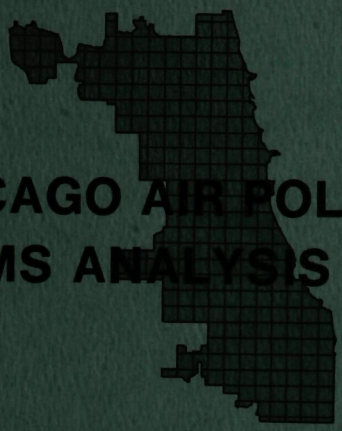
ANL/ES-CC-006

ANL/ES-CC-006

APR 14 1970 ABK



CHICAGO AIR POLLUTION SYSTEMS ANALYSIS PROGRAM

APICS

A COMPUTERIZED AIR POLLUTION DATA MANAGEMENT SYSTEM

ARGONNE NATIONAL LABORATORY

• • •

CHICAGO DEPARTMENT OF ENVIRONMENTAL CONTROL

• • •

DEPARTMENT OF HEALTH, EDUCATION, AND WELFARE
NATIONAL AIR POLLUTION CONTROL ADMINISTRATION

RETURN TO REFERENCE FILE
TECHNICAL PUBLICATIONS
DEPARTMENT

U of C-AUA-USAEC

CENTER FOR ENVIRONMENTAL STUDIES

The facilities of Argonne National Laboratory are owned by the United States Government. Under the terms of a contract (W-31-109-Eng-38) between the U. S. Atomic Energy Commission, Argonne Universities Association and The University of Chicago, the University employs the staff and operates the Laboratory in accordance with policies and programs formulated, approved and reviewed by the Association.

MEMBERS OF ARGONNE UNIVERSITIES ASSOCIATION

The University of Arizona
Carnegie-Mellon University
Case Western Reserve University
The University of Chicago
University of Cincinnati
Illinois Institute of Technology
University of Illinois
Indiana University
Iowa State University
The University of Iowa
Kansas State University
The University of Kansas
Loyola University
Marquette University
Michigan State University
The University of Michigan
University of Minnesota
University of Missouri
Northwestern University
University of Notre Dame
The Ohio State University
Ohio University
The Pennsylvania State University
Purdue University
Saint Louis University
Southern Illinois University
University of Texas
Washington University
Wayne State University
The University of Wisconsin

LEGAL NOTICE

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:

A. Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or

B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method, or process disclosed in this report.

As used in the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission, or employee of such contractor, to the extent that such employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission, or his employment with such contractor.

Printed in the United States of America
Available from
Clearinghouse for Federal Scientific and Technical Information
National Bureau of Standards, U. S. Department of Commerce
Springfield, Virginia 22151
Price: Printed Copy $3.00; Microfiche $0.65

ANL/ES-CC-006
Meteorology

ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

CHICAGO AIR POLLUTION SYSTEMS ANALYSIS PROGRAM

ARGONNE NATIONAL LABORATORY

CHICAGO DEPARTMENT OF ENVIRONMENTAL CONTROL

DEPARTMENT OF HEALTH, EDUCATION, AND WELFARE
National Air Pollution Control Administration

APICS

A Computerized Air Pollution Data Management System

by

C. Chamot,* A. S. Kennedy,* E. J. Croke,
J. J. Roberts, H. Moses,** and J. B. Anderson*

Center for Environmental Studies

February 1970

* Applied Mathematics Division
** Radiological Physics Division

The Chicago Air Pollution Systems Analysis Program

The Chicago Air Pollution Systems Analysis Program is a joint effort conducted and funded by the Atomic Energy Commission, the National Air Pollution Control Administration and the Chicago Department of Environmental Control.

Contract management is provided by the Division of Meteorology of the National Air Pollution Control Administration. Program direction, meteorology studies, computer programming, and model development are provided by Argonne National Laboratory. The Chicago Department of Environmental Control supplies emission and air quality data for the atmospheric dispersion studies.

Program Director

E. J. Croke - Argonne Center for Environmental Studies

Policy Review Committee

W. Stanley - Director, Chicago Department of Environmental Control

L. Link - Director, Argonne Center for Environmental Studies

R. Engelmann - Division of Biology and Medicine, U.S. Atomic Energy Commission

R. McCormick - National Air Pollution Control Administration

Technical Review Committee

S. Booras - Environmental Pollution Control Systems

L. Niemeyer - National Air Pollution Control Administration

FOREWORD

One of the more difficult problems which confronts the environmental scientist, the urban and regional planner, and the environmental control agency official is associated with the organization, storage, retrieval, processing, analysis, and display of the very large and diverse inventory of data that is often required to characterize an environmental system. It is not, perhaps, surprising that the environmentalist should turn to the techniques of computerized data management, but in doing so, he has presented the computer systems analyst with a somewhat more complex problem than has hitherto been encountered in the development of data management systems. This is due largely to the enormous diversity, in kind and quantity, of the data with which the environmentalist seeks to deal, and is quite symptomatic of the interdisciplinary nature of most environmental problems. Another special aspect of environmental data management is associated with the tendency of the analyst or planner to seek to couple his data bank with one or more relatively complex physical or economic mathematical models of environmental phenomena or systems.

In the field of air resource management and planning, three distinct types of information must be manipulated in order to devise effective air pollution control implementation plans. These are emission inventory data, meteorological observations, and calculated or measured air quality data.

It is often desirable, for purposes of analysis and planning, to partition this data into subsets, to perform statistical correlation studies of raw or processed data, to couple the data bank with an atmospheric dispersion model which can be used to calculate ambient air quality throughout an urban or regional area, and to display the results of these computations in graphical or tabular form. The Chicago Air Pollution Systems Analysis Program required that a data bank including nearly four years of hourly emission inventory, meteorology and air quality information be managed and analyzed in this way.

In order to facilitate the storage, retrieval, processing, analysis, and display of this extremely large inventory of data, an operational air pollution information and computation system was developed. This system, dubbed APICS (Air Pollution Information and Computation System), consists of the IBM PL-1 data storage and retrieval code nested in a series of subroutines and computational codes tailored to the analysis of air pollution data. With the APICS system, the data stored in a master file can be partitioned in any way desired. Any combination, array or subset of emission, meteorology, and/or air quality data can be formed and retrieved. The selected data array can be accessed through a FORTRAN subroutine which allows the user to manipulate the array in any desired way. For example, the analyst may form a new data array which consists of products of components of the original array raised to some power, etc. Standard subroutines for computing atmospheric stability, smoke plume rise, smoke plume dispersion, etc., are included as options in the APICS system. The array of data in the master APICS file may also be analyzed in a multivariate linear regression analysis code or a discriminant analysis code which are part of the system and the results may be displayed as tables, histograms, maps, or time series plots.

Once completed and operational, the APICS system proved to be an invaluable analytical tool for the meteorology, systems analyses, mathematical modeling, and planning studies conducted by the Argonne air pollution research team. We believe that the system is well adapted to the creation of an urban or regional air pollution data bank.

Although the development of APICS involved a relatively high level of technical expertise in the field of computer systems analysis, the code itself has been designed for the use of analysts who have only a very limited acquaintance with computer techniques. Once the program is operational on a given computer, it may be used quite readily by any individual who has mastered the

basics of the FORTRAN language. APICS is therefore an authentic air pollution computer "language" which permits the user to perform a series of complex manipulations by means of a set of relatively simple, standardized commands.

This manual describes APICS in detail, and provides the instructions and computer listings required to render the system operational. It is only fair to caution the reader that the bulk of the information contained herein is provided for the benefit of the programmer assigned to render APICS operational on the computer facility which is available to the prospective user. A consequence of this orientation is that the user himself, whether he be a meteorologist, a systems analyst, an urban or regional planner, or a control agency official, is likely to find much of the APICS manual to be obtuse and difficult. We therefore advise the prospective user to survey this document only to the extent necessary to determine whether or not APICS will serve his needs, and leave the details of implementation to a computer scientist.

Edward J. Croke, Director
Chicago Air Pollution
Systems Analysis Program
Center for Environmental Studies
Argonne National Laboratory

Organization of the Manual

Section 1 provides background information, discusses the philosophy behind the construction of the APICS system, and indicates current applications of the system for air pollution studies. The remainder of the manual describes the system in greater detail.

Section 2 expands on the system design philosophy, summarizes data types, introduces the facilities, and describes the computer configuration within which the system operates. Reference should be made to Fig. 2.2 in order to follow the flow of information through the system. The programs provided for the storage and retrieval of information are described in subsections entitled:

Data File Preparation
Master File Maintenance
File Search for Specified Data Items
Accessing Retrieved Data
Data Display Program
Analysis and Computation Programs

References to the more detailed sections of the manual are provided for each of these.

Sections 3 to 8 expand on each of these functions and contain detailed directions for using the various programs.

The Appendices contain flowcharts, sample deck structures, and listings of programs. The Appendix section of the manual should serve as a reference for using the system once its capabilities are understood by the prospective user.

Extending the System

As indicated in Sections 1 and 2, the system has been so constructed that it is open-ended with respect to both data and programs. Instructions for adding new data types are provided in Sections 2.2.2, 3.1, and 4.3; and instructions for adding new programs to operate on the retrieved data are provided in Sections 2.2.5 and 6.0.

Availability of the System

All of the programs described in the manual are available at Argonne National Laboratory on magnetic tape with each program in a separate file. Source programs of general purpose subroutines for dynamic storage allocation and PLI/FORTRAN interfacing are included on the tape. A copy of the master tape may be obtained, at cost, by contacting the Argonne Center for Environmental Studies.

TABLE OF CONTENTS

		Page
	The Chicago Air Pollution Systems Analysis Program	iii
	Foreword	iv
	Organization of Manual	vii
	Abstract	xvii
1.0	Introduction	2
2.0	System Overview	6
2.1	The System Design	6
2.2	Operational Facilities of APICS	8
2.2.1	Master File Format and Organization	10
2.2.2	Data File Preparation	10
2.2.3	Master File Maintenance	12
2.2.4	Search for Specified Data Items	13
2.2.5	Accessing Retrieved Data	13
2.2.6	Data Display Programs	14
2.2.7	Analysis and Computation Programs	14
2.3	Current Implementation of APICS	14
3.0	Master Data File Preparation	18
3.1	General Data Input Program (CREATE5)	27
3.2	Calendar Program (CREATE9)	29
3.3	Air Quality Data (CREATE3)	30
3.4	Meteorological Data (CREATE1 and LITTLEC1)	33
3.5	Functions of File Parameters (TURSTAB)	35
3.6	Specialized Data Input Program (MASTER2)	37
3.7	Point Source Emission Data (PLANTSIM)	39
3.7.1	Industrial Source Simulation	39
3.7.2	Utility Source Simulation	46
3.7.3	Operation of Emission Simulation Program	53
4.0	Master File Maintenance	58
4.1	Merging Programs	58
4.1.1	MERGE1	59
4.1.2	MERGE2	61
4.1.3	MERGE3	61
4.2	The SUBFIL Program	62
4.3	The FUNCTION Program	64

TABLE OF CONTENTS

		Page
5.0	Master File Search	70
5.1	Formats of Search Input and Output Tapes	75
5.1.1	Master File Format	76
5.1.2	Search Output Format	77
5.2	SEARCH1	77
5.3	SEARCH2	78
5.4	SEARCH3	79
6.0	Access of Retrieved Data	82
6.1	Input and FORT Routines for the FORTRN Program	85
6.2	Input and FORT Routines for the FORTRNS Program	86
6.3	Input and FORT Routines for the FORTRNK Program	86
7.0	Display Programs	88
7.1	PRINT Data Display Program	89
7.2	Time Series Plotting Program (YOLYPLOT)	91
7.2.1	Programming Notes on the Time Series Program	97
7.2.2	YOLYPLOT Programming Information	98
7.3	Mesometeorological Mapping Program	100
7.4	Printed Graphical Displays	105
7.4.1	Use of Subroutine JJPLOT	108
7.4.2	Use of Subroutine HIST	109
8.0	Computational Programs	112
8.1	Linear Regression Analysis	113
8.2	Stepwise Multiple Discriminant Analysis	117
8.3	Tabulation Prediction Program	131
8.4	Atmospheric Diffusion Calculations	134

TABLE OF CONTENTS

		Page
9.0	The Tabulation Technique for Forecasting Concentrations of Air Pollutants	142
9.1	Description of the Tabulation Prediction Technique	142
9.2	The Influence of Meteorological Variables on Air Pollutant Concentrations	144
9.3	Selection of Variables	144
9.4	Construction of the Tabulations	149
9.5	Construction of a Simplified Version of the Tabulation Prediction Systems	152
9.6	Application of the Tabulation Prediction System to Daily Operations	153
9.7	Advantages and Disadvantages of the System	158
9.8	Tabulation Prediction System Computer Program	158
9.8.1	Input Information	158
9.8.2	Program Description	160
9.8.3	Tabulation Prediction System Computer Listing	165
Acknowledgments		166
References		167
Appendix A.	Guides to Using the APICS System: Schematic Flow Charts.	A-1
Appendix B.	Specifications for Using the Individual Codes in the APICS System (One Page Summary for Each Code)	B-1
Appendix C.	Data Preparation Programs: Source Listings	C-1
	1. General Data Input Program (CREATE5)	C-2
	2. Calendar Program (CREATE9)	C-5
	3. Air Quality Data (CREATE3)	C-8
	4. Meteorology Data Preprocessor (LITTLEC1)	C-12
	5. Meteorology Data (CREATE1)	C-13

TABLE OF CONTENTS

		Page
	6. Functions of File Parameters (TURSTAB)	C-20
	7. Specialized Data Input Program (MASTER2)	C-26
	8. Emission Data from a Theoretical Source (PLANTSIM)	C-28
Appendix D.	Master Tape Maintenance Programs: Source Listings	D-1
	1. Merging Program (MERGE1)	D-2
	2. Merging Program (MERGE2)	D-4
	3. Merging Program (MERGE3)	D-6
	4. Subfile Generation (SUBFIL)	D-10
	5. Functions of Parameters (FUNCTION)	D-14
Appendix E.	Search Programs: Source Listings and Sample Input	E-1
	1. SEARCH1	E-2
	2. SEARCH2	E-8
	3. SEARCH3	E-24
Appendix F.	Accessing Data from SEARCH: Source Listings and Programming Notes	F-1
	1. Retrieval Program (FORTRN)	F-2
	2. Retrieval Program (FORTRNS)	F-3
	3. Retrieval Program (FORTRNK)	F-6
Appendix G.	Data Display Programs: Listings, Deck Setups for SEARCH, and Sample Output	G-1
	1. Printing of Selected Data (PRINT)	G-2
	2. Plotting of Time Series Graphs (TIMESER and YOLYPLOT)	G-6
	3. Mesometeorological Mapping (CITYMAP)	G-9
	4. Printed Histograms and Two-Dimensional Frequency Distributions (HIST and JJPLOT)	G-12

TABLE OF CONTENTS

Page

Appendix H. Computational Programs: Listings, Deck Setup for SEARCH, and Sample Output H-1

1. Linear Regression Program (REGRES). H-2

2. Multiple Discriminant Analysis (STEPMDA). H-12

3. Tabulation Program (TABPRED). H-33

4. Atmospheric Diffusion Calculation (PUFF). H-38

FIGURES

		Section	Page
2.1	The Structure of Information and Computation Systems	2.1	7
2.2	Information Flow through APICS	2.2	9
2.3	Sample Master File Record Format	2.2.1	11
3.1	Industrial File Card Formats	3.7	41
3.2	Utility File Card Formats	3.7	47
3.3	Typical Gross Thermal Input vs. Load Curve and Least Squares Approximating Line	3.7	51
3.4	Input-Output Diagram of Plant Simulation Program	3.7	54
4.1	Data Translation and Master File Update	4.1	60
4.2	Subfile Generation	4.2	63
5.1	Master File Data Search	5.0	71
6.1	Analysis of Retrieved Data	6.0	83
7.1	Time Series Plots for TAM Data	7.2	92
7.2	Time Series Plots for Weather Data	7.2	93
7.3	Mesometeorological Map for Chicago	7.3	101
7.4	Two-D Frequency Distribution of SO_2 Concentration vs. Temperature	7.4	106
7.5	Histogram Plot of Temperature Distribution	7.4	107
8.1	Regression Analysis	8.1	114
8.2	Discriminant Analysis	8.2	118
8.3	Input-Output Diagram of Dispersion Model Program	8.4	135
9.1	Sample Tabulation Prediction System Data Display	9.1	143
9.2	SO_2 Concentrations vs. Wind Direction at Selected Percentile Values	9.3	145
9.3	SO_2 Concentrations vs. Time at Selected Percentile Values	9.3	146
9.4	Tabulation Prediction System Bands for Wind Direction and Hour of Day	9.3	148

FIGURES (Continued)

		Section	Page
9.5	Units and Group Intervals for Variables Used in the Tabulation Prediction System	9.4	150
9.6	"Standard" Intervals for Wind Direction and Time Bands	9.5	152
9.7a	Air Quality Prediction System Data Form AIR QUALITY PREDICTION SYSTEM	9.6	154
9.7b	Air Quality Prediction System Data Form SO_2 PREDICTIONS AND ACTUAL READINGS	9.6	155
9.8	Information Requirements of Tabulation Prediction Program	9.8	159
9.9	Tabulation Prediction System Computer Program Schematic	9.8	161

TABLES

		Section	Page
I.	Summary of the Uses of APICS at Argonne	1.0	4
II.	Master File Table of Contents	3.0	19
III.	Meteorological Data Codes	3.0	21
IV.	Tape Format for TAM Data	3.3	31
V.	Tape Format for Airways Surface Observations	3.4	34
VI.	Industrial File Card Descriptions	3.7.1	42
VII.	Utility File Card Descriptions	3.7.2	48

APICS

A Computerized Air Pollution Data Management System

Abstract

A computerized system was developed as an aid to research associated with the Chicago Air Pollution System Model Project at Argonne National Laboratory. The acronym selected, APICS, stands for Air Pollution Information and Computation System. This system includes programs to create and maintain master files of meteorological and air pollution data; to search master files for desired information; and to pass the data to the various analysis and computation programs. A multiple source atmospheric dispersion model program may be used to calculate projected pollution levels. Other programs provide graphical and printed display of selected data, and results of statistical analysis, including tables for predicting local pollution levels based on meteorological phenomena.

The manual documents those aspects of the system which may be generally applied to similar situations in other cities, and indicates how other data types and analysis programs could be incorporated into the system.

APICS USER MANUAL

1.0 Introduction

1.0 Introduction

This manual describes a computerized data management and analysis system which was developed during the course of the Chicago Air Pollution Systems Analysis Program conducted by Argonne National Laboratory. The acronym selected, APICS, stands for Air Pollution Information and Computation System.

Air resource management planning frequently involves the acquisition, storage retrieval, processing, and analysis of very large inventories of emission, meteorological, and air quality data.[1-3] The Chicago Air Pollution Systems Analysis Program was confronted with such a data management problem. In order to render tractable the voluminous data base available for the Chicago megalopolis, a computerized system of data collection, storage, and retrieval was implemented. The system includes programs to:

1) Preprocess data on weather observations and on pollutant emission and retention in the air, to create master files,

2) Update and combine the master files,

3) Search the master tapes for desired information,

4) Display the information, and

5) Perform mathematical analyses of emission, meteorological, and air quality data.

The primary purpose of this manual is to document those aspects of the system which may be generally applied to similar studies in other cities or regions. However, we believe that concepts and philosophies embodied in this system can be extended to the data management and analytical problems associated with other areas of environmental science.

The principal design criterion for the development of APICS was flexibility in the following two respects:

1) The ability to incorporate data in the master file from sources that could not be anticipated in advance;

2) The facility to add new computational programs to operate on the stored data base.

The various studies which have been performed using APICS at Argonne indicate that this criterion has been satisfied. New programs continually added to the system by users have increased its capability.

The emphasis placed on a well-planned system from the outset of the project has contributed significantly to the successful operation of APICS. This system design resulted from the combined efforts of participating analysts and computer system design personnel in drawing up system specifications.

The applications of the system in the City of Chicago Air Pollution Systems Analysis Program are listed in Table I. Those programs which are of a general nature have been incorporated into this manual.

Table I

A Summary of Uses of APICS*

Category	Description	Reference
Data Retrieval	1. Selective searches for meteorological phenomena which affect air pollution concentrations such as inversions, fumigations, lake breezes, stagnations, etc.	[5]
	2. Time series charts of air pollution and meteorological variables for 1966 and 1967	[3]
	3. Pollution incident identification studies	[3-5]
Data Analysis	1. Pollution incident frequency studies	[3]
	2. Air pollution percentile tabulations as a function of meteorological variables	[5]
	3. Wind direction frequency studies	[3]
	4. Emission source fuel use regression analysis study	[2,3]
	5. Analysis of adverse medical effects of SO_2	[5]
Prediction Models	1. Development of receptor-oriented air quality forecasting and analysis models based on linear regression analysis	[3]
	2. Multiple discriminant analysis air quality prediction system	[5]
	3. Validation of multiple source atmospheric dispersion model	[5]
Air Pollution Control Resource Allocation Models	1. Power plant system optimal fuel switching and load shifting control model	[3]
	2. Gas allocation model for industrial plants with dual fuel capability	[5]
	3. Computation of natural gas demand for dual fuel switching during pollution "season"	[5]

*The APICS applications listed are among those devised for the Chicago Air Polluti
Systems Analysis Program. The potential uses of the system are not restricted to
these, and are constrained primarily by the ingen

APICS USER MANUAL

2.0 A System Overview

2.0 A System Overview

2.1 The System Design

The Chicago Air Pollution Systems Analysis Program involved the development of a multiple source atmospheric dispersion model, the development of episode abatement strategies, control resource management, and long term air resource management planning. The APICS system played a major role in all of these studies.

One way to visualize the systems approach to the air resource management planning problem is depicted in Fig. 2.1. The system to be managed yields data (measurements, observations, complaints, etc.) which indicate the prevalent air pollution situation of the city or region. This data inventory includes air quality and meteorological information provided by a TAM* network, weather information provided by U.S. Weather Bureau stations, and industrial, residential, institutional, and commercial emissions based on fuel use records and operational schedules.

By applying medical, engineering, meteorological, and economic analytical techniques to this data base, a thorough understanding of the air resource management problem can be achieved, and feasible alternative air resource management strategies can be developed within the current and projected physical, economic, and sociopolitical constraints imposed on the city or region.

Fig. 2.1 suggests four general categories of analysis which may be performed as part of an air resource management planning study. These are as follows:

1) Data retrieval - interrogation of the data file to obtain a better understanding of the conditions which produced the observed data; to search for patterns and relationships between critical parameters; or to reveal significant anomalies, variations, etc.;

---

*Telemetered Air Monitoring Network - A system of air quality monitoring stations measuring wind speed, wind direction, and air pollution concentrations.

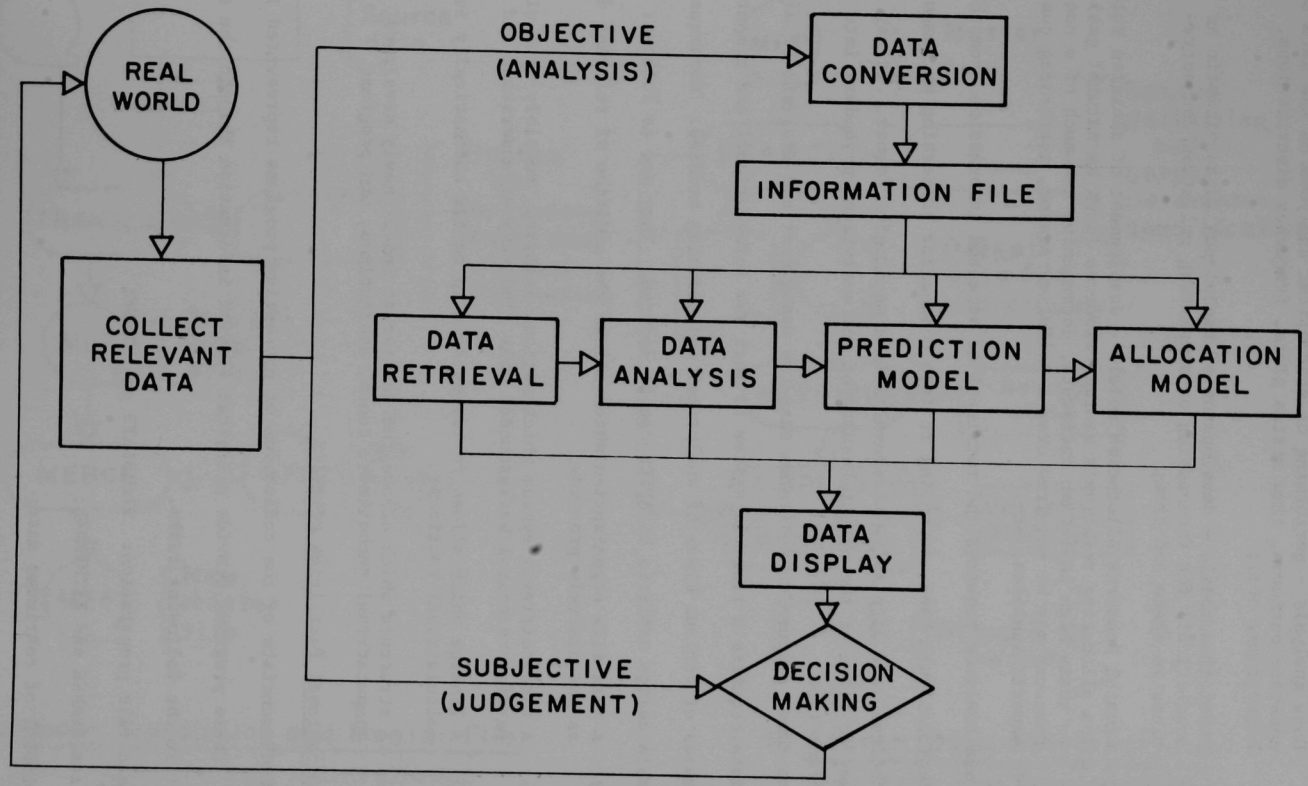


Fig. 2.1 The Structure of Information and Computation Systems

113-1138

2) Data analysis - performing of statistical analyses on the data to generate patterns, time series plots, frequency distributions, histograms, etc.;

3) Prediction models - development of techniques (deterministic or statistical) for forecasting or simulating pollution concentrations in space and time;

4) Control Resource Allocation models - development of decision rules for allocating relatively scarce resources (such as natural gas) to abate high pollution incidents, determining how much of a control resource may be required over a pollution season, performing cost-benefit analyses, etc.

Each of these analyses requires the retrieval of selected information from the urban or regional data base, funneling it into a specific processing or computational technique, and displaying the results in a meaningful format. It may also be be necessary to transfer the output results of one analytical procedure into another of a different type.

The above discussion indicates that the management and analysis of air resource data requires a flexible system to meet the informational and computational demands of various kinds of analytical and planning studies. Consequently, the desirable design criteria of APICS can be deduced. They are as follows:

1) A data file organization which allows the addition of relevant data as the analysis proceeds;

2) A data retrieval system which various analysts, possibly with widely varying technical backgrounds, will find easy to understand and use.

3) A linkage which allows retrieved data to be fed automatically into calculational routines;

4) A structure which allows the analyst to insert newly developed computational techniques, tools, subroutines, and programs.

2.2 Operational Facilities of APICS

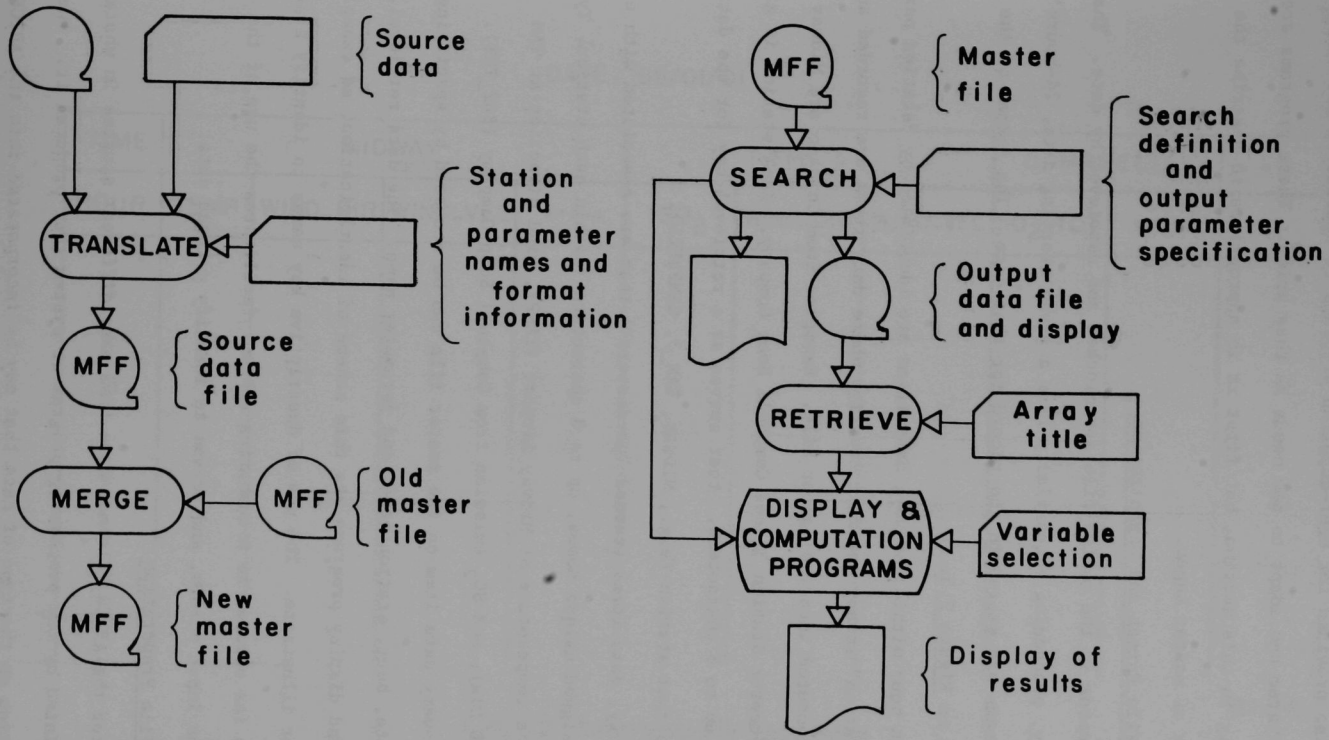
APICS consists of the collection of processing programs represented in Fig. 2.2. These programs provide a logical flow of information through the system to accomplish the following tasks.

Data file preparation: TRANSLATE and MERGE;

Data SEARCH and RETRIEVAL;

DISPLAY of retrieved data;

COMPUTATION and analysis operations on the retrieved data.



(A) Data Translation and Master File Update and Maintenance

(B) Search Data Retrieval, and Analysis and Computation

113-1139

Fig. 2.2 Information Flow through APICS

A facility is also provided for user-supplied programs to operate on the data, so that the APICS system can adapt to new needs as they arise. These programs are discussed briefly in this section, but first it is appropriate to describe the data file format of master tapes.

2.2.1 Master File Format and Organization

Data records in the master file are blocked and retrieved by date. That is, for each day, proceeding sequentially from a given starting date, 24-hours' worth of data items are stored in the record for that date. The format of the record is shown in Figure 2.3.

For each hour within the day, data items are identified by "station parameters." A "station" is any physical location where data items are recorded and are to be incorporated into the master file. Examples from Chicago are Midway Airport, TAM receptor Station 3, and Campbell Soup Company. Each station is given a unique name, up to 8 characters, that serves as a retrieval key for the data associated with that station (e.g., MIDWAY, TAM_3, CAMPBELL).

Similarly, data items (termed "parameters") that are associated with each station are assigned unique names, up to 8 characters (within each station). Typical examples are the temperature at Midway Airport (TEMP), wind direction at TAM station 3 (WIND_DIR), and SO_2 emission from Campbell Soup Company (SO2_TOT).

Thus, every data item on the master file can be accessed by specifying the correct date, hour, station name, and parameter name. The data retrieval, computation, and display programs use this scheme of identification, as examples presented later illustrate. The use of descriptive key names to identify stations and parameters (as opposed to a numbering scheme) facilitates the use of the system, reduces input errors, and serves to identify printed data.

2.2.2 Data File Preparation

Data for the APICS file may come from many different sources in whatever format is dictated by the processing program or system that produced it. There is no restriction on the type of data that may be incorporated into the APICS

DATE: 66/01/01

TIME	MIDWAY				TAM_3			CAMPBELL		...
HOUR	TEMP	WIND_DIR	WIND_VEL	.....	SO_2	WIND_DIR	WIND_VEL	SO_2_TOT	.....	
1										
2										
3										
.										
.										
.										
23										
24										

113-1140

Fig. 2.3 Sample Master File Record Format

system. The first stage of file preparation involves the use of a preprocessing program to translate the data into the standard master file format (MFF). In the standard format, eight-character (or less) station and parameter names are associated with each data item.

A series of computer routines (indicated by the TRANSLATE box in Fig. 2.2A) has been generated to translate data for the following data sources:

1) TAM stations (up to 8);

2) Power plant emission data;

3) Industrial plant emission data;

4) National Weather Records Center meteorological data for airport weather stations.* (Reference 10)

Each additional data source will, in general, require a separate preprocessing program. However, these preprocessing programs have been standardized to the extent that each new program is created from a previous one by merely inserting new station and parameter names and format information. Each of the preprocessing programs mentioned above is described in detail in Section 3. Also see Section 4.3 for calculating new parameters as functions of parameters already existing on master tapes.

2.2.3 Master File Maintenance

When a new set of data has been translated into the master file format, it is input to a merging program (the MERGE box in Fig. 2.2A) along with the old master file to generate a new master file containing the new information. Thus, the new data is available for processing together with all old information. It is possible to generate new master files (or subfiles) containing any desired combination of new and old parameters and dates. Programs for updating, combining, and partitioning the information on master files are described in detail in Section 4.

---

*The Argonne file includes Midway, O'Hare, Meigs, and Glenview Airport data.

2.2.4 File Search for Specified Data Items

The first step in accessing data on a master tape is to use a searching program (the SEARCH box in Fig. 2.2B) to search the master file for the desired information. Search specifications may include time delays and date, hour, and parameter ranges of interest. For example, an array containing hourly average air quality data combined with hourly emission data for the preceding hours can be formed and retrieved.

The resulting output array is then available on tape for further processing with a title and the eight-character (or less) station and parameter names identifying the array. Some of the programs discussed in this manual can operate on arbitrary sets of data, and some require a specific set of data from the search run. Details of how a search of the master file is accomplished are provided in Section 5.

2.2.5 Accessing Retrieved Data

The result of a sequence of such search passes through the master file is a corresponding set of output data arrays (each identified by a unique title) on the output data file. These data arrays are then retrieved from tape (the RETRIEVE box in Fig. 2.2B), and are, in turn, input to display or analysis and computation programs. The mechanisms for inputting retrieved data and for passing it to user-coded FORT routines is described in Section 6, where directions are also provided for incorporating additional programs into the system.

Appendix F provides listings of the retrieval programs. See the programs described in Sections 7 and 8 for examples of FORT routines. Listings for these are provided in Appendices G and H.

It is also possible to bypass the RETRIEVE box (in Fig. 2.2B) and to pass the information in the output array directly to a FORT subroutine. See Sections 5 and 6.1 for further information on this feature of the system.

2.2.6 Data Display

The results of a search run are displayed (optionally) on a standard printing device. Additional display features (described in detail in Section 7) include the following:

1) Printed display of master file parameters for selected dates;

2) Graphical display (time series) of master file parameters;

3) Mesometeorological maps;

4) Histogram plotting routine;

5) 2-Dimensional frequency distributions.

Also see Appendix G for listings.

2.2.7 Analysis and Computation Programs

In addition to data display programs, four of the analytical programs cited in Table I are of a general nature and have been documented in Section 8 of this manual. Also see Appendix H for program listings. The programs included are as follows:

1) Linear Regression Analysis

2) Multiple Discriminant Analysis

3) Tabulation prediction system for short range air quality forecasting

4) Atmospheric Diffusion Model Calculation

2.3 Current Implementation of APICS

APICS is currently implemented on the IBM 360/75 computer and operates under MVT using the OS/360 operating system. Standard input/output is routed through a Model 50 under control of ASP (IBM's Attached Support Processor). Programs are coded in PL/1 and Fortran. In general, PL/1 has been used for the data manipulation tasks and FORTRAN for computational programs.

The master file is a sequential data set stored on 9 track, 800 bpi magnetic tapes. Current practice is to store two years' "worth" of data on one tape. Argonne's master file is virtually complete for the years 1966 and 1967. At this writing, data for the years 1968 and 1969 is being collected and stored.

Many of the APICS programs have been stored on the data cell of Argonne's 360 system. Directions for link-editing APICS programs onto the data cell and for executing programs from the data cell are provided in Appendix A. Further information for using the data cell is contained in Reference (15).

APICS USER MANUAL

3.0 Master Data File Preparation

3.0 Master Data File Preparation

A separate code has been written for each type of data in order to convert this information into the master file format. Tapes containing data which has been generated in the master file format are referred to as master tapes. These master tapes may then be updated, combined, and partitioned as desired by using the codes described in the next section under "Master File Maintenance." Tables II and III define the data items on Argonne's Master File.

This section provides details on using the preprocessing codes for the following data types:

1) Arbitrary station and parameter values recorded on punched cards

2) Calendar information; e.g.: day, month, year, etc.

3) Air quality data for up to eight TAM stations

4) Airways surface meteorological observations for individual U.S. Weather Bureau stations

5) Atmospheric stability classes calculated as a function of meteorological data on a master tape

6) Precalculated (mixing-height) values on punched cards

7) Point source emission data based on a simulation of plant operating characteristics

Refer to Appendix B for card deck setups and specifications for running these programs.

Most of the preprocessing programs listed above are simply minor modifications of others. Items such as date limits and number of days in a month, etc., may have to be changed from one run to the next. The programs for card input (1 and 6 above) are similar, but "1" is a more general input program that reads date limits and station and parameter names from cards. The facility for incorporating mixing height information is included here, since this data, along with source emission data, are regarded as particularly critical elements in an air pollution data analysis and because both are required as inputs to the atmospheric dispersion model described in Section 8.4.

TABLE II: Master File Table of Contents

Station	Parameter	Range	Description	Unit
CALENDAR	HOUR	0→23	Hour, Local Standard Time(24 hour clock) 00 through 23 CST(or local standard time)	Hours
	DAY_WEEK	1→17	Day of Week;(1 through 7) or for Holidays; 10+Day of Week (Monday is the 1st day of the week)	Days
	YEAR	66→69	Year; last 2 digits only	Years
	DAY_YEAR	1→366	Julian Day; 1 through 365 Leap Year; 1 through 366	Days
	MONTH	1→12	Month of Year	Months
	DAY_MON	1→31	Day of Month	Days
Airport Weather Stations:	WIND_DIR	0→360	Wind Direction to nearest 10 deg.[a]	Degrees
	WIND_VEL	0→100	Wind Speed	Knots
MIDWAY	TEMP	-25→105	Dry Bulb Temperature	Degrees F
MEIGS	CEIL	0→999	Ceiling Height[b]	Hundreds of Ft.
GLENVIEW	VIS	0→16	Visibility to nearest .125 mile(s)	Statute mi.
OHARE	DEW_PT	-40→100	Dew Point Temperature to nearest degree with Respect to Water	Degree F
	REL_HUM	0→100	Relative Humidity to nearest percent with Respect to Water	Percent
	PRESS	28→32	Station Pressure	Inches of Mercury
	SKY_CON1 SKY_CON2 SKY_CON3 SKY_CON4	0→11	Sky Condition in each of 4 sky layers (lowest to highest)[c]	Code
	CLD_AMT1 CLD_AMT2 CLD_AMT3 CLD_AMT4	0→10	Cloud Cover in each of the 4 cloud Layers	Tenths of sky covered
	CLD-TYP1 CLD_TYP2 CLD_TYP3 CLD_TYP4	0→20	Cloud Type in each of 4 cloud Layers[d]	Code

Note: Superscript letters refer to explanatory sections of Table III.

TABLE II: Master File Table of Contents (Cont'd.)

Station	Parameter	Range	Description	Unit
Airport Weather Stations:	CLD_HT1 CLD_HT2 CLD_HT3 CLD_HT4	0→888	Cloud Height within each of 4 cloud Layers[e]	Hundreds of Ft.
	CLD_SUM1 CLD_SUM2 CLD_SUM3 CLD_SUM4	0→1	Cumulative fractional cloud cover in the lower "n" layers. Thus CLD_SUM4 is the total cloud cover.	Fraction
	WIND_PH	0→9	Wind Phenomena[f]	Code
	LIQ_PCP1 LIQ_PCP2	0→9	Liquid Precipitation in each of 2 classes[gh]	Code
	FRO_PCP1 FRO_PCP2 FRO_PCP3	0→9	Frozen Precipitation in each of 3 classes[ijk]	Code
	OBS_VIS1 OBS_VIS2	0→9	Obstruction of Visibility[ℓm]	Code
TAM Stations:	SO_2	0→2	Average of Five 15-minute Sulfur Dioxide Measurements	ppm
TAM_1 TAM_2 . . . TAM_8	WIND_VEL	0→100	Wind Speed (average of five measurements)	Knots
	WIND_DIR	0→360	Wind Direction to nearest degree[a] (average of five measurements as described in Section 3.3)	Degrees
TURNER	TURNER_C	1→5	Atmospheric Stability class index to indicate unstable to stable conditions[n]	Code
STAB	TURNER_A	1→225	Atmospheric Stability class which Identifies the Two Hours about Sunrise 10i and Sunset 20i[o]	Code
	TURNER_B	100→515	Atmospheric Stability class which Identifies the two hours about Sunrise i03 and Sunset i13[p]	Code
NORCO	MIX_HT	0→7000	Mixing Height	Feet

Note: Superscript letters refer to explanatory sections of Table III.

TABLE III. Meteorological Data Codes

Parameter	Definition
***	Missing value stored as 10^{20} = 1.0E+20
a. WIND_DIR	Wind direction from 0 to 360 degrees increasing clockwise and ending with North
	0 = Calm
	90 = East wind
	180 = South wind
	270 = West wind
	360 = North wind
b. CEIL	Ceiling height in hundreds of feet
	888 = Ceiling formed by cirroform clouds of unknown height
	999 = Unlimited ceiling
c. SKY_CONn	Sky condition for the nth layer where
	n = 1 to 4 starting with the lowest layer
	0 = Clear or less than 1/10 sky cover
	1 = Thin scattered 1/10 - 5/10 sky cover
	2 = Scattered 1/10 - 5/10 sky cover
	(3)= (Dark scattered 1/10 - 5/10 sky cover)
	4 = Thin broken 6/10 - 9/10 sky cover
	5 = Broken 6/10 - 9/10 sky cover
	(6)= (Dark broken 6/10 - 9/10 sky cover)
	7 = Thin overcast 10/10 sky cover
	8 = Overcast 10/10 sky cover
	(9)= (Dark overcast 10/10 sky cover)
	10 = Obscuration
	11 = Partial obscuration

( ) not reported recently.

TABLE III. Meteorological Data Codes (Cont'd.)

Parameter	Definition
d. CLD_TYPn	Cloud type for the nth layer where n = 1 to 4 starting with the lowest layer 0 = none 1 = Fog 2 = Stratus 3 = Stratocumculus 4 = Cumulus 5 = Cumulonimbus 6 = Altostratus 7 = Altocumulus 8 = Cirrus 9 = Cirrostratus 12 = Stratus Fractus/Fractostratus 14 = Cumulus Fractus/Fractocumulus 15 = Cumulonimbus mamma 16 = Nimbostratus 17 = Altocumulus castellanus 19 = Obscuring phenomenon 20 = Obscuring phenomenon other than fog.
e. CLD_HTn	Cloud height for the nth layer where n = 1 to 4 starting with the lowest layer XXX = Height in hundreds of feet. above station level recorded as follows to the nearest 100 ft up to 5000 ft, to the nearest 500 ft between 5000 and 10,000 ft, and to the nearest 1000 ft above 10,000 ft. 888 = Cirroform clouds of unknown height
f. WIND_PH	Wind phenomena 0 = No thunderstorm, tornado, squall, or other listed phenomena 1 = Thunderstorm (T) 2 = Heavy thunderstorm (T+) 3 = Tornado (TORNADO) 4 = Light squall (Q-) 5 = Moderate squall (Q) 6 = Heavy squall (Q+)

TABLE III. Meteorological Data Codes (Cont'd.)

Parameter	Definition
g. LIQ_PCP1	Liquid precipitation, Class 1
	0 = No rain, rain showers, or freezing rain
	1 = Light rain (R-)
	2 = Moderate rain (R)
	3 = Heavy rain (R+)
	4 = Light rain showers (RW-)
	5 = Moderate rain showers (RW)
	6 = Heavy rainshowers (RW+)
	7 = Light freezing rain (ZR-)
	8 = Moderate freezing rain (ZR)
	9 = Heavy freezing rain (ZR+)
h. LIQ_PCP2	Liquid precipitation, Class 2
	0 = No drizzle, freezing drizzle, or rain squalls
	1 = Light rain squalls (RQ-)
	2 = Moderate rain squalls (RQ)
	3 = Heavy rain squalls (RQ+)
	4 = Light drizzle (L-)
	5 = Moderate drizzle (L)
	6 = Heavy drizzle (L+)
	7 = Light freezing drizzle (ZL-)
	8 = Moderate freezing drizzle (ZL)
	9 = Heavy freezing drizzle (freezing drizzle (ZL+)
i. FRO_PCP1	Frozen precipitation, Class 1
	0 = No snow, snow pellets, or ice crystals
	1 = Light snow (S-)
	2 = Moderate snow (S)
	3 = Heavy snow (S+)
	4 = Light snow pellets (SP-)
	5 = Moderate snow pellets (SP)
	6 = Heavy snow pellets (SP+)
	7 = Light ice crystals (IC-)
	8 = Moderate ice crystals (IC)
	9 = Heavy ice crystals (IC+)

TABLE III. Meteorological Data Codes (Cont'd.)

Parameter	Definition
j. FRO_PCP2	Frozen precipitation, Class 2
	0 = No showers, snow grains, or snow squalls
	1 = Light snow showers (SW-)
	2 = Moderate snow showers (SW)
	3 = Heavy snow showers (SW+)
	4 = Light snow squall (SQ-)
	5 = Moderate snow squall (SQ)
	6 = Heavy snow squall (SQ+)
	7 = Light snow grains (SG-)
	8 = Moderate snow grains (SG)
	9 = Heavy snow grains (SG+)
k. FRO_PCP3	Frozen precipitation, Class3
	0 = No sleet, hail or small hail
	1 = Light sleet or sleet showers (E-, EW-)
	2 = Moderate sleet or sleet showers (E, EW)
	3 = Heavy sleet or sleet showers (E+, EW+)
	4 = Light hail (A-)
	5 = Moderate hail (A)
	6 = Heavy hail (A+)
	7 = Light small hail (AP-)
	8 = Moderate small hail (AP)
	9 = Heavy small hail (AP+)
ℓ. OBS_VIS1	Obstructions to visibility, Class 1
	0 = None listed below
	1 = Fog (F)
	2 = Ice fog (IF)
	3 = Ground fog (GF)
	4 = Blowing fog (BD)
	5 = Blowing sand (BN)
m. OBS_VIS2	Obstructions to visibility, Class 2
	0 = None listed below
	1 = Smoke (K)
	2 = Haze (H)
	3 = Smoke and haze (KH)

TABLE III. Meteorological Data Codes (Cont'd,)

Parameter	Definition
	4 = Dust (D)
	5 = Blowing snow (BS)
	6 = Blowing spray (BY)
n. TURNER_C	Stability class index from 1 to 5 indicating unstable to very stable weather conditions calculated as described in Section 3.5.
o. TURNER_A	TURNER_C stability class index i combined with sunrise and sunset information
	08i = 2 hours before sunrise 09i = 1 hour before sunrise 10i = sunrise 11i = 1 hour after sunrise 12i = 2 hours after sunrise
	18i = 2 hours before sunset 19i = 1 hour before sunset 20i = sunset 21i = 1 hour after sunset 22i = 2 hours after sunset
	00i = all other hours
p. TURNER_B	TURNER_C stability class index i combined with sunrise and sunset information
	i01 = 2 hours before sunrise i02 = 1 hour before sunrise i03 = sunrise i04 = 1 hour after sunrise i05 = 2 hours after sunrise
	i11 = 2 hours before sunset i12 = 1 hour before sunset i13 = sunset i14 = 1 hour after sunset i15 = 2 hours after sunset
	i00 = all other hours

The program for preprocessing air quality (TAM) data could be used in any situation where similar TAM measurements can be made available on tape; i.e.: pollutant, wind direction, and wind speed. The programs for the TAM data include averaging procedures to reduce 15 min. observations to hourly average data.

Tapes of airways surface meteorological observations may be obtained from the National Weather Records Center for any airport of interest. The Chicago Study included data for four airports in the vicinity, so that micro and mesometeorological studies could be carried out to evaluate local situations near specific TAM receptors. The airport weather data program translates each item into appropriate units and checks for missing data.

All values on a master tape are written in "floating point" with a decimal point, and all programs substitute a value of 10^{20} (1.0E+20) for missing data. For the ranges, conventions, and units on such items as day of the week, Julian day, holiday key, stability, cloud height, temperature, pressure, wind direction, wind velocity, mixing height, and SO_2 concentration, reference should be made to Tables II and III. For details on the master file format (for the information on master tapes) see Section 5.1.1.

When executing these programs to create master files, DD cards must be supplied for the input and output datasets SOURCE and NEWFILE, unless otherwise noted. The dsname for SOURCE is normally irrelevant since "raw" data is usually obtained on unlabelled tapes, but the DCB specification must match the format of the information on tape. For sample specifications and deck structures see Appendix B. A sample of concatenating raw data from two tapes is included and Reference (14) gives further information on DD card parameters for accessing multiple files. The DD card parameters for NEWFILE should include an appropriate dsname and DCB parameters of RECFM=VB and BLKSIZE=5000.

In general, the only printed output from these preprocessing programs is the date written for each day as the record is generated for that day. The programs are written in PL1 and include coding in Fortran subroutines to print

the hourly values. However, when executing on the IBM 360 under MVT, the information in the tables would be separated into two parts with the headings given in SYSPRINT and the values printed in the FT06F001 dataset. For the versions of the programs incorporated into the APICS system, the print statements have C's in column 1 so they are treated as comments only. To print the tables of information, simply remove the C's in the first columns of these cards and remove /*....*/ surrounding the CLOSE SYSPRINT statements, etc., near the end of the PL1 codes that call TWODPT (see listings).

For some of the preprocessing programs, the date limits and possibly the station specification must be modified in the source code each time it is used. A complete description of each code is given below. Listings of all of the programs described in this section may be found in Appendix C of this manual.

3.1 General Input Program (CREATE5)

Given date limits, an arbitrary station name, and any number of parameter values for that station, the CREATE5 program generates a complete master tape. All of the information is taken from input data cards in two data sets where the first set is always read with the standard input and is called SYSIN, and the second set may be read from a separate tape if desired. If the second set is to be included with the standard input, it must follow the SYSIN data cards and be preceded by a //GO.SOURCE DD * card.

SYSIN INPUT DESCRIPTION (Note: Information enclosed in parentheses is explanatory only and should not be punched)

TITLE (80 characters - used to identify the printed output only)

(Card of starting and ending dates for header label on the master tape)

'YL/ML/DL','YU/MU/DU' (Lower date, upper date)

(Station definition card)

STATION='NAME ' PARAMS=NP; (NP=Number of parameters)

(Parameter definition cards - one card per parameter)

PARAM=']

SOURCE Input Description

Information for each day must include a date card followed by (3*NP) cards of parameter values as follows, with missing values punched as +1.000E+20.

Card No.	Columns	Content
1	1-8	The date: YY/MM/DD (e.g.: 67/12/31 for December 31 1967).
2 to 4	1-80	The 24 hourly parameter values for the first parameter. Use 3 cards with 8 values per card and 10 columns for each value. FORMAT(8E10.0)
5 to 7	1-80	The 24 hourly parameter values for the second parameter. Use 3 cards as above.
.	.	.
.	.	.
.	.	.
(3·NP-1) to (3·NP+1)	1-80	The 24 hourly parameter values for the last parameter. Use 3 cards as above.

Sample SYSIN and SOURCE Input Data

```
//GO.SYSIN DD *
TEST DUMMY INPUT PARAMETER
'660101','660101'
STATION='DUMMY' PARAMS=2;
PARAM='DUM_1';
PARAM='DUM_2';
//GO.SOURCE DD *
66/01/01
1.0       1.0       1.0       1.0       1.0       1.0       1.0       1.0
1.0       1.0       1.0       1.0       1.0       1.0       1.0       1.0
1.0       1.0       1.0       1.0       1.0       1.0       1.0       1.0
2.0       2.0       2.0       2.0       2.0       2.0       2.0       2.0
2.0       2.0       2.0       2.0       2.0       2.0       2.0       2.0
2.0       2.0       2.0       2.0       2.0       2.0       2.0       2.0
/*
```

3.2 The Calendar Program (CREATE9)

Time and date information is frequently used as a basis for studying and correlating air pollution effects. The CREATE9 program generates master tapes containing 6 such data items as parameter values for the "station" named CALENDAR (see Table II of Section 3.0). The parameter names are HOUR, DAY_WEEK, YEAR, DAY_YEAR, MONTH, and DAY_MON. The second of these parameters is included in order to flag holidays, weekends, etc. The rest of the parameters are self-explanatory. Decimal values are generated for all of these date parameters.

The master tape which results from using this program may be combined with other data on other master tapes to form new masters by using the merge programs described in Section 4. The resulting master tapes may then be searched for desired calendar, weather, and air pollution parameter values as described in Section 5, and the results passed to "user-coded" FORT routines as described in Section 6.

Data searches may be based on date- and hour-limits even if the calendar station information is not included in the master file. However, in this case no time and date information would be provided in the output array from the search. Also, calendar station information must be present to specify and search on HOUR bands, etc. (for example, the analyst might wish to retrieve all data recorded between 1300 and 1400 hours at a given meteorological station).

To execute the program for a particular year (or two), the program must be modified so the "starting" and "ending" dates are correct. Also, the number of days in each month is built-into the program and the figure for February may need to be modified for leap year. Present values are '680101', '691231', '29', and '28'.

The program requires a card deck of input for each year for which a master tape is to be generated. This input consists of ONE title card to identify the printed output followed by a card for each day of the year as follows:

Columns	Contents
1	Normally 0 or blank, but set to 1 for holidays.
2	Day of the week, starting with Monday.
11-14	Day of the year (Julian day = 1 to 366).

Cards for a second year may be included if desired. The title card is omitted for the second year, and it may be convenient to add a constant (e.g.: 1000) to the Julian days for successive years. Then the value of this constant must be "remembered" and incorporated into any FORT routines operating on this data item.

When executing the program, a DD card must be provided for the output tape NEWFILE. See Appendix B for sample deck structure. A listing of the code is given in Appendix C2.

3.3 Air Quality Data (CREATE3)

Air quality data consisting of pollutant concentrations, wind directions, and wind velocities must be available on tape with values given for every 15 minutes of each day starting with midnight. The processing program assumes there are data for eight TAM* stations but the program can be modified to accommodate any number of reporting stations (with different data items).

The format of the information on tape should be as described in Table IV for the version of CREATE3 included in the APICS system. The date and time are given for each observation; the information in columns 11 to 19 is repeated for each station; and the information in columns 1 to 82 is repeated for each 15-minute period of the day. Each record then contains observations for one day consisting of 7872 characters with missing data indicated by all 7's.

---

*Telemetered Air Monitoring Stations

TABLE IV

Tape Format for TAM Data

Tape Position Number	Contents
1-2	Month
3-4	Day
5-6	Year
7-10	Time (e.g.: 0000, 0015, 0030,...)
11-13	SO_2 concentration in ppm (X.XX)
14-16	Wind Direction in Degrees, Clockwise, with North=360 (XXX.)
17-19	Wind velocity in knots (XX.X)

The CREATE3 program calculates hourly averages for each of the three measurements for each of the eight TAM stations and writes this information onto a tape in the master file format. This format is described in Section 2.2.1 with detailed information given in Section 5.1.1. The parameter values are in units as described in Tables II and III in Section 3.0.

The processes involved in averaging the 15-minute readings are described below. (Data validation procedures described in Reference 5 are carried out before the information is transferred to magnetic tape for processing by the APICS system.)

The air quality data averaging process involves the consideration of five readings (in the general case) for a given hour. Specifically, in order to determine the average pollutant concentration, wind direction, and wind speed for hour n, the sequence of readings taken at n-30 minutes, n-15 minutes, n, n+15 minutes, and n+30 should be averaged using the relationship:

$$(\text{Average } SO_2 \text{ at hour } n) = \frac{1/2n_{-30} + n_{-15} + n + n_{+15} + 1/2n_{+30}}{4}$$

The method of averaging yields a value which is "centered" on hour n. If one or more of the data points is missing for any reason, the divisor used to develop an hourly average is, of course, a number less than 4. For example, if n_{-15}, n, <u>or</u> n_{+15} is missing the appropriate divisor is 3. If n_{-30} <u>or</u> n_{+30} is missing, the divisor is 3.5, etc. An alternative averaging procedure, such as averaging of the four readings preceding hour n, can be employed by modifying the appropriate statement in the CREATE3 listing.

As indicated above, the average pollutant reading is derived by a simple arithmetic averaging process for (at most) five consecutive readings. Wind direction and wind speed averages, however, are calculated as follows:

Given the wind direction readings θ_i, i=1,2,3,4,5 and the wind speed readings S_i, i=1,2,3,4,5

$$x_i = S_i \cos \theta_i \quad , \qquad i = 1,2,3,4,5$$

$$y_i = S_i \cos \theta_i \quad , \qquad i = 1,2,3,4,5$$

$$X = x_1 + x_2 + x_3 + x_4 + x_5$$

$$Y = y_1 + y_2 + y_3 + y_4 + y_5$$

Having thus found the legs of a resultant right triangle, the resultant speed and direction are determined

$$\theta = \tan^{-1} \frac{Y}{X}$$

and

$$S = (Y^2 + X^2)^{1/2}/5$$

Note that it is important to keep track of the quadrant location of θ.

The date limits are built into the program and must be modified whenever the program is executed. A title card is read by the program. DD cards must be supplied for the SOURCE and NEWFILE data sets as described above in Section 3.0. Sample deck structures are given in APPENDIX B.

A listing of the source program may be found in Appendix C. The printed output from this program includes a complete data printout with appropriate headings for each table of hourly values as they were written onto the master tape.

3.4 Meteorological Data (CREATE1 and LITTLEC1)

The APICS system includes programs to translate hourly meteorological data provided by ESSA (U.S. Dept. of Commerce), into the master file format. Tapes of Airways Surface Observations may be obtained from the National Weather Records Center (NWRC) at Asheville, North Carolina. "Tape deck 1440 generated from card deck 144" should be specified. The record format for the data is described in detail in the Tape Reference Manual TDF 14 (Reference 10).

Each airport recording station is associated with a station number* (WBAN). This number is included in the information on the weather data tape in the format summarized in Table V. Note that six (6) complete hourly observations are contained in each record. The information in columns 1 through 15 appears at the beginning of each record and positions 16 through 95 are then repeated six times. Ten (10) years of information for a single station may be contained in one BCD high density 7-track tape. Each record is 495 characters in length and the information for each day starts with midnight, local standard time.

The CREATE1 program processes a tape containing data for one airport weather station. The data on the tape is converted as appropriate and written on a master tape with one record for each day's information in the master file format. Eight-character (or less) station and parameter names are associated with each data item as described in Figure 3 (Section 2.2.1). A detailed description of the format is given in Section 5.1.1. The data items on the tape include most of the information from the weather tape and are described in Tables II and III in Section 3.0. All data are converted to decimal numbers and missing values are written as 10^{20} (1.0E+20).

---

*Stations involved in the Chicago study are Midway (14819), Meigs (94866), Glenview (14880), and O'Hare (94846).

Table V

TAPE FORMAT FOR AIRWAYS SURFACE OBSERVATIONS

Tape Position No.	Contents
1-4	Card Deck (1440)
5-9	Station Number (WBAN)
10-11	Year
12-13	Month
14-15	Day
16-17	Hour (00, 06, 12, 18)
18	Ceiling Indicator
19-21	Ceiling Height
22	Visibility Indicator
23-25	Visibility Code
26-27	Wind Direction
28-30	Wind Speed (Signed Plus Over Position 30)
31-33	Dry Bulb Temperature (Signed Plus or Minus Over 33)
34-36	Wet Bulb Temperature (Signed Plus or Minus Over 36)
37-39	Dew Point Temperature (Signed Plus or Minus Over 39)
40	Relative Humidity Indicator
41-43	Relative Humidity
44-48	Sea Level Pressure
49-52	Station Pressure
53	Sky Condition Indicator
54-57	Sky Condition Code
58	Total Cloud Amount
59	Opaque Amount
60	Amount of First Cloud Layer
61	Type of First Cloud Layer
62-64	Height of First Cloud Layer
65	Amount of Second Cloud Layer
66	Type of Second Cloud Layer
67-69	Height of Second Cloud Layer
70	Summation through the Second Cloud Layer
71	Amount of Third Cloud Layer
72	Type of Third Cloud Layer
73-75	Height of Third Cloud Layer
76	Summation through the Third Cloud Layer
77	Amount of Fourth Cloud Layer
78	Type of Fourth Cloud Layer
79-81	Height of Fourth Cloud Layer
82-89	Present Weather Codes
90-94	Blank Positions*
95	Record Mark Referred to as "Zero Zone Zero"

*Since January 1964, cols 90-91 contain wind directions in 10's of degrees.

An additional preprocessing program LITTLEC1 is provided to first extract only the data of interest from an NWRC weather tape. Such tapes frequently contain data for several airports. In such cases, LITTLEC1 may be used to extract the information for just one station for further processing by CREATE1. LITTLEC1 may also be used to extract information for only a portion of the dates for which information is on the tape from the NWRC. The format of the resultant tape from this program is the same as the input weather tape, so either may be used as the input tape to CREATE1.

Date limits (and station numbers) are built into the programs and must be modified as appropriate whenever the program is executed. Input tapes can be concatenated and DD cards must be supplied for SOURCE and NEWFILE datasets as described above in Section 3.0. The only card input for the CREATE1 program consists of a station identification card with the airport name given in the first eight columns. See Appendix B for a sample deck structure.

A listing of the source programs may be found in Appendix C. Although there is no printed output from LITTLEC1, there is complete output from CREATE1. However, under MVT the headings and the information in the tables appear in two different datasets, so the user may find it desirable to change the code to suppress part of the printout.

3.5 Functions of File Parameters (TURSTAB)

This program calculates an atmospheric stability index using surface weather data already on a master tape. It also provides an example of using the FUNCTION program described in Section 4.3 to create a master tape of calculated or derived parameters based on raw data or existing parameter values. The new master tape containing atmospheric stability values will contain this information for the same dates as the input master.

The pertinent parameters for calculating atmospheric stability are the cloud cover, cloud ceiling, and wind speed. Date information is also needed, but this is obtained automatically from the beginning of each record on a master file.

(See the description of the Master File Format in Section 5.1.1). The time of day, which is also required, is provided by the input records which contain 24 hours' worth of meteorological information starting at midnight for each day.

The atmospheric stability class is determined as described by D. Bruce Turner in Reference (9). Sunrise and sunset information is incorporated into the stability index for times within 2 hours of sunrise and sunset. The resulting parameter values may then be interpreted as described in Tables II and III (n-p) in Section 3.0.

It is frequently of interest to the air pollution analyst to correlate observed air quality or meteorology data with proximity to sunrise or sunset. In consequence, TURSTAB is equipped to flag the atmospheric stability index with indicators of this proximity. TURSTAB will output either of two alternative expressions for atmospheric stability. These are:

TURNER_A To the basic stability index is added 100 for sunrise hours, 200 for sunset hours, and (10 - number of hours to sunrise or sunset)

TURNER_B The basic stability index is multiplied by 100. 10 is added for sunset hours plus (3 - number of hours to sunrise or sunset)

For a listing of the code used to calculate these parameters see Appendix C6. A station name of STAB is normally assigned in the input for these parameters. However, with a minor modification to the program (see the listing in Appendix C6), the unflagged stability class may be obtained as TURNER_C. A station name of TURNER is normally specified for this parameter.

The format of the card input is as described for the FUNCTION program in Section 4.3, and sample inputs for TURNER_A and TURNER_B are also provided in that section. However, the following card input would be used with the modified code in order to output TURNER_C.

```
SAMPLE TURSTAB
IN_STATIONS=1;
STATION='MIDWAY'     PARAMS=3;
PARAM='CLD_SUM4';
PARAM='CEIL';
PARAM='WIND_VEL';
FUNCTION='TURNER'    PARAMS=1;
PARAM='TURNER_C';
END
STOP
```

When executing the program, DD cards must be provided for the CARDS, MASTER, and SUBFILE data sets as indicated in Section 4.3. See Appendix B for sample deck structures.

3.6 Specialized Data Input Program (MASTER2)

Given an arbitrary station name and parameter values on data cards, the MASTER2 program generates a complete master tape. Most of the information required to generate a master tape comes from the input cards, but part of the master file header, including date information, is built into the MASTER2 program.

The program has been used to create specialized master tapes of mixing-heights and stability types calculated by other programs not included in the general APICS system. For example, urban mixing height data for Chicago was developed by interpolation in time and space between two nearly rural radiosonde balloon stations, followed by a translation of the resultant "constructed rural" Chicago lapse rate profile to the urban core area on the basis of observed urban surface temperatures (this procedure is described in detail in refs. 4 and 8). Hourly values for the derived urban mixing layer height recorded on punched cards [illegible] the APICS data file through the use of MASTER2. The

program may be used to incorporate any station and parameter information into the master file format as follows:

a) Appropriate date-limits and parameter names must be set up in the declare statements of the PL1 source code before compiling the MASTER2 program. As presently coded -

The STARTING and ENDING dates are given as '670801' and '671231' on the 15th and 16th cards. These may be altered by the user, as appropriate.

Input parameter names of 'MIX_HT' and 'KEY' are assigned on the 23rd and 24th cards. These names may, of course, be changed at the discretion of the user.

b) If the user wishes to input values for the second parameter, these must be inserted for KEY before compiling the MASTER2 program. Otherwise this second parameter is set to 0.0.

c) The card input deck must contain the following information:

A station identification card with the station name starting in column 1, up to a maximum of 8 columns.

The next card must be blank.

Hourly parameter values for the (first) parameter are then given 12 per card with two cards for each day as follows:

cols. 1-6: the date with columns 1 and 2 for the day of the month, 3 and 4 for the month, 5 and 6 for the year (e.g.: 311267 for 12/31/67).

col. 7: blank

col. 8: a "1" for the 1st card for the day and a "2" for the 2nd card for the day (not used)

(continued)

cols. 9-80: 12 parameter values with 6 columns for each. In each 6 columns the first 4 contain an integer and the last 2 may be blank or contain an exponent (power of 10) associated with the number.

Missing values must be indicated as 9999 with a blank exponent. These are converted to 10^{20} for writing on the master tape.

3.7 Point Source Emission Data (PLANTSIM)

Point source emission data is treated somewhat differently than air quality and meteorological data for the following two reasons:

1) The number of significant sources is large compared to the number of air quality and meteorological recording stations;
2) Hourly data for each individual source is difficult, if not impossible, to obtain.

