types of statistical tests: Loss Function Tests and Natural Histogram Tests. A Loss Function can be written in the form:

$$L_m = \frac{1}{\sum_{d=1}^{D} N_d} \sum_{d=1}^{D} K_d \sum_{i=1}^{N_d} F\,[M(d,i),\ C_m(d,i)]$$

where:

L_m = the loss incurred using Model m.
D = the number of data-sets being used in the evaluation
N_d = the number of elements in the d-th data-set
M(d,i) = the i-th measurement of pollution concentration made for the d-th data-set
$C_m(d,i)$ = the value calculated by model m corresponding to the i-th measure of pollution concentration made for the d-th data-set.
F = a function of the disparity between M(d,i) and $C_m(d,i)$

Thus L_m is a measure of the differences between the measured values and the model calculated values, with K_d's allowing for selected data-sets to be given more weight. Typical F functions would be

a. $F = |M-C_m|$

b. $F = (M-C_m)^2$

c. $F = \beta m^{\alpha} (\log C_m - \log M)^2$

where $\beta = \beta_1 \quad C_m > M$

$\beta_2 \quad C_m < M$

DIMOTE will present tables of several different loss functions, at the user's option, comparing the various models.

The second major type of statistical test employed by DIMOTE will be the Natural Histogram. The Natural Histogram is a new, sophisticated statistical technique. It allows an estimate to be made of

$$F_m(x,y) = Pr(M > x \mid C_m = x)$$

i.e., the probability that the measured or actual value is greater than a certain value (e.g., a standard) given that the calculated value for model m is a certain value. This type of conditional

probability is useful in evaluating models since it allows estimates to be made as to whether a model will work well near a given standard. Thus, some models may perform better for low pollution cases, while others may work better for higher pollution cases. In addition, the Natural Histogram is a valuable tool in using a model program since it allows an estimate to be made of the probability that the actual pollution concentration will exceed the critical value, given that the model program predicted a certain value. DIMOTE will produce tables of $F_m(x,y)$ for all models and for selected values of x and y.

In addition to the above, DIMOTE will produce several types of graphs, for example:

a. Scatter plots of calculated values vs measured values.

b. Frequency distribution plots.

c. Graphs showing results of Loss Function and Natural Histogram tests.

d. Graphs showing other statistics of interest.

All of the aforementioned DIMOTE functions are handled by the DIMOTE Routine. The DIMOTE Routine uses the Results Directory which is produced by SMOG to access values calculated by the model programs. It accesses measured values directly from the site and case entries in the TAPS Data Base. DIMOTE then calls user specified subroutines from a DIMOTE Subroutine Library to perform the statistics and produce the output tables and graphs.

4. USER'S GUIDES

The following are guides for the use of the FORmat MANipulator (FORMAN) and the DAta REtrieval System (DARES). The function, usage, and format of each command is given.

Note that in the Instruction Format descriptions:

a. The character % is used to represent a blank.

b. All fields are right justified except when specified otherwise.

4.1 FORMAN USER'S GUIDE

For a given data-set, a set of FORMAN commands is written specifying the method for converting the data-set to TAPS Standard Format and for storage in the TAPS Data Base. These commands, along with the data-set, form the input to the FORMAN Processor which accomplishes the conversion and storage.

4.1.1 FORMAN Commands

There are five FORMAN Commands:

COMMAND	FUNCTION
NSIT	Create a new Site entry.
USIT	Update a Site entry.
NCAS	Create a new Case entry and its associated Parameter and Data entries.
UCAS	Update a Case entry and its associated Parameter and Data entries.
END	End of FORMAN Commands.

Each command is composed of one or more instructions. Each instruction comprises one card (or card-image). The first instruction of each Command has the Command name in the first four columns of the card.

4.1.1.1 NSIT Command

ACTION: Create a Level I entry in the TAPS Data Base.

USAGE: Command can be used unless there already exists a Level I entry with the same site number as indicated in the command. If a Level I entry already exists, an error message is produced.

FORM:
1. An "NSIT" instruction
2. Site entry instruction No. 1
3. Site entry instruction No. 2
4. A set of Site Description instructions

FORMAT OF INSTRUCTIONS:

INSTRUCTION	COLUMNS	CONTENTS	MEANING
NSIT	1-4	NSIT	Command call
	5-8	integer	Site number
Site Entry No. 1	1-4	HWY% or APT%	Highway or Airport
	5-36	alphanumerics	Site identification
	37-40	integer	Number of cases with wind speed < 5
	41-44	integer	Number of cases with wind speed 5 to 10
	45-48	integer	Number of cases with wind speed 10 to 20
	49-52	integer	Number of cases with wind speed > 20
	53-56	integer	Number of cases with stability A
	57-60	integer	Number of cases with stability B
	61-64	integer	Number of cases with stability C
	65-68	integer	Number of cases with stability D
	69-72	integer	Number of cases with stability E
	73-76	integer	Number of cases with stability F

INSTRUCTION	COLUMNS	CONTENTS	MEANING
Site-Entry No. 2	1-4	integer	Number of Cases measuring CO
	5-8	integer	Number of Cases measuring SO_2
	9-12	integer	Number of Cases measuring Particulates
	13-16	integer	Number of Cases measuring NO_x
	17-20	integer	Number of Cases measuring Lead
	21-24	integer	Number of Cases measuring Total Hydrocarbons
	25-28	integer	Number of Cases measuring Reactive Hydrocarbons
	29-32	integer	Number of Cases measuring aldehydes
	33-36	integer (1 to 4)	Road or airport type
	37-40	integer	Number of Site Description instructions to follow
Site Description	1-80	one number, left justified	Each indicates one number from site description, in order, as described in Section 2.5.

4.1.1.2 USIT

ACTION: Update a Level I entry in the TAPS Data Base.

USAGE: Command can be used only if there already exists a Level I entry with the same Site number as indicated in the command. If this Level I entry does not exist, an error message is produced.

FORM:
1. A "USIT" instruction.
2. Site entry instruction No. 1.
3. Site entry instruction No. 2.
4. A set of Site Description instructions.

FORMAT OF INSTRUCTIONS:

INSTRUCTION	COLUMNS	CONTENTS	MEANING
USIT	1-4	USIT	Command call
	5-8	integer	Site number
Site-Entry No. 1 Site Entry No. 2 Site Description	As for NSIT		

4.1.1.3 NCAS Command

ACTION: Create a Level II entry and associated Level III and Level IV entries in the TAPS Data Base.

USAGE: Command can be used only if there already exists a Level I entry with the same Site number as indicated in the command, and if there does not exist a Level II entry for the Site with the same Case number as indicated in the command, and if there does not exist a Level II entry for the Site with the same Case number as indicated in the command. Otherwise, an error message is produced.

FORM:
1. An "NCAS" instruction.
2. A set of data cards containing the data-set.
3. A "FORMAN" instruction.
4. A set of Insert instructions (indicating which data to extract from the data-set and where to insert them in the TAPS Data Base).
5. A header instruction after each Insert instruction that references a Parameter (by Block and Item number) for the first time. There is a different Header instruction for each possible header.

FORMAT OF INSTRUCTIONS:

INSTRUCTION	COLUMNS	CONTENTS	MEANING
NCAS	1-4	NCAS	Command call
	5-8	integer	Site number
	9-12	integer	Case number
Data Card	1-80	any	Data as received
FORMAN	1-8	FORMAN%%	Indicates beginning of Insert Instructions.
Insert	1-3	integer (1 to 3 or -1 to -99)	Block number of parameter if positive. If negative, number of Text Cards to be skipped, (in which case card columns 4-80 should be blank).
	4-6	integer	Item number for Parameter.
	7-8	integer or %%	Integer is header number.
	9	* or %	* if command's reference is to next data-card (or first card). % if reference is to present data card.
	10-11	integer	Number of values per data-card.
	12-15	integer	First subscript for storage in vector form.
	16-19	integer	Last subscript for storage in vector form (1 if scalar)
	20	R or %	R indicates data is in the form K*M. (This must be the only data read by this insert command).
	21-24	Int% or Real	Signifies if data to be stored is integer or real mode.
	25-80	FORTRAN Format left justified	Format on data card of data to be read. The nX format is used to cover up any undesired characters on card, indicating the K* part of K*M data. Format is enclosed in parentheses, as is standard.

INSTRUCTION	COLUMNS	CONTENTS	MEANING
Header	1-4	integer	Form No.
	5-8	number	Second entry in header.
	9-12	number	Third entry in header. if required by header.
	13-16	number	Fourth entry in header if required by header.
	etc.	etc.	etc.

4.1.1.4 UCAS Command

ACTION: Update the Level III and Level IV entries associated with a Level II entry in the TAPS Data Base.

USAGE: Command can be used only if there already exists a Level I entry and an associated Level II entry with the same Site and Case numbers as indicated in the command. Otherwise, an error message is produced.

FORM:
1. A "UCAS" instruction.
2. A set of data cards.
3. A "FORMAN" instruction.
4. A set of Insert instructions.
5. A header instruction after each.

Insert instruction that references a Parameter for the first time.

FORMAT OF INSTRUCTIONS:

INSTRUCTION	COLUMNS	CONTENTS	MEANING
UCAS	1-4	UCAS	Command call
	5-8	integer	Site number
	9-12	integer	Case number
Data card FORMAN Insert	As for NCAS		

4.1.1.5 END Command

ACTION: Terminates the FORMAN Processor run.

USAGE: Last command of a command set.

FORM: An END instruction.

FORMAT OF INSTRUCTIONS:

INSTRUCTION	COLUMNS	CONTENTS	MEANING
END	1-4	END%	Command call

4.2 DARES USER'S GUIDE

For a given model program, a set of DARES commands is written specifying the format in which model input card-images should be formed in order to run the model program. This format includes the specification of the data required by the model, (which must be retrieved from the TAPS Data Base). The DARES commands are input to the DARES Processor, which retrieves the data from a specified Site and Case in the Data Base and forms the input card-images for the model program.

4.2.1 DARES Commands

A DARES Command set is comprised of two parts:

1. A set of Text Cards.
2. A set of DARES Commands.

The Text Cards contain text for use with certain DARES Commands described below. Each of these Commands references the present Text Card or the next Text Card, with the first Text Card being considered the first "next" card.

A data buffer is used by DARES to store retrieved data. Some DARES Commands are used to retrieve data from the TAPS Data Base and put them in the Data Buffer, and other DARES Commands are used to take the data from the Data Buffer and form the cards (or card-images) which will be used as the input to the model program.

There are nine DARES Commands:

COMMAND	FUNCTION
DARE	Specify Site and Case numbers from which data shall be retrieved.
RET	Retrieve data from the TAPS Data Base and place them at the top of the Data Buffer.
RETA	Retrieve data from the TAPS Data Base and add them to the Data Buffer.
CARDCOPY	Copy the next Text Card.
CARDDATA	Create a card containing data from the Data Buffer.
CARDOVER	Create a card by overwriting designated fields of a copy of the next Text Card with data from the Data Buffer.
APNDDATA	Append data from the Data Buffer to a previously created card in designated fields.
APNDOVER	Append data from the Data Buffer to a previously created card in designated fields, and overwrite designated fields with fields from the present Text Card.
END	End of commands for the present Site and Case numbers.

All DARES commands comprise one card (or card-image), with the exception of the DARE command, which is two cards.

4.2.1.1 DARE Command

ACTION: Specify Site number and Case number from which RET and RETA Commands will retrieve data.

USAGE: The first Command directly following the Text-Cards. If another Site and Case is required in the same DARES run, the DARE Command is used again, directly following an END Command.

FORM: 1. "DARE"
2. Site and Case

FORMAT OF INSTRUCTIONS:

INSTRUCTION	COLUMNS	CONTENTS	MEANING
DARE	1-4	DARE	Command
Site and Case	1-4	integer	Site Number
	5-8	integer	Case Number

4.2.1.2 RET Command

ACTION: Retrieve data from the Site and Case in the TAPS Data Base that was specified by the previous DARE command. Place the data in the Data Buffer, in order, starting at the top of the Buffer.

USAGE: Command can be used anywhere between a DARE Command and an END Command.

FORMAT OF COMMAND:

COLUMNS	CONTENTS	MEANING
	RET%	Command call
	1,2, or 3	Level III Block number of data to be retrieved.
	integer	Level III Item number of data to be retrieved.
	integer	Form number of data to be retrieved. (If Form number of stored data is different than this, DARES attempts to convert to Form specified here.)
	integer	Units number of data to be retrieved. (If Units number stored is different than this, number retrieved data is converted to units specified here.)
17-18	integer	Default Option number. If Form conversion algorithm is not known to DARES, or if required data is missing, DARES takes Default Option indicated (0-Abort).

COLUMNS	CONTENTS	MEANING
19-22	integer	Subscript of first data value to be retrieved from Level IV.
23-26	integer	Subscript of last data value to be retrieved from Level IV. (1 if single number.)