The approach taken here has been to create an emission simulation program for the industrial/commercial and utility plant aggregates. In this program utility plants are modeled in more detail than industrial or commercial plants because (1) they are large emitters, (2) they are relatively few in number, and (3) accurate hourly power generation data is usually available.

3.7.1 Industrial Source Simulation

The industrial source data file is designed to include enough information about each plant to characterize its "expected" emission pattern by shift for each day throughout the year.

For present purposes, we define "shifts" as follows:

Shift 1: 12 midnight to 8 a.m. (0000 to 0800)

Shift 2: 8 a.m. to 4 p.m. (0800 to 1600)

Shift 3: 4 p.m. to 12 midnight (1600 to 2400)

Up to three daily operational emission patterns may be assigned to each plant. These are as follows: (1) W - weekday or "normal operation; (2) S - [illegible]y or special holidays.

Finally, a monthly weighting is provided to allow for variations due to seasonal patterns, etc.

Each plant requires three computer cards; (1) a source identification card, (2) a source description card, and (3) a source emission card. Formats and an example of each are shown in Figure 3.1. A description of each item on the cards is given in Table VI.

Computations

When the temperature, T, is between -10°F and 55°F, a linear relationship for the space heating thermal load L^S is assumed, as follows:

$$L^S = L^S_{MAX} \frac{[55 - T]}{65}, (-10<T<55) \tag{1}$$

Then the total thermal load is

$$L = L^S + L^P, \tag{2}$$

where L^P, the process load, is computed within the PLANTSIM routine from the appropriate month, day, and shift factors. The amount of load due to coal, L^C, and the amount of load due to oil, L^O, is then determined, and the SO_2 emission due to each fuel is computed as follows:

$$C(\text{Tons/Hour}) = \frac{L^C(\text{Th/Hour}) \times 10^5(\text{Btu/Th})}{12000(\text{Btu/Lb}) \times 2000(\text{Lbs/Ton})} \tag{3}$$

$$SO_2^C(\text{Lb/Hour}) = C(\text{Tons/Hour}) \times 36.8 \times \%S^C \tag{4}$$

$$O(\text{Kgal/Hour}) = \frac{L^O(\text{Th}) \times 10^5(\text{Btu/Th})}{18000(\text{Btu/Lb}) \times 8000(\text{Lbs/Kgal})} \tag{5}$$

$$SO_2^O(\text{Lb/Hour}) = O(\text{Kgal/Hour}) \times 157.9 \times \%S^O \tag{6}$$

Thus, the total SO_2 emission is

$$SO_2 = SO_2^C + SO_2^O \tag{7}$$

Obviously, a similar procedure could be employed to [illegible] sions on the basis of fuel use, control device eff[illegible]

INDUSTRIAL SOURCE IDENTIFICATION CARD

SOURCE NO.	SOURCE I.D. NAME	NAME AND ADDRESS
1 3	5 12	14 80
2	CAMPBELL	CAMPBELL SOUP 2550 WEST 35 STREET

INDUSTRIAL SOURCE DESCRIPTION CARD

SOURCE NO.	TYPE	GRID	COORDINATES		STACK DATA								WORK SCHEDULE							
			x	y	1		2		3		4									
					HGT	%	HGT	%	HGT	%	HGT	%	M	T	W	T	F	S	S	
1 3	5 8	10 13	15 19	20 24	26 28	29 30	31 33	34 35	36 38	39 40	41 43	44 45	47	48	49	50	51	52	53	55 80
2	2032	0809	8.2	-3.5	175	25	175	25	250	50			W	W	W	W	W	S	H	

INDUSTRIAL SOURCE EMISSION CARD

SOURCE NO	MAXIMUM SPACE HTG. LOAD (THERM/hr)	MAXIMUM PROCESS LOAD (THERM/hr)	MONTHLY FUEL USE WGT (% OF MAXIMUM PROCESS)												SHIFT FUEL WEIGHTING (% OF MONTHLY PROCESS)												% SUL-PHUR COAL	AVG. % COAL	% SUL-PHUR OIL	AVG % OIL	% GAS SP HTG	% GAS PRO-CESS	AVG TEMP GAS (°F)	
															W			S			H													
			J	F	M	A	M	J	J	A	S	O	N	D																				
1 3	5 9	11 15	17	18	19	20	21	22	23	24	25	26	27	28	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	49 51	53 55	57 59	61 63	65 67	69 71	73 74	76 79	80
2	500	5000	6	6	6	5	5	4	2	4	6	7	6	6	70	99	99	70	20	20	30	30	30				2.8	100			100	100	40	

113-1141

Fig. 3.1 Industrial File Card Formats

Table VI Industrial File Card Descriptions

INDUSTRIAL SOURCE I.D.CARD

NO.	ITEM	DESCRIPTION	CARD COLS.
1	SOURCE	Plant sequence number	1 - 3
2	SOURCE I.D. NAME	A symbolic plant code name (not to exceed 8 characters) which will be used for printout identification	5 - 12
3	NAME AND ADDRESS	Complete plant name and address	14 - 80

INDUSTRIAL SOURCE DESCRIPTION CARD

1	SOURCE	Plant sequence number	1 - 3
2	TYPE	Standard industrial code	5 - 8
3	GRID	Square mile number	9 - 13
4	COORDINATES	x and y coordinates relative to standard city coordinate system L.C.	15 - 25
5	Stack DATA	Four stacks maximum (1, 2, 3, 4) HGT - Stack height in feet % - The percent of total emission from each stack	27 - 45
6	WORK SCHEDULE	Each day of the week is assigned to one of the following patterns; i) W - Weekday ii) S - Saturday iii) H - Sunday or Holiday	47 - 53

Table VI (Cont'd)

INDUSTRIAL SOURCE EMISSION CARD

NO.	ITEM	DESCRIPTION	CARD COLS
1	SOURCE NO.	Plant sequence number	1 - 3
2	MAX SPACE HTG LOAD	Maximum hourly thermal requirements for space heating (therms/hr)	5 - 9
3	MAX PROCESS LOAD	Maximum hourly thermal requirements for process (therms/hr)	11 - 15
4	MONTHLY FUEL USE WEIGHTING	Achieves monthly variation in process fuel use. Each month is assigned a weight (0, 1, ...,9) which corresponds to the average percent (0%, 10%,...,90%) of maximum process fuel that is used during that month	17 - 28
5	SHIFT FUEL USE WEIGHTING	Within each daily pattern (W,S,H), a weight (0%,,,,,99%) is assigned to each shift which corresponds to the average percent of current monthly process fuel that is used during that shift	30 - 47
6	% SULPHUR COAL	Average percent of sulfur in coal	49 - 51
7	AVG % COAL	Average percent of thermal load due to coal (0 implies coal is not used)	53 - 55
8	% SULPHUR OIL	Average percent of sulfur in oil	57 - 59
9	AVG % OIL	Average percent of thermal load due to oil (0 implies oil is not used)	61 - 63
10	% GAS SP HTG	The percent of space heating thermal requirements that can be supplied by gas	65 - 67
11	% GAS PROCESS	The percent of process thermal requirements that can be supplied by gas	69 - 71
12	AVG GAS TEMP	The average temperature (°F) at which the dual fuel plant receives gas	73 - 74

emission factors. This capability is not included in this edition of the APICS system but it can easily be added by the user, if desired.

When the temperature is such that a dual fuel, "interruptible" plant* is probably receiving gas, the amount of SO_2 produced is correspondingly reduced. For the purpose of storing data, each plant is allotted four stacks with the following parameters for each stack:

1. SO_2 emission in lbs per hour
2. Heat emission in therms per hour.

These parameters are determined by simply weighting the total SO_2 and heat emissions by the percentage emitted from each stack. The heat emission, H, is assumed to be 15% of thermal input. Plants having less than four stacks have zeros filled in for non-existent stacks.

Example

Consider the example of the Campbell Soup Company for the following set of conditions:

1. January
2. temperature 30°F
3. weekday pattern
4. first shift.

Then,

$$L^S = 500\left(\frac{55 - 30}{65}\right) = 200 \text{ Th/Hr}$$

$$L^P = (.70)\ (.60)5000 - 2100 \text{ Th/Hr}$$

$$L = 2300 \text{ Th/Hr}$$

$$L^C = 2300 \text{ Th/Hr}$$

$$L^O = 0$$

$$C = \frac{2300 \times 10^5}{1200 \times 2000} = 10(\text{Ton/Hr})$$

*See Ref. 5 for a detailed discussion of the operating characteristics of a natural gas supply utility and the constraints imposed on various classes of gas consumers.

$$SO_2 = 10 \times 36.8 \times 2.8 = 1000 \text{ Lbs/Hr}$$

$$H = (.15)2300 = 345 \text{ Th/Hr}$$

<u>For Stack 1:</u> $SO_2^{(1)} = (.25)1000 = 250$ Lbs/Hr

$H^{(1)} = (.25)345 = 85$ Th/Hr

<u>For Stack 2:</u> $SO_2^{(2)} = (.25)1000 = 250$ Lbs/Hr

$H^{(2)} = (.25)345 = 85$ Th/Hr

<u>For Stack 3:</u> $SO_2^{(3)} = (.50)1000 = 500$ Lbs/Hr

$H^{(3)} = (.50)345 = 173$ Th/Hr

<u>For Stack 4:</u> $SO_2^{(4)} = 0$

$H^{(4)} = 0$

A sample PLANTSIM input for the Campbell Soup Co. example follows:

```
SAMPLE INPUT FOR PLANT SIMULATION PROGRAM
DATE_LIMITS=2;
'66/01/01','66/01/01','67/01/01','67/01/05'
UTILITIES= 1;
  2 FISK      FISK 1111 W. CERMACK
  2 4911 0407    6.4 -1.9   5 1 2902 2903 2904 2905 450        2 3.0 50
  2 18    170    0 98.922-72.52 1   252   253   254   25      0 3.182
  2 19    353   62 84.855-262.5 5 100                        2.935 3.112
SOURCES=1   ;
  2 CAMPBELL CAMPBELL SOUP COMPANY    2550 WEST 35TH STREET
  2 2032 0809 8.2   -3.5   175251752525050        WWWWWSH
  2 500    5000   666554246766 709999702020303030 2.8 100    0    0 100 100 40 .042
```

3.7.2 Utility Source Simulation

As previously mentioned, the simulation of utility plant emissions is handled separately from other sources within the plant simulation program. Hourly power or fuel consumption data is usually available for each generating unit in the utility system. Thus the utility card file is designed primarily to allow for the input of utility plant physical and operating characteristics. Also, a quantitative indicator of unit to stack coupling is provided, since this enables an estimation of stack thermal emissions - a critical factor in estimating smoke plume rise.

Utility File Card Formats - Each utility plant requires a plant identification card, and a plant description card. Each generating unit within the plant requires a unit emission card. Formats and an example of each are shown in Figure 3.2. A description of each item on the cards is provided in Table VII.

Computations

As with the industrial sources, the hourly SO_2 emission and thermal emission are computed. In this case, hourly unit power load data are provided. This data is assumed to reside in the master file.

The following notation is employed:

j the plant index, $j = 1,2,\ldots,J$;

i_j the unit index for plant j,

$i_j = 1,2,\ldots,I_j$;

k_j the stack index for plant j,

$k_j = 1,2,\ldots,K_j$;

L_{i_j} the hourly load on unit i_j;

$T^I_{i_j}$ and $T^O_{i_j}$ the hourly thermal input to and thermal emission from unit i_j, respectively;

C_{i_j} the hourly coal (tons) burned in unit i_j;

UTILITY SOURCE IDENTIFICATION CARD

UTILITY NO.	UTILITY I.D. NAME	NAME AND ADDRESS
1 3	5 12	14 80
2	FISK	FISK 1111 W. CERMACK

UTILITY SOURCE DESCRIPTION CARD

UTILITY NO.	TYPE	GRID	COORDINATES		NO. STACK	STACK HEIGHTS												NO. UNITS	% SULPHUR COAL	TEMP GAS AVAIL	
			x	y		NO.*	HGT	NO.*	HGT	NO.*	HGT.	NO.*	HGT	NO.*	HGT	NO.*	HGT				
1 3	5 8	10 13	15 19	20 24	26 27	29 30	31 33	34 35	36 38	39 40	41 43	44 45	46 48	49 50	51 53	54 55	56 58	60 61	63 65	67 70	72 80
2	4911	0407	6.4	-1.9	5	1	290	2	290	3	290	4	290	5	450			2	3.0	50	

*STACK AND UNIT IDENTIFIERS' NUMBERS MUST BE CONSISTENTLY LEFT OR RIGHT ADJUSTED

UNIT EMISSION CARD

UTILITY NO.	UNIT NO.	MAXIMUM LOAD MW	% GAS CONVERT	COEFFICIENTS OF THERMAL INPUT VS LOAD (LINEAR)		UNIT TO STACK COUPLING								PRICE BOILER GAS (¢/THERM)	PRICE COAL (¢/THERM)	PRICE POLLUTION GAS (¢/THERM)	PRICE LOAD SHIFT (¢/THERM)	MAX POLLUTION GAS (THERMS)
						STACK NO.	WGT	STACK NO.	WGT	STACK NO.	WGT	STACK NO.	WGT					
1 3	5 6	8 12	14 16	18 23	24 29	31 32	33 35	36 37	38 40	41 42	43 45	46 47	48 50	52 56	58 62	64 68	70 74	76 80
2	18	170	0	98.92	- 72.5	1	25	2	25	3	25	4	25					
2	19	353	62	84.85	-262.	5	100											

113-1142

Fig. 3.2 Utility File Card Formats

Table VII Utility File Card Descriptions

UTILITY SOURCE I.D. CARD

NO.	ITEM	DESCRIPTION	CARD COLS.
1	UTILITY NO.	Plant sequence number	1 - 3
2	UTILITY I.D. NAME	A symbolic plant code name (not to exceed 8 characters) which will be used for print out identification	5 - 12
3	NAME and ADDRESS	Complete plant name and address	14 - 80

UTILITY SOURCE DESCRIPTION CARD

1	UTILITY NO.	Plant sequence number	1 - 3
2	TYPE	Standard industrial code	5 - 8
3	GRID	City of Chicago square mile number	9 - 13
4	COORDINATES	x and y coordinates relative to standard city coordinate system.	15 - 25
5	NUMBER OF STACKS	Number of plant stacks	26 - 27
6	STACK DATA	Six stacks maximum NO - Stack identification number HGT Stack height in feet	29 - 58
7	NUMBER OF UNITS	Number of plant generating units	60 - 61
8	% SULPHUR COAL	Average percent of sulphur in coal	62 - 64
9	TEMP GAS AVAIL	The average ambient temperature (°F) at which the plant usually receives "interruptible" boiler gas.	66 - 68

Table VII (Cont)

UNIT EMISSION CARD

NO.	ITEM	DESCRIPTION	CARD COLS
1	UTILITY NO.	Plant sequence number	1 - 3
2	UNIT NO.	Unit identification number	5 - 6
3	MAXIMUM LOAD	The maximum unit load (megawatts)	8 - 12
4	% GAS CONVERT	The percent of maximum load which can be supplied by gas.	14 - 16
5	COEFFICIENTS OF THERMAL INPUT VS. LOAD	The production function (assumed linear) for each unit. i.e. $T = A L + B$ where T is thermal input L is unit loading A, B are coefficients	18 - 29
6	UNIT STACK COUPLING	Unit to stack coupling description STACK NUMBER - must agree with same stack identification on utility description card. WEIGHT - The average percent of this unit emission which goes up stack identified by this stack number	31 - 50

A_{i_j} and B_{i_j} the coefficients of the (assumed) linear production function of unit i_j (Figure 3.3 shows that the error associated with a linear assumption is not great).

The plant subscript j is dropped for simplicity. Thus,

$$T_i^I = A_i L_i + B_i \text{ and} \tag{1}$$

$$T_i^O = .15\ T_i^I \tag{2}$$

The total unit SO_2 emission is then computed as follows:

$$C_i \text{ (Tons/Hour)} = \frac{T_i^I \text{ (Th/Hour)} \times 10^5 \text{(Btu/Th)}}{12000 \text{(Btu/Lb)} \times 2000 \text{ (Lb/Ton)}} \tag{3}$$

$$SO_{2_i} \text{ (Lb/Hour)} = C_i \times 36.8 \times \%S_i \tag{4}$$

where % S is the percent of sulphur in the coal.

When the ambient temperature is such that dual fuel generator units are receiving gas, the amount of SO_2 is correspondingly reduced. These emissions are now distributed among the stacks:

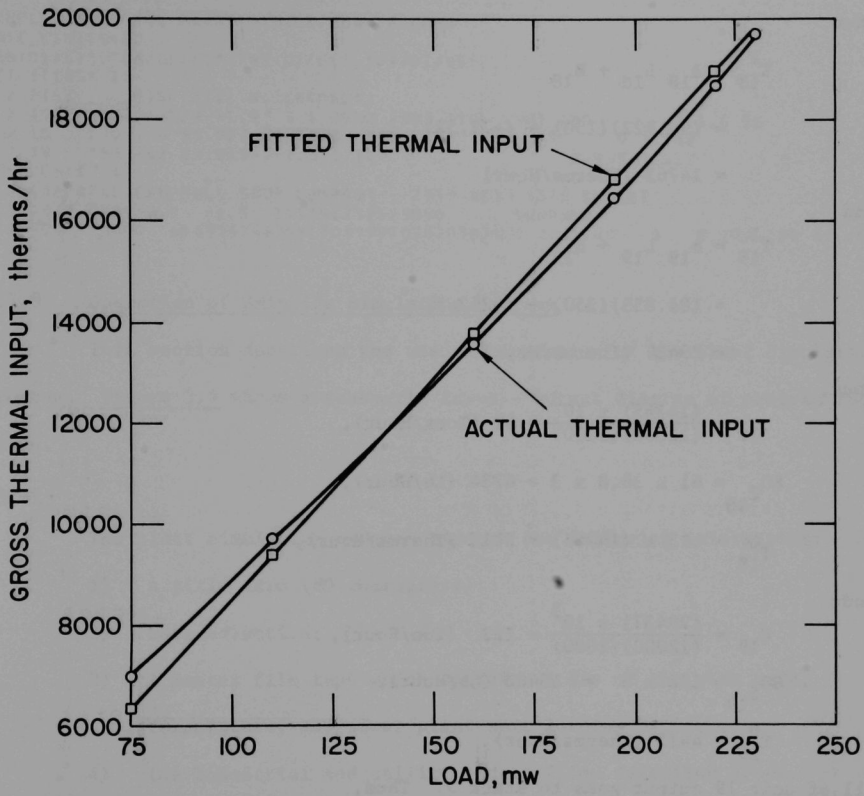
$$SO_{2_k} = \sum_{i=1}^{I} W_{ki}\ SO_{2_i} \tag{5}$$

and $$T_k^O = \sum_{i=1}^{I} W_{ki}\ T_i^O \tag{6}$$

where W_{ki} are the unit to stack coupling coefficients (weights). Note that

$$\sum_{k=1}^{K} W_{ki} = 1 \tag{7}$$

for all i.



112-9796

Fig. 3.3 Typical Gross Thermal Input vs. Load Curve and Least Squares Approximating Line

Example

Consider the example of the FISK power plant. Assume the loads on unit 18 and 19 are 150 and 350 Mw, respectively, and 3% sulfur coal is being used as fuel.

Then

$$T^I_{18} = A_{18}\ L_{18} + B_{18}$$

$$= (98.922)(150) + (-72.52)$$

$$= 14765 \text{ (Therms/Hour)}$$

and

$$T^I_{19} = A_{19}\ L_{19} + B_{19}$$

$$= (84.855)(350) + (-262.5)$$

$$= 29437 \text{ (Therms/Hour)}$$

Thus,

$$C_{18} = \frac{(14765)\ \times\ 10^5}{(12000)(2000)} = 61 \text{ (Tons/Hour)},$$

$$SO_{2_{18}} = 61 \times 36.8 \times 3 = 6734 \text{ (Lb/Hour)},$$

$$T^O_{18} = .15 \times (14765) = 2215 \text{ (Therms/Hour)},$$

and

$$C_{19} = \frac{(29437)\ \times\ 10^5}{(12000)(2000)} = 122 \text{ (Ton/Hour)},$$

$$SO_{2_{19}} = 122 \times 3 = 13468 \text{ (Lb/Hour)},$$

$$T^O_{19} = 4415 \text{ (Therms/Hour)}.$$

All of unit 19 output goes to stack 5. Thus,

$$SO_{2_5} = SO_{2_{19}} = 13468$$

$$T^O_5 = T^O_{19} = 4415$$

Unit 18 output is equally distributed between stacks 1 through 4. Thus,

$$SO_{2_1} = (.25)(6734) = 1683$$

$$= SO_{2_2} = SO_{2_3} = SO_{2_4}$$

$$T_1^0 = (.25)(2215) = 554$$

$$= T_2^0 = T_3^0 = T_4^0$$

A sample input for the FISK power plant is as follows:

```
SAMPLE INPUT FOR PLANT SIMULATION PROGRAM
DATE_LIMITS=2;
'66/01/01','66/01/01','67/01/01','67/01/05'
UTILITIES= 1;
  2 FISK      FISK 1111 W. CERMACK
  2 4911 0407    6.4 -1.9   5 1 2902 2903 2904  2905 450        2 3.0 50
  2 18   170    0 98.922-72.52 1   252   253   254  25       0 3.182
  2 19   353   62 84.855-262.5 5 100                    2.935 3.112
SOURCES=1   ;
  2 CAMPBELL CAMPBELL SOUP COMPANY    2550 WEST 35TH STREET
  2 2032 0809 8.2   -3.5   175251752525050        WWWWWSH
  2 500    5000   666554246766 709999702020303030 2.8 100    0    0 100 100 40 .042
```

3.7.3 Operation of Emission Simulation Program

This section describes the use and operation of the plant simulation program. Figure 3.4 shows a schematic input - output diagram of program requirements.

Input

The plant simulation program expects the following information:

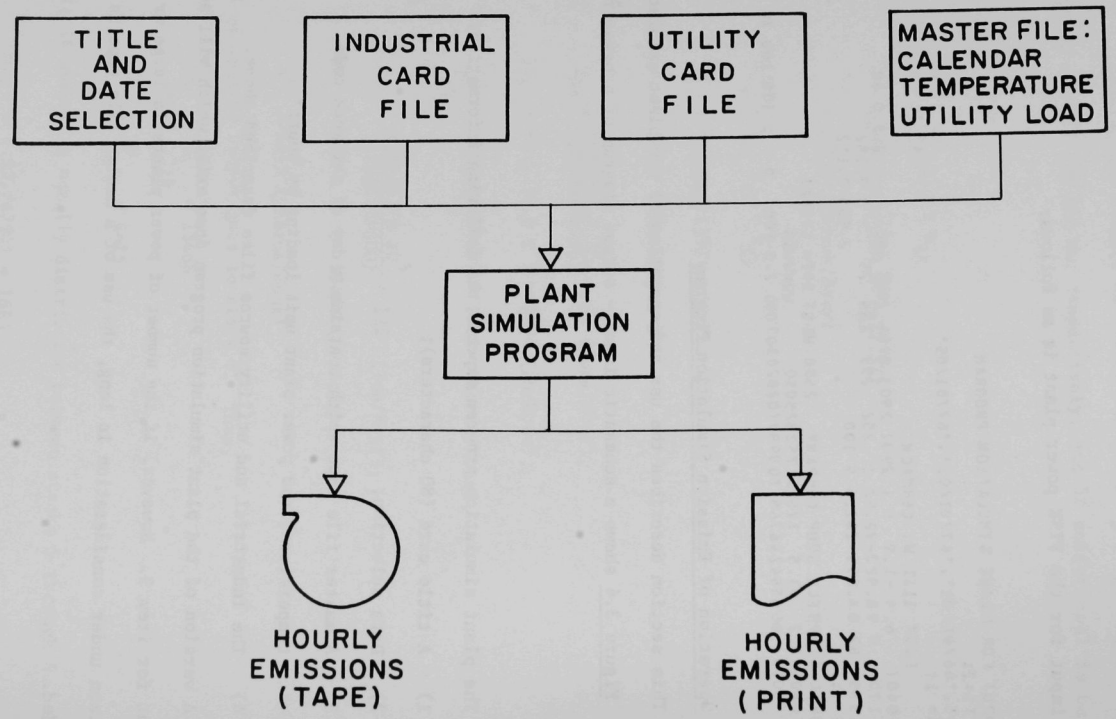
1) A title card (80 characters);

2) Data selection;

3) A master file tape which contains a day of the week code, temperature, and power plant unit loading by hour;

4) The industrial and utility source files described above.

A version of the plant simulation program does exist which will accept card input for item 3. However, if the number of power plants is large or the time horizon under consideration is long, the use of a master file tape is recommended.



113-1143

Fig. 3.4 Input-Output Diagram of Plant Simulation Program

The day of the week code is a number from 1 to 7 corresponding to Monday through Sunday. If a holiday occurs, the day of the week code is preceded by a 1. Thus, if Wednesday is a holiday the code would be 13. To access the day of the week code, the plant simulation program looks for a station with the name 'CALENDAR' and a parameter name "DAY_WEEK'.

The hourly temperature is currently accessed by the station name 'MIDWAY' (corresponding to that airport meteorological station) and the parameter name 'TEMP'. These identifiers can easily be changed by referring to the listing in the appendix.

To access the utility plant data, the program combines the five characters 'LOAD___' with the two character unit identification number to create the unit load parameter name. Thus, referring to the Fisk plant example introduced in the previous section, the program will look for the station name 'FISK' and the parameter names 'LOAD__18' (generator unit 18) and 'LOAD__19' (generator unit 19) on the master file.

The appendix section shows an example of the input and output DD card formats (B) as well as program listings (C8). The program expects to operate for a minimum of one twenty-four hour period. The data selection cards provide the facility of selecting any desired date combinations which are within the master file date limits. The only restriction is that dates must be selected in increasing order from oldest to most recent. In the example, we have chosen to compute plant emissions for January 1, 1966 only and January 1 through January 5, 1967, inclusive.

Output

The output of the plant simulation program is an hourly history of both industrial-commercial and utility plant SO_2 and thermal emissions. The example below shows the output for the FISK utility plant and the Campbell Soup Company. This output is also recorded on magnetic tape in a format that is accepted by the atmospheric dispersion model which is incorporated in the APICS system. The dispersion model program and its operation are discussed in Section

PLANTSIM: Sample Output

SAMPLE INPUT FOR PLANT SIMULATION PROGRAM
DATE:67/01/05

HOUR	FISK _1 Q_SO2	FISK _1 Q_HEAT	FISK _2 Q_SO2	FISK _2 Q_HEAT	FISK _3 Q_SO2
1	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0
6	37.164	12.119	37.164	12.119	37.164
7	924.493	301.465	924.493	301.465	924.493
8	1527.422	498.073	1527.422	498.073	1527.422
9	1698.063	553.716	1698.063	553.716	1698.063
10	1811.823	590.812	1811.823	590.812	1811.823
11	1868.703	609.360	1868.703	609.360	1868.703
12	1641.183	535.168	1641.183	535.168	1641.183
13	1641.183	535.168	1641.183	535.168	1641.183
14	1811.823	590.812	1811.823	590.812	1811.823
15	1811.823	590.812	1811.823	590.812	1811.823
16	1311.278	427.591	1311.278	427.591	1311.278
17	1607.055	524.040	1607.055	524.040	1607.055
18	1868.703	609.360	1868.703	609.360	1868.703
19	1675.311	546.297	1675.311	546.297	1675.311
20	1561.550	509.201	1561.550	509.201	1561.550
21	1868.703	609.360	1868.703	609.360	1868.703
22	1629.807	531.459	1629.807	531.459	1629.807
23	1072.381	349.690	1072.381	349.690	1072.381
24	0.0	0.0	0.0	0.0	0.0

HOUR	CAMPBELL Q_SO2	CAMPBELL Q_HEAT
1	984.163	343.846
2	984.163	343.846
3	984.163	343.846
4	984.163	343.846
5	990.768	346.154
6	990.768	346.154
7	994.071	347.307
8	984.163	343.846
9	1354.379	473.192
10	1354.379	473.192
11	1354.379	473.192
12	1354.379	473.192
13	1354.379	473.192
14	1347.774	470.884
15	1351.077	472.038
16	1351.077	472.038
17	1357.682	474.346
18	1364.287	476.654
19	1364.287	476.654
20	1367.590	477.807
21	1370.892	478.961
22	1370.892	478.961
23	1367.590	477.807
24	1367.590	477.807

APICS USER MANUAL

4.0 Master File Maintenance

4.0 Master File Maintenance

An appropriate philosophy is to store at least two years' worth of information on a master tape. It is, of course, desirable to start a master tape the first year and to update it as additional information becomes available. Magnetic tapes containing information in the master file format are first created as described in Section 3 above. Then the information on these "masters" may be combined, segmented, and used as a basis for function calculations as desired to obtain new masters by using the various file manipulation programs described below.

There are three MERGE programs to combine the information from two tapes onto a third tape to make a new master; a SUBFIL program extracts a subset of the information from a master file to generate a new master; and a FUNCTION program calls a user-coded subroutine to evaluate a function of parameters from an old master tape and then puts these on a new master tape. See Appendix A for schematic flow charts indicating when it is appropriate to use each in generating the master files. For sample deck structures see Appendix B.

A large number of tapes are generated by the APICS system necessitating a method to index information pertaining to the contents of the tapes. It becomes desirable to maintain a card file of identifying information such as data set names, data types, dates, source tape numbers, and generating programs. A suggested format for these cards is given in Appendix A4. It is further recommended that raw data be kept on unlabelled 7-track tapes and that labelled 9-track tapes be used for processed data such as master tapes and output arrays from search runs.

4.1 Merging Programs

Each of the three MERGE programs has a unique purpose. MERGE1 and MERGE2 are fast special-purpose routines, and MERGE3 is a slightly slower, more general program. In most cases, the order of the two input tapes is important, and there is no card input to these programs.

When running merge programs, DD cards must be supplied for the two input tapes (OLDFIL1 and OLDFIL2) and for the output tape (NEWFILE). The usual DCB specifications for NEWFILE are RECFM=V and BLKSIZE=20000.

Thus, to modify the DD cards in the running deck for another merge run, it is only necessary to change dsnames to correspond to those on the appropriate magnetic tapes. (The volume serial number for the tapes are specified on ASP Setup cards.)

The printed output includes listings of the dates, stations, and parameter names on all tapes. Each date is printed as the record for the day is written onto the new master. Figure 4.1 illustrates the general merge operation required to update a master file.

4.1.1 MERGE1

This program merges the information for the same hours of observation on two tapes in the master file format to give a new master tape. This program is very fast but there are some restrictions.

(1) The two masters must have the same starting dates,

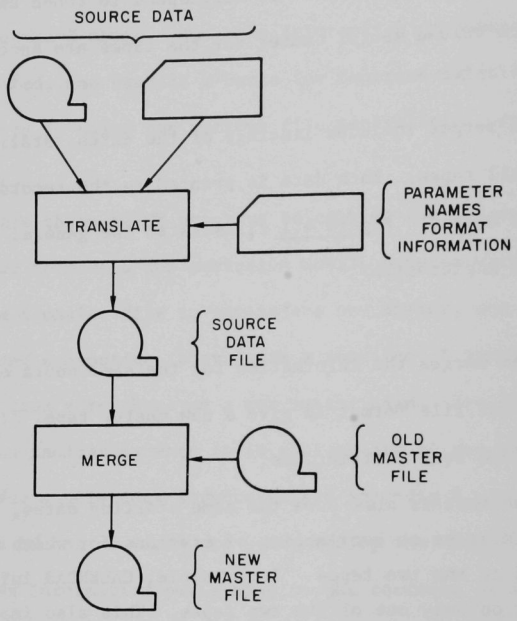
(2) There must be no overlapping of stations for which data is contained on the two tapes. For example, CALENDAR information* should appear on only one of the two tapes. This also implies that values for additional parameters may not be added to those for a station on an existing master tape.

(3) If the dates for which information is given on the first tape extend beyond the second then the output master tape must be padded with "missing" information for the parameters that appeared on the second tape.

(4) The program terminates when it gets to the end of the first tape. Thus, if the dates on the second tape extend beyond those on the first tape, the rest of the data is ignored and the new master goes only as far as the first master.

---

*Remember that the first 6 characters of each record serve as the date identification for matching up and retrieving information from master tapes, while the CALENDAR information is simply treated as another station.



112-9726

Fig. 4.1 Data Translation and Master File Update

4.1.2 MERGE2

This program adds all of the information from one tape to the end of the information on another to generate a new master tape. The usefulness of this program is more limited than is the case for the other merge routines, but it is fast and should be used when appropriate. The second tape must exactly match the first tape in two respects.

(1) The starting date for informaiton on the second tape must match the day after the last data on the first tape.

(2) The same stations and parameters must be given on each tape.

4.1.3 MERGE3

This is a completely general program for merging the information on any two master tapes to create a new master file containing all of the relevant information. There is no restriction on the dates, stations, or parameters contained on the input tapes. However, the generality of this program makes it rather slow, so MERGE1 and MERGE2 should be used wherever possible.

In general MERGE3 is most efficient when the (larger) tape with the most parameters is designated OLDFIL1.

When both tapes contain some of the same data, values are taken from the second tape (OLDFIL2). However, if the value is recorded as missing on one tape (i.e., 1.0E+20), the value from the other tape is used.

If information on one tape is given for dates extending beyond those on the other input tape, missing values are supplied by the program wherever necessary to fill out a complete set of data on the resulting master tape.

In addition to the printed output summary as obtained from MERGE1 and MERGE2, the final parameter values for all three tapes are printed for MERGE3. This printout is also obtained for the first and/or last matching date for tapes with different starting and/or ending dates. Thus it is possible to check on how this program is working and to check on which values are still missing when updating master tapes as information becomes available from a wide variety of sources.

4.2 The SUBFIL Program

This program is used to generate a new master tape containing a subset of the information contained on an input master tape. It is obvious that as the volume of data in the master file grows, the processing time to access portions of this data increases rapidly. Since the usual procedure will be to concentrate on a subset of the master file for a sequence of processing runs or analyses, a program has been written to extract the desired subset and place it on a separate "subfile." (See Figure 4.2.)

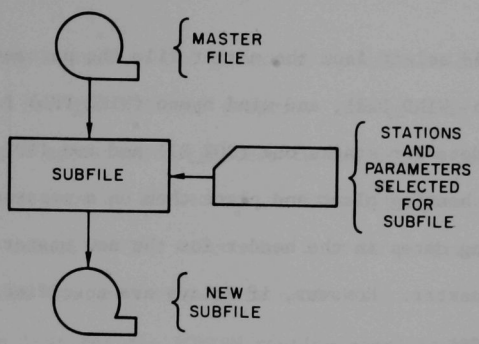
The subfile will contain information for the same dates as the input file, but only the stations and parameters of interest will appear. Information in this new master tape may then be retrieved more economically.

The input cards for this program consist of a title card to identify the printed output followed by out-stations and parameter specifications in a format similar to that for search runs. (An "out-station" is any station for which data is available in a master file and from which data is to be retrieved in order to form a subfile.) The parameter names may be omitted if all are desired for a station, and time delays may be specified for appropriate parameters.

The input format to the SUBFIL program is as follows:

```
SUBFIL INPUT DESCRIPTION (Note: Information enclosed in parentheses is explana-
                                tory only and should not be punched)

TITLE       (80 characters - descriptive information)
OUT_STATIONS=NO; (NO=Number of OUT stations)
(OUT station definition cards - one set per station)
STATION='NAME    ' PARAMS=NP; (NP=Number of output parameters or PARAMS=may be
                               omitted if all parameters for this station are
                               desired)
(Output parameter definition cards - one card per parameter)
PARAM='NAME    ';
END     (This card must follow each case)
STOP    (This card must follow the run)
```



112-9726

Fig. 4.2 Subfile Generation

A sample input deck might be -

```
SAMPLE SUBFILE
OUT_STATIONS=2;
STATION='MIDWAY'     PARAMS=3;
PARAM='TEMP';
PARAM='WIND_DIR';
PARAM-'WIND_VEL';
STATION='U_OF_C'     PARAMS=2;
PARAM='SO2_S1';
PARAM='SO2_S2';
END
STOP
```

This example would select from the master file the parameters temperature (TEMP), wind direction (WIND_DIR), and wind speed (WIND_VEL) for Midway Airport and the SO_2 emission data for stacks one (SO2_S1) and two (SO2_S2) of the University of Chicago heating plant and place them on a separate master file. The starting and ending dates in the header for the new master would be the same as those for the old master. However, if delays are specified, data for the last day will not appear.

When executing the SUBFIL program DD cards must be supplied for a temporary data/set (CARDS), an input tape (MASTER), and an output tape (SUBFILE). (See Appendix B.) The usual DCB specifications for SUBFILE are, RECFM=V and BLKSIZE=20000. The correct dsname for MASTER must be specified, and an appropriate dsname specified for SUBFILE for each run. The volume serial numbers of the tapes are specified on ASP Setup cards.

The printed output includes listings of the dates, station, and parameter names on both tapes. Each date is printed as the record for the day is written onto the new master. (See Appendix D.)

4.3 The FUNCTION Program

This PL1 program is used to create new master tapes containing information based on or derived from parameter values which already exist on master tapes. For example, the atmospheric stability index could be derived from raw meteoro-

logical data and stored on a new master tape. The SUBFIL program is used as a basis for this program. The FUNCTION program requires a user-coded subroutine to calculate the desired functions of the parameters. The calculated values passed back to the FUNCTION program are then output on a new master tape. The new master tape will have information for the same time period as the input master, but only the parameters for the specified "function" station will be included.

Input and output station and parameter specifications are given in the search type input for the FUNCTION program. The program reads the master tape and passes the specified parameters to the user-coded FORTRN subroutine. (FORTRN is also the name of a distinct retrieval program discussed in Section 6.0 of this manual.) The subroutine must be programmed to process the information for one day each time it is called.

The argument list for the FORTRN routine consists of seven items as follows:

NYEAR NMONTH NDAY	These specify the year, month, and day for the information in the arrays for this day.
NIN	The number of input parameters from the master tape.
PARIN	The input array dimensioned 24*NIN. The parameters are in the order specified in the search-like input to the program. Hourly parameter values are given for 24 hrs of the day starting with midnight.
NOUT	The number of output parameters specified for the FUNCTION (output) station in the input.

PAROUT The array of calculated value dimensioned 24*NOUT. The parameters should be in the order specified in the input with values given for 24 hrs of the day starting with midnight. A value of 1.0E+20(10^{20}) should be inserted for missing values in order to be compatible with the rest of the programs in the APICS system.

The input format to the FUNCTION program is as follows:

FUNCTION INPUT DESCRIPTION (Note: Information enclosed in parentheses is explanatory only and should not be punched)

```
TITLE          (80 characters - descriptive information)
IN_STATIONS=NI; (NI=number of IN stations)
(IN station definition cards - one set per station)
STATION='NAME    ' PARAMS=NP; (NP=number of input parameters or PARAMS=may
                              be omitted if all parameters for this station
                              are desired)
(Input parameter definition cards - one card per parameter)
PARAM='NAME    '    DELAY=TD;  (TD=time delay in hours up to 24 max, or
                               DELAY=may be omitted if no delay is appropriate
                               for this parameter)
(A function definition card for the output station)
FUNCTION='NAME    '    PARAMS=NP; (NP=number of output parameters associated
                                  with the station name specified after
                                  FUNCTION=)
(Output parameter definition cards - one card per parameter)
PARAM='NAME    ';

END            (This card must follow each case)
STOP           (This card must follow the run)
```

A sample input deck might be

```
SAMPLE FUNCTION
IN_STATION=1;
STATION='MIDWAY'    PARAMS=3;
PARAM='CLD_SUM4';
PARAM='CEIL';
PARAM='WIND_VEL';
FUNCTION='STAB'     PARAMS=2;
PARAM='TURNER_A';
PARAM='TURNER_B';
END
STOP
```

This example would select from the master file the parameters cloud cover (CLD_SUM4), ceiling (CEIL), and wind speed (WIND_VEL) for Midway Airport, and pass them to a user-coded subroutine. A new master file could then be created containing a "station" named STAB with parameters indicating the atmospheric stability (TURNER_A and TURNER_B) as described in Section 3.5.

As in the case of the SUBFIL program, starting and ending dates in the header information for the new master tape would be the same as those for the old master; and if time delays are specified, data for the last day will not appear.

When executing the FUNCTION program, DD cards must be supplied for a temporary data set (CARDS), an input tape (MASTER), and an output tape (SUBFILE). (See Appendix A.) The usual DCB specifications for SUBFILE are, RECFM=V and BLKSIZE=20000. The correct dsname for MASTER must be specified, and an appropriate dsname specified for SUBFILE for each run. The volume serial numbers of the tapes are specified on ASP Setup cards.

The printed output includes listings of the dates, station and parameter names on both tapes. Each date is printed as the record for the day is written onto the new master. (See Appendix D.)

This program may also be used for purposes other than creating new master files [e.g., to calculate and punch a deck of cards containing historical 12-hr SO_2 averages for studying medical effects of air pollution†]. In such cases, the specifications for the SUBFILE data set may be DDNAME=DUMMY, but it is still necessary to include a DCB specification. [The FUNCTION program is written in PL1; and as PL1 is presently implemented under OS/360, DCB specifications must always be included for dummy data sets.]

---

† A reference to this facility is given in Table I, but the program has not been incorporated into the APICS system.

APICS USER MANUAL

5.0 Master File Search Programs

5.0 Master File Search

Searches of master tapes are made to generate output arrays on tape containing the information of interest in the order specified in the search input. The data is then retrieved from tape by retrieval programs and passed to user-coded FORT routines as described in Section 6. There is also an option in one of the search programs to pass the information in the output array directly to a user-coded FORT routine. The general search routine is illustrated in Figure 5.1.

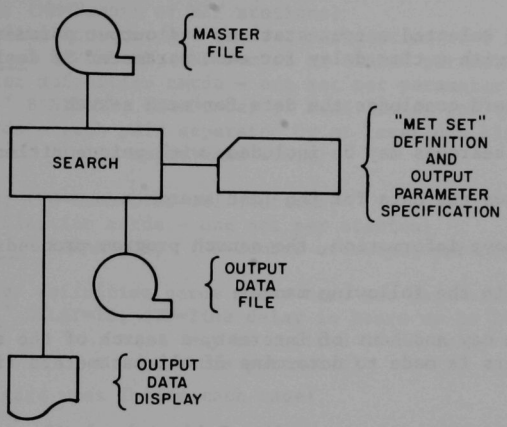
The search program locates the data of interest on the master tape by using the 8-character station and parameter names. The parameters for which values are to appear in the output array are specified as "out-stations." It is possible to specify "band-limits" as restriction on the parameters of interest or on other parameters stored under "met-stations." For example, one or more "date-limits" are normally given, and "hour-limits" (0-23) may also be specified. The programs will then retrieve only data which lies within these specified time bands. These must be specified in ascending order for a particular search. Time delays may be specified for appropriate out-station parameters to facilitate comparing values of some parameters with the values of others recorded a few hours earlier.

A complete printout of the output array under appropriate headings may be obtained directly from the search run. The information in the output array may be retrieved and operated upon by a FORT routine as a second job step for the same computer run in which the search was made. However, it is desirable to always save the output array from a search, so that, in case the second job step did not work properly, it is not necessary to repeat the search.

A search of the master file is defined by input of the following information:

1) A title which serves to identify the output array for this search on the output data file;

2) An indicator for how the output ar



112-9727

Fig. 5.1 Master File Data Search

3) A set of date limits which select days of interest;

4) A set of hour limits which select the hours of interest for each day;

5) A set of met-stations and search parameters for each, if it is desired to limit the search to values in specified ranges of interest. These ranges are defined as a set of "band limits" for each search parameter which delimits the search on that parameter;

6) A set of selected output stations and output parameters for each station with a time delay for each parameter if desired;

7) An END card concludes the data for each search.

Data for several searches may be included, with unique titles used for each.

A STOP card follows the data for the last search.

Given the above information, the search program proceeds to search the master file data in the following manner:

1) For each day and hour of interest, a search of the specified search parameters is made to determine if all parameters lie within some band;

2) If some parameter does not lie within a band, the search proceeds to the next hour;

3) If all parameters lie within some band, the corresponding output parameter values for that day and hour (minus the time delay) are recorded on the output data file;

4) When the search of the master file is complete, the resulting output data may be displayed on the printer.

The input format for the SEARCH program is as follows:

<u>SEARCH INPUT DESCRIPTION</u> (Note: Information enclosed in parentheses is explanatory only and should not be punched)

TITLE (80 characters - Used to identify array on output tape)

either

PRINT='NO or YES'; (Indicates whether the output array should be printed in addition to generating a tape from <u>SEARCH1</u>)

or

OPTLIST='PRINT and/or FILE and/or FORT'; (Each option is separated by a comma for <u>SEARCH2</u>, where these options specify what to do with the output array; i.e., whether to print it and/or to generate an output tape and/or to pass it directly to a FORT routine)

```
DATE_LIMITS=ND;  (ND=Number of date limits or 0 if all are desired)
(Date limit cards - Each pair separated by at least one blank)
'YL/ML/DL','YU/MU/DU' (Lower date limit','upper date limit')

HOUR_LIMITS=NH; (NH=Number of hour limits or 0 if all are desired)
(Hour limit cards - each pair separated by at least one blank)
HL, HU (Lower hour limit, upper hour limit)

MET_STATIONS=NM; (NM=Number of MET stations)
(MET station definitions cards - one set per station or 0 if none are desired)
STATION='NAME____' PARAMS=NP; (NP=Number of SEARCH parameters)
(SEARCH parameter definition cards - one set per parameter)
PARAM='NAME     ' BANDS=NB; (NB=Number of band limits for this parameter)
(Band limit cards - each pair separated by at least one blank)
BL, BU (Lower band limit, upper band limit)

OUT_STATIONS=NO; (NO=Number of OUT stations)
(OUT station definition cards - one set per station)
STATION='NAME____' PARAMS=NP;  (NP=Number of output parameters)

(Output parameter definition cards - one card per parameter)
PARAM='NAME____' DELAY=TD; (TD=Time delay in hours up to 24 max or DELAY=
                              may be omitted if no delay is appropriate for
                              this parameter)
END       (This card must follow each case)

STOP      (This card must follow the run)
```

A sample input deck might be as follows for SEARCH1:

```
SAMPLE INPUT 1
PRINT='NO';
DATE_LIMITS=1;
'66/01/01','66/01/10'
HOUR_LIMITS=1;        6,18
MET_STATIONS=1;
STATION='MIDWAY'    PARAMS=2;
PARAM='WIND_VEL'    BANDS=1;    5,10
PARAM='WIND_DIR'    BANDS=1;  200,250
OUT_STATIONS=2;
STATION='MIDWAY'    PARAMS=3;
PARAM='WIND_DIR';
PARAM='WIND_VEL';
PARAM='TEMP'  DELAY=1;
STATION='U_OF_C'  PARAMS=2;
PARAM='SO2_S1'  DELAY=2;
PARAM='SO2_S2'  DELAY=2;
END
STOP
```

For SEARCH2, the second card must be of the form:

```
OPTIONS='FILE';
```

This example searches the master file between the dates 66/01/01 and 66/01/10 and between the hours 0600 and 1800 for each day. For each hour of interest, the wind speed (WIND_VEL) must be between 5 and 10 knots and the wind direction (WIND_DIR) must be between 200 and 250 degrees. If these conditions are satisfied, the values of the parameters wind direction, wind speed, and temperature (TEMP minus a one hour time delay at Midway Airport; and SO_2 emissions data for stacks one (SO2_S1) minus a two hour time delay and two (SO2_S2 minus a two hour time delay) from the University of Chicago heating plant are placed on the output data file. The printed results of this search are shown below.

SAMPLE INPUT 1

	MIDWAY TEMP	MIDWAY WIND_DIR	MIDWAY WIND_VEL	U_OF_C SO2_S1	U_OF_C SO2_S2
66/01/03 6	24.000	240.000	10.000	396.410	349.480
66/01/03 14	33.000	240.000	10.000	492.610	380.120
66/01/03 16	37.000	220.000	9.000	498.150	386.200
66/01/03 17	37.000	220.000	10.000	489.350	377.080
66/01/04 10	29.000	210.000	10.000	605.290	331.120
66/01/05 15	45.000	210.000	6.000	454.620	297.190
66/01/05 16	47.000	210.000	6.000	434.540	313.700
66/01/05 17	48.000	200.000	6.000	420.650	290.590
66/01/08 13	12.000	220.000	7.000	719.220	396.380
66/01/08 15	18.000	230.000	10.000	658.240	377.950
66/01/08 17	18.000	220.000	7.000	644.820	347.220

Choosing a Search Program

There are three search programs in the APICS system. Their function and the use of the computer storage units differ somewhat so there are times when it is appropriate to use each of them.

SEARCH1 is most useful for small searches and it MUST be used when more than 250,000 values will result from the search. There is a limit of about 150,000* search items if printing is desired.

SEARCH2 is more efficient for large searches and when printing the array is the only object. Up to 250,000 values may be printed and/or stored in the computer at once to be passed to the user-coded FORT routines for further processing. (About 150,000* values at a time is the limit using the retrieval programs described in Section 6.)

SEARCH3 is similar to SEARCH1 but data points are discarded if missing values are encountered.

There is a limit of 100 output parameters for the search routines as presently coded. This restriction could be removed by changing space allocations and tests as appropriate in each routine.

When running the search programs, it is important to specify blocked records for the OUTPUT and SCRATCH data sets. The usual DCB specifications are RECFM=VB with BLKSIZE=20000 for OUTPUT and 5000 for SCRATCH.

5.1 Formats of SEARCH Input and Output Tapes

The format of the information on the input and output tapes should NOT be of particular interest to users of the APICS system. However, the following information may be helpful.

---

*The limit of about 150,000 data items assumes a million bytes of fast core are available for the system.

5.1.1 Master File Format

The format of the master tape used as input for search runs is described below:

(a) The first record is always two bytes* long.

The first 8 bytes contain binary information with 4 bytes for the number of stations and 4 bytes for the total number of parameters, NPAR.

The last 12 characters gives the starting and ending dates for the tape; e.g., 680101690630 would specify a tape starting on 1/1/68 and ending on 6/30/69.

(b) The second record gives station information with 16 bytes of information for each station as follows:

The first 8 characters give the station name.

The next 8 bytes contain binary information with 4 bytes for the pointer to the first parameter for the station and 4 bytes for the number of parameters for the station.

[Number of bytes in the record = 16 * number of stations.]

(c) The third record on a master tape consists only of the 8-character parameter names given one after the other in the same order as the parameter values appear in subsequent records.

[Number of bytes in the record = 8 * NPAR.]

(d) Subsequent records then give the hourly parameter values with 24 hours worth of information in each record starting with midnight values.

The first 6 characters of each of these records contain the date; e.g., 690630 for 6/30/69.

The remainder of the record contains binary information with 4 bytes for each of the NPAR parameter values given for each hour of the day.

All parameter values are in floating point representation with a value of 10^{20} (i.e., 1.0E+20) used to indicate missing data.

[Number of bytes in each record = 6 + 4*NPAR*24]

---

*A byte contains 8 bits of binary information and may represent an alphanumeric character.

5.1.2 SEARCH Output Format

The format of the information on the output tape from a search run is described below:

(a) The first record is always 96 bytes long.

The first 80 characters contain the title as specified on the first card in the search input.

The next 16 bytes contain binary information with 4 bytes each for the number of stations, the number of parameters NPAR, and the number of observations NOBS (= the number of hours of observations that satisfied the search criteria).

(b) The second record contains information about each station specified in the search input in a format similar to that for the second record on the master tape. There are 16 bytes of information for each station.

The first 8 characters give the station name.

The next 8 bytes contain binary information with 4 bytes for the pointer to the first parameter for the station and 4 bytes for the number of parameters for the station.

[Number of bytes in the record = 16 * number of stations.]

(c) The third record contains station and parameter information for each parameter in the order specified in the search input.

There are 16 bytes for each parameter with 8 characters for the station name followed by 8 characters for the parameter name.

[Number of bytes in the record = 16 * NPAR.]

(d) Subsequent records then give the parameter values for each hour that satisfied the search criteria. The information for each observation time is in a separate binary record. There are thus NOBS records, each of which contains NPAR parameter values with 4 bytes for each in the order specified in the search input. All parameter values are in floating point representations just as they appeared on the master tape.

[Number of bytes in each record = 4 * NPAR.]

5.2 SEARCH1

This program uses only the fast core storage of the computer and in addition uses a scratch tape for collecting the data to be incorporated into the output array from the search. Thus it is somewhat cumbersome to use when searching for large amounts of information, but it is good for making smaller searches.

The master tape is rewound in between searches so the date-limits for successive searches may be in any order. The ddnames for which DD cards must be supplied are MASTER, OUTPUT, CARDS, and SCRATCH. See Appendix B for the deck setup to run a case.

5.3 SEARCH2

This program is quite different from SEARCH1 internally in that it utilizes both fast core and bulk storage* and has additional options for processing the output array. The input and output are essentially the same as for the other search routines. Only the input card specifying the print options, etc., has been changed to accommodate the additional options available in SEARCH2.

The three options PRINT, FILE, and FORT indicated in the input format description above may be specified in any combination. When a printout is the purpose of the search run, SEARCH2 is the most efficient since the others always create an output tape.

This is also the most efficient program for large searches, and in fact it is the only routine that can pass an array as large as 250,000 elements to a FORT routine ALL AT ONCE. This is the amount of storage that would be required for two years worth of date with 14 parameters for each hour, which is not an unusual amount of data to be concerned with at one time.

SEARCH2 stores the output array in bulk core instead of using a scratch tape. The listing of this program given in Appendix E includes the source code for BPOINTER, a general purpose routine used for dynamic storage allocation, as described in Reference 17. Also included is the listing of the code for the PL/I to Fortran interfacing routine described in References 11 and 12.

When running a search, DD cards must be supplied for ddnames of MASTER, OUTPUT, and CARDS. See Appendix B for sample deck setup.

---

*The term "bulk storage" refers to the IBM 2361 Large Core Storage device. If this unit is not available, modifications can be made so the SEARCH2 program can be used to take advantage of some of its other

5.4 SEARCH3

The program is similar to SEARCH1 but data points are omitted in case of missing data. Thus hourly parameter values are discarded if any of the values specified for that hour are missing (i.e., 10^{20}). This program should be used to search for data for programs which will not interpret missing data properly (e.g.: REGRES and STEPMDA).

APICS USER MANUAL

6.0 Access of Retrieved Data

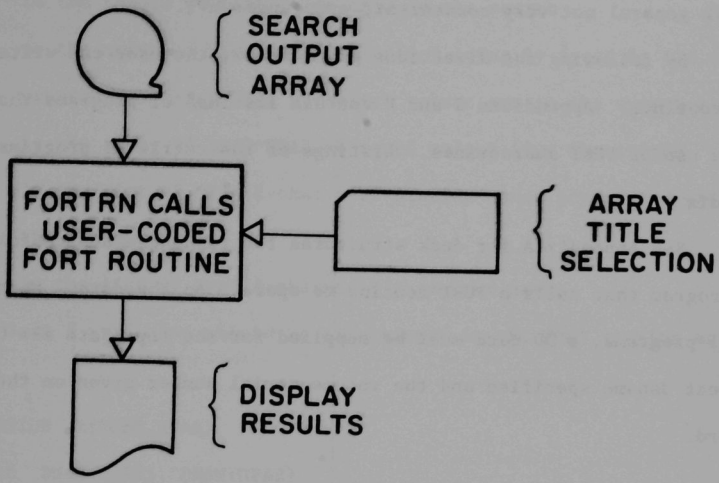
6.0 Access of Retrieved Data

A series of similar PL/1 programs (FORTRN, FORTRNS and FORTRNK) is available for retrieving the information from a search run, and for passing it to a user coded FORT subroutine for further processing. All of these programs perform essentially the same task. Program FORTRN reads the complete output array generated by a search, passes the information to a FORT subroutine, and when control is returned to it, it retrieves another output array from a search run, etc. All of the programs return an array with an ID that matches an input title card. The sequence of retrieving a data set and processing it in a user coded FORT routine is illustrated in Figure 6.1.

The FORTRN program rewinds the tape from the search run after retrieving each array, while FORTRNS continues searching through the tape for additional output arrays (if requested). The programs FORTRNS and FORTRNK pass the observed values for only a specified number of hours "worth" of data at a time instead of the whole data array. This feature makes it possible to process large amounts of data in a small amount of computer storage space, and also provides for passing different amounts of data to FORT routines to evaluate different data sets without having to modify the FORT routine.

An important feature of FORTRNK is that it uses PL1/FORTRAN interface routine (Refs. 11 and 12) to insure proper error handling, etc. This feature is most valuable to the user when he is coding new FORT routines that may not work correctly the first time. Also, FORTRNK was made more efficient for passing the information a little at a time.

It is important to use the correct form of this program with the various display and computational programs described in Sections 7 and 8, since there are different specifications for coding the various FORT routines. Also, FORTRNS cannot be used with all sequences of searches, but it is the most efficient retrieval program in some cases.



113-1144

Fig. 6.1 Analysis of Retrieved Data

Card input for the retrieval programs is described below. The FORT programs currently included in the APICS system require no additional input. If a user coded FORT routine does require additional input, it will be necessary to provide for an additional input data set FT05F001 separate from SYSIN for PL1. This is in general not very convenient, even under MVT.

By following the directions given below, the user can write his own FORT subroutines. Appendices G and H contain listings of programs that illustrate the use of FORT subroutines. Listings of the retrieval programs are given in Appendix F.

See Appendix A for deck structures for running a search followed by a FORTRN program that calls a FORT routine to operate on the data. When executing retrieval programs, a DD card must be supplied for the tape data set OUTPUT with the correct dsname specified and the volume serial number given on the ASP Setup card.

6.1 Input and FORT Routines for the FORTRN Program

Subroutine FORT is called with the five arguments - NPAR, NOBS, X, SNAM, and PNAM, where

NPAR is the number of parameters used in the search

NOBS is the number of hours of observation actually found by the search program for which NPAR parameter values are in the output array

X is the array of NOBS*NPAR observed values from the search

SNAM is the list of NPAR 8-character station names associated with the parameters

PNAM is the list of NPAR 8-character parameter names

Thus, the last 3 variables should be dimensioned in the FORT routine as follows:

```
DIMENSION  X(NOBS,NPAR)
REAL*8  SNAM(NPAR), PNAM(NPAR)
```

where the number 1 may be substituted for the NPAR's'. Although the station and parameter names could continue to be used to find parameter values of interest in FORT routines, it has been most convenient to simply "remember" the order of parameters specified for the search, and to specify the parameters by an index number based on this order. The station and parameter names are then only used for identifying printed and plotted output.

Input for the FORTRN program consists of a title card for each search output array of interest, followed by a STOP card (cols. 1-4). (See Appendix B).

FORTRAN statements to manipulate the data then normally involve DO loops with I going from 1 to NOBS and J going from 1 to NPAR with calculations performed using X(I,J).

6.2 Input and FORT Routines for the FORTRNS Program

All specifications are as described below for FORTRNK, but the title cards identifying the output arrays must be in the order used in the search.

6.3 Input and FORT Routines for the FORTRNK Program

The five arguments have the same meanings as described above for FORTRN. However, the FORT routine may be called several times for each search output array with n (a specified number of hours worth of) observations in the X array each time.

In any case, NOBS gives the number of hours of information in X for each call of FORT. When all of the observations for a search have been passed to the FORT routine, an additional call of FORT occurs with NOBS=0 after each search. Then FORT is called one last time after all search output arrays have been passed, with NOBS=-1. Coding must be incorporated into FORT to provide for the additional calls.

The input for FORTRNK is as described for FORTRN preceded by a card specifying n in cols. 1-5 where the value of n applies to all search output arrays included in this run. (In practice, the last set of values passed to FORT from a search run may contain less than n values, but the value of NOBS will be adjusted properly.) A value of 99999 may be specified for n if all the values are to be passed at once as for FORTRN, but provision must still be made for the 2 additional calls to FORT. Note that when NOBS=-1 it is appropriate to close out the buffers if Calcomp plots have been generated.

APICS USER MANUAL

7.0 Display Programs

7.0 Display Programs

Display programs in the APICS system may be used to:

1. Print the data specified in search-like input.

2. Generate time-series graphs on the Calcomp plotter (YOLYPLOT, a general-purpose plotting subroutine, is described in detail below enabling other types of plots to be obtained easily.)

3. Produce mesometeorological maps of weather and pollution data on the Calcomp plotter, including an outline map of the city or region.

4. Print graphical displays of frequency distribution in both 2 dimensions (JJPLOT) and one dimension (HIST). (The use of each of these subroutines is described in detail below enabling their use in other programs.)

All of these programs are written so as to be generally applicable to air pollution studies in Chicago or any other cities and regions. Minor changes that can be made in the programs to adapt them to other situations, purposes, and cities or regions are indicated in the writeups for each in this section.

Listings of the programs are given in Appendix G. Deck structures for executing the programs are given in Appendix B.

Except for the PRINT program, the display programs are written as FORT subroutines which operate on data generated in a SEARCH run. The data are normally retrieved, using a FORTRN input program which passes the information to the FORT routine as described in Section 6. Thus two jobsteps are normally required; however, these may be executed together as one job with the OUTPUT tape from the search run passed to the input program in the second jobstep. Alternatively, the programs could be run as one jobstep using SEARCH2 with FORT option specified as described in Section 5.3. For programs that generate Calcomp plots, it is necessary to use either the FORTRNS or FORTRNK input program in two jobsteps as described in Section 6. These routines arrange to call FORT one last time after all the data from the searches have been processed. Whichever approach is employed, the search output should be saved so that it is not necessary to repeat the search run if only the second part of the job is to be rerun.

When executing these programs as consecutive steps of the same job, control cards must be provided for the SEARCH and FORTRN/FORT steps as follows:

ASP Setup Cards

These must be provided for tapes with ddnames of MASTER and OUTPUT (also for PLOTTAPE for the Calcomp plotting programs described in Secs. 7.2 and 7.3). The volume serial numbers of the tapes are specified on these cards.

DD Cards for the SEARCH Step of the Job

These must be provided for MASTER, CARDS, and OUTPUT (also SCRATCH for SEARCH1) as described in Section 5. The specification for OUTPUT should include VOLUME=RETAIN and DISP=PASS.

DD Cards for the FORTRN/FORT Step of the Job

These must be provided for OUTPUT and, if plots are desired, for PLOTTAPE. The dsname for OUTPUT must agree with that used in generating the output array. Each of the title cards in the input for the FORTRN retrieval programs must agree with the corresponding one used in generating the output array of interest. For further information see Section 6. When an output array is already available from a search run, use control cards for a single jobstep as described in Section 6 and Appendix A.

7.1 PRINT Data Display Program

This program searches the master file for selected dates and prints parameters for specified stations. The output is similar to that obtained from search runs when the PRINT option is specified. This program is more efficient to use when printed output only is desired for up to 150,000 parameter values. Otherwise one of the search programs described in Section 5 should be used.

The input format for the PRINT program is as follows.

```
PRINT INPUT DESCRIPTION        (Note;  Information enclosed in parentheses is
                                       explanatory only and should not be punched.)

TITLE                          (80 characters - descriptive information)

DATE_LIMITS=ND;                (ND=number of date limits or 0 if all are desired)

(Date limit cards - each pair separated by at least one blank)
'YL/ML/DL','YU/MU/DU'          (lower date limit, upper date limit)

OUT_STATIONS=NS;               (NS=number of stations)
(Out station definition cards - one set per station)
STATION='NAME     '    PARAMS=NP;  (NP=number of output parameters or PARAMS=
                                      may be omitted if all are desired for
                                      this station)
(Output parameter definition cards - one to a parameter)
PARAM='NAME     '    DELAY=TD;  (TD=time delay in hours up to 24 or DELAY= may be
                                  omitted if no delay is desired for the parameter)
END    (This card must follow each case)
STOP   (This card must follow the run)
```

A sample input deck might be

```
SAMPLE INPUT
DATE_LIMITS=1;
'66/01/01','66/01/10'
OUT_STATIONS=2;
STATION='MIDWAY'  PARAMS=3;
PARAM='TEMP'   DELAY=1;
PARAM='WIND_DIR';
PARAM='WIND_VEL';
STATION='U_OF_C'  PARAMS=2;
PARAM='SO2_S1'   DELAY=2;
PARAM='SO2_S2'   DELAY=2;
END
STOP
```

This example will print the temperature (minus a one hour delay), wind direction, and wind speed at Midway Airport and the SO_2 emission data for stacks one and two (minus a two hour delay) for the University of Chicago heating plant for the first ten days of January 1966. The printed output is shown below.

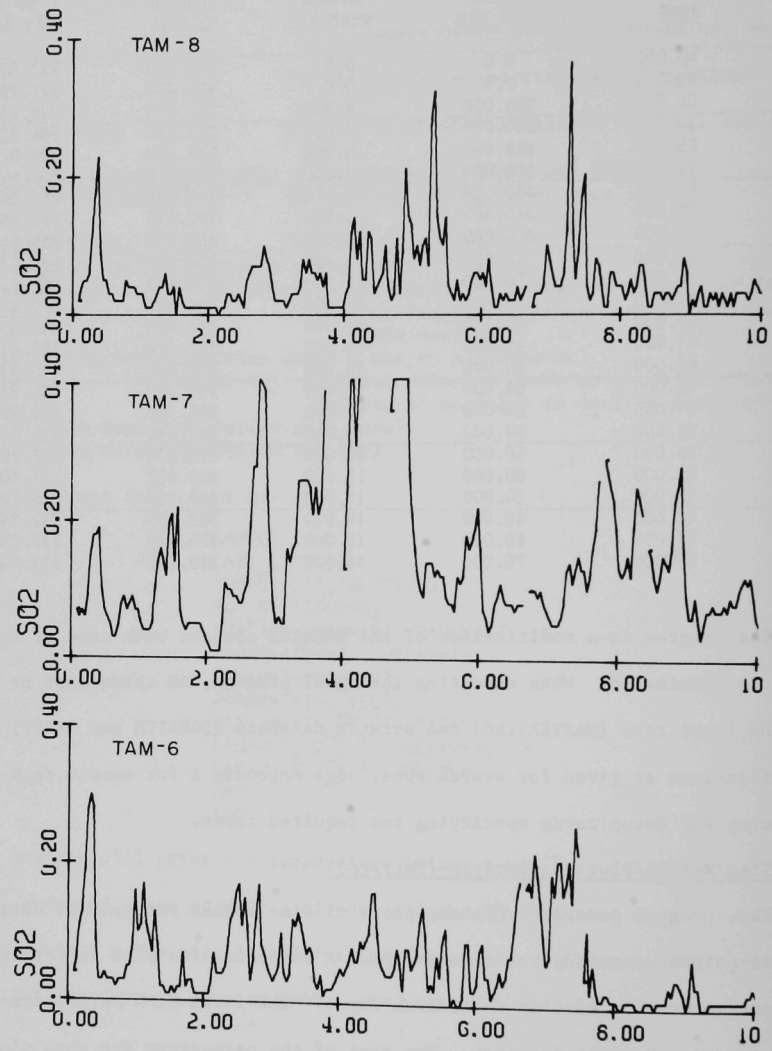
SAMPLE INPUT
DATE:66/01/01

HOUR	MIDWAY TEMP	MIDWAY WIND_DIR	MIDWAY WIND_VEL	U_OF_C SO2_S1	U_OF_C SO2_S2
1	40.000	0.0	0.0	375.080	337.570
2	35.000	0.0	0.0	325.260	327.100
3	34.000	300.000	4.000	280.460	238.620
4	33.000	330.000	4.000	289.230	238.620
5	79.000	360.000	4.000	283.390	229.070
6	31.000	350.000	3.000	286.310	238.620
7	30.000	350.000	6.000	315.520	257.710
8	30.000	360.000	5.000	327.210	262.480
9	30.000	360.000	5.000	362.270	269.640
10	29.000	60.000	5.000	368.110	298.270
11	33.000	50.000	11.000	415.610	365.510
12	37.000	70.000	10.000	389.640	362.490
13	39.000	50.000	14.000	370.160	362.490
14	41.000	50.000	13.000	357.170	350.410
15	42.000	60.000	14.000	353.920	338.330
16	42.000	50.000	13.000	344.180	335.310
17	41.000	60.000	12.000	344.180	332.290
18	39.000	50.000	13.000	366.910	338.330
19	38.000	50.000	12.000	339.910	308.790
20	37.000	60.000	15.000	363.350	327.100
21	39.000	50.000	12.000	369.220	329.720
22	40.000	60.000	14.000	366.290	332.340
23	41.000	60.000	13.000	375.080	332.340
24	40.000	70.000	15.000	369.220	332.340

The program is a modification of the SEARCH1 program with some of the features eliminated. When executing the PRINT program, DD cards must be supplied for the input tape (MASTER) and two scratch datasets (SCRATCH and CARDS) with specifications as given for search runs. See Appendix B for sample deck structures, including ASP Setup cards specifying the required tapes.