4.2.1.3 RETA Command

ACTION: Retrieve data from the Site and Case in the TAPS Data Base that was specified by the previous DARE Command. Add the data to the Data Buffer, in order, starting after the data entered by the last RET or RETA Command.

USAGE: Command should follow a RET Command or another RETA Command, with possible intervening CARDDATA, CARDOVER, CARDCOPY, APNDDATA, or APNDOVER Commands.

FORMAT OF COMMAND:

COLUMNS	CONTENTS	MEANING
1-4	RETA	Command call
5-26	Same as for the RET Command	

4.2.1.4 CARDCOPY Command

ACTION: A model program input card is created by copying the next Text Card.

USAGE: Command can be used anywhere between a DARE Command and an END Command.

FORMAT OF COMMAND:

COLUMNS	CONTENT	MEANING
1-8	CARDCOPY	Command call

4.2.1.5 CARDDATA Command

ACTION: A model program input card is created from data taken, in order, from the Data Buffer.

USAGE: This Command should follow a RET or RETA Command, with possible intervening CARDDATA, CARDOVER, CARDCOPY, APNDDATA, or APNDOVER Commands

FORMAT OF COMMAND:

COLUMNS	CONTENT	MEANING
1-8	CARDDATA	Command call
9-12	INT% or REAL	Data mode. Signifies if data to be put on created card is integer or real mode. Modes cannot be mixed on a single Command card. (If mixed modes are needed, use APNDDATA Commands following the CARDDATA Command, still keeping one mode per command.)
13-16	integer	Number of data values for the Data Buffer to be put on the created card.
17-76	FORTRAN Format, left justified	Format of the data placement. The format is within parentheses, as is conventional.

4.2.1.6 CARDOVER Command

ACTION: A model program input card is created by overwriting designated fields of a copy of the next Text Card with data taken, in order, from the Data Buffer.

USAGE: This Command should follow a RET or RETA Command, with possible intervening CARDDATA, CARDOVER, CARDCOPY, APNDDATA, OR APNDOVER Commands

FORMAT OF COMMAND:

COLUMNS	CONTENT	MEANING
1-8	CARDOVER	Command call
9-12	INT% or REAL	Signifies if data to be put on created card is integer or real mode. Modes cannot be mixed on a single Command Card. (If mixed modes

COLUMNS	CONTENT	MEANING
		are needed, use APNDOVER Commands following the CARDOVER Command, still keeping one mode in Command.)
13-16	integer	Number of data values to be put on the created card.
17-76	FORMAN Format, left justified	Format of the data placement to be overwritten upon a copy of the next Text Card. Only those card columns covered by the data will be changed on the Text Card. The nX format is used to allow text which is before or between data values to be unaltered. The format is within parentheses, as is conventional.

4.2.1.7 APNDDATA Command

ACTION: A model program input card previously created by a CARDDATA Command is appended with data taken, in order, from the Data Buffer.

USAGE: This Command should follow a CARDDATA Command or another APNDDATA Command, with possible intervening RET or RETA Commands.

FORMAT OF COMMAND:

COLUMNS	CONTENT	MEANING
1-8	APNDDATA	Command call
9-12	INT% or REAL	Data mode (integer or real). Modes cannot be mixed in a single Command.
13-16	integer	Number of data values from the Data Buffer to be put on the created card.
17-76	FORMAN Format, left justified	Format of the data placement, with nX format used to allow for areas of the card which have been created by previous CARDDATA or APNDDATA Commands.

4.2.1.8 APNDOVER Command

ACTION: A model program input card previously created by a CARDOVER Command is appended with data taken,

in order, from the Data Buffer, and overwritten with designated fields from the present Text Card.

USAGE: This Command should follow a CARDOVER Command or another APNDOVER Command, with possible intervening RET and RETA Commands.

FORMAT OF COMMAND

COLUMNS	CONTENT	MEANING
1-8	APNDOVER	Command call
9-12	INT% or REAL	Data mode (integer or real). Modes cannot be mixed in a single command.
13-16	integer	Number of data values from the Data Buffer to be put on created card.
17-76	FORMAN Format, left justified	Format of the data placement to be overwritten upon the present Text Card (which has already been modified by a preceeding CARDOVER and possibly APNDOVER Commands). Only those card columns covered by the data will be changed on the Text Card.

4.2.1.9 END Command

ACTION: Signifies the end of the set of commands for the present Site and Case number.

USAGE: The last Command of a Command set for each Site and Case number. As such, it will be followed directly by a DARE Command, or it will be the last Command for the run.

FORMAT OF COMMAND:

COLUMNS	CONTENTS	MEANING
1-4	END%	Command call

5. SAMPLE TAPS USAGE

This section contains examples of both FORMAN and DARES usage. It should be noted that the FORMAN output appears only to show that the FORMAN input was read and stored correctly.

5.1 DATA STORAGE USING FORMAN

```
COMMAND=NSIT
HWY    I-70    BALTIMORE MARYLAND          1   0   0   0   0   0   1   0   0   0
   1   0   0   0   0   0   0   0   2   2   2   0
```

```
COMMAND=USIT
```

```
COMMAND=NCAS
 NEXT AVAILABLE UNUSED RECORD ON LEVEL3 =     2
HWY   I-70   BALTIMORE MARYLAND          1   0   0   0   0   0   1   0   0   0
   1   0   0   0   0   0   0   0   2   2   2   2   1
NEXT AVAILABLE UNUSED RECORD ON LEVEL2 =     3
LEVEL2 - POINTERS TO LEVEL3
   2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
DATA VALUES COMING IN
   15       12.2 18.7  6.1
3 100    200    300
17.5     6.7  2.9  10000.7
FORMAN  COMMANDS
2  5 2* 3    1    3 REAL(10X,3F5.1)
LEVEL3 - POINTERS TO LEVEL4
EMISSION POINTER BLOCK
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
METEOROLOGICAL POINTER BLOCK
   0   0   0   0  32   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
GENERAL POINTER BLOCK
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
HEADER NUMBER
   2
HEADER DETAILS
   3   5   5
DATA VALUES ON LEVEL4 - THAT IS ON THE STANDARD DATA BASE
 0.12200000E 02 0.18699997E 02 0.60999994F 01 0.99999994E-38 0.99999994E-38
NEXT AVAILABLE UNUSED RECORD ON LEVEL4 =    35
FORMAN  COMMANDS
1  3 1* 1    1    1 INT (2X,I3)
LEVEL3 - POINTERS TO LEVEL4
EMISSION POINTER BLOCK
   0   0  35   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
METEOROLOGICAL POINTER BLOCK
   0   0   0   0  32   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
GENERAL POINTER BLOCK
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
HEADER NUMBER
   1
HEADER DETAILS
   1   2
DATA VALUES ON LEVEL4 - THAT IS ON THE STANDARD DATA BASE
 0.10000000E 03
NEXT AVAILABLE UNUSED RECORD ON LEVEL4 =    38
FORMAN  COMMANDS
2 12 1* 1    1    1 REAL(F4.1)
LEVEL3 - POINTERS TO LEVEL4
EMISSION POINTER BLOCK
   0   0  35   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
METEOROLOGICAL POINTER BLOCK
   0   0   0   0  32   0   0   0   0  38   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
GENERAL POINTER BLOCK
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
HEADER NUMBER
   1
HEADER DETAILS
   1   3
DATA VALUES ON LEVEL4 - THAT IS ON THE STANDARD DATA BASE
 0.17500000E 02
NEXT AVAILABLE UNUSED RECORD ON LEVEL4 =    41
FORMAN  COMMANDS
3 30 2  2   1    2 REAL(8X,F3.1,2X,F3.1)
LEVEL3 - POINTERS TO LEVEL4
EMISSION POINTER BLOCK
   0   0  35   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
METEOROLOGICAL POINTER BLOCK
   0   0   0   0  32   0   0   0   0   0   0  38   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
GENERAL POINTER BLOCK
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  41
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
HEADER NUMBER
   2
HEADER DETAILS
   3   5   5
DATA VALUES ON LEVEL4 - THAT IS ON THE STANDARD DATA BASE
 0.66999998E 01 0.28999996E 01 0.99999994E-38 0.99999994E-38 0.99999994E-38
NEXT AVAILABLE UNUSED RECORD ON LEVEL4 =    44
FORMAN  COMMANDS
1 17 1  1   1    1 REAL(18X,F7.1)
LEVEL3 - POINTERS TO LEVEL4
EMISSION POINTER BLOCK
   0   0  35   0   0   0   0   0   0   0   0   0   0   0   0   0  44   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
METEOROLOGICAL POINTER BLOCK
   0   0   0   0  32   0   0   0   0   0   0  38   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
GENERAL POINTER BLOCK
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  41
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

```
HEADER NUMBER
   1
HEADER DETAILS
   1   6
DATA VALUES ON LEVEL4 - THAT IS ON THE STANDARD DATA BASE
 0.10000699E 05
NEXT AVAILABLE UNUSED RECORD ON LEVEL4 =   47

COMMAND=NSIT
APT MOFFAT,TALBOT,SHAH AIRPORT, KHL    1   0   0   0   0   0   1   0   0   0
   1   0   0   0   0   0   0   0   1   3  47   0

COMMAND=USIT
COMMAND=NCAS
 NEXT AVAILABLE UNUSED RECORD ON LEVEL3 =     5
APT MOFFAT,TALBOT,SHAH AIRPORT, KHL    1   0   0   0   0   1   0   0   0
   1   0   0   0   0   0   0   0   1   3  47   1
NEXT AVAILABLE UNUSED RECORD ON LEVEL2 =    4
LEVEL2 - POINTERS TO LEVEL3
   5   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
DATA VALUES COMING IN
 2.75,      0.668,      1.17,       0.289,      0.0695,     0.346,
 AEM(1,1,1)=37500.,     3280.,      4360.,      1400.,      121.,
  201    2055     6014         1     512     50000
 487.,  328.,  4*0.0,  582.,
FORMAN  COMMANDS
1  3 2* 3   1   3 REAL(F5.2,7X,F5.3,7X,F4.2)
LEVEL3 - POINTERS TO LEVEL4
EMISSION POINTER BLOCK
   0   0  71   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
METEOROLOGICAL POINTER BLOCK
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
GENERAL POINTER BLOCK
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
HEADER NUMBER
   2
HEADER DETAILS
   1  12  11
DATA VALUES ON LEVEL4 - THAT IS ON THE STANDARD DATA BASE
 0.27500000E 01 0.66799998E 00 0.11699991E 01 0.99999994E-38 0.99999994E-38 0.99999994E-38 0.99999994E-38 0.99999994E-38
 0.99999994E-38 0.99999994E-38 0.99999994E-38
NEXT AVAILABLE UNUSED RECORD ON LEVEL4 =   74
FORMAN  COMMANDS
2  7 2* 2   1   2 REAL(24X,F5.0,7X,F5.0)
LEVEL3 - POINTERS TO LEVEL4
EMISSION POINTER BLOCK
   0   0  71   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
METEOROLOGICAL POINTER BLOCK
   0   0   0   0   0   0  74   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
GENERAL POINTER BLOCK
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
HEADER NUMBER
   2
HEADER DETAILS
   1  13   4
DATA VALUES ON LEVEL4 - THAT IS ON THE STANDARD DATA BASE
 0.32800000E 04 0.43600000E 04 0.99999994E-38 0.99999994E-38
NEXT AVAILABLE UNUSED RECORD ON LEVEL4 =   77
FORMAN  COMMANDS
3  1 1  1   1   1 REAL(60X,F4.0)
LEVEL3 - POINTERS TO LEVEL4
EMISSION POINTER BLOCK
   0   0  71   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
METEOROLOGICAL POINTER BLOCK
   0   0   0   0   0   0  74   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
GENERAL POINTER BLOCK
  77   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
HEADER NUMBER
   1
HEADER DETAILS
  11  12
DATA VALUES ON LEVEL4 - THAT IS ON THE STANDARD DATA BASE
 0.12100000E 03
NEXT AVAILABLE UNUSED RECORD ON LEVEL4 =   80
FORMAN  COMMANDS
3 17 2* 3   2   4 INT (I5,13X,I4,14X,I3)
LEVEL3 - POINTERS TO LEVEL4
EMISSION POINTER BLOCK
   0   0  71   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
METEOROLOGICAL POINTER BLOCK
   0   0   0   0   0   0  74   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
GENERAL POINTER BLOCK
  77   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  80   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
HEADER NUMBER
   2
HEADER DETAILS
  10   3  20
```

```
DATA VALUES ON LEVEL4 - THAT IS ON THE STANDARD DATA BASE
 0.99999994E-38 0.20100000E 03 0.60140000E 04 0.51200000E 03 0.99999994E-38 0.99999994E-38 0.99999994E-38 0.99999994E-38
 0.99999994E-38 0.99999994E-38 0.99999994E-38 0.99999994E-38 0.99999994E-38 0.99999994E-38 0.99999994E-38 0.99999994E-38
 0.99999994E-38 0.99999994E-38 0.99999994E-38 0.99999994E-38
NEXT AVAILABLE UNUSED RECORD ON LEVEL4 =   83
FORMAN  COMMANDS
2 25 2* 4   1   4RREAL(17X, F5.1)
LEVEL3 - POINTERS TO LEVEL4
EMISSION POINTER BLOCK
   0   0  71   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
METEOROLOGICAL POINTER BLOCK
   0   0   0   0   0   0  74   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  83   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
GENERAL POINTER BLOCK
  77   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  80   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
HEADER NUMBER
   2
HEADER DETAILS
  15  10   7
DATA VALUES ON LEVEL4 - THAT IS ON THE STANDARD DATA BASE
 0.0            0.0            0.0            0.0            0.99999994E-38 0.99999994E-38 0.99999994E-38
NEXT AVAILABLE UNUSED RECORD ON LEVEL4 =   86

COMMAND=UCAS
DATA VALUES COMING IN
 2.75,      0.668,      1.17,      0.289,      0.0695,     0.346,
  201    2055    6014
FORMAN  COMMANDS
3 17 0* 2   4   5 REAL(48X,F5.3,6X,F6.4)
HEADER NUMBER
   2
HEADER DETAILS
  16   3  20
DATA VALUES ON LEVEL4 - THAT IS ON THE STANDARD DATA BASE
 0.99999994E-38 0.20100000E 03 0.60140000E 04 0.68999946E-01 0.34599996E 00 0.99999994E-38 0.99999994E-38 0.99999994E-38
 0.99999994E-38 0.99999994E-38 0.99999994E-38 0.99999994E-38 0.99999994E-38 0.99999994E-38 0.99999994E-38 0.99999994E-38
 0.99999994E-38 0.99999994E-38 0.99999994E-38 0.99999994E-38
NEXT AVAILABLE UNUSED RECORD ON LEVEL4 =   86
FORMAN  COMMANDS
1  3 0* 3   6   8 INT (I5,4X,I4,5X,I4)
HEADER NUMBER
   2
HEADER DETAILS
   1  12  11
DATA VALUES ON LEVEL4 - THAT IS ON THE STANDARD DATA BASE
 0.27500000E 01 0.66799998E 00 0.11699991E 01 0.99999994E-38 0.99999994E-38 0.20100000E 03 0.20550000E 04 0.60140000E 04
 0.99999994E-38 0.99999994E-38 0.99999994E-38 0.99999994E-38
NEXT AVAILABLE UNUSED RECORD ON LEVEL4 =   86
```

5.2 MODEL INPUT PREPARATION USING DARES

DARES INPUT

```
DARE
   1   1
CARDCOPY
CARDCOPY
RET 2  6    0    0 0    1    2
RETA1 35    0    0 0    1    1
RETA1 33    0    0 0    1    1
RETA3 26    0    0 0    1    1
RETA3 27    0    0 0    1    4
RETA2 11    0    0 0    1    1
RETA1 17    0    0 0    1    1
CARDOVERINT     2(7X,I4,10X,I4)
CARDCOPY
CARDOVERREAL    1(5X,F3.1)
APNDOVERINT     1(14X,I2)
CARDOVERINT     1(5X,I1)
CARDCOPY
CARDOVERINT     4(8X,I1,3(10X,I1))
CARDCOPY
CARDOVERINT     2(7X,I2,9X,I1)
END
/*
```

DARES OUTPUT

```
**********************************************************************
EPA DATA FOR SITE #1
DISTRIBUTION OF CARS
N1=    5200     N2=   4800
WIND SPEED AND DIRECTION
WS=  4.5 IWD= 42
NL=  8
POLLUTION CONCENTRATIONS AT RECEPTORS
IPOL1=  2   IPOL2=  1   IPOL3=  0   IPOL4=  0
EMISSION RATE AND STABILITY CLASS
IRATE= 25   ISTAB= 3
**********************************************************************
```

6. SUMMARY

The purpose of this report is to describe in detail the design, implementation, and use of the Transportation Air Pollution Studies (TAPS) System and Data Base. These were designed to allow the Transportation Systems Center to test, evaluate, and compare transportation air pollution dispersion models.

The TAPS Data Base, as discussed in Section 2 of this report, is designed to handle the storage of the large amounts of air pollution data that will be used by TSC in the testing of the model programs. The Data Base was designed for ease of retrieval and identification. It is quite flexible and open-ended. The Data Base can be expanded to accommodate new data types or data storage formats without affecting the data previously stored. These changes could be incorporated with a minimal programming effort.

The Data Base is a four-level structure. The top level describes the site of air pollution measurements. The second level indicates the different cases measured at the site. The third level indicates which input parameters comprise each case. The fourth level contains the actual data. The Data Base also contains results of previous model program runs for comparison purposes.

The TAPS System is designed to access the TAPS Data Base and use the data therein to test the air pollution dispersion model programs. The System, which is discussed in Section 3, consists of four parts: FORMAN, DARES, SMOG and DIMOTE. FORMAN reformats the incoming data. DARES retrieves data from the Data Base and produces input acceptable to a model program being tested. SMOG puts model program output into a standard form. DIMOTE performs statistical tests on model outputs to evaluate and compare the models.

The TAPS System has been implemented and is operational. Current listings of FORMAN, DARES, SMOG, AND DIMOTE are found in Section 7. As System usage increases, it is expected that the programs will be modified and updated. The latest listings of the TAPS System may be obtained by contacting the authors.

7. TAPS PROGRAM LISTINGS

```
C
C
C
C
C
C
C
C
C                                        FORMAN
C
C********************  THIS PROGRAM CREATES STANDARD DATA BASE  ******************
C
C                                        FORMAN
C
C
C
C
C
C
C
C
      DEFINE FILE  1 (105,250,L,ID1)
      DEFINE FILE 2(500,210,L,ID2)
      DEFINE FILE 3(500,210,L,ID3)
      DEFINE FILE 4(500,80,L,ID4)
      DEFINE FILE16(500,210,L,ID2A)
      DEFINE FILE 15(500,80,L,ID15)
      DIMENSION VUNITS(100),X(100),Y(100),Z(100)
      DIMENSION MASIND(35)
      DIMENSION IPOINT (51)
      DIMENSION  VVAR(100)
      DIMENSION MIND(35)
      DIMENSION IVAR(100)
      DIMENSION   VAR(100)
      DIMENSION DAT(20),FRMN(2)
      DIMENSION FMT(14 )
      INTEGER VARN ,REAL
      INTEGER SITDIS
      INTEGER END
      INTEGER CASREF (200)
      INTEGER REC4
      INTEGER POINT(51)
      INTEGER SITE
      INTEGER VUNITS
      INTEGER DUMY,BLCK,UNITS,FORM
      INTEGER BLKSYZ
      INTEGER DMYHDR
      INTEGER HEADER
      INTEGER CMD,USIT,UCAS
      INTEGER SITNUB,CASNUB,SITREF(100)
      LOGICAL START
      DATA REAL,INT/4HREAL,4HINT /
      DATA USIT/4HUSIT/
```

```
      DATA NSIT,NCAS,END/4HNSIT,  4HNCAS,4HEND /
      DATA NSITES,NREC2,NREC3/3,2,2/
      DATA NREC4/2/
      DATA ASTRIC /1H*/
      DATA FRMN/4HFORM,4HAN   /
      DATA START/.TRUE./
      DATA UCAS/4HUCAS/
      DATA R/1HR/
      INU=5
C
C
C
C**   FOLLOWING STATEMENTS INITIALIZE THE ASSOCIATED VARIABLE
C     FOR EACH OF THE 4 LEVELS AND A DUMMY FILE IN ORDER,
C     AT THE CONCLUSION OF EACH READ OR WRITE OPERATION,THE
C     ASSOCIATED VARIABLE IS SET TO A VALUE THAT POINTS TO THE
C     RECORD THAT IMMIDIATELY FOLLOWS THE LAST RECORD TRANSMITTED
C     UNLESS OTHERWISE THE ASSOCIATED VARIABLE IS SET TO SOME
C     OTHER VALUE AT A PARTICULAR INSTANT IN THE PROGRAM
C
C
C
      ID1=1
      ID2=1
      ID3=1
      ID4=1
      ID15=1
C
C**   UNIT NUMBER FOR LEVEL1 FILE
C
      LEV1=1
C
C**   UNIT NUMBER FOR LEVEL2 FILE
C
      LEV2=2
C
C**   UNIT NUMBER FOR LEVEL3 FILE
C
      LEV3=3
C
C**   UNIT NUMBER FOR LEVEL4 FILE
C
      LEV4=4
C
C**   UNIT NUMBER FOR DUMMY FILE USED WITHIN THE PROGRAM
C
      DUMY=15
C
C**   MAXIMUM NUMBER OF CASE POINTERS TO LEVEL3 IN A FIXED LENGTH
C     BLOCK
C
      MAXCAS=50
C
C**   SIZE OF THE FIXED LENGTH BLOCK,ONE MORE THAN THE MAXIMUM
C     NUMBER OF CASE POINTERS TO LEVEL3
```

```
C
      BLKSYZ=MAXCAS+1
      LEV2A=16
      DO 193 I=1,100
  193 SITREF(I)=0
      DO 199 I=1,200
  199 CASREF (I)=0
      WRITE(LEV1'1) NSITES,SITREF
      WRITE(LEV2'1) NREC2
      WRITE(LEV3'1) NREC3
      WRITE(LEV4'1) NREC4
C
C**   READ COMMAND AND SITE NUMBER OF A PARTICULAR CASE TO BE
C     PROCESSED
C
   50 READ(5,100,END=3000) CMD,SITNUB,CASNUB
  100 FORMAT(A4,2I4)
      WRITE(6,62) CMD
   62 FORMAT(1H1,'COMMAND=',A4)
C
C
C
C**   COMMAND IS ANY ONE OF THE FOLLOWING
C     - CREATING A NEW SITE 'NSIT'
C     - UPDATING A CREATED SITE 'USIT'
C     - CREATING A NEW CASE OF THE CREATED SITE 'NCAS'
C     - UPDATING LEVEL3 BLOCKS FOR DATA POINTERS TO LEVEL4 FOR
C       EMISSION,METEOROLOGICAL AND/OR GENERAL POINTERS FOR A
C       PARTICULAR CASE 'UCAS'
C
C
C
      IF(CMD.EQ.NSIT) GO TO 1000
      IF(CMD.EQ.NCAS) GO TO 2000
      IF(CMD.EQ.UCAS) GO TO 2003
      IF( CMD.EQ.USIT ) GO TO 1001
      IF(CMD.EQ.END) GO TO 3000
      WRITE(6,6)
    6 FORMAT(1X,'*************************  ERROR - CHECK DATA FOR THE C
     XOMMAND CARD *************************')
      STOP
C
C**   CREATING A NEW SITE
C
 1000 READ(LEV1'1) NSITES,SITREF
C
C**   READ THE NEXT AVAILABLE UNUSED RECORD ON LEVEL2 FILE
C
      READ(LEV2'1) NREC2
C
C**   READ THE NEXT AVAILABLE UNUSED RECORD ON LEVEL3 FILE
C
      READ(LEV3'1) NREC3
C
C
```

```
C**   READ MASTER INDEX FILE - AIRPORT OR HIGHWAY,WIND SPEED
C     GEOGRAPHICAL CONFIGURATION,POLLUTANTS,NUMBER OF SITE
C     DESCRIPTION ENTRIES, ETC.
C
C
      READ(5,700) (MASIND(I),I=1,35)
  700 FORMAT(   9A4,10I4/   12I4,4A4)
      NCARD=MASIND(29)
      ID4=NREC4
      NSAVR4=NREC4
      WRITE(LEV4'ID4) NCARD
      DO 7 I=1,NCARD
      READ(5,580) SITDIS
      WRITE(LEV4'ID4) SITDIS
    7 CONTINUE
      MASIND(30)=NREC4
      NREC4=ID4
      WRITE(LEV4'1) NREC4
      MASIND(29)=NREC2
      MASIND(31)=0
      IF(SITREF(SITNUB).NE.0) GO TO 191
      SITREF(SITNUB)=NSITES
      ID1=NSITES
      NSITES=NSITES+1
      WRITE(LEV1'ID1) MASIND
      ID2A = (SITNUB-1) * 5 + 1
      WRITE(LEV2A'ID2A) CASREF
      WRITE(6,63) MASIND
   63 FORMAT(1X,9A4,10I4/1X,12I4,4A4)
      WRITE(LEV1'1) NSITES,SITREF
C
C**   INITIALIZE LEVEL2 POINTERS
C
      DO 10 I=1,BLKSYZ
      POINT(I)=0
   10 CONTINUE
      ID2=NREC2
      WRITE(LEV2'ID2) POINT
      NREC2=ID2
      WRITE(LEV2'1) NREC2
      GO TO 50
C
C**   UPDATING A SITE
C
 1001 READ(LEV1'1)NSITES,SITREF
      IF(SITREF(SITNUB).EQ.0) GO TO 197
      READ (5,700) M  IND
      NCARD=  MIND(29)
      ID4=NSAVR4
      WRITE (LEV4'ID4) NCARD
      DO 9001 I=1,NCARD
      READ (5,700) SITDIS
      WRITE (LEV4'ID4) SITDIS
 9001 CONTINUE
      ID1=SITREF(SITNUB)
```