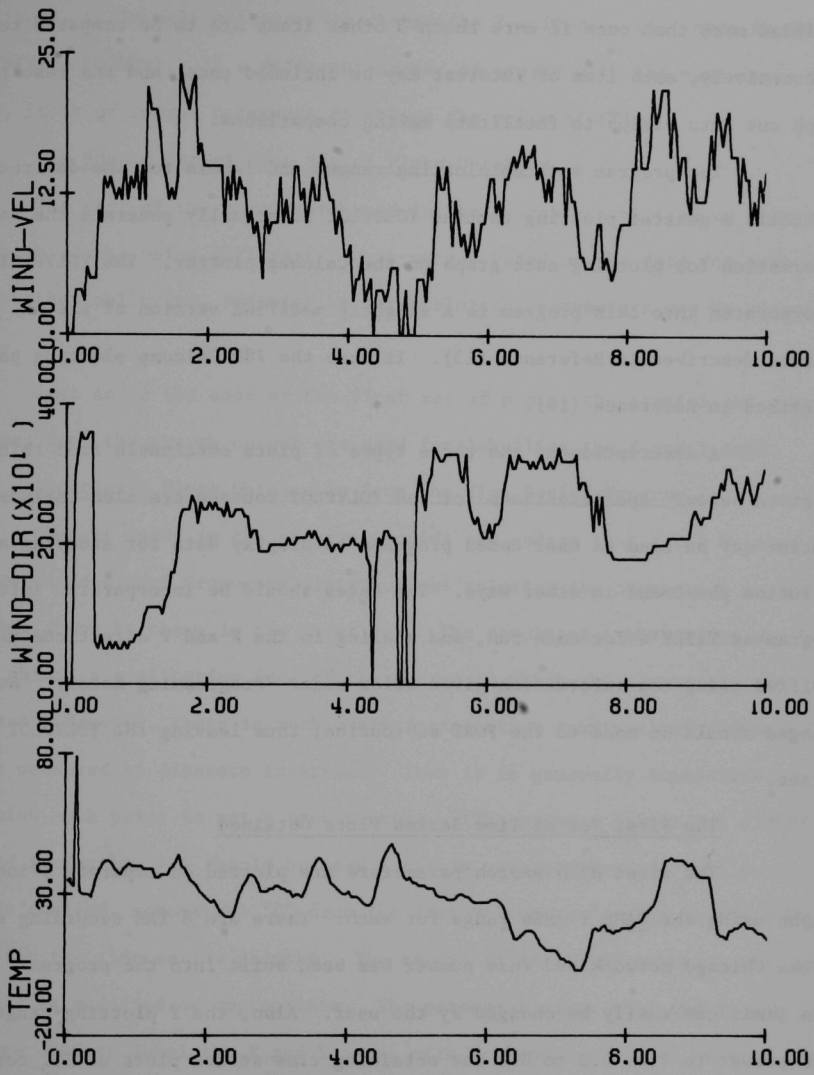
7.2 Time Series Plotting Program (YOLYPLOT)

This program generates Calcomp plots of time series for sets of data items resulting from searching the master tape. A separate line plot is obtained for each parameter specified in the search input. The first NTAM parameters are plotted using the same Y-scale for each. The rest of the parameters are then plotted with appropriate Y-scales determined from the data for each plot. Sets of up to 4 parameters are plotted in 2 inch high graphs one above another across the 10 inch wide Calcomp graph paper. Except for missing data, hourly values are plotted, and the c he X-axis. (See Figs. 7.1 and 7.2.)



113-1145

Fig. 7.1 Time Series Plots for TAM Data



113-1146

Fig. 7.2 Time Series Plots for Weather Data

The data of interest are specified in a search run where the order of parameters in the search input determines the plotting order. A parameter may be included more than once if more thatn 3 other items are to be compared to it. Alternatively, each item of interest may be included once, and the resulting graph cut into strips to facilitate making comparisons.

The program sets up plotting ranges and labels for the desired graphs and calls a general plotting routine YOLYPLOT to actually generate the tape of information for plotting each graph on the Calcomp plotter. The YOLYPLOT routine incorporated into this program is a slightly modified version of the ANL plotting routine described in Reference (13). It uses the 780 Calcomp plotting package as described in Reference (18).

A description of the three types of plots obtainable from this program is given below. Specifications for the YOLYPLOT routine are also included so the routine may be used in user coded programs to display data for studying air pollution phenomena in other ways. The dates should be incorporated into the program as TITLE's for each run, and scaling in the X and Y directions may be modified using the information given below under "Programming Notes." Such changes should be made to the FORT subroutine; thus leaving the YOLYPLOT routine intact.

The First Set of Time Series Plots Obtained

The first NTAM search parameters are plotted on separate 2-inch high graphs using the SAME Y axis range for each. There are 8 TAM recording stations in the Chicago network and this number has been built into the program. However, this limit can easily be changed by the user. Also, the Y plotting range has been preset to from 0.0 to 0.4 for obtaining time series plots of SO_2 concentrations at all eight TAM stations. This range can also be altered by the user.

The station and parameter names from the search output are used for labelling the X and Y axes. Thus if a different parameter is specified in the search input, it is only necessary to change the Y-axis plotting range in the code (YMIN to SO2MAX). If a different number of graphs with the same scale is desired, it is of course necessary to change the value of NTAM.

The Second Set of Time Series Plots Obtained

The rest of the parameters used in the search are plotted on separate 2-inch high graphs using different Y axis ranges based on the data for each. Thus any other pollution, emission or meteorological parameters may be accommodated.

Just as in the case of the first set of plots, the station and parameter names from the search output are used for labelling the X and Y axes.

Additional Plots Obtainable from the Time Series Program

Comment cards in the program provide the coding that could be used to obtain two-dimensional frequency distributions relating the parameters in the second set above to those in the first set. The resulting symbol plots may be used to obtain a qualitative idea of the degree to which the parameters are correlated. However, there is much overprinting since many weather phenomena are only measured at discrete intervals. Thus it is generally impossible to distinguish each point to get a true picture. (The printed output for JJPLOT described in Section 7.4 should be used for this purpose instead.) If such plots are desired, remove the "C" from the last 9 "comment" cards in the FORT routine for the time series program. The second title TITLE2 contains the date and should be modified as appropriate. The parameter names from the search output are used to label the X and Y axes, with no mention made of station names. The plots will contain the symbol "X" at each data point.

Sample Search Input

```
SAMPLE DATA FOR GRAPH PLOTTING OF TIME SERIES AND 2-DIM FREQ DIST. / CCC
PRINT='NO';
DATE_LIMITS=1;   '68/01/01','68/01/31'
HOUR_LIMITS=0;
MET_STATIONS=0;
OUT_STATIONS=10;
STATION='TAM_1'      PARAMS=1;
PARAM='SO2';
STATION='TAM_2'      PARAMS=1;
PARAM='SO2';
STATION='TAM_3'      PARAMS=1;
PARAM='SO2';
STATION='TAM_4'      PARAMS=1;
PARAM='SO2';
STATION='TAM_5'      PARAMS=1;
PARAM='SO2';
STATION='TAM_6'      PARAMS=1;
PARAM='SO2';
STATION='TAM_7'      PARAMS=1;
PARAM='SO2';
STATION='TAM_8'      PARAMS=1;
PARAM='SO2';
STATION='MIDWAY'     PARAMS=4;
PARAM='PRESS';
PARAM='TEMP';
PARAM='WIND_DIR';
PARAM='WIND_VEL';
STATION='TURNER'     PARAMS=1;
PARAM='TURNER_C';
STATION='NORCO'      PARAMS=1;
PARAM='MIX_HT';
END
STOP
```

Sample Input for Time Series Program

```
99999
SAMPLE DATA FOR GRAPH PLOTTING OF TIME SERIES AND 2-DIM FREQ DIST. / CCC
STOP
```

7.2.1 Programming Notes on the Time Series Program

Some other features of the graphs may be altered by making simple changes in the FORT program. Such changes should not be required in normal usage of the APICS system, but the following information may be useful.

(1) The height of the graphs may be altered to obtain fewer or more graphs together within the 10 inch wide Calcomp graph paper. The present value of this height (TWO) is 2.0. This produces up to 4 2-inch graphs with $\frac{1}{2}$ inch in between each. The value of TWO could be reset to other values to produce the following results.

Single 10 inch graphs would be obtained if TWO = 10.0

Two 4.5 inch graphs would be obtained if TWO = 4.5

Three 3 inch graphs would be obtained if TWO = 3.0

Five $1\frac{1}{2}$ inch graphs would be obtained if TWO = 1.5

(2) The scale along the X axis may be altered to stretch the graphs out further or to condense them. At present, DAYSPR = 2.0 so there are 2 days per inch plotted in the X direction, but this may be changed to any desired value.

(3) The number appearing along the X axis is the number of days starting at 1 assuming there are 24 hours of information each day. These figures are involved in the calculation of DX, XMAX, and the A array.

(4) The output array retrieved from the search contains NPAR hourly parameters values at each of NOBS observation times. These numbers are used as the basis for determining

the total number of time series plots, and

the number of observation times for plotting data for each graph.

The value of NOBS appears in the lower left hand corner of each graph. If it would be appropriate to have some other number appear there to identify each graph, simply replace the 8th argument in the YOLYPLOT calls.

7.2.2 YOLYPLOT Programming Information

Given X and Y arrays, labels, and plotting range specifications, YOLYPLOT generates a complete Calcomp graph. Either a line or symbol plot may be specified and the routine can find suitable plotting ranges if none are provided.

The plotting program described in Reference (13) has been modified to incorporate the following features for use in the APICS system.

(1) The height of the graphs may be less than 10 inches. (Entry point TWOTEN)

(2) Instead of plotting additional information on a graph for subsequent calls, complete graphs are plotted one above another to facilitate comparing air pollution phenomena. The routine starts on a new section of graph paper automatically when no more graphs will fit on top of those already plotted on the 10 inch wide paper.

(3) In determining plotting ranges, values over 9999.0 are considered as missing data and ignored.

(4) Line plotting is suppressed for missing data by raising and lowering the pen as appropriate.

(5) If all the data points are missing for a graph, it is not plotted.

Complete specifications for using the YOLYPLOT routine in the APICS system are given below:

To adjust the height of the graph

```
CALL TWOTEN (HGT)
```

where HGT is the height of each graph in inches. The next graph will start $\frac{1}{2}$ inch above this if another graph will fit in the 10 inches. If not, the next graph will start at the base line about 6 inches past the end of the last graph. This entry point may be called anytime.

For each graph to be plotted one above another

```
CALL YOLYPLOT (X,Y,XMAX,XPINCH,YMAX,
NCYCLE,N,K,XLABEL,YLABEL,IFSTCALL,
INCX,INCY,XMIN,YMIN,SYM,TITLE,NN)
```

where the 18 arguments have the following meanings:

X: The values to be plotted in the horizontal direction.

Y: The values corresponding to the X's for plotting in the vertical direction.

XMAX: The maximum value of X.

XPINCH: The desired change in X per inch of plot in the X direction. Thus the length of the graph will be (XMAX - XMIN)/XPINCH.

YMAX: The maximum value of Y.

NCYCLES: A control number. NCYCLES = 0 for plotting Y vx. X. (Values > 0 signify log plots but this facility has not been adapted for the APICS system.) NCYCLES = -1 signifies *end of all plotting*.

N: Another control number equal in magnitude to the number of data points. If N > 0 the character in SYM is plotted at each data point, but if N < 0 a *line plot* is generated and the pen is raised for missing data.

K: A number $<8.4 \cdot 10^6$ to identify the graph. This appears in the lower left corner.

XLABEL: Up to 8 Hollerith characters to label the X axis; e.g.: 8HTIME.

YLABEL: Up to 8 Hollerith characters to label the Y axis; e.g.: 8HPRESSURE.

IFSTCALL: Must be 1 the first time the routine is called and to force a graph to be placed on the base line, or set to any other value to plot additional graphs above each other. (If the graph will not fit in the 10 inches available, it will automatically be placed on the base line after the last graph.)

INCX: The spacing between consecutive X locations in memory, normally 1 if X is type REAL*4.

INCY: Analogous to INCX.

XMIN: The minimum value of X. (If XMAX < XMIN, then XPINCH must be negative.)

YMIN: The minimum value of Y

SYM: An alphanumeric character used as a plotting character; e.g.: 1HX (or 1H. for line plots, since this symbol appears at the first point plotted).

TITLE: A 72 character title to appear at the left of the graph.

NN: Irrelevant but must be present.

Also note that it is possible for the routine to determine ranges for plotting. This option is achieved by setting XMIN = XMAX and/or YMIN = YMAX before the first call for each graph. The ranges (calculated by the DXDY subroutine) will then be based on the data given for that call, and will not exceed 10 (or TWO) inches.

To close out all Calcomp plotting

CALL YOLYPLOT with arguments

as above but with the value of the sixth argument NCYCLE set to -1; all other arguments must be present.

7.3 Mesometeorological Mapping Program

The program CITYMAP generates Calcomp maps showing hourly weather phenomena and pollution data at various TAM and meteorological stations in an area. An outline of the city is included. A search program is first used to store the required parameter values in an array containing hourly information for each day of interest. Maps may then be obtained for every n^{th} hour starting with midnight, where n is specified in the input for the retrieval program. Figure 7.3 is an example of such a map.

A master tape containing data for all the required stations must be available, but the number of airport weather stations may vary from one to the number of station coordinates built into the program. The program is structured to accommodate arbitrary numbers of station types. They are listed below together with the number of parameters provided in the version of the program used for the Chicago study.

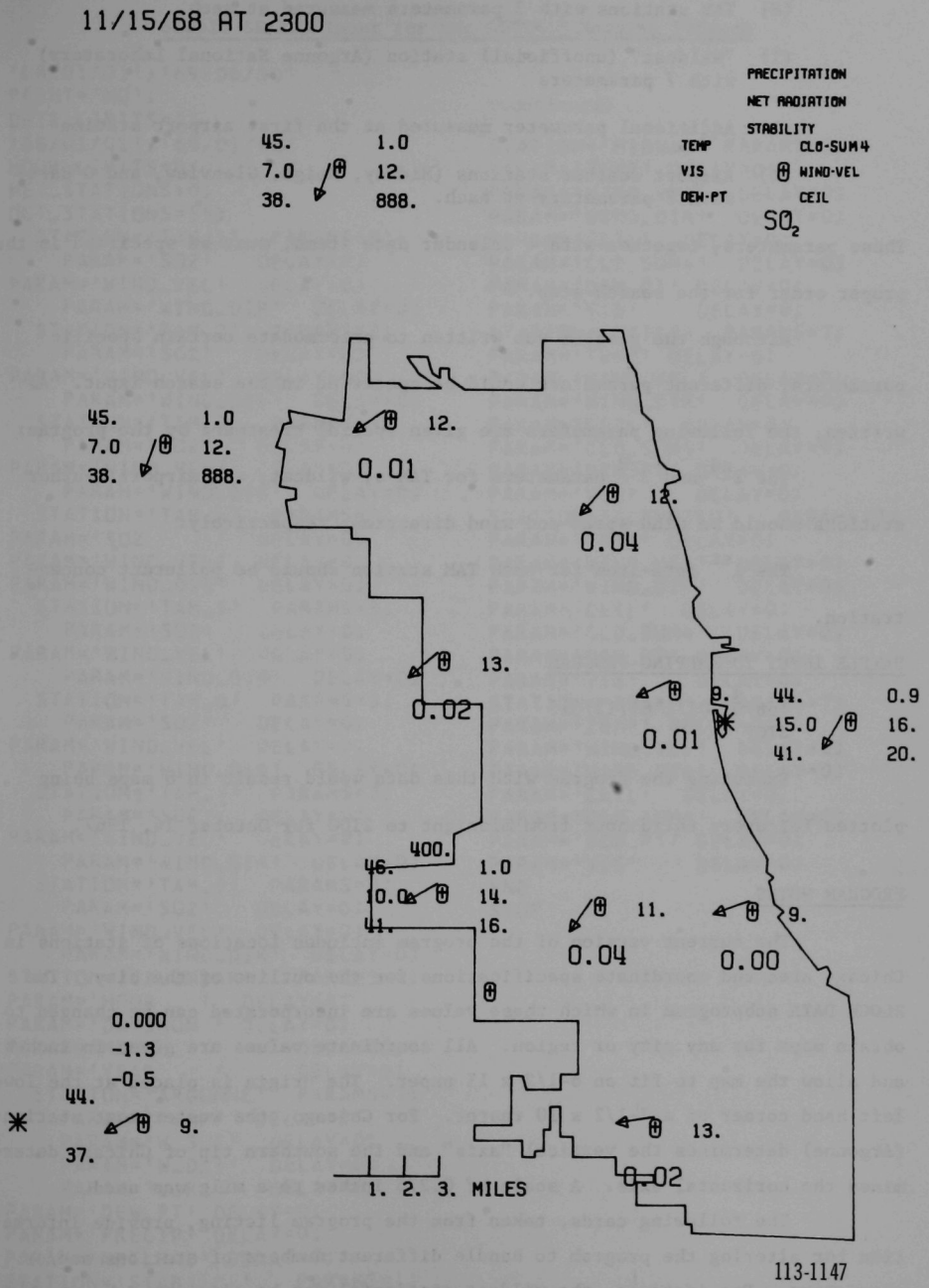


Fig. 7.3 Mesometeorological Map for Chicago

(8) TAM stations with 3 parameters measured at each

(1) "Wildcat" (unofficial) station (Argonne National Laboratory) with 7 parameters

(1) Additional parameter measured at the first airport station

(4) Airport weather stations (Midway, Meigs, Glenview, and O'Hare) with 7 parameters at each.

These parameters, together with 4 Calendar date items, must be specified in the proper order for the search step.

Although the program was written to accommodate certain specific parameters, different parameters could be specified in the search input. As written, the following parameters are given special treatment by the program:

The 2nd and 3rd parameters for TAM's, wildcat, and airport weather stations should be wind speed and wind direction, respectively.

The 1st data item for each TAM station should be pollutant concentration.

SAMPLE INPUT TO MAPPING PROGRAM

```
3
'66/01/01','67/12/31'
STOP
```

Executing the program with this data would result in 8 maps being plotted for every third hour from midnight to 2100 for October 14, 1967.

PROGRAM NOTES

The current version of the program includes locations of stations in Chicago area and coordinate specifications for the outline of the city. The BLOCK DATA subprogram in which these values are incorporated can be changed to obtain maps for any city or region. All coordinate values are given in inches and allow the map to fit on 8-1/2 x 11 paper. The origin is placed at the lower left hand corner of a 7-1/2 x 10 raster. For Chicago, the westernmost station (Argonne) determines the vertical "axis" and the southern tip of Chicago determines the horizontal axis. A scale of 0.275 inches to a mile was used.

The following cards, taken from the program listing, provide information for altering the program to handle different numbers of stations and/or parameters. For instance, the wildcat station(s) may be eliminated by recompiling the program with NWILDS and NWILDP set to 0 and with the coordinates of the station deleted in the Block Data program.

Sample SEARCH Input for the CITYMAP Mapping Program

```
'68/01/01','69/06/30'
PRINT='NO';
DATE_LIMITS=1;
'68/01/01','68/01/02'
HOUR_LIMITS=0;
MET_STATIONS=0;
OUT_STATIONS=15;
  STATION='TAM_1'  PARAMS=3;
    PARAM='SO2'    DELAY=0;
PARAM='WIND_VEL'  DELAY=0;
    PARAM='WIND_DIR'  DELAY=0;
  STATION='TAM_2'  PARAMS=3;
    PARAM='SO2'    DELAY=0;
PARAM='WIND_VEL'  DELAY=0;
    PARAM='WIND_DIR'  DELAY=0;
  STATION='TAM_3'  PARAMS=3;
    PARAM='SO2'    DELAY=0;
PARAM='WIND_VEL'  DELAY=0;
    PARAM='WIND_DIR'  DELAY=0;
  STATION='TAM_4'  PARAMS=3;
PARAM='SO2      '  DELAY=0;
PARAM='WIND_VEL'  DELAY=0;
PARAM='WIND_DIR'  DELAY=0;
  STATION='TAM_5'  PARAMS=3;
    PARAM='SO2'    DELAY=0;
PARAM='WIND_VEL'  DELAY=0;
    PARAM='WIND_DIR'  DELAY=0;
  STATION='TAM_6'  PARAMS=3;
    PARAM='SO2'    DELAY=0;
PARAM='WIND_VEL'  DELAY=0;
    PARAM='WIND_DIR'  DELAY=0;
  STATION='TAM_7'  PARAMS=3;
    PARAM='SO2'    DELAY=0;
PARAM='WIND_VEL'  DELAY=0;
    PARAM='WIND_DIR'  DELAY=0;
  STATION='TAM_8'  PARAMS=3;
    PARAM='SO2'    DELAY=0;
PARAM='WIND_VEL'  DELAY=0;
    PARAM='WIND_DIR'  DELAY=0;
STATION='CALENDAR'   PARAMS=4;
PARAM='HOUR    '  DELAY=0;
PARAM='DAY_MON '  DELAY=0;
PARAM='MONTH   '      DELAY=0;
PARAM='YEAR    '      DELAY=0;
  STATION='ARGONNE'  PARAMS=7;
    PARAM='TEMP'  DELAY=0;
    PARAM='W_S_1'  DELAY=0;
    PARAM='W_D_1'  DELAY=0;
    PARAM='L_R_4'  DELAY=0;
PARAM='DEW_PT' DELAY=0;
PARAM='PRECIP' DELAY=0;
PARAM='NET_RAD' DELAY=0;
STATION='STAB    '   PARAMS=1;
PARAM='TURNER_B'         DELAY=0;
```

(continued)

```
STATION='MIDWAY' PARAMS=7;
PARAM='TEMP' DELAY=0;
PARAM='WIND_VEL'  DELAY=0;
PARAM='WIND_DIR'  DELAY=0;
PARAM='CEIL'  DELAY=0;
PARAM='CLD_SUM4'  DELAY=0;
PARAM='DEW_PT' DELAY=0;
PARAM='VIS'    DELAY=0;
STATION='MEIGS'  PARAMS=7;
PARAM='TEMP' DELAY=0;
PARAM='WIND_VEL'  DELAY=0;
PARAM='WIND_DIR'  DELAY=0;
PARAM='CEIL'  DELAY=0;
PARAM='CLD_SUM4'  DELAY=0;
PARAM='DEW_PT' DELAY=0;
PARAM='VIS'    DELAY=0;
STATION='GLENVIEW'   PARAMS=7;
PARAM='TEMP' DELAY=0;
PARAM='WIND_VEL'  DELAY=0;
PARAM='WIND_DIR'  DELAY=0;
PARAM='CEIL'  DELAY=0;
PARAM='CLD_SUM4'  DELAY=0;
PARAM='DEW_PT' DELAY=0;
PARAM='VIS'    DELAY=0;
STATION='OHARE'  PARAMS=7;
PARAM='TEMP' DELAY=0;
PARAM='WIND_VEL'  DELAY=0;
PARAM='WIND_DIR'  DELAY=0;
PARAM='CEIL'  DELAY=0;
PARAM='CLD_SUM4'  DELAY=0;
PARAM='DEW_PT' DELAY=0;
PARAM='VIS'    DELAY=0;
END
STOP
```

```
C N(TYPE)(S OR P)  = NUMBER OF STATIONS OR PARAMS OF THAT TYPE
C I(TYPE)(S OR Z)  = INDEX OF 1ST STATION OR PARAM OF THAT TYPE
C MAY SET IRRELEVANT ITEMS TO ZERO BUT DO NOT OMIT ANY INITIALIZATIONS
      AMILE=.275
      NTAMS=8
      NTAMP=3
      ICALZ=NTAMS*NTAMP+1
      IWILDZ=ICALZ+4
      NWILDS=1
      NWILDP=7
      IWILDS=NTAMS+1
      IADDZ=IWILDZ+NWILDS*NWILDP
      NADDP=1
      IAIRS=IWILDS+NWILDS
      IAIRZ=IADDZ+NADDP
      NAIRP=7
```

The layout of the information associated with each station on the map is indicated in the following comment cards in the listings of the FORT and KEYS routines.

```
C
C                          10=PRECIPITATION                    (WILDCAT STATIONS
C                           9=NET RADIATION                      OR ADDED TO THE
C                           8=STABILITY                           FIRST AIRPORT)
C   1=TEMPERATURE        ------------------      5=CLOUD COVER
C   7=VISIBILITY         | 3=WIND DIRECTION |    2=WIND VELOCITY
C   6=DEW POINT          ------------------      4=CEILING
C                           1=TAM SO2 (PPM)
C                            =1ST PARAMETER
C                             FOR TAMS ONLY
C
```

This arrangement was adopted in order to allow 0.3 inches around each station for the SWD routine to draw an arrow for the wind.

The following comment cards from the program give the actual coordinate displacements for plotting the information in each of the 10 locations around a station.

```
C
C                          10=(-.24,+.76)
C                           9=(-.24,+.53)
C                           8=(.24,+.30)
C   1=(-.6,+.16)         ----------------      5=(+.3,+.16)
C   7=(-.6,+.07)         |  HEIGHT=.10  |      2=(+.3,-.07)
C   6=(-.6,-.30)         ----------------      4=(+.3,-.30)
C                           1=(-.24,-.45)
C                             HEIGHT=.14
C                            FOR SO2 ONLY
C
```

Wind speeds appear to the right of each station and SO_2 concentrations appear below each TAM station. At most 6 weather data items appear on either side of the stations (airport and wildcat). Additional parameter values appear above selected airport and wildcat stations. Depending upon the spacing of the stations, there may be room for printing the values of additional parameters below some stations.

Additional features appearing on each map include

"L" brackets at each corner,

a mileage scale,

the date and hour, and

a key with all the parameter names.

The information for two of the stations has been placed an inch to the right (East) of their actual locations in order to consolidate the graph and to avoid overprinting values for adjacent stations.

The maps are generated by making appropriate calls to the 780 Calcomp plotting package described in Reference (18).

7.4 Printed Graphical Displays

A FORT routine has been written to demonstrate the use of two general routines which produce printed frequency distributions. The JJPLOT routine tabulates 2-dimensional frequency distributions (Figure 7.4), and the HIST routine prints histogram frequency distributions (Figure 7.5) for a single variable. The JJPLOT routine can also be used to obtain a 2-dimensional array of frequencies for further processing.

The program expects to operate on the same set of search data as is described in Section 7.2 for obtaining time series output. The first NTAM parameters are pollutant (SO_2) concentrations for the TAM stations. These are followed by airport weather data for pressure, temperature, etc. The program can be used with different sets of parameters following the TAM data since the parameter names from the searches are incorporated into the titles for the various types of output.

JANUARY 1967 TAM SO_2 VS. TEMPERATURE

0.500	19	2	2	3	1	0
0.450	0	0	0	0	0	0
0.400	0	0	0	0	0	0
0.350	0	0	0	0	1	0
0.300	0	0	1	1	1	0
0.250	0	0	2	3	2	0
0.200	0	0	8	12	3	0
0.150	0	3	14	19	3	0
0.100	4	13	35	49	5	1
0.050	62	115	113	135	87	25
0.0						
	-10	+10	+30	+50		

113-1148

Fig. 7.4 Two-D Frequency Distribution of SO_2 Concentration vs. Temperature

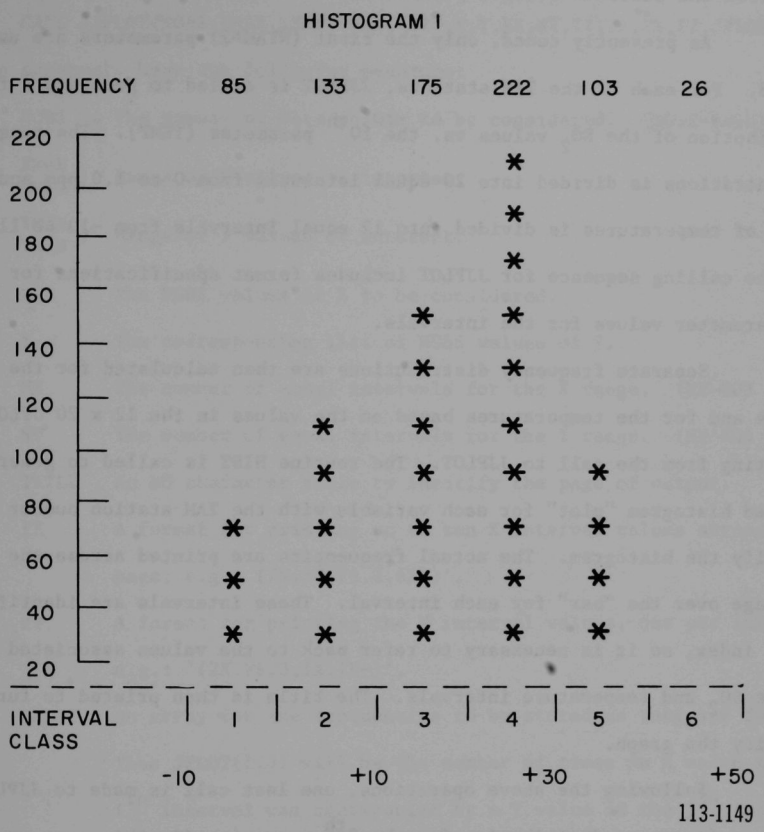


Fig. 7.5 Histogram Plot of Temperature Distribution

In any case, the date information in the first 16 characters of the title should be changed as appropriate, or the program can be changed to use the date information from the calendar station parameters.

As presently coded, only the first (NTAM+2) parameters are used, where NTAM=8. For each of the TAM stations, JJPLOT is called to generate a frequency distribution of the SO_2 values vs. the 10th parameter (TEMP). The range of SO_2 concentrations is divided into 20 equal intervals from 0 to 1.0 ppm and the range of temperatures is divided into 12 equal intervals from -10 to 110 degrees F. The calling sequence for JJPLOT includes format specifications for printing the parameter values for the intervals.

Separate frequency distributions are then calculated for the SO_2 values and for the temperatures based on the values in the 12 x 20 JPLOT array resulting from the call to JJPLOT. The routine HIST is called to generate a printed histogram "plot" for each variable with the TAM station number used to identify the histogram. The actual frequencies are printed across the top of the page over the "bar" for each interval. These intervals are identified only by an index, so it is necessary to refer back to the values associated with each of the SO_2 and temperature intervals. The title is then printed to further identify the graph.

Following the above operations, one last call is made to JJPLOT to obtain a frequency distribution for the 9th parameter (in this case pressure) vs. the 10th parameter (in this case temperature) with 16 intervals for pressure between 28 and 32 (inches of mercury). The 12 temperature ranges are as before from -10 to 110. These routines were used extensively in studying the air pollution effects in the Chicago area. Complete information is given below for using the JJPLOT and HIST subroutines.

7.4.1 Use of Subroutine JJPLOT

Given a set of values (X,Y) and interval specifications JJPLOT may be used to print 2-dimensional frequency distributions for combinations of the items

in the data array. An array JPLOT containing the frequencies is also obtained and may be used in further computations.

The calling sequence to use the program is

CALL JJPLOT(NOBS,XMAX,XMIN,YMAX,YMIN,X,Y,NX,NY,TITLE,FX,FY,JPLOT)

where the arguments have the following meanings:

NOBS is the number of data points to be considered. (NOBS $\leq$ 2400)

XMAX, XMIN) Range of X values of interest.

YMAX, YMIN) Range of Y values of interest.

X The NOBS values of X to be considered.

Y The corresponding list of NOBS values of Y.

NX The number of equal intervals for the X range. (NX $\leq$ 20)

NY The number of equal intervals for the Y range. (NY $\leq$ 50)

TITLE An 80 character title to identify the page of output.

FX A format for printing up to ten X interval values across the page; e.g.: (7X,10(F6.2,6X))'.

FY A format for printing the Y interval values, one per line; e.g.: '(2X,F6.3,1X,1H-)'.

JPLOT An array for the frequencies to be stored as they are found. Thus JPLOT(I,J) will be the number of times an X value in the ith interval was accompanied by a Y value in the jth interval. (JPLOT <u>MUST</u> be dimensioned 25 x 50 as presently coded).

7.4.2 <u>Use of the Subroutine HIST</u>

Given a series of up to 20 values defining a frequency distribution, HIST may be used to print the information in the form of a histogram.

The calling sequence to use this program is

CALL HIST(NUM,FREQ,NF)

where NUM is a number to identify the histogram ($\leq$ 999),

FREQ is the vector of NF frequencies, and

NF is the number of intervals for which frequencies are given ($\leq$ 20).

This routine is described in IBM's Scientific Subroutine Package. (Ref. 6)

APICS USER MANUAL

8.0 Computational Programs

8.0 Computational Programs

Most of the computational programs are written as FORT subroutines which operate on data generated in a SEARCH run. The data are normally retrieved, using a FORTRN input program which passes the information to the FORT routine as described in Section 6. Thus two jobsteps are normally required which may be executed together as one job with the OUTPUT tape from the search run passed to the input program in the second jobstep. Alternatively the programs could be run as one jobstep using SEARCH2 with the FORT option specified as described in Section 5.3. In any case, the search output should be saved so that it is not necessary to repeat the search run if only the second part of the job is to be rerun.

When executing these programs as consecutive steps of the same job, control cards must be provided for the SEARCH and FORTRN/FORT steps as follows:

ASP Setup Cards

These must be provided for tapes with ddnames of MASTER and OUTPUT. The volume serial numbers of the tapes are specified on these cards.

DD Cards for the SEARCH Step of the Job

These must be provided for MASTER, CARDS, and OUTPUT (also SCRATCH for SEARCH1) as described in Section 5. The specification for OUTPUT should include VOLUME=RETAIN and DISP=PASS.

DD Cards for the FORTRN/FORT Step of the Job

These must be provided for OUTPUT. The dsname for OUTPUT must agree with that used in generating the output array.

Each of the title cards in the input for the FORTRN retrieval programs must agree with the corresponding one used in generating the output array of interest. For further information see Section 6. When an output array is already available from a search run, use control cards for a single jobstep as described in Section 6 and Appendix A.

8.1 Linear Regression Analysis

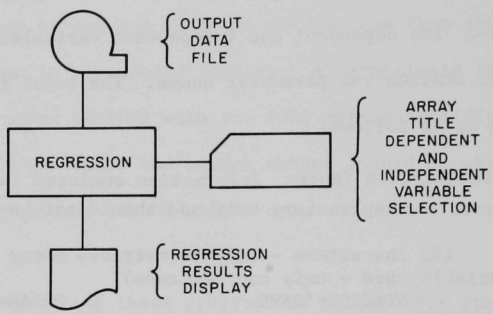
An input program exists in APICS which accepts data from an output file generated by the SEARCH program. This program in turn calls a set of subroutines in the IBM Scientific Subroutine Package[6] which is designed to perform linear regression analysis. The dependent and independent variables are selected by inputing the proper station and parameter names. The input format to the regression program is as follows:

```
REGRES INPUT DESCRIPTION (Note:  Information enclosed in parentheses is
                          explanatory only and should not be punched)

TITLE            (80 characters - only to retrieve array on output tape)
(Dependent variable card - only one per case)
TYPE='DEP'          STATION='NAME    '            PARAM='NAME    ';
(Independent variable cards)
TYPE='IND'          STATION='NAME    '            PARAM='NAME    ';
END            (This card must follow each case)
STOP           (This card must follow the run)
```

Since the search process is completely general, every parameter in the master file is a possible candidate for use as either an independent or dependent variable in a regression study. (Fig. 8.1)

The search input specifications for the sample case given below were also included as the sample for the Search program in Section 5.0. The output array from this search contains selected parameter values for Midway wind direction, velocity, and temperature, and for SO_2 emissions from stack 1 and stack 2 for the U. of C. plant. Only the first, third, and fifth of these parameters enter into the regression analysis.



112-9728

Fig. 8.1 Regression Analysis

Sample SEARCH Input

```
SAMPLE INPUT 1
PRINT='NO';
DATE_LIMITS=1;
'66/01/01','66/01/10'
HOUR_LIMITS=1;        6,18
MET_STATIONS=1;
STATION='MIDWAY'    PARAMS=2;
PARAM='WIND_VEL'     BANDS=1;     5,10
PARAM='WIND_DIR'    BANDS=1;    200,250
OUT_STATIONS=2;
STATION='MIDWAY'   PARAMS=3;
PARAM='WIND_DIR';
PARAM='WIND_VEL';
PARAM='TEMP'  DELAY=1;
STATION='U_OF_C'  PARAMS=2;
PARAM='SO2_S1'  DELAY=2;
PARAM='SO2_S2'  DELAY=2;
END
STOP
```

Sample Input

```
SAMPLE INPUT 1
TYPE='IND'   STATION='MIDWAY'        PARAM='TEMP'
TYPE='IND'  STATION='MIDWAY'        PARAM='WIND_VEL';
TYPE='DEP'   STATION='U_OF_C'        PARAM='SO2_S2';
END
STOP
```

This example searches the output file for the array titled "SAMPLE INPUT 1" produced by a previous search run. The parameter SO_2 emission data from stack two (SO2-S2) for the University of Chicago heating plant is selected as the dependent variable, and the parameters temperature (TEMP) and wind speed (WIND-VEL) at Midway Airport are selected as the independent variables. The regression analysis results for this set of data are also shown.

Sample Output

REGRESSION ANALYSIS RESULTS

STATION	PARAMETER	VAR NO	REG COEF	S.D. REG COEF
MIDWAY	TEMP	1	-0.1479165E 01	0.7283614E 00
MIDWAY	WIND VEL	3	9.8626674E 01	0.4927691E 01
DEPENDENT VARIABLE				
U_OF_C	SO2_S2	5	0.3251589E 03	0.2674812E 02

MULTIPLE CORRELATION COEFFICIENT= 0.7666295E 00

SUM OF SQUARES ATTRIBUTABLE TO REGRESSION(SSAR)= 0.8159359E 04

DEGREES OF FREEDOM ASSOCIATED WITH SSAR= 0.2000000E 01

MEAN SQUARE OF SSAR= 0.4079680E 04

SUM OF SQUARES OF DEVIATIONS FROM REGRESSION(SSDR)= 0.5723699E 04

DEGREES OF FREEDOM ASSOCIATED WITH SSDR= 0.8000000E 01

MEAN SQUARE OF SSDR= 0.7154624E 03

F-VALUE= 0.5702158E 01

Program listings, including the source code for subroutine POINTER, are given in Appendix H. POINTER is used for dynamic storage allocation as described in Reference 16. For sample deck structures see Appendices A and B.

8.2 Stepwise Multiple Discriminant Analysis

Given one output file of data selected by the master search program, a set of subroutines has been written to perform a stepwise multiple discriminant analysis on complete data, allowing up to 50 variables, 20 groups and 1000 observations per group. This set of subroutines is to be used with the APICS program FORTRN and a user constructed subroutine SETUP described later. (Fig. 8.2)

The stepwise multiple discriminant analysis procedure is an IBM360 adaptation of Northwestern University's CDC 6400 program EIDISC. For a detailed explanation of the statistical theory, input and output information see Reference (20).

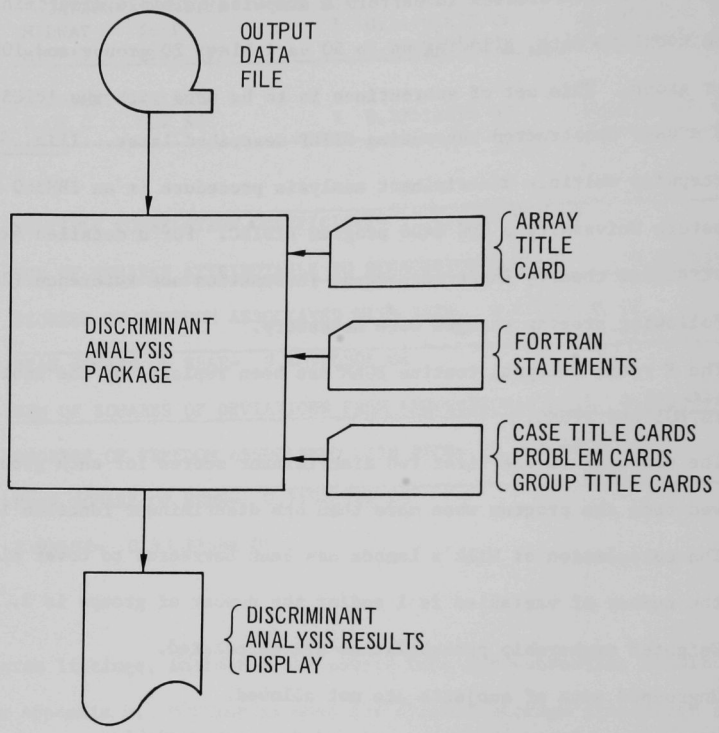
The following program changes were necessary.

1. The F ratio integral routine BGAM has been replaced by the equivalent routines INTEG1 and FFUNC.
2. The plotting of the first two discriminant scores for each group has been removed from the program when more than one discriminant function is used.
3. The calculation of Wilk's Lambda has been corrected to cover all cases in which the number of variables is 1 and/or the number of groups is 2.
4. Weighted membership probabilities are calculated.
5. Ungrouped sets of subjects are not allowed.
6. STATION and PARAM names are printed for each variable.
7. Classification matrices are calculated from weighted and unweighted membership probabilities.
8. Input to the program has been simplified.

INPUT INFORMATION

A. One input card which specifies a title present on the output file produced by the APICS SEARCH program followed by a card with STOP in first four columns. (DDNAME=SYSIN)

B. A set of input cards for each case. This set consists of a case identification card, a problem card, and group identification cards. (DDNAME=ET05F001)



113-1131

Fig. 8.2 Discriminant Analysis

The format for these cards is as follows.

Case Identification Card - 80 columns of descriptive information.

Problem Card

Columns	Name	Contents
1-6		Blank
7	OPT2	1 - if plot of discriminant function scores desired
		0 - if not
8		Blank
9	L(7)	1 - if total matrices are to be printed
		0 - if not
10	L(8)	1 - if group matrices are to be printed
		0 - if not
11-14	ENTER	Chi-square value to limit variables entering analysis
15-17	TROOT	Percent of total variance to be explained by discriminant functions
18-80		Blank

Group Identification Cards - 80 columns of descriptive information (one card for each group).

C. A set of FORTRAN statements to be inserted into the subroutine SETUP. For each discriminant case, these statements define the number of groups to be analyzed, the number of variables within each group, the variable names (STATION and PARAM), and the appropriate subset of the search output data to be analyzed. (The storage for this data subset is dynamically allocated into fast core storage using subroutines described in Reference (19). The source code for this routine is included with the listings of the program given in Appendix H.) The subroutine SETUP then calls the routine PRINT which prints this information for each case. The structure of the SETUP routine is indicated in the following listing.

SKELETON SETUP Routine

```
      SUBROUTINE SETUP(NPAR,NDATA,X,STANAM,PARNAM)
      REAL*8 VNAME,STANAM,PARNAM
      COMMON/B/ NOBS(20),NVAR,NG
      COMMON/NAMES/ VNAME(2 ,50)
      DIMENSION X(NDATA,NPAR)
      DIMENSION STANAM(1),PARNAM(1)
C
C        FOR EACH CASE SETUP
C              NVAR - NUMBER OF VARIABLES
C              NG - NUMBER OF GROUPS
C        STOP AT END OF ALL CASES
C
C
C        INSERT FORTRAN STATEMENTS HERE
      RETURN
      ENTRY DATA(XX,ISIZE)
      DIMENSION XX(NG,NVAR,1000)
C
C        FOR EACH CASE SETUP THE FOLLOWING FOR ALL I,J,AND K
C              VNAME(1,I) - STATION NAME FOR I-TH VARIABLE
C              VNAME(2,I) - PARAM NAME FOR I-TH VARIABLE
C              NOBS(J) - NUMBER OF OBSERVATIONS IN J-TH GROUP
C                        (MUST NOT BE ZERO)
C              XX(J,I,K) - K-TH DATA POINT FORI-TH VARIABLE
C                          IN J-TH GROUP
C
C        INSERT FORTRAN STATEMENTS HERE
      CALL PRINT (XX)
      CALL EIDISC(XX)
   20 CONTINUE
      RETURN
      END
```

OUTPUT INFORMATION:

1. For each group: means, standard deviations, and correlation matrix.
2. For each group: dispersion matrix, raw- and deviation-sum-of-squares-and-cross-products matrices (optional).
3. Total sample: within-groups, between-groups, and total-sums-of-squares and cross-products matrices. Labeled W, A, and T matrices respectively. (optional)
4. Total sample: means, standard deviations, correlation matrix, and total sample size.
5. Univariated F-ratios for each variable. (Standard one-way analysis of variance Fs.)
6. Table of the generalized distance statistic (Rao's V) from the stepwise procedure.
7. Eigenvalues (roots) of $W^{-1}*A$, trace of roots, and roots as percents of trace.
8. Eigenvectors of $W^{-1}*A$ (discriminant functions) associated with above roots.
9. Scaled vectors to show the relative contribution of the selected variables to the functions.
10. Wilk's Lambda and F test for the hypothesis that the group means are the same over the groups in question and the variables selected.
11. Centroids of the groups in the discriminant space.
12. Dispersion matrix for each group in the discriminant space.
13. Chi-squares and group membership probabilities for the group Centroids.
14. Discriminant scores, Chi-squares and group membership probabilities for each subject.
15. Classification matrix of correct and incorrect group memberships.
16. Plot of discriminant function scores for each group (optional).

RUNNING INSTRUCTIONS:

The discriminant analysis code calls for two scratch data sets, thus DD cards for FT01F001 and FT04001 must be included with the deck.

SAMPLE PROBLEM

The following example of a SETUP routine divides the data into two groups based on the wind direction, which must be the second parameter in the output array from a search. Two sets of data are set up by incorporating the first five parameter values from the search into the XX array as appropriate. That is data points are placed in group 1 if the wind direction is less than 266°, and group 2 otherwise. The number of observations falling into each group is determined during the process. This information is then passed on to the EIDISC package for statistical analyses. Sample input for the search and for the multiple discriminant analysis program is also provided. These input specifications are appropriate to use with the following SETUP routine.

Sample SETUP Routine

```
      SUBROUTINE SETUP(NPAR,NDATA,X,STANAM,PARNAM)
      REAL*8 VNAME,STANAM,PARNAM
      COMMON/B/ NOBS(20),NVAR,NG
      COMMON/NAMES/ VNAME(2 ,50)
      DIMENSION X(NDATA,NPAR)
      DIMENSION STANAM(1),PARNAM(1)
C
C        FOR EACH CASE SETUP
C             NVAR - NUMBER OF VARIABLES
C             NG - NUMBER OF GROUPS
C        STOP AT END OF ALL CASES
C
C
C        INSERT FORTRAN STATEMENTS HERE
      DATA IND1,IND2/0,0/
      IF ( IND1 .EQ. 1) STOP
      NVAR=5
      NG=2
      IND1=IND1+1
C
C
      RETURN
      ENTRY DATA(XX,ISIZE)
      DIMENSION XX(NG,NVAR,1000)
C
C        FOR EACH CASE SETUP THE FOLLOWING FOR ALL I,J,AND K
C             VNAME(1,I) - STATION NAME FOR I-TH VARIABLE
C             VNAME(2,I) - PARAM NAME FOR I-TH VARIABLE
C             NOBS(J) - NUMBER OF OBSERVATIONS IN J-TH GROUP
C                       (MUST NOT BE ZERO)
C             XX(J,I,K) - K-TH DATA POINT FORI-TH VARIABLE
C                         IN J-TH GROUP
C
C        INSERT FORTRAN STATEMENTS HERE
      NOBS(1)=0
      NOBS(2)=0
      DO 10 I=1,NDATA
      IF ( X(I,2) .LT. 266. ) III=1
      IF ( X(I,2) .GE. 266. ) III=2
      MMM=NOBS(III)+1
      DO 5 J=1,5
      VNAME(1,J)=STANAM(J)
      VNAME(2,J)=PARNAM(J)
    5 XX(III,J,MMM)=X(I,J)
      NOBS(III)=MMM
      XX(III,2,MMM)=X(I,4)-8.
   10 VNAME(1,2)=STANAM(4)
      IF (NOBS(1) .EQ. 0  .OR.NOBS(2).EQ.0) GO TO 20
C
C
      CALL PRINT (XX)
      CALL EIDISC(XX)
   20 CONTINUE
      RETURN
      END
```

Sample Search Input

```
SO2 AT TAM_4 L.E. .30
PRINT='NO';
DATE_LIMITS=1;
'66/01/01','66/02/15'
HOUR_LIMITS=0;
MET_STATIONS=1;
STATION='TAM_4'  PARAMS=1;
PARAM='SO2'   BANDS=1; 0.,.3
OUT_STATIONS=5;
STATION='TURNER'  PARAMS=1;
PARAM='TURNER_C';
STATION='TAM_4' PARAMS=2;
PARAM='WIND_DIR';
PARAM='WIND_VEL';
STATION='MIDWAY'  PARAMS=4;
PARAM='TEMP';
PARAM='REL_HUM';
PARAM='WIND_DIR';
PARAM='CEIL';
STATION='NORCO'  PARAMS=1;
PARAM='MIX_HT';
STATION='CALENDAR'  PARAMS=1;
PARAM='HOUR';
END
STOP
```

Sample Input

```
//GO.SYSIN DD *
SO2 AT TAM_4 L.E. .30
STOP
//GO.MYSIS DD *
WIND DIRECTION TEST CASE ****************************************
        11.01698.
WIND DIRECTION LESS THAN 226 ************************************
WIND DIRECTION GREATER THAN OR EQUAL TO 226 *********************
/*
```

The above input causes statistical analyses to be performed based on the following five parameters:

Turner stability

Wind direction

Wind velocity

Temperature

Relative humidity

Only data accompanying SO_2 levels below .3 ppm are used and the data are divided into 2 groups about a wind direction of 266°. The following set of output includes sample printouts from the search program, the retrieval program, and the stepwise multiple discriminant analysis program.

STEPMDA: Output of Specifications from Search Program

```
INPUT DATA
SO2 AT TAM_4 L.E. .30
PRINT='NO ';
LOWER_DATE(1)='66/01/01'                                UPPER_DATE(1)='66/02/15';
LOWER_HOUR(1)=        0 UPPER_HOUR(1)=          23;
MET STATION REQUESTS
STATION='TAM_4   ';
PARAM='SO2     ';
BANDS                              0.000                          0.300
CUT STATION REQUESTS
STATION='TURNER  ';
PARAM='TURNER_C'          DELAY=    0;
STATION='TAM_4   ';
PARAM='WIND_DIR'          DELAY=    0;
PARAM='WIND_VOL'          DELAY=    0;
STATION='MIDWAY  ';
PARAM='TEMP    '          DELAY=    0;
PARAM='REL_HUM '          DELAY=    0;
PARAM='WIND_DIR'          DELAY=    0;
PARAM='CEIL    '          DELAY=    0;
STATION='NORCO   ';
PARAM='MIX_HT  '          DELAY=    0;
STATION='CALENDAR';
PARAM='HOUR    '          DELAY=    0;
DATA STORAGE HAS BEEN ALLOCATED
```

STEPMDA: Sample Output from Search Retrieval Program

```
TAPE SEARCH BEGUN FOR LABEL:
OUTPUT_HEADER.LABEL='SO2 AT TAM_4 L.E. .30
OUTPUT_HEADER.STATIONS=                5
OUTPUT_HEADER.NUMBER_OBS=              552;
ARRAY LABEL FOUND
OUTPUT_STATION.NAME(1)='TURNER   '
OUTPUT_STATION.NAME(2)='TAM_4    '
OUTPUT_STATION.NAME(3)='MIDWAY   '
OUTPUT_STATION.NAME(4)='NORCO    '
OUTPUT_STATION.NAME(5)='CALENDAR'
OUTPUT_PARAM.STATION(1)='TURNER   '
OUTPUT_PARAM.NAME(2)='WIND_DIR'
OUTPUT_PARAM.STATION(4)='MIDWAY   '
OUTPUT_PARAM.NAME(5)='REL_HUM '
OUTPUT_PARAM.STATION(7)='MIDWAY   '
OUTPUT_PARAM.NAME(8)='MIX_HT   '
```

```
SO2 AT TAM_4 L.E. .30

OUTPUT_HEADER.NUMBER_PARAMS=                9

OUTPUT_STATION.POINTER(1)=          0
OUTPUT_STATION.POINTER(2)=          1
OUTPUT_STATION.POINTER(3)=          3
OUTPUT_STATION.POINTER(4)=          7
OUTPUT_STATION.POINTER(5)=          8
OUTPUT_PARAM.NAME(1)='TURNER_C'
OUTPUT_PARAM.STATION(3)='TAM_4    '
OUTPUT_PARAM.NAME(4)='TEMP     '
OUTPUT_PARAM.STATION(6)='MIDWAY   '
OUTPUT_PARAM.NAME(7)='CEIL     '
OUTPUT_PARAM.STATION(9)='CALENDAR'
```

```
OUTPUT_STATION.PARAMS(1)=           1
OUTPUT_STATION.PARAMS(2)=           2
OUTPUT_STATION.PARAMS(3)=           4
OUTPUT_STATION.PARAMS(4)=           1
OUTPUT_STATION.PARAMS(5)=           1;
OUTPUT_PARAM.STATION(2)='TAM_4    '
OUTPUT_PARAM.NAME(3)='WIND_VEL'
OUTPUT_PARAM.STATION(5)='MIDWAY   '
OUTPUT_PARAM.NAME(6)='WIND_DIR'
OUTPUT_PARAM.STATION(8)='NORCO    '
OUTPUT_PARAM.NAME(9)='HOUR     ';
```

STEPMDA: Sample Output

```
PERCENT OF TRACE= 100.006 USING   1 DISCRIMINANT FUNCTIONS.

CENTROIDS AND DISPERSIONS IN REDUCED SPACE

NO. OF GROUPS =  2

NO. OF TEST VARIABLES =  5

NO. OF LINEAR COMBINATIONS =  1

CENTROIDS OF GROUPS IN REDUCED SPACE, ROW-WISE

GROUP  1-1.42856121E 00

GROUP  2-3.52720928E 00

DISPERSION IN REDUCED SPACE FOR GROUP  1
3.96259689E 00
       0.3962597E 01

DISPERSION IN REDUCED SPACE FOR GROUP  2
8.39111519E 00
       0.8391115E 01

CHI SQUARES AND GROUP MEMBERSHIP PROBABILITIES FOR GROUP  1 CENTROID
2                                  0.0          0.5249
3        1          0.8896        0.8896       0.1104
4        1          0.6542        0.6542       0.3458

CHI SQUARES AND GROUP MEMBERSHIP PROBABILITIES FOR GROUP  2 CENTROID
2                                  1.1115       0.0
3        1          0.7804        0.7804       0.2196
4        2          0.5450        0.4550       0.5450
```

Sample Output (Cont.)

```
WIND DIRECTION TEST CASE ***************************************************

MULTIPLE DISCRIMINANT ANALYSIS

NO. OF GROUPS =  2

NO. OF VARIABLES =  5


F RATIOS FOR FOR INDIVIDUAL VARIABLES


NUMBER OF DEGREES OF FREEDOM ARE   1. AND  550.

AMONG MEAN SQR     WITHIN MEAN SQR     F RATIO     VARIABLE
      1.193              0.400            2.98        1  TURNER   -TURNER_C
   3653.922            111.162           32.87        2  MIDWAY   -WIND_DIR
    364.824             14.269           25.57        3  TAM_4    -WIND_VEL
   3654.047            111.162           32.87        4  MIDWAY   -TEMP
   1251.750            180.711            6.93        5  MIDWAY   -REL_HUM


              VARIABLE  RAOS V    V ADDED     SIGNIFICANCE OF
              ENTERED   FOR TOTAL  BY VAR.  TOTAL SET  ENTERED VARIABLE
                         SET       ENTERED              AS ADDED ALONE
 4 MIDWAY  -TEMP         32.87     32.87  2.3395E-03 2.34E-03 9.98E-01
 3 TAM_4   -WIND_VEL     67.38     34.51 -6.2943E-05 2.34E-03 9.98E-01
 1 TURNER  -TURNER_C     71.61      4.23 -2.8610E-06 4.41E-02 9.13E-01

VARIABLES ENTERED TO HERE WILL BE USED FOR ANALYSIS
 5 MIDWAY  -REL_HUM      46.71    -24.90 -2.8610E-06 4.41E-02 9.13E-01
```

Sample Output (Concluded)

CLASSIFICATION MATRIX 1

GROUP	1	2
1	441	6
2	85	20

CLASSIFICATION MATRIX 2

GROUP	1	2
1	355	92
2	42	63

8.3 Tabulation Prediction Program

A tabulation prediction technique has been developed for use with the APICS system. It incorporates specified wind bands and other atmospheric phenomena that were determined to be most influential in establishing correlation between meteorological measurements and air quality measurements.

The program generates tables of predicted air pollution levels for the whole range of values of pertinent atmospheric phenomena based on historical experience. That is: actual SO_2 levels are compared with weather observations (possibly with delays up to 24 hours). The search and data retrieval features of the APICS system are used in accessing data on the master tape for statistical analyses.

The program is written as a FORT subroutine called by the retrieval program FORTRN or directly by SEARCH2.

A listing of the program is given in Appendix H and a complete description of the principles behind the development of the code and its use is provided in Section 9. Samples of tables of expected SO_2 values vs. the weather phenomena are also included. The program may be used as is, or the routines may be modified to generate tables to apply to other localities.

The following input specifications were used with the program at Argonne. The search input must be exactly as indicated below (except for date limits) while the ranges given in the additional input may be modified somewhat. (See Section 9.)

SAMPLE SEARCH INPUT

```
SPRING DATA (MARCH,APRIL,MAY)  FOR TABULATION PREDICTION TECHNIQUE
PRINT='NO';
DATE_LIMITS=2;  '66/03/01','66/05/31'  '67/03/01','67/05/31'
HOUR_LIMITS=0;
MET_STATIONS=0;
OUT_STATIONS=10;
  STATION='TAM_1'  PARAMS=3;
    PARAM='SO2'   DELAY=0;
PARAM='WIND_VEL'  DELAY=0;
    PARAM='WIND_DIR'  DELAY=0;
  STATION='TAM_2'  PARAMS=3;
    PARAM='SO2'   DELAY=0;
PARAM='WIND_VEL'  DELAY=0;
    PARAM='WIND_DIR'  DELAY=0;
  STATION='TAM_3'  PARAMS=3;
    PARAM='SO2'   DELAY=0;
PARAM='WIND_VEL'  DELAY=0;
    PARAM='WIND_DIR'  DELAY=0;
  STATION='TAM_4'  PARAMS=3;
PARAM='SO2      '  DELAY=0;
PARAM='WIND_VEL'  DELAY=0;
PARAM='WIND_DIR'  DELAY=0;
  STATION='TAM_5'  PARAMS=3;
    PARAM='SO2'   DELAY=0;
PARAM='WIND_VEL'  DELAY=0;
    PARAM='WIND_DIR'  DELAY=0;
  STATION='TAM_6'  PARAMS=3;
    PARAM='SO2'   DELAY=0;
PARAM='WIND_VEL'  DELAY=0;
    PARAM='WIND_DIR'  DELAY=0;
  STATION='TAM_7'  PARAMS=3;
    PARAM='SO2'   DELAY=0;
PARAM='WIND_VEL'  DELAY=0;
    PARAM='WIND_DIR'  DELAY=0;
  STATION='TAM_8'  PARAMS=3;
    PARAM='SO2'   DELAY=0;
PARAM='WIND_VEL'  DELAY=0;
    PARAM='WIND_DIR'  DELAY=0;
STATION='CALENDAR'   PARAMS=4;
PARAM='HOUR    ';
PARAM='DAY_WEEK';
PARAM='MONTH   ';
PARAM='YEAR    ';
STATION='MIDWAY'   PARAMS=7;
PARAM='CEIL'  DELAY=0;
PARAM='TEMP' DELAY=0;
PARAM='LIQ_PCP1'  DELAY=0;
PARAM='LIQ_PCP2'  DELAY=0;
PARAM='FRO_PCP1'  DELAY=0;
PARAM='FRO_PCP2'  DELAY=0;
PARAM='FRO_PCP3'  DELAY=0;
END
STOP
/*
```

Sample Input:

```
//GO.SYSIN     DD     *
SPRING DATA (MARCH,APRIL,MAY) FOR TABULATION PREDICTION TECHNIQUE
STOP
//GO.FT05F001     DD     *
ASON:  MARCH,APRIL,MAY                PRECIPITATION:  NO    JAN 1,1966 - DEC 31,1967
  8      4      6     12
  0   1000   2500   7000  1000000
  0      4      8     13       19      25      99
-20    -10      0     10       20      30      40      50      60      70
 80     90    100
  4      4      4      4        4       4       4       4
  8      8      8      8        8       8       8       8
  0
```

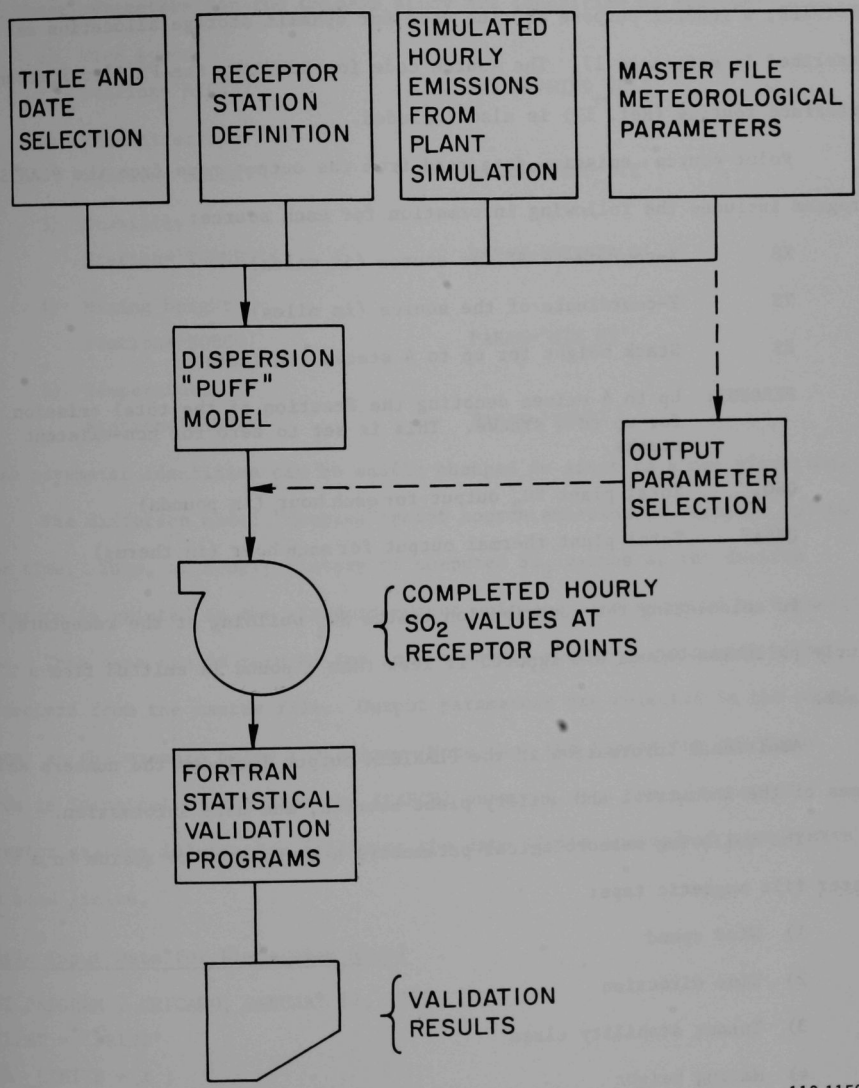
Note that the "0" on the last card indicates that data points with precipitation present are to be discarded; and the title card serves only to identify the tabulated output. The second card specifies 8 TAM stations and the number of ranges of ceiling(4), wind speed(6), and temperature(12) given on the next 3 cards, while the 6th and 7th card specify the number of built-in hour(4) and wind direction(8) bands that are to be used.

8.4 Atmospheric Diffusion Calculations

The estimation of pollution levels via a multiple source atmospheric dispersion model is essential to the design of episodal and long range emission control strategies. PUFF, an atmospheric diffusion model (4,7) has been programmed to operate within the APICS system. Meteorological and source emission data are available as constant hourly values from the master file and from the output from PLANTSIM (see Section 3.7). Dates, receptor points, and output parameters are specified in the input. An input-output diagram of the model operation is shown in Figure 8.3.

The present structure of the program is designed to accommodate the integrated puff dispersion code (Roberts, et al, 1969) which is particularly well suited to describing transient conditions. However, since this model is presently under evaluation by NAPCA, a simpler standard PLUME model based on a steady state gaussian distribution has been substituted primarily to demonstrate the means by which an atmospheric dispersion model is linked to the APICS system. This standard is specified in the Workbook on Atmospheric Dispersion Estimates (Turner, 1968).

The ground level SO_2 concentrations (PPM) at up to fifty dose points (specified as RECEPTORS in the input) are calculated as the sum of contributions from each point source in the output file from Program PLANTSIM. The X,Y, coordinates of the receptor points must be in the same system as was used to specify source locations for the PLANTSIM program. Furthermore, to be consistent with the PUFF program as presently coded, X must increase going West and Y must increase going South. The names specified for the receptors in the input are arbitrary but the fifth character in each should be the (same) number of the TAM station whose aerovane information is to be used in the calculations.



113-1150

Fig. 8.3 Input-Output Diagram of Dispersion Model Program

The listing of the program in Appendix H includes the source code for BPOINTER, a general purpose routine used for dynamic storage allocation as described in Reference 17. The source code for FORTRAN, the PL/I to Fortran interface routine (Ref. 12) is also included.

Point source emission data read from the output tape from the PLANTSIM program includes the following information for each source:

XS	X-coordinate of the source (in miles)
YS	Y-coordinate of the source (in miles)
ZS	Stack height for up to 4 stacks (in feet)
PERCENT:	Up to 4 values denoting the fraction of the total emission for up to 4 stacks. This is set to zero for non-existent stacks.
QSO_2	Total plant SO_2 output for each hour (in pounds)
QHEAT	Total plant thermal output for each hour (in therms)

In calculating the contribution to the SO_2 building at the receptors, hourly pollution values are ignored if less than 1 pound is emitted from a stack.