```
      READ (LEV1'ID1) MASIND
      ID1=SITREF(SITNUB)
      MIND(29)=MASIND(29)
      MIND(30)=MASIND(30)
      MIND(31)=MASIND(31)
      WRITE(LEV1'ID1)   MIND
      GO TO 50
C
C**   CREATING A NEW CASE FOR THE CREATED SITE
C
 2000 READ(LEV3'1) NREC3
      ID2A = (SITNUB-1) * 5 + 1
      READ(LEV2A'ID2A) CASREF
      WRITE(6,22) NREC3
   22 FORMAT(1X,' NEXT AVAILABLE UNUSED RECORD ON LEVEL3 = ',I4)
      READ(LEV4'1) NREC4
      READ(LEV1'1) NSITES,SITREF
      ID1=SITREF(SITNUB)
      READ(LEV1'ID1) MASIND
      MASIND(31)=MASIND(31)+1
      NCASES=MASIND(31)
      CASREF(CASNUB)= NCASES
      ID2A = (SITNUB-1) * 5 + 1
      WRITE(LEV2A'ID2A) CASREF
      ID1=SITREF(SITNUB)
      WRITE(LEV1'ID1) MASIND
      WRITE(6,63) MASIND
C
C
C**   FOLLOWING    STATEMENTS DETERMINE  NUMBER OF BLOCKS REQUIRED
C     FOR LEVEL2 FILE , INITIALIZE  BLOCKS IF MORE THAN ONE NEEDED
C     FOR LEVEL2 FILE , READ AND WRITE ON LEVEL2 FILE APPROPRIATELY
C
C
      IPSV=MASIND(29)
      NCASES=MASIND(31)
      IDEEP=(NCASES-1)/50
      ICASE=MOD(NCASES-1,50)+1
      IF(IDEEP.EQ.0) GO TO 2020
      DO 2001 I=1,IDEEP
      READ(LEV2'IPSV)POINT
      READ(LEV2'1) NREC2
      WRITE(6,65) NREC2
   65 FORMAT(1X,'NEXT AVAILABLE UNUSED RECORD ON LEVEL2 = ',I4)
      WRITE(6,702)
  702 FORMAT(1X,'LEVEL2 - POINTERS TO LEVEL3')
      WRITE(6,66)POINT
   66 FORMAT(1X,30I4)
      IPSV= POINT(51)
 2001 CONTINUE
 2020 CONTINUE
      MAXI=(IDEEP*50)+50
      IF(NCASES.EQ.MAXI) GO TO 4000
      GO TO 4001
 4000 CONTINUE
```

```
      POINT(51)=NREC2
      WRITE(LEV2'IPSV) POINT
      NREC2=NREC2+ID2-IPSV
      DO 8004 I=1,BLKSYZ
 8004 POINT(I)=0
      WRITE(LEV2'ID2) POINT
      WRITE(LEV2'1) NREC2
 4001 READ(LEV2'IPSV) POINT
       POINT(ICASE)=NREC3
      WRITE(LEV2'IPSV) POINT
      READ(LEV2'1) NREC2
      WRITE(6,65) NREC2
      WRITE(6,702)
      WRITE(6,66)POINT
 2003 ID15=1
      READ(LEV4'1) NREC4
C
C**   FOLLOWING    STATEMENTS READ DATA FROM CARDS AND WRITE THEM
C     ON DUMMY UNIT TILL THE COMMAND IS 'FORMAN'
C
      WRITE(6,67)
   67 FORMAT(1X,'DATA VALUES COMING IN')
 2002 READ(INU,5) DAT
    5 FORMAT(20A4)
      IF(DAT(1).NE.FRMN(1)) GO TO 4
      IF(DAT(2).NE.FRMN(2)) GO TO 4
      ID15=1
      GO TO 2
    4 WRITE(6,68) DAT
   68 FORMAT(1X,20A4)
      WRITE(DUMY'ID15) DAT
      GO TO 2002
C
C
C**   READ TO WHAT BLOCK OF LEVEL3 THE DATA BELONG - EMISSION,
C     METEOROLOGICAL OR GENERAL,HEADER NUMBER,COMMAND TO READ THE
C     NEXT DATA CARD(FIRST CARD) OR READ THE PRESENT DATA CARD
C     FIRST AND LAST SUBSCRIPTS FOR STORAGE IN THE VECTOR FORM,
C     REPEAT COMMAND TO READ THE DATA IN THE FORM K*A,THE FORMAT
C     CONTROL WITH WHICH TO READ THE DATA VALUES
C
C
    2 WRITE(6,69)
   69 FORMAT(1X,'FORMAN  COMMANDS')
  183 READ(5,3) BLCK,ITEM,HEADER,CARD,NWORD,IST,IEND,REPT,VARN,FMT
      IF(BLCK.LE.0) GO TO 184
      GO TO 182
  184 ID15=ID15+IABS(BLCK)
      GO TO 183
  182 WRITE(6,93) BLCK,ITEM,HEADER,CARD,NWORD,IST,IEND,REPT,VARN,FMT
   93 FORMAT(1X,I1,I3,I2,A1,I2,2I4,A1,15A4)
      DMYHDR=HEADER
C
C
C**   IF HEADER IS ZERO, THE DATA BELONG TO THE CASE
```

```
C      ALREADY CREATED BEFORE , GO TO STATEMENT NUMBER 2060 AND FIND
C      THE HEADER NUMBER
C
C
      IF(HEADER.EQ.0) GO TO 2060
      ID4=NREC4
      ISAVE=NREC4
      WRITE(LEV4'ID4) HEADER
      NREC4=ID4
C
C
C**    IF HEADER IS NOT EQUAL TO ZERO, READ APPROPRIATE HEADER
C      DETAILS, FOR EXAMPLE, HEADER NUMBER 2 HAS FORM NUMBER,
C      UNIT NUMBER AND NUMBER OF VECTORS AND SO ON FOR OTHER HEADERS
C
C
      GO TO(500,501,502,503,504,505,506,507,508,509,510,511),HEADER
  500 READ(5,580)FORM,UNITS
      NELMNT=1
      GO TO 550
  501 READ(5,580)FORM,UNITS,NROWS
      NELMNT=NROWS
      GO TO 550
  502 READ(5,580)FORM,NROWS
      READ(5,590)(VUNITS(I),I=1,NROWS)
      NELMNT=NROWS
      GO TO 550
  503 READ(5,580)FORM,UNITS,NROWS,NCOLS
      NELMNT=NROWS*NCOLS
      GO TO 550
  504 READ(5,580)FORM,NROWS,NCOLS
      READ(5,590)(VUNITS(I),I=1,NCOLS)
      NELMNT=NROWS*NCOLS
      GO TO 550
  505 READ(5,580)FORM,UNITS,NX,NY,DX,DY,X0,Y0,NVAL
      READ(5,590)(VUNITS(I),I=1,NVAL)
      NELMNT=NX*NY
      GO TO 550
  506 READ(5,580)FORM,UNITS,NPTS,NVAL
      READ(5,590)(VUNITS(I),I=1,NVAL)
      READ(5,595)(X(I),Y(I),I=1,NPTS)
      NELMNT=NPTS
      GO TO 550
  507 READ(5,581)FORM,UNITS,NROWS,NCOLS,NLEVS
      NELMNT=NROWS*NCOLS*NLEVS
      GO TO 550
  508 READ(5,581)FORM,UNITS,NX,NY,NZ,DX,DY,DZ,X0,Y0,Z0,NVAL
      READ(5,590)(VUNITS(I),I=1,NVAL)
      NELMNT=NX*NY*NZ
      GO TO 550
  509 READ(5,580)FORM,UNITS,NPTS,NVAL
      READ(5,590)(VUNITS(I),I=1,NVAL)
      READ(5,596)(X(I),Y(I),Z(I),I=1,NPTS)
      NELMNT=NPTS
      GO TO 550
```

```
  510 GO TO 550
  511 GO TO 550
  550 CONTINUE
  580 FORMAT(    4I4,2E15.7/2E15.7,I4)
  590 FORMAT(    5I4)
  595 FORMAT(    4E15.7)
  581 FORMAT(    5I4,3E15.7/3E15.7,I4)
  596 FORMAT(    3E15.7)
  601 FORMAT(1X,7I4)
  606 FORMAT(1X,6E15.7)
    3 FORMAT(I3,I3,I2,A1,I2,2I4,A1,15A4)
      GO TO 999
C
C
C**   FOLLOWING    STATEMENTS FIND HEADER NUMBER FROM LEVEL4 FILE
C     FOR A CREATED CASE TO UPDATE THE LEVEL4 FILE(STANDARD DATA BASE)
C     KNOWING THE SITE NUMBER AND THE CASE NUMBER FOR THE CASE
C     TO BE UPDATED
C
C
 2060 ID1=SITREF(SITNUB)
      READ(LEV1'ID1) MASIND
      ID2A = (SITNUB-1) * 5 + 1
      READ(LEV2A'ID2A) CASREF
      IF (CASREF(CASNUB) .EQ. 0) GO TO  195
      NCASES = CASREF (CASNUB )
      IREC2=MASIND(29)
  111 READ(LEV2'IREC2) POINT
      IF(POINT(51).EQ.0) GO TO 407
      IREC2=POINT(51)
      GO TO 111
  407 IF(BLCK.EQ.1) GO TO 401
      IF(BLCK.EQ.2) GO TO 402
      IF(BLCK.EQ.3) GO TO 403
  401 IREC3=POINT(NCASES)
  421 READ(LEV3'IREC3) IPOINT
      IF(IPOINT(51).EQ.0) GO TO 405
      IREC3=IPOINT(51)
      GO TO 421
  402 IREC3=POINT(NCASES)
      IREC3=IREC3+IREC
  422 READ(LEV3'IREC3) IPOINT
      IF(IPOINT(51).EQ.0) GO TO 405
      IREC3=IPOINT(51)
      GO TO 422
  403 IREC3=POINT(NCASES)
      IREC3=IREC3+2*IREC
  423 READ(LEV3'IREC3) IPOINT
      IF(IPOINT(51).EQ.0) GO TO 405
      IREC3=IPOINT(51)
      GO TO 423
  405 IREC4=IPOINT(ITEM)
      READ(LEV4'IREC 4) HEADER
```

```
C
C
C**    TEST -IS THIS FIRST DATA CARD (READ IN BY READ STATEMENT NUMBERED
C      2)
C         YES - SET START =.FALSE.
C         NO  - TEST TO READ THE PRESENT RECORD OR THE NEXT RECORD
C
C
  999 IF(START) GO TO 51
   52 IF(CARD.EQ.ASTRIC) GO TO 53
      ID15=ID15-1
      GO TO 53
   51 START=.FALSE.
      GO TO 52
C
C**    TEST - DATA COMING IN IS IN THE FORM OF K*A
C
   53 IF(REPT.NE.R) GO TO 480
      IF(VARN .EQ. INT ) GO TO 944
      READ(DUMY'ID15,FMT) VAR(IST)
      NEX=IST+1
      DO 485 I=NEX,IEND
  485 VAR(I)=VAR(IST)
      GO TO 11
  480 IF(VARN .EQ. INT ) GO TO 941
      GO TO 946
  944 READ (DUMY'ID15,FMT ) IVAR(IST)
      NEX=IST+1
      DO 943 I=NEX,IEND
  943 VAR(I)=IVAR(IST)
      GO TO 11
  941 READ (DUMY'ID15,FMT ) (IVAR(I),I=IST,IEND)
      DO 945 I=IST,IEND
  945 VAR(I)=IVAR(I)
      GO TO 11
  946 READ(DUMY'ID15,FMT) (VAR(I),I=IST,IEND)
   11 CONTINUE
      IF(DMYHDR.EQ.0) GO TO  850
C
C
C**    TEST - IS LEVEL4 TO BE CREATED
C         YES - INITIALIZE LEVEL3 BLOCKS(EMISSION,METEOROLOGICAL AND
C               GENERAL)
C         NO  - GO TO STATEMENT NUMBER 2061 TO TEST WHICH OF THE ABOVE
C               THREE BLOCKS THE DATA BELONG TO
C
C
      IF(CMD.NE.NCAS) GO TO 2061
   12 DO 35 I=1,BLKSYZ
   35 IPOINT(I)=0
      ID3=NREC3
      WRITE(LEV3'ID3) IPOINT
C
C**    NUMBER OF RECORDS REQUIRED FOR EACH OF THE THREE BLOCKS OF
C      LEVEL3 FILE
C
      IREC=ID3-NREC3
```

```
      WRITE(LEV3'ID3) IPOINT
      WRITE(LEV3'ID3) IPOINT
      NSAVE=ID3
      WRITE(LEV3'1) NSAVE
C
C
C
C**   TEST - DOES DATA BELONG TO EMISSION POINTER
C        YES - WRITE POINTER TO LEVEL4 FILE
C        NO  - TEST IF DATA BELONGS TO METEOROLOGICAL POINTER
C
C        YES - WRITE POINTER TO LEVEL4 FILE
C        NO  - TEST IF DATA BELONGS TO GENERAL POINTER
C
C        YES - WRITE POINTER TO LEVEL4 FILE
C        NO  - CHECK DATA FOR VARIABLE 'BLCK'
C
C
C
 2061 IF(BLCK.NE.1) GO TO 31
      ID3=NREC3
      READ(LEV3'ID3) IPOINT
      IPOINT(ITEM)=ISAVE
      ID3=NREC3
      WRITE(LEV3'ID3) IPOINT
      GO TO 32
   31 IF(BLCK.NE.2) GO TO 33
      ID3=NREC3+IREC
      READ(LEV3'ID3) IPOINT
      IPOINT(ITEM)=ISAVE
      ID3=NREC3+IREC
      WRITE(LEV3'ID3) IPOINT
      GO TO 32
   33 IF(BLCK.NE.3) GO TO 3000
      ID3=NREC3+2*IREC
      READ(LEV3'ID3) IPOINT
      IPOINT(ITEM)=ISAVE
      ID3=NREC3+2*IREC
      WRITE(LEV3'ID3) IPOINT
   32 ID3=NREC3
      WRITE(6,71)
   71 FORMAT(1X,'LEVEL3 - POINTERS TO LEVEL4')
      READ(LEV3'ID3) IPOINT
      WRITE(6,72)
   72 FORMAT(1X,'EMISSION POINTER BLOCK')
      WRITE(6,23) IPOINT
      READ(LEV3'ID3) IPOINT
      WRITE(6,73)
   73 FORMAT(1X,'METEOROLOGICAL POINTER BLOCK')
      WRITE(6,23) IPOINT
      READ(LEV3'ID3) IPOINT
      WRITE(6,74)
   74 FORMAT(1X,'GENERAL POINTER BLOCK')
      WRITE(6,23) IPOINT
   23 FORMAT(1X,30I4)
```

```
C
C**   FOLLOWING    STATEMENTS,DEPENDING UPON HEADER,WRITE HEADER
C     DETAILS ON LEVEL4 FILE
C
      ID4=NREC4
      GO TO(900,901,902,903,904,905,906,907,908,909,910,911),HEADER
  900 WRITE(LEV4'ID4)FORM,UNITS
      GO TO 930
  901 WRITE(LEV4'ID4)FORM,UNITS,NROWS
      GO TO 930
  902 WRITE(LEV4'ID4)FORM,NROWS
      WRITE(LEV4'ID4)(VUNITS(I),I=1,NROWS)
      GO TO 930
  903 WRITE(LEV4'ID4)FORM,UNITS,NROWS,NCOLS
      GO TO 930
  904 WRITE(LEV4'ID4)FORM,NROWS,NCOLS
      WRITE(LEV4'ID4)(VUNITS(I),I=1,NCOLS)
      GO TO 930
  905 WRITE(LEV4'ID4) FORM,UNITS,NX,NY,DX,DY,X0,Y0,NVAL
      WRITE(LEV4'ID4)(VUNITS(I),I=1,NVAL)
      GO TO 930
  906 WRITE(LEV4'ID4)FORM,UNITS,NPTS,NVAL
      WRITE(LEV4'ID4)(VUNITS(I),I=1,NVAL)
      WRITE(LEV4'ID4)(X(I),Y(I),I=1,NPTS)
      GO TO 930
  907 WRITE(LEV4'ID4)FORM,UNITS,NROWS,NCOLS,NLEVS
      GO TO 930
  908 WRITE(LEV4'ID4)FORM,UNITS,NX,NY,NZ,DX,DY,DZ,X0,Y0,Z0,NVAL
      WRITE(LEV4'ID4)(VUNITS(I),I=1,NVAL)
      GO TO 930
  909 WRITE(LEV4'ID4)FORM,UNITS,NPTS,NVAL
      WRITE(LEV4'ID4)(VUNITS(I),I=1,NVAL)
      WRITE(LEV4'ID4)(X(I),Y(I),Z(I),I=1,NPTS)
      GO TO 930
  910 CONTINUE
  911 CONTINUE
  930 CONTINUE
      ID4=NREC4
      READ (LEV4'ISAVE)HEALER
  850 WRITE(6,75)
   75 FORMAT(1X ,'HEADER NUMBER')
      WRITE(6,601) HEADER
      WRITE(6,76)
   76 FORMAT(1X,'HEADER DETAILS')
      GO TO(800,801,802,803,804,805,806,807,808,809,810,811),HEADER
  800 READ (LEV4'ID4)FORM,UNITS
      WRITE(6,601) FORM,UNITS
      NELMNT=1
      GO TO 830
  801 READ (LEV4'ID4)FORM,UNITS,NROWS
      WRITE(6,601) FORM,UNITS,NROWS
      NELMNT=NROWS
      GO TO 830
  802 READ (LEV4'ID4)FORM,NROWS
      READ (LEV4'ID4)(VUNITS(I),I=1,NROWS)
```

```
      WRITE(6,601) FORM,NROWS
      WRITE(6,601)              (VUNITS(I),I=1,NROWS)
      NELMNT=NROWS
      GO TO 830
  803 READ (LEV4'ID4)FORM,UNITS,NROWS,NCOLS
      WRITE(6,601) FORM,UNITS,NROWS,NCOLS
      NELMNT=NROWS*NCOLS
      GO TO 830
  804 READ (LEV4'ID4)FORM,NROWS,NCOLS
      READ (LEV4'ID4)(VUNITS(I),I=1,NCOLS)
      WRITE(6,601) FORM,NROWS,NCOLS
      WRITE(6,601)                    (VUNITS(I),I=1,NCOLS)
      NELMNT=NROWS*NCOLS
      GO TO 830
  805 READ (LEV4'ID4) FORM,UNITS,NX,NY,DX,DY,X0,Y0,NVAL
      READ (LEV4'ID4)(VUNITS(I),I=1,NVAL)
      WRITE(6,601) FORM,UNITS,NX,NY
      WRITE (6,606) DX,DY,X0,Y0

      WRITE(6,601) FORM,UNITS,NPTS,NVAL
      WRITE(6,601) (VUNITS(I),I=1,NVAL)
      WRITE(6,606) (X(I),Y(I),I=1,NPTS)
      NELMNT=NPTS
      GO TO 830
  807 READ (LEV4'ID4)FORM,UNITS,NROWS,NCOLS,NLEVS
      WRITE(6,601) FORM,UNITS,NROWS,NCOLS,NLEVS
      NELMNT=NROWS*NCOLS*NLEVS
      GO TO 830
  808 READ (LEV4'ID4)FORM,UNITS,NX,NY,NZ,DX,DY,DZ,X0,Y0,Z0,NVAL
      READ (LEV4'ID4)(VUNITS(I),I=1,NVAL)
      WRITE(6,601) FORM,UNITS,NX,NY,NZ
      WRITE(6,606)EE          DX,DY,DZ,X0,Y0,Z0
      WRITE(6,601)NVAL,(VUNITS(I),I=1,NVAL)
      NELMNT=NX*NY*NZ
      GO TO 830
  809 READ (LEV4'ID4)FORM,UNITS,NPTS,NVAL
      READ (LEV4'ID4)(VUNITS(I),I=1,NVAL)
      READ (LEV4'ID4)(X(I),Y(I),Z(I),I=1,NPTS)
      WRITE(6,601) FORM,UNITS,NPTS,NVAL
      WRITE(6,601) (VUNITS(I),I=1,NVAL)
      WRITE(6,606) (X(I),Y(I),Z(I),I=1,NPTS)
      NELMNT=NPTS
      GO TO 830
  810 CONTINUE
C**   WRITE(6,611) HEADER NUMBER 11
  811 CONTINUE
C**   WRITE(6,612) HEADER NUMBER 12
  830 CONTINUE
C
C
C
C**   TEST - IS HEADER ZERO
CE    YES - THE CASE IS ALREADY CREATED,UPDATE LEVEL4 FILE
CE          IF NECESSARY
CE    NO  - THIS IS A NEW CASE BEING CREATED,ZERO OUT THE
```

```
CE          SPACE FOR THE PARTICULAR CASE WITH THE NUMBER
CE          1.E-38
C
C
C
      IF(DMYHDR.EQ.0) GO TO 8000
      DO 8001 I=1,NELMNT
 8001 VVAR(I)=1.E-38
 8000 IF(IST.GT.NELMNT) GO TO 189
      IF(IEND.GT.NELMNT) GO TO 189
      REC4=ID4
      IF(DMYHDR.EQ.0) READ(LEV4'ID4)(VVAR(I),I=1,NELMNT)
      DO 8002 I=IST,IEND
 8002 VVAR(I)=VAR(I)
      ID4=REC4
C
C**   WRITE DATA VALUES ON LEVEL4 FILE
C
      WRITE(LEV4'ID4) (VVAR(I),I=1,NELMNT)
      ID4=REC4
      READ (LEV4'ID4) (VVAR(I),I=1,NELMNT)
      WRITE(6,77)
   77 FORMAT(1X,'DATA VALUES ON LEVEL4 - THAT IS ON THE STANDARD DATA BA
     XSE')
      WRITE(6,24)E  (VVAR(I),I=1,NELMNT)
   24 FORMAT(1X,8E15.8)
   17 CONTINUE
      IF(DMYHDR.EQ.0) GO TO 185
      NREC4=ID4
  185 WRITE(LEV4'1) NREC4
      READ(LEV4'1) NREC4
      WRITE(6,78) NREC4
   78 FORMAT(1X,'NEXT AVAILABLE UNUSED RECORD ON LEVEL4 = ',I4)
C
C
C
C**   READ IN NEXT COMMAND
C     TEST - IS COMMAND 'NSIT'
CE    YES - REREAD THE PREVIOUS CARD AND GO TO STATEMENT NUMBER 50
CE    NO - TEST - IS COMMAND 'USIT'

CE    YES - REREAD THE PREVIOUS CARD AND GO TO STATEMENT NUMBER 50
CE    NO  - TEST - IS COMMAND 'NCAS'

CE    YES - REREAD THE PREVIOUS CARD AND GO TO STATEMENT NUMBER 50
CE    NO  - TEST - IS COMMAND 'UCAS'

CE    YES - REREAD THE PREVIOUS CARD AND GO TO STATEMENT NUMBER 50
CE    NO  - TEST - IS COMMAND 'END'

C     YES - REREAD THE PREVIOUS CARD AND GO TO STATEMENT NUMBER 50
C     NO  - REREAD THE PREVIOUS CARD AND GO TO STATEMENT NUMBER  2
```

```
      READ(5,101) CMD
  101 FORMAT(A4)
      IF(CMD.EQ.NSIT) GO TO 102
      IF(CMD.EQ.USIT) GO TO 102
      IF(CMD.EQ.NCAS) GO TO 102
      IF(CMD.EQ.UCAS) GO TO 102
      IF(CMD.EQ.END ) GO TO 3000
      BACKSPACE 5
      GO TO 2
  102 BACKSPACE 5
      GO TO 50
  189 WRITE (6,187) SITNUB,CASNUB
  187 FORMAT(1H1,'***********ETHE SPACE PROVIDED ON THE DISK IS NOT ENOU
     XGH FOR SITE # ',I6,2X,'CASE # ',I6,2X,'***********')
      GO TO 3000
  191 WRITE(6,192) SITE
  192 FORMAT(1H1,'******** SITE # ',I6,'IS ALREADY ON THE DATA BASE, CHEC
     XK THE INPUT FOR MASIND(1) ***********')
      GO TO 3000
  195 WRITE (6,194) CASNUB
  194 FORMAT (1H1,'***********  CASE',I6,2X,'CANNOT BE UPDATED, AS IT IS
     XNOT YET CREATED  ***********')
      GO TO 3000
  197 WRITE(6,198) SITNUB
  198 FORMAT(1H1,'***********ESITE',I6,2X,'CANNOT BE UPDATED, AS IT IS
     XNOT YET CREATED  ***********')
 3000 STOP
      END
C
C
C
C
C
C
C
C
C
C
C
C
C******************  DATA  RETRIEVAL  SYSTEM  ******************
C
C                           DARES
C
C**** THIS PROGRAM CREATES DATA DECK FOR TEST MODEL TO RUN ****
C********* USING STANDARD DATA BASE CREATED BY FORMAN *********
C
C                           DARES
C
C
C
C
C
C
C
```