Additional information in the PLANTSIM output includes the numbers and names of the industrial and utility plant sources, and date information.

The following meteorological parameters are expected to reside on a master file magnetic tape:

1) Wind speed
2) Wind direction
3) Turner stability class
4) Mixing height
5) Temperature

Reference to the listing of the model program in the appendix will show that these parameters for the Chicago study are identified as follows:

1) Wind speed*
   Station='MIDWAY' PARAM='WIND_VEL'

2) Wind direction*
   Station='MIDWAY' PARAM='WIND_DIR'

3) Stability
   Station='TURNER' PARAM='TURNER_C'

4) Mixing height
   Station='NORCO' PARAM='MIX_HT'

5) Temperature
   Station='MIDWAY' PARAM='TEMP'

These parameter identities can be easily changed to adapt to a new situation.

The diffusion model "couples" point source emission to receptor points over time. Thus, an hourly history of computed SO_2 values at the desired receptors is printed on the standard output unit and also recorded on magnetic tape. This tape can in turn be fed into analysis routines along with other parameters from the master file. Output parameters are selected in the usual manner as the example below indicates. Note that the format for the input cards is identical to that for the SEARCH2 program, with the addition of receptor station information following the date limits instead of hour limits and band limits.

Sample Input Data for Dispersion Model

```
TEST PRØGRAM . CHICAGO, JANUARY 12, 1967
OPTLIST = 'PRINT'
DATE _LIMITS = 1 ;
'67/01/12' , '67/01/12'
RECEPTØRS = 2;
RECEPTOR = 'LC_T5'    X = 5.0    Y = -3.0    Z = 0.0    ;
RECEPTOR = 'JR_T5'    X = 7.0    Y =  3.0    Z = 0.0    ;
```

---

*The [illegible] and wind direction must also be available for 8 TAM stations wit[illegible]_n'.

Sample Input Data for Dispersion Model (Continued)

```
OUT_STATIONS = 4;
STATION='MIDWAY'        PARAMS=3;
PARAM='TEMP';
PARAM='WIND_DIR';
PARAM='WIND_VEL';
STATION='TURNER'        PARAMS=1;
PARAM='TURNER_C';
STATIØN='NORCO'         PARAMS=1;
PARAM='MIX_HT';
STATION='TAM_5'         PARAMS=3;
PARAM='WIND_VEL';
PARAM='WIND_DIR';
PARAM='SO2';
END
STOP
```

The receptor card (e.g., RECEPTOR=TAM_5) specified which aerovane is used to determine the wind field (assumed uniform over the region). The OUT station options do not influence the program execution. See sample output for this example which follows.

Programming Notes:

The PUFF model program is a collection of FORTRAN SUBROUTINES linked to APDATA by SUBROUTINE IPLUME and in particular by entry PLUMES (IR,TEMP,WSA,WDA, STAB,HLID,NSRC,XS,YS,ZS,PERCENT,QSØ2,QHEAT,NREC,XR,YR,ZR,RECDAT,MM,KTAM,SØBS) . The elements in the calling sequence are defined as follows:

SAMPLE OUTPUT: TEST PROGRAM PLUMES

	MIDWAY TEMP	MIDWAY WIND_DIR	MIDWAY WIND_VEL	TURNER TURNER_C	NORCO MIX_HT
67/01/12 0	31.000	210.000	16.000	4.000	1259.000
67/01/12 1	31.000	200.000	15.000	4.000	1222.000
67/01/12 2	32.000	200.000	19.000	4.000	1355.000
67/01/12 3	33.000	210.000	18.000	4.000	1530.000
67/01/12 4	33.000	210.000	20.000	4.000	1541.000
67/01/12 5	32.000	210.000	15.000	4.000	1289.000
67/01/12 6	32.000	240.000	12.000	4.000	1233.000
67/01/12 7	32.000	230.000	13.000	4.000	1061.000
67/01/12 8	32.000	230.000	12.000	4.000	911.000
67/01/12 9	33.000	240.000	13.000	4.000	1091.000
67/01/12 10	35.000	250.000	12.000	4.000	1664.000
67/01/12 11	36.000	270.000	13.000	4.000	1832.000
67/01/12 12	37.000	250.000	10.000	4.000	1991.000
67/01/12 13	38.000	290.000	6.000	4.000	2149.000
67/01/12 14	38.000	240.000	9.000	3.000	2030.000
67/01/12 15	38.000	240.000	12.000	4.000	1915.000
67/01/12 16	38.000	260.000	8.000	5.000	1810.000
67/01/12 17	35.000	240.000	5.000	5.000	538.000
67/01/12 18	36.000	240.000	6.000	5.000	740.000
67/01/12 19	35.000	220.000	7.000	4.000	430.000
67/01/12 20	35.000	240.000	4.000	5.000	406.000
67/01/12 21	35.000	220.000	6.000	5.000	385.000
67/01/12 22	35.000	200.000	7.000	4.000	366.000
67/01/12 23	35.000	260.000	6.000	5.000	348.000

	TAM_5 WIND_VEL	TAM_5 WIND_DIR	TAM_5 SO2	LC_T5 SO2	JR_T5 SO2
67/01/12 0	17.800	201.000	0.020	0.043	0.035
67/01/12 1	17.100	210.000	0.020	0.052	0.003
67/01/12 2	18.100	208.000	0.020	0.038	0.009
67/01/12 3	15.700	212.000	0.020	0.075	0.001
67/01/12 4	15.900	213.000	0.020	0.097	0.000
67/01/12 5	13.500	215.000	0.020	0.144	0.000
67/01/12 6	12.500	224.000	0.060	0.252	0.000
67/01/12 7	11.500	228.000	0.090	0.380	0.001
67/01/12 8	11.600	234.000	0.070	0.065	0.037
67/01/12 9	10.900	247.000	0.090	0.019	0.118
67/01/12 10	11.800	253.000	0.080	0.016	0.429
67/01/12 11	10.200	255.000	0.070	0.023	0.613
67/01/12 12	9.300	278.000	0.270	0.025	0.005
67/01/12 13	7.600	272.000	0.280	0.021	0.003
67/01/12 14	8.100	254.000	0.090	0.015	0.589
67/01/12 15	7.900	251.000	0.060	0.016	0.234
67/01/12 16	8.600	250.000	0.060	0.010	0.069
67/01/12 17	6.400	241.000	0.090	0.072	0.011
67/01/12 18	5.300	246.000	0.080	0.066	0.038
67/01/12 19	5.400	249.000	0.090	0.031	0.367
67/01/12 20	4.700	239.000	0.110	0.072	0.022
67/01/12 21	6.300	233.000	0.070	0.068	0.008
67/01/12 22	6.300	242.000	0.050	0.095	0.064
67/01/12 23	5.000	251.000	0.070	0.018	0.409

Variables in Entry PLUMES:

Variable	Description	Units
IR	Hour of day (1→24 for times 0000 to 2300)	-
TEMP	Temperature for hour IR	°F
WSA	Wind speed (array) for aerovane station specified by input receptor cards	MPH
WDA	Wind direction (array) for aerovane stations specified by input receptor cards	DEGREES clockwise with 0 as North
STAB	Stability index for hour IR	-
HLID	Mixing layer height for hour IR	FT
NSRC	Number of point sources in PLANTSIM tape	-
XS,YS,ZS	Coordinates and physical stack heights of sources (up to 4 stacks for each source)	MI,MI,FT
PERCENT	Array indicating fraction of total source output allocated to each stack	-
QSØ2	Plant SO_2 output	LB/HR
QHEAT	Plant thermal output	THERM/HR
NREC	Number of receptor concentrations to be calculated	-
XR,YR,ZR	Coordinates of receptors with ZR (height) = 0	MI,MI,-
RECDAT	Array of receptor concentrations for transmittal to APDATA	PPM
MM	Number of consecutive hours for which concentrations have been calculated (for use with integrated puff model)	(not used) -
KTAM	Array indicating the choice of aerovane stations on output cards (0 or 1)	(not used) -
SØBS	Actual measured SO_2 concentrations at aerovane stations selected on output cards (for statistical comparison with calculations)	(not used) PPM

APICS USER MANUAL

9.0 The Tabulation Technique for Forecasting Concentrations of Air Pollution

9.0 The Tabulation Technique for Forecasting Concentrations of Air Pollutants

When material is injected into the atmosphere, whether it be from a distributed area source or a point source, the meteorological conditions determine the rates of dispersion and consequently the concentrations observed downwind. This section of the manual describes a method of using the APICS system in order to correlate historical meteorology and air quality data in such a way that a short-range (24 hour) receptor-oriented air quality forecasting system can be produced. The APICS system is required to create a set of prediction tables, but once they are available, real-time access to a computer is not required in order to make an air quality prediction. The system may be updated as often as changes in source emissions or distribution indicate the need.

9.1 Description of the Tabulation Prediction Technique

The relation between meteorological variables and pollutant concentrations may be displayed by arranging combinations of significant meteorological variables in an ordered sequence and presenting the associated probability distribution for each entry, as shown in Fig. 9.1. This procedure is referred to as the "Tabulation Prediction Technique" or the "Look-up Technique" and is based on hourly readings. For each combination of meteorological variables the minimum value, 10, 25, 50, 75, 90, 95, 98, 99, percentiles and maximum value of SO_2 concentrations are shown. Also presented are the interquartile range i.e., the difference in SO_2 concentration between the 75th and 25th percentiles, and that of the 95th and 75th percentiles. The number of cases observed for each combination of meteorological variables is shown in the last column. To arrange combinations of meteorological variables into an ordered sequence, each of the selected variables must first be divided into appropriate bands. For example, wind direction might be grouped into three classes, time of day also into three, wind speed categories into five and so forth. Since the combinations of

SEASON: MARCH,APRIL,MAY PRECIPITATION: NO STATION: TAM 4 PERIOD OF DATA: JAN 1,1966 - DEC 31,1967

WIND DIR DEGREES	CEILING FEET	WIND SP KT	TEMP DEG F	HOUR CST	PERCENTILE VALUES OF SO2(PPM) CONCENTRATIONS MIN	25	50	75	90	95	98	99	MAX	75-25	95-75	MEAN	ST DEV	FREQ
0-140	7000-****	0- 3	70- 79	0- 3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0-140	7000-****	0- 3	70- 79	4-15	0.01	0.01	0.03	0.04	0.06	0.06	0.06	0.06	0.06	0.03	0.02	0.03	0.0183	6
0-140	7000-****	0- 3	70- 79	16-23	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.0	0.0	0.01	0.0	1
0-140	7000-****	0- 3	80- 89	0- 3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0-140	7000-****	0- 3	80- 89	4-15	0.02	0.02	0.02	0.03	0.03	0.03	0.03	0.03	0.03	0.01	0.0	0.02	0.0050	2
0-140	7000-****	0- 3	80- 89	16-23	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.0	0.0	0.02	0.0	1
0-140	7000-****	4- 7	10- 19	0- 3	0.01	0.01	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.01	0.0	0.02	0.0047	3
0-140	7000-****	4- 7	10- 19	4-15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0-140	7000-****	4- 7	10- 19	16-23	0.08	0.08	0.10	0.12	0.12	0.12	0.12	0.12	0.12	0.04	0.0	0.10	0.0200	2
0-140	7000-****	4- 7	20- 29	0- 3	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.0	0.0	0.02	0.0	1
0-140	7000-****	4- 7	20- 29	4-15	0.09	0.10	0.15	0.20	0.21	0.21	0.21	0.21	0.21	0.10	0.01	0.15	0.0460	4
0-140	7000-****	4- 7	20- 29	16-23	0.05	0.05	0.07	0.10	0.27	0.27	0.27	0.27	0.27	0.05	0.17	0.12	0.0864	4
0-140	7000-****	4- 7	30- 39	0- 3	0.01	0.01	0.03	0.04	0.05	0.06	0.08	0.08	0.08	0.03	0.02	0.02	0.0175	17
0-140	7000-****	4- 7	30- 39	4-15	0.01	0.02	0.04	0.09	0.13	0.15	0.28	0.29	0.30	0.07	0.06	0.06	0.0650	24
0-140	7000-****	4- 7	30- 39	16-23	0.01	0.02	0.04	0.09	0.13	0.15	0.27	0.27	0.27	0.06	0.06	0.06	0.0652	41
0-140	7000-****	4- 7	40- 49	0- 3	0.01	0.01	0.03	0.04	0.05	0.07	0.09	0.09	0.10	0.03	0.03	0.02	0.0249	25
0-140	7000-****	4- 7	40- 49	4-15	0.01	0.01	0.03	0.04	0.05	0.08	0.09	0.10	0.14	0.03	0.03	0.02	0.0233	96
0-140	7000-****	4- 7	40- 49	16-23	0.01	0.01	0.03	0.04	0.05	0.09	0.12	0.13	0.13	0.03	0.05	0.03	0.0253	59
0-140	7000-****	4- 7	50- 59	0- 3	0.01	0.01	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.01	0.0	0.01	0.0043	8
0-140	7000-****	4- 7	50- 59	4-15	0.01	0.01	0.03	0.04	0.05	0.05	0.09	0.17	0.20	0.03	0.01	0.02	0.0246	127
0-140	7000-****	4- 7	50- 59	16-23	0.01	0.02	0.03	0.05	0.09	0.13	0.25	0.25	0.25	0.03	0.08	0.04	0.0467	36
0-140	7000-****	4- 7	60- 69	0- 3	0.04	0.04	0.05	0.07	0.09	0.09	0.09	0.09	0.09	0.03	0.02	0.06	0.0250	2
0-140	7000-****	4- 7	60- 69	4-15	0.01	0.01	0.03	0.04	0.05	0.06	0.06	0.06	0.06	0.03	0.02	0.02	0.0172	59
0-140	7000-****	4- 7	60- 69	16-23	0.01	0.01	0.03	0.04	0.11	0.12	0.12	0.12	0.12	0.03	0.08	0.03	0.0355	8
0-140	7000-****	4- 7	70- 79	0- 3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0-140	7000-****	4- 7	70- 79	4-15	0.01	0.01	0.03	0.04	0.07	0.08	0.09	0.09	0.09	0.03	0.04	0.03	0.0227	26
0-140	7000-****	4- 7	70- 79	16-23	0.01	0.02	0.04	0.12	0.14	0.14	0.15	0.15	0.15	0.10	0.02	0.06	0.0592	7
0-140	7000-****	4- 7	80- 89	0- 3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0-140	7000-****	4- 7	80- 89	4-15	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.0	0.0	0.02	0.0	4
0-140	7000-****	4- 7	80- 89	16-23	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.0	0.0	0.10	0.0	1
0-140	7000-****	8-12	30- 39	0- 3	0.01	0.01	0.03	0.04	0.06	0.08	0.08	0.08	0.08	0.03	0.04	0.03	0.0257	8
0-140	7000-****	8-12	30- 39	4-15	0.01	0.02	0.03	0.05	0.26	0.28	0.28	0.28	0.28	0.03	0.23	0.06	0.0849	25
0-140	7000-****	8-12	30- 39	16-23	0.01	0.02	0.04	0.06	0.08	0.09	0.10	0.10	0.10	0.04	0.03	0.04	0.0254	14
0-140	7000-****	8-12	40- 49	0- 3	0.01	0.02	0.03	0.05	0.06	0.06	0.06	0.06	0.06	0.03	0.01	0.03	0.0228	8
0-140	7000-****	8-12	40- 49	4-15	0.01	0.01	0.03	0.04	0.08	0.10	0.13	0.14	0.15	0.03	0.05	0.03	0.0313	44
0-140	7000-****	8-12	40- 49	16-23	0.01	0.02	0.03	0.05	0.07	0.09	0.12	0.13	0.14	0.03	0.05	0.03	0.0278	35
0-140	7000-****	8-12	50- 59	0- 3	0.01	0.01	0.02	0.03	0.03	0.03	0.03	0.03	0.03	0.02	0.0	0.02	0.0100	2
0-140	7000-****	8-12	50- 59	4-15	0.01	0.01	0.02	0.04	0.04	0.04	0.04	0.04	0.04	0.02	0.00	0.02	0.0099	44
0-140	7000-****	8-12	50- 59	16-23	0.01	0.02	0.03	0.05	0.09	0.11	0.13	0.13	0.13	0.04	0.06	0.03	0.0371	15
0-140	7000-****	8-12	60- 69	0- 3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0-140	7000-****	8-12	60- 69	4-15	0.01	0.02	0.03	0.05	0.10	0.13	0.13	0.13	0.13	0.03	0.08	0.03	0.0342	19
0-140	7000-****	8-12	60- 69	16-23	0.01	0.02	0.04	0.11	0.13	0.14	0.14	0.14	0.14	0.09	0.03	0.05	0.0613	3
0-140	7000-****	8-12	70- 79	0- 3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0-140	7000-****	8-12	70- 79	4-15	0.01	0.01	0.03	0.04	0.06	0.06	0.06	0.06	0.06	0.03	0.02	0.02	0.0186	6
0-140	7000-****	8-12	70- 79	16-23	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.0	0.0	0.02	0.0	1
0-140	7000-****	8-12	80- 89	0- 3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0-140	7000-****	8-12	80- 89	4-15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
0-140	7000-****	8-12	80- 89	16-23	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.0	0.0	0.10	0.0	1
0-140	7000-****	13-18	30- 39	0- 3	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.0	0.0	0.05	0.0000	2
0-140	7000-****	13-18	30- 39	4-15	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.0	0.0	0.03	0.0	1
0-140	7000-****	13-18	30- 39	16-23	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.0	0.0	0.05	0.0000	5

Fig. 9.1 Sample Tabulation Prediction System Data Display

meteorological variables are ordered, it is possible to look up any combination just as one would look up a name in a telephone book or a word in a dictionary to obtain the probability distribution of SO_2 concentrations for this combination of meteorological variables.

To develop the Tabulation Prediction Technique for a system of air quality monitoring stations, values of air pollution concentrations, wind speed and wind direction are obtained for the appropriate monitoring station; computer tape or punch card records for other meteorological variables are obtained for the local area from the National Weather Records Center at Asheville, N.C. A computer code which is provided in this manual is then used to generate a set of prediction tables for each monitoring station.

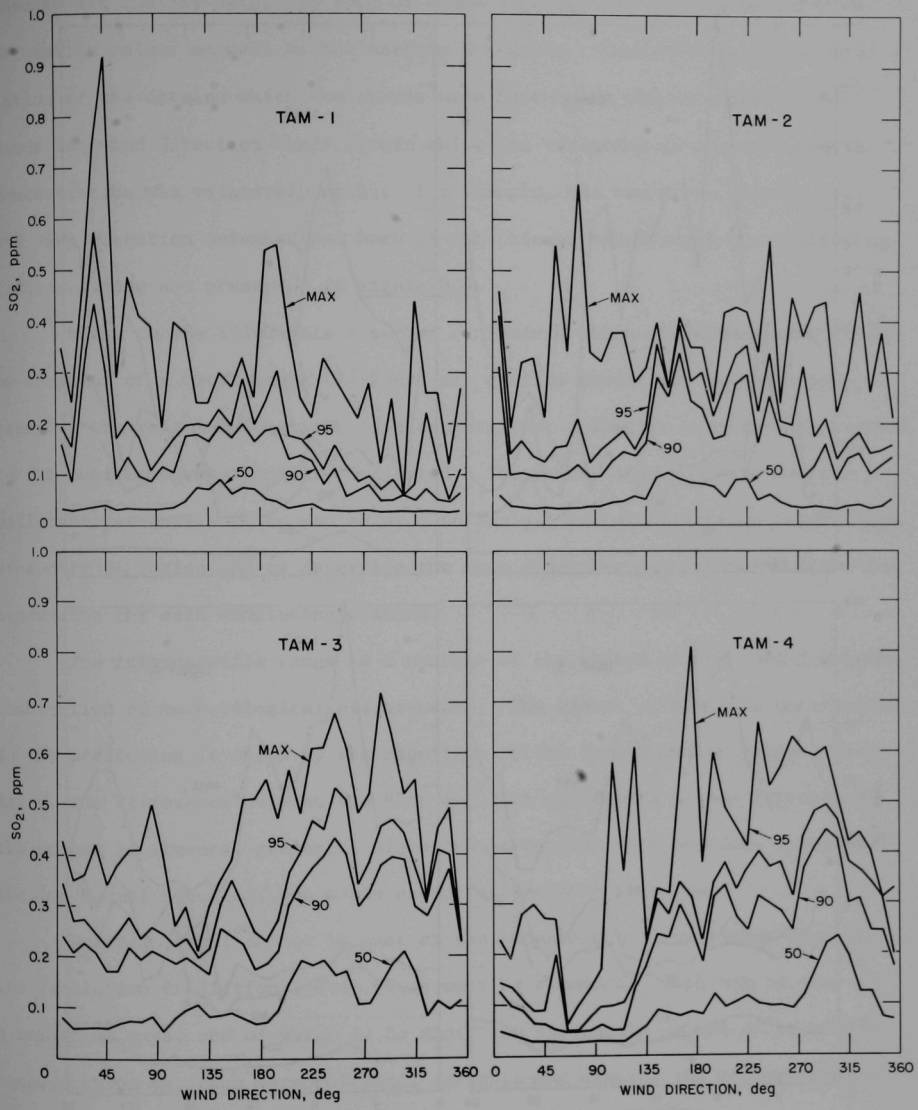
9.2 The Influence of Meteorological Variables on Air Pollutant Concentrations

Since the Tabulation Prediction method depends on the relations between observed air pollution concentration at a receptor and certain critical meteorological variables, it is necessary to understand the principles underlying the meteorological processes. Each of the meteorological variables; wind speed, wind direction, temperature, relative humidity, stability, net solar-terrestrial radiation flux, mixing depth, ceiling, or visibility may play an important role depending on the local air pollution climatology and type of pollutant involved.

A discussion of the role played by meteorological variables in processes which affect air pollution concentration levels will be found in Reference 8.

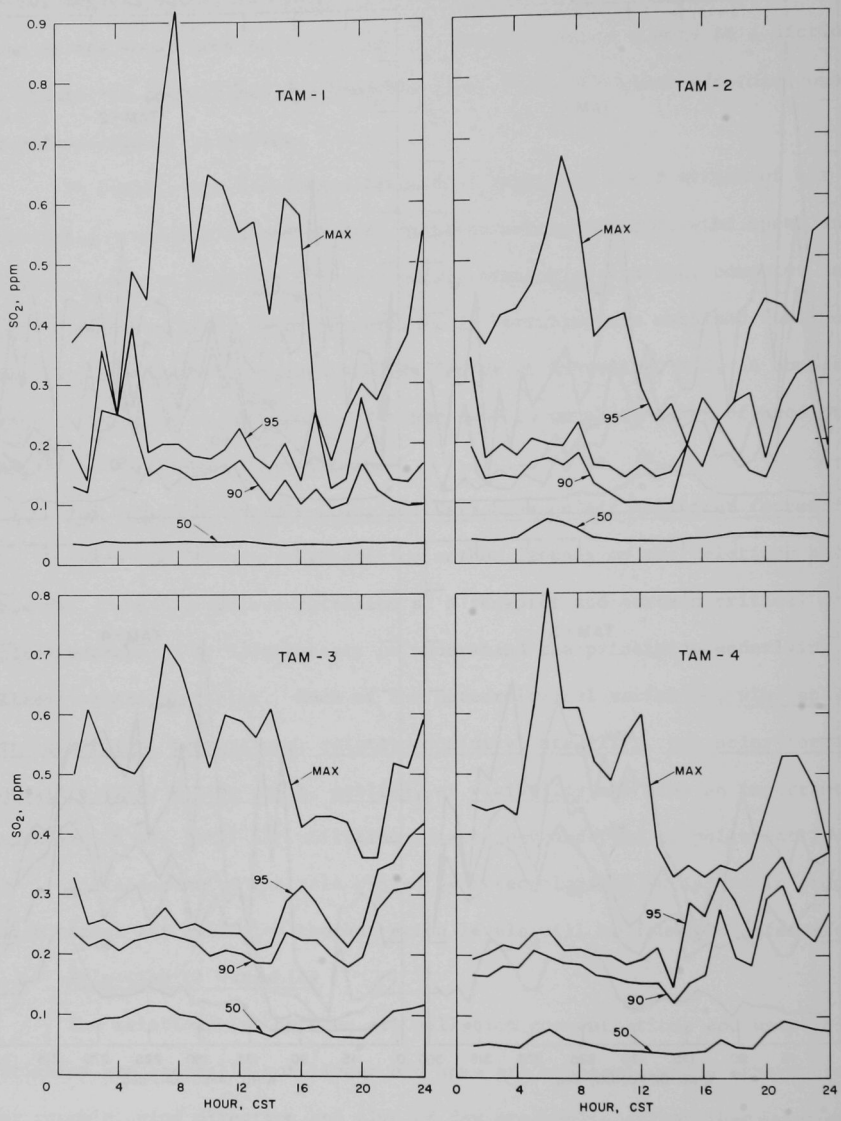
9.3 Selection of Variables

The relationships between air pollution concentrations and meteorological variables are generally not linear and some are monotonic while others are not. For example, wind direction and time of day are cyclic rather than monotonic predictor variables. In order to treat these two variables, plots were made of SO_2 concentration vs. wind direction and SO_2 concentration vs. time of day for each of the telemetered air quality monitoring (TAM) stations in the system. Examples of these charts are shown in Figures 9.2 and



113-2343

Fig. 9.2 So_2 Concentrations vs. Wind Direction at Selected Percentile Values



113-2342

Fig. 9.3 SO_2 Concentrations vs. Time at Selected Percentile Values

Chicago air quality data. In each of these figures, the 50th, 90th and 95th percentile values as well as the maximum are shown. These charts and a tabulation of the data on which the charts were based were used to select time bands and wind direction bands within which the variation of the pollutant concentration was relatively small. For example, the bands for time of day and wind direction selected for each of the Chicago telemetered air monitoring (TAM) stations are presented in Figure 9.4.

These charts illustrate a number of factors which are significant in the development of a short range (24-hour) air quality prediction system. For example, the relative positions of clean areas or pollution sources with respect to the monitoring stations are indicated. The charts also indicate how air pollutant percentile plots can be used to analyze the air pollution meteorology of a city or region and to determine the most effective time and wind direction bandwidths for each monitoring station.

The interquartile range is a measure of the spread of the data for each combination of meteorological measurements. The amount of noise or uncertainty in the prediction is shown by the magnitude of the interquartile range. Similarly, the difference between the 95th and 75th percentile values represents the amount of skewness present. Since the percentile distributions approximate the log normal function, one would expect appreciable skewness.

Before a final choice is made of the meteorological variables used in the Tabulation Prediction System, each must be examined. This may be done in a number of ways, one of which is to plot[+] the percentile values of pollutant concentration vs. each meteorological variable and examine the relationships. The 50, 90, and 95 percentile and maximum curves can be plotted for this purpose in the same format as shown in Figures 9.2 and 9.3.

[+]The computer routines used to plot these percentile values are not included in this manual.

(Chicago Air Quality Monitoring System)

TAM Station #	Wind Direction (Degrees)			Hour (CST)		
	Band I	Band II	Band III	Band I	Band II	Band III
1	55-105 205-355 5-15	111-195	25-45	1-2 18-24	6-17	3-5
2	15-135 275-355	145-265 5		9-14	1-8 15-24	
3	355 5-195	205-345		2-20	1 21-24	
4	5-135	145-265 335-355	275-325	1-4	5-16	17-24
5	165-205 245-255	45-95 135-155 215-235 265-355	5-45 105-125	1-3 5-6 9-10 12-24	4 7-8 11	
6	15-55 145-285 335-345	5 295-325 355	65-135	1-7 23-24	14-22	8-13
7	5-65 275-325 355	225-265 335-345	75-215	2-4	1 5-9 17-18 23-24	10-16 19-22
8	45-65 135-305	5-35 75-125 315-355		1-6 11-15 17-24	7-10 16	

113-1152

Fig. 9.4 Tabulation Prediction System Bands for Wind Direction and Hour of Day

Where a network of stations is available, such as exists in New York City, Los Angeles, Washington, or Chicago, the receptor-oriented tabulation prediction technique may be applied to each of the stations to obtain isopleths of pollut- and concentrations.

9.4 Construction of the Tabulations

The meteorological variables recommended for the construction of a Tabulation Prediction Scheme are:

1. Wind direction
2. Ceiling height
3. Wind speed
4. Temperature
5. Hour of the day

The importance of each of these variables is described in Reference 8.

The significance of atmospheric stability has been noted, but such data are usually unavailable. A substitute is provided by the choice of the variables: ceiling height, hour of the day, and wind speed. Combinations of these are closely related to stability. For example, during high winds, e.g., near 15 knots, the stability is usually neutral. This is true during the day or night. During periods with unlimited ceilings accompanied by light winds, an inversion is quite likely at night, but unstable conditions are to be expected during the day.

Recommended bandwidths for the variables, ceiling height, wind speed and temperature are shown in Figure 9.5. As a first step, the data are stratified by season and by the presence or absence of precipitation. An intermediate tabulation is then prepared in which wind direction and hour of day are listed as Band I, Band II and so on. For example, for TAM station 1, in Chicago, the wind directions 5-15, 55-105, 205-355 are all contained in wind band category I. The final form of the tabulation should substitute actual values of wind

1. Wind Speed

   Unit: miles per hour

   Group Interval: 0-3, 4-7, 8-12, 13-18, 19-24, >24

2. Wind Direction

   Unit: degrees

   Group Interval: standard 45° grouping as indicated in Section 9.5 (0-45, 45-89, 90-135, etc.) or according to bandwidths selected as indicated in Section 9.3

3. Temperature

   Unit: °F

   Group Interval: -20--11, -10--1, 0-9, 10-19, 20-29, 30-39, 40-49, 50-59, 60-69, 70-79, 80-89, 90-99

4. Ceiling Height

   Unit: feet

   Group Interval: 0-999, 1000-2499, 2500-6999, >7000

5. Hour of Day

   Unit: one hour

   Group Interval: Standard intervals recommended in Section 9.5 or according to bandwidths selected as indicated in Section 9.3.

6. Season

   Group Interval: Dec. - Jan. - Feb.

   Mar. - Apr. - May

   June - July - Aug.

   Sept - Oct. - Nov.

7. Precipitation (Optional):

   1. All hours for which no more than trace amounts of precipitation are recorded.

   2. All hours for which more than trace amounts of precipitation are recorded.

113-1153

Fig. 9.5 Units and Group Intervals for Variables Used in the Tabulation Prediction System

direction and hour in numerical order, as shown in Fig. 9.1. The percentile distribution of SO_2 or any other pollutant corresponding to the wind band category, in combination with the appropriate meteorological variables, is then presented for wind direction in each of these ranges.

The order of the variables in the columns are arranged so that the last column represents the variable which influences the SO_2 concentrations the least and the first column the most. Further, wind speed and temperature are presented in descending order since lowest values of each correspond to highest SO_2 concentrations. Thus, the tabular arrangement shows a general increase in values of SO_2 concentration as one reads down from the top of the table.

In summary, the construction of a tabulation prediction table for a system of telemetered air monitoring stations consists of the following steps:

1. Partition the meteorology and air quality data for each station by season.
2. Prepare plots of pollution concentration vs. wind direction and vs. time of day for each group of data obtained in 1, above.
3. From the plots prepare in 2, above, select from two to four time intervals and up to eight wind direction intervals within which the variation of pollution concentration is relatively small. Several different wind direction intervals may be collected together in one wind band, if the pollution concentrations observed for these intervals do not vary significantly. For example, if 95% of the SO_2 concentrations observed at a given station were between 0.1 and 0.2 ppm for winds in the intervals 10°-20° and 95°-120°, data from both of these wind direction intervals could be assigned to the same wind band.
4. Partition and print out the data in each wind direction and time band determined from 3, above, by time band, ceiling height, wind direction, wind speed and temperature. This operation is performed by the computer code included in this manual once the wind direction and time bands have been selected. The computer code (see Section 9.8 of this manual) contains an option to accept or reject data associated with periods during which precipitation occurred.

5. Repeat the above procedure for each monitoring station in the system. The wind direction and time bands selected will normally differ from one station to the next due to differences in the operating characteristics and distribution of sources in the vicinity of the monitoring stations.

9.5 Construction of a Simplified Version of the Tabulation Prediction System

The procedure described above was designed to make the most efficient use of the available data by minimizing the number of wind direction and time bands and therefore maximizing the amount of data displayed in each band. As indicated above, an analysis of the available meteorological and air quality data is required in order to select the most efficient set of wind and time bands. This analysis may be avoided, and a somewhat simplified version of the tabulation prediction system may be produced if standard wind direction and time bands are used for all stations. For example (see Figure 9.6 below), a set of "standard" wind direction bands might be:

Wind Direction		Time	
Band 1:	0-45°	Band 1:	0700-1100
Band 2:	45-90°	Band 2:	1100-1700
Band 3:	90-135°	Band 3:	1700-2300
Band 4:	135-180°	Band 4:	2300-0700
Band 5:	180-225°		
Band 6:	225-270°		
Band 7:	270-315°		
Band 8:	315-360°		

113-1154

Fig. 9.6 "Standard" Intervals for Wind Direction and Time Bands

The tabulation prediction system computer listing provided in this manual is arranged to partition input data according to these standard bands; however, as indicated in Section 9.8, the wind direction and time bands may be adjusted to correspond to the special characteristics of each air quality monitoring station in the system. For example, inspection of a source distribution map for the control region may indicate that certain wind direction bands could be expected to be significant for a given monitoring station. Use of these bands would be more effective than an arbitrary grouping of data by wind quadrants or octants, etc.

9.6 Application of the Tabulation Prediction System to Daily Operations

Since the variables used in the Tabulation Prediction Scheme are wind direction, wind speed, temperature, ceiling height, and hour of day, these are the variables (except of course the hour of day) for which meteorological forecasts are required. The forecasts should be made for each hour. The range of variability, e.g., a wind speed forecast over the range of 4-7 miles per hour, should correspond to the ranges recommended in Figure 9.5

The meteorological forecast for the day is preferably made between 0700 and 0800 so that the early morning pollution peak will be covered. The development of forecasts for 12 to 24 hours is a reasonably routine procedure. These forecasts may be issued every 6, 12, or 24 hours depending on the local requirements. Of course, during potential or actual severe pollution conditions, constant surveillance is necessary and automatic 6-hour updating of the 24-hour forecast is appropriate.

A format for recording the forecasts and the computed and observed pollution concentrations is provided in Fig. 9.7. More variables are indicated on the form than may actually be used in preparing an air quality prediction; but a control agency may use this form as a guide in preparing one that is tailored to its specific needs. (The symbol Prb on this form stands for the probability est[illegible]cation.)

DATE: _____

TIME: _____

GIVEN BY: _____

RECEIVED BY: _____

CONTROL POINT:

MIDWAY AIRPORT:

HOUR	1. SURFACE WIND DIRECTION	Prb.	2. SURFACE WIND SPEED (MPH)	Prb.	3. MIXING HEIGHT INDEX	Prb.	4. DRY BULB TEMPERATURE (SURFACE) (°F)	Prb.	5. CEILING BELOW 7,000 (YES/NO)	Prb.	6. RELATIVE HUMIDITY (%)	Prb.	7. CLOUD COVER	Prb.	8. CEILING HEIGHT (ft.)	Prb.	9. PRECIPITATION (OVER A TRACE) (YES/NO)	Prb.	10. VISIBILITY (mi.)	Prb.	11. STABILITY CLASS	12. MIXING HEIGHT (ft.)	Prb.
1																							
2																							
3																							
4																							
5																							
6																							
7																							
8																							
9																							
10																							
11																							
12																							
13																							
14																							
15																							
16																							
17																							
18																							
19																							
20																							
21																							
22																							
23																							
24																							

113-1155

Fig. 9.7a Air Quality Prediction System Data Form

HOUR	1		2		3		4		5		6		7		8	
	PRED.	ACT.	PRED.	ACT.	PRED.	ACT.	PRED.	ACT.	PRED.	ACT.	PRED.	ACT.	PRED.	ACT.	PRED.	ACT.
1																
2																
3																
4																
5																
6																
7																
8																
9																
10																
11																
12																
13																
14																
15																
16																
17																
18																
19																
20																
21																
22																
23																
24																

113-1156

Fig. 9.7b Air Quality Prediction System Data Form
SO_2 Predictions and Actual Readings

After the meteorological forecast values are received, the 50th percentile entry of the Tabulation Prediction Scheme Tables may be used to produce an air pollution concentration prediction for each hour of the forecast period. A better method of using the tables, however, is to determine from the actual, measured air pollution concentrations for the first hour of the forecast period, what percentile entry should be used to make the prediction for the next 6 hours. For example, if the actual SO_2 value for the first hour of the forecast period is 0.3 ppm and this value corresponds to the 80th percentile entry in the appropirate part of the table, then the air quality predictions for the next 6 hours should be taken from the 80th percentile entries. The 50th percentile may be used for the remaining 18 hours of the forecast period. When the forecast is updated six hours later, a new percentile entry is determined from the most recent measured air quality data, and the procedure is repeated to produce an updated air quality prediction.

Municipal air pollution control agencies may find it uneconomical to mount a full scale, 24-hour meteorological forecasting operation. Hourly forecasts must therefore be obtained from the U.S. Weather Bureau or from a commercial meteorological firm. The services of one or two meteorologists employed by the air pollution control agency may be used to develop and interpret the forecasts of both the weather and pollutant concentration.

The control agency meteorologist determines whether high air pollution concentrations are probably within the forecast period. His access to the latest upper air and surface weather charts enables him to judge whether; 1) a land-sea breeze will develop, 2) a thunderstorm is likely, 3) precipitation or fog will appear, 4) a frontal system will pass, or 5) a lowering of the mixing depth is probable because of pronounced subsidence. Any one of the above or similar phenomena markedly affects the pollution potential.

The control agency meteorologist may issue forecasts such as a 24-hour IDA air pollution index for the entire control region (see Reference 8) or the maximum concentration levels for selected areas where special forecasts are warranted.

With the availability of on-line computer facilities, some functions may be automated. For example, the hourly meteorological forecast may be entered into the computer and a printout of the 50th percentile air pollution level may be produced, based on Tabulation Prediction Scheme Tables stored on a disc or tape memory. It is possible to produce computer plotted maps of the control region with pollutant concentration levels and meteorological variables. Such maps are currently generated at Argonne National Laboratory using the APICS display program described in Section 7.3.

The Tabulation Prediction System is a relatively new development in air pollution forecasting. There are a number of areas in which improvement is possible, but the improvements must be based on the individual needs of the user. For example, the control agency meteorologist should conduct investigations to determine the conditions under which high values of pollutant concentrations occur; e.g., such as are found in the 90th to 100th percentile range. An indication of how this may be done is provided in Reference 8 which discusses the synoptic weather conditions associated with a selected group of 20 air pollution episodes. Case studies of high pollution episodes should also be made. The results of these may be applied to optimize the use of the Tabulation Prediction Scheme. Further, an investigation involving the persistence of high pollution levels should be made in each city desiring an air pollution advance warning capability. As indicated above, the percentile value of the SO_2 concentration at the beginning of the forecast period may be a good indicator of the probability that the concentration values will exceed the 50th percentile value.

9.7 Advantages and Disadvantages of the System

Among the advantages of the tabulation prediction scheme are:

1. It is easy to use and does not require on-line, real time access to a computer.
2. It provides rapid on-line prediction of pollution concentrations.
3. It provides the entire percentile distribution of pollutant concentrations to allow a forecaster to "fine tune" his predictions based on synoptic situations.
4. It takes into account non-linearities in the relationship between meteorological variables and pollution concentrations.
5. It is an effective method of analyzing and displaying air quality and air pollution meteorology data.

Disadvantages of this method are that:

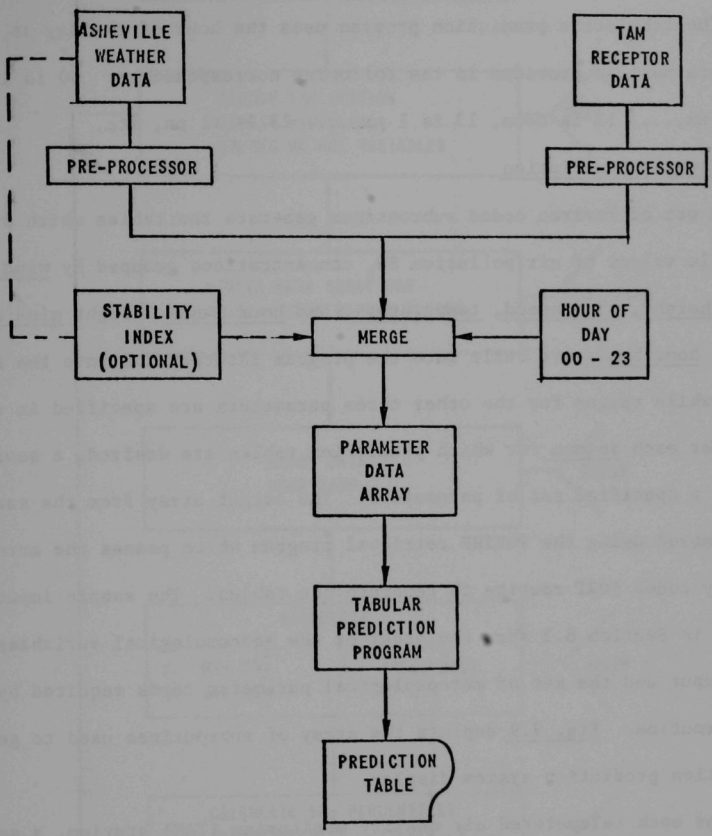
1. A digital computer is required in order to construct the tables.
2. It must be updated at 1-2 year intervals as the source strength and distribution patterns of the area change.
3. Since it is a receptor-oriented method, it cannot predict air pollution concentrations at points between monitoring stations, except indirectly through interpolation or logical inference.

9.8 Tabulation Prediction System Computer Program

To construct the prediction table described above, hourly values of selected meteorological variables over the time range of interest must be available. The bases for selecting these variables are discussed above and in Reference 8. This section describes the computer program which creates the prediction table and provides the necessary information for its utilization.

9.8.1 Input Information

The information flow to the computer program is depicted schematically in Fig. 9.8. In order to use the tabulation prediction program, the input



113-1157

Fig. 9.8 Information Requirements of Tabulation Prediction Program

parameter values must be prepared in an array with the format described below. This requires extracting the parameter values from the appropriate data sources as shown in the figure.

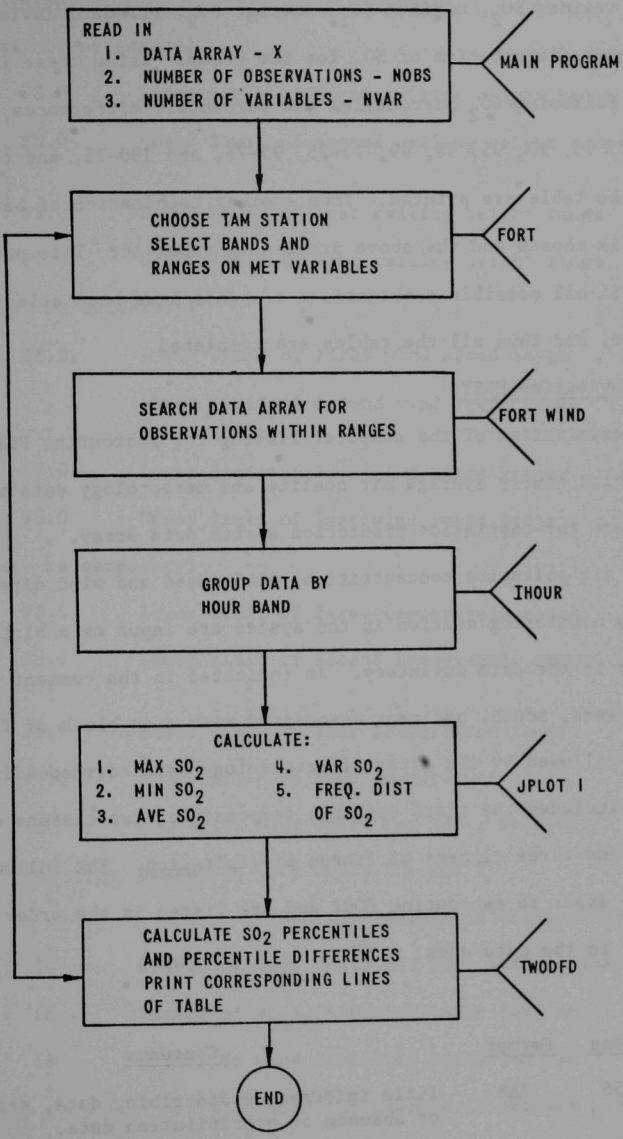
The tabulation prediction program uses the hour of the day as a predictor. This must be provided in the following correspondence: 00 is 12 midnight; 01 is 1 am,..., 12 is noon, 13 is 1 pm,..., 23 is 11 pm, etc.

9.8.2 Program Description

A set of Fortran coded subroutines generate the tables which display percentile values of air pollution SO_2 concentrations grouped by wind direction, ceiling height, wind speed, temperature, and hour bands. Eight wind directions and four hour bands are built into the program incorporated into the APICS system, while ranges for the other three parameters are specified in the input.

For each season for which prediction tables are desired, a search is made for a specified set of parameters. The output array from the search is then accessed using the FORTRN retrieval program which passes the array to the specially coded FORT routine to generate the tables. The sample input listings provided in Section 8.3 show the order of the meteorological variables in the search input and the set of meteorological parameter cards required by the FORT subroutine. Fig. 9.9 depicts the array of subroutines used to generate a tabulation prediction system display.

For each telemetered air quality monitoring (TAM) station, a series of nested DO loops indexes over each combination of the set of bands (for wind direction) and the sets of ranges (for ceiling height, wind speed and temperature). The data in array X is scanned to determine which observed values fall within the selected bands and ranges. Any variable in X which has been recorded as $\geq 10^5$ (implying that the data point was actually missing) causes the observation to be discarded. The remaining observations in X which meet the selection criterion are then grouped by hour bands and their SO_2 values are used to



113-1158

Fig. 9.9 Tabulation Prediction System Computer Program Schematic

determine maximum SO_2, minimum SO_2, average SO_2, standard deviation of SO_2, and the frequency distribution of SO_2 for the group. Using these frequency distributions the following SO_2 percentiles and percentile differences are determined: 10, 25, 50, 75, 90, 95, 98, 99, 75-25, 95-75, and 100-75, and the corresponding lines of the table are printed. Then another combination of bands for the variables is chosen and the above process is repeated. This procedure continues until all possible combinations of bands have been selected for each TAM station, and thus all the tables are completed.

Input to Subroutine FORT

An examination of the computer listing for subroutine FORT indicates the order in which hourly average air quality and meteorology data must be input in order to form the tabulation prediction system data array.

The air pollution concentration, wind speed and wind direction for each air quality monitoring station in the system are input as a block for each hour of each day in the data inventory. As indicated in the comment statement, the hour, day, week, month, and year associated with each block of TAM station data are input, followed by the array of meteorology data corresponding to that hour. This array includes the cloud ceiling, temperature, two classes of liquid precipitation and three classes of frozen precipitation. The following cards are required as input to subroutine FORT and are listed in the order in which they must appear in the data deck.

Card Type	Columns	Format	Contents
1	1-56	7A8	Title information describing data, season, and presence or absence of precipitation data.
	57-80	3A8	Period of data
2	1-8	I8	Number of TAM stations
	9-16	I8	Number of ceiling height ranges (less than 20)
	17-24	I8	Number of wind speed [illegible]
	25-32	I8	Number of temperatu[illegible]

Card Type	Columns	Format	Contents
3	(two cards may be necessary)		
	1-8	F8.0	Lower limit of first ceiling height range
	9-16	F8.0	Lower limit of second ceiling height range
	.	:	:
	.	:	:
	.	F8.0	Lower limit of last ceiling height range
	.		
	.	F8.0	Upper limit of last ceiling height range
4	(two cards may be necessary)		
	1-8	F8.0	Lower limit of first wind speed range
	9-16	.	Lower limit of second wind speed range
	.	:	:
	.	:	:
	.	F8.0	Lower limit of last wind speed range
	.		
	.	F8.0	Upper limit of last wind speed range
5	(two cards may be necessary)		
	1-8	F8.0	Lower limit of first temperature range
	9-16	F8.0	Lower limit of second temperature range
	.	:	:
	.	:	:
	.	F8.0	Lower limit of last temperature range
	.		
	.	F8.0	Upper limit of last temperature range
6	1-8	I8	Number of hour bands for TAM 1 receptor
	9-16	I8	Number of hour bands for TAM 2
	.	.	.
	.	.	. (See the example in Section 8.3)
	.	.	.
	.	I8	Number of hour bands for last TAM station
7	1-8	I8	Number of wind direction bands for TAM 1
	9-16	I8	Number of wind direction bands for TAM 2
	.	.	.
	.	.	. (See the example in Section 8.3)
	.	.	.
	.	I8	Number of wind direction bands for last TAM station
8	1-8	I8	1 -- use data with precipitation present
			0 -- use data with no precipitation present

In order to modify the standard hour and wind direction bands, the user must alter the existing version of the function IHOUR and the subroutine WIND which, respectively, define the hour and wind direction bands for each TAM station. The listing of the tabulation prediction code as described in Section 9.8.3 of this manual is arranged to partition data for all monitoring stations into the "standard" wind direction and time bands recommended in Section 9.6. That is, octant wind bands and four selected significant time bands. If the user wishes to modify this standard system, the following changes are required.

Wind Direction Bands

The partitioning of data by wind direction bands is performed by subroutine WIND which appears in the listing as:

WIND (W, K, J1, IND)

where: W = wind direction associated with each data point in the input data array

K = number of the air quality monitoring station to which the input data point corresponds

J1 = number of the wind direction band

IND = data classification index = 1 if the input data point does fall in band J1

= 0 if the input data point does not fall in band J1

Subroutine WIND contains a series of IF statements written for W which perform the function of determining whether a given input data point belongs in band J1. To modify the recommended standard wind bands, therefore, the user must replace this series of IF statements with an alternative set. If the wind bands are not the same for all monitoring stations, then a set of IF statements on W must be written into WIND for each monitoring station K in the system.

Time Bands

The partitioning of data by hour bands is performed by function IHOUR which appears in the listing as:

IHOUR(Y,K)

where: Y = hour associated with each data point in the input data array

K = number of the air quality monitoring station to which the input data point corresponds.

The value of IHOUR is set equal to the number of the time band in which the particular input data point belongs. Function IHOUR contains a series of IF statements, written for Y, which perform the function of determining in which time band a given input data point belongs. As indicated in Section 9.6, there are four bands in the recommended standard time band system.

To modify the standard time bands, the user must replace the series of IF statements with an alternative set. If the time bands are not the same for all monitoring stations, then a set of IF statements must be provided for each monitoring station K in the system, as shown in the computer listing for IHOUR.

Each data point from a given monitoring station is assigned to a wind direction band and time band by WIND and IHOUR. When a complete array of data points has been compiled for each set of bands, the percentile distributions maxima and minima are calculated and printed by subroutine TWODFD.

9.8.3 Tabulation Prediction System Computer Listing

Appendix H provides a listing of the Fortran computer program required to generate a set of tabulation prediction system tables. The version of the code presented here incorporates the recommended standard meteorological and time bands and has been written to accommodate data from as many as eight telemetered air monitoring stations.

Acknowledgements

The authors of this document are indebted to R. A. McCormick and L. Niemeyer of the Division of Meteorology of the National Air Pollution Control Administration, whose guidance, patience, and sponsorship were essential to the evolution of the APICS system.

Argonne personnel who contributed directly to the development of APICS are:

Participating Scientists

J. E. Carson, D. F. Gatz, B. S. Garbow,
M. R. Kraimer, M. T. Matthies, and S. M. Prastein.

Technical Support

R. Carter, L. Conley, W. Grynczyk,
S. Katilavas, J. Korn, G. Krause,
B. Nieubuurt, and O. Skala

Scientists from other institutions who contributed are:

D. B. Turner, National Air Pollution Control Administration,

S. Booras* and E. Klappenbach of the Chicago Department of Environmental Control.

*Mr. Booras is currently associated with Environmental Pollution Control Systems, Hinsdale, Illinois.

REFERENCES:

1. Croke, E. J., et al., *Chicago Air Pollution System* Model, ANL/ES-CC-001 (February 1968).

2. Croke, E. J., et al., *Chicago Air Pollution System Model*, ANL/ES-CC-002 (May 1968).

3. Croke, E. J., et al., *Chicago Air Pollution System Model*, ANL/ES-CC-003 (October 1968).

4. Bradley, J. P., unpublished communcation to H. Moses (February 1969).

5. Croke, E. J., et al., *Chicago Air Pollution System Model*, ANL/ES-CC-004 (April 1969).

6. IBM System/360 Scientific Subroutine Package - Programmer's Manual.

7. Roberts, J. J., Croke, E. J., and Kennedy, A. S., *An Urban Atmospheric Dispersion Model*, ANL/ES-CC-005 (October 1969).

8. Croke, E. J., and Roberts, J. J., *Air Pollution Incident Control Manual*, NAPCA (to be published).

9. Turner, D. Bruce, *A Diffusion Model for an Urban Area*, J. Applied Meteorology 3 83-91 (1964).

10. Airways Surface Observations, *Tape Reference Manual*, ESSA/EDS-NWRC TDF 14.

11. Matthies, Michael T., *The FORTRAN-PL/I Interface Problem*, ANL Report No. 7435 (1968).

12. Matthies, Michael T., *A PL/I-to-FORTRAN Interface Routine*, ANL-AMD Technical Memorandum No. 181 (1969).

13. Chamot, C., *YOLYPLOT: S/360 Plotting Routine*, AMD Subroutine Library No. ANL J952S (1966).

14. Chamot, C., *Guide to Accessing Multifile Magnetic Tapes from FORTRAN Programs Under OS/360*, ANL Report No. 7507 (1968).

15. Chamot, C., *Execution of Production Programs on the IBM System 360 Computer*, ANL-AMD Technical Memorandum No. 177 (1969).

REFERENCES (Cont'd.):

16. Kennedy, Allen S., *POINTR, A Dynamic Storage Allocation Program*, ANL-AMD Technical Memorandum No. 98 (1967).

17. Kennedy, Allen S., *BPOINTR: New Version of POINTR for LCS*, ANL-AMD Internal Memorandum dated December 13, 1968.

18. Krupp, Ronald F., *S/360 Programming Techniques for the CALCOMP 780*, ANL-AMD Technical Memorandum No. 130 (1967).

19. Ohde, John A., *ALLOC$: S/360 Routine for Dynamic Allocation of Memory*, AMD Subroutine Library No. ANL Z071S (1968).

20. Morrison, Donald M., and Art, Richard C., *EIDISC: A FORTRAN Program for Stepwise Multiple Discriminant Analysis*, Northwestern University, Vogelbeck Computer Center Library No. NUC0068 (1968).

APICS USER MANUAL

APPENDIX A

GUIDES TO USING THE APICS SYSTEM

APPENDIX A: Guides to Using the APICS System

1. User Manual References to APICS Data Formats
2. Flow of Information through APICS
3. Sample Construction of a Master Tape
4. Systematic Identification of APICS Data Tapes
5. DD Card Specifications for Input/Output Tapes
6. Execution of a Program from Source Code
   a. PL/I Coding Only
   b. PL/I and Fortran
7. Storing Programs on the Data Cell from Source Code
   a. Complete Programs in Executable Form
   b. Incomplete Programs to be Combined with User-Coded FORT Routines
8. Using Programs from the Data Cell
   a. Complete Programs
   b. Combining an Incomplete Program with a User-Coded FORT Routine (Source Code
9. Deck Structures to Utilize Data on Master Tapes
   a. One Jobstep (SEARCH2 with FORT Option Specified)
   b. Two Jobsteps to Search and Operate on Retrieved Data
      (Example: CITYMAP Display Program)
10. Maintaining the APICS Library on the Data Cell
   a. Generating a Backup Copy on Tape
   b. Restoring the Data Set from Tape (Scratch, Save, and Copy)
   c. Cataloguing and Listing the PDS APSLIB

NOTE: A catalogued data set is assumed for the library of APICS routines on the data cell. Otherwise the DD cards must include unit and volume serial parameters; e.g., UNIT=2321 and VOLUME=SER=CELLBH.

APPENDIX A1: User Manual References to APICS Data Formats

a. Master File Table of Contents (Table II)

Summary of observed (and calculated) data items included in the master file (TAM, Calendar, Airport, etc.)

b. Meteorological Data Codes (Table III)

Supplementary information on the data items in Table II

c. Tape Format for TAM Data (Table IV)

Detailed format of raw TAM data tapes

d. Tape Format for Airways Surface Observations (Table V)

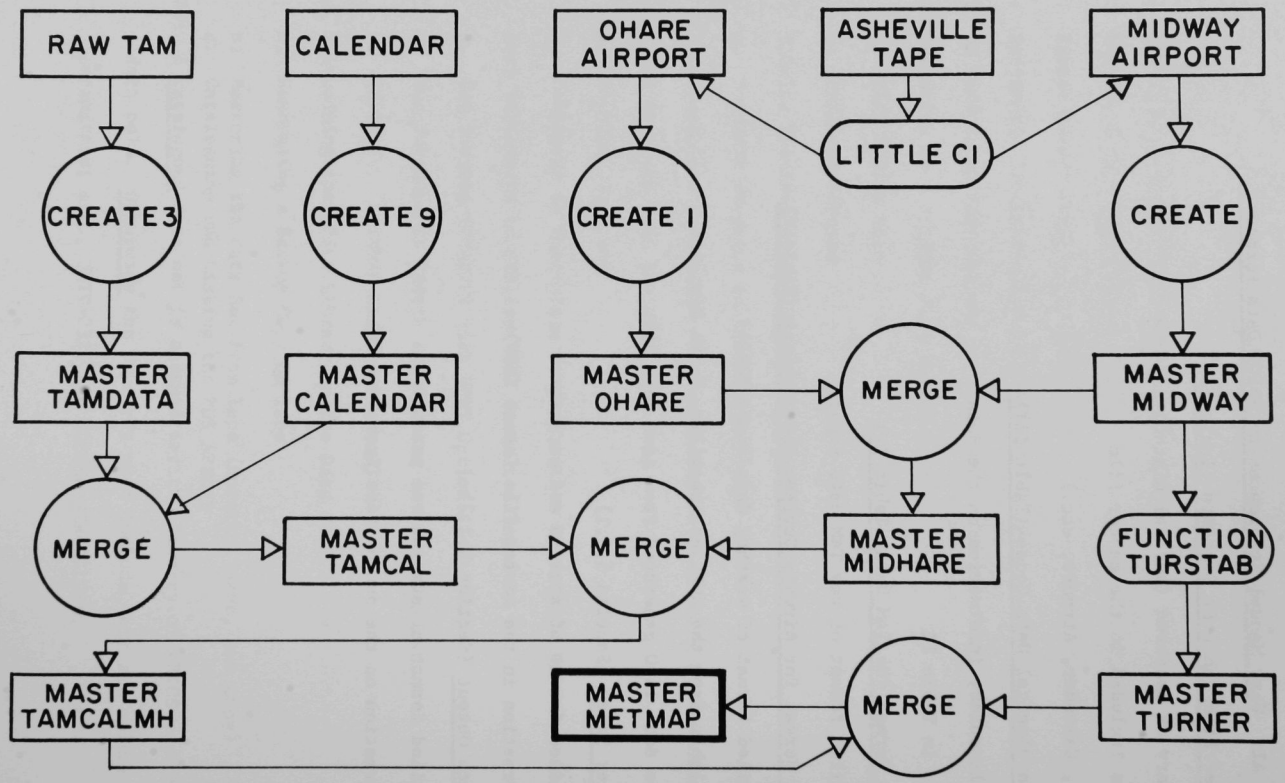
Detailed format of weather data tapes-TDF14-available from the NWRC at Asheville, North Carolina (tape deck 1440 generated from card deck 144)

e. Master Tapes (Section 5.1.1)

Detailed format of station and parameter information in the master file format (MFF)

f. Search Output (Section 5.1.2)

Detailed format of station and parameter information on the output tape from a search



APPENDIX A3

SAMPLE CONSTRUCTION OF MASTERTAPE

APPENDIX A4: Systematic Identification of APICS Data Tapes

Information should be punched on data cards as follows with each card representing a data tape. (∿8 column fields)

Columns	Contents
1	7 or 9 to indicate a 7-track or 9-track tape
2	status indicator: (optional) = means this is a duplicate of another (library) tape + means a sequel exist - means something about the tape is not right X means the tape is not needed any more Y means the run to generate information on the tape has not yet been completed successfully Z means the tape has been released
3-8	date (year, month, and day) the tape was created, e.g.: 690815
10-14	library tape number, e.g: 69519
15-16	-1,-2, or -3 if there are 2 or more files on the tape (a separate card is used for each file)
18-23	date (year, month, and day) of first data in the file
26-31	date (year, month, and day) of last data in the file
33-40	dsname for 9-track labelled tapes or NONE for 7-track tapes
41-56	summary of the contents of the tape
57-64	name of generating program, or of off-line computer in case of copies
65-72	library number of source tape used to create the tape (or NWRC in case of copies)
73-80	a secondary source tape used as input to the run creating the tape

These cards may then be sorted as desired and used as a basis for picking appropriate tapes to use with the various programs.

APPENDIX A5 (Cont'd.):

p. Concatenating TWO Tapes (7-track unlabelled) for CREATE1

ASP Setup Cards

```
/*SETUP      DDNAME=SOURCE,DEVICE=2400-7,ID=(066666,NORING,SAVE,NL)
//*SETUP     DDNAME=SOURCE,DEVICE=2400-7,ID=(066667,NORING,SAVE,NL)
```

JCL DD Cards

```
//GO.SOURCE      DD      LABEL=(,NL),UNIT=TAPE7TRK,DISP=(OLD,PASS),
//               VOLUME=(SER=066666,RETAIN),
//               DCB=(,DEN=1,TRTCH=ET,RECFM=F,BLKSIZE=495)
//               DD      LABEL=(,NL),UNIT=TAPE7TRK,DISP=(OLD,PASS),
//               VOLUME=(SER=066667,RETAIN),
//               DCB=(,DEN=1,TRTCH=ET,RECFM=F,BLKSIZE=495)
```

q. Concatenating TWO Files on a Tape (7-track unlabelled) for CREATE 3

ASP Setup Card

```
/*SETUP      DDNAME=SOURCE,DEVICE=2400-7,ID=(071112,NORING,SAVE,NL)
```

JCL DD Cards

```
//GO.SOURCE      DD      LABEL=(,NL),UNIT=TAPE7TRK,DISP=(OLD,PASS),
//               DCB=(,DEN=1,TRTCH=ET,RECFM=F,BLKSIZE=7872)
//               DD      LABEL=(2,NL),UNIT=TAPE7TRK,DISP=(OLD,PASS),
//               VOLUME=(,RETAIN,REF=*.SOURCE),
//               DCB=(,DEN=1,TRTCH=ET,RECFM=F,BLKSIZE=7872)
```

APPENDIX A6: Execution of Programs from Source Code

a. Deck Structure for Programs Written in PL/I Only

Job and Accounting Cards

```
/*SETUP        cards as needed (See Appendix B and A5)
//     EXEC     PL1CLG
//PL1.SYSIN     DD     *
        PL/I source code
/*
†→
//GO.ddname     DD     cards as needed (See Appendix B and A5)
//GO.SYSIN     DD     *
        Input data cards
/*
```

---

†→ Indicates location for a SYSLMOD DD card referenced in Appendix A7a.

APPENDIX A5: DD Card Specifications (and ASP Setup Cards)

a. CARDS: A scratch data set used for intermediate storage of search-like inpu

```
//GO.CARDS      DD      UNIT=DISK,DISP=(NEW,DELETE),
//      SPACE=(TRK,(10,10)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
```

b. FT06F001: Standard Fortran printed output unit

```
//GO.FT06F001      DD      SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=798)
```

c. MASTER: A master tape to be used for a Search (MFF)

```
//GO.MASTER      DD      DSNAME=METMAP,UNIT=TAPE9TRK,
//      VOLUME=(,RETAIN),DISP=(OLD,KEEP,KEEP)
```

d. NEWFILE: A master tape from a Create or Merge (MFF)

```
//GO.NEWFILE      DD      DSNAME=METMAP,UNIT=TAPE9TRK,
//      DISP=(NEW,KEEP),DCB=(RECFM=V,BLKSIZE=20000)
```

e. OLDFIL1: A master tape to be Merged with OLDFIL2 (MFF)

```
//GO.OLDFIL1      DD      DSNAME=TAMDATA,UNIT=TAPE9TRK,DISP=(OLD,KEEP)
```

f. OLDFIL2: A master tape to be Merged with OLDFIL1 (MFF)

```
//GO.OLDFIL2      DD      DSNAME=METDATA,UNIT=TAPE9TRK,DISP=(OLD,KEEP)
```

g. OUTPUT: Contains an Output Array { from a Search program / for an Analysis program }

```
//GO.OUTPUT      DD      DSNAME=OUTPUT,UNIT=TAPE9TRK,
//      DISP={(NEW,PASS),DCB=(RECFM=VB,BLKSIZE=20000)
            (OLD,KEEP)
```

h. SCRATCH: Intermediate storage for collecting the output array for a Search

```
//GO.SCRATCH      DD      UNIT=DISK,SPACE=(CYL,(5,5)),
//      DCB=(RECFM=VB,BLKSIZE=5000),DISP=(NEW,DELETE)
```

i. SOURCE: Data input for Create programs

```
//GO.SOURCE      DD      LABEL=(,NL),UNIT=TAPE7TRK,DISP=(OLD,KEEP),
//      DCB=(,DEN=1,TRTCH=ET,RECFM=F,BLKSIZE=495),VOLUME=(,RETAIN)
```

APPENDIX A5 (Cont'd.):

j. SUBFILE: A master tape from SUBFILE or FUNCTION (MFF)

```
//GO.SUBFILE     DD     DSNAME=SUBFIL,UNIT=TAPE9TRK,
//     DISP=(NEW,KEEP),DCB=(RECFM=V,BLKSIZE=10000)
```

k. SYSPRINT: Standard PL/I printed output unit

```
//GO.SYSPRINT     DD     SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=798)
```

ℓ. 7-track unlabelled tape: ASP Setup card for creating the tape

```
/*SETUP     DDNAME=SOURCE,DEVICE=2400-7,ID=(,RING,SAVE,NL)
```

m. 7-track unlabelled tape: ASP Setup card for using the tape

```
/*SETUP     DDNAME=SOURCE,DEVICE=2400-7,ID=(071010,NORING,SAVE,NL)
```

n. 9-track labelled tape: ASP Setup card for creating the tape

```
/*SETUP     DDNAME=OUTPUT,DEVICE=2400-9,ID=(,RING,SAVE,SL)
```

o. 9-track labelled tape: ASP Setup card for using the tape

```
/*SETUP     DDNAME=OUTPUT,DEVICE=2400-9,ID=(069519,NORING,SAVE,SL)
```

APPENDIX A6 (Cont'd.):

b. Deck Structure for Programs Written in Fortran and PL/I

Job and Accounting Cards

```
/*SETUP        cards as needed (See Appendix B and A5)

//      EXEC      FTHCP

//FTH.SYSLIN      DD      DSNAME=OBJECT,DISP=(NEW,PASS),UNIT=2314,
//      SPACE=(CYL,(2,1,1))
//FTH.SYSIN       DD      *

          Fortran source code

/*

//      EXEC      PL1CLG

//PL1.SYSIN       DD      *

          PL/I source code

/*

//EDT.SYSLIB      DD      DSNAME=SYS1.FORTLIB,DISP=SHR

//                DD      DSNAME=SYS1.PL1LIB,DISP=SHR

†→

//EDT.STEPONE      DD      DSNAME=OBJECT,DISP=(OLD,DELETE)

//EDT.SYSIN        DD      *

      INCLUDE STEPONE

      ENTRY IHENTRY

/*

//GO.ddname       DD      cards as needed (See Appendix B and A5)

//GO.SYSIN        DD      *

          Input data cards

/*
```

---

†→ Indicates location for a SYSLMOD DD card referenced in Appendix A7a.

APPENDIX A7: Storing Programs on the Data Cell from Source Code

a. Complete Programs in Executable Form

The deck structures are as described in Appendices A6 and A8b, but a SYSLMOD DD card must be inserted in the locations indicated by arrows (→). The (member) name of the program is given in parentheses after the dataset name in one of the following forms:

To Create a New PDS (Partitioned Data Set)

```
//EDT.SYSLMOD      DD      DSNAME=APSLIB(member name),
//     UNIT=2321,DISP=(NEW,KEEP),VOLUME=SER=CELLBH,
//     SPACE=(CYL,(25,5,2),,CONTIG),DCB=(RECFM=U,BLOCKSIZE=1024)
```

To Add to an Existing PDS, APSLIB

```
//EDT.SYSLMOD      DD      DSNAME=APSLIB(member name),DISP=SHR
```

APPENDIX A7 (Cont'd.):

b. <u>Incomplete Programs for Combining with User-coded Fortran Routines</u>

Note that there is no reference to the Fortran system routines (SYSL.FORTLIB) at this stage since further link-editing will be required for the user-coded FORT routine as in Appendix A8b. The following deck structure would be used for storing SEARCH2 on the data cell. The first jobstep would be omitted for the retrieval programs and for the FUNCTION program.

```
//      EXEC      FTHCP

//FTH.SYSLIN      DD      DSNAME=OBJECT,DISP=(NEW,PASS),UNIT=2314,
//      SPACE=(CYCL,(2,1,1))
//FTH.SYSIN       DD      *

        Fortran source code

/*

//      EXEC      PL1CLP,PARM.EDT=(LET,LIST,MAP)

//PL1.SYSIN       DD      *

        PL/I source code

/*

//EDT.SYSLMOD     DD      DSNAME=APSLIB(SEARCH2),DISP=SHR

//EDT.STEPONE     DD      DSNAME=OBJECT,DISP=(OLD,DELETE)

//EDT.SYSIN       DD      *

    INCLUDE STEPONE

    ENTRY IHENTRY

/*
```

APPENDIX A8: Using Programs from the Data Cell

a. Complete Programs†

Job and Accounting Cards

```
/*SETUP      cards as needed (See Appendix B and A5)

//JOBLIB     DD     DSNAME=APSLIB,DISP=SHR

//GO   EXEC     PGM=name of program (=member of PDS)

//GO.ddname     DD     cards as needed (See Appendix B and A5)

//GO.FT06F001     DD     card (See Appendix A5 for specifications)
//GO.SYSPRINT     DD     card (See Appendix A5 for specifications)
//GO.SYSIN     DD     *

        Input data to PL/I program

/*
```

† Note that SEARCH2 may be considered a complete program in this context if the FORT option is not specified.

APPENDIX A8 (Cont'd.):

b. Combining an Incomplete Program with a User-coded Fortran Routine

Job and Accounting Cards

```
/*SETUP      cards as needed (See Appendix B and A5)
//     EXEC     FTHCLG
//FTH.SYSIN     DD     *
        Fortran source code for Fortran subroutine
/*
†→
//EDT.APICSL     DD     DSNAME=APSLIB,DISP=SHR
//EDT.SYSIN      DD     *
    INCLUDE APICSL (name of incomplete program, e.g.: FORTRNK)
    ENTRY IHENTRY
/*

//GO.FT05F001     DD     DDNAME=MYSIS
//GO.ddname     DD     cards as needed (See Appendix B and A5)

//GO.SYSPRINT     DD     card (See Appendix A5 for specifications)
//GO.SYSIN        DD     *
        Input data for PL/I program
/*
```

---

†→ Indicates location for a SYSLMOD DD card referenced in Appendix A7a

APPENDIX A9: Deck Structures to Utilize Data on Master Tapes

a. One Jobstep (SEARCH2 with FORT Option Specified)

Job and Accounting Cards with CLASS=B and REGION=204K

```
/*SETUP      DDNAME=MASTER  (See Appendix A5.o)
/*SETUP      DDNAME=OUTPUT  (See Appendix A5.n)
//     EXEC      FTHCLG
//FTH.SYSIN      DD      *
        Fortran source code for FORT routine
/*
//EDT.APICSL      DD      DSNAME=APSLIB,DISP=SHR
//EDT.SYSIN       DD      *
    INCLUDE APICSL(SEARCH2)
    ENTRY IHENTRY
/*
//GO.FT05F001      DD      DDNAME=MYSIS
//GO.MASTER        DD
//GO.OUTPUT        DD      See Appendix A5 for specifications
//GO.CARDS         DD
//GO.SYSPRINT      DD
//GO.SYSIN         DD      *
        Input for SEARCH2 with OPTLIST='FILE,FORT';
/*
```

APPENDIX A9 (Cont'd.):

b. <u>Two Jobsteps to Search and Operate on Retrieved Data</u>

Job and Accounting Card

```
/*SETUP     DDNAME=MASTER  (See Appendix A5.o)
/*SETUP     DDNAME=OUTPUT  (See Appendix A5.n)
/*SETUP     DDNAME=PLOTTAPE  (See Appendix A5.ℓ)
//JOBLIB     DD     DSNAME=APSLIB,DISP=SHR
//GO     EXEC     PGM=SEARCH1
//GO.SYSPRINT     DD
//GO.FT06F001     DD
//GO.OUTPUT       DD
//GO.MASTER       DD
//GO.CARDS        DD
//GO.SCRATCH      DD
//GO.SYSIN        DD     *
```

See Appendix A5 for specifications

Search input

```
/*
//GO     EXEC     PGM=CITYMAP
//GO.OUTPUT     DD     See Appendix A5.g,DISP=(OLD,KEEP)
//GO.PLOTTAPE   DD     DSNAME=PLOT780,DISP=(NEW,KEEP),
//     UNIT=(2400-2,,DEFER),LABEL=(,NL)
//GO.SYSPRINT     DD     card  (See Appendix A5 for specifications)
//GO.FT06F001     DD     card  (See Appendix A5 for specifications)
//GO.SYSIN     DD     *
```

Input data for the FORTRNK retrieval program (See Appendix B)

```
/*
```

APPENDIX A10: Maintaining the APICS Library on the Data Cell

a. Generating a Backup Copy on Tape

```
/*SETUP     DDNAME=DD2,DEVICE=2400-9,ID=(075816,RING,SAVE,SL)
//     EXEC     PGM=IEHMOVE
//SYSPRINT     DD     SYSOUT=A
//DD1     DD     DISP=OLD,UNIT=2321,VOLUME=SER=CELLBH
//DD2     DD     UNIT=(2400,,DEFER),DISP=(NEW,KEEP),
//        VOLUME=(,RETAIN,SER=(075816)),
//        DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSUT1     DD     UNIT=2321,VOLUME=SER=CELLBH,DISP=OLD
//SYSIN      DD     *
    COPY   PDS=APSLIB,FROM=2321=CELLBH,TO=2400=(075816,1)
/*
```

APPENDIX A10 (Cont'd.):

b. Restoring the Data Set from Tape (Scratch, Save, Copy)

SCRATCH old dataset

```
//      EXEC      PGM=IEHPROGM
//SYSPRINT      DD      SYSOUT=A
//DD2           DD      UNIT=2321,VOLUME=SER=CELLBH,DISP=OLD
//SYSIN         DD      *
   SCRATCH      DSNAME=APSLIB,VOL=2321=CELLBH
/*
```

SAVE space

```
//      EXEC      PGM=IEHPROGM
//SYSPRINT      DD      SYSOUT=A
//APICSL        DD      DSNAME=APSLIB,UNIT=2321,
//              DISP=(NEW,KEEP),VOLUME=SER=CELLBH,
//              SPACE=(CYL,(25,5,2),,CONTIG),DCB=(RECFM=U,BLKSIZE=1024)
//SYSIN         DD      *
/*
```

COPY from tape

```
/*SETUP     DDNAME=DD2,DEVICE=2400-9,ID=(075816,NORING,SAVE,SL)
//      EXEC      PGM=IEHMOVE
//SYSPRINT      DD      SYSOUT=A
//DD1     DD      UNIT=2321,DISP=SHR,VOL=SER=CELLBH
//DD2     DD      UNIT=(2400,,DEFER),DISP=(OLD,KEEP),VOL=SER=075816
//SYSUT1      DD      UNIT=2314,VOL=SER=SCRTWO,DISP=OLD
//SYSIN       DD      *
   COPY DSNAME=APSLIB,FROM=2400=(075816,1),TO=2321=CELLBH
/*
```

APPENDIX A10 (Cont'd.)

c. Cataloguing and Listing the PDS APSLIB

The dataset is catalogued using IEHPROGM with the following card included with the SYSIN data:

```
CATLG     DSNAME=APSLIB,VOL=2321=CELLBH
```

To list the members of the dataset

```
//      EXEC      PGM=IEHLIST,COND=EVEN
//SYSPRINT      DD      SYSOUT=A
//DD2           DD      UNIT=2321,VOLUME=SER=CELLBH,DISP=OLD
//SYSIN         DD      *
   LISTPDS VOL=2321=CELLBH,DSNAME=APSLIB
/*
```

APICS USER MANUAL

APPENDIX B

SPECIFICATIONS FOR USING THE CODES IN THE APICS SYSTEM
ONE PAGE SUMMARY FOR EACH

Appendix B: Specifications for Using the Individual Codes in the APICS System (One Page Summary for Each Code)

The programs in the APICS system are listed below with the numbers corresponding to the file number of the source programs on magnetic tape. Note that the JCL DD card specifications in the deck structures provided for using each program assume that programs in the APICS system have been pre-stored on the data cell in a dataset named APSLIB. Furthermore, this dataset is assumed to have been catalogued. Otherwise, the JCL DD card specifications must include unit and volume serial parameters; e.g.: UNIT=2321 and VOLUME=SER=CELLBH.

1.	CREATE5	General Data Input Program
2.	CREATE9	Calendar Program
3.	CREATE3	Air Quality Data
4.	LITTLEC1	Preprocessor for Meteorological Data
5.	CREATE1	Meteorology Data
6.	TURSTAB	Functions of File Parameters (Turner Stability Example)
7.	MASTER2	Specialized Data Input Program (Mixing Height Example)
8.	PLANTSIM	Plant Emission Simulation Program
9.	MERGE1	Merge Program for Tapes with ∿ Same Dates
10.	MERGE2	Merge Program for Concatenating Two Tapes
11.	MERGE3	Merge Program for Tapes with Overlapping Parameters
12.	SUBFIL	Program to Extract a Subset of a Master Tape
13.	FUNCTION	Functions of Parameters on a Master Tape
14.	SEARCH1	Search of Master Tape for Specified Information
15.	SEARCH2	Search of Master Tape with Option to Pass Information to FORT
16.	SEARCH3	Search of Master Tape Omitting Missing Data
17.	FORTRN	Retrieval of Search Output to Pass to a FORT Routine
18.	FORTRNS	Retrieval of Search Output Sequentially and Pass n at a Time
19.	FORTRNK	Retrieval of Search Output and Pass n at a Time to a FORT Routin
20.	PRINT	Data Display Program
21.	TIMESER	Time Series Plotting Program (YOLYPLOT)
22.	CITYMAP	Mesometeorological Mapping Program
23.	IHISTJJ	Printed Graphical Displays (JJPLOT and HIST)
24.	REGRES	Linear Regression Analysis
25.	STEPMDA	Stepwise Multiple Discriminant Analysis
26.	TABPRED	Tabulation Prediction Program
27.	PUFF	Atmospheric Diffusion Calculations Dispersion Model

APPENDIX B1: CREATE5 (Ref. Section 3.1)

Description: General Data Input Program to Create a Master Tape.

Tape Input: SOURCE contains parameter values (may follow SYSIN).

Tape Output: NEWFILE(MFF) contains information in the Master File Format.

Printed Output: Summary of Newfile header information, i.e.: dates, station and parameter names followed by a listing of the parameter values as they are output.

Running time: Short

Class & Region: A(92K)

Card Input: SYSIN: dates, station name, number of parameters, and names.

Built-In Values: None - limited to one station.

Programming Notes: PL/I program calls a Fortran printing routine TWODPT.

Operating Procedures: Prestore the program (App. A6b & A7a) and execute as in App. A8a.

Deck Structure:

```
/*SETUP  DDNAME=NEWFILE,DEVICE=2400-9,ID=(,RING,SAVE,SL)

//JOBLIB  DD  DSNAME=APSLIB,DISP=SHR

//GO      EXEC     PGM=CREATE5

//GO.ddname  DD  cards for SYSPRINT and FT06F001 (See App. A5)

//GO.NEWFILE DD  DSNAME=TESTFILE,UNIT=TAPE9TRK,

//            DISP=(NEW,KEEP),DCB=(RECFM=VB,BLKSIZE=5000)

//GO.SYSIN   DD    *

Input data cards for SYSIN dataset

//GO.SOURCE  DD    *

Input data cards for SOURCE dataset

/*
```

Note: Underlined items should be changed from run to run.

APPENDIX B2: CREATE9 (Ref. Section 3.2)

Description: Calendar Program to Create a Master Tape.

Tape Input: None

Tape Output: NEWFILE(MFF) contains 6 parameters (See Table II: CALENDAR station)

Printed Output: Summary of header label information, i.e.: dates, station, and parameter names followed by the date for each day as the data is processed.

Running Time: 1½ minutes per year.

Class & Region: A(80K)

Card Input: SYSIN: Title card and a card for each day of each year giving holiday key, day of week, and Julian day.

Built-In Values: Starting date, ending date, number of days per month.

Programming Notes: PL/I program calls a Fortran printing routine TWODPT.

Operating Procedures: Compile the program for each use as in App. A6b.

Deck Structure:

```
/*SETUP  DDNAME=NEWFILE, DEVICE=2400-9,ID=(,RING,SAVE,SL)
//       EXEC  FTHCP   )
           :            }  Include cards for Fortran and PL/I steps exactly as
//       EXEC  PL1CLG  }  shown in App. A6b
           :            )
//GO.ddname   DD   card for FT06F001 (See App. A5).
//GO.NEWFILE  DD   DSNAME=CALENDAR,UNIT=TAPE9TRK,DISP=(NEW,KEEP),
//            DCB=(RECFM=VB,BLKSIZE=5000)
//GO.SYSIN    DD    *
Input data cards
/*
```

Note: Underlined items should be changed from run to run.

APPENDIX B3: CREATE3 (Ref. Section 3.3)

Description: Air Quality Data Program to Create a Master Tape.

Tape Input: SOURCE contains data from TAM recording stations (See Table IV)

Tape Output: NEWFILE(MFF) contains TAM data (See Table II: TAM stations)

Printed Output: Summary of header label information, i.e.: dates, station, and parameter names. Also tables of all parameter values are printed as the data is processed.

Running Time: 12 minutes per year

Class & Region: C(122K)

Card Input: SYSIN: Title card to identify the printed output.

Built-In Values: Starting date, and ending date. Assumes 8 TAM stations with 3 parameters each. Converts missing values of 777,888,999, etc. to 10^{20}.

Programming Notes: PL/I program calls a Fortran printing routine TWODPT. Includes a special arctangent routine. (May concatenate SOURCE tapes.)

Operating Procedures: Compile the program for each use as in App. A6b.

Deck Structure:

```
/*SETUP  DDNAME=SOURCE,DEVICE=2400-7,ID=(072307,NORING,SAVE,SL)

/*SETUP  DDNAME=NEWFILE,DEVICE=2400-9,ID =(,RING,SAVE,SL)

//       EXEC  FTHCP      )
          :               |
//       EXEC  PL1CLG     |   Include cards for Fortran and PL/I steps as in
          :               |   App. A6b.
                          )
//GO.SOURCE   DD   DSNAME=RAWTAM,UNIT=TAPE7TRK,DISP=(OLD,KEEP),

//          LABEL=(1,NL),DCB=(RECFM=F,BLKSIZE=7872,TRTCH=ET,DEN=1)

//GO.NEWFILE  DD   DSNAME=TAMDATA,UNIT=TAPE9TRK,DISP(NEW,PASS),

//            DCB=(RECFM=V,BLKSIZE=5000)

//GO.SYSIN    DD   *

TAM  NETWORK RECEPTOR DATA

/*
```

Note: Underlined items should be changed from run to run.

APPENDIX B4: LITTLEC1 (Ref. Section 3.4)

Description: Preprocessor for Meteorological Data to Obtain a Subset.

Tape Input: SOURCE: NWRC Tape of Airways Surface Observations (See Table V)

Tape Output: NEWFILE contains a selected portion of SOURCE in the same format.

Printed Output: The date for each day as the data is transferred to NEWFILE.

Running Time: 1 minute for 6 months.

Class & Region: A(42K)

Card Input; None

Built-In Values: Starting date, ending date, WBAN station number

Programming Notes: PL/I program

Operating Procedures: Compile the program for each use as in App. A6b.

Deck Structure:

```
/*SETUP  DDNAME=SOURCE,DEVICE=2400-7,ID=(072622,NORING,SAVE,NL)
/*SETUP  DDNAME=NEWFILE,DEVICE=2400-7,ID=(,RING,SAVE,NL)
//      EXEC   PL1CLG
//PL1.SYSIN   DD   *
PL/I source code
/*
//GO.SOURCE   DD   LABEL=(,NL),UNIT=TAPE7TRK,DISP=(OLD,KEEP),
//           DCB=(DEN=1,TRTCH=ET,RECFM=F,BLKSIZE=495)
//GO.NEWFILE  DD  same specifications as for SOURCE
```

Note: Underlined items should be changed from run to run.

APPENDIX B5: CREATE1 (Ref. Section 3.4)

Description: Meteorology Data Conversion to Master File Format

Tape Input: SOURCE: NWRC Tape of Airways Surface Observations and/or LITTLEC1 output tape.

Tape Output: NEWFILE(MFF) contains airport weather data (See Tables II and III)

Printed Output: Summary of header label information, i.e.: dates, station, and parameter names followed by table headings, values, and the date for each day as the data is processed.

Running Time: 4 minutes per year.

Class & Region: C(120K)

Card Input: Airport Weather Station Name in cols. 1-8 (one card only)

Built-In Values: Starting date, ending date.

Programming Notes: PL/I program calls Fortran printing routine TWODPR. Processes 36 parameters for one airport weather station. (May concatenate source tapes.)

Operating Procedures: Compile the program for each use as in App. A6b.

Deck Structure:

```
/*SETUP  DDNAME=SOURCE,DEVICE=2400-7,ID=(072347,NORING,SAVE,NL)

/*SETUP  DDNAME=NEWFILE,DEVICE=2400-9,ID=(,RING,SAVE,SL)

//      EXEC   FTHCP   )
         :             |
//      EXEC   PL1CLG  |  Include cards for Fortran and Pl/I steps as
         :             |  in App. A6b.

//GO.ddname   DD   card for FT06F001  (See App. A5)

//GO.SOURCE   DD   LABEL=(1,NL),UNIT=TAPE7TRK,DISP=(OLD,KEEP),

//           DCB=(,DEN=1,TRTCH=ET,RECFM=F,BLKSIZE=495)

//GO.NEWFILE  DD  DSNAME=MIDWAY,UNIT=TAPE9TRK,DISP=(NEW,KEEP),

//           DCB=(RECFM=V,BLKSIZE=5000)

//GO.SYSIN    DD   *

MIDWAY    AIRPORT WEATHER DATA

/*
```

APPENDIX B6: TURSTAB (Ref. Section 3.5)

Description: Functions of File Parameters (Turner Stability Example).

Tape Input: MASTER(MFF) containing airport weather data.

Tape Output: SUBFILE(MFF): Master tape of stabilities (See end of Tables II & III).

Printed Output: Summary of the master tape stations and parameters. Also input data specifications followed by the date for each day as the data is processed.

Running Time: 1½ min. per year.

Class & Region: B(194K)

Card Input: Search-like data as described in Section 4.3 specifying 3 input weather parameters and 2 output parameters (Turner A and B) for a station STAB.

Built-In Values: Calculates 2 stability parameters based on 3 weather items.

Programming Notes: Specially coded FORTRN (Fortran) routine called by the general purpose FUNCTION (PL/I) program.

Operating Procedures: Prestore the program (App. A7a & A8b) and execute as in App. A8a.

Deck Structure:

```
/*SETUP  DDNAME=MASTER,DEVICE=2400-9,ID=(072067,NORING,SAVE,SL)

/*SETUP  DDNAME=SUBFILE,DEVICE=2400-9,ID=(,RING,SAVE,SL)

//JOBLIB  DD  DSNAME=APSLIB,DISP=SHR

//GO  EXEC  PGM=TURSTAB

//GO.ddname  DD  cards for FT06F001,SYSPRINT,&CARDS (See App. A5)

//GO.MASTER  DD  DSNAME=MIDWAY,UNIT=TAPE9TRK,DISP=(OLD,KEEP)

//GO.SUBFILE DD  DSNAME=TURNER,UNIT=TAPE9TRK,DISP=(NEW,KEEP),

//           DCB=(RECFM=VB,BLKSIZE=10000)

//GO.SYSIN   DD   *

Input data cards

/*
```

Note: Underline items should be changed from run to run.

APPENDIX B7: MASTER2 (Ref. Section 3.6)

Description: Specialized Data Input Program (Mixing Height Example).

Tape Input: None (or can read SOURCE data from tape instead of cards).

Tape Output: NEWFILE(MFF) contains information in the Master File Format.

Printed Output: Summary of Newfile header information, i.e.: dates, station and parameter names, followed by the date for each day as the data is processed.

Running Time: 1 minute for two years worth of parameters.

Class & Region: A(100k)

Card Input: SOURCE (or None if the data is read from tape): The station name on card no. 1. followed by a blank card and hourly parameter values (12 per card) for a single parameter.

Built-In Values: Starting date, ending date, parameter names of MIX-HT and KEY. Generates zero values for KEY. Input values of 9999 (and 4021) are considered missing and converted to 10^{20}.

Programming Notes: PL/I program calls a Fortran printing routine TWODPT. Processes values for one parameter and generates two parameters for one station.

Operating Procedures: Compile the program for each use as in App. A6b.

Deck Structure:

```
/*SETUP   DDNAME=NEWFILE,DEVICE=2400-9,ID=(,RING,SAVE,SL)
//        EXEC   FTHCP   )
          :              )
//        EXEC   PL1CLG  )   Include cards for Fortran and PL/I steps as
          :              )   in App. A6b.
//GO.SYSIN   DD   DDNAME=MYSIS
//GO.ddname  DD   cards for SYSPRINT and FT06F001 (See App. A5)
//GO.NEWFILE DD   DSNAME=MIXINGHT,UNIT=TAPE9TRK,DISP=(NEW,KEEP),
//            DCB=(RECFM=VB,BLKSIZE=5000)
//GO.SOURCE  DD   *
NORCO
blank card
 10166 1  9999  9999  1477  1638  1914  3543  5643. . .
   :
/*
```

Note: Underlined items should be changed from r

APPENDIX B8: PLANTSIM (Ref. Section 3.7)

Description: Plant Emission Simulation Program

Tape Input: MASTER(MFF) containing Calendar, Airport, and Power Plant Stations.

Tape Output: OUTPUT: contains hourly emissions for PUFF program

Printed Output: Summary of Master & Output header labels, followed by a listing of the input cards. Dates and calculated hourly emissions are printed as the data is processed.

Running Time: 2 minutes for a sample case.

Class & Region: B(204K)

Card Input: Search-like input (See Section 3.7)

Built-In Values: SO_2 content of coal and other quantities as described in Section 3.7.

Programming Notes: PL/I program calls a Fortran printing routine TWODPT.

Operating Procedures: Prestore the program (App. A6b & A7a) and execute as in App. A8

Deck Structure:

```
/*SETUP   DDNAME=MASTER,DEVICE=2400-9,ID=(073061,NORING,SAVE,SL)

/*SETUP   DDNAME=OUTPUT,DEVICE=2400-9,ID=(,RING,SAVE,SL)

//JOBLIB  DD   DSNAME=APSLIB,DISP=SHR

//GO      EXEC   PGM=PLANTSIM

//GO.ddname   DD   cards for SYSPRINT,FT06F001,andCARDS (See App. A5)

//GO.MASTER   DD   DSNAME=CECOMET,UNIT=TAPE9TRK,DISP=(OLD,KEEP)

//GO.OUTPUT   DD   DSNAME=PSOUTPUT,UNIT=TAPE9TRK,DISP=(NEW,PASS),

//            DCB=(RECFM=VB,BLKSIZE=10000)

//GO.SYSIN    DD   *

Title card to identify the output from the search-like run

Date limits and specifications for Utilities and Sources

/*
```

Note: Underlined items should be changed from run to run.

APPENDIX B9: MERGE1 (Ref. Section 4.1.1)

Description: Merge Program for Tapes with ∿ Same Dates.

Tape Input: OLDFIL1(MFF) and OLDFIL2(MFF): Two master tapes.

Tape Output: NEWFILE(MFF) contains all the information on the old master tapes up to the cutoff date for OLDFIL1.

Printed Output: Summary of the header labels for all 3 masters, i.e.: dates, station, and parameter names, followed by the date for each day as the date is processed.

Running Time: 2 minutes per year with 200 parameters.

Class & Region: B(142K)

Card Input: None

Built-In Values: None-generates values of 10^{20} for missing data at the end of OLDFIL2.

Programming Notes: PL/I program

Operating Procedures: Prestore the program (App. A6a & A7a) and execute as in App. A8a.

Deck Structure:

```
/*SETUP     DD     DDNAME=OLDFIL1,DEVICE=2400-9,ID=(073064,NORING,SAVE,SL)

/*SETUP     DD     DDNAME=OLDFIL2,DEVICE=2400-9,ID=(073947,NORING,SAVE,SL)

//SETUP     DD     DDNAME=NEWFILE,DEVICE=2400-9,ID=(,RING,SAVE,SL)

//JOBLIB    DD     DSNAME=APSLIB,DISP=SHR

//GO        EXEC      PGM=MERGE1

//GO.ddname  DD   card for SYSPRINT (See App.A5)

//GO.OLDFIL1 DD   DSNAME=TAMDATA,UNIT=TAPE9TRK,DISP=(OLD,KEEP)

//GO.OLDFIL2 DD   DSNAME=CALENDAR,UNIT=TAPE9TRK,DISP=(OLD,KEEP)

//GO.NEWFILE DD   DSNAME=TAMCAL,UNIT=TAPE9TRK,DISP=(NEW,KEEP),

//            DCB=(RECFM=V,BLKSIZE=20000)
```

Note: Underlined items should be changed from run to run.

APPENDIX B10: MERGE2 (Ref. Section 4.1.2)

Description: Merge Program for Concatenating Two Tapes.

Tape Input: OLDFIL1(MFF) and OLDFIL2(MFF): Two master tapes.

Tape Output: NEWFILE(MFF) contains all the information on the old master tapes.

Printed Output: Summary of the header labels for all 3 masters, i.e.: dates, station, and parameter names, followed by the date for each day as the data is processed.

Running Time: 2 minutes per year with 200 parameters.

Class & Region: B(150K)

Card Input: None

Built-In Values: None

Programming Notes: PL/I program

Operating Procedures: Prestore the program (App. A6a & A7a) and execute as in App. A8a.

Deck Structure:

```
/*SETUP    DD    OLDFIL1,DEVICE=2400-9,ID=(071032,NORING,SAVE,SL)

/*SETUP    DD    OLDFIL2,DEVICE=2400-9,ID=(072026,NORING,SAVE,SL)

/*SETUP    DD    NEWFILE,DEVICE=2400-9,ID=(,RING,SAVE,SL)

//JOBLIB   DD    DSNAME=APSLIB,SHR

//GO       EXEC    PGM=MERGE2

//GO.ddname   DD   card for SYSPRINT (see App. A5)

//GO.OLDFIL1  DD   DSNAME=MIDWAY68,UNIT=TAPE9TRK,DISP=(OLD,KEEP)

//GO.OLDFIL2  DD   DSNAME=MIDWAY69,UNIT=TAPE9TRK,DISP=(OLD,KEEP)

//GO.NEWFILE  DD   DSNAME=MIDWAY,UNIT=TAPE9TRK,DISP=(NEW,KEEP),

//          DCB=(RECFM=V,BLKSIZE=20000)
```

Note: Underlined items should be changed from run to run.

APPENDIX B11: MERGE3 (Ref. Section 4.1.3)

Description: Merge Program for Tapes with Overlapping Parameters.

Tape Input: OLDFIL1(MFF) and OLDFIL2(MFF): Two master tapes.

Tape Output: NEWFILE(MFF) contains all the information on the old master tapes.

Printed Output: Summary of the information on all 3 masters followed by the date for each day as the data is processed. Also final parameter values for all tapes.

Running Time: 3 minutes per year with 200 parameters.

Class & Region: C(250K=50K+100K × # of parameters)

Card Input: None

Built-In Values: None-generates values of 10^{20} for missing data at the beginning or end of either of the tapes. If values appear on both tapes, the OLDFIL2 value is used.

Programming Notes: PL/I program.

Operating Procedures: Prestore the program (App. A6a & A7a) and execute as in App. A8a.

Deck Structure:

```
/*SETUP     DDNAME=OLDFIL1,DEVICE=2400-9,ID=(072138,NORING,SAVE,SL)

/*SETUP     DDNAME=OLDFIL2,DEVICE=2400-9,ID=(073067,NORING,SAVE,SL)

/*SETUP     DDNAME=NEWFILE,DEVICE=2400-9,ID=(,RING,SAVE,SL)

//JOBLIB    DD   DSNAME=APSLIB,DISP=SHR

//GO        EXEC       PGM=MERGE3

//GO.ddname     DD    card for SYSPRINT (See App. A5)

//GO.OLDFIL1    DD    DSNAME=TAMCAL,UNIT=TAPE9TRK,DISP=(OLD,KEEP)

//GO.OLDFIL2    DD    DSNAME=MIDWAY,UNIT=TAPE9TRK,DISP=(OLD,KEEP)

//GO.NEWFILE    DD    DSNAME=METMAP,UNIT=TAPE9TRK,DISP=(NEW,KEEP),

//              DCB=(RECFM=V,BLKSIZE=20000)
```

Note: Underlined items should be changed from run to run.

APPENDIX B12: SUBFIL (Ref. Section 4.2)

Description: Program to Extract a Subset of a Master File.

Tape Input: MASTER(MFF) must include the items specified in the input.

Tape Output: SUBFILE(MFF) contains specified parameters for same dates as MASTER.

Printed Output: Summary of header labels for both the input and output master tapes followed by the date for each day as the data is processed.

Running Time: 3 minutes per year with 200 parameters.

Class & Region: C(190K)

Card Input: Search-like data: Title card and output parameters (with delays) for OUT stations followed by END and STOP cards.

Built-In Values: None-limit of 100 output parameters.

Programming Notes: PL/I program

Operating Procedures: Prestore/the program (App. A6a & A7a) and execute as in App. A8a.

Deck Structure:

```
/*SETUP     DDNAME=MASTER,DEVICE=2400-9,ID=(073090,NORING,SAVE,SL)

/*SETUP     DDNAME=SUBFILE,DEVICE=2400-9,ID=(,RING,SAVE,SL)

//JOBLIB    DD      DSNAME=APSLIB,DISP=SHR

//GO        EXEC    PGM=SUBFIL

//GO.ddname      DD      cards for SYSPRINT and CARDS (See App.A5)

//GO.MASTER      DD      DSNAME=METMAP,UNIT=TAPE9TRK,DISP=(OLD,KEEP)

//GO.SUBFILE     DD      DSNAME=MIDWAY,UNIT=TAPE9TRK,DISP=(NEW,KEEP),

//                DCB=(RECFM=V,BLKSIZE=10000)

//GO.SYSIN       DD    *

Input data

/*
```

Note: Underlined items should be changed from run to run.

APPENDIX B13: FUNCTION (Ref. Section 4.3)

Description: Function of Parameters on Master Tapes.

Tape Input: MASTER(MFF) must include the items specified in the input.

Tape Output: SUBFILE(MFF) contains calculated values for same dates as MASTER.

Printed Output: Summary of header labels for both the input and output master tapes followed by the date for each day as the data is processed.

Running Time: $\gtrsim 1\frac{1}{2}$ minutes per year with 200 parameters on MASTER.

Class & Region: B(194K): depends on the function evaluation procedure.

Card Input: Search-like data (specifications for input and output parameters must be consistent with the FORTRN function evaluation routine): Title card, parameter specifications (with delays) for IN stations, and output parameters for the FUNCTION station, followed by END and STOP cards.

Built-In Values: None-limit of 100 output parameters.

Programming Notes: PL/I program calls a user-coded FORTRN routine with 7 arguments (NYEAR,NMONTH,NDAY,NIN,PARIN,NOUT,PAROUT)

Operating Procedures: Prestore incomplete program (App. A6a with EDT options and a SYSLMOD card as in App. A7b). Compile FORTRN routine and access stored FUNCTION as in App. A8b.

Deck Structure:

```
/*SETUP     DDNAME=MASTER,DEVICE=2400-9,ID=(072027,NORING,SAVE,SL)

/*SETUP     DDNAME=SUBFILE,DEVICE=2400-9,ID=(,RING,SAVE,SL)

//          EXEC     FTHCLG . . . Include all cards indicated in App. A8b,
```

specifying INCLUDE APICSL(FUNCTION) and provide DD cards as follows

```
//GO.FT05F001        DD      DDNAME=MYSIS

//GO.ddname          DD      cards for FT06F001,SYSPRINT,and CARDS (See App. A5)

//GO.MASTER          DD      DSNAME=MIDWAY,UNIT=TAPE9TRK,DISP=(OLD,KEEP)

//GO.SUBFILE         DD      DSNAME=TURNER,UNIT=TAPE9TRK,DISP=(NEW,KEEP),

//                   DCB=(RECFM=VB,BLKSIZE=10000)

//GO.SYSIN           DD      *

Input data

/*
```

Note: Underlined items should be changed from run to run.

APPENDIX B14: SEARCH1 (Ref. Section 5.2)

Description: Search of Master Tape for Specified Information.

Tape Input. MASTER(MFF): must include all items specified in the input.

Tape Output: OUTPUT tape contains array of selected parameter values.

Printed Output: Summary of station and parameter information on input and output tapes; and, if specified, tables of parameter values in the output array.

Running Time: 1 minute per year for 6 parameters out of 100 on the master tape

Class & Region: A(204K): allow additional space if PRINT is specified.

Card Input: Search data: Title card, PRINT option (YES or NO), specifications for date limits, hour limits, band limits for MET station parameters, and output parameters (with delays) for OUT stations, followed by END and STOP cards.

Built-In Values: None-limit of 100 output parameters.

Programming Notes: PL/I program calls a Fortran printing routine TWODPT.

Operating Procedures: Prestore the program (App. A6b & A7a) and execute as in App. A8a.

Deck Structure:

```
/*SETUP     DDNAME=MASTER,DEVICE=2400-9,ID=(072166,NORING,SAVE,SL)

/*SETUP     DDNAME=OUTPUT,DEVICE=2400-9,ID=(,RING,SAVE,SL)

//JOBLIB    DD     DSNAME=APSLIB,DISP=SHR

//GO        EXEC   PGM=SEARCH1

//GO.ddname     DD    cards for FT06F001,SYSPRINT,CARDS,and SCRATCH (See App. A5)

//GO.MASTER     DD    DSNAME=TAMCALMD,UNIT=TAPE9TRK,DISP=(OLD,KEEP)

//GO.OUTPUT     DD    DSNAME=OUTPUT,UNIT=TAPE9TRK,DISP=(NEW,PASS),

//            DCB=(RECFM=VB,BLKSIZE=20000)

//GO.SYSIN      DD    *

Input data

/*
```

Note: U [illegible] should be changed from run to run.

APPENDIX B15: SEARCH2 (Ref. Section 5.3)

Description: Search of Master Tape with Option to Pass Array to FORT(LCS).

Tape Input: MASTER(MTF) must include all items specified in the input.

Tape Output: OUTPUT tape contains array of selected parameter values.

Printed Output: Summary of station and parameter information on input and output tapes; and, if specified, tables of parameter values in the output array.

Running Time: 1 minute per year for 6 parameters out of 100 on the master tape.

Class & Region: A(650K,1000K)

Card Input: Search data: title card. OPTLIST options (PRINT,FILE, and FORT), specifications for date limits, hour limits, band limits for MET station parameters, and output parameters (with delays) for OUT stations, followed by END and STOP cards.

Built-In Values: None-limit of 100 output parameters and 32767 observation times.

Programming Notes: PL/I program calls Fortran routines for printing (TWODPT) and for storage allocation (LCSSET and BPOINTER). Optionally calls a user-coded FORT routine indirectly through FORK via the PL/I to Fortran interface routine (FORTRAN).

Operating Procedures: Prestore the program (App. A6b & A7b) and execute as in A9a or as indicated below (without a FORT routine)

Deck Structure:

```
/*SETUP      DDNAME=MASTER,DEVICE=2400-9,ID=073090,NORING,SAVE,SL)

/*SETUP      DDNAME=OUTPUT,DEVICE=2400-9,ID=(,RING,SAVE,SL)

//          EXEC      FTHLG,PARM.EDT='LET,LIST,MAP'

//EDT.APICSL     DD     DSNAME=APSLIB,DISP=SHR

//EDT.SYSIN      DD     *

        INCLUDE APICSL(SEARCH2)

/*

//GO.ddname      DD      cards for FT06F001,SYSPRINT,CARDS,and OUTPUT (See App. A5)

//GO.MASTER      DD      DSNAME=METMAP8,UNIT=TAPE9TRK,DISP=(OLD,KEEP)

//GO.SYSIN       DD     *

Input data as desired with FILE and/or PRINT options specified

/*
```

Note: Underlined items should be changed from ru...

APPENDIX B16: SEARCH3 (Ref. Section 5.3)

Description: Search of Master Tape Omitting Missing Data.

Tape Input: MASTER(MFF): must include all items specified in the input.

Tape Output: OUTPUT tape contains arrays of selected parameter values.

Printed Output: Summary of station and parameter information on input and output tapes; and, if specified, tables of parameter values in the output array.

Running Time: 1 minute per year for 6 parameters out of 100 on the master tape.

Class & Region: A(204K): allow additional space if PRINT is specified.

Card Input: Search data: Title card, PRINT option (YES or NO), specifications for date limits, hour limits, band limits for MET station parameters, and output parameters (with delays) for OUT stations, followed by END and STOP cards.

Built-In Values: Limit of 100 output parameters for each hour. Values of 10^{20} are interpreted as missing data and no parameter values will be output for hours with any missing data.

Programming Notes: PL/I program calls a Fortran printing routine TWODPT.

Operating Procedures: Prestore the program (App. A6b & A7a) and execute as in App. A8a.

Deck Structure:

```
/*SETUP     DDNAME=MASTER,DEVICE=2400-9,ID=(072166,NORING,SAVE,SL)

/*SETUP     DDNAME=OUTPUT,DEVICE=2400-9,ID=(,RING,SAVE,SL)

//JOBLIB    DD      DSNAME=APSLIB,DISP=SHR

//GO        EXEC      PGM=SEARCH3

//GO.ddname       DD      cards for FT06F001,SYSPRINT,CARDS,and SCRATCH (See App. A5)

//GO.MASTER       DD      DSNAME=TAMCALMD,UNIT=TAPE9TRK,DISP=(OLD,KEEP)

//GO.OUTPUT       DD      DSNAME=OUTPUT,UNIT=TAPE9TRK,DISP=(NEW,KEEP),

//                DCB=(RECFM=VB,BLKSIZE=20000)

//GO.SYSIN        DD      *

Input data

/*
```

[illegible] are changed from run to run.

APPENDIX B17: FORTRN (Ref. Section 6.1)

Description: Retrieval of Search Output to Pass (to FORT) All at Once.

Tape Input: OUTPUT from a search.

Tape Output: None

Printed Output: Summary of stations and parameters for the retrieved data followed by output from the user-coded routines.

Running Time: $\gtrsim$1 minute per year for 24 parameters.

Class & Region: C($\gtrsim$124K)

Card Input: One or more Title Cards to identify the output arrays from the search(es) followed by a card with STOP in cols. 1-4.

Search Input: Any combination of parameters may be specified for some uses and others require specific ordered parameters.

Built-In Values: None

Programming Notes: PL/I program calls a user-coded FORT routine with arguments NPAR, NOBS, X, STANAM, PARNAM where X is dimensioned NOBS by NPAR.

Operating Procedures: Prestore incomplete program (App. A6a with EDT options and a SYSLMOD card specified as in App. A7b). Then compile the FORT routine and access FORTRN as in App. A8b (or A9b).

Deck Structure:

```
/*SETUP      DDNAME=OUTPUT,DEVICE=2400-9,ID=(073734,NORING,SAVE,SL)

//         EXEC      FTHCLG          and so forth . . .
```

Include all cards indicated in App. A8b, specifying

INCLUDE APICSL(FORTRN), and provide the additional DD card:

```
//GO.OUTPUT      DD      DSNAME=OUTPUT,UNIT=TAPE9TRK,DISP=(OLD,KEEP)
```

If desired, data may be input to the FORT routine by inserting it before the last /* card, preceded by the DD card:

```
//GO.MYSIS      DD       *
```

Note: Underlined items should be changed from run to run.

APPENDIX B18: FORTRNS (Ref. Section 6.2)

Description: Retrieval of Search Output Sequentially to Pass (to FORT) n Hours at a Time.

Tape Input: OUTPUT from a search.

Tape Output: None

Printed Output: Summary of stations and parameters for the retrieved data followed by output from the user-coded routines.

Running Time: $\gtrsim$1 minute per year for 24 parameters.

Class & Region: C($\gtrsim$124K)

Card Input: "n" and one or more Title Cards to identify the output arrays from the search(es) followed by a card with STOP in cols. 1-4. Title cards must be in order. (If n=99999 all the data from a search is passed at once.)

Built-In Values: Limit of 300 parameters.

Programming Notes: PL/I program calls a FORT routine via the FORTRAN interface routine (must concatenate a library containing the latter to SYSLIB when link-editing). Also calls FORT at the end of the search(es) with NOBS=0 after each search and -1 at the very end.

Operating Procedures: Prestore incomplete program (App. A6a with EDT options and and a SYSLMOD card specified as in App. A7b). Then compile the FORT routine and access FORTRN as in App. A8b (or A9b).

Deck Structure:

```
/*SETUP     DDNAME=OUTPUT,DEVICE=2400-9,ID=(073734,NORING,SAVE,SL)

//          EXEC     FTHCLG        and so forth . . .
```

Include all cards indicated in App. A8b, specifying

INCLUDE APICSL(FORTRNS), and provide the additional DD card:

```
//GO.OUTPUT     DD     DSNAME=OUTPUT,UNIT=TAPE9TRK,DISP=(OLD,KEEP)
```

If desired, data may be input to the FORT routine by inserting it before the last /* card, preceded by the DD card:

```
//GO.MYSIS     DD     *
```

Note: Underlined items should be changed from run to run.

APPENDIX B19: FORTRNK (Ref. Section 6.3)

Description: Retrieval of Search Output to Pass (to FORT) n Hours at a Time.

Tape Input: OUTPUT from a search.

Tape Output: None

Printed Output: Summary of stations and parameters for the retrieved data followed by output from the user-coded routines.

Running Time: $\gtrsim$1 minute per year for 24 parameters.

Class & Region: C($\gtrsim$124K)

Card Input: "n" and one or more Title Cards to identify the output arrays from the search(es) followed by a card with STOP in cols. 1-4.
(If n=99999 all the data from a search is passed at once.)

Built-In Values: Limit of 300 parameters.

Programming Notes: PL/I program calls a FORT routine via the FORTRAN interface routine (must concatenate a library containing the latter to SYSLIB when link-editing). Also calls FORT at the end of the search(es) with NOBS=0 after each search and -1 at the very end.

Operating Procedures: Prestore incomplete program (App. A6a with EDT options and a SYSLMOD card specified as in App. A7b. Then compile the FORT routine and access FORTRN as in App. A8b (or A9b).

Deck Structure:

```
/*SETUP     DDNAME=OUTPUT,DEVICE=2400-9,ID=(073734,NORING,SAVE,SL)

//          EXEC      FTHCLG        and so forth . . .
```

Include all cards indicated in App. A8b,specifying

INCLUDE APICSL (FORTRNK), and provide the additional DD card:

```
//GO.OUTPUT     DD     DSNAME=OUTPUT,UNIT=TAPE9TRK,DISP=(OLD,KEEP)
```

If desired, data may be input to the FORT routine by inserting it before the last /* card, preceded by the DD card:

```
//GO.MYSIS     DD     *
```

Note: Underlined items should be changed from run to run.

APPENDIX B20: PRINT (Ref. Section 7.1)

Description: Data Display Program to Print Selected Parameters Values.

Tape Input: MASTER(MFF) must include all items specified in the input.

Tape Output: None

Printed Output: Summary of stations and parameters on the master tape and the specified parameters for printing, followed by tables of values from the master tape.

Running Time: 3 minutes per year for 10 parameters out of 50 on the master tape.

Class & Region: A(142K)

Card Input: Search-like data: Title card, date limits, and output parameters (with delays) for OUT stations, followed by END and STOP cards.

Built-In Values: Limit of 100 output parameters for printing.

Programming Notes: PL/I program calls a Fortran printing routine TWODPT.

Operating Procedures: Prestore the program (App. A6b and A7a) and execute as in App. A8a.

Deck Structure:

```
/*SETUP      DDNAME=MASTER,DEVICE=2400-9,ID=(073740,NORING,SAVE,SL)

//JOBLIB     DD      DSNAME=APSLIB,DISP=SHR

//GO         EXEC       PGM=PRINT

//GO.ddname     DD     cards for SYSPRINT,FT06F001,CARDS,and SCRATCH (See App. A5)

//GO.MASTER     DD     DSNAME=CETAMET,UNIT=TAPE9TRK,DISP=(OLD,KEEP)

//GO.SYSIN     DD     *

Input data

/*
```

Note: Underlined items should be changed from run to run.

APPENDIX B21: TIMESER (Ref. Section 7.2)

Description: Time Series Plotting Program (YOLYPLOT)

Tape Input: OUTPUT from an appropriate search

Tape Output: PLOTTAPE for Calcomp plotting.

Printed Output: Summary of stations and parameters retrieved from a search.

Plotted Output: Time series plots with up to 4 plots above each other (length = 15x# of months).

Running Time: About 100 plots per minute (+10 seconds per month to search for 15 parameters)

Class & Region: B(272K)

Card Input: "n" (=99999) and a Title Card as was used for the search.

Search Input: Any combination of stations and parameters may be specified but the first NTAM(=8) items will be plotted over a range of 0 to 0.4 as is appropriate for SO_2 conc.

Built-In Values: NTAM, parameter range for first graphs, and X-axis scale in days.

Programming Notes: FORT routine called by a retrieval program FORTRNK or S, includes a special YOLYPLOT and DXDY, uses the Calcomp package (must concatenate a library containing the latter to SYSLIB when link-editing the program).

Operating Procedures: Combine with FORTRNK or S and prestore the program (App. A3b and A7a) and execute as in App. A8a. Also see App. A9b for executing jobs in two steps: SEARCH1 & TIMESER

Deck Structure:

```
/*SETUP     DDNAME=OUTPUT,DEVICE=2400-9,ID=(075021,NORING,SAVE,SL)

/*SETUP     DDNAME=PLOTTAPE,DEVICE=2400-7,ID=(CALCMT,RING,SAVE,NL)

//JOBLIB    DD     DSNAME=APSLIB,DISP=SHR

//GO        EXEC        PGM=TIMESER

//GO.ddname     DD     cards for PLOTTAPE,OUTPUT,SYSPRINT,& FT06F001 (See App. A5)

//GO.SYSIN     DD     *

    n in columns 1 to 5

    Title card to identify the output array from a search.

STOP

/*
```

APPENDIX B22: CITYMAP (Ref. Section 7.3)

Description: Mesometeorological Mapping Program.

Tape Input: OUTPUT from a search run with prescribed specifications.

Tape Output: PLOTTAPE for Calcomp plotting of a map every n^{th} hour.

Printed Output: Summary of search output and the value of n as each map is plotted.

Plotted Output: Calcomp maps (8½ × 11)

Running Time: 1 minute for 50 maps (+1½ minutes to search for 10 days in a year).

Class & Region: C(144K), or 240K if the first jobstep is the search.

Card Input: "n" and a Title card as was used for the search.

Search Input: Title card, dates, and parameter specifications.

Built-In Values: Number of stations and parameters (must be in order).

Programming Notes: FORT program called by a retrieval program FORTRNK or S, uses Calcomp package (must concatenate a library containing the latter to SYSLIB when link-editing the program).

Operating Procedures: Combine with FORTRNK or S and prestore the program (App. A8b and A7a) and execute as in App. A8a. Also see App. A9b for executing jobs in two steps: SEARCH1 & CITYMAP

Deck Structure:

```
/*SETUP      DDNAME=OUTPUT,DEVICE=2400-9,ID=(075021,NORING,SAVE,SL)

/*SETUP      DDNAME=PLOTTAPE,DEVICE=2400-7,ID=(CALCMP,RING,SAVE,NL)

//JOBLIB     DD      DSNAME=APSLIB,DISP=SHR

//GO         EXEC       PGM=CITYMAP

//GO.ddname        DD      cards for PLOTTAPE,OUTPUT,SYSPRINT,& FT06F001 (See App. A5)

//GO.SYSIN       DD      *

   n in columns 1 to 5

   Title card to identify the output array from a search.

STOP

/*
```

Note: Underlined items should be changed from run to run.

APPENDIX B23: IHISTJJ (Ref. Section 7.4)

Description: Printed Graphical Displays (JJPLOT and HIST)

Tape Input: OUTPUT from an appropriate search.

Tape Output: None

Printed Output: Summary of stations and parameters retrieved from a search followed by two-dimensional frequency distributions and histograms.

Running Time: 100 frequency distributions and associated histograms per minute (+10 seconds per month to search for 15 parameters).

Class & Region: B(174K)

Card Input: "n" and a Title Card as was used for the search. (No card input is required if SEARCH2 is used to pass the array directly.)

Search Input: Must be compatible with the code.

Built-In Values: Ranges and intervals of interest. (The first NTAM parameters are SO_2 concentrations)

Programming Notes: FORT routine called by SEARCH2 or any retrieval program, includes HIST and JJPLOT (the latter returns an array of frequencies in JPLOT)

Operating Procedures: Combine with FORTRN, FORTRNK or S and prestore the program (App. A7a and A8b) and execute as in App. A8a. Also see App. A9 for executing jobs in one or two steps in conjunction with the search.

Deck Structure:

```
/*SETUP      DDNAME=OUTPUT,DEVICE=2400-9,ID=(069519,NORING,SAVE,SL)

//JOBLIB     DD      DSNAME=APSLIB,DISP=SHR

//GO         EXEC        PGM=IHISTJJ

//GO.ddname     DD       cards for SYSPRINT,FT06F001,& OUTPUT (See App. A5)

//GO.SYSIN      DD    *

  n in cols. 1-5: normally=99999 to base the analysis on all the observations.

  Title card as used for the search

STOP

/*
```

Note: Underlined items should be changed from run to run.

APPENDIX B24: REGRES (Ref. Section 8.1)

Description: Linear Regression Analysis

Tape Input: OUTPUT from a search including the parameters of interest.

Tape Output: None

Printed Output: Summary of stations and parameters in the output array from the search, subscripts for pertinent parameters, results of statistical calculations.

Running Time: 10 seconds for 3 independent parameters and 1 month's data (+10 seconds per month to search for 15 parameters).

Class & Region: B(230K)

Card Input: Search-like data, Title and specifications for dependent and independent parameters followed by END and STOP cards.

Search Input: Any combination of parameters. (Use SEARCH3 to omit observation times with missing data.)

Built-In Values: Limit of 5000 observation times and 1 dependent parameter.

Programming Notes: PL/I program calls a Fortran printing routine TWODPR and REGRES, includes routines from IBM's SSP, uses POINTER for dynamic storage allocation.

Operating Procedures: Prestore the program (App. A6b & A7a) and execute as in App. A8a. Also see App. A9b for executing jobs in two steps, SEARCH3 & REGRES.

Deck Structure:

```
/*SETUP      DDNAME=OUTPUT,DEVICE=2400-9,ID=(075021,NORING,SAVE,SL)

//JOBLIB     DD     DSNAME=APSLIB,DISP=SHR

//GO     EXEC     PGM=REGRES

//GO.ddname     DD     cards for SYSPRINT,FT06F001,CARDS,& OUTPUT (See App. A5).

//GO.SYSIN     DD     *

   Title card as used for search run

TYPE='DEP'     STATION='TAM_8'     PARAM='SO2';

TYPE='IND'     STATION='MIDWAY'     PARAM='WIND_VEL';

END


/*
```

Note: Underlined items should be changed from run to run.

APPENDIX B25: STEPMDA (Ref. Section 8.2)

Description: Stepwise Multiple Discriminant Analysis

Tape Input: OUTPUT from an appropriate search

Tape Output: None

Printed Output: Summary of stations and parameters in the output array from the search, tabulated data for each group, and results of statistical calculations (group memberships).

Running Time: 3 minutes for a sample case (+3 minutes per year to search for 9 out of 200 parameters).

Class & Region: A(336K)

Card Input: SYSIN: Title card as was used for the search, and a STOP card.
FT05F001: Title card, "problem" card, and group identification cards.

Search Input: Any combination of parameters. (Use SEARCH3 to omit observation times with missing data.)

Built-In Values: Limit of 50 variables and 20 groups with up to 1000 data points per group. The APICS SETUP routine provides for five variables and divides the data into 2 groups.

Programming Notes: FORT routine called by FORTRN or SEARCH2, uses ALLOC for dynamic storage allocation (must concatenate a library containing the latter when link-editing the program), calls a specially coded SETUP routine.

Operating Procedure: Combine with FORTRN and prestore the program (App. A7a and A8b) and execute as in App. A8a. Also see App. A9 for executing jobs in one or two steps in conjunction with the search.

Deck Structure:

```
/*SETUP      DDNAME=OUTPUT,DEVICE=2400-9,ID=(075399,NORING,SAVE,SL)
//JOBLIB      DD      DSNAME=APSLIB,DISP=SHR
//GO         EXEC       PGM=STEPMDA
//GO.ddname     DD     cards for SYSPRINT,FT06F001,& OUTPUT (See App. A5)
//GO.FT01F001    DD      UNIT=DISK,DISP=(NEW,PASS),SPACE=(CYL,(20,10))
//GO.FT04F001    DD      UNIT=DISK,DISP=(NEW,PASS),SPACE=(CYL,(20,10))
//GO.SYSIN       DD     *
```

Title card and STOP card

```
//GO.FT05F001       DD       *
```

Input data cards

```
/*
```

APPENDIX B26: TABPRED (Ref. Section 8.3)

Description: Tabulation (Prediction) Program.

Tape Input: OUTPUT from a search with prescribed specifications.

Tape Output: None

Printed Output: Summary of stations and parameters retrieved from a search followed by tables for predicting SO_2 levels from meteorological variables.

Running Time: 3 minutes to generate a complete set of tables (+15 seconds per month to search for 35 parameters out of 200)

Class & Region: B(200K)

Card Input: SYSIN: Title card as was used for the search, and a STOP card.
FT05F001: Title card, NTAM, range and precipitation specifications.

Search Input: Includes prescribed weather phenomenon with desired date limits.

Built-In Values: The APICS program expects (3 x NTAM + 11) specific ordered parameters. Up to 20 ranges may be specified for ceiling, wind speed, and temperature (wind direction and hour bands are incorporated into the specially coded IHOUR & WIND routines). Parameter values $\geq 10^5$ are considered missing and discarded.

Programming Notes: FORT routine called by FORTRN or SEARCH2.

Operating Procedure: Combine with FORTRN and prestore the program (App. A7a & A8b) and execute as in App. A8a. Also see App. A9 for executing jobs in one or two steps in conjunction with the search.

Deck Structure:

```
/*SETUP      DDNAME=OUTPUT,DEVICE=2400-9,ID=(077777,NORING,SAVE,SL)
//JOBLIB     DD      DSNAME=APSLIB,DISP=SHR
//GO         EXEC      PGM=TABPRED
//GO.ddname     DD     cards for SYSPRINT,FT06F001,and OUTPUT (See App. A5)
//GO.SYSIN      DD     *
   Title card and STOP card
//GO.FT05F001     DD      *
   Input data cards
/*
```

Note: Underlined items should be changed from run to run.

APPENDIX B27: PUFF (Ref. Section 8.4)

Description: Atmospheric Diffusion Calculations (Dispersion Model)

Tape Input: MASTER(MFF) must include all items specified in the input and prescribed parameters for Midway and TAM_n stations.
SOURCE from PLANTSIM: contains data from any number of emission point sources.

Tape Output: OUTPUT (optional): contains array of selected parameter values and calculated SO_2 values.

Printed Output: Summary of station and parameter information on input and output tapes; and, if specified, tables of parameter values in the output array. Intermediate output includes the weather data and calculated SO_2 values for each hour.

Running Time: 1 minute for sample case (depends on number of sources and receptors).

Class & Region: C(650K,1000K)

Card Input: Search-like data: title card, OPTLIST options (PRINT&FILE), specifications for date limits, receptor specifications, and output parameters (with delays) for OUT stations, followed by END and STOP cards.

Built-In Values: Reads specified weather data, assumes an x,y coordinate system, limit of 100 output parameters, 32767 observation times, and 50 receptor points.

Programming Notes: PL/I program calls Fortran routines for printing (TWODPT) and for storage allocation (LCSSET and BPOINTER). Calls a specially-coded Fortran routine via the PL/I to Fortran interface routine (FORTRAN). Must concatenate a library containing FORTRAN and BPOINTER to SYSLIB when link-editing the program.

Operating Procedures: Prestore the program (App. A6b & A7a) and execute as in App. A8a.

Deck Structure:

```
/*SETUP    DDNAME=MASTER,DEVICE=2400-9,ID=073740,NORING,SAVE,SL)
/*SETUP    DDNAME=SOURCE,DEVICE=2400-9,ID=073089,NORING,SAVE,SL)
/*SETUP    DDNAME=OUTPUT,DEVICE=2400-9,ID=(,RING,SAVE,SL)
//JOBLIB   DD     DSNAME=APSLIB,DISP=SHR
//GØ       EXEC    PGM=PUFF

//GO.ddname    DD   cards for FT06F001,SYSPRINT,CARDS,and OUTPUT (See App. A5)
//GO.MASTER    DD   DSNAME=CETAMET,UNIT=TAPE9TRK,DISP=(OLD,KEEP)
//GO.SOURCE    DD   DSNAME=OUTPUT,UNIT=TAPE9TRK,DISP=(OLD,KEEP)
//GO.SYSIN     DD   *
Input data as desired with FILE and/or PRINT options specified
/*
```

Note: Underlined items should be changed from run to run.