```
C
C
C
      LOGICAL FIRST
      LOGICAL START
      INTEGER CASREF (200),CASES
      INTEGER ATYPE,REAL
      INTEGER SITREF(100)
      INTEGER HEADER
      INTEGER VUNITS(100)
      INTEGER POINT(51)
      INTEGER DUMY
      INTEGER OUT,CASE
      INTEGER FORM,CONV1,CONV2,CONV3,DFAULT
      INTEGER SITE
      DIMENSION STOR(100),ISTOR(100)
      DIMENSION X(100),Y(100),Z(100)
      DIMENSION ADAT(80)
      DIMENSION IPOINT(51),MASIND(35)
      DIMENSION DAT(80),FMT(15),VAR(100),IVAR(100)
      EQUIVALENCE (STOR,ISTOR)
      EQUIVALENCE (POINT,IPOINT)
      DATA FIRST/.FALSE./
      DATA ID1,ID2,ID3,ID4 /1,1,1,1/
      DATA START/.TRUE./
      DATA RET,CARD,END/4HRET ,4HCARD,4HEND /
      DATA IN8,IN9,OUT/1,1,1/
      DATA I8,DARES/2,4HDARE/
      DATA RETA,APND,DATA/4HRETA,4HAPND,4HDATA/
      DATA OVER,COPY,BLANK/4HOVER,4HCOPY,4H    /
      DATA ONEBLK/1H /
      DATA INT,REAL/3HINT,4HREAL/
      DEFINE FILE  1 (105,250,L,ID1)
      DEFINE FILE 2(500,210,L,ID2)
      DEFINE FILE 3(500,210,L,ID3)
      DEFINE FILE 4(500,80,L,ID4)
      DEFINE FILE 8(50,320,L,IN8)
      DEFINE FILE 9(50,320,L,IN9)
      DEFINE FILE10(50,320,L,OUT)
      DEFINE FILE15(50,320,L,IDUM)
      DEFINE FILE 16(500,210,L,ID2A)
      LEV2A=16
      IOUT=1
      KOUNT=1
C
C**   UNIT NUMBER FOR LEVEL 1 FILE
C
      LEV1=1
C
C**   UNIT NUMBER FOR LEVEL 2 FILE
C
      LEV2=2
C
C**   UNIT NUMBER FOR LEVEL 3 FILE
C
```

```
      LEV3=3
C
C**   UNIT NUMBER FOR LEVEL 4 FILE
C
      LEV4=4
C
C**   UNIT NUMBERS FOR DUMMY FILES
C
      DUMY=11
      IDUMY=15
C
C**   READ DATA AND WRITE ON UNIT 8 TIL THE COMMAND IS 'DARE'
C
    1 READ(5,100) (DAT(I),I=1,20)
  100 FORMAT(20A4)
      IF(DAT(1).EQ.DARES) GO TO 200
      WRITE(8'IN8,100)(DAT(I),I=1,20)
      GO TO 1
  200 CONTINUE
      IN8=1
    3 READ(5,100,END=300)(DAT(I),I=1,20)
C
C**   WRITE INPUT DATA DECK ON UNIT 9
C
      WRITE(9'IN9,100)  (DAT(I),I=1,20)
      GO TO 3
C
C**   ZERO OUT ARRAY 'DAT'
C
  300 DO 40 I=1,80
   40 DAT(I)= ONEBLK
      IN9=1
C
C**   READ IN SITE NUMBER AND CASE NUMBER
C
 1002 READ(9'IN9,3601) SITE,CASE
 3601 FORMAT(2I4)
      ISTOP=0
C
C**   READ IN TYPE DEPENDING UPON THE TYPE OF DATA DECK TO BE
C     GENERATED
C
    2 READ(9'IN9,3602) TYPE
 3602 FORMAT(A4)
C
C**   BACKSPACING ON UNIT 9 FOR REREADING THE SAME RECORD
C
      IN9=IN9-1
C
C
C
C
C
C**   TEST - TYPE EQUALS TO  'RET'
C         YES - GO TO STATEMENT NUMBER 1000
```

```
C
C          NO  - TEST - TYPE EQUALS TO 'RETA'
C          YES - GO TO STATEMENT NUMBER 1000
C          NO  - TEST - TYPE EQUALS TO 'CARD'
C
C          YES - GO TO STATEMENT NUMBER 2000
C          NO  - TEST - TYPE EQUALS TO 'END'
C
C          YES - GO TO STATEMENT NUMBER 3000
C          NO  - TEST - TYPE EQUALS TO 'APND'
C
C          YES - GO TO STATEMENT NUMBER 5001
C          NO  - READ IN TYPE,COMMAND,TYPE OF VARIABLE(REAL OR INTEGER).
C                FORMAT CONTROL WITH WHICH TO READ DATA VALUES ETC.
C
C
C
C
      IF(TYPE .EQ. RET) GO TO 1000
      IF(TYPE.EQ.RETA) GO TO 1001
      IF(TYPE .EQ.CARD) GO TO 2000
      IF(TYPE .EQ. END) GO TO 3000
      IF(TYPE.NE.APND) GO TO 5001
      READ(9'IN9,3603) TYPE,CMD,ATYPE,NWORD,FMT
 3603  FORMAT(3A4, I4,15A4)
      GO TO 51
C
C
C**   READ IN TYPE,LEVEL3 BLOCK (EMISSION,METEOROLOGICAL
C     OR GENERAL ),ITEM NUMBER OF ONE OF THE LEVEL3 BLOCKS,
C     CONVERSION FACTORS 1 AND 2,DEFAULT FACTOR,FIRST AND LAST
C     SUBSCRIPTS FOR STORAGE IN THE VECTOR FORM
C
C
 1000 INI=0
 1001 READ(9'IN9,3604) TYPE,LOC1,LOC2,CONv1,CONV2,DFAULT,IST,IEND
 3604 FORMAT(A4,I1,I3,2I4,I2,2I4)
C
C**   READ LEVEL 1 FILE(KNOWING THE SITE NUMBER)
C
      READ(LEV1'1) NSITES,SITREF
      ID1=SITREF(SITE)
      READ(LEV1'ID1) MASIND
C
C**   POINTER TO LEVEL 2 FILE
C
      ID2=MASIND(29)
C
C**   READ LEVEL 2 FILE
C
      ID2A = (SITE  -1) * 5 + 1
      READ (LEV2A'ID2A) CASREF
      CASES = CASREF (CASE)
  111 READ(LEV2'ID2  ) POINT
C
```

```
C
C      TEST - MORE THAN ONE BLOCK REQUIRED
C          YES - READ THE REST OF THE BLOCKS
C          NO  - GO TO STATEMENT NUMBER 407 TO READ LEVEL 3 FILE
C
C
       IF(POINT(51).EQ.0) GO TO 407
       NEXT2=POINT(51)
       ID2=POINT(51)
       GO TO 111
C
C
C
C
C**    TEST - EMISSION BLOCK
C         YES - GO TO STATEMENT NUMBER 401
C         NO  - TEST - METEOROLOGICAL BLOCK
C
C         YES - GO TO STATEMENT NUMBER 402
C         NO  - TEST - GENERAL BLOCK
C
C         YES - GO TO STATEMENT NUMBER 403
C         NO  - TEST - WRITE THE ERROR MESSAGE 'CHECK DATA FOR LEVEL 3
C                BLOCK
C
C
C
C
  407  IF(LOC1.EQ.1) GO TO 401
       IF(LOC1.EQ.2) GO TO 402
       IF(LOC1.EQ.3) GO TO 403
       WRITE(6,404)
  404  FORMAT(1H1,'ERROR - CHECK DATA FOR LOC1')
       STOP
  401  ID3=POINT(CASES)
C
C**    READ LEVEL 3 FILE (EMISSION BLOCK)
C
  421  READ(LEV3'ID3   ) IPOINT
C
C
C      TEST - MORE THAN ONE BLOCK REQUIRED
C          YES - READ THE REST OF THE BLOCKS
C          NO  - GO TO STATEMENT NUMBER 405 TO READ LEVEL 4 FILE
C
C
       IF(IPOINT(51).EQ.0) GO TO 405
       ID3=IPOINT(51)
       GO TO 421
  402  ID3=POINT(CASES)
       READ(LEV3'ID3   ) IPOINT
C
C**    READ LEVEL 3 FILE (METEOROLOGICAL BLOCK)
C
C
```

```
  422 READ(LEV3'ID3   ) IPOINT
C
C
C     TEST - MORE THAN ONE BLOCK REQUIRED
C         YES - READ THE REST OF THE BLOCKS
C         NO  - GO TO STATEMENT NUMBER 405 TO READ LEVEL 4 FILE
C
C
      IF(IPOINT(51).EQ.0) GO TO 405
      ID3=IPOINT(51)
      GO TO 422
  403 ID3=POINT(CASES)
C
C**   READ LEVEL 3 FILE (GENERAL BLOCK)
C
      READ(LEV3'ID3   ) IPOINT
      READ(LEV3'ID3   ) IPOINT
  423 READ(LEV3'ID3   ) IPOINT
C
C
C     TEST - MORE THAN ONE BLOCK REQUIRED
C         YES - READ THE REST OF THE BLOCKS
C         NO  - GO TO STATEMENT NUMBER 405 TO READ LEVEL 4 FILE
C
C
      IF(IPOINT(51).EQ.0) GO TO 405
      ID3=IPOINT(51)
      GO TO 423
  405 ID4=IPOINT(LOC2)
C
C**   READ LEVEL 4 FILE (HEADER NUMBER)
C
      READ(LEV4'ID4) HEADER
C
C**   DEPENDING UPON HEADER NUMBER,READ HEADER DETAILS
C
      GO TO (600,601,602,603,604,605,606,607,608,609,610,611),HEADER
  600 READ (LEV4'ID4)FORM,UNITS
      WRITE(6,801) FORM,UNITS
      NELMNT=1
      GO TO 630
  601 READ (LEV4'ID4)FORM,UNITS,NROWS
      WRITE(6,801) FORM,UNITS,NROWS
      NELMNT=NROWS
      GO TO 630
  602 READ (LEV4'ID4)FORM,NROWS
      READ (LEV4'ID4)(VUNITS(I),I=1,NROWS)
      WRITE(6,803) FORM,UNITS,(VUNITS(I),I=1,NROWS)
      NELMNT=NROWS
      GO TO 630
  603 READ (LEV4'ID4)FORM,UNITS,NROWS,NCOLS
      WRITE(6,801) FORM,UNITS,NROWS,NCOLS
      NELMNT=NROWS*NCOLS
      GO TO 630
  604 READ (LEV4'ID4)FORM,NROWS,NCOLS
```

```
      READ (LEV4'ID4)(VUNITS(I),I=1,NCOLS)
      WRITE(6,805) FORM,NROWS,NCOLS,(VUNITS(I),I=1,NCOLS)
      NELMNT=NROWS*NCOLS
      GO TO 630
  605 READ (LEV4'ID4) FORM,UNITS,NX,NY,DX,DY,XO,YO,NVAL
      READ (LEV4'ID4)(VUNITS(I),I=1,NVAL)
      WRITE(6,806) FORM,UNITS,NX,NY,DX,DY,XO,YO,NVAL,(VUNITS(I),I=1,NVAL
     X             )
      NELMNT=NX*NY
      GO TO 630
  606 READ (LEV4'ID4)FORM,UNITS,NPTS,NVAL
      READ (LEV4'ID4)(VUNITS(I),I=1,NVAL)
      READ (LEV4'ID4)(X(I),Y(I),I=1,NPTS)
      WRITE(6,807) FORM,UNITS,NPTS,NVAL,(VUNITS(I),I=1,NVAL),
     X             (X(I),Y(I),I=1,NPTS)
      NELMNT=NPTS
      GO TO 630
  607 READ (LEV4'ID4)FORM,UNITS,NROWS,NCOLS,NLEVS
      WRITE(6,801) FORM,UNITS,NROWS,NCOLS,NLEVS
      NELMNT=NROWS*NCOLS*NLEVS
      GO TO 630
  608 READ (LEV4'ID4)FORM,UNITS,NX,NY,NZ,DX,DY,DZ,XO,YO,ZO,NVAL
      READ (LEV4'ID4)(VUNITS(I),I=1,NVAL)
      WRITE(6,809) FORM,UNITS,NX,NY,NZ,DX,DY,DZ,XO,YO,ZO,NVAL,
     X             (VUNITS(I),I=1,NVAL)
      NELMNT=NX*NY*NZ
      GO TO 630
  609 READ (LEV4'ID4)FORM,UNITS,NPTS,NVAL
      READ (LEV4'ID4)(VUNITS(I),I=1,NVAL)
      READ (LEV4'ID4)(X(I),Y(I),Z(I),I=1,NPTS)
      WRITE(6,807) FORM,UNITS,NPTS,NVAL,(VUNITS(I),I=1,NVAL),
     X             (X(I),Y(I),Z(I),I=1,NPTS)
      NELMNT=NPTS
      GO TO 630
  610 CONTINUE
C**   WRITE(6,811) HEADER NUMBER 11
  611 CONTINUE
C**   WRITE(6,812) HEADER NUMBER 12
      GO TO 630
  630 CONTINUE
  801 FORMAT(1X,5I4)
  803 FORMAT(1X,2I4,5E15.7)
  805 FORMAT(1X,3I4,4E15.7)
  806 FORMAT(1X,4I4,4E15.7,I4/1X,5E15.7)
  807 FORMAT(1X,4I4,5E15.7/1X,4E15.7)
  809 FORMAT(1X,5I4,6E15.7,I4/1X,5E15.7)
C
C**   READ VARIABLE VALUES FROM LEVEL 4 FILE
C
      READ(LEV4'ID4     ) (VAR(I),I=1,NELMNT)
      DO 61 I=IST,IEND
      INI=INI+1
      STOR(INI)=VAR(I)
   61 CONTINUE
      GO TO  2
```