APICS USER MANUAL

APPENDIX C

DATA PREPARATION PROGRAMS
SOURCE LISTINGS

1. CREATE5 General Data Input Program

```
      SUBROUTINE TWODPT(ARAY,ICT,JCT,TITLE,DATE,COLT1,COLT2)
C                                  TWODPT PRINTS A TWO-DIMENSIONAL ARRAY.  IT
C                                 PRINTS A TITLE LINE AND TWO LINES OF COLUMN
C                                 TITLES, WHICH ARE FOLLOWED BY 50 LINES UP
C                                 TO TEN COLUMNS WIDE PLUS THE INDEX COLUMN,
C                                 WHICH CONTAINS DATE AND HOUR.
C                                 ARGUMENTS ARE -
C
C                                       ARAY - THE TWO-DIMENSIONAL ARRAY
C                                              TO BE PRINTED.
C                                        ICT - NUMBER OF ROWS IN ARRAY.
C                                        JCT - NUMBER OF COLUMNS IN ARRAY.
C                                      TITLE - PAGE TITLE (10A8).
C                                      COLT1 - ARRAY DIMENSIONED BY JCT, CON-
C                                              TAINING FIRST LINE OF TITLE
C                                              FOR EACH COLUMN (A8).
C                                      COLT2 - ARRAY DIMENSIONED BY JCT, CON-
C                                              TAINING SECOND LINE OF TITLE
C                                              FOR EACH COLUMN (A8).
      REAL*8 TITLE(10),DATE(1) ,COLT1(1),COLT2(1)
      REAL*4 ARAY(ICT,1)
      II=1
      IJ=1
      IMAX=0
      JMAX=0
      MAX=0
      IFLAG=0
   20 IEND=JCT
   40 MIN=MAX+1
      MAX=MINO(MAX+10,IEND)
      IF(IFLAG.NE.0) GO TO 47
   45 NROW=0
   46 CONTINUE
      PRINT 210,(TITLE(I),I=1,10)
      PRINT 220,DATE
   47 PRINT 230,(COLT1(JC),JC=MIN,MAX)
      PRINT 235,(COLT2(JC),JC=MIN,MAX)
      PRINT 235
      IMAX=MINO(II+52,ICT)
      IDIF=IMAX-II+5
      NROW=NROW+IDIF
      JMAX=MINO(IJ+9,JCT)
      DO 65 I=II,IMAX
      PRINT 240     ,              I,(ARAY(I,J),J=IJ,JMAX)
   65 CONTINUE
      IF(IMAX.GE.ICT) GO TO 70
      II=IMAX+1
      GO TO 46
   70 IF(JMAX.GE.JCT) GO TO 130
   80 II=1
      IJ=JMAX+1
      IF((NROW+IDIF-57).GT.0) GO TO 84
   82 IFLAG=1
      GO TO 40
   84 IFLAG=0
      GO TO 40
  130 CONTINUE
      DO 140 I=1,25
      PRINT 150
  150 FORMAT(' ')
  140 CONTINUE
      RETURN
  210 FORMAT(1H1,9X,10A8)
  220 FORMAT(10X'DATE:'A8)
  230 FORMAT(1H07X4HHOUR,10(4X,A8))
  235 FORMAT(12X,10(4X,A8))
  240 FORMAT(10XI2,10F12.3)
      END
```

```
CREATE5 :PROC OPTIONS(MAIN);
/*      CREATES MASTER FILE WITH ARBITRARY STATION DATA */
DCL     SOURCE_DATE CHAR(6);
DCL    1 SOURCE_DATA CONTROLLED,
                 3 YEAR CHAR(2),
                 3 MONTH CHAR(2),
                 3 DAY CHAR(2),
          2 VALUE (24,NEW_FILE_HEADER.PARAMS);
```

```
DCL     1 NEW_FILE_HEADER STATIC,
            2 STATIONS FIXED BINARY INITIAL (1),
            2 PARAMS FIXED BINARY ,
            2 STARTING_DATE CHAR(6),
            2 ENDING_DATE CHAR(6) ;
DCL     1 NEW_FILE_STATION (1) STATIC,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY INITIAL (0),
            2 PARAMS FIXED BINARY ;
DCL     1 NEW_FILE_PARAM (NEW_FILE_HEADER.PARAMS) CONTROLLED,
            2 NAME CHAR(8);
DCL     1 NEW_FILE_DATA CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (NEW_FILE_HEADER.PARAMS) ;
DCL       TITLE (10) CHAR(8);
DCL       PRINT_TITLE(10) FLOAT DECIMAL (16),
          PRINT_DATE  (1) FLOAT DECIMAL (16),
          PRINT_STATION (NEW_FILE_HEADER.PARAMS) CONTROLLED
          FLOAT DECIMAL (16),
          PRINT_NAME (NEW_FILE_HEADER.PARAMS) CONTROLLED
          FLOAT DECIMAL (16),
          PRINT_PARAM (NEW_FILE_HEADER.PARAMS,24) CONTROLLED;
DCL       IDUM FIXED BINARY (31,0), JDUM FIXED BINARY (31,0);
          CALL INPUT;
          IDUM=24; JDUM=NEW_FILE_HEADER.PARAMS;
          /* DATA UPDATE */
DCL       HOUR FIXED BINARY;
          ON ENDFILE(SOURCE) BEGIN;
          PUT LIST('END OF FILE ON SOURCE BEFORE STOP DATE REACHED')SKIP;
          GO TO STOP;END;
READ:     CALL GETDATA;
          NEW_FILE_DATA.DATE=SOURCE_DATE;
          DO HOUR=1 TO 24;
          CALL CONVERT;
          END; /* HOUR LOOP */
          WRITE FILE(NEWFILE) FROM (NEW_FILE_DATA);
          YEAR=SUBSTR(NEW_FILE_DATA.DATE,1,2);
          MONTH=SUBSTR(NEW_FILE_DATA.DATE,3,2);
          DAY  =SUBSTR(NEW_FILE_DATA.DATE,5,2);
          UNSPEC(PRINT_DATE(1))=UNSPEC(YEAR||'/'||MONTH||'/'||DAY);
          DO I=1 TO 24;
          DO J=1 TO NEW_FILE_HEADER.PARAMS;
          PRINT_PARAM(J,I)=NEW_FILE_DATA.PARAM(I,J);
          END;END;
          IF NEW_FILE_DATA.DATE¬=ENDING_DATE THEN GO TO READ;
GETDATA:  PROC;
DCL       M1 CHAR(1), M2 CHAR(1), D1 CHAR(1), D2 CHAR(1),
          BLANKS CHAR(72);
          GET FILE(SOURCE) EDIT
                  (YEAR,M1,M2,D1,D2,BLANKS)
          (A(2),X(1),2A(1),X(1),2A(1),A(72));
          GET FILE(SOURCE) EDIT
          (((VALUE(N,I) DO N=1 TO 24) DO I=1 TO NEW_FILE_HEADER.PARAMS))
          (8E(10,0));
          IF M1=' ' THEN M1='0';
          MONTH=M1||M2;
          IF D1=' ' THEN D1='0';
          DAY=D1||D2;
          SOURCE_DATE      =YEAR||MONTH||DAY;
          PUT DATA(SOURCE_DATE,SOURCE_DATA)SKIP;
          END GETDATA;
CONVERT:  PROC;
          DO J=1 TO NEW_FILE_HEADER.PARAMS;
          NEW_FILE_DATA.PARAM(HOUR,J)=VALUE(HOUR,J);
          END;
          END CONVERT;
INPUT:    PROC;
DCL       STATION CHAR(8), PARAM CHAR(8),
          PARAMS FIXED BINARY;
          GET                EDIT(TITLE)(10A(8));
          PUT EDIT(TITLE)(10A(8))PAGE;
          DO I=1 TO 10;
          UNSPEC(PRINT_TITLE(I))=UNSPEC(TITLE(I));
          END;
          GET LIST(STARTING_DATE,ENDING_DATE);
          PUT DATA(STARTING_DATE,ENDING_DATE)SKIP;
          GET DATA(STATION,PARAMS);
          PUT DATA(STATION,PARAMS)SKIP;
          NEW_FILE_HEADER.PARAMS=PARAMS;
          NEW_FILE_STATION.NAME(1)=STATION;
```

```
        NEW_FILE_STATION.PARAMS(1)=PARAMS;
        ALLOCATE NEW_FILE_PARAM;
        DO J=1 TO PARAMS;
        GET DATA(PARAM);
        PUT DATA(PARAM)SKIP;
        NEW_FILE_PARAM.NAME(J)=PARAM;
        END;
        PUT LIST('FILE HEADER INFORMATION')SKIP;
        WRITE FILE(NEWFILE) FROM (NEW_FILE_HEADER);
        PUT DATA(NEW_FILE_HEADER)SKIP;
        WRITE FILE(NEWFILE) FROM (NEW_FILE_STATION);
        PUT DATA(NEW_FILE_STATION)SKIP;
        WRITE FILE (NEWFILE) FROM (NEW_FILE_PARAM);
        PUT DATA(NEW_FILE_PARAM)SKIP;
        ALLOCATE SOURCE_DATA;
        ALLOCATE NEW_FILE_DATA;
        ALLOCATE PRINT_STATION;
        ALLOCATE PRINT_NAME;
        ALLOCATE PRINT_PARAM;
        N=0;
        DO I=1 TO NEW_FILE_HEADER.STATIONS;
        DO J=1 TO NEW_FILE_STATION(I).PARAMS;
        N=N+1;
        UNSPEC(PRINT_STATION(N))=UNSPEC(NEW_FILE_STATION(I).NAME)   ;
        UNSPEC(PRINT_NAME   (N))=UNSPEC(NEW_FILE_PARAM(J).NAME)   ;
        END;END;
        END INPUT;
STOP:   END CREATE5;
```

2. CREATE9 Calendar Program

```
      SUBROUTINE TWODPT(ARAY,ICT,JCT,TITLE,DATE,COLT1,COLT2)
C
C                                  TWODPT PRINTS A TWO-DIMENSIONAL ARRAY.  IT
C                                 PRINTS A TITLE LINE AND TWO LINES OF COLUMN
C                                 TITLES, WHICH ARE FOLLOWED BY 50 LINES UP
C                                 TO TEN COLUMNS WIDE PLUS THE INDEX COLUMN,
C                                 WHICH CONTAINS DATE AND HOUR.
C                                 ARGUMENTS ARE -
C
C                                       ARAY -  THE TWO-DIMENSIONAL ARRAY
C                                               TO BE PRINTED.
C                                        ICT -  NUMBER OF ROWS IN ARRAY.
C                                        JCT -  NUMBER OF COLUMNS IN ARRAY.
C                                      TITLE -  PAGE TITLE (10A8).
C                                      COLT1 -  ARRAY DIMENSIONED BY JCT, CON-
C                                               TAINING FIRST LINE OF TITLE
C                                               FOR EACH COLUMN (A8).
C                                      COLT2 -  ARRAY DIMENSIONED BY JCT, CON-
C                                               TAINING SECOND LINE OF TITLE
C                                               FOR EACH COLUMN (A8).
C
      REAL*8 TITLE(10),DATE(1) ,COLT1(1),COLT2(1)
      REAL*4 ARAY(ICT,1)
      II=1
      IJ=1
      IMAX=0
      JMAX=0
      MAX=0
      IFLAG=0
   20 IEND=JCT
   40 MIN=MAX+1
      MAX=MINO(MAX+10,IEND)
      IF(IFLAG.NE.0) GO TO 47
   45 NROW=0
   46 CONTINUE
C     PRINT 210,(TITLE(I),I=1,10)
      PRINT 220,DATE
C  47 PRINT 230,(COLT1(JC),JC=MIN,MAX)
C     PRINT 235,(COLT2(JC),JC=MIN,MAX)
C     PRINT 235
      IMAX=MINO(II+52,ICT)
      IDIF=IMAX-II+5
      NROW=NROW+IDIF
      JMAX=MINO(IJ+9,JCT)
C     DO 65 I=II,IMAX
C     PRINT 240     ,                I,(ARAY(I,J),J=IJ,JMAX)
C  65 CONTINUE
      IF(IMAX.GE.ICT) GO TO 70
      II=IMAX+1
      GO TO 46
   70 IF(JMAX.GE.JCT) GO TO 130
   80 II=1
      IJ=JMAX+1
      IF((NROW+IDIF-57).GT.0) GO TO 84
   82 IFLAG=1
      GO TO 40
   84 IFLAG=0
      GO TO 40
  130 CONTINUE
C     DO 140 I=1,25
C     PRINT 150
  150 FORMAT(' ')
C 140 CONTINUE
      RETURN
  210 FORMAT(1H1,9X,10A8)
  220 FORMAT(10X'DATE:'A8)
  230 FORMAT(1H07X4HHOUR,10(4X,A8))
  235 FORMAT(12X,10(4X,A8))
  240 FORMAT(10XI2,10F12.3)
      END
 CREATE9 :PROC OPTIONS(MAIN);
 /*       CREATES MASTER FILE WITH CALENDER INFORMATION  */
 DCL     1 CAL_DATA (2) STATIC,
                  3 YEARS CHAR(2) INITIAL ('68','69'),
                  3 MONTHS (12) CHAR(2) INITIAL
          ('01','02','03','04','05','06','07','08','09','10','11','12',
           '01','02','03','04','05','06','07','08','09','10','11','12'),
                  3 DAYS (12) FIXED(2) INITIAL
          (31,29,31,30,31,30,31,31,30,31,30,31,
           31,28,31,30,31,30,31,31,30,31,30,31);
 DCL             YEAR_NO FIXED (2) INITIAL(1),
```

```
                MONTH_NO FIXED (2) INITIAL(1),
                  DAY_NO FIXED (2) INITIAL(0);
DCL      SOURCE_DATE CHAR(6);
DCL     1 SOURCE_DATA,
             2 CALENDER (24),
                  3 DAY_WEEK CHAR(2),
                  3 DAY_MON  CHAR(2),
                  3 DAY_YEAR CHAR(5),
                  3 MONTH CHAR(2),
                  3 YEAR CHAR(2);
DCL     1 NEW_FILE_HEADER STATIC,
             2 STATIONS FIXED BINARY INITIAL (1),
             2 PARAMS FIXED BINARY INITIAL( 6),
             2 STARTING_DATE CHAR(6) INITIAL('680101'),
             2 ENDING_DATE CHAR(6) INITIAL('691231');
DCL     1 NEW_FILE_STATION (1) STATIC,
             2 NAME CHAR(8) INITIAL('CALENDAR'),
             2 POINTER FIXED BINARY INITIAL (0),
             2 PARAMS FIXED BINARY INITIAL( 6);
DCL     1 NEW_FILE_PARAM ( 6)  STATIC,
             2 NAME CHAR(8) INITIAL (
                    'HOUR',
                    'DAY_WEEK',
                    'DAY_MON',
                    'DAY_YEAR',
                    'MONTH',
                    'YEAR'
                   );
DCL     1 NEW_FILE_DATA CONTROLLED,
             2 DATE CHAR(6),
             2 HOUR(24),
                  3 PARAM (NEW_FILE_HEADER.PARAMS) ;
DCL      TITLE (10) CHAR(8);
DCL      PRINT_TITLE(10) FLOAT DECIMAL (16),
         PRINT_DATE  (1) FLOAT DECIMAL (16),
         PRINT_STATION (NEW_FILE_HEADER.PARAMS) CONTROLLED
         FLOAT DECIMAL (16),
         PRINT_NAME (NEW_FILE_HEADER.PARAMS) CONTROLLED
         FLOAT DECIMAL (16),
         PRINT_PARAM (NEW_FILE_HEADER.PARAMS,24) CONTROLLED;
DCL      IDUM FIXED BINARY (31,0), JDUM FIXED BINARY (31,0);
         IDUM=24; JDUM=NEW_FILE_HEADER.PARAMS;
         GET EDIT(TITLE)(10A(8));
         PUT EDIT(TITLE)(10A(8))PAGE;
         DO I=1 TO 10;
         UNSPEC(PRINT_TITLE(I))=UNSPEC(TITLE(I));
         END;
         PUT LIST('FILE HEADER INFORMATION')SKIP;
         WRITE FILE(NEWFILE) FROM (NEW_FILE_HEADER);
         PUT DATA(NEW_FILE_HEADER)SKIP;
         WRITE FILE(NEWFILE) FROM (NEW_FILE_STATION);
         PUT DATA(NEW_FILE_STATION)SKIP;
         WRITE FILE (NEWFILE) FROM (NEW_FILE_PARAM);
         PUT DATA(NEW_FILE_PARAM)SKIP;
         ALLOCATE NEW_FILE_DATA;
         ALLOCATE PRINT_STATION;
         ALLOCATE PRINT_NAME;
         ALLOCATE PRINT_PARAM;
         N=0;
         DO I=1 TO NEW_FILE_HEADER.STATIONS;
         DO J=1 TO NEW_FILE_STATION(I).PARAMS;
         N=N+1;
         UNSPEC(PRINT_STATION(N))=UNSPEC(NEW_FILE_STATION(I).NAME)   ;
         UNSPEC(PRINT_NAME   (N))=UNSPEC(NEW_FILE_PARAM(J).NAME)   ;
         END;END;
         /* DATA UPDATE */
DCL      HOUR FIXED BINARY;
         ON ENDFILE(SYSIN ) BEGIN;
         PUT LIST('END OF FILE ON SOURCE BEFORE STOP DATE REACHED')SKIP;
         GO TO STOP;END;
READ:    CALL GETDATA;
         NEW_FILE_DATA.DATE=SOURCE_DATE;
         DO HOUR=1 TO 24;
         CALL CONVERT;
         END; /* HOUR LOOP */
         WRITE FILE(NEWFILE) FROM (NEW_FILE_DATA);
         UNSPEC(PRINT_DATE(1))=UNSPEC(YEAR(1)||'/'||MONTH(1)||'/'||
         DAY_MON(1));
         DO I=1 TO 24;
         DO J=1 TO NEW_FILE_HEADER.PARAMS;
         PRINT_PARAM(J,I)=NEW_FILE_DATA.PARAM(I,J);
```

```
        END;END;
/*      CLOSE FILE(SYSPRINT);    */
        CALL TWODPT(PRINT_PARAM(1,1),IDUM,JDUM,PRINT_TITLE(1),
        PRINT_DATE(1),PRINT_STATION(1),PRINT_NAME(1));
        IF NEW_FILE_DATA.DATE¬=ENDING_DATE THEN GO TO READ;
GETDATA:PROC;
DCL     IGNORE CHAR(65);
        CALL DATE;
        YEAR(1)=SUBSTR(SOURCE_DATE,1,2);
        MONTH(1)=SUBSTR(SOURCE_DATE,3,2);
        DAY_MON(1)=SUBSTR(SOURCE_DATE,5,2);
        GET              EDIT(DAY_WEEK(1),DAY_YEAR(1),IGNORE)
        (A(2),X(8),A(5),A(65));
        DO N=2 TO 24          ;
        DAY_WEEK(N)=DAY_WEEK(1);
        DAY_MON (N)=DAY_MON (1);
        DAY_YEAR(N)=DAY_YEAR(1);
        MONTH(N)=MONTH(I);
        YEAR(N)=YEAR(1);
        END;
DATE:   PROC;
DCL     DUMDAY          PICTURE '99';
        IF DAY_NO+1<=DAYS(YEAR_NO,MONTH_NO) THEN GO TO SKIP;
        DAY_NO=0;
        IF MONTH_NO+1> 12 THEN BEGIN;
        MONTH_NO=1; YEAR_NO=YEAR_NO+1; END;
        ELSE
        MONTH_NO=MONTH_NO+1;
SKIP:   DAY_NO=DAY_NO+I;
        DUMDAY=DAY_NO;
        SOURCE_DATE=YEARS(YEAR_NO)||MONTHS(YEAR_NO,MONTH_NO)||DUMDAY;
        END DATE;
        END GETDATA;
CONVERT:PROC;
        NEW_FILE_DATA.PARAM(HOUR,1)=
        HOUR-1;
        NEW_FILE_DATA.PARAM(HOUR,2)=
        DAY_WEEK(HOUR);
        NEW_FILE_DATA.PARAM(HOUR,3)=
        DAY_MON (HOUR);
        NEW_FILE_DATA.PARAM(HOUR,4)=
        DAY_YEAR(HOUR);
        NEW_FILE_DATA.PARAM(HOUR,5)=
        MONTH(HOUR);
        NEW_FILE_DATA.PARAM(HOUR,6)=
        YEAR(HOUR);
        END CONVERT;
STOP:   END CREATE9;
```

3. CREATE3 Air Quality Data

```
      SUBROUTINE TWODPT(ARAY,ICT,JCT,TITLE,DATE,COLT1,COLT2)
C                                 TWODPT PRINTS A TWO-DIMENSIONAL ARRAY.  IT
C                                PRINTS A TITLE LINE AND TWO LINES OF COLUMN
C                                TITLES, WHICH ARE FOLLOWED BY 50 LINES UP
C                                TO TEN COLUMNS WIDE PLUS THE INDEX COLUMN,
C                                WHICH CONTAINS DATE AND HOUR.
C                                ARGUMENTS ARE -
C
C                                      ARAY - THE TWO-DIMENSIONAL ARRAY
C                                             TO BE PRINTED.
C                                       ICT - NUMBER OF ROWS IN ARRAY.
C                                       JCT - NUMBER OF COLUMNS IN ARRAY.
C                                     TITLE - PAGE TITLE (10A8).
C                                     COLT1 - ARRAY DIMENSIONED BY JCT, CON-
C                                             TAINING FIRST LINE OF TITLE
C                                             FOR EACH COLUMN (A8).
C                                     COLT2 - ARRAY DIMENSIONED BY JCT, CON-
C                                             TAINING SECOND LINE OF TITLE
C                                             FOR EACH COLUMN (A8).
C
      REAL*8 TITLE(10),DATE(1) ,COLT1(1),COLT2(1)
      REAL*4 ARAY(ICT,1)
      II=1
      IJ=1
      IMAX=0
      JMAX=0
      MAX=0
      IFLAG=0
   20 IEND=JCT
   40 MIN=MAX+1
      MAX=MIN0(MAX+10,IEND)
      IF(IFLAG.NE.0) GO TO 47
   45 NROW=0
   46 CONTINUE
      PRINT 210,(TITLE(I),I=1,10)
      PRINT 220,DATE
   47 PRINT 230,(COLT1(JC),JC=MIN,MAX)
      PRINT 235,(COLT2(JC),JC=MIN,MAX)
      PRINT 235
      IMAX=MIN0(II+52,ICT)
      IDIF=IMAX-II+5
      NROW=NROW+IDIF
      JMAX=MIN0(IJ+9,JCT)
      DO 65 I=II,IMAX
      PRINT 240     ,              I,(ARAY(I,J),J=IJ,JMAX)
   65 CONTINUE
      IF(IMAX.GE.ICT) GO TO 70
      II=IMAX+1
      GO TO 46
   70 IF(JMAX.GE.JCT) GO TO 130
   80 II=1
      IJ=JMAX+1
      IF((NROW+IDIF-57).GT.0) GO TO 84
   82 IFLAG=1
      GO TO 40
   84 IFLAG=0
      GO TO 40
  130 CONTINUE
      DO 140 I=1,25
      PRINT 150
  150 FORMAT(' ')
  140 CONTINUE
      RETURN
  210 FORMAT(1H1,9X,10A8)
  220 FORMAT(10X'DATE:'A8)
  230 FORMAT(1H07X4HHOUR,10(4X,A8))
  235 FORMAT(12X,10(4X,A8))
  240 FORMAT(10XI2,10F12.3)
      END
      SUBROUTINE ARCTAN(Y,X,THETA)
      DATA PI/3.14159265/                                                3
C                                                                        3
C     SPECIAL CASES                                                      3
C                                                                        3
      IF(ABS (X)-1.E-8)10,10,50
   10 CONTINUE                                                           5
      IF(Y)40,20,30                                                      6
   20 CONTINUE                                                           7
      THETA =0.
      RETURN                                                             9
```

```
   30 CONTINUE                                                          10
      THETA =.5*PI
      RETURN                                                            12
   40 CONTINUE                                                          13
      THETA =1.5*PI
      RETURN                                                            15
   50 CONTINUE                                                          16
      IF(X)90,10,60                                                     17
   60 CONTINUE                                                          18
      IF(Y)80,20,70                                                     19
   70 CONTINUE                                                          20
C                                                                       20
C     ANGLE IN FIRST QUADRANT                                           20
C                                                                       20
      THETA =ATAN (Y/X)
      RETURN                                                            22
   80 CONTINUE                                                          23
C                                                                       23
C     ANGLE IN FOURTH QUADRANT                                          23
C                                                                       23
      THETA =2.*PI-ATAN (ABS (Y/X))
      RETURN                                                            25
   90 CONTINUE                                                          26
      IF(Y)110,20,100                                                   27
  100 CONTINUE                                                          28
C                                                                       28
C     ANGLE IN SECOND QUADRANT                                          28
C                                                                       28
      THETA =PI-ATAN (ABS (Y/X))
      RETURN                                                            30
  110 CONTINUE                                                          31
C                                                                       31
C     ANGLE IN THIRD QUADRANT                                           31
C                                                                       31
      THETA =PI+ATAN (ABS (Y/X))
      RETURN                                                            33
      END
 CREATE3:PROC OPTIONS(MAIN);
/*       CREATES MASTER FILE WITH TAM RECEPTOR DATA   */
 DCL      SOURCE_DATE CHAR(6);
 DCL     1 SOURCE_DATA ,
             2 DAY(24),
                 3 HOUR(4),
                     4 M1 CHAR(1),
                     4 M2 CHAR(1),
                     4 D1 CHAR(1),
                     4 D2 CHAR(1),
                     4 YY   CHAR(2),
                     4 HOUR CHAR(2),
                     4 MIN  CHAR(2),
                      4 STA(8),
                           5 SO2 CHAR(3),
                           5 WIND_DIR CHAR(3),
                           5 WIND_VEL CHAR(3);
 DCL 1    AVE (8),
             2 SO_2 (5) CHAR(3),
             2 WD  (5) CHAR(3),
             2 WS  (5) CHAR(3);
 DCL     1 NEW_FILE_HEADER STATIC,
             2 STATIONS FIXED BINARY INITIAL (8),
             2 PARAMS FIXED BINARY INITIAL(24),
             2 STARTING_DATE CHAR(6) INITIAL('690201'),
             2 ENDING_DATE CHAR(6) INITIAL('690630');
 DCL     1 NEW_FILE_STATION (8) STATIC,
             2 NAME CHAR(8) INITIAL(
                   'TAM_1',
                   'TAM_2',
                   'TAM_3',
                   'TAM_4',
                   'TAM_5',
                   'TAM_6',
                   'TAM_7',
                   'TAM_8' ),
             2 POINTER FIXED BINARY INITIAL (0,3,6,9,12,15,18,21),
             2 PARAMS FIXED BINARY INITIAL(3,3,3,3,3,3,3,3);
 DCL     1 NEW_FILE_PARAM  (24) STATIC,
             2 NAME CHAR(8) INITIAL (
                   'SO2',
                   'WIND_DIR',
                   'WIND_VEL',
                   'SO2',
```

```
                    'WIND_DIR',
                    'WIND_VEL',
                    'SO2',
                    'WIND_DIR',
                    'WIND_VEL',
                    'SO2',
                    'WIND_DIR',
                    'WIND_VEL',
                    'SO2',
                    'WIND_DIR',
                    'WIND_VEL',
                    'SO2',
                    'WIND_DIR',
                    'WIND_VEL',
                    'SO2',
                    'WIND_DIR',
                    'WIND_VEL',
                    'SO2',
                    'WIND_DIR',
                    'WIND_VEL'
                    );
DCL     1 NEW_FILE_DATA CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (NEW_FILE_HEADER.PARAMS) ;
DCL       TITLE (10) CHAR(8);
DCL       PRINT_TITLE(10) FLOAT DECIMAL (16),
          PRINT_DATE  (1) FLOAT DECIMAL (16),
          PRINT_STATION (NEW_FILE_HEADER.PARAMS) CONTROLLED
          FLOAT DECIMAL (16),
          PRINT_NAME (NEW_FILE_HEADER.PARAMS) CONTROLLED
          FLOAT DECIMAL (16),
          PRINT_PARAM (NEW_FILE_HEADER.PARAMS,24) CONTROLLED;
DCL       YEAR CHAR(2), MONTH CHAR(2), DAY CHAR (2);
DCL       IDUM FIXED BINARY (31,0), JDUM FIXED BINARY (31,0);
          IDUM=24; JDUM=NEW_FILE_HEADER.PARAMS;
          GET EDIT(TITLE)(10A(8));
          PUT EDIT(TITLE)(10A(8))PAGE;
          DO I=1 TO 10;
          UNSPEC(PRINT_TITLE(I))=UNSPEC(TITLE(I));
          END;
          PUT LIST('FILE HEADER INFORMATION')SKIP;
          WRITE FILE(NEWFILE) FROM (NEW_FILE_HEADER);
          PUT DATA(NEW_FILE_HEADER)SKIP;
          WRITE FILE(NEWFILE) FROM (NEW_FILE_STATION);
          PUT DATA(NEW_FILE_STATION)SKIP;
          WRITE FILE (NEWFILE) FROM (NEW_FILE_PARAM);
          PUT DATA(NEW_FILE_PARAM)SKIP;
          ALLOCATE NEW_FILE_DATA;
          ALLOCATE PRINT_STATION;
          ALLOCATE PRINT_NAME;
          ALLOCATE PRINT_PARAM;
          N=0;
          DO I=1 TO NEW_FILE_HEADER.STATIONS;
          DO J=1 TO NEW_FILE_STATION(I).PARAMS;
          N=N+1;
          UNSPEC(PRINT_STATION(N))=UNSPEC(NEW_FILE_STATION(I).NAME)   ;
          UNSPEC(PRINT_NAME   (N))=UNSPEC(NEW_FILE_PARAM(J).NAME)   ;
          END;END;
          /* DATA UPDATE */
          ON ENDFILE(SOURCE) BEGIN;
          PUT LIST('END OF FILE ON SOURCE BEFORE STOP DATE REACHED')SKIP;
          GO TO STOP;END;
          DO K=1 TO 8;
          DO M=1 TO 2;
          SO_2(K,M)='###';
          WD  (K,M)='###';
          WS  (K,M)='###';
          END; END;
READ:     CALL GETDATA;
          NEW_FILE_DATA.DATE=SOURCE_DATE;
          WRITE FILE(NEWFILE) FROM (NEW_FILE_DATA);
          YEAR=SUBSTR(NEW_FILE_DATA.DATE,1,2);
          MONTH=SUBSTR(NEW_FILE_DATA.DATE,3,2);
          DAY  =SUBSTR(NEW_FILE_DATA.DATE,5,2);
          UNSPEC(PRINT_DATE(1))=UNSPEC(YEAR||'/'||MONTH||'/'||DAY);
          DO I=1 TO 24;
          DO J=1 TO NEW_FILE_HEADER.PARAMS;
          PRINT_PARAM(J,I)=NEW_FILE_DATA.PARAM(I,J);
          END;END;
          CLOSE FILE(SYSPRINT);
```

```
        CALL TWODPT(PRINT_PARAM(1,1),IDUM,JDUM,PRINT_TITLE(1),
        PRINT_DATE(1),PRINT_STATION(1),PRINT_NAME(1);
        IF NEW_FILE_DATA.DATE¬=ENDING_DATE THEN GO TO READ;
GETDATA:PROC;
        READ FILE (SOURCE) INTO (SOURCE_DATA);
        IF M1(1,1)=' ' THEN M1(1,1)='0';
        IF D1(1,1)=' ' THEN D1(1,1)='0';
        MONTH=M1(1,1)||M2(1,1);
        DAY  =D1(1,1)||D2(1,1);
        SOURCE_DATE=YY(1,1)||MONTH||DAY;
        ON CONVERSION ONCHAR='0';
        DO N=1 TO 24;
        DO K=1 TO 8;
        IF N=1 THEN GO TO SKIP1;
        AVE(K).SO_2(1)=SO2(N-1,3,K);
        AVE(K).SO_2(2)=SO2(N-1,4,K);
        AVE(K).WS(1)=WIND_VEL(N-1,3,K);
        AVE(K).WS(2)=WIND_VEL(N-1,4,K);
        AVE(K).WD(1)=WIND_DIR(N-1,3,K);
        AVE(K).WD(2)=WIND_DIR(N-1,4,K);
SKIP1:
        AVE(K).SO_2(3)=SO2(N,1,K);
        AVE(K).SO_2(4)=SO2(N,2,K);
        AVE(K).SO_2(5)=SO2(N,3,K);
        AVE(K).WD(3)=WIND_DIR(N,1,K)  ;
        AVE(K).WD(4)=WIND_DIR(N,2,K)  ;
        AVE(K).WD(5)=WIND_DIR(N,3,K)  ;
        AVE(K).WS(3)=WIND_VEL(N,1,K)  ;
        AVE(K).WS(4)=WIND_VEL(N,2,K)  ;
        AVE(K).WS(5)=WIND_VEL(N,3,K)  ;
        END;  /*  K LOOP   */
        DO K=1 TO 8;
        KSO2=0; SO2SUM=0.;
        KXY=0; XSUM=0.; YSUM=0.;
        DO M=1 TO 5;
        IF AVE.SO_2(K,M)='###' | AVE.SO_2(K,M)='999'|
           AVE.SO_2(K,M)='888' | AVE.SO_2(K,M)='777' |
           AVE.SO_2(K,M)='   '
        THEN GO TO JUMP1;
        KSO2=KSO2+1;  SO2SUM=SO2SUM+AVE.SO_2(K,M);
JUMP1:  IF AVE.WD(K,M)='###' | AVE.WD(K,M)='999' |
           AVE.WD(K,M)='888' | AVE.WD(K,M)='777' |
           AVE.WS(K,M)='###' | AVE.WS(K,M)='999'|
           AVE.WS(K,M)='888' | AVE.WS(K,M)='777' |
           AVE.WD(K,M)='   ' | AVE.WS(K,M)='   '
        THEN GO TO JUMP2;
        KXY=KXY+1;
        ARG=AVE.WD(K,M)    *3.14159265/180.;
        XSUM=XSUM+AVE.WS(K,M)*.1 *COS(ARG);
        YSUM=YSUM+AVE.WS(K,M)*.1 *SIN(ARG);
JUMP2:  END;  /* M LOOP  */
        IF KSO2=0 THEN
        NEW_FILE_DATA.PARAM(N   ,3*(K-1)+1)=1.E20;
        ELSE
        NEW_FILE_DATA.PARAM(N   ,3*(K-1)+1)=SO2SUM/KSO2* .01;
        IF KXY=0 THEN BEGIN;
        NEW_FILE_DATA.PARAM(N   ,3*(K-1)+2)=1.E20;
        NEW_FILE_DATA.PARAM(N   ,3*(K-1)+3)=1.E20;
        END;
        ELSE BEGIN;
        CALL ARCTAN(YSUM,XSUM,THETA)       ;
        NEW_FILE_DATA.PARAM(N   ,3*(K-1)+2)=
        THETA              *180./3.14159265;
        NEW_FILE_DATA.PARAM(N   ,3*(K-1)+3)=
        SQRT(YSUM**2+XSUM**2)/KXY;
        END;
        IF N¬=24 THEN GO TO SKIP2;
        AVE(K).SO_2(1)=SO2(  1,3,K);
        AVE(K).SO_2(2)=SO2(  1,4,K);
        AVE(K).WS(1)=WIND_VEL(  1,3,K);
        AVE(K).WS(2)=WIND_VEL(  1,4,K);
        AVE(K).WD(1)=WIND_DIR(  1,3,K);
        AVE(K).WD(2)=WIND_DIR(  1,4,K);
SKIP2:  END;  /*  K LOOP   */
        END;  /* END HOUR LOOP  */
        END GETDATA;
STOP:   END CREATE3;
```

4. LITTLEC1 Preprocessor for Meteorological Data

```
CREATE1 :PROC OPTIONS(MAIN);
/*      SCANS ASHVILLE WEATHER TAPES FOR PROPER STATION AND DATES */
DCL     SOURCE_DATE CHAR(6);
        DCL 1 SOURCE_DATA,
                2 WEATHER(24),
                        3 HOUR                          CHAR(2),
                        3 CEILING                       CHAR(4),
                        3 VISIBILITY                    CHAR(4),
                        3 WIND_DIR_1                    CHAR(2),
                        3 WIND_SPEED                    CHAR(3),
                        3 DRY_BULB_TEMP                 CHAR(3),
                        3 WET_BULB_TEMP                 CHAR(3),
                        3 DEW_POINT                     CHAR(3),
                        3 RELATIVE_HUMIDITY             CHAR(4),
                        3 SEA_LEVEL_PRESS               CHAR(5),
                        3 STATION_PRESS                 CHAR(4),
                        3 IGNORE_I                      CHAR(1),
                        3 SKY_CONDITION_1               CHAR(1),
                        3 SKY_CONDITION_2               CHAR(1),
                        3 SKY_CONDITION_3               CHAR(1),
                        3 SKY_CONDITION_4               CHAR(1),
                        3  CLOUD_TOTAL_AMOUNT           CHAR(1),
                        3  CLOUD_OPAQUE_AMOUNT          CHAR(1),
                        3  CLOUD_LAYER_I_AMOUNT         CHAR(1),
                        3  CLOUD_LAYER_1_TYPE           CHAR(1),
                        3  CLOUD_LAYER_1_HEIGHT         CHAR(3),
                        3  CLOUD_LAYER_2_AMOUNT         CHAR(1),
                        3  CLOUD_LAYER_2_TYPE           CHAR(1),
                        3  CLOUD_LAYER_2_HEIGHT         CHAR(3),
                        3  CLOUD_LAYER_2_SUM            CHAR(1),
                        3  CLOUD_LAYER_3_AMOUNT         CHAR(1),
                        3  CLOUD_LAYER_3_TYPE           CHAR(1),
                        3  CLOUD_LAYER_3_HEIGHT         CHAR(3),
                        3  CLOUD_LAYER_3_SUM            CHAR(1),
                        3  CLOUD_LAYER_4_AMOUNT         CHAR(1),
                        3  CLOUD_LAYER_4_TYPE           CHAR(1),
                        3  CLOUD_LAYER_4_HEIGHT         CHAR(3),
                        3 WIND_PHENOMENA                CHAR(1),
                        3 LIQUID_PRECIPITATION_1        CHAR(1),
                        3 LIQUID_PRECIPITATION_2        CHAR(1),
                        3 FROZEN_PRECIPITATION_1        CHAR(1),
                        3 FROZEN_PRECIPITATION_2        CHAR(1),
                        3 FROZEN_PRECIPITATION_3        CHAR(1),
                        3 VISION_OBSTRUCTIONS_I         CHAR(1),
                        3 VISION_OBSTRUCTIONS_2         CHAR(1),
                        3 WIND_DIR_2                    CHAR(2),
                        3 BLANK                         CHAR(3),
                        3 REC_MARK                      CHAR(1);
DCL             STARTING_DATE CHAR(6) INITIAL('670601'),
                ENDING_DATE CHAR(6) INITIAL('670731'),
                STATION CHAR(5) INITIAL('14855'),
                STA_TEST CHAR(5);
        ON ENDFILE(SOURCE) BEGIN;
        PUT LIST('END OF FILE ON SOURCE BEFORE STOP DATE REACHED')SKIP;
        GO TO STOP;END;
READ:   CALL GETDATA;
        IF SOURCE_DATE ¬= ENDING_DATE THEN GO TO READ;
        IF STA_TEST¬=STATION THEN GO TO READ;
GETDATA:PROC;
    DCL 1 SOURCE_IN,
            2 IGNORE CHAR(4),
            2 STA_NO CHAR(5),
            2 DATE CHAR(6),
            2 HOURS(6) LIKE WEATHER;
        DO N=1 TO 4;
        READ FILE(SOURCE ) INTO (SOURCE_IN);
        IF STA_NO¬=STATION THEN GO TO ENDN;
        IF DATE<STARTING_DATE|DATE>ENDING_DATE THEN GO TO ENDN;
        PUT DATA(DATE)SKIP; WRITE FILE(NEWFILE) FROM(SOURCE_IN);
ENDN:   END;    /* END OF N LOOP    */
        SOURCE_DATE      =SOURCE_IN.DATE;
        STA_TEST=STA_NO;
        RETURN;
        END GETDATA;
STOP:   END CREATE1;
```

5. CREATE1 Meteorology Data

```
      SUBROUTINE TWODPR(A)
      REAL*4 A(24,36)
      INTEGER*4 I(24,36)
      DO 10 M=1,24
      DO 10 N=1,36
      IF(A(M,N).GT.1.E10)GO TO 5
      I(M,N)=A(M,N)
      GO TO 10
    5 CONTINUE
      I(M,N)=9999999
   10 CONTINUE
      DO 30 M=1,24
      N=M-1
      PRINT 20,(N,I(M,1),I(M,2),I(M,3),I(M,6),I(M,7),A(M,8),
     1I(M,4),A(M,28),A(M,5),I(M,9),I(M,10),I(M,11),I(M,12),
     2A(M,13),A(M,14),I(M,17),I(M,18),I(M,21),I(M,22),
     3I(M,29),I(M,30),I(M,31),I(M,32),I(M,33),I(M,34),I(M,35),I(M,36))
   20 FORMAT(1X I2,2XI3,2XI2,2XI3,2XI3,2XI3,2XF5.2,2XI3,4XF3.1,4XF5.2,
     12XI2,1XI2,1XI2,1XI2,2XF3.1,1XF3.1,3XI2,1XI2,3XI3,1XI3,3XI1,4XI1,
     11XI1,5XI1,2XI1,2XI1,4XI1,1XI1)
   30 CONTINUE
      RETURN
      END
CREATE1 :PROC OPTIONS(MAIN);
/*      CREATES MASTER FILE WITH ASHVILLE WEATHER INFORMATION */
DCL     SOURCE_DATE CHAR(6);
        DCL 1 SOURCE_DATA,
                2 WEATHER(24),
                        3 HOUR                          CHAR(2),
                        3 CEILING                       CHAR(4),
                        3 VISIBILITY                    CHAR(4),
                        3 WIND_DIR_1                    CHAR(2),
                        3 WIND_SPEED                    CHAR(3),
                        3 DRY_BULB_TEMP                 CHAR(3),
                        3 WET_BULB_TEMP                 CHAR(3),
                        3 DEW_POINT                     CHAR(3),
                        3 RELATIVE_HUMIDITY             CHAR(4),
                        3 SEA_LEVEL_PRESS               CHAR(5),
                        3 STATION_PRESS                 CHAR(4),
                        3 IGNORE_I                      CHAR(1),
                        3 SKY_CONDITION_1               CHAR(1),
                        3 SKY_CONDITION_2               CHAR(1),
                        3 SKY_CONDITION_3               CHAR(1),
                        3 SKY_CONDITION_4               CHAR(1),
                        3  CLOUD_TOTAL_AMOUNT          CHAR(1),
                        3  CLOUD_OPAQUE_AMOUNT         CHAR(1),
                        3  CLOUD_LAYER_I_AMOUNT        CHAR(1),
                        3  CLOUD_LAYER_1_TYPE          CHAR(1),
                        3  CLOUD_LAYER_1_HEIGHT        CHAR(3),
                        3  CLOUD_LAYER_2_AMOUNT        CHAR(1),
                        3  CLOUD_LAYER_2_TYPE          CHAR(1),
                        3  CLOUD_LAYER_2_HEIGHT        CHAR(3),
                        3  CLOUD_LAYER_2_SUM           CHAR(1),
                        3  CLOUD_LAYER_3_AMOUNT        CHAR(1),
                        3  CLOUD_LAYER_3_TYPE          CHAR(1),
                        3  CLOUD_LAYER_3_HEIGHT        CHAR(3),
                        3  CLOUD_LAYER_3_SUM           CHAR(1),
                        3  CLOUD_LAYER_4_AMOUNT        CHAR(1),
                        3  CLOUD_LAYER_4_TYPE          CHAR(1),
                        3  CLOUD_LAYER_4_HEIGHT        CHAR(3),
                        3 WIND_PHENOMENA                CHAR(1),
                        3 LIQUID_PRECIPITATION_1        CHAR(1),
                        3 LIQUID_PRECIPITATION_2        CHAR(1),
                        3 FROZEN_PRECIPITATION_1        CHAR(1),
                        3 FROZEN_PRECIPITATION_2        CHAR(1),
                        3 FROZEN_PRECIPITATION_3        CHAR(1),
                        3 VISION_OBSTRUCTIONS_I         CHAR(1),
                        3 VISION_OBSTRUCTIONS_2         CHAR(1),
                        3 WIND_DIR_2                    CHAR(2),
                        3 BLANK                         CHAR(3),
                        3 REC_MARK                      CHAR(1);
DCL     1 NEW_FILE_HEADER STATIC,
            2 STATIONS FIXED BINARY INITIAL (1),
            2 PARAMS FIXED BINARY INITIAL(36),
            2 STARTING_DATE CHAR(6) INITIAL('680101'),
            2 ENDING_DATE CHAR(6) INITIAL('690531');
DCL     1 NEW_FILE_STATION (1) STATIC,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY INITIAL (0),
            2 PARAMS FIXED BINARY INITIAL(36);
DCL     1 NEW_FILE_PARAM  (36) STATIC,
```

```
            2 NAME CHAR(8) INITIAL (
                    'WIND_DIR',
                    'WIND_VEL',
                    'TEMPT'                    ,
                    'CEIL'     ,
                    'VIS'      ,
                    'DEW_PT'   ,
                    'REL_HUM'  ,
                    'PRESS'          ,
                    'SKY_CON1',
                    'SKY_CON2',
                    'SKY_CON3',
                    'SKY_CON4',
                    'CLD_AMT1',
                    'CLD_AMT2',
                    'CLD_AMT3',
                    'CLD_AMT4',
                    'CLD_TYP1',
                    'CLD_TYP2',
                    'CLD_TYP3',
                    'CLD_TYP4',
                    'CLD_HT1',
                    'CLD_HT2',
                    'CLD_HT3',
                    'CLD_HT4',
                    'CLD_SUM1',
                    'CLD_SUM2',
                    'CLD_SUM3',
                    'CLD_SUM4',
                    'WIND_PH',
                    'LIQ_PCP1',
                    'LIQ_PCP2',
                    'FRO_PCP1',
                    'FRO_PCP2',
                    'FRO_PCP3',
                    'OBS_VIS1',
                    'OBS_VIS2'
                    );
DCL     1 NEW_FILE_DATA CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                  3 PARAM (NEW_FILE_HEADER.PARAMS) ;
DCL       TITLE (10) CHAR(8);
DCL       PRINT_TITLE(10) FLOAT DECIMAL (16),
          PRINT_DATE  (1) FLOAT DECIMAL (16),
          PRINT_STATION (NEW_FILE_HEADER.PARAMS) CONTROLLED
          FLOAT DECIMAL (16),
          PRINT_NAME (NEW_FILE_HEADER.PARAMS) CONTROLLED
          FLOAT DECIMAL (16),
          PRINT_PARAM (NEW_FILE_HEADER.PARAMS,24) CONTROLLED;
DCL       YEAR CHAR(2), MONTH CHAR(2), DAY CHAR (2);
DCL       IDUM FIXED BINARY (31,0), JDUM FIXED BINARY (31,0);
          IDUM=24; JDUM=NEW_FILE_HEADER.PARAMS;
          GET EDIT(TITLE)(10A(8));
          PUT EDIT(TITLE)(10A(8))PAGE;
          NEW_FILE_STATION(1).NAME=TITLE(1);
          DO I=1 TO 10;
          UNSPEC(PRINT_TITLE(I))=UNSPEC(TITLE(I));
          END;
          PUT LIST('FILE HEADER INFORMATION')SKIP;
          WRITE FILE(NEWFILE) FROM (NEW_FILE_HEADER);
          PUT DATA(NEW_FILE_HEADER)SKIP;
          WRITE FILE(NEWFILE) FROM (NEW_FILE_STATION);
          PUT DATA(NEW_FILE_STATION)SKIP;
          WRITE FILE (NEWFILE) FROM (NEW_FILE_PARAM);
          PUT DATA(NEW_FILE_PARAM)SKIP;
          ALLOCATE NEW_FILE_DATA;
          ALLOCATE PRINT_STATION;
          ALLOCATE PRINT_NAME;
          ALLOCATE PRINT_PARAM;
          N=0;
          DO I=1 TO NEW_FILE_HEADER.STATIONS;
          DO J=1 TO NEW_FILE_STATION(I).PARAMS;
          N=N+1;
          UNSPEC(PRINT_STATION(N))=UNSPEC(NEW_FILE_STATION(I).NAME)   ;
          UNSPEC(PRINT_NAME    (N))=UNSPEC(NEW_FILE_PARAM(J).NAME)   ;
          END;END;
          /* DATA UPDATE */
DCL       HOUR FIXED BINARY;
          JUMP=0;
          ON ENDFILE(SOURCE) BEGIN;
```

```
        PUT LIST('END OF FILE ON SOURCE BEFORE STOP DATE REACHED')SKIP;
        GO TO STOP;END;
READ:   CALL GETDATA;
        NEW_FILE_DATA.DATE=SOURCE_DATE;
        DO HOUR=1 TO 24;
        CALL CONVERT;
        END; /* HOUR LOOP */
        WRITE FILE(NEWFILE) FROM (NEW_FILE_DATA);
        YEAR=SUBSTR(NEW_FILE_DATA.DATE,1,2);
        MONTH=SUBSTR(NEW_FILE_DATA.DATE,3,2);
        DAY  =SUBSTR(NEW_FILE_DATA.DATE,5,2);
        UNSPEC(PRINT_DATE(1))=UNSPEC(YEAR||'/'||MONTH||'/'||DAY);
        DO I=1 TO 24;
        DO J=1 TO NEW_FILE_HEADER.PARAMS;
        PRINT_PARAM(J,I)=NEW_FILE_DATA.PARAM(I,J);
        END;END;
        IF JUMP=0 THEN BEGIN;
        PUT EDIT(TITLE)(10A(8))SKIP; /* PAGE FOR PCP OR SKIP FOR MVT */
        JUMP=1; END;
        ELSE  JUMP=0;
        PUT EDIT('DATE: ',YEAR,'/',MONTH,'/',DAY)(A(6),A(2),A(1),
        A(2),A(1),A(2))SKIP;
        PUT EDIT('TTL CLD','SKY CONDITN','CLD AMT','CLD TYP','CLD HGT',
        'LIQ PCP','FRZN PCP','OBS VIS')
        (COLUMN(41),A(7),COLUMN(57),A(11),X(2),A(7),X(2),A(7),X(2),
        A(7),X(6),A(7),X(2),A(8),X(2),A(7))SKIP;
        PUT EDIT('HR','WD','WS','TEM','DP','RH','PRESS','CH','COVER',
        'VIS','1','2','3','4','1','2','1','2','1','2','WP','1','2',
        '1','2','3','1','2')
        (A(2),X(3),A(2),X(2),A(2),X(2),A(3),X(3),A(2),X(3),A(2),X(2),
        A(5),X(3),A(2),X(3),A(5),X(4),A(3),X(4),A(1),X(2),A(1),X(2),
        A(1),X(2),A(1),X(3),A(1),X(3),A(1),X(5),A(1),X(2),A(1),X(4),
        A(1),X(3),A(1),X(3),A(2),X(4),A(1),X(1),A(1),X(5),A(1),X(2),
        A(1),X(2),A(1),X(4),A(1),X(1),A(1))SKIP;
/*      CLOSE FILE(SYSPRINT); FOR PCP BUT NOT FOR MVT */
        CALL TWODPR(PRINT_PARAM(1,1));
        IF NEW_FILE_DATA.DATE¬=ENDING_DATE THEN GO TO READ;
GETDATA:PROC;
    DCL 1 SOURCE_IN,
            2 IGNORE CHAR(9),
            2 DATE CHAR(6),
            2 HOURS(6) LIKE WEATHER;
        DO N=1 TO 4;
        READ FILE(SOURCE ) INTO (SOURCE_IN);
        DO I=1 TO 6;
        WEATHER              ((N-1)*6+I)=SOURCE_IN.HOURS(I);
        END;   /* END OF I LOOP   */
        END;   /* END OF N LOOP   */
        SOURCE_DATE      =SOURCE_IN.DATE;
        RETURN;
        END GETDATA;
CONVERT:PROC;
        IF WIND_DIR_2(HOUR)='  ' |
           WIND_DIR_2(HOUR)=' *' THEN
        NEW_FILE_DATA.PARAM(HOUR,1)= 1.E20;
        ELSE
        NEW_FILE_DATA.PARAM(HOUR,1)=
              WIND_DIR_2(HOUR)*10.;
        IF WIND_SPEED(HOUR)='  '|
           WIND_SPEED(HOUR)='  *' THEN
        NEW_FILE_DATA.PARAM(HOUR,2)=1.E20;
        ELSE BEGIN;
        UNSPEC(WIND_SPEED(HOUR))=
        UNSPEC(        WIND_SPEED(HOUR))|'000000000000000C1111000'B;
        NEW_FILE_DATA.PARAM(HOUR,2)=
        WIND_SPEED(HOUR);
        END;
        IF DRY_BULB_TEMP(HOUR)='   '|
           DRY_BULB_TEMP(HOUR)='  *' THEN
        NEW_FILE_DATA.PARAM(HOUR,3)=1.E20;
        ELSE BEGIN;
        FACT=UNSPEC(SUBSTR(DRY_BULB_TEMP(HOUR),3,1))&'00010000'B;
        FACT=FACT/16.;
        NFACT=FACT;
        SIGN=(-1)**NFACT;
        UNSPEC(DRY_BULB_TEMP(HOUR))=
        (UNSPEC(DRY_BULB_TEMP(HOUR))|'000000000000000011110000'B);
        NEW_FILE_DATA.PARAM(HOUR,3)=
        SIGN*DRY_BULB_TEMP(HOUR);
        END;
        IF CEILING(HOUR)='-   '|
```

```
   CEILING(HOUR)='    *'|
   CEILING(HOUR)='     ' THEN
NEW_FILE_DATA.PARAM(HOUR,4)=1.E20;
ELSE
NEW_FILE_DATA.PARAM(HOUR,4)=
CEILING(HOUR);
IF VISIBILITY(HOUR)='     '|
   VISIBILITY(HOUR)='    *' THEN
NEW_FILE_DATA.PARAM(HOUR,5)=1.E20;
ELSE BEGIN;
N1=SUBSTR(VISIBILITY(HOUR),4,1);
N2=SUBSTR(VISIBILITY(HOUR),3,1);
N3=SUBSTR(VISIBILITY(HOUR),2,1);
IF N3>0|N2>2 THEN
NEW_FILE_DATA.PARAM(HOUR,5)=
VISIBILITY(HOUR)/10.     ;
ELSE BEGIN;
IF N1>6 THEN
NEW_FILE_DATA.PARAM(HOUR,5)=
N2+(N1+(N1-7+1))*.0625;
ELSE
NEW_FILE_DATA.PARAM(HOUR,5)=
N2+N1*.0625;END;
END;
IF DEW_POINT(HOUR)='    '|
   DEW_POINT(HOUR)='   *' THEN
NEW_FILE_DATA.PARAM(HOUR,6)=1.E20;
ELSE BEGIN;
FACT=UNSPEC(SUBSTR(DEW_POINT     (HOUR),3,1))&'00010000'B;
FACT=FACT/16.;
NFACT=FACT;
SIGN=(-1)**NFACT;
UNSPEC(DEW_POINT     (HOUR))=
(UNSPEC(DEW_POINT     (HOUR))|'000000000000000011110000'B);
NEW_FILE_DATA.PARAM(HOUR,6)=
SIGN*DEW_POINT     (HOUR);
END;
IF RELATIVE_HUMIDITY(HOUR)='     ' THEN
NEW_FILE_DATA.PARAM(HOUR,7)=1.E20;
ELSE
NEW_FILE_DATA.PARAM(HOUR,7)=
RELATIVE_HUMIDITY(HOUR);
IF STATION_PRESS(HOUR)='     ' THEN
NEW_FILE_DATA.PARAM(HOUR,8)=1.E20;
ELSE
NEW_FILE_DATA.PARAM(HOUR,8)=
STATION_PRESS(HOUR)*.01;
IF SKY_CONDITION_1(HOUR)='*' THEN
NEW_FILE_DATA.PARAM(HOUR,9)=1.E20;
ELSE BEGIN;
IF SKY_CONDITION_1(HOUR)='-' THEN
NEW_FILE_DATA.PARAM(HOUR,9)=10.;
ELSE BEGIN;
IF SKY_CONDITION_1(HOUR)=' ' THEN
NEW_FILE_DATA.PARAM(HOUR,9)=11.;
ELSE
NEW_FILE_DATA.PARAM(HOUR,9)=SKY_CONDITION_1(HOUR);
END; END;
IF SKY_CONDITION_2(HOUR)='*' THEN
NEW_FILE_DATA.PARAM(HOUR,10)=1.E20;
ELSE BEGIN;
IF SKY_CONDITION_2(HOUR)='-' THEN
NEW_FILE_DATA.PARAM(HOUR,10)=10.;
ELSE BEGIN;
IF SKY_CONDITION_2(HOUR)=' ' THEN
NEW_FILE_DATA.PARAM(HOUR,10)=11.;
ELSE
NEW_FILE_DATA.PARAM(HOUR,10)=SKY_CONDITION_2(HOUR);
END; END;
IF SKY_CONDITION_3(HOUR)='*' THEN
NEW_FILE_DATA.PARAM(HOUR,11)=1.E20;
ELSE BEGIN;
IF SKY_CONDITION_3(HOUR)='-' THEN
NEW_FILE_DATA.PARAM(HOUR,11)=10.;
ELSE BEGIN;
IF SKY_CONDITION_3(HOUR)=' ' THEN
NEW_FILE_DATA.PARAM(HOUR,11)=11.;
ELSE
NEW_FILE_DATA.PARAM(HOUR,11)=SKY_CONDITION_3(HOUR);
END; END;
IF SKY_CONDITION_4(HOUR)='*' THEN
```

```
NEW_FILE_DATA.PARAM(HOUR,12)=1.E20;
ELSE BEGIN;
IF SKY_CONDITION_4(HOUR)='-' THEN
NEW_FILE_DATA.PARAM(HOUR,12)=10.;
ELSE BEGIN;
IF SKY_CONDITION_4(HOUR)=' ' THEN
NEW_FILE_DATA.PARAM(HOUR,12)=11.;
ELSE
NEW_FILE_DATA.PARAM(HOUR,12)=SKY_CONDITION_4(HOUR);
END; END;
IF CLOUD_LAYER_1_AMOUNT(HOUR)='-' THEN
NEW_FILE_DATA.PARAM(HOUR,13)=1.0;
ELSE BEGIN;
IF CLOUD_LAYER_1_AMOUNT(HOUR)=' ' THEN
NEW_FILE_DATA.PARAM(HOUR,13)=1.0E20;
ELSE
NEW_FILE_DATA.PARAM(HOUR,13)=CLOUD_LAYER_1_AMOUNT(HOUR)*.1;
END;
IF CLOUD_LAYER_2_AMOUNT(HOUR)='-' THEN
NEW_FILE_DATA.PARAM(HOUR,14)=1.0;
ELSE BEGIN;
IF CLOUD_LAYER_2_AMOUNT(HOUR)=' ' THEN
NEW_FILE_DATA.PARAM(HOUR,14)=1.0E20;
ELSE
NEW_FILE_DATA.PARAM(HOUR,14)=CLOUD_LAYER_2_AMOUNT(HOUR)*.1;
END;
IF CLOUD_LAYER_3_AMOUNT(HOUR)='-' THEN
NEW_FILE_DATA.PARAM(HOUR,15)=1.0;
ELSE BEGIN;
IF CLOUD_LAYER_3_AMOUNT(HOUR)=' ' THEN
NEW_FILE_DATA.PARAM(HOUR,15)=1.0E20;
ELSE
NEW_FILE_DATA.PARAM(HOUR,15)=CLOUD_LAYER_3_AMOUNT(HOUR)*.1;
END;
IF CLOUD_LAYER_4_AMOUNT(HOUR)='-' THEN
NEW_FILE_DATA.PARAM(HOUR,16)=1.0;
ELSE BEGIN;
IF CLOUD_LAYER_4_AMOUNT(HOUR)=' ' THEN
NEW_FILE_DATA.PARAM(HOUR,16)=1.0E20;
ELSE
NEW_FILE_DATA.PARAM(HOUR,16)=CLOUD_LAYER_4_AMOUNT(HOUR)*.1;
END;
IF CLOUD_LAYER_1_TYPE  (HOUR)='-' THEN
NEW_FILE_DATA.PARAM(HOUR,17)=20.;
ELSE BEGIN;
IF CLOUD_LAYER_1_TYPE  (HOUR)=' ' THEN
NEW_FILE_DATA.PARAM(HOUR,17)=1.E20;
ELSE BEGIN;
TEST=UNSPEC( CLOUD_LAYER_1_TYPE(HOUR))&'00100000'B;
IF TEST¬=0. THEN
NEW_FILE_DATA.PARAM(HOUR,17)= CLOUD_LAYER_1_TYPE(HOUR);
ELSE BEGIN;
UNSPEC( CLOUD_LAYER_1_TYPE(HOUR))=
UNSPEC( CLOUD_LAYER_1_TYPE(HOUR))|'11110000'B;
NEW_FILE_DATA.PARAM(HOUR,17)= CLOUD_LAYER_1_TYPE(HOUR)+10.;
END; END; END;
IF CLOUD_LAYER_2_TYPE  (HOUR)='-' THEN
NEW_FILE_DATA.PARAM(HOUR,18)=20.;
ELSE BEGIN;
IF CLOUD_LAYER_2_TYPE  (HOUR)=' ' THEN
NEW_FILE_DATA.PARAM(HOUR,18)=1.E20;
ELSE BEGIN;
TEST=UNSPEC( CLOUD_LAYER_2_TYPE(HOUR))&'00100000'B;
IF TEST¬=0. THEN
NEW_FILE_DATA.PARAM(HOUR,18)= CLOUD_LAYER_2_TYPE(HOUR);
ELSE BEGIN;
UNSPEC( CLOUD_LAYER_2_TYPE(HOUR))=
UNSPEC( CLOUD_LAYER_2_TYPE(HOUR))|'11110000'B;
NEW_FILE_DATA.PARAM(HOUR,18)= CLOUD_LAYER_2_TYPE(HOUR)+10.;
END; END; END;
IF CLOUD_LAYER_3_TYPE  (HOUR)='-' THEN
NEW_FILE_DATA.PARAM(HOUR,19)=20.;
ELSE BEGIN;
IF CLOUD_LAYER_3_TYPE  (HOUR)=' ' THEN
NEW_FILE_DATA.PARAM(HOUR,19)=1.E20;
ELSE BEGIN;
TEST=UNSPEC( CLOUD_LAYER_3_TYPE(HOUR))&'00100000'B;
IF TEST¬=0. THEN
NEW_FILE_DATA.PARAM(HOUR,19)= CLOUD_LAYER_3_TYPE(HOUR);
ELSE BEGIN;
UNSPEC( CLOUD_LAYER_3_TYPE(HOUR))=
```

```
UNSPEC( CLOUD_LAYER_3_TYPE(HOUR))|'11110000'B;
NEW_FILE_DATA.PARAM(HOUR,19)= CLOUD_LAYER_3_TYPE(HOUR)+10.;
END; END; END;
IF CLOUD_LAYER_4_TYPE  (HOUR)='-' THEN
NEW_FILE_DATA.PARAM(HOUR,20)=20.;
ELSE BEGIN;
IF CLOUD_LAYER_4_TYPE  (HOUR)=' ' THEN
NEW_FILE_DATA.PARAM(HOUR,20)=1.E20;
ELSE BEGIN;
TEST=UNSPEC( CLOUD_LAYER_4_TYPE(HOUR))&'00100000'B;
IF TEST¬=0. THEN
NEW_FILE_DATA.PARAM(HOUR,20)= CLOUD_LAYER_4_TYPE(HOUR);
ELSE BEGIN;
UNSPEC( CLOUD_LAYER_4_TYPE(HOUR))=
UNSPEC( CLOUD_LAYER_4_TYPE(HOUR))|'11110000'B;
NEW_FILE_DATA.PARAM(HOUR,20)= CLOUD_LAYER_4_TYPE(HOUR)+10.;
END; END; END;
IF  CLOUD_LAYER_1_HEIGHT(HOUR)='---'THEN
NEW_FILE_DATA.PARAM(HOUR,21)=0.;
ELSE BEGIN;
IF  CLOUD_LAYER_1_HEIGHT(HOUR)='   '|
    CLOUD_LAYER_1_HEIGHT(HOUR)='  *' THEN
NEW_FILE_DATA.PARAM(HOUR,21)=1.E20;
ELSE
NEW_FILE_DATA.PARAM(HOUR,21)= CLOUD_LAYER_1_HEIGHT(HOUR);
END;
IF  CLOUD_LAYER_2_HEIGHT(HOUR)='---'THEN
NEW_FILE_DATA.PARAM(HOUR,22)=0.;
ELSE BEGIN;
IF  CLOUD_LAYER_2_HEIGHT(HOUR)='   '|
    CLOUD_LAYER_2_HEIGHT(HOUR)='  *' THEN
NEW_FILE_DATA.PARAM(HOUR,22)=1.E20;
ELSE
NEW_FILE_DATA.PARAM(HOUR,22)= CLOUD_LAYER_2_HEIGHT(HOUR);
END;
IF  CLOUD_LAYER_3_HEIGHT(HOUR)='---'THEN
NEW_FILE_DATA.PARAM(HOUR,23)=0.;
ELSE BEGIN;
IF  CLOUD_LAYER_3_HEIGHT(HOUR)='   '|
    CLOUD_LAYER_3_HEIGHT(HOUR)='  *' THEN
NEW_FILE_DATA.PARAM(HOUR,23)=1.E20;
ELSE
NEW_FILE_DATA.PARAM(HOUR,23)= CLOUD_LAYER_3_HEIGHT(HOUR);
END;
IF  CLOUD_LAYER_4_HEIGHT(HOUR)='---'THEN
NEW_FILE_DATA.PARAM(HOUR,24)=0.;
ELSE BEGIN;
IF  CLOUD_LAYER_4_HEIGHT(HOUR)='   '|
    CLOUD_LAYER_4_HEIGHT(HOUR)='  *' THEN
NEW_FILE_DATA.PARAM(HOUR,24)=1.E20;
ELSE
NEW_FILE_DATA.PARAM(HOUR,24)= CLOUD_LAYER_4_HEIGHT(HOUR);
END;
NEW_FILE_DATA.PARAM(HOUR,25)=
NEW_FILE_DATA.PARAM(HOUR,13);
IF CLOUD_LAYER_2_SUM   (HOUR)='-' THEN
NEW_FILE_DATA.PARAM(HOUR,26)=1.0;
ELSE BEGIN;
IF CLOUD_LAYER_2_SUM   (HOUR)=' ' THEN
NEW_FILE_DATA.PARAM(HOUR,26)=1.0E20;
ELSE
NEW_FILE_DATA.PARAM(HOUR,26)=CLOUD_LAYER_2_SUM   (HOUR)*.1;
END;
IF CLOUD_LAYER_3_SUM   (HOUR)='-' THEN
NEW_FILE_DATA.PARAM(HOUR,27)=1.0;
ELSE BEGIN;
IF CLOUD_LAYER_3_SUM   (HOUR)=' ' THEN
NEW_FILE_DATA.PARAM(HOUR,27)=1.0E20;
ELSE
NEW_FILE_DATA.PARAM(HOUR,27)=CLOUD_LAYER_3_SUM   (HOUR)*.1;
END;
IF CLOUD_TOTAL_AMOUNT  (HOUR)='-' THEN
NEW_FILE_DATA.PARAM(HOUR,28)=1.0;
ELSE BEGIN;
IF CLOUD_TOTAL_AMOUNT  (HOUR)=' ' THEN
NEW_FILE_DATA.PARAM(HOUR,28)=1.0E20;
ELSE
NEW_FILE_DATA.PARAM(HOUR,28)=CLOUD_TOTAL_AMOUNT  (HOUR)*.1;
END;
IF WIND_PHENOMENA(HOUR)=' '|
   WIND_PHENOMENA(HOUR)='*' THEN
```

```
        NEW_FILE_DATA.PARAM(HOUR,29)=10.;
        ELSE
        NEW_FILE_DATA.PARAM(HOUR,29)=WIND_PHENOMENA(HOUR);
        IF LIQUID_PRECIPITATION_1(HOUR)=' '|
           LIQUID_PRECIPITATION_1(HOUR)='*' THEN
        NEW_FILE_DATA.PARAM(HOUR,30)=10.;
        ELSE
        NEW_FILE_DATA.PARAM(HOUR,30)=LIQUID_PRECIPITATION_1(HOUR);
        IF LIQUID_PRECIPITATION_2(HOUR)=' '|
           LIQUID_PRECIPITATION_2(HOUR)='*' THEN
        NEW_FILE_DATA.PARAM(HOUR,31)=10.;
        ELSE
        NEW_FILE_DATA.PARAM(HOUR,31)=LIQUID_PRECIPITATION_2(HOUR);
        IF FROZEN_PRECIPITATION_1(HOUR)=' '|
           FROZEN_PRECIPITATION_1(HOUR)='*' THEN
        NEW_FILE_DATA.PARAM(HOUR,32)=10.;
        ELSE
        NEW_FILE_DATA.PARAM(HOUR,32)=FROZEN_PRECIPITATION_1(HOUR);
        IF FROZEN_PRECIPITATION_2(HOUR)=' '|
           FROZEN_PRECIPITATION_2(HOUR)='*' THEN
        NEW_FILE_DATA.PARAM(HOUR,33)=10.;
        ELSE
        NEW_FILE_DATA.PARAM(HOUR,33)=FROZEN_PRECIPITATION_2(HOUR);
        IF FROZEN_PRECIPITATION_3(HOUR)=' '|
           FROZEN_PRECIPITATION_3(HOUR)='*' THEN
        NEW_FILE_DATA.PARAM(HOUR,34)=10.;
        ELSE
        NEW_FILE_DATA.PARAM(HOUR,34)=FROZEN_PRECIPITATION_3(HOUR);
        IF VISION_OBSTRUCTIONS_1(HOUR)=' '|
           VISION_OBSTRUCTIONS_1(HOUR)='*' THEN
        NEW_FILE_DATA.PARAM(HOUR,35)=10.;
        ELSE
        NEW_FILE_DATA.PARAM(HOUR,35)=VISION_OBSTRUCTIONS_1(HOUR);
        IF VISION_OBSTRUCTIONS_2(HOUR)=' '|
           VISION_OBSTRUCTIONS_2(HOUR)='*' THEN
        NEW_FILE_DATA.PARAM(HOUR,36)=10.;
        ELSE
        NEW_FILE_DATA.PARAM(HOUR,36)=VISION_OBSTRUCTIONS_2(HOUR);
        RETURN;
        END CONVERT;
STOP   :END CREATE1 ;
```

6. TURSTAB Functions of File Parameters (Turner Stability Example)

```
      SUBROUTINE FORTRN(NYEAR,NMONTH,NDAY,NIN,PARIN,NOUT,PAROUT)
C
      REAL*4 PARIN(24,1),PAROUT(24,1)
C
C     YOUR PROGRAM STARTS HERE
C
      DO 10 NHR=1,24
      COV=PARIN(NHR,1)
      CEIL=PARIN(NHR,2)
      WS=PARIN(NHR,3)
      NHS=NHR-1
      CALL STAB(NMONTH,NDAY,NHS,COV,CEIL,WS,NCLASA ,NCLASB )
      PAROUT(NHR,1)=NCLASA
      PAROUT(NHR,2)=NCLASB
   10 CONTINUE
      RETURN
      END
      SUBROUTINESTAB(MO,NDAY,NHR,COV,CEIL,WS,NCLASA,NCLASB)
C     CALCULATES TURNER STABILITY CLASSIFICATION FOR
C     EACH HOUR.
C     IDENTIFIES SUNRISE AND SUNSET AND
C     TWO HOURS EITHER SIDE.
C     CLASSES ARE 1 THROUGH 5, 81 THROUGH 85
C     FOR TWO HOURS BEFORE SUNRISE, 91 THROUGH 95,
C     101 THROUGH 105 FOR SUNRISE HOUR,
C     111 THROUGH 115, 121 THROUGH 125,
C     181 THROUGH 185 FOR 2 HRS BEFORE SUNSET,
C     191 THROUGH 195, 201-205,211-215,
C     AND 221-225.
C
      CEIL=100.*CEIL
C     CALCULATE JULIAN DAY
      J=30.42*(MO-1)+NDAY
C     CALC. DECLINATION OF SUN IN DEGREES
      DECL=23.27*SIN( .0172 *(J-80))
C     CALC. SUNRISE AND SUNSET HOURS
C
      IF(DECL.GE.20.)GOTO11
      IF(DECL.GE.0.)GOTO12
      IF(DECL.GE.-15.)GOTO13
      GOTO14
   11 NSR=4
      NSS=19
      GOTO20
   12 NSR=5
      NSS=18
      GOTO20
   13 NSR=6
      NSS=17
      GOTO20
   14 NSR=7
      NSS=16
   20 CONTINUE
      KRISE=3
      KSET=3
      IF(NHR.GT.(NSR+2).AND.NHR.LT.(NSS-2))GOTO200
      IF(NHR.LT.(NSR-2).OR.NHR.GT.(NSS+2))GOTO100
      IF(NHR.GT.12)GOTO40
      KRISE=NHR-NSR
      GOTO50
   40 KSET=NHR-NSS
   50 CONTINUE
      IF(NHR.GT.NSR.AND.NHR.LT.NSS)GOTO200
C     NIGHT TIME
  100 IF(COV.GE..95.AND.CEIL.LE.7000.)GOTO110
      IF(COV.LE..4)GOTO120
      NRAD=-1
      GOTO300
  110 NRAD=0
      GOTO300
  120 NRAD=-2
      GOTO300
C     DAYTIME
C
  200 NSUN=IABS(12-NHR)
      IF(COV.GE..95.AND.CEIL.LT.7000.)GOTO290
C
C     CALC.ANGLE OF SUN IN DEGREES
      DECL=DECL*.01745
      HRANG=NSUN*.2654
```

```
      ALP=57.29*ARSIN(.670*SIN(DECL)+.743*COS(DECL)*COS(HRANG))
C
C     TURNERS TABLE A-2
C
      IF(ALP.GT.60.)GOTO211
      IF(ALP.GT.35.)GOTO212
      IF(ALP.GT.15.)GOTO213
C
      NRAD=1
      GOTO220
  211 NRAD=4
      GOTO220
  212 NRAD=3
      GOTO220
  213 NRAD=2
  220 IF(COV.LT..51)GOTO300
      IF(CEIL.LT.7000.)NRAD=NRAD-2
      IF(CEIL.GE.7000..AND.CEIL.LT.16000.)
     1NRAD=NRAD-1
      IF(COV.GT..99)NRAD=NRAD-1
      IF(NRAD.LT.1)NRAD=1
      GOTO300
  290 NRAD=0
C
  300 CONTINUE
      CEIL=.01*CEIL
C     TURNERS TABLE A-1
      NCLASS=4
      NR=NRAD+3
      GOTO(310,320,330,340,350,360,370),NR
  310 IF(WS.LE.10.5)NCLASS=5
      GOTO380
  320 IF(WS.LE.6.5)NCLASS=5
  330 GOTO380
  340 IF(WS.LE.3.5)NCLASS=3
      GOTO380
  350 NCLASS=3
      IF(WS.LE.3.5)NCLASS=2
      IF(WS.GE.9.5)NCLASS=4
      GOTO380
  360 IF(WS.LE.7.5)GOTO362
      NCLASS=3
      IF(WS.GE.11.5)NCLASS=4
      GOTO380
  362 NCLASS=2
      IF(WS.LE.1.5)NCLASS=1
      GOTO380
  370 NCLASS=2
      IF(WS.LE.5.5)NCLASS=1
      IF(WS.GE.9.5)NCLASS=3
      GOTO380
  380 CONTINUE
      NCLASB=NCLASS*100
C
C     ASSIGN SPECIAL CLASS TO HOURS ABOUT
C     SUNRISE AND SUNSET
C
      IF(KRISE.GE.3)GOTO390
      NCLASS=NCLASS+100+KRISE*10
      NCLASB=NCLASB+KRISE+3
      GOTO395
  390 IF(KSET.GE.3)GOTO395
      NCLASS=NCLASS+200+KSET*10
      NCLASB=NCLASB+KSET+13
  395 NCLASA=NCLASS
      RETURN
      END
FUNCTN : PROC OPTIONS (MAIN);
    DCL DATE_LIMITS FIXED BINARY INITIAL(0),
        LOWER_DATE(DATE_LIMITS) CHAR(8) CONTROLLED,
        UPPER_DATE(DATE_LIMITS) CHAR(8) CONTROLLED;
DCL     1 FILE_HEADER,
            2 STATIONS FIXED BINARY,
            2 PARAMS FIXED BINARY,
            2 STARTING_DATE CHAR(6),
            2 ENDING_DATE CHAR(6);
DCL     1 FILE_STATION (FILE_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 FILE_PARAM (FILE_HEADER.PARAMS) CONTROLLED,
```

```
            2 NAME CHAR(8);
DCL     1 TODAY        CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (FILE_HEADER.PARAMS);
DCL     1 YESTERDAY CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (FILE_HEADER.PARAMS);
DCL     1 SUBFILE_HEADER,
            2 STATIONS FIXED BINARY,
            2 PARAMS FIXED BINARY,
            2 STARTING_DATE CHAR(6),
            2 ENDING_DATE CHAR(6);
DCL     1 SUBFILE_STATION (SUBFILE_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 SUBFILE_PARAM (SUBFILE_HEADER.PARAMS) CONTROLLED,
            2 NAME CHAR(8);
DCL     1 SUBFILE_DATA CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (SUBFILE_HEADER.PARAMS);
    DCL OUT_STATIONS FIXED BINARY (31,0),
      IN_STATIONS FIXED BINARY (31,0),
     OUT_PARAMS   FIXED BINARY (31,0),
      IN_PARAMS   FIXED BINARY (31,0);
DCL     1 OUT_PARAM (OUT_PARAMS) CONTROLLED,
            2 POINTER FIXED BINARY,
            2 DELAY FIXED DECIMAL (2);
DCL     1 DUM_OUT_PARAM (100)  CONTROLLED,
            2 POINTER FIXED BINARY,
            2 DELAY FIXED DECIMAL (2);
DCL     1 OUTPUT_HEADER,
            2 LABEL (10) CHAR(8),
            2 STATIONS FIXED BINARY (31,0),
            2 PARAMS FIXED BINARY (31,0),
            2 NUMBER_OBS FIXED BINARY (31,0);
DCL     1 OUTPUT_STATION (OUTPUT_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 OUTPUT_PARAM (OUTPUT_HEADER.PARAMS) CONTROLLED,
            2 STATION CHAR(8),
            2 NAME CHAR(8);
DCL     1 DUM_OUTPUT_PARAM(100)  CONTROLLED,
            2 STATION CHAR(8),
            2 NAME CHAR(8);
DCL       PRINT_TITLE (10) FLOAT DECIMAL (16),
          PRINT_DATE(1) FLOAT DECIMAL (16),
          PRINT_STATION (OUT_PARAMS) CONTROLLED FLOAT DECIMAL (16),
          PRINT_NAME     (OUT_PARAMS) CONTROLLED FLOAT DECIMAL (16),
          PRINT_PARAM (JDUM,IDUM              ) CONTROLLED,
            DUM_PARAM (KDUM,IDUM              ) CONTROLLED;
DCL       PARAMS FIXED BINARY,BANDS FIXED BINARY,
          DELAY FIXED DECIMAL (2);
     DCL  STATION CHAR(8),PARAM CHAR(8);
DCL       IDUM FIXED BINARY (31,0), JDUM FIXED BINARY (31,0);
DCL       KDUM FIXED BINARY (31,0);
DCL       CARD CHAR(80);
          PUT LIST('MASTER FILE HEADER INFORMATION')PAGE;
          READ FILE(MASTER) INTO (FILE_HEADER);
          PUT DATA(FILE_HEADER)SKIP;
          ALLOCATE FILE_STATION;
          READ FILE(MASTER) INTO (FILE_STATION);
          PUT DATA(FILE_STATION)SKIP;
          ALLOCATE FILE_PARAM;
          READ FILE(MASTER) INTO (FILE_PARAM);
          PUT DATA (FILE_PARAM)SKIP;
          ALLOCATE TODAY; ALLOCATE YESTERDAY;
NEXT:     /* BEGIN NEW CASE */
          CLOSE FILE(MASTER);
          ON ENDFILE(MASTER)  BEGIN;
          PUT LIST('END OF FILE SENSED ON MASTER')SKIP;
          GO TO NEXT ;END;
          READ FILE (MASTER) IGNORE(3);
          CLOSE FILE(CARDS);
GETCARD:GET EDIT(CARD)(A(80));
          IF INDEX(CARD,'END ')¬=0 THEN GO TO END;
          IF INDEX(CARD,'STOP')¬=0 THEN GO TO STOP;
```

```
        WRITE FILE(CARDS) FROM (CARD);
        GO TO GETCARD;
END:    CLOSE FILE(CARDS);
        CALL INPUT;
        CALL SEARCH;
        GO TO NEXT;
INPUT:  PROC;
        ON NAME(CARDS) BEGIN;
        PUT LIST('THE FOLLOWING DATA IS OUT OF ORDER: ',DATAFIELD)SKIP;
        GO TO NEXT;END;
        ON ENDFILE(CARDS)  BEGIN;
        PUT LIST('UNEXPECTED EOF ON CARD INPUT')SKIP;
        GO TO NEXT;END;
        PUT LIST('INPUT DATA')PAGE;
        GET FILE(CARDS) EDIT(LABEL)(10A(8));
        DO I=1 TO 10;
        UNSPEC(PRINT_TITLE(I))=UNSPEC(LABEL(I));
        END;
        PUT EDIT(LABEL)(10A(8))SKIP;
        FREE     LOWER_DATE; FREE     UPPER_DATE;
        DATE_LIMITS=1;
        ALLOCATE LOWER_DATE; ALLOCATE UPPER_DATE;
        LOWER_DATE(1)=FILE_HEADER.STARTING_DATE;
        UPPER_DATE(1)=FILE_HEADER.ENDING_DATE;
        PUT DATA(LOWER_DATE(1),UPPER_DATE(1))SKIP;
        OUT_PARAMS=0;
        PUT LIST(' IN STATION REQUESTS')SKIP;
        GET FILE(CARDS) DATA ( IN_STATIONS);
        OUT_STATIONS=IN_STATIONS;
        IF OUT_STATIONS=0 THEN BEGIN;
        PUT LIST('NO  IN STATIONS REQUESTED')SKIP;
        GO TO RETURN; END;
        OUTPUT_HEADER.STATIONS=OUT_STATIONS;
        FREE     OUTPUT_STATION;
        ALLOCATE OUTPUT_STATION;
        OUT_PARAMS=0;
        ALLOCATE DUM_OUT_PARAM;
        ALLOCATE DUM_OUTPUT_PARAM;
        DO I=1 TO OUT_STATIONS;
        PARAMS=0;
        GET FILE(CARDS) DATA(STATION,PARAMS);
        PUT DATA(STATION)SKIP;
        DO N=1 TO FILE_HEADER.STATIONS;
        IF STATION=FILE_STATION(N).NAME THEN GO TO FIND_N2;
        END;/* N LOOP*/
        PUT LIST('STATION NAME ',STATION,' NOT RECOGNIZED');GO TO NEXT;
FIND_N2:OUTPUT_STATION(I).NAME=STATION;
        OUTPUT_STATION(I).POINTER=OUT_PARAMS;
        IF PARAMS=0 THEN BEGIN;
        PARAMS=FILE_STATION(N).PARAMS;
        IF OUT_PARAMS+PARAMS>100 THEN BEGIN;
        PUT LIST('NUMBER OF PARAMETERS GREATER THAN 100')SKIP;
        GO TO NEXT;END;
        ELSE BEGIN   ;
        OUTPUT_STATION(I).PARAMS=PARAMS;
        DELAY=0;
        DO J=1 TO PARAMS;
        DUM_OUTPUT_PARAM(OUT_PARAMS+J).STATION=STATION;
        PARAM=
        FILE_PARAM(FILE_STATION(N).POINTER+J).NAME;
        PUT DATA(PARAM,DELAY)SKIP;
        DUM_OUTPUT_PARAM(OUT_PARAMS+J).NAME=  PARAM;
        DUM_OUT_PARAM(OUT_PARAMS+J).POINTER=FILE_STATION(N).POINTER+J;
        DUM_OUT_PARAM(OUT_PARAMS+J).DELAY=DELAY;
        END; /* PARAM LOOP */
        GO TO NEWSTA; END;          END;
        IF OUT_PARAMS+PARAMS>100 THEN BEGIN;
        PUT LIST('NUMBER OF PARAMETERS GREATER THAN 100')SKIP;
        GO TO NEXT;END;
        OUTPUT_STATION(I).PARAMS=PARAMS;
        DO J=1 TO PARAMS;
        DELAY=0;
        GET FILE(CARDS) DATA(PARAM,DELAY);
        PUT DATA(PARAM,DELAY)SKIP;
        DUM_OUTPUT_PARAM(OUT_PARAMS+J).NAME=PARAM;
        DUM_OUTPUT_PARAM(OUT_PARAMS+J).STATION=STATION;
        DO M=1 TO FILE_STATION (N).PARAMS;
        IF PARAM=FILE_PARAM(FILE_STATION(N).POINTER+M).NAME
        THEN GO TO FIND_M2;
        END;/* M LOOP*/
        PUT LIST('PARAMETER NAME ',PARAM,' NOT RECOGNIZED');GO TO NEXT;
```

```
FIND_M2:DUM_OUT_PARAM(OUT_PARAMS+J).POINTER=FILE_STATION(N).POINTER+M;
        DUM_OUT_PARAM(OUT_PARAMS+J).DELAY=DELAY;
        END; /* PARAM LOOP */
NEWSTA: OUT_PARAMS=OUT_PARAMS+PARAMS;
        END; /*STATION LOOP */
        OUTPUT_HEADER.PARAMS=OUT_PARAMS;
        FREE     OUTPUT_PARAM;
        ALLOCATE OUTPUT_PARAM;
        FREE     OUT_PARAM;
        ALLOCATE OUT_PARAM;
        FREE     PRINT_STATION;
        ALLOCATE PRINT_STATION;
        FREE     PRINT_NAME;
        ALLOCATE PRINT_NAME;
        DO J=1 TO OUT_PARAMS;
        OUTPUT_PARAM(J)=DUM_OUTPUT_PARAM(J);
        OUT_PARAM(J)=DUM_OUT_PARAM(J);
        STATION=OUTPUT_PARAM(J).STATION;
        PARAM  =OUTPUT_PARAM(J).NAME    ;
        UNSPEC(PRINT_STATION(J))=UNSPEC(STATION);
        UNSPEC(PRINT_NAME   (J))=UNSPEC(PARAM);
        END;
        FREE DUM_OUTPUT_PARAM;
        FREE DUM_OUT_PARAM;
RETURN: IN_PARAMS=OUT_PARAMS;
        RETURN;
        END INPUT;
SEARCH: PROC;
DCL     HOUR FIXED BINARY;
DCL     DATE_KEY BIT(1) INITIAL('0'B);
DCL     DATE_LIMIT FIXED BINARY;
DCL     DELAY_HOUR FIXED BINARY;
DCL     YEAR CHAR(2), MONTH CHAR(2), DAY CHAR(2);
DCL     FUNCTION CHAR(8);
        IDUM=24; JDUM= IN_PARAMS;
        DATE_LIMIT=1;
        GET FILE(CARDS)DATA(FUNCTION,PARAMS);
        PUT DATA(FUNCTION)SKIP;
        IF PARAMS=0 THEN BEGIN;
        PUT LIST('NO FUNCTION PARAMETERS REQUESTED')SKIP;
        GO TO NEXT; END;
        OUT_PARAMS=PARAMS;
        KDUM=OUT_PARAMS;
        PUT LIST('SUB FILE HEADER INFORMATION')PAGE;
        /* CREATE SUB FILE HEADER */
        SUBFILE_HEADER.STATIONS=1;
        SUBFILE_HEADER.PARAMS=PARAMS;
        SUBFILE_HEADER.STARTING_DATE=FILE_HEADER.STARTING_DATE;
        SUBFILE_HEADER.  ENDING_DATE=FILE_HEADER.  ENDING_DATE;
        PUT DATA(SUBFILE_HEADER)SKIP;
        WRITE FILE(SUBFILE) FROM (SUBFILE_HEADER);
        /* CREATE SUBFILE STATION DEFINITIONS */
        FREE     SUBFILE_STATION;
        ALLOCATE SUBFILE_STATION;
        SUBFILE_STATION.NAME(1)=FUNCTION;
        SUBFILE_STATION.POINTER(1)=0;
        SUBFILE_STATION.PARAMS(1)=PARAMS;
        PUT DATA(SUBFILE_STATION)SKIP;
        WRITE FILE(SUBFILE) FROM (SUBFILE_STATION);
        /* CREATE SUB FILE PARAMETER DEFINITIONS  */
        FREE     SUBFILE_PARAM;
        ALLOCATE SUBFILE_PARAM;
        DO N=1 TO OUT_PARAMS;
        GET FILE(CARDS) DATA(PARAM);
        PUT DATA(PARAM)SKIP;
        SUBFILE_PARAM.NAME(N)=PARAM;
        END;
        PUT DATA(SUBFILE_PARAM)SKIP;
        WRITE FILE(SUBFILE) FROM (SUBFILE_PARAM);
        FREE     SUBFILE_DATA;
        ALLOCATE SUBFILE_DATA;
        FREE     PRINT_PARAM;
        ALLOCATE PRINT_PARAM;
        FREE     DUM_PARAM;
        ALLOCATE DUM_PARAM;
        ON ENDFILE(MASTER)  BEGIN;
        PUT LIST('END OF FILE SENSED ON MASTER')SKIP;
        GO TO RETURN; END;
        READ FILE(MASTER) INTO (TODAY);
        YESTERDAY=TODAY;
DATES:  /* LOOP ON DATES - TEST BETWEEN DATE LIMITS */
```

```
        IF TODAY.DATE=LOWER_DATE(DATE_LIMIT) THEN BEGIN;
        DATE_KEY='1'B; GO TO HOURS;
        END;
        IF TODAY.DATE=UPPER_DATE(DATE_LIMIT) THEN BEGIN;
        DATE_KEY='0'B;DATE_LIMIT=DATE_LIMIT+1;
        GO TO HOURS;
        END;
        IF DATE_KEY='0'B THEN GO TO NEWDATE;
HOURS:  /* LOOP ON HOURS */
        SUBFILE_DATA.DATE=TODAY.DATE;
     PUT DATA(SUBFILE_DATA.DATE)SKIP;
        YEAR =SUBSTR(TODAY.DATE,1,2);
        MONTH=SUBSTR(TODAY.DATE,3,2);
        DAY  =SUBSTR(TODAY.DATE,5,2);
        UNSPEC(PRINT_DATE(1))=UNSPEC(YEAR||'/'||MONTH||'/'||DAY);
        IF IN_PARAMS=0 THEN GO TO SKIPP;
        DO HOUR=1 TO 24;
        DO J=1 TO  IN_PARAMS;
        DELAY_HOUR=HOUR-OUT_PARAM(J).DELAY;
        IF DELAY_HOUR<1 THEN BEGIN;
        PRINT_PARAM(J,HOUR)=
        YESTERDAY.PARAM(24+DELAY_HOUR,OUT_PARAM.POINTER(J));
        GO TO SKIP;END;
        ELSE   BEGIN;
        PRINT_PARAM(J,HOUR)=
        TODAY.PARAM(DELAY_HOUR,OUT_PARAM.POINTER(J));   END;
SKIP:   END; /* PARAM LOOP */
        END; /* HOUR LOOP */
SKIPP:  NYEAR=YEAR; NMONTH=MONTH; NDAY=DAY;
        CALL FORTRN(NYEAR,NMONTH,NDAY,IN_PARAMS,PRINT_PARAM(1,1),
        OUT_PARAMS,DUM_PARAM(1,1));
        DO HOUR=1 TO 24;
        DO K=1 TO OUT_PARAMS;
        SUBFILE_DATA.PARAM(HOUR,K)=DUM_PARAM(K,HOUR);
        END;END;
        WRITE FILE(SUBFILE) FROM (SUBFILE_DATA);
NEWDATE:IF DATE_LIMIT>DATE_LIMITS THEN RETURN;
        YESTERDAY=TODAY;
        READ FILE(MASTER) INTO (TODAY);
        GO TO DATES;
RETURN: RETURN;
        END SEARCH;
STOP:   END FUNCTN;
```

7. MASTER2 Specialized Data Input Program (Mixing Height Example)

```
      SUBROUTINE TWODPT(ARAY,ICT,JCT,TITLE,DATE,COLT1,COLT2)
C
C                                  TWODPT PRINTS A TWO-DIMENSIONAL ARRAY.  IT
C                                 PRINTS A TITLE LINE AND TWO LINES OF COLUMN
C                                 TITLES, WHICH ARE FOLLOWED BY 50 LINES UP
C                                 TO TEN COLUMNS WIDE PLUS THE INDEX COLUMN,
C                                 WHICH CONTAINS DATE AND HOUR.
C                                 ARGUMENTS ARE -
C
C                                       ARAY - THE TWO-DIMENSIONAL ARRAY
C                                              TO BE PRINTED.
C                                        ICT - NUMBER OF ROWS IN ARRAY.
C                                        JCT - NUMBER OF COLUMNS IN ARRAY.
C                                      TITLE - PAGE TITLE (10A8).
C                                      COLT1 - ARRAY DIMENSIONED BY JCT, CON-
C                                              TAINING FIRST LINE OF TITLE
C                                              FOR EACH COLUMN (A8).
C                                      COLT2 - ARRAY DIMENSIONED BY JCT, CON-
C                                              TAINING SECOND LINE OF TITLE
C                                              FOR EACH COLUMN (A8).
C
      REAL*8 TITLE(10),DATE(1) ,COLT1(1),COLT2(1)
      REAL*4 ARAY(ICT,1)
C     II=1
C     IJ=1
C     IMAX=0
C     JMAX=0
C     MAX=0
C     IFLAG=0
C  20 IEND=JCT
C  40 MIN=MAX+1
C     MAX=MINO(MAX+10,IEND)
C     IF(IFLAG.NE.0) GO TO 47
C  45 NROW=0
C  46 CONTINUE
C     PRINT 210,(TITLE(I),I=1,10)
C     PRINT 220,DATE
C  47 PRINT 230,(COLT1(JC),JC=MIN,MAX)
C     PRINT 235,(COLT2(JC),JC=MIN,MAX)
C     PRINT 235
C     IMAX=MINO(II+52,ICT)
C     IDIF=IMAX-II+5
C     NROW=NROW+IDIF
C     JMAX=MINO(IJ+9,JCT)
C     DO 65 I=II,IMAX
C     PRINT 240     ,              I,(ARAY(I,J),J=IJ,JMAX)
C  65 CONTINUE
C     IF(IMAX.GE.ICT) GO TO 70
C     II=IMAX+1
C     GO TO 46
C  70 IF(JMAX.GE.JCT) GO TO 130
C  80 II=1
C     IJ=JMAX+1
C     IF((NROW+IDIF-57).GT.0) GO TO 84
C  82 IFLAG=1
C     GO TO 40
C  84 IFLAG=0
C     GO TO 40
C 130 CONTINUE
C     DO 140 I=1,25
C     PRINT 150
C 150 FORMAT(' ')
C 140 CONTINUE
      RETURN
  210 FORMAT(1H1,9X,10A8)
  220 FORMAT(10X'DATE:'A8)
  230 FORMAT(1H07X4HHOUR,10(4X,A8))
  235 FORMAT(12X,10(4X,A8))
  240 FORMAT(10XI2,10F12.3)
      END
 MASTER2 :PROC OPTIONS(MAIN);
 /*       CREATES MASTER FILE WITH INDUSTRIAL PLANT SOURCE DATA */
 DCL      SOURCE_DATE CHAR(6);
 DCL     1 SOURCE_DATA,
                 3 YEAR CHAR(2),
                 3 MONTH CHAR(2),
                 3 DAY CHAR(2),
                 3 KEY FIXED DECIMAL (1),
             2 SO2(24),
                 3 CHAR FIXED DECIMAL(4),
                 3 EXP FIXED DECIMAL(2);
```

```
DCL     1 NEW_FILE_HEADER STATIC,
            2 STATIONS FIXED BINARY INITIAL (1),
            2 PARAMS FIXED BINARY INITIAL(2),
            2 STARTING_DATE CHAR(6) INITIAL('670801'),
            2 ENDING_DATE CHAR(6) INITIAL('671231');
DCL     1 NEW_FILE_STATION (1) STATIC,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY INITIAL (0),
            2 PARAMS FIXED BINARY INITIAL(2);
DCL     1 NEW_FILE_PARAM    (2) STATIC,
            2 NAME CHAR(8) INITIAL (
                    'MIX_HT',
                    'KEY'
                   );
DCL     1 NEW_FILE_DATA CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                 3 PARAM (NEW_FILE_HEADER.PARAMS) ;
DCL      TITLE (10) CHAR(8) INITIAL ('        ','        ','        ',
             '        ','        ','        ','        ',
             '        ','        ','        ','        ');
DCL       PRINT_TITLE(10) FLOAT DECIMAL (16),
          PRINT_DATE  (1) FLOAT DECIMAL (16),
          PRINT_STATION (NEW_FILE_HEADER.PARAMS) CONTROLLED
          FLOAT DECIMAL (16),
          PRINT_NAME (NEW_FILE_HEADER.PARAMS) CONTROLLED
          FLOAT DECIMAL (16),
          PRINT_PARAM (NEW_FILE_HEADER.PARAMS,24) CONTROLLED;
DCL       IDUM FIXED BINARY (31,0), JDUM FIXED BINARY (31,0);
          IDUM=24; JDUM=NEW_FILE_HEADER.PARAMS;
DCL       DUMMY CHAR(4);
          GET FILE(SOURCE) EDIT(TITLE)(10A(8));
          PUT EDIT(TITLE)(10A(8))PAGE;
          NEW_FILE_STATION(1).NAME=TITLE(1);
          DO I=1 TO 10;
          UNSPEC(PRINT_TITLE(I))=UNSPEC(TITLE(I));
          END;
          GET FILE(SOURCE) EDIT(TITLE)(10A(8));
          PUT LIST('FILE HEADER INFORMATION')SKIP;
          WRITE FILE(NEWFILE) FROM (NEW_FILE_HEADER);
          PUT DATA(NEW_FILE_HEADER)SKIP;
          WRITE FILE(NEWFILE) FROM (NEW_FILE_STATION);
          PUT DATA(NEW_FILE_STATION)SKIP;
          WRITE FILE (NEWFILE) FROM (NEW_FILE_PARAM);
          PUT DATA(NEW_FILE_PARAM)SKIP;
          ALLOCATE NEW_FILE_DATA;
          ALLOCATE PRINT_STATION;
          ALLOCATE PRINT_NAME;
          ALLOCATE PRINT_PARAM;
          N=0;
          DO I=1 TO NEW_FILE_HEADER.STATIONS;
          DO J=1 TO NEW_FILE_STATION(I).PARAMS;
          N=N+1;
          UNSPEC(PRINT_STATION(N))=UNSPEC(NEW_FILE_STATION(I).NAME)   ;
          UNSPEC(PRINT_NAME   (N))=UNSPEC(NEW_FILE_PARAM(J).NAME)   ;
          END;END;
          /* DATA UPDATE */
DCL       HOUR FIXED BINARY;
          ON ENDFILE(SOURCE) BEGIN;
          PUT LIST('END OF FILE ON SOURCE BEFORE STOP DATE REACHED')SKIP;
          GO TO STOP;END;
READ:     CALL GETDATA;
          NEW_FILE_DATA.DATE=SOURCE_DATE;
          DO HOUR=1 TO 24;
          CALL CONVERT;
          END; /* HOUR LOOP */
          WRITE FILE(NEWFILE) FROM (NEW_FILE_DATA);
          YEAR=SUBSTR(NEW_FILE_DATA.DATE,1,2);
          MONTH=SUBSTR(NEW_FILE_DATA.DATE,3,2);
          DAY  =SUBSTR(NEW_FILE_DATA.DATE,5,2);
          UNSPEC(PRINT_DATE(1))=UNSPEC(YEAR||'/'||MONTH||'/'||DAY);
          DO I=1 TO 24;
          DO J=1 TO NEW_FILE_HEADER.PARAMS;
          PRINT_PARAM(J,I)=NEW_FILE_DATA.PARAM(I,J);
          END;END;
/*        CLOSE FILE(SYSPRINT);                    */
          CALL TWODPT(PRINT_PARAM(1,1),IDUM,JDUM,PRINT_TITLE(1),
          PRINT_DATE(1),PRINT_STATION(1),PRINT_NAME(1));
          IF NEW_FILE_DATA.DATE¬=ENDING_DATE THEN GO TO READ;
GETDATA:PROC;
DCL       M1 CHAR(1), M2 CHAR(1), D1 CHAR(1), D2 CHAR(1);
```

```
        GET FILE(SOURCE) EDIT
                (D1 ,D2,M1,M2,YEAR,KEY,(SO2(N)  DO N=1 TO 12))
        (4A(1),A(2),X(1),F(1),12(F(4),F(2)));
        GET FILE(SOURCE) EDIT
                (D1 ,D2,M1,M2,YEAR,KEY,(SO2(N)  DO N=13 TO 24))
        (4A(1),A(2),X(1),F(1),12(F(4),F(2)));
        IF M1=' ' THEN M1='0';
        MONTH=M1||M2;
        IF D1=' ' THEN D1='0';
        DAY=D1||D2;
        SOURCE_DATE      =YEAR||MONTH||DAY;
        END GETDATA;
CONVERT:PROC;
        IF CHAR(HOUR)=9999 THEN GO TO BAD;
        IF CHAR(HOUR)=4021 THEN GO TO BAD;
        DUM=CHAR(HOUR);
        GO TO GOOD;
BAD:    DUM=1.E20;
GOOD:   NEW_FILE_DATA.PARAM(HOUR,1)=DUM;
        NEW_FILE_DATA.PARAM(HOUR,2)=KEY;
        END CONVERT;
STOP:   END MASTER2;
```

8. PLANTSIM Plant Emission Simulation Program

```
      SUBROUTINE TWODPT(ARAY,ICT,JCT,TITLE,DATE,COLT1,COLT2)
C
C                                TWODPT PRINTS A TWO-DIMENSIONAL ARRAY.  IT
C                               PRINTS A TITLE LINE AND TWO LINES OF COLUMN
C                               TITLES, WHICH ARE FOLLOWED BY 50 LINES UP
C                               TO TEN COLUMNS WIDE PLUS THE INDEX COLUMN,
C                               WHICH CONTAINS DATE AND HOUR.
C                               ARGUMENTS ARE -
C
C                                     ARAY - THE TWO-DIMENSIONAL ARRAY
C                                            TO BE PRINTED.
C                                      ICT - NUMBER OF ROWS IN ARRAY.
C                                      JCT - NUMBER OF COLUMNS IN ARRAY.
C                                    TITLE - PAGE TITLE (10A8).
C                                    COLT1 - ARRAY DIMENSIONED BY JCT, CON-
C                                            TAINING FIRST LINE OF TITLE
C                                            FOR EACH COLUMN (A8).
C                                    COLT2 - ARRAY DIMENSIONED BY JCT, CON-
C                                            TAINING SECOND LINE OF TITLE
C                                            FOR EACH COLUMN (A8).
C
      REAL*8 TITLE(10),DATE(1) ,COLT1(1),COLT2(1)
      REAL*4 ARAY(ICT,1)
      II=1
      IJ=1
      IMAX=0
      JMAX=0
      MAX=0
      IFLAG=0
   20 IEND=JCT
   40 MIN=MAX+1
      MAX=MINO(MAX+10,IEND)
      IF(IFLAG.NE.0) GO TO 47
   45 NROW=0
   46 CONTINUE
      PRINT 210,(TITLE(I),I=1,10)
      PRINT 220,DATE
   47 PRINT 230,(COLT1(JC),JC=MIN,MAX)
      PRINT 235,(COLT2(JC),JC=MIN,MAX)
      PRINT 235
      IMAX=MINO(II+52,ICT)
      IDIF=IMAX-II+5
      NROW=NROW+IDIF
      JMAX=MINO(IJ+9,JCT)
      DO 65 I=II,IMAX
      PRINT 240     ,              I,(ARAY(I,J),J=IJ,JMAX)
   65 CONTINUE
      IF(IMAX.GE.ICT) GO TO 70
      II=IMAX+1
      GO TO 46
   70 IF(JMAX.GE.JCT) GO TO 130
   80 II=1
      IJ=JMAX+1
      IF((NROW+IDIF-57).GT.0) GO TO 84
   82 IFLAG=1
      GO TO 40
   84 IFLAG=0
      GO TO 40
  130 CONTINUE
      DO 140 I=1,25
      PRINT 150
  150 FORMAT(' ')
  140 CONTINUE
      RETURN
  210 FORMAT(1H1,9X,10A8)
  220 FORMAT(10X'DATE:'A8)
  230 FORMAT(1H07X4HHOUR,10(4X,A8))
  235 FORMAT(12X,10(4X,A8))
  240 FORMAT(10XI2,10F12.3)
      END
APDATA: PROC OPTIONS(MAIN);
    DCL DATE_LIMITS FIXED BINARY INITIAL(0),
        LOWER_DATE(DATE_LIMITS) CHAR(8) CONTROLLED,
        UPPER_DATE(DATE_LIMITS) CHAR(8) CONTROLLED;
DCL     1 FILE_HEADER,
            2 STATIONS FIXED BINARY,
            2 PARAMS FIXED BINARY,
            2 STARTING_DATE CHAR(6),
            2 ENDING_DATE CHAR(6);
DCL     1 FILE_STATION (FILE_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
```

```
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 FILE_PARAM (FILE_HEADER.PARAMS) CONTROLLED,
            2 NAME CHAR(8);
DCL     1 TODAY      CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (FILE_HEADER.PARAMS);
DCL     1 YESTERDAY CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (FILE_HEADER.PARAMS);
DCL      CAL_POINTER     FIXED BINARY;
DCL      MET_POINTER     FIXED BINARY;
DCL      UTILITIES FIXED BINARY, TOTAL_STACKS FIXED BINARY;
DCL     1 UTIL_DATA (UTILITIES) CONTROLLED,
            2 STA_NAME CHAR(8),
            2 COORDINATES,
                3 UX,
                3 UY,
            2 STACKS FIXED BINARY,
            2 STACK (6),
                3 NUMBER CHAR(2),
                3 UZ,
            2 UNITS FIXED BINARY,
            2 PERCENT_SULPHER,
            2 TEMP_GAS_AVAIL,
            2 GAS_PRICE,
            2 UNIT (6),
                3 NUMBER CHAR (2),
                3 LOAD_POINTER FIXED BINARY,
                3 MAX_LOAD FLOAT DECIMAL,
                3 PERCENT_GAS_CONVERT,
                3 COEF_A,
                3 COEF_B,
                3 COUPLE (4),
                    4 STACK_NUMBER FIXED BINARY,
                    4 WEIGHT;
DCL     1 UTIL_FILE_CARD_1,
            2 NUMBER CHAR(3),
            2 NAME CHAR(8);
DCL     1 UTIL_FILE_CARD_2,
            2 NUMBER CHAR(3),
            2 XU CHAR(5),
            2 YU CHAR(5),
            2 NO_STACKS CHAR(2),
            2 STACK (6),
                3 NUMB CHAR(2),
                3 ZU CHAR(3),
            2 NO_UNITS CHAR(2),
            2 PER_SUL_COAL CHAR (3),
            2 AVAIL_TEMP_GAS CHAR(3),
            2 PRICE_GAS CHAR(4);
DCL     1 UTIL_FILE_CARD_3,
            2 NUMBER CHAR(3),
            2 UNIT_NO CHAR(2),
            2 LOAD_MAX CHAR(5),
            2 PER_GAS_CONVERT CHAR(3),
            2 A CHAR(6),
            2 B CHAR(6),
            2 COUPLE (4),
                3 STACK_NO CHAR(2),
                3 WT CHAR (3);
DCL      SOURCES       FIXED BINARY;
DCL     1 SRC_DATA (SOURCES) CONTROLLED,
            2 STA_NAME CHAR(8),
            2 COORDINATES,
                3 SX,
                3 SY,
            2 STACK,
                3 HEIGHT (4),
                3 PERCENT (4),
            2 WORK_SCHEDULE (7) CHAR(1),
            2 MAX_SPACE FLOAT DECIMAL,
            2 MAX_PROCESS FLOAT DECIMAL,
            2 MONTHLY_FUEL_USE (12) FLOAT DECIMAL,
            2 SHIFT_FUEL_USE,
                3 WEEKDAY (3),
                3 SATURDAY (3),
                3 HOLIDAY (3),
            2 SULPHER_COAL,
```

```
            2 PERCENT_COAL,
            2 SULPHER_OIL ,
            2 PERCENT_OIL ,
            2 PERCENT_SPACE_GAS,
            2 PERCENT_PROCESS_GAS,
            2 GAS_TEMP,
            2 GAS_COST;
DCL     1 IND_FILE_CARD_1,
            2 NUMBER CHAR(3),
            2 NAME CHAR(8);
DCL     1 IND_FILE_CARD_2,
            2 NUMBER CHAR(3),
            2 XS CHAR(5),
            2 YS CHAR(5),
            2 STACK (4),
                3 HGT CHAR(3),
                3 PER CHAR(2),
            2 WORK (7) CHAR(1);
DCL     1 IND_FILE_CARD_3,
            2 NUMBER CHAR(3),
            2 MAX_SPC CHAR(5),
            2 MAX_PRO CHAR(5),
            2 MONTHLY_FUEL_PAT (12) CHAR(1),
            2 SHIFT_FUEL_PAT,
                3 WEEK (3) CHAR (2),
                3 SAT  (3) CHAR (2),
                3 HOL  (3) CHAR (2),
            2 PERCENT_S_COAL CHAR(3),
            2 AVE_COAL CHAR(3),
            2 PERCENT_S_OIL  CHAR(3),
            2 AVE_OIL  CHAR(3),
            2 PERCENT_SPC_GAS CHAR(3),
            2 PERCENT_PRO_GAS CHAR(3),
            2 TEMP_GAS CHAR(2),
            2 COST_GAS CHAR(4);
DCL     1 OUTPUT_HEADER,
            2 LABEL (10) CHAR(8),
            2 STATIONS FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 OUTPUT_STATION CONTROLLED,
            2 NAME(OUTPUT_HEADER.STATIONS) CHAR(8),
            2 COORD_X (OUTPUT_HEADER.STATIONS),
            2 COORD_Y (OUTPUT_HEADER.STATIONS),
            2 COORD_Z (OUTPUT_HEADER.STATIONS,4),
            2 PER_EMIS(OUTPUT_HEADER.STATIONS,4);
DCL     1 OUTPUT_PARAM (OUTPUT_HEADER.PARAMS) CONTROLLED,
            2 STA_NAME CHAR(8),
            2 NAME CHAR(8);
DCL     1 OUTPUT_DATA CONTROLLED,
            2 DATE CHAR(8),
            2 HOUR (24),
                3 VALUE (OUTPUT_HEADER.PARAMS);
DCL      PRINT_TITLE (10) FLOAT DECIMAL (16),
         PRINT_DATE (1) FLOAT DECIMAL (16),
         PRINT_STATION (OUTPUT_HEADER.PARAMS) CONTROLLED
         FLOAT DECIMAL (16),
         PRINT_NAME    (OUTPUT_HEADER.PARAMS) CONTROLLED
         FLOAT DECIMAL (16),
         PRINT_PARAM (JDUM,IDUM) CONTROLLED;
DCL      CARD CHAR(80);
DCL      ENDTEST CHAR(3);
DCL      STATION CHAR(8),PARAM CHAR(8), POINT FIXED BINARY;
         PUT LIST('MASTER FILE HEADER INFORMATION')PAGE;
         READ FILE(MASTER) INTO (FILE_HEADER);
         PUT DATA(FILE_HEADER)SKIP;
         ALLOCATE FILE_STATION;
         READ FILE(MASTER) INTO (FILE_STATION);
         PUT DATA(FILE_STATION)SKIP;
         ALLOCATE FILE_PARAM;
         READ FILE(MASTER) INTO (FILE_PARAM);
         PUT DATA (FILE_PARAM)SKIP;
         ALLOCATE TODAY; ALLOCATE YESTERDAY;
         STATION='CALENDAR'; PARAM='DAY_WEEK'; CALL FIND;
         IF POINT=0 THEN GO TO STOP;
         CAL_POINTER   =POINT;
         STATION='MIDWAY'; PARAM='TEMP    '; CALL FIND;
         IF POINT=0 THEN GO TO STOP;
         MET_POINTER   =POINT;
NEXT:    /* BEGIN NEW CASE */
         CLOSE FILE(MASTER);
         ON ENDFILE(MASTER)  BEGIN;
```

```
        PUT LIST('END OF FILE SENSED ON MASTER')SKIP;
        GO TO NEXT ;END;
        READ FILE (MASTER) IGNORE(3);
        CLOSE FILE(CARDS);
GETCARD:GET EDIT(CARD)(A(80));
        WRITE FILE(CARDS) FROM (CARD);
        ENDTEST=SUBSTR(CARD,1,3);
        IF ENDTEST='END' THEN GO TO END;
        IF INDEX(CARD,'STOP')¬=0 THEN GO TO STOP;
        GO TO GETCARD;
END:    CLOSE FILE(CARDS);
        CALL INPUT;
        CALL SEARCH;
        GO TO NEXT;
INPUT:  PROC;
DCL     L1 CHAR(2),L2 CHAR(2),L3 CHAR(2),
        U1 CHAR(2),U2 CHAR(2),U3 CHAR(2);
DCL     BLANKS CHAR(68);
        ON NAME(CARDS) BEGIN;
        PUT LIST('THE FOLLOWING DATA IS OUT OF ORDER: ',DATAFIELD)SKIP;
        GO TO NEXT;END;
        ON ENDFILE(CARDS)  BEGIN;
        PUT LIST('UNEXPECTED EOF ON CARD INPUT')SKIP;
        GO TO NEXT;END;
        PUT LIST('INPUT DATA')PAGE;
        GET FILE(CARDS) EDIT(LABEL)(10A(8));
        DO I=1 TO 10;
        UNSPEC(PRINT_TITLE(I))=UNSPEC(LABEL(I));
        END;
        PUT EDIT(LABEL)(10A(8))SKIP;
        GET FILE(CARDS) DATA(DATE_LIMITS);
        FREE     LOWER_DATE; FREE     UPPER_DATE;
        IF DATE_LIMITS=0 THEN BEGIN;
        DATE_LIMITS=1;
        ALLOCATE LOWER_DATE; ALLOCATE UPPER_DATE;
        LOWER_DATE(1)=STARTING_DATE;
        UPPER_DATE(1)=ENDING_DATE;
        PUT DATA(LOWER_DATE(1),UPPER_DATE(1))SKIP;
        GO TO SKIP1;END;
        ELSE BEGIN;
        ALLOCATE LOWER_DATE; ALLOCATE UPPER_DATE;
        GET FILE(CARDS) LIST
                ((LOWER_DATE(I),UPPER_DATE(I) DO I=1 TO DATE_LIMITS));
        DO I=1 TO DATE_LIMITS;
        PUT DATA(LOWER_DATE(I),UPPER_DATE(I))SKIP;
        L1=SUBSTR(LOWER_DATE(I),1,2)  ;
        L2=SUBSTR(LOWER_DATE(I),4,2)  ;
        L3=SUBSTR(LOWER_DATE(I),7,2)  ;
        U1=SUBSTR(UPPER_DATE(I),1,2)  ;
        U2=SUBSTR(UPPER_DATE(I),4,2)  ;
        U3=SUBSTR(UPPER_DATE(I),7,2)  ;
        LOWER_DATE(I)=L1||L2||L3; UPPER_DATE(I)=U1||U2||U3;
        END;END;
SKIP1:
        PUT LIST('UTILITY REQUESTS')SKIP;
        TOTAL_STACKS=0;
        GET FILE(CARDS) DATA (UTILITIES);
        PUT DATA (UTILITIES)SKIP;
        IF UTILITIES=0 THEN GO TO SKIP2;
        FREE UTIL_DATA; ALLOCATE UTIL_DATA;
        GET FILE (CARDS) EDIT (BLANKS) (A(67));
        DO J=1 TO UTILITIES;
        GET FILE (CARDS) EDIT (UTIL_FILE_CARD_1,BLANKS)
        (A(3),X(1),A(8),A(68));
        PUT DATA(UTIL_FILE_CARD_1)SKIP;
        GET FILE (CARDS) EDIT (UTIL_FILE_CARD_2,BLANKS)
        (A(3),X(11),2A(5),X(1),A(2),X(1),6(A(2),A(3)),X(1),A(2),X(1),
        A(3),X(1),A(3),X(1),A(4),A(6));
        PUT DATA(UTIL_FILE_CARD_2)SKIP;
        IF UTIL_FILE_CARD_1.NUMBER¬= UTIL_FILE_CARD_2.NUMBER
        THEN BEGIN;
        PUT LIST ('UTILITY DATA FILE CARDS OUT OF ORDER ')SKIP;
        GO TO NEXT; END;
        UTIL_DATA.STA_NAME(J)=UTIL_FILE_CARD_1.NAME;
        UX(J)=XU;    UY(J)=YU;
        STACKS(J)=NO_STACKS; TOTAL_STACKS=TOTAL_STACKS+STACKS(J);
        DO I=1 TO 6;
        IF ZU(I)='   ' THEN UZ(J,I)=0.; ELSE UZ(J,I)=ZU(I);
        STACK(J,I).NUMBER=STACK(I).NUMB;
        END;
        PERCENT_SULPHER(J)  =PER_SUL_COAL;
```

```
        TEMP_GAS_AVAIL(J)  =AVAIL_TEMP_GAS;
        UNITS(J)=NO_UNITS;
        DO I=1 TO UNITS(J);
        GET FILE (CARDS) EDIT (UTIL_FILE_CARD_3,BLANKS)
        (A(3),X(1),A(2),X(1),A(5),X(1),A(3),X(1),2A(6),X(1),
        4(A(2),A(3)),A(30));
        PUT DATA (UTIL_FILE_CARD_3)SKIP;
        IF UTIL_FILE_CARD_1.NUMBER¬= UTIL_FILE_CARD_3.NUMBER
        THEN BEGIN;
        PUT LIST ('UTILITY DATA FILE CARDS OUT OF ORDER ')SKIP;
        GO TO NEXT; END;
        UNIT.NUMBER(J,I)=UNIT_NO;
        STATION=UTIL_DATA.STA_NAME(J); PARAM='LOAD_'||UNIT.NUMBER(J,I);
        CALL FIND; IF POINT=0 THEN GO TO NEXT;
        LOAD_POINTER(J,I)=POINT;
        MAX_LOAD(J,I)=LOAD_MAX;
        PERCENT_GAS_CONVERT(J,I)=PER_GAS_CONVERT*.01;
        COEF_A(J,I)=A;
        COEF_B(J,I)=B;
        DO M=1 TO 4;
        IF STACK_NO(M)='   ' THEN BEGIN;
        STACK_NUMBER(J,I,M)=0; WEIGHT(J,I,M)=0.; END;
        ELSE BEGIN;
        STACK_NUMBER(J,I,M)=STACK_NO(M); WEIGHT(J,I,M)=WT(M)*.01; END;
        END;
        END; /* UNIT LOOP */
        END; /* UTILITY LOOP */
        PUT DATA(UTIL_DATA)SKIP;
SKIP2:
        PUT LIST('SOURCE REQUESTS')SKIP;
        GET FILE(CARDS) DATA(SOURCES);
        PUT DATA(SOURCES)SKIP;
        IF SOURCES+UTILITIES=0 THEN BEGIN;
        PUT LIST ('NO SOURCES OR UTILITIES REQUESTED')SKIP;
        GO TO NEXT; END;
        IF SOURCES=0 THEN BEGIN;
        PUT LIST(' NO SOURCE REQUESTS')SKIP;
        GO TO SKIP3; END;
        FREE SRC_DATA; ALLOCATE SRC_DATA;
        GET FILE (CARDS) EDIT (BLANKS) (A(68));
        DO J=1 TO SOURCES;
        GET FILE (CARDS) EDIT (IND_FILE_CARD_1,BLANKS)
        (A(3),X(1),A(8),A(68));
        PUT DATA(IND_FILE_CARD_1)SKIP;
        GET FILE (CARDS) EDIT (IND_FILE_CARD_2,BLANKS)
        (A(3),X(11),2A(5),X(1),4(A(3),A(2)),X(1),7A(1),A(27));
        PUT DATA(IND_FILE_CARD_2)SKIP;
        GET FILE (CARDS) EDIT (IND_FILE_CARD_3,BLANKS)
        (A(3),2(X(1),A(5)),X(1),12A(1),X(1),9A(2),6(X(1),A(3)),
        X(1),A(2),X(1),A(4),A(1));
        PUT DATA(IND_FILE_CARD_3)SKIP;
        IF IND_FILE_CARD_1.NUMBER¬= IND_FILE_CARD_2.NUMBER |
           IND_FILE_CARD_1.NUMBER¬= IND_FILE_CARD_3.NUMBER THEN BEGIN;
        PUT LIST('INDUSTRIAL DATA FILE CARDS OUT OT ORDER')SKIP;
        GO TO NEXT; END;
        SRC_DATA.STA_NAME(J)=IND_FILE_CARD_1.NAME;
        SX(J)=XS;     SY(J)=YS;
        DO I=1 TO 4;
        IF HGT(I)='   ' THEN HEIGHT(J,I)=0.; ELSE HEIGHT(J,I)=HGT(I);
        IF PER(I)='   ' THEN PERCENT(J,I)=0.; ELSE PERCENT(J,I)=
        PER(I)*.01;
        END;
        DO I=1 TO 7;
        WORK_SCHEDULE(J,I)=WORK(I);
        END;
        MAX_SPACE(J)=MAX_SPC; MAX_PROCESS(J)=MAX_PRO;
        DO I=1 TO 12;
        MONTHLY_FUEL_USE(J,I)=MONTHLY_FUEL_PAT(I)*.1;
        END;
        DO I=1 TO 3;
        WEEKDAY(J,I)=WEEK(I)*.01;
        SATURDAY(J,I)=SAT(I)*.01;
        HOLIDAY(J,I)=HOL(I)*.01;
        END;
        SULPHER_COAL(J)=PERCENT_S_COAL;
        SULPHER_OIL (J)=PERCENT_S_OIL ;
        PERCENT_COAL(J)=AVE_COAL*.01;
        PERCENT_OIL (J)=AVE_OIL *.01;
        PERCENT_SPACE_GAS(J)=PERCENT_SPC_GAS*.01;
        PERCENT_PROCESS_GAS(J)=PERCENT_PRO_GAS*.01;
        GAS_TEMP(J)=TEMP_GAS;
```

```
        END; /* SOURCE LOOP */
        PUT DATA(SRC_DATA)SKIP;
SKIP3:
        OUTPUT_HEADER.STATIONS=TOTAL_STACKS+SOURCES;
        FREE      OUTPUT_STATION;
        ALLOCATE OUTPUT_STATION;
        OUTPUT_HEADER.PARAMS=2*(TOTAL_STACKS+SOURCES);
        FREE      OUTPUT_PARAM;
        ALLOCATE OUTPUT_PARAM;
        IF UTILITIES=0 THEN GO TO SKIP4;
        K=0;
        DO J=1 TO UTILITIES;
        DO L=1 TO STACKS(J);
        K=K+1;
DCL     DUM_CHAR CHAR(5);
        DUM_CHAR=SUBSTR(UTIL_DATA.STA_NAME(J),1,5);
        OUTPUT_STATION.NAME(K)=DUM_CHAR||'_'||STACK(J,L).NUMBER;
        OUTPUT_STATION.COORD_X(K)=UX(J);
        OUTPUT_STATION.COORD_Y(K)=UY(J);
        OUTPUT_STATION.COORD_Z(K,1)=UZ(J,L);
        OUTPUT_STATION.PER_EMIS(K,1)=1.0;
        DO M=2 TO 4;
        OUTPUT_STATION.COORD_Z(K,M)=0.;
        OUTPUT_STATION.PER_EMIS(K,M)=0.;
        END; /* STACK LOOP */
        OUTPUT_PARAM.STA_NAME(2*(K-1)+1)=OUTPUT_STATION.NAME(K);
        OUTPUT_PARAM.STA_NAME(2*(K-1)+2)=OUTPUT_STATION.NAME(K);
        OUTPUT_PARAM.NAME(2*(K-1)+1)='Q_SO2';
        OUTPUT_PARAM.NAME(2*(K-1)+2)='Q_HEAT';
        END; /* UNIT LOOP */
        END; /* UTILITY LOOP */
SKIP4:
        IF SOURCES=0 THEN GO TO SKIP5;
        DO J=1 TO SOURCES;
        OUTPUT_STATION.NAME(TOTAL_STACKS+J)=SRC_DATA.STA_NAME(J);
        OUTPUT_STATION.COORD_X(TOTAL_STACKS+J)=SX(J);
        OUTPUT_STATION.COORD_Y(TOTAL_STACKS+J)=SY(J);
        DO I=1 TO 4;
        OUTPUT_STATION.COORD_Z(TOTAL_STACKS+J,I)=HEIGHT(J,I);
        OUTPUT_STATION.PER_EMIS(TOTAL_STACKS+J,I)=PERCENT(J,I);
        END; /* STACK LOOP */
        OUTPUT_PARAM.STA_NAME(2*(TOTAL_STACKS+(J-1))+1)=
        SRC_DATA.STA_NAME(J);
        OUTPUT_PARAM.STA_NAME(2*(TOTAL_STACKS+(J-1))+2)=
        SRC_DATA.STA_NAME(J);
        OUTPUT_PARAM.NAME     (2*(TOTAL_STACKS+(J-1))+1)=
        'Q_SO2';
        OUTPUT_PARAM.NAME     (2*(TOTAL_STACKS+(J-1))+2)=
        'Q_HEAT';
        END; /* SOURCE LOOP */
SKIP5:
        FREE      PRINT_STATION;
        ALLOCATE PRINT_STATION;
        FREE      PRINT_NAME;
        ALLOCATE PRINT_NAME;
        DO J=1 TO OUTPUT_HEADER.PARAMS;
        STATION=OUTPUT_PARAM(J).STA_NAME;
        PARAM  =OUTPUT_PARAM(J).NAME    ;
        UNSPEC(PRINT_STATION(J))=UNSPEC(STATION);
        UNSPEC(PRINT_NAME   (J))=UNSPEC(PARAM);
        END;
        RETURN;
        END INPUT;
SEARCH: PROC;
DCL     HOUR FIXED BINARY, DAY_WEEK FIXED BINARY;
DCL     DATE_KEY BIT(1) INITIAL('0'B), HOUR_KEY BIT(1) INITIAL ('0'B);
DCL     DATE_LIMIT FIXED BINARY, HOUR_LIMIT FIXED BINARY;
DCL     YEAR CHAR(2), MONTH CHAR(2), DAY CHAR(2);
DCL     UTILITY CHAR(8), UNIT_NO CHAR(2);
DCL     LOAD (UTILITIES,6,24) CONTROLLED FLOAT DECIMAL,
        STACK_SO2(UTILITIES,6) CONTROLLED,
        STACK_HEAT(UTILITIES,6) CONTROLLED;
        IDUM=24; JDUM=OUTPUT_HEADER.PARAMS;
        DATE_LIMIT=1;
        FREE LOAD; ALLOCATE LOAD;
        FREE STACK_SO2; ALLOCATE STACK_SO2;
        FREE STACK_HEAT; ALLOCATE STACK_HEAT;
        FREE OUTPUT_DATA; ALLOCATE OUTPUT_DATA;
        FREE PRINT_PARAM; ALLOCATE PRINT_PARAM;
        WRITE FILE(OUTPUT) FROM(OUTPUT_HEADER);
        WRITE FILE(OUTPUT) FROM(OUTPUT_STATION);
```

```
        WRITE FILE(OUTPUT) FROM(OUTPUT_PARAM);
        ON ENDFILE(MASTER)   BEGIN;
        PUT LIST('END OF FILE SENSED ON MASTER')SKIP;
        GO TO WRITE;END;
        PUT LIST('SEARCH STARTED  AT TIME ',TIME)SKIP;
DATES:  /* LOOP ON DATES - TEST BETWEEN DATE LIMITS */
        READ FILE(MASTER) INTO (TODAY);
        IF TODAY.DATE=LOWER_DATE(DATE_LIMIT) THEN BEGIN;
        DATE_KEY='1'B; END;
        IF TODAY.DATE=UPPER_DATE(DATE_LIMIT) THEN BEGIN;
        DATE_KEY='0'B;DATE_LIMIT=DATE_LIMIT+1; GO TO HOURS;
        END;
        IF DATE_KEY='0'B THEN GO TO NEWDATE;
HOURS:  /* LOOP ON HOURS */
        YEAR =SUBSTR(TODAY.DATE,1,2);
        MONTH=SUBSTR(TODAY.DATE,3,2);
        DAY  =SUBSTR(TODAY.DATE,5,2);
        OUTPUT_DATA.DATE=YEAR||'/'||MONTH||'/'||DAY;
        UNSPEC(PRINT_DATE(1))=UNSPEC(OUTPUT_DATA.DATE);
        MON=MONTH;
        DO HOUR=1 TO 24;
        DAY_WEEK=TODAY(HOUR).PARAM(CAL_POINTER);
        NO_SHIFT=(HOUR-1)/8; NO_SHIFT=NO_SHIFT+1;
        TEMP=TODAY(HOUR).PARAM(MET_POINTER);
        PUT DATA(MON,DAY_WEEK,NO_SHIFT,TEMP)SKIP;
        IF UTILITIES=0 THEN GO TO JUMP6;
        K=0; STACK_SO2=0.; STACK_HEAT=0.;
        DO J=1 TO UTILITIES;
        IF TEMP>TEMP_GAS_AVAIL(J) & MON >=4 & MON <= 10 THEN
        GAS_FLAG=1.0; ELSE GAS_FLAG=0.;
        PUT DATA (GAS_FLAG)SKIP;
        DO I=1 TO UNITS(J);
        LOAD(J,I,HOUR)=TODAY(HOUR).PARAM(LOAD_POINTER(J,I));
        PUT DATA(UNIT.NUMBER(J,I),LOAD(J,I,HOUR))SKIP;
        IF LOAD(J,I,HOUR)=0. THEN GO TO JUMP5;
        UNIT_THERMS=COEF_A(J,I)*LOAD(J,I,HOUR)+COEF_B(J,I);
        IF UNIT_THERMS<0. THEN UNIT_THERMS=0.;
        PUT DATA (UNIT_THERMS)SKIP;
        UNIT_THERMS_GAS=(COEF_A(J,I)*MAX_LOAD(J,I)+COEF_B(J,I))*
        PERCENT_GAS_CONVERT(J,I)*GAS_FLAG;
        IF UNIT_THERMS_GAS < 0. THEN UNIT_THERMS_GAS=0.;
        IF UNIT_THERMS_GAS>UNIT_THERMS THEN
        UNIT_THERMS_GAS=UNIT_THERMS;
        PUT DATA(UNIT_THERMS_GAS)SKIP;
        Q_HEAT=.15*UNIT_THERMS;
        PUT DATA(Q_HEAT)SKIP;
        UNIT_THERMS_COAL=UNIT_THERMS-UNIT_THERMS_GAS;
        COAL=UNIT_THERMS_COAL*1.E5/(12000.*2000.);
        PUT DATA(UNIT_THERMS_COAL,COAL)SKIP;
        Q_SO2=COAL*36.8*PERCENT_SULPHER(J);
        PUT DATA(Q_SO2 )SKIP;
        DO M=1 TO 4;
        IF STACK_NUMBER(J,I,M)=0  THEN GO TO JUMP5;
        DO L=1 TO STACKS(J);
        IF STACK_NUMBER(J,I,M)= STACK.NUMBER(J,L) THEN GO TO JUMP4;
        END; /* STACK SEARCH LOOP */
        PUT LIST('STACK NUMBER ',STACK_NUMBER(J,I,M),' CANNOT BE FOUND'
       ,' FOR UNIT NUMBER ',UNIT.NUMBER(J,I),
       ' AND UTILITY ',UTIL_DATA.STA_NAME(J))SKIP; RETURN;
JUMP4:
        STACK_SO2(J,L)=STACK_SO2(J,L)+WEIGHT(J,I,M)*Q_SO2;
        STACK_HEAT(J,L)=STACK_HEAT(J,L)+WEIGHT(J,I,M)*Q_HEAT;
        PUT DATA(STACK.NUMBER(J,L),STACK_SO2(J,L),STACK_HEAT(J,L))SKIP;
        END; /* UNIT - STACK COUPLE LOOP */
JUMP5:
        END; /* UNIT LOOP */
        DO L=1 TO STACKS(J);
        K=K+1;
        OUTPUT_DATA.VALUE(HOUR,2*(K-1)+1)=STACK_SO2(J,L);
        OUTPUT_DATA.VALUE(HOUR,2*(K-1)+2)=STACK_HEAT(J,L);
        PRINT_PARAM(2*(K-1)+1,HOUR)=STACK_SO2(J,L);
        PRINT_PARAM(2*(K-1)+2,HOUR)=STACK_HEAT(J,L);
        END; 7* STACK LOOP */
        END; /* UTILITY LOOP */
JUMP6:
        IF SOURCES=0 THEN GO TO JUMP7;
        DO J=1 TO SOURCES;
        IF TEMP>= GAS_TEMP(J) & MON>=4 & MON <=10
        THEN GAS_FLAG=1.0; ELSE GAS_FLAG=0.;
        PUT DATA(GAS_FLAG)SKIP;
        IF TEMP> 55. THEN THERMS_SPACE=0.; ELSE BEGIN;
```

```
        IF TEMP<-10. THEN THERMS_SPACE=MAX_SPACE(J);
        ELSE THERMS_SPACE=MAX_SPACE(J)*(55.-TEMP)/65.; END;
        THERMS_SPACE_GAS = PERCENT_SPACE_GAS(J)*MAX_SPACE(J)*GAS_FLAG;
        IF THERMS_SPACE_GAS>=THERMS_SPACE THEN
        THERMS_SPACE_GAS=THERMS_SPACE;
        PUT DATA(THERMS_SPACE,THERMS_SPACE_GAS)SKIP;
        IF DAY_WEEK >7 THEN BEGIN;
        THERMS_PROCESS=MAX_PROCESS(J)*MONTHLY_FUEL_USE(J,MON)*
                        HOLIDAY(J,NO_SHIFT);
        GO TO CONT1; END;
        IF WORK_SCHEDULE(J,DAY_WEEK)='W' THEN BEGIN;
        THERMS_PROCESS=MAX_PROCESS(J)*MONTHLY_FUEL_USE(J,MON)*
                        WEEKDAY(J,NO_SHIFT);
        GO TO CONT1; END;
        IF WORK_SCHEDULE(J,DAY_WEEK)='S' THEN BEGIN;
        THERMS_PROCESS=MAX_PROCESS(J)*MONTHLY_FUEL_USE(J,MON)*
                        SATURDAY(J,NO_SHIFT);
        GO TO CONT1; END;
        IF WORK_SCHEDULE(J,DAY_WEEK)='H' THEN BEGIN;
        THERMS_PROCESS=MAX_PROCESS(J)*MONTHLY_FUEL_USE(J,MON)*
                        HOLIDAY(J,NO_SHIFT);
        END;
CONT1:
        THERMS_PROCESS_GAS=PERCENT_PROCESS_GAS(J)*MAX_PROCESS(J)*
        GAS_FLAG;
        IF THERMS_PROCESS_GAS>=THERMS_PROCESS THEN
        THERMS_PROCESS_GAS=THERMS_PROCESS;
        PUT DATA(THERMS_PROCESS,THERMS_PROCESS_GAS)SKIP;
        Q_HEAT=.15*(THERMS_SPACE+THERMS_PROCESS);
        PUT DATA(Q_HEAT)SKIP;
        THERMS_COAL=PERCENT_COAL(J)*
        ((THERMS_SPACE-THERMS_SPACE_GAS)+
        (THERMS_PROCESS-THERMS_PROCESS_GAS));
        THERMS_OIL =PERCENT_OIL (J)*
        ((THERMS_SPACE-THERMS_SPACE_GAS)+
        (THERMS_PROCESS-THERMS_PROCESS_GAS));
        PUT DATA(THERMS_COAL,THERMS_OIL)SKIP;
        COAL=THERMS_COAL*1.E5/(12000.*2000.);
        OIL=THERMS_OIL*1.E5/(18000.*8000.);
        PUT DATA(COAL,OIL)SKIP;
        SO2_COAL=COAL*36.8*SULPHER_COAL(J);
        SO2_OIL=OIL*157.*SULPHER_OIL(J);
        PUT DATA(SO2_COAL,SO2_OIL)SKIP;
        Q_SO2  =SO2_COAL+SO2_OIL;
        PUT DATA(Q_SO2)SKIP;
        OUTPUT_DATA.VALUE(HOUR,2*(TOTAL_STACKS+(J-1))+1)=Q_SO2;
        OUTPUT_DATA.VALUE(HOUR,2*(TOTAL_STACKS+(J-1))+2)=Q_HEAT;
        PRINT_PARAM(2*(TOTAL_STACKS+(J-1))+1,HOUR)=Q_SO2;
        PRINT_PARAM(2*(TOTAL_STACKS+(J-1))+2,HOUR)=Q_HEAT;
        END; /* SOURCE LOOP */
JUMP7:
        END; /* HOUR LOOP */
        WRITE FILE(OUTPUT) FROM(OUTPUT_DATA);
        CLOSE FILE(SYSPRINT);
        CALL TWODPT(PRINT_PARAM(1,1),IDUM,JDUM,PRINT_TITLE(1),
        PRINT_DATE(1),PRINT_STATION(1),PRINT_NAME(1));
NEWDATE:IF DATE_LIMIT>DATE_LIMITS THEN GO TO WRITE;
        GO TO DATES;
WRITE:  /* SEARCH FINISHED */
        PUT LIST('SEARCH FINISHED AT TIME ',TIME)SKIP;
        RETURN;
        END SEARCH;
FIND:   PROC;
        DO N=1 TO FILE_HEADER.STATIONS;
        IF STATION=FILE_STATION(N).NAME THEN BEGIN;
        DO M=1 TO FILE_STATION(N).PARAMS;
        IF PARAM=FILE_PARAM(FILE_STATION(N).POINTER+M).NAME
        THEN GO TO FINDIT;
        END;/*M LOOP*/ END;/* BEGIN BLOCK*/
        END; /*N LOOP*/
        PUT LIST('THE COMBINATION STATION NAME ',STATION,
        ' AND PARAMETER NAME ',PARAM,' CANNOT BE FOUND')SKIP;
        POINT=0; RETURN;
FINDIT: POINT=FILE_STATION(N).POINTER+M;
        PUT DATA(STATION,PARAM,POINT)SKIP;
        RETURN;
        END FIND;
STOP:   END APDATA;
```

APICS USER MANUAL

APPENDIX D

MASTER TAPE MAINTENANCE PROGRAMS
SOURCE LISTINGS

1. MERGE1 Merge Program for Tapes with ~ Same Dates

```
MERGE:  PROC OPTIONS (MAIN);
DCL     1 OLD_FILE_1_HEADER,
            2 STATIONS FIXED BINARY,
            2 PARAMS FIXED BINARY,
            2 STARTING_DATE CHAR(6),
            2 ENDING_DATE CHAR(6);
DCL     1 OLD_FILE_2_HEADER,
            2 STATIONS FIXED BINARY,
            2 PARAMS FIXED BINARY,
            2 STARTING_DATE CHAR(6),
            2 ENDING_DATE CHAR(6);
DCL     1 NEW_FILE_HEADER,
            2 STATIONS FIXED BINARY,
            2 PARAMS FIXED BINARY,
            2 STARTING_DATE CHAR(6),
            2 ENDING_DATE CHAR(6);
DCL     1 OLD_FILE_1_STATION (OLD_FILE_1_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 OLD_FILE_2_STATION (OLD_FILE_2_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 NEW_FILE_STATION (NEW_FILE_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 OLD_FILE_1_PARAM (OLD_FILE_1_HEADER.PARAMS) CONTROLLED,
            2 NAME CHAR(8);
DCL     1 OLD_FILE_2_PARAM (OLD_FILE_2_HEADER.PARAMS) CONTROLLED,
            2 NAME CHAR(8);
DCL     1 NEW_FILE_PARAM   (NEW_FILE_HEADER.PARAMS   ) CONTROLLED,
            2 NAME CHAR(8);
DCL     1 OLD_FILE_1_DATA CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (OLD_FILE_1_HEADER.PARAMS);
DCL     1 OLD_FILE_2_DATA CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (OLD_FILE_2_HEADER.PARAMS);
DCL     1 NEW_FILE_DATA CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (NEW_FILE_HEADER.PARAMS);
        PUT LIST('NEW FILE HEADER INFORMATION')PAGE;
        /* FILE HEADER UPDATE */
        READ FILE(OLDFIL1) INTO (OLD_FILE_1_HEADER);
        READ FILE(OLDFIL2) INTO (OLD_FILE_2_HEADER);
        NEW_FILE_HEADER.STATIONS=OLD_FILE_1_HEADER.STATIONS+
                                 OLD_FILE_2_HEADER.STATIONS;
        NEW_FILE_HEADER.PARAMS=OLD_FILE_1_HEADER.PARAMS+
                               OLD_FILE_2_HEADER.PARAMS;
        IF OLD_FILE_1_HEADER.STARTING_DATE¬=
           OLD_FILE_2_HEADER.STARTING_DATE THEN BEGIN;
        PUT LIST('STARTING DATES ON OLD FILES DO NOT AGREE')SKIP;
        GO TO STOP;END;
/*      IF OLD_FILE_1_HEADER.ENDING_DATE  ¬=
           OLD_FILE_2_HEADER.ENDING_DATE   THEN BEGIN;
        PUT LIST('ENDING   DATES ON OLD FILES DO NOT AGREE')SKIP;
        GO TO STOP;END;      */
        NEW_FILE_HEADER.STARTING_DATE=OLD_FILE_1_HEADER.STARTING_DATE;
        NEW_FILE_HEADER.ENDING_DATE=OLD_FILE_1_HEADER.ENDING_DATE;
        WRITE FILE(NEWFILE) FROM (NEW_FILE_HEADER);
        PUT DATA(NEW_FILE_HEADER)SKIP;
        /* FILE STATION UPDATE */
        ALLOCATE OLD_FILE_1_STATION;
        ALLOCATE OLD_FILE_2_STATION;
        ALLOCATE NEW_FILE_STATION;
        READ FILE(OLDFIL1) INTO (OLD_FILE_1_STATION);
        READ FILE(OLDFIL2) INTO (OLD_FILE_2_STATION);
        DO I=1 TO OLD_FILE_1_HEADER.STATIONS;
        NEW_FILE_STATION(I)=OLD_FILE_1_STATION(I)      ;
        END;
        DO I=1 TO OLD_FILE_2_HEADER.STATIONS;
        NEW_FILE_STATION(OLD_FILE_1_HEADER.STATIONS+I).NAME=
        OLD_FILE_2_STATION(I).NAME;
        NEW_FILE_STATION(OLD_FILE_1_HEADER.STATIONS+I).POINTER=
        OLD_FILE_1_HEADER.PARAMS+OLD_FILE_2_STATION(I).POINTER;
        NEW_FILE_STATION(OLD_FILE_1_HEADER.STATIONS+I).PARAMS=
```

```
        OLD_FILE_2_STATION(I).PARAMS;
        END;
        WRITE FILE(NEWFILE) FROM (NEW_FILE_STATION);
        PUT DATA(NEW_FILE_STATION)SKIP;
        /* FILE PARAMETER UPDATE */
        ALLOCATE OLD_FILE_1_PARAM;
        ALLOCATE OLD_FILE_2_PARAM;
        ALLOCATE NEW_FILE_PARAM;
        READ FILE(OLDFIL1) INTO (OLD_FILE_1_PARAM);
        READ FILE(OLDFIL2) INTO (OLD_FILE_2_PARAM);
        DO J=1 TO OLD_FILE_1_HEADER.PARAMS;
        NEW_FILE_PARAM(J)=OLD_FILE_1_PARAM(J);
        END;
        DO J=1 TO OLD_FILE_2_HEADER.PARAMS;
        NEW_FILE_PARAM(OLD_FILE_1_HEADER.PARAMS+J).NAME=
        OLD_FILE_2_PARAM(J).NAME;
        END;
        WRITE FILE (NEWFILE) FROM (NEW_FILE_PARAM);
        PUT DATA(NEW_FILE_PARAM)SKIP;
        /* DATA UPDATE */
DCL     HOUR FIXED BINARY;
DCL     JUMP BINARY;
        JUMP='0'B;
        ALLOCATE OLD_FILE_1_DATA;
        ALLOCATE OLD_FILE_2_DATA;
        ALLOCATE NEW_FILE_DATA;
        ON ENDFILE(OLDFIL1) BEGIN;
        PUT LIST('UNEXPECTED END OF FILE ON OLDFIL1')SKIP;
        GO TO STOP; END;
        ON ENDFILE(OLDFIL2) BEGIN;
        PUT LIST('UNEXPECTED END OF FILE ON OLDFIL2')SKIP;
        JUMP='1'B; GO TO CONT; END;
READ:   READ FILE(OLDFIL1) INTO (OLD_FILE_1_DATA);
        PUT DATA(OLD_FILE_1_DATA.DATE)SKIP;
        IF JUMP='1'B THEN GO TO CONT;
        READ FILE(OLDFIL2) INTO (OLD_FILE_2_DATA);
        IF OLD_FILE_1_DATA.DATE¬= OLD_FILE_2_DATA.DATE THEN BEGIN;
        PUT LIST('DATES ON OLD FILES DO NOT AGREE')SKIP;
        GO TO STOP; END;
CONT:   NEW_FILE_DATA.DATE=OLD_FILE_1_DATA.DATE;
        DO HOUR=1 TO 24;
        DO J=1 TO OLD_FILE_1_HEADER.PARAMS;
        NEW_FILE_DATA.PARAM(HOUR,J)=OLD_FILE_1_DATA.PARAM(HOUR,J);
        END;
        DO J=1 TO OLD_FILE_2_HEADER.PARAMS;
        IF JUMP='1'B THEN
        NEW_FILE_DATA.PARAM(HOUR,OLD_FILE_1_HEADER.PARAMS+J)=1.E20;
        ELSE
        NEW_FILE_DATA.PARAM(HOUR,OLD_FILE_1_HEADER.PARAMS+J)=
        OLD_FILE_2_DATA.PARAM(HOUR,J);
        END;
        END; /* HOUR LOOP */
        WRITE FILE(NEWFILE) FROM (NEW_FILE_DATA);
        IF NEW_FILE_DATA.DATE¬=NEW_FILE_HEADER.ENDING_DATE THEN
        GO TO READ;
STOP:   END MERGE;
```