```
C
C
C**    TEST - LOGICAL VARIABLE START EQUALS TRUE
C         YES - GO TO STATEMENT NUMBER 11
C         NO  - WRITE VARIABLE 'DAT' ON UNIT 10
C
C
 2000 IF(START) GO TO 11
      WRITE(10'IOUT) DAT
      IOUT=IOUT+1
   11 START=.FALSE.
      DO 41 I=1,80
   41 DAT(I)= ONEBLK
C
C**    READ ON UNIT 9(TYPE,COMMAND,TYPE OF VARIABLE,FORMAT ETC.)
C
      READ(9'IN9,3603) TYPE,CMD,ATYPE,NWORD,FMT
C
C
C
C
C**    TEST - COMMAND EQUALS OVER (OVER WRITE COMMAND)
C         YES - GO TO STATEMENT NUMBER 5
C         NO  - TEST - COMMAND EQUALS DATA
C
C         YES - GO TO STATEMENT NUMBER 5
C         NO  - TEST - COMMAND EQUALS COPY (COPY COMMAND)
C
C         YES - GO TO STATEMENT NUMBER 7
C         NO  - WRITE THE ERROR MESSAGE 'ERROR - CHECK CMD'
C
C
C
C
   51 IF(CMD.EQ.OVER) GO TO 5
      IF(CMD.EQ. DATA) GO TO 55
      IF(CMD.EQ.COPY) GO TO 7
      WRITE(6,8)
    8 FORMAT(1H1,'ERROR - CHECK CMD')
  501 FORMAT(80A1)
      STOP
    5 CONTINUE
      IF(TYPE.EQ.CARD) READ(8'IN8,501) DAT
C
C
C
C**    TEST - VARIABLE TYPE INTEGER
C         YES - GO TO STATEMENT NUMBER 31
C         NO  -TEST - VARIABLE TYPE REAL
C
C         YES - GO TO STATEMENT NUMBER 32
C         NO  - WRITE ERROR MESSAGE 'ERROR - CHECK ATYPE'
C
C
C
```

```
   55 CONTINUE
      IBGN=ISTOP+1
      ISTOP=ISTOP+NWORD
      IF(ATYPE.EQ.INT) GO TO 31
      IF(ATYPE.EQ.REAL) GO TO 32
  293 WRITE(6,33)
   33 FORMAT(1H1,'ERROR - CHECK ATYPE')
      STOP
   31 CONTINUE
C
C**   CHANGING VARIABLE NAME TO INTEGER VARIABLE NAME
C
      DO 510 I=IBGN,ISTOP
  510 ISTOR(I)=STOR(I)
      WRITE(DUMY,FMT) (ISTOR(I),I=IBGN,ISTOP)
      GO TO 34
   32 CONTINUE
C
C**   WRITE VARIABLE VALUES WITH FORMAT CONTROL ON DUMY UNIT
C
      WRITE(DUMY,FMT) (STOR(I),I=IBGN,ISTOP)
C
C**   REWIND DUMY UNIT
C
   34 REWIND DUMY
C
C**   READ DUMY UNIT AS VARIABLE 'ADAT'
C
      READ(DUMY,20) ADAT
   20 FORMAT(80A1)
      DO 30 I=1,80
C
C
C**   TEST - VARIABLE 'ADAT' EQUALS BLANK(1 CHARACTER BLANK)
C        YES - GO TO STATEMENT NUMBER 30
C        NO  - SET VARIABLE 'DAT' EQUALS VARIABLE 'ADAT'
C
C
      IF(ADAT(I).EQ.ONEBLK) GO TO 30
      DAT(I)=ADAT(I)
   30 CONTINUE
      REWIND DUMY
      GO TO 2
    7 CONTINUE
      IF(TYPE.EQ.CARD) READ(8'IN8,501) DAT
      GO TO 2
C
C**   WRITE VARIABLE 'DAT' ON UNIT 10(OUTPUT UNIT)
C
 3000 WRITE(10'IOUT) DAT
      IOUT=IOUT+1
      WRITE(6,3500)
      WRITE(6,3501)
      K=IOUT-1
      DO 3600 I=1,K
```

```
      READ(10'I) DAT
      WRITE(6,35) DAT
3600  CONTINUE
      WRITE(6,3501)
3500  FORMAT(1H1,10X,'DARES OUTPUT'//)
3501  FORMAT(1X,'************************************************************
     X*****************')
   35 FORMAT(1X,80A1)
      START=.TRUE.
      IN9=IN9+1
      IOUT=1
      READ(9'IN9) CMD
      IF(CMD.EQ.DARES) GO TO 1002
      GO TO 5004
5001  WRITE(6,5002)
5002  FORMAT(1H1,'**********  CHECK INPUT FOR VARIABLE TYPE, SHOULD BE O
     XNE OF THE FOLLOWING - DARE,RET,RETA,CARD,APND, OR END  *********')
5004  STOP
      END
```

```
C
C
C
C
C
C
C**********                    S M O G                    **********
C
C   S T A N D A R D    M O D E L    O U T P U T    G E N E R A T O R
C
C
C
C
C
C
C
C
      DEFINE FILE 21(100,30,L,IC1)
      DEFINE FILE 22(500,100,L,ID2)
      DIMENSION ARRAY(1)
      DIMENSION FMT(2)
      INTEGER SITE,CASE
      INTEGER FARRAY
      DATA FMT /4H(6E1,4H5.8)/
      INTEGER BLOCK
      DATA NREC1,NREC2,LEV1,LEV2/ 2,2,21,22/
      WRITE (LEV1'1) NREC1
      WRITE (LEV2'1) NREC2
      READ (LEV1'1) NREC1
      ID1=NREC1
      WRITE (LEV1'ID1) NREC2,MODEL,SITE,CASE
      NREC1=ID1
      WRITE (LEV1'1) NREC1
      READ (LEV2'1) NREC2
      ID2=NREC2
      WRITE (LEV2'ID2,FMT) (ARRAY(I),I=1,FARRAY)
      NREC2=ID2
      WRITE (LEV2'1) NREC2
      RETURN
      END

C
C
C
C
C
C
C
C
C
C**********                    D I M O T E                    **********
C
C          D I S P E R S I O N    M O D E L    T E S T E R
C
C
C
C
C
C
C
C
C
C
C
      DEFINE FILE  1 (105,250,L,ID1)
      DEFINE FILE 2(500,210,L,ID2)
      DEFINE FILE 3(500,210,L,ID3)
      DEFINE FILE 4(500,80,L,ID4)
      DEFINE FILE16(500,210,L,ID2A)
      DEFINE FILE 21(100,30,L,IC1)
      DEFINE FILE 22(500,100,L,ID2)
      DIMENSION FMT(2)
      DATA FMT /4H(6E1,4H5.8)/
      DIMENSION X(100),Y(100),Z(100)
      DIMENSION ARRAY( 5, 5,100),VAR( 5, 5,100),VVAR(100)
      COMMON/FORSMG/MODEL,DN,ND,ARRAY,VAR,F
      INTEGER SITNUB,CASNUB,HEADER,FORM,UNITS,BLCK
```

```
      INTEGER SITREF(100),CASREF(200),MASIND(35),POINT(51),IPOINT(51),
     1VUNITS(100)
      INTEGER SITE,CASE,SITEN,CASEN,BLOCK
      INTEGER DN( 5),NC( 5,100)
      DATA LEV1S,LEV2S /21,22/
      DATA LEV1,LEV2,LEV2A,LEV3,LEV4 /1,2,16,3,4 /
      DO 10 I= 1, 5
   10 DN(I)=0
   40 READ(5,1,END=3000) MODEL,SITE,CASE,BLOCK,ITEM
    1 FORMAT(5I4)
C
C** RETRIEVE MEASURED VALUES FROM THE TAPS DATA BASE
C
      BLCK=BLOCK
      SITNUB=SITE
      CASNUB=CASE
      READ (LEV1'1) NSITES,SITREF
 2060 ID1=SITREF(SITNUB)
      READ(LEV1'ID1) MASIND
      ID2A = (SITNUB-1) * 5 + 1
      READ(LEV2A'ID2A) CASREF
      IF (CASREF(CASNUB) .EQ. 0) GO TO  195
      NCASES = CASREF (CASNUB )
      IREC2=MASIND(29)
  111 READ(LEV2'IREC2) POINT
      IF(POINT(51).EQ.0) GO TO 407
      IREC2=POINT(51)
      GO TO 111
  407 IF(BLCK.EQ.1) GO TO 401
      IF(BLCK.EQ.2) GO TO 402
      IF(BLCK.EQ.3) GO TO 403
  401 IREC3=POINT(NCASES)
  421 READ(LEV3'IREC3) IPOINT
      IF(IPOINT(51).EQ.0) GO TO 405
      IREC3=IPOINT(51)
      GO TO 421
  402 IREC3=POINT(NCASES)
      READ(LEV3'IREC3) IPOINT
      IREC3=ID3
  422 READ(LEV3'IREC3) IPOINT
      IF(IPOINT(51).EQ.0) GO TO 405
      IREC3=IPOINT(51)
      GO TO 422
  403 IREC3=POINT(NCASES)
      READ(LEV3'IREC3) IPOINT
      IREC3=ID3
      READ(LEV3'IREC3) IPOINT
      IREC3=ID3
  423 READ(LEV3'IREC3) IPOINT
      IF(IPOINT(51).EQ.0) GO TO 405
      IREC3=IPOINT(51)
      GO TO 423
  405 IREC4=IPOINT(ITEM)
      READ(LEV4'IREC 4) HEADER
      GO TO(800,801,802,803,804,805,806,807,808,809,810,811),HEADER
  800 READ (LEV4'ID4)FORM,UNITS
      NELMNT=1
      WRITE(6,601) FORM,UNITS
      GO TO 830
  801 READ (LEV4'ID4)FORM,UNITS,NROWS
      WRITE(6,601) FORM,UNITS,NROWS
      NELMNT=NROWS
      GO TO 830
  802 READ (LEV4'ID4)FORM,NROWS
      READ (LEV4'ID4)(VUNITS(I),I=1,NROWS)
      WRITE(6,601) FORM,NROWS
      WRITE(6,601)              (VUNITS(I),I=1,NROWS)
      NELMNT=NROWS
      GO TO 830
  803 READ (LEV4'ID4)FORM,UNITS,NROWS,NCOLS
      WRITE(6,601) FORM,UNITS,NROWS,NCOLS
      NELMNT=NROWS*NCOLS
      GO TO 830
  804 READ (LEV4'ID4)FORM,NROWS,NCOLS
      READ (LEV4'ID4)(VUNITS(I),I=1,NCOLS)
      WRITE(6,601) FORM,NROWS,NCOLS
      WRITE(6,601)                    (VUNITS(I),I=1,NCOLS)
      NELMNT=NROWS*NCOLS
```

```
      GO TO 830
  805 READ (LEV4'ID4) FORM,UNITS,NX,NY,DX,DY,X0,Y0,NVAL
      READ (LEV4'ID4)(VUNITS(I),I=1,NVAL)
      WRITE(6,601) FORM,UNITS,NX,NY
      WRITE (6,606) DX,DY,X0,Y0
      WRITE(6,601)NVAL,(VUNITS(I),I=1,NVAL)
      NELMNT=NX*NY
      GO TO 830
  806 READ (LEV4'ID4)FORM,UNITS,NPTS,NVAL
      READ (LEV4'ID4)(VUNITS(I),I=1,NVAL)
      READ (LEV4'ID4)(X(I),Y(I),I=1,NPTS)
      WRITE(6,601) FORM,UNITS,NPTS,NVAL
      WRITE(6,601) (VUNITS(I),I=1,NVAL)
      WRITE(6,606) (X(I),Y(I),I=1,NPTS)
      NELMNT=NPTS
      GO TO 830
  807 READ (LEV4'ID4)FORM,UNITS,NROWS,NCOLS,NLEVS
      WRITE(6,601) FORM,UNITS,NROWS,NCOLS,NLEVS
      NELMNT=NROWS*NCOLS*NLEVS
      GO TO 830
  808 READ (LEV4'ID4)FORM,UNITS,NX,NY,NZ,DX,DY,DZ,X0,Y0,Z0,NVAL
      READ (LEV4'ID4)(VUNITS(I),I=1,NVAL)
      WRITE(6,601) FORM,UNITS,NX,NY,NZ
      WRITE(6,606)                          DX,DY,DZ,X0,Y0,Z0
      WRITE(6,601)NVAL,(VUNITS(I),I=1,NVAL)
      NELMNT=NX*NY*NZ
      GO TO 830
  809 READ (LEV4'ID4)FORM,UNITS,NPTS,NVAL
      READ (LEV4'ID4)(VUNITS(I),I=1,NVAL)
      READ (LEV4'ID4)(X(I),Y(I),Z(I),I=1,NPTS)
      WRITE(6,601) FORM,UNITS,NPTS,NVAL
      WRITE(6,601) (VUNITS(I),I=1,NVAL)
      WRITE(6,606) (X(I),Y(I),Z(I),I=1,NPTS)
      NELMNT=NPTS
      GO TO 830
  810 CONTINUE
C**   WRITE(6,611) HEADER NUMBER 11
  811 CONTINUE
C**   WRITE(6,612) HEADER NUMBER 12
  830 CONTINUE
  601 FORMAT(1X,7I4)
  606 FORMAT(1X,6E15.7)
      READ (LEV4'ID4) (VVAR(I),I=1,NELMNT)
      WRITE(6,77)
   77 FORMAT(1X,'DATA VALUES ON LEVEL4 - THAT IS ON THE STANDARD DATA BA
     XSE')
      WRITE(6,24)      (VVAR(I),I=1,NELMNT)
   24 FORMAT(1X,8E15.8)
      DN(MODEL)=DN(MODEL)+1
      NDD=DN(MODEL)
      ND(MODEL,NDD)=NELMNT
      DO 50 I=1,NELMNT
      VAR(MODEL,NDD,I)=VVAR(I)
   50 WRITE (6,24) VAR (MODEL,NDD,I)
C
C** RETRIEVE CALCULATED VALUES FROM THE SMOG DATA BASE
C
      READ(LEV1S'1) NREC1
    2 READ(LEV1S'IC1) NREC2,MODLN,SITEN,CASEN
      WRITE (6,3) MODEL,MODLN,SITE,SITEN,CASE,CASEN,NREC1,ID1
    3 FORMAT (1H1,8I4)
      IF(MODEL.NE.MODLN)  GO TO 20
      IF(SITE .NE.SITEN)  GO TO 20
      IF(CASE .NE.CASEN)  GO TO 20
      READ(LEV2S'NREC2,FMT)  (ARRAY(MODEL,NDD,I),I=1,NELMNT)
      WRITE (6,24) (ARRAY(MODEL,NDD,I),I=1,NELMNT)
      GO TO 40
   20 IF((ID1-1).GE.NREC1)  GO TO 30
      GO TO 2
   30 WRITE(6,100) MODEL,SITE,CASE
  100 FORMAT(1H1,'THERE IS NO COMBINATION OF MODEL # ',I4,' SITE # ',I4,
     1' CASE # ',I4,' ON THE SMOG DATA BASE')
      STOP
 3000 CONTINUE
      WRITE(6,200)
  200 FORMAT(1H1,10X,'MODEL #',13X,'LOSS')
      DO 70 I=1, 5
      IF(DN(I).EQ.0) GO TO 70
```

```
      MODEL=I
      CALL FLOSS(3,1.0,1.0,2.0)
      CALL FLOSS(3,0.0,1.0,1.0)
      CALL FLOSS(4,1.0,1.0,2.0)
   70 CONTINUE
      WRITE(6,202)
  202 FORMAT(1H1,10X,'MODEL #',13X,'RANK CORRELATION COEFFICIENT')
      DO 72 I=1,5
      IF(DN(I).EQ.0) GO TO 72
      MODEL=I
      CALL RANKCC
   72 CONTINUE
      WRITE(6,204)
  204 FORMAT(1H1,2X,'NATURAL HISTOGRAM TEST')
      DO 74 I=1,5
      IF(DN(I).EQ.0) GO TO 74
      MODEL=I
      CALL NATHIS
   74 CONTINUE
      GO TO 199
  195 WRITE(6,196) CASE,SITE
  196 FORMAT(1H1,'CASE # ',I4,'  OF SITE # ',I4,'  IS NOT ON THE TAPS DA
     1TA BASE')
  199 STOP
      END
      SUBROUTINE FLOSS(IF1,ALPHA,BETA1,BETA2)
      COMMON/FORSMG/MODEL,DN,ND,ARRAY,VAR,F
      REAL LOS( 5)
      DIMENSION ARRAY( 5, 5,100),VAR( 5, 5,100)
      INTEGER DN( 5)
      DIMENSION ND(5,100)
      ISET=DN(MODEL)
      IVAL= ND(MODEL,ISET)
      IVALT=0
      FF=0.
      DO 1 I=1,ISET
      DO 2 J=1,IVAL
      GO TO (4,5,6,7),IF1
    4 F=ABS(ARRAY(MODEL,I,J)-VAR(MODEL,I,J))
      GO TO 50
    5 F=SQRT((ARRAY(MODEL,I,J)-VAR(MODEL,I,J))**2)
      GO TO 50
    6 F=(ALOG10(ARRAY(MODEL,I,J))-ALOG10(VAR(MODEL,I,J)))**2
      IF(ARRAY(MODEL,I,J).GT.VAR(MODEL,I,J)) GO TO 10
      IF(ARRAY(MODEL,I,J).LT.VAR(MODEL,I,J)) GO TO 20
   10 BETA=BETA1
      GO TO 40
   20 BETA=BETA2
   40 F=BETA*(VAR(MODEL,I,J))**ALPHA *F
      GO TO 50
    7 F=(ALOG10(ARRAY(MODEL,I,J))-ALOG10(VAR(MODEL,I,J)))**2
   50 CONTINUE
      FF=FF+F
    2 CONTINUE
      IVALT=IVALT+IVAL
    1 CONTINUE
      RIVALT=IVALT
      LOS(MODEL)=FF/RIVALT
      WRITE(6,200) MODEL,LOS(MODEL)
  200 FORMAT(1H0,  10X,I4  , 9X,E15.8)
      RETURN
      END
      SUBROUTINE RANKCC
      COMMON/FORSMG/MODEL,DN,ND,ARRAY,VAR,F
      DIMENSION ARRAY(5,5,100), VAR(5,5,100), VALRAY(2,100)
      INTEGER DN(5)
      DIMENSION ND(5,100)
      ISET=DN(MODEL)
      IVAL=ND(MODEL,ISET)
      R=0.
      IVALT=0
      DO 1 K=1,ISET
      DO 2 J=1,IVAL
      VALRAY(1,J)=VAR(MODEL,K,J)
      VALRAY(2,J)=ARRAY(MODEL,K,J)
    2 CONTINUE
      IND1=1
```

```
      IND2=2
  100 CONTINUE
      ISORT=IVAL-1
   50 IMON=1
      DO 60 I=1,ISORT
      IF(VALRAY(IND1,I+1).LT.VALRAY(IND1,I)) GO TO 70
      GO TO 60
   70 SAVE=VALRAY(IND1,I)
      VALRAY(IND1,I)=VALRAY(IND1,I+1)
      VALRAY(IND1,I+1)= SAVE
      SAV=VALRAY(IND2,I)
      VALRAY(IND2,I)=VALRAY(IND2,I+1)
      VALRAY(IND2,I+1)=SAV
      IMON=IMON+1
      IX=I
   60 CONTINUE
      IF(IMON.EQ.1) GO TO 80
      ISORT=IX
      GO TO 50
   80 IF(IND2.EQ.1) GO TO 110
      DO 90 I=1,IVAL
   90 VALRAY(1,I)=1
      IND1=2
      IND2=1
      GO TO 100
  110 CONTINUE
      DO 120 I=1,IVAL
  120 R=R+(VALRAY(1,I)-FLOAT(I))**2
      IVALT=IVALT+IVAL
    1 CONTINUE
      RIVALT=IVALT
      R=R/RIVALT
      WRITE(6,200) MODEL,R
  200 FORMAT(1H0,10X,I4,21X,E15.8)
      RETURN
      END
      SUBROUTINE NATHIS
      COMMON/FORSMG/MODEL,DN,ND,ARRAY,VAR,F
      DIMENSION ARRAY(5,5,100),VAR(5,5,100), VALRAY(2,100)
      DIMENSION PROBA(10,10),PROBB(10,100)
      INTEGER DN(5)
      REAL LEVELS(2,10)
      DIMENSION ND(5,100)
      DIMENSION F(10),NS(10),NT(10)
      ISET=DN(MODEL)
      IVAL=ND(MODEL,ISET)
      DO 1 N=1,ISET
      DO 2 J=1,IVAL
      VALRAY(1,J)=VAR(MODEL,N,J)
      VALRAY(2,J)=ARRAY(MODEL,N,J)
    2 CONTINUE
      IND1=1
      IND2=2
  100 CONTINUE
      ISORT=IVAL-1
   50 IMON=1
      DO 60 I=1,ISORT
      IF(VALRAY(IND1,I+1).LT.VALRAY(IND1,I)) GO TO 70
      GO TO 60
   70 SAVE=VALRAY(IND1,I)
      VALRAY(IND1,I)=VALRAY(IND1,I+1)
      VALRAY(IND1,I+1)=SAVE
      SAV=VALRAY(IND2,I)
      VALRAY(IND2,I)=VALRAY(IND2,I+1)
      VALRAY(IND2,I+1)=SAV
      IMON=IMON+1
      IX=I
   60 CONTINUE
      IF(IMON.EQ.1) GO TO 80
      ISORT=IX
      GO TO 50
   80 CONTINUE
      DO 90 K=1,10
      LEVELS(IND1,K)=VALRAY(IND1,1)+ (FLOAT(K)/11.)*(VALRAY(IND1,IVAL)-
     X VALRAY(IND1,1))
   90 CONTINUE
      IF(IND2.EQ.1) GO TO 110
      IND1=2
```

```
      IND2=1
      GO TO 100
  110 E=1./FLOAT(2*IVAL)
      VALRAY(2,IVAL+1)=VALRAY(2,IVAL) + VALRAY(2,IVAL)/10000.
      DO 200 L=1,10
      S=LEVELS(1,L)
      IS=0
      DO 290 K=1,IVAL
      IS=IS+1
      NT(IS)=1
      NS(IS)=0
      IF(VALRAY(1,K).GT.S) NS(IS)=1
  210 FLOTNS=NS(IS)
      FLOTNT=NT(IS)
      P(IS)=FLOTNS/FLOTNT
      IF(IS.EQ.1) GO TO 290
      IF(P(IS).GT.(P(IS-1)+E)) GO TO 290
      NS(IS-1)=NS(IS-1)+NS(IS)
      NT(IS-1)=NT(IS-1)+NT(IS)
      IS=IS-1
      GO TO 210
  290 CONTINUE
      WRITE(6,300) LEVELS(1,L),LEVELS(2,L)
  300 FORMAT(1H0,2X,2F10.4)
      DO 302 M=1,IS
      WRITE(6,304) P(M),NS(M),NT(M)
  304 FORMAT(10X,F10.4,4X,I10,4X,I10)
  302 CONTINUE
      L2=1
      IEND=0
      DO 400 M=1,IS
      ISTART=IEND+1
      IEND=IEND+NT(M)
      DO 380 I=ISTART,IEND
      PROBB(L,I)=P(M)
  360 IF(L2.GT.10) GO TO 380
      IF(LEVELS(2,L2).LT.VALRAY(2,I).OR.LEVELS(2,L2).GE.VALRAY(2,I+1))
     X GO TO 380
      PROBA(L,L2)=P(M)
      L2=L2+1
      GO TO 360
  380 CONTINUE
  400 CONTINUE
      DO 500 M=1,IVAL
      WRITE(6,502) VALRAY(2,M),PROBB(L,M)
  502 FORMAT(4X,F15.6,3X,F7.4)
  500 CONTINUE
      DO 600 M=1,10
      WRITE(6,602) LEVELS(2,M),PROBA(L,M)
  602 FORMAT(4X,F15.6,3X,F7.4)
  600 CONTINUE
  200 CONTINUE
    1 CONTINUE
      RETURN
      END
```