2. MERGE2 Merge Program for Concatenating Two Tapes

```
MERGE:  PROC OPTIONS (MAIN);
DCL     1 OLD_FILE_1_HEADER,
            2 STATIONS FIXED BINARY,
            2 PARAMS FIXED BINARY,
            2 STARTING_DATE CHAR(6),
            2 ENDING_DATE CHAR(6);
DCL     1 OLD_FILE_2_HEADER,
            2 STATIONS FIXED BINARY,
            2 PARAMS FIXED BINARY,
            2 STARTING_DATE CHAR(6),
            2 ENDING_DATE CHAR(6);
DCL     1 NEW_FILE_HEADER,
            2 STATIONS FIXED BINARY,
            2 PARAMS FIXED BINARY,
            2 STARTING_DATE CHAR(6),
            2 ENDING_DATE CHAR(6);
DCL     1 OLD_FILE_1_STATION (OLD_FILE_1_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 OLD_FILE_2_STATION (OLD_FILE_2_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 NEW_FILE_STATION (NEW_FILE_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 OLD_FILE_1_PARAM (OLD_FILE_1_HEADER.PARAMS) CONTROLLED,
            2 NAME CHAR(8);
DCL     1 OLD_FILE_2_PARAM (OLD_FILE_2_HEADER.PARAMS) CONTROLLED,
            2 NAME CHAR(8);
DCL     1 NEW_FILE_PARAM   (NEW_FILE_HEADER.PARAMS   ) CONTROLLED,
            2 NAME CHAR(8);
DCL     1 OLD_FILE_1_DATA CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (OLD_FILE_1_HEADER.PARAMS);
DCL     1 OLD_FILE_2_DATA CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (OLD_FILE_2_HEADER.PARAMS);
DCL     1 NEW_FILE_DATA CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (NEW_FILE_HEADER.PARAMS);
         PUT LIST('NEW FILE HEADER INFORMATION')PAGE;
         /* FILE HEADER UPDATE */
         READ FILE(OLDFIL1) INTO (OLD_FILE_1_HEADER);
         READ FILE(OLDFIL2) INTO (OLD_FILE_2_HEADER);
         IF OLD_FILE_1_HEADER.ENDING_DATE¬=
         OLD_FILE_2_HEADER.STARTING_DATE
         THEN BEGIN;
         PUT LIST('ENDING DATE ON OLD FILE 1 DOES NOT AGREE WITH STARTIN
         G DATE ON OLD FILE 2')SKIP; END;
         NEW_FILE_HEADER.STARTING_DATE=OLD_FILE_1_HEADER.STARTING_DATE;
         NEW_FILE_HEADER.ENDING_DATE=OLD_FILE_2_HEADER.ENDING_DATE;
         IF OLD_FILE_1_HEADER.STATIONS¬= OLD_FILE_2_HEADER.STATIONS
         THEN BEGIN;
         PUT LIST('NUMBER OF STATIONS ON OLD FILES DO NOT AGREE')SKIP;
         GO TO STOP; END;
         NEW_FILE_HEADER.STATIONS=OLD_FILE_1_HEADER.STATIONS;
         IF OLD_FILE_1_HEADER.PARAMS  ¬= OLD_FILE_2_HEADER.PARAMS
         THEN BEGIN;
         PUT LIST('NUMBER OF PARAMS   ON OLD FILES DO NOT AGREE')SKIP;
         GO TO STOP; END;
         NEW_FILE_HEADER.PARAMS=OLD_FILE_1_HEADER.PARAMS;
         WRITE FILE(NEWFILE) FROM (NEW_FILE_HEADER);
         PUT DATA(NEW_FILE_HEADER)SKIP;
         /* FILE STATION UPDATE */
         ALLOCATE OLD_FILE_1_STATION;
         ALLOCATE OLD_FILE_2_STATION;
         ALLOCATE NEW_FILE_STATION;
         READ FILE(OLDFIL1) INTO (OLD_FILE_1_STATION);
         READ FILE(OLDFIL2) INTO (OLD_FILE_2_STATION);
         DO I=1 TO OLD_FILE_1_HEADER.STATIONS;
         NEW_FILE_STATION(I)=OLD_FILE_1_STATION(I)      ;
         END;
         WRITE FILE(NEWFILE) FROM (NEW_FILE_STATION);
         PUT DATA(NEW_FILE_STATION)SKIP;
         /* FILE PARAMETER UPDATE */
```

```
        ALLOCATE OLD_FILE_1_PARAM;
        ALLOCATE OLD_FILE_2_PARAM;
        ALLOCATE NEW_FILE_PARAM;
        READ FILE(OLDFIL1) INTO (OLD_FILE_1_PARAM);
        READ FILE(OLDFIL2) INTO (OLD_FILE_2_PARAM);
        DO J=1 TO OLD_FILE_1_HEADER.PARAMS;
        NEW_FILE_PARAM(J)=OLD_FILE_1_PARAM(J);
        END;
        WRITE FILE (NEWFILE) FROM (NEW_FILE_PARAM);
        PUT DATA(NEW_FILE_PARAM)SKIP;
        /* DATA UPDATE */
DCL     JUMP BINARY;
        JUMP='0'B;
        ALLOCATE OLD_FILE_1_DATA;
        ALLOCATE OLD_FILE_2_DATA;
        ALLOCATE NEW_FILE_DATA;
        ON ENDFILE(OLDFIL1) BEGIN;
        PUT LIST('EOF REACHED ON OLD FILE 1')SKIP;
        JUMP='1'B; GO TO READ2; END;
        ON ENDFILE(OLDFIL2) BEGIN;
        PUT LIST('UNEXPECTED END OF FILE ON OLDFIL2')SKIP;
        GO TO STOP; END;
READ1:  READ FILE(OLDFIL1) INTO (OLD_FILE_1_DATA);
        PUT DATA(OLD_FILE_1_DATA.DATE)SKIP;
        NEW_FILE_DATA=OLD_FILE_1_DATA;
        GO TO WRITE;
READ2:  READ FILE(OLDFIL2) INTO (OLD_FILE_2_DATA);
        PUT DATA(OLD_FILE_2_DATA.DATE)SKIP;
        NEW_FILE_DATA=OLD_FILE_2_DATA;
WRITE:  WRITE FILE(NEWFILE) FROM (NEW_FILE_DATA);
        IF NEW_FILE_DATA.DATE¬=NEW_FILE_HEADER.ENDING_DATE THEN BEGIN;
        IF JUMP='0'B THEN GO TO READ1; ELSE GO TO READ2;  END;
STOP:   END MERGE;
```

3. MERGE3 Merge Program for Tapes with Overlapping Parameters

```
MERGE: PROC OPTIONS (MAIN);
DECLARE 1 OLD_FILE_1_HEADER,
            2 STATIONS FIXED BINARY,
            2 PARAMS FIXED BINARY,
            2 STARTING_DATE CHAR(6),
            2 ENDING_DATE CHAR(6),
        1 OLD_FILE_2_HEADER,
            2 STATIONS FIXED BINARY,
            2 PARAMS FIXED BINARY,
            2 STARTING_DATE CHAR(6),
            2 ENDING_DATE CHAR(6),
        1 NEW_FILE_HEADER,
            2 STATIONS FIXED BINARY,
            2 PARAMS FIXED BINARY,
            2 STARTING_DATE CHAR(6),
            2 ENDING_DATE CHAR(6);
DECLARE 1 OLD_FILE_1_STATION(OLD_FILE_1_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY,
        1 OLD_FILE_2_STATION(OLD_FILE_2_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY,
        1 NEW_FILE_STATION(NEW_FILE_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DECLARE 1 OLD_FILE_1_PARAM(OLD_FILE_1_HEADER.PARAMS) CONTROLLED,
            2 NAME CHAR(8),
        1 OLD_FILE_2_PARAM(OLD_FILE_2_HEADER.PARAMS) CONTROLLED,
            2 NAME CHAR(8),
        1 NEW_FILE_PARAM(NEW_FILE_HEADER.PARAMS) CONTROLLED,
            2 NAME CHAR(8);
DECLARE DUPLICATE_STATION(OLD_FILE_1_HEADER.STATIONS) FIXED BINARY
            CONTROLLED,
        NEW_STATION(NUMBER_NEW_STATIONS) FIXED BINARY
            CONTROLLED;
DECLARE POINTER_FROM_OLD_FILE_1(OLD_FILE_1_HEADER.PARAMS) FIXED BINARY
            CONTROLLED,
        POINTER_FROM_OLD_FILE_2(OLD_FILE_2_HEADER.PARAMS) FIXED BINARY
            CONTROLLED,
        DUPLICATE_POINTER(OLD_FILE_2_HEADER.PARAMS) FIXED BINARY
                CONTROLLED;
DECLARE 1 OLD_FILE_1_DATA CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM(OLD_FILE_1_HEADER.PARAMS),
        1 OLD_FILE_2_DATA CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM(OLD_FILE_2_HEADER.PARAMS),
        1 NEW_FILE_DATA CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM(NEW_FILE_HEADER.PARAMS);
DECLARE DATE_NOW CHAR(6);
DECLARE READ_OLD_FILE_1 BIT(1),
        READ_OLD_FILE_2 BIT(1);
DECLARE HOUR FIXED BINARY;
     READ FILE(OLDFIL1) INTO (OLD_FILE_1_HEADER);
     READ FILE(OLDFIL2) INTO (OLD_FILE_2_HEADER);
     ALLOCATE OLD_FILE_1_STATION,OLD_FILE_2_STATION;
     READ FILE(OLDFIL1) INTO (OLD_FILE_1_STATION);
     READ FILE(OLDFIL2) INTO (OLD_FILE_2_STATION);
     ALLOCATE OLD_FILE_1_PARAM,OLD_FILE_2_PARAM;
     READ FILE(OLDFIL1) INTO (OLD_FILE_1_PARAM);
     READ FILE(OLDFIL2) INTO (OLD_FILE_2_PARAM);
        PUT DATA(OLD_FILE_1_HEADER ) SKIP;
        PUT DATA(OLD_FILE_2_HEADER ) SKIP;
        PUT DATA(OLD_FILE_1_STATION) SKIP;
        PUT DATA(OLD_FILE_2_STATION) SKIP;
        PUT DATA(OLD_FILE_1_PARAM  ) SKIP;
        PUT DATA(OLD_FILE_2_PARAM  ) SKIP;
     ALLOCATE DUPLICATE_STATION;
     NUMBER_DUPLICATE_STATIONS=0;
     DO I=1 TO OLD_FILE_1_HEADER.STATIONS;
        DUPLICATE_STATION(I)=0;
        DO J=1 TO OLD_FILE_2_HEADER.STATIONS;
           IF OLD_FILE_1_STATION.NAME(I)=OLD_FILE_2_STATION.NAME(J)
           THEN
```

```
            DO;
               DUPLICATE_STATION(I)=J;
               NUMBER_DUPLICATE_STATIONS=NUMBER_DUPLICATE_STATIONS+1;
               GO TO NEXT_STATION;
            END;
         END;
NEXT_STATION:
      END;
      NUMBER_NEW_STATIONS=OLD_FILE_2_HEADER.STATIONS
                          -NUMBER_DUPLICATE_STATIONS;
      IF NUMBER_NEW_STATIONS¬=0 THEN ALLOCATE NEW_STATION;
      NEXT_NEW_STATION=1;
      DO I=1 TO OLD_FILE_2_HEADER.STATIONS;
         DO J=1 TO OLD_FILE_1_HEADER.STATIONS;
            IF I=DUPLICATE_STATION(J) THEN GO TO NOT_NEW_STATION;
         END;
         NEW_STATION(NEXT_NEW_STATION)=I;
         NEXT_NEW_STATION=NEXT_NEW_STATION+1;
NOT_NEW_STATION:
      END;
      NEW_FILE_HEADER.STATIONS=OLD_FILE_1_HEADER.STATIONS
                               +NUMBER_NEW_STATIONS;
      IF OLD_FILE_1_HEADER.STARTING_DATE<OLD_FILE_2_HEADER.STARTING_DATE
        THEN
        NEW_FILE_HEADER.STARTING_DATE=OLD_FILE_1_HEADER.STARTING_DATE;
        ELSE
        NEW_FILE_HEADER.STARTING_DATE=OLD_FILE_2_HEADER.STARTING_DATE;
      IF OLD_FILE_1_HEADER.ENDING_DATE>OLD_FILE_2_HEADER.ENDING_DATE
        THEN
        NEW_FILE_HEADER.ENDING_DATE=OLD_FILE_1_HEADER.ENDING_DATE;
        ELSE
        NEW_FILE_HEADER.ENDING_DATE=OLD_FILE_2_HEADER.ENDING_DATE;
      NUMBER_NEW_PARAMS=0;
      ALLOCATE POINTER_FROM_OLD_FILE_1,POINTER_FROM_OLD_FILE_2,
               NEW_FILE_STATION,DUPLICATE_POINTER;
      DUPLICATE_POINTER=0;
      DO I=1 TO OLD_FILE_1_HEADER.STATIONS;
         NEW_FILE_STATION(I).NAME=OLD_FILE_1_STATION(I).NAME;
         NEW_FILE_STATION(I).POINTER=NUMBER_NEW_PARAMS;
         ISUM=0;
         IF DUPLICATE_STATION(I)¬=0 THEN
         DO;
            L1=OLD_FILE_2_STATION(DUPLICATE_STATION(I)).POINTER;
            L2=OLD_FILE_2_STATION(DUPLICATE_STATION(I)).PARAMS+L1;
            DO L=L1+1 TO L2;
               POINTER_FROM_OLD_FILE_2(L)=0;
            END;
         END;
         DO J=1 TO OLD_FILE_1_STATION(I).PARAMS;
            NUMBER_NEW_PARAMS=NUMBER_NEW_PARAMS+1;
            ISUM=ISUM+1;
            M=OLD_FILE_1_STATION(I).POINTER+J;
            POINTER_FROM_OLD_FILE_1(M)=NUMBER_NEW_PARAMS;
            IF DUPLICATE_STATION(I)¬=0 THEN
            DO L=L1+1 TO L2;
               IF OLD_FILE_1_PARAM(M).NAME=OLD_FILE_2_PARAM(L).NAME THEN
               DO;
                  POINTER_FROM_OLD_FILE_2(L)=NUMBER_NEW_PARAMS;
                  DUPLICATE_POINTER(L)=I;
                  GO TO NEXT_PARAMETER;
               END;
            END;
NEXT_PARAMETER:
         END;
         IF DUPLICATE_STATION(I)¬=0 THEN
         DO L=L1+1 TO L2;
            IF POINTER_FROM_OLD_FILE_2(L)=0 THEN
            DO;
               ISUM=ISUM+1;
               NUMBER_NEW_PARAMS=NUMBER_NEW_PARAMS+1;
               POINTER_FROM_OLD_FILE_2(L)=NUMBER_NEW_PARAMS;
            END;
         END;
         NEW_FILE_STATION(I).PARAMS=ISUM;
      END;
      DO I=1 TO NUMBER_NEW_STATIONS;
         N=NEW_STATION(I);
         M=I+OLD_FILE_1_HEADER.STATIONS;
         ISUM=0 ;
         NEW_FILE_STATION(M).POINTER=NUMBER_NEW_PARAMS;
         NEW_FILE_STATION(M).NAME=OLD_FILE_2_STATION(N).NAME;
```

```
        DO J=1 TO OLD_FILE_2_STATION(N).PARAMS;
           NUMBER_NEW_PARAMS=NUMBER_NEW_PARAMS+1;
           POINTER_FROM_OLD_FILE_2(OLD_FILE_2_STATION(N).POINTER+J)
                 =NUMBER_NEW_PARAMS;
           ISUM=ISUM+1;
        END;
        NEW_FILE_STATION(M).PARAMS=ISUM;
     END;
     NEW_FILE_HEADER.PARAMS=NUMBER_NEW_PARAMS;
     ALLOCATE NEW_FILE_PARAM;
     DO I=1 TO OLD_FILE_1_HEADER.PARAMS;
        NEW_FILE_PARAM(POINTER_FROM_OLD_FILE_1(I)).NAME
                 =OLD_FILE_1_PARAM(I).NAME;
     END;
     DO I=1 TO OLD_FILE_2_HEADER.PARAMS;
        NEW_FILE_PARAM(POINTER_FROM_OLD_FILE_2(I)).NAME
                 =OLD_FILE_2_PARAM(I).NAME;
     END;
     PUT DATA(NEW_FILE_HEADER) SKIP;
     PUT DATA(NEW_FILE_STATION) SKIP;
     PUT DATA(NEW_FILE_PARAM) SKIP;
     WRITE FILE(NEWFILE) FROM (NEW_FILE_HEADER);
     WRITE FILE(NEWFILE) FROM (NEW_FILE_STATION);
     WRITE FILE(NEWFILE) FROM (NEW_FILE_PARAM);
        PUT DATA(POINTER_FROM_OLD_FILE_1) SKIP;
        PUT DATA(POINTER_FROM_OLD_FILE_2) SKIP;
        PUT DATA(DUPLICATE_POINTER) SKIP;
     ALLOCATE OLD_FILE_1_DATA,OLD_FILE_2_DATA,NEW_FILE_DATA;
     DATE_NOW=NEW_FILE_HEADER.STARTING_DATE;
     IF DATE_NOW=OLD_FILE_1_HEADER.STARTING_DATE THEN
        READ_OLD_FILE_1='1'B;ELSE READ_OLD_FILE_1='0'B;
     IF DATE_NOW=OLD_FILE_2_HEADER.STARTING_DATE THEN
        READ_OLD_FILE_2='1'B;ELSE READ_OLD_FILE_2='0'B;
     ON ENDFILE(OLDFIL1)
     BEGIN;
        IF OLD_FILE_1_HEADER.ENDING_DATE¬=NEW_FILE_DATA.DATE THEN
        DO;
          PUT LIST('UNEXPECTED END OF FILE ON OLDFIL1') SKIP;
          GO TO STOP;
        END;
        READ_OLD_FILE_1='0'B;
     END;
     ON ENDFILE(OLDFIL2)
     BEGIN;
        IF OLD_FILE_2_HEADER.ENDING_DATE¬=NEW_FILE_DATA.DATE THEN
        DO;
           PUT LIST('UNEXPECTED END OF FILE ON OLDFIL2') SKIP;
           GO TO STOP;
        END;
        READ_OLD_FILE_2='0'B;
     END;
READ:
     IF READ_OLD_FILE_1 THEN
        READ FILE(OLDFIL1) INTO (OLD_FILE_1_DATA);
     IF READ_OLD_FILE_2 THEN
        READ FILE(OLDFIL2) INTO (OLD_FILE_2_DATA);
     IF ¬READ_OLD_FILE_1 THEN
        DO;
           OLD_FILE_1_DATA.DATE=OLD_FILE_2_DATA.DATE;
           OLD_FILE_1_DATA.HOUR=1.E20;
           IF OLD_FILE_1_DATA.DATE=OLD_FILE_1_HEADER.STARTING_DATE
              | ¬READ_OLD_FILE_2 THEN
           DO;
              READ_OLD_FILE_1='1'B;
              READ FILE(OLDFIL1) INTO (OLD_FILE_1_DATA);
           END;
        END;
     IF ¬READ_OLD_FILE_2 THEN
        DO;
           OLD_FILE_2_DATA.DATE=OLD_FILE_1_DATA.DATE;
           OLD_FILE_2_DATA.HOUR=1.E20;
           IF OLD_FILE_2_DATA.DATE=OLD_FILE_2_HEADER.STARTING_DATE
              | ¬ READ_OLD_FILE_1 THEN
           DO;
PUT DATA (OLD_FILE_1_DATA) SKIP;
PUT DATA (OLD_FILE_2_DATA) SKIP;
PUT DATA (NEW_FILE_DATA) SKIP;
              READ_OLD_FILE_2='1'B;
              READ FILE(OLDFIL2) INTO (OLD_FILE_2_DATA);
PUT DATA (OLD_FILE_2_DATA) SKIP;
           END;
```

```
        END;
     IF OLD_FILE_1_DATA.DATE¬=OLD_FILE_2_DATA.DATE THEN
        DO;
           PUT LIST('DATES ON OLD FILES DO NOT AGREE') SKIP;
           GO TO STOP;
        END;
     NEW_FILE_DATA.DATE=OLD_FILE_1_DATA.DATE;
PUT DATA(NEW_FILE_DATA.DATE) SKIP;
     DO HOUR=1 TO 24;
        DO J=1 TO OLD_FILE_1_HEADER.PARAMS;
           NEW_FILE_DATA.PARAM(HOUR,POINTER_FROM_OLD_FILE_1(J))
                 =OLD_FILE_1_DATA.PARAM(HOUR,J);
        END;
        DO J=1 TO OLD_FILE_2_HEADER.PARAMS;
           IF DUPLICATE_POINTER(J)<1 THEN
             NEW_FILE_DATA.PARAM(HOUR,POINTER_FROM_OLD_FILE_2(J))
                 =OLD_FILE_2_DATA.PARAM(HOUR,J);
           ELSE IF OLD_FILE_2_DATA.PARAM(HOUR,J)¬=1.E20 THEN
             NEW_FILE_DATA.PARAM(HOUR,POINTER_FROM_OLD_FILE_2(J))
                 =OLD_FILE_2_DATA.PARAM(HOUR,J);
        END;
     END;
     WRITE FILE(NEWFILE) FROM (NEW_FILE_DATA);
     IF NEW_FILE_DATA.DATE¬=NEW_FILE_HEADER.ENDING_DATE THEN
        GO TO READ;
STOP:
PUT DATA (OLD_FILE_1_DATA) SKIP;
PUT DATA (OLD_FILE_2_DATA) SKIP;
PUT DATA (NEW_FILE_DATA) SKIP;
END MERGE;
```

4. SUBFIL Program to Extract a Subset of a Master Tape

```
SUBFIL : PROC OPTIONS (MAIN);
    DCL DATE_LIMITS FIXED BINARY INITIAL(0),
        LOWER_DATE(DATE_LIMITS) CHAR(8) CONTROLLED,
        UPPER_DATE(DATE_LIMITS) CHAR(8) CONTROLLED;
DCL     1 FILE_HEADER,
            2 STATIONS FIXED BINARY,
            2 PARAMS FIXED BINARY,
            2 STARTING_DATE CHAR(6),
            2 ENDING_DATE CHAR(6);
DCL     1 FILE_STATION (FILE_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 FILE_PARAM (FILE_HEADER.PARAMS) CONTROLLED,
            2 NAME CHAR(8);
DCL     1 TODAY     CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (FILE_HEADER.PARAMS);
DCL     1 YESTERDAY CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (FILE_HEADER.PARAMS);
DCL     1 SUBFILE_HEADER,
            2 STATIONS FIXED BINARY,
            2 PARAMS FIXED BINARY,
            2 STARTING_DATE CHAR(6),
            2 ENDING_DATE CHAR(6);
DCL     1 SUBFILE_STATION (SUBFILE_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 SUBFILE_PARAM (SUBFILE_HEADER.PARAMS) CONTROLLED,
            2 NAME CHAR(8);
DCL     1 SUBFILE_DATA CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (SUBFILE_HEADER.PARAMS);
    DCL OUT_STATIONS FIXED BINARY (31,0),
        OUT_PARAMS   FIXED BINARY (31,0);
DCL     1 OUT_PARAM (OUT_PARAMS) CONTROLLED,
            2 POINTER FIXED BINARY,
            2 DELAY FIXED DECIMAL (2);
DCL     1 DUM_OUT_PARAM (100)  CONTROLLED,
            2 POINTER FIXED BINARY,
            2 DELAY FIXED DECIMAL (2);
DCL     1 OUTPUT_HEADER,
            2 LABEL (10) CHAR(8),
            2 STATIONS FIXED BINARY (31,0),
            2 PARAMS FIXED BINARY (31,0),
            2 NUMBER_OBS FIXED BINARY (31,0);
DCL     1 OUTPUT_STATION (OUTPUT_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 OUTPUT_PARAM (OUTPUT_HEADER.PARAMS) CONTROLLED,
            2 STATION CHAR(8),
            2 NAME CHAR(8);
DCL     1 DUM_OUTPUT_PARAM(100)  CONTROLLED,
            2 STATION CHAR(8),
            2 NAME CHAR(8);
DCL      PRINT_TITLE (10) FLOAT DECIMAL (16),
         PRINT_DATE(1) FLOAT DECIMAL (16),
         PRINT_STATION (OUT_PARAMS) CONTROLLED FLOAT DECIMAL (16),
         PRINT_NAME    (OUT_PARAMS) CONTROLLED FLOAT DECIMAL (16),
         PRINT_PARAM (JDUM,IDUM                ) CONTROLLED;
DCL      IDUM FIXED BINARY (31,0), JDUM FIXED BINARY (31,0);
DCL      CARD CHAR(80);
         PUT LIST('MASTER FILE HEADER INFORMATION')PAGE;
         READ FILE(MASTER) INTO (FILE_HEADER);
         PUT DATA(FILE_HEADER)SKIP;
         ALLOCATE FILE_STATION;
         READ FILE(MASTER) INTO (FILE_STATION);
         PUT DATA(FILE_STATION)SKIP;
         ALLOCATE FILE_PARAM;
         READ FILE(MASTER) INTO (FILE_PARAM);
         PUT DATA (FILE_PARAM)SKIP;
         ALLOCATE TODAY; ALLOCATE YESTERDAY;
NEXT:    /* BEGIN NEW CASE */
         CLOSE FILE(MASTER);
         ON ENDFILE(MASTER)  BEGIN;
```

```
        PUT LIST('END OF FILE SENSED ON MASTER')SKIP;
        GO TO NEXT ;END;
        READ FILE (MASTER) IGNORE(3);
        CLOSE FILE(CARDS);
GETCARD:GET EDIT(CARD)(A(80));
        IF INDEX(CARD,'END ')¬=0 THEN GO TO END;
        IF INDEX(CARD,'STOP')¬=0 THEN GO TO STOP;
        WRITE FILE(CARDS) FROM (CARD);
        GO TO GETCARD;
END:    CLOSE FILE(CARDS);
        CALL INPUT;
        CALL SEARCH;
        GO TO NEXT;
INPUT:  PROC;
DCL     PARAMS FIXED BINARY,BANDS FIXED BINARY,
        DELAY FIXED DECIMAL (2);
    DCL STATION CHAR(8),PARAM CHAR(8);
        ON NAME(CARDS) BEGIN;
        PUT LIST('THE FOLLOWING DATA IS OUT OF ORDER: ',DATAFIELD)SKIP;
        GO TO NEXT;END;
        ON ENDFILE(CARDS)  BEGIN;
        PUT LIST('UNEXPECTED EOF ON CARD INPUT')SKIP;
        GO TO NEXT;END;
        PUT LIST('INPUT DATA')PAGE;
        GET FILE(CARDS) EDIT(LABEL)(10A(8));
        DO I=1 TO 10;
        UNSPEC(PRINT_TITLE(I))=UNSPEC(LABEL(I));
        END;
        PUT EDIT(LABEL)(10A(8))SKIP;
        FREE     LOWER_DATE; FREE     UPPER_DATE;
        DATE_LIMITS=1;
        ALLOCATE LOWER_DATE; ALLOCATE UPPER_DATE;
        LOWER_DATE(1)=FILE_HEADER.STARTING_DATE;
        UPPER_DATE(1)=FILE_HEADER.ENDING_DATE;
        PUT DATA(LOWER_DATE(1),UPPER_DATE(1))SKIP;
        PUT LIST('OUT STATION REQUESTS')SKIP;
        GET FILE(CARDS) DATA (OUT_STATIONS);
        IF OUT_STATIONS=0 THEN BEGIN;
        PUT LIST('NO OUT STATIONS REQUESTED')SKIP;
        GO TO NEXT;END;
        OUTPUT_HEADER.STATIONS=OUT_STATIONS;
        FREE     OUTPUT_STATION;
        ALLOCATE OUTPUT_STATION;
        OUT_PARAMS=0;
        ALLOCATE DUM_OUT_PARAM;
        ALLOCATE DUM_OUTPUT_PARAM;
        DO I=1 TO OUT_STATIONS;
        PARAMS=0;
        GET FILE(CARDS) DATA(STATION,PARAMS);
        PUT DATA(STATION)SKIP;
        DO N=1 TO FILE_HEADER.STATIONS;
        IF STATION=FILE_STATION(N).NAME THEN GO TO FIND_N2;
        END;/* N LOOP*/
        PUT LIST('STATION NAME ',STATION,' NOT RECOGNIZED');GO TO NEXT;
FIND_N2:OUTPUT_STATION(I).NAME=STATION;
        OUTPUT_STATION(I).POINTER=OUT_PARAMS;
        IF PARAMS=0 THEN BEGIN;
        PARAMS=FILE_STATION(N).PARAMS;
        IF OUT_PARAMS+PARAMS>100 THEN BEGIN;
        PUT LIST('NUMBER OF PARAMETERS GREATER THAN 100')SKIP;
        GO TO NEXT;END;
        ELSE BEGIN  ;
        OUTPUT_STATION(I).PARAMS=PARAMS;
        DELAY=0;
        DO J=1 TO PARAMS;
        DUM_OUTPUT_PARAM(OUT_PARAMS+J).STATION=STATION;
        PARAM=
        FILE_PARAM(FILE_STATION(N).POINTER+J).NAME;
        PUT DATA(PARAM,DELAY)SKIP;
        DUM_OUTPUT_PARAM(OUT_PARAMS+J).NAME=  PARAM;
        DUM_OUT_PARAM(OUT_PARAMS+J).POINTER=FILE_STATION(N).POINTER+J;
        DUM_OUT_PARAM(OUT_PARAMS+J).DELAY=DELAY;
        END; /* PARAM LOOP */
        GO TO NEWSTA; END;          END;
        IF OUT_PARAMS+PARAMS>100 THEN BEGIN;
        PUT LIST('NUMBER OF PARAMETERS GREATER THAN 100')SKIP;
        GO TO NEXT;END;
        OUTPUT_STATION(I).PARAMS=PARAMS;
        DO J=1 TO PARAMS;
        DELAY=0;
        GET FILE(CARDS) DATA(PARAM,DELAY);
```

```
          PUT DATA(PARAM,DELAY)SKIP;
          DUM_OUTPUT_PARAM(OUT_PARAMS+J).NAME=PARAM;
          DUM_OUTPUT_PARAM(OUT_PARAMS+J).STATION=STATION;
          DO M=1 TO FILE_STATION (N).PARAMS;
          IF PARAM=FILE_PARAM(FILE_STATION(N).POINTER+M).NAME
          THEN GO TO FIND_M2;
          END;/* M LOOP*/
          PUT LIST('PARAMETER NAME ',PARAM,' NOT RECOGNIZED');GO TO NEXT;
FIND_M2:  DUM_OUT_PARAM(OUT_PARAMS+J).POINTER=FILE_STATION(N).POINTER+M;
          DUM_OUT_PARAM(OUT_PARAMS+J).DELAY=DELAY;
          END; /* PARAM LOOP */
NEWSTA:   OUT_PARAMS=OUT_PARAMS+PARAMS;
          END; /*STATION LOOP */
          OUTPUT_HEADER.PARAMS=OUT_PARAMS;
          FREE     OUTPUT_PARAM;
          ALLOCATE OUTPUT_PARAM;
          FREE     OUT_PARAM;
          ALLOCATE OUT_PARAM;
          FREE     PRINT_STATION;
          ALLOCATE PRINT_STATION;
          FREE     PRINT_NAME;
          ALLOCATE PRINT_NAME;
          DO J=1 TO OUT_PARAMS;
          OUTPUT_PARAM(J)=DUM_OUTPUT_PARAM(J);
          OUT_PARAM(J)=DUM_OUT_PARAM(J);
          STATION=OUTPUT_PARAM(J).STATION;
          PARAM  =OUTPUT_PARAM(J).NAME    ;
          UNSPEC(PRINT_STATION(J))=UNSPEC(STATION);
          UNSPEC(PRINT_NAME   (J))=UNSPEC(PARAM);
          END;
          FREE DUM_OUTPUT_PARAM;
          FREE DUM_OUT_PARAM;
          RETURN;
          END INPUT;
SEARCH:   PROC;
DCL       HOUR FIXED BINARY;
DCL       DATE_KEY BIT(1) INITIAL('0'B);
DCL       DATE_LIMIT FIXED BINARY;
DCL       DELAY_HOUR FIXED BINARY;
DCL       YEAR CHAR(2), MONTH CHAR(2), DAY CHAR(2);
          IDUM=24; JDUM=OUT_PARAMS;
          DATE_LIMIT=1;
          PUT LIST('SUB FILE HEADER INFORMATION')PAGE;
          /* CREATE SUB FILE HEADER */
          SUBFILE_HEADER.STATIONS=OUTPUT_HEADER.STATIONS;
          SUBFILE_HEADER.PARAMS=OUTPUT_HEADER.PARAMS;
          SUBFILE_HEADER.STARTING_DATE=FILE_HEADER.STARTING_DATE;
          SUBFILE_HEADER.  ENDING_DATE=FILE_HEADER.  ENDING_DATE;
          PUT DATA(SUBFILE_HEADER)SKIP;
          WRITE FILE(SUBFILE) FROM (SUBFILE_HEADER);
          /* CREATE SUBFILE STATION DEFINITIONS */
          FREE     SUBFILE_STATION;
          ALLOCATE SUBFILE_STATION;
          SUBFILE_STATION=OUTPUT_STATION;
          PUT DATA(SUBFILE_STATION)SKIP;
          WRITE FILE(SUBFILE) FROM (SUBFILE_STATION);
          /* CREATE SUB FILE PARAMETER DEFINITIONS  */
          FREE     SUBFILE_PARAM;
          ALLOCATE SUBFILE_PARAM;
          DO N=1 TO OUT_PARAMS;
          SUBFILE_PARAM.NAME(N)=OUTPUT_PARAM.NAME(N);
          END;
          PUT DATA(SUBFILE_PARAM)SKIP;
          WRITE FILE(SUBFILE) FROM (SUBFILE_PARAM);
          FREE     SUBFILE_DATA;
          ALLOCATE SUBFILE_DATA;
          FREE     PRINT_PARAM;
          ALLOCATE PRINT_PARAM;
          ON ENDFILE(MASTER)  BEGIN;
          PUT LIST('END OF FILE SENSED ON MASTER')SKIP;
          GO TO RETURN; END;
          READ FILE(MASTER) INTO (TODAY);
          YESTERDAY=TODAY;
DATES:    /* LOOP ON DATES - TEST BETWEEN DATE LIMITS */
          IF TODAY.DATE=LOWER_DATE(DATE_LIMIT) THEN BEGIN;
          DATE_KEY='1'B; GO TO HOURS;
          END;
          IF TODAY.DATE=UPPER_DATE(DATE_LIMIT) THEN BEGIN;
          DATE_KEY='0'B;DATE_LIMIT=DATE_LIMIT+1;
          GO TO HOURS;
          END;
```

```
          IF DATE_KEY='0'B THEN GO TO NEWDATE;
HOURS:    /* LOOP ON HOURS */
          SUBFILE_DATA.DATE=TODAY.DATE;
      PUT DATA(SUBFILE_DATA.DATE)SKIP;
          YEAR =SUBSTR(TODAY.DATE,1,2);
          MONTH=SUBSTR(TODAY.DATE,3,2);
          DAY  =SUBSTR(TODAY.DATE,5,2);
          UNSPEC(PRINT_DATE(1))=UNSPEC(YEAR||'/'||MONTH||'/'||DAY);
          DO HOUR=1 TO 24;
          DO J=1 TO OUT_PARAMS;
          DELAY_HOUR=HOUR-OUT_PARAM(J).DELAY;
          IF DELAY_HOUR<1 THEN BEGIN;
          SUBFILE_DATA.PARAM (HOUR,J)=
          YESTERDAY.PARAM(24+DELAY_HOUR,OUT_PARAM.POINTER(J));
          PRINT_PARAM(J,HOUR)=
          YESTERDAY.PARAM(24+DELAY_HOUR,OUT_PARAM.POINTER(J));
          GO TO SKIP;END;
          ELSE   BEGIN;
          SUBFILE_DATA.PARAM (HOUR,J)=
          TODAY.PARAM(DELAY_HOUR,OUT_PARAM.POINTER(J));
          PRINT_PARAM(J,HOUR)=
          TODAY.PARAM(DELAY_HOUR,OUT_PARAM.POINTER(J));   END;
SKIP:     END; /* PARAM LOOP */
          END; /* HOUR LOOP */
          WRITE FILE(SUBFILE) FROM (SUBFILE_DATA);
NEWDATE:IF DATE_LIMIT>DATE_LIMITS'THEN RETURN;
          YESTERDAY=TODAY;
          READ FILE(MASTER) INTO (TODAY);
          GO TO DATES;
RETURN:   RETURN;
          END SEARCH;
STOP:     END SUBFIL;
```

5. FUNCTION Functions of Parameters on a Master Tape

```
FUNCTN : PROC OPTIONS (MAIN);
    DCL DATE_LIMITS FIXED BINARY INITIAL(0),
        LOWER_DATE(DATE_LIMITS) CHAR(8) CONTROLLED,
        UPPER_DATE(DATE_LIMITS) CHAR(8) CONTROLLED;
DCL     1 FILE_HEADER,
            2 STATIONS FIXED BINARY,
            2 PARAMS FIXED BINARY,
            2 STARTING_DATE CHAR(6),
            2 ENDING_DATE CHAR(6);
DCL     1 FILE_STATION (FILE_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 FILE_PARAM (FILE_HEADER.PARAMS) CONTROLLED,
            2 NAME CHAR(8);
DCL     1 TODAY     CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (FILE_HEADER.PARAMS);
DCL     1 YESTERDAY CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (FILE_HEADER.PARAMS);
DCL     1 SUBFILE_HEADER,
            2 STATIONS FIXED BINARY,
            2 PARAMS FIXED BINARY,
            2 STARTING_DATE CHAR(6),
            2 ENDING_DATE CHAR(6);
DCL     1 SUBFILE_STATION (SUBFILE_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 SUBFILE_PARAM (SUBFILE_HEADER.PARAMS) CONTROLLED,
            2 NAME CHAR(8);
DCL     1 SUBFILE_DATA CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (SUBFILE_HEADER.PARAMS);
    DCL OUT_STATIONS FIXED BINARY (31,0),
         IN_STATIONS FIXED BINARY (31,0),
        OUT_PARAMS   FIXED BINARY (31,0),
         IN_PARAMS   FIXED BINARY (31,0);
DCL     1 OUT_PARAM (OUT_PARAMS) CONTROLLED,
            2 POINTER FIXED BINARY,
            2 DELAY FIXED DECIMAL (2);
DCL     1 DUM_OUT_PARAM (100)  CONTROLLED,
            2 POINTER FIXED BINARY,
            2 DELAY FIXED DECIMAL (2);
DCL     1 OUTPUT_HEADER,
            2 LABEL (10) CHAR(8),
            2 STATIONS FIXED BINARY (31,0),
            2 PARAMS FIXED BINARY (31,0),
            2 NUMBER_OBS FIXED BINARY (31,0);
DCL     1 OUTPUT_STATION (OUTPUT_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 OUTPUT_PARAM (OUTPUT_HEADER.PARAMS) CONTROLLED,
            2 STATION CHAR(8),
            2 NAME CHAR(8);
DCL     1 DUM_OUTPUT_PARAM(100)  CONTROLLED,
            2 STATION CHAR(8),
            2 NAME CHAR(8);
DCL      PRINT_TITLE (10) FLOAT DECIMAL (16),
         PRINT_DATE(1) FLOAT DECIMAL (16),
         PRINT_STATION (OUT_PARAMS) CONTROLLED FLOAT DECIMAL (16),
         PRINT_NAME    (OUT_PARAMS) CONTROLLED FLOAT DECIMAL (16),
         PRINT_PARAM (JDUM,IDUM                ) CONTROLLED,
           DUM_PARAM (KDUM,IDUM                ) CONTROLLED;
DCL      PARAMS FIXED BINARY,BANDS FIXED BINARY,
         DELAY FIXED DECIMAL (2);
    DCL STATION CHAR(8),PARAM CHAR(8);
DCL      IDUM FIXED BINARY (31,0), JDUM FIXED BINARY (31,0);
DCL      KDUM FIXED BINARY (31,0);
DCL      CARD CHAR(80);
         PUT LIST('MASTER FILE HEADER INFORMATION')PAGE;
         READ FILE(MASTER) INTO (FILE_HEADER);
         PUT DATA(FILE_HEADER)SKIP;
         ALLOCATE FILE_STATION;
         READ FILE(MASTER) INTO (FILE_STATION);
         PUT DATA(FILE_STATION)SKIP;
```

```
        ALLOCATE FILE_PARAM;
        READ FILE(MASTER) INTO (FILE_PARAM);
        PUT DATA (FILE_PARAM)SKIP;
        ALLOCATE TODAY; ALLOCATE YESTERDAY;
NEXT:   /* BEGIN NEW CASE */
        CLOSE FILE(MASTER);
        ON ENDFILE(MASTER)  BEGIN;
        PUT LIST('END OF FILE SENSED ON MASTER')SKIP;
        GO TO NEXT ;END;
        READ FILE (MASTER) IGNORE(3);
        CLOSE FILE(CARDS);
GETCARD:GET EDIT(CARD)(A(80));
        IF INDEX(CARD,'END ')¬=0 THEN GO TO END;
        IF INDEX(CARD,'STOP')¬=0 THEN GO TO STOP;
        WRITE FILE(CARDS) FROM (CARD);
        GO TO GETCARD;
END:    CLOSE FILE(CARDS);
        CALL INPUT;
        CALL SEARCH;
        GO TO NEXT;
INPUT:  PROC;
        ON NAME(CARDS) BEGIN;
        PUT LIST('THE FOLLOWING DATA IS OUT OF ORDER: ',DATAFIELD)SKIP;
        GO TO NEXT;END;
        ON ENDFILE(CARDS)  BEGIN;
        PUT LIST('UNEXPECTED EOF ON CARD INPUT')SKIP;
        GO TO NEXT;END;
        PUT LIST('INPUT DATA')PAGE;
        GET FILE(CARDS) EDIT(LABEL)(10A(8));
        DO I=1 TO 10;
        UNSPEC(PRINT_TITLE(I))=UNSPEC(LABEL(I));
        END;
        PUT EDIT(LABEL)(10A(8))SKIP;
        FREE     LOWER_DATE; FREE     UPPER_DATE;
        DATE_LIMITS=1;
        ALLOCATE LOWER_DATE; ALLOCATE UPPER_DATE;
        LOWER_DATE(1)=FILE_HEADER.STARTING_DATE;
        UPPER_DATE(1)=FILE_HEADER.ENDING_DATE;
        PUT DATA(LOWER_DATE(1),UPPER_DATE(1))SKIP;
        OUT_PARAMS=0;
        PUT LIST(' IN STATION REQUESTS')SKIP;
        GET FILE(CARDS) DATA ( IN_STATIONS);
        OUT_STATIONS=IN_STATIONS;
        IF OUT_STATIONS=0 THEN BEGIN;
        PUT LIST('NO  IN STATIONS REQUESTED')SKIP;
        GO TO RETURN; END;
        OUTPUT_HEADER.STATIONS=OUT_STATIONS;
        FREE     OUTPUT_STATION;
        ALLOCATE OUTPUT_STATION;
        OUT_PARAMS=0;
        ALLOCATE DUM_OUT_PARAM;
        ALLOCATE DUM_OUTPUT_PARAM;
        DO I=1 TO OUT_STATIONS;
        PARAMS=0;
        GET FILE(CARDS) DATA(STATION,PARAMS);
        PUT DATA(STATION)SKIP;
        DO N=1 TO FILE_HEADER.STATIONS;
        IF STATION=FILE_STATION(N).NAME THEN GO TO FIND_N2;
        END;/* N LOOP*/
        PUT LIST('STATION NAME ',STATION,' NOT RECOGNIZED');GO TO NEXT;
FIND_N2:OUTPUT_STATION(I).NAME=STATION;
        OUTPUT_STATION(I).POINTER=OUT_PARAMS;
        IF PARAMS=0 THEN BEGIN;
        PARAMS=FILE_STATION(N).PARAMS;
        IF OUT_PARAMS+PARAMS>100 THEN BEGIN;
        PUT LIST('NUMBER OF PARAMETERS GREATER THAN 100')SKIP;
        GO TO NEXT;END;
        ELSE BEGIN   ;
        OUTPUT_STATION(I).PARAMS=PARAMS;
        DELAY=0;
        DO J=1 TO PARAMS;
        DUM_OUTPUT_PARAM(OUT_PARAMS+J).STATION=STATION;
        PARAM=
        FILE_PARAM(FILE_STATION(N).POINTER+J).NAME;
        PUT DATA(PARAM,DELAY)SKIP;
        DUM_OUTPUT_PARAM(OUT_PARAMS+J).NAME=  PARAM;
        DUM_OUT_PARAM(OUT_PARAMS+J).POINTER=FILE_STATION(N).POINTER+J;
        DUM_OUT_PARAM(OUT_PARAMS+J).DELAY=DELAY;
        END; /* PARAM LOOP */
        GO TO NEWSTA; END;        END;
        IF OUT_PARAMS+PARAMS>100 THEN BEGIN;
```

```
        PUT LIST('NUMBER OF PARAMETERS GREATER THAN 100')SKIP;
        GO TO NEXT;END;
        OUTPUT_STATION(I).PARAMS=PARAMS;
        DO J=1 TO PARAMS;
        DELAY=0;
        GET FILE(CARDS) DATA(PARAM,DELAY);
        PUT DATA(PARAM,DELAY)SKIP;
        DUM_OUTPUT_PARAM(OUT_PARAMS+J).NAME=PARAM;
        DUM_OUTPUT_PARAM(OUT_PARAMS+J).STATION=STATION;
        DO M=1 TO FILE_STATION (N).PARAMS;
        IF PARAM=FILE_PARAM(FILE_STATION(N).POINTER+M).NAME
        THEN GO TO FIND_M2;
        END;/* M LOOP*/
        PUT LIST('PARAMETER NAME ',PARAM,' NOT RECOGNIZED');GO TO NEXT;
FIND_M2:DUM_OUT_PARAM(OUT_PARAMS+J).POINTER=FILE_STATION(N).POINTER+M;
        DUM_OUT_PARAM(OUT_PARAMS+J).DELAY=DELAY;
        END; /* PARAM LOOP */
NEWSTA: OUT_PARAMS=OUT_PARAMS+PARAMS;
        END; /*STATION LOOP */
        OUTPUT_HEADER.PARAMS=OUT_PARAMS;
        FREE     OUTPUT_PARAM;
        ALLOCATE OUTPUT_PARAM;
        FREE     OUT_PARAM;
        ALLOCATE OUT_PARAM;
        FREE     PRINT_STATION;
        ALLOCATE PRINT_STATION;
        FREE     PRINT_NAME;
        ALLOCATE PRINT_NAME;
        DO J=1 TO OUT_PARAMS;
        OUTPUT_PARAM(J)=DUM_OUTPUT_PARAM(J);
        OUT_PARAM(J)=DUM_OUT_PARAM(J);
        STATION=OUTPUT_PARAM(J).STATION;
        PARAM  =OUTPUT_PARAM(J).NAME    ;
        UNSPEC(PRINT_STATION(J))=UNSPEC(STATION);
        UNSPEC(PRINT_NAME   (J))=UNSPEC(PARAM);
        END;
        FREE DUM_OUTPUT_PARAM;
        FREE DUM_OUT_PARAM;
RETURN: IN_PARAMS=OUT_PARAMS;
        RETURN;
        END INPUT;
SEARCH: PROC;
DCL     HOUR FIXED BINARY;
DCL     DATE_KEY BIT(1) INITIAL('0'B);
DCL     DATE_LIMIT FIXED BINARY;
DCL     DELAY_HOUR FIXED BINARY;
DCL     YEAR CHAR(2), MONTH CHAR(2), DAY CHAR(2);
DCL     FUNCTION CHAR(8);
        IDUM=24; JDUM= IN_PARAMS;
        DATE_LIMIT=1;
        GET FILE(CARDS)DATA(FUNCTION,PARAMS);
        PUT DATA(FUNCTION)SKIP;
        IF PARAMS=0 THEN BEGIN;
        PUT LIST('NO FUNCTION PARAMETERS REQUESTED')SKIP;
        GO TO NEXT; END;
        OUT_PARAMS=PARAMS;
        KDUM=OUT_PARAMS;
        PUT LIST('SUB FILE HEADER INFORMATION')PAGE;
        /* CREATE SUB FILE HEADER */
        SUBFILE_HEADER.STATIONS=1;
        SUBFILE_HEADER.PARAMS=PARAMS;
        SUBFILE_HEADER.STARTING_DATE=FILE_HEADER.STARTING_DATE;
        SUBFILE_HEADER.  ENDING_DATE=FILE_HEADER.  ENDING_DATE;
        PUT DATA(SUBFILE_HEADER)SKIP;
        WRITE FILE(SUBFILE) FROM (SUBFILE_HEADER);
        /* CREATE SUBFILE STATION DEFINITIONS */
        FREE     SUBFILE_STATION;
        ALLOCATE SUBFILE_STATION;
        SUBFILE_STATION.NAME(1)=FUNCTION;
        SUBFILE_STATION.POINTER(1)=0;
        SUBFILE_STATION.PARAMS(1)=PARAMS;
        PUT DATA(SUBFILE_STATION)SKIP;
        WRITE FILE(SUBFILE) FROM (SUBFILE_STATION);
        /* CREATE SUB FILE PARAMETER DEFINITIONS  */
        FREE     SUBFILE_PARAM;
        ALLOCATE SUBFILE_PARAM;
        DO N=1 TO OUT_PARAMS;
        GET FILE(CARDS) DATA(PARAM);
        PUT DATA(PARAM)SKIP;
        SUBFILE_PARAM.NAME(N)=PARAM;
        END;
```

```
        PUT DATA(SUBFILE_PARAM)SKIP;
        WRITE FILE(SUBFILE) FROM (SUBFILE_PARAM);
        FREE      SUBFILE_DATA;
        ALLOCATE SUBFILE_DATA;
        FREE      PRINT_PARAM;
        ALLOCATE PRINT_PARAM;
        FREE      DUM_PARAM;
        ALLOCATE DUM_PARAM;
        ON ENDFILE(MASTER)  BEGIN;
        PUT LIST('END OF FILE SENSED ON MASTER')SKIP;
        GO TO RETURN; END;
        READ FILE(MASTER) INTO (TODAY);
        YESTERDAY=TODAY;
DATES:  /* LOOP ON DATES - TEST BETWEEN DATE LIMITS */
        IF TODAY.DATE=LOWER_DATE(DATE_LIMIT) THEN BEGIN;
        DATE_KEY='1'B; GO TO HOURS;
        END;
        IF TODAY.DATE=UPPER_DATE(DATE_LIMIT) THEN BEGIN;
        DATE_KEY='0'B;DATE_LIMIT=DATE_LIMIT+1;
        GO TO HOURS;
        END;
        IF DATE_KEY='0'B THEN GO TO NEWDATE;
HOURS:  /* LOOP ON HOURS */
        SUBFILE_DATA.DATE=TODAY.DATE;
     PUT DATA(SUBFILE_DATA.DATE)SKIP;
        YEAR =SUBSTR(TODAY.DATE,1,2);
        MONTH=SUBSTR(TODAY.DATE,3,2);
        DAY  =SUBSTR(TODAY.DATE,5,2);
        UNSPEC(PRINT_DATE(1))=UNSPEC(YEAR||'/'||MONTH||'/'||DAY);
        IF IN_PARAMS=0 THEN GO TO SKIPP;
        DO HOUR=1 TO 24;
        DO J=1 TO  IN_PARAMS;
        DELAY_HOUR=HOUR-OUT_PARAM(J).DELAY;
        IF DELAY_HOUR<1 THEN BEGIN;
        PRINT_PARAM(J,HOUR)=
        YESTERDAY.PARAM(24+DELAY_HOUR,OUT_PARAM.POINTER(J));
        GO TO SKIP;END;
        ELSE   BEGIN;
        PRINT_PARAM(J,HOUR)=
        TODAY.PARAM(DELAY_HOUR,OUT_PARAM.POINTER(J));   END;
SKIP:   END; /* PARAM LOOP */
        END; /* HOUR LOOP */
SKIPP:  NYEAR=YEAR; NMONTH=MONTH; NDAY=DAY;
        CALL FORTRN(NYEAR,NMONTH,NDAY,IN_PARAMS,PRINT_PARAM(1,1),
        OUT_PARAMS,DUM_PARAM(1,1));
        DO HOUR=1 TO 24;
        DO K=1 TO OUT_PARAMS;
        SUBFILE_DATA.PARAM(HOUR,K)=DUM_PARAM(K,HOUR);
        END;END;
        WRITE FILE(SUBFILE) FROM (SUBFILE_DATA);
NEWDATE:IF DATE_LIMIT>DATE_LIMITS THEN RETURN;
        YESTERDAY=TODAY;
        READ FILE(MASTER) INTO (TODAY);
        GO TO DATES;
RETURN: RETURN;
        END SEARCH;
STOP:   END FUNCTN;
```

APICS USER MANUAL

APPENDIX E

SEARCH PROGRAMS
SOURCE LISTINGS

1. SEARCH1 Search of Master Tape for Specified Information

```
      SUBROUTINE TWODPT(ARAY,ICT,JCT,TITLE,ROWT1,ROWT2,COLT1,COLT2)
C
C                                TWODPT PRINTS A TWO-DIMENSIONAL ARRAY.  IT
C                               PRINTS A TITLE LINE AND TWO LINES OF COLUMN
C                               TITLES, WHICH ARE FOLLOWED BY 50 LINES UP
C                               TO TEN COLUMNS WIDE PLUS THE INDEX COLUMN,
C                               WHICH CONTAINS DATE AND HOUR.
C                               ARGUMENTS ARE -
C
C                                     ARAY - THE TWO-DIMENSIONAL ARRAY
C                                            TO BE PRINTED.
C                                      ICT - NUMBER OF ROWS IN ARRAY.
C                                      JCT - NUMBER OF COLUMNS IN ARRAY.
C                                    TITLE - PAGE TITLE (10A8).
C                                    ROWT1 - ARRAY DIMENSIONED BY ICT, CON-
C                                            TAINING DATE FOR EACH ROW
C                                            (A8).
C                                    ROWT2 - ARRAY DIMENSIONED BY ICT, CON-
C                                            TAINING HOUR FOR EACH ROW
C                                            (A4, PRINTED AS A2).
C                                    COLT1 - ARRAY DIMENSIONED BY JCT, CON-
C                                            TAINING FIRST LINE OF TITLE
C                                            FOR EACH COLUMN (A8).
C                                    COLT2 - ARRAY DIMENSIONED BY JCT, CON-
C                                            TAINING SECOND LINE OF TITLE
C                                            FOR EACH COLUMN (A8).
C
      REAL*8 TITLE(10),ROWT1(1),COLT1(1),COLT2(1)
      REAL*4 ARAY(ICT,1),ROWT2(1)
      II=1
      IJ=1
      IMAX=0
      JMAX=0
      MAX=0
      IFLAG=0
   20 IEND=JCT
   40 MIN=MAX+1
      MAX=MIN0(MAX+10,IEND)
      IF(IFLAG.NE.0) GO TO 47
   45 NROW=0
   46 CONTINUE
      PRINT 210,(TITLE(I),I=1,10)
   47 PRINT 220
      PRINT 230,(COLT1(JC),JC=MIN,MAX)
      PRINT 230,(COLT2(JC),JC=MIN,MAX)
      PRINT 230
      IMAX=MIN0(II+52,ICT)
      IDIF=IMAX-II+5
      NROW=NROW+IDIF
      JMAX=MIN0(IJ+9,JCT)
      DO 65 I=II,IMAX
      PRINT 240,ROWT1(I),ROWT2(I),(ARAY(I,J),J=IJ,JMAX)
   65 CONTINUE
      IF(IMAX.GE.ICT) GO TO 70
      II=IMAX+1
      GO TO 46
   70 IF(JMAX.GE.JCT) GO TO 130
   80 II=1
      IJ=JMAX+1
      IF((NROW+IDIF-57).GT.0) GO TO 84
   82 IFLAG=1
      GO TO 40
   84 IFLAG=0
      GO TO 40
  130 CONTINUE
      DO 140 I=1,25
      PRINT 150
  150 FORMAT(' ')
  140 CONTINUE
      RETURN
  210 FORMAT(1H1,19X,10A8)
  220 FORMAT(///)
  230 FORMAT(12X,10(4X,A8))
  240 FORMAT(1X,A8,1X,A2,10F12.3)
      END
APDATA: PROC OPTIONS(MAIN);
     DCL DATE_LIMITS FIXED BINARY INITIAL(0),
         LOWER_DATE(DATE_LIMITS) CHAR(8) CONTROLLED,
         UPPER_DATE(DATE_LIMITS) CHAR(8) CONTROLLED;
     DCL HOUR_LIMITS FIXED BINARY INITIAL(0),
         LOWER_HOUR(HOUR_LIMITS) FIXED BINARY CONTROLLED
```

```
        UPPER_HOUR(HOUR_LIMITS) FIXED BINARY CONTROLLED;
DCL     1 FILE_HEADER,
            2 STATIONS FIXED BINARY,
            2 PARAMS FIXED BINARY,
            2 STARTING_DATE CHAR(6),
            2 ENDING_DATE CHAR(6);
DCL     1 FILE_STATION (FILE_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 FILE_PARAM (FILE_HEADER.PARAMS) CONTROLLED,
            2 NAME CHAR(8);
DCL     1 TODAY     CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (FILE_HEADER.PARAMS);
DCL     1 YESTERDAY CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (FILE_HEADER.PARAMS);
DCL     1 MET_PARAM (MET_PARAMS) CONTROLLED,
            2 POINTER FIXED BINARY,
            2 BANDS FIXED BINARY,
            2 BAND(10),
                3 LOWER_BAND FIXED DECIMAL (10,3),
                3 UPPER_BAND FIXED DECIMAL (10,3);
DCL     1 DUM_MET_PARAM (100)  CONTROLLED,
            2 POINTER FIXED BINARY,
            2 BANDS FIXED BINARY,
            2 BAND(10),
                3 LOWER_BAND FIXED DECIMAL (10,3),
                3 UPPER_BAND FIXED DECIMAL (10,3);
    DCL OUT_STATIONS FIXED BINARY (31,0),
        OUT_PARAMS   FIXED BINARY (31,0);
DCL     1 OUT_PARAM (OUT_PARAMS) CONTROLLED,
            2 POINTER FIXED BINARY,
            2 DELAY FIXED DECIMAL (2);
DCL     1 DUM_OUT_PARAM (100)  CONTROLLED,
            2 POINTER FIXED BINARY,
            2 DELAY FIXED DECIMAL (2);
DCL     1 OUTPUT_HEADER,
            2 LABEL (10) CHAR(8),
            2 STATIONS FIXED BINARY (31,0),
            2 PARAMS FIXED BINARY (31,0),
            2 NUMBER_OBS FIXED BINARY (31,0);
DCL     1 OUTPUT_STATION (OUTPUT_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 OUTPUT_PARAM (OUTPUT_HEADER.PARAMS) CONTROLLED,
            2 STATION CHAR(8),
            2 NAME CHAR(8);
DCL     1 DUM_OUTPUT_PARAM(100)  CONTROLLED,
            2 STATION CHAR(8),
            2 NAME CHAR(8);
DCL     1 OUTPUT_DATA CONTROLLED,
            2 DATE CHAR(8),
            2 HOUR CHAR(4),
            2 PARAM(OUTPUT_HEADER.PARAMS) ;
DCL     1 WRITE_DATA (OUTPUT_HEADER.PARAMS) CONTROLLED;
DCL       PRINT_TITLE (10) FLOAT DECIMAL (16),
          PRINT_DATE (NUMBER_OBS) CONTROLLED FLOAT DECIMAL (16),
          PRINT_HOUR (NUMBER_OBS) CONTROLLED FLOAT DECIMAL (6),
          PRINT_STATION (OUT_PARAMS) CONTROLLED FLOAT DECIMAL (16),
          PRINT_NAME    (OUT_PARAMS) CONTROLLED FLOAT DECIMAL (16),
          PRINT_PARAM (OUT_PARAMS,NUMBER_OBS) CONTROLLED;
DCL       CARD CHAR(80);
DCL       PRINT CHAR(3);
          PUT LIST('MASTER FILE HEADER INFORMATION')PAGE;
          READ FILE(MASTER) INTO (FILE_HEADER);
          PUT DATA(FILE_HEADER)SKIP;
          ALLOCATE FILE_STATION;
          READ FILE(MASTER) INTO (FILE_STATION);
          PUT DATA(FILE_STATION)SKIP;
          ALLOCATE FILE_PARAM;
          READ FILE(MASTER) INTO (FILE_PARAM);
          PUT DATA (FILE_PARAM)SKIP;
          ALLOCATE TODAY; ALLOCATE YESTERDAY;
NEXT:     /* BEGIN NEW CASE */
          CLOSE FILE(MASTER);
          ON ENDFILE(MASTER)  BEGIN;
```

```
        PUT LIST('END OF FILE SENSED ON MASTER')SKIP;
        GO TO NEXT ;END;
        READ FILE (MASTER) IGNORE(3);
        CLOSE FILE(CARDS);
GETCARD:GET EDIT(CARD)(A(80));
        WRITE FILE(CARDS) FROM (CARD);
        IF INDEX(CARD,'END ')¬=0 THEN GO TO END;
        IF INDEX(CARD,'STOP')¬=0 THEN GO TO STOP;
        GO TO GETCARD;
END:    CLOSE FILE(CARDS);
        CALL INPUT;
        CALL SEARCH;
        GO TO NEXT;
INPUT:  PROC;
DCL     PARAMS FIXED BINARY,BANDS FIXED BINARY,
        DELAY FIXED DECIMAL (2);
    DCL STATION CHAR(8),PARAM CHAR(8);
    DCL L1 CHAR(2),L2 CHAR(2),L3 CHAR(2),
        U1 CHAR(2),U2 CHAR(2),U3 CHAR(2);
        ON NAME(CARDS) BEGIN;
        PUT LIST('THE FOLLOWING DATA IS OUT OF ORDER: ',DATAFIELD)SKIP;
        GO TO NEXT;END;
        ON ENDFILE(CARDS)  BEGIN;
        PUT LIST('UNEXPECTED EOF ON CARD INPUT')SKIP;
        GO TO NEXT;END;
        PUT LIST('INPUT DATA')PAGE;
        GET FILE(CARDS) EDIT(LABEL)(10A(8));
        DO I=1 TO 10;
        UNSPEC(PRINT_TITLE(I))=UNSPEC(LABEL(I));
        END;
        PUT EDIT(LABEL)(10A(8))SKIP;
        GET FILE (CARDS) DATA(PRINT);
        PUT DATA(PRINT)SKIP;
        GET FILE(CARDS) DATA(DATE_LIMITS);
        FREE     LOWER_DATE; FREE     UPPER_DATE;
        IF DATE_LIMITS=0 THEN BEGIN;
        DATE_LIMITS=1;
        ALLOCATE LOWER_DATE; ALLOCATE UPPER_DATE;
        LOWER_DATE(1)=STARTING_DATE;
        UPPER_DATE(1)=ENDING_DATE;
        PUT DATA(LOWER_DATE(1),UPPER_DATE(1))SKIP;
        GO TO SKIP1;END;
        ELSE BEGIN;
        ALLOCATE LOWER_DATE; ALLOCATE UPPER_DATE;
        GET FILE(CARDS) LIST
                ((LOWER_DATE(I),UPPER_DATE(I) DO I=1 TO DATE_LIMITS));
        DO I=1 TO DATE_LIMITS;
        PUT DATA(LOWER_DATE(I),UPPER_DATE(I))SKIP;
        L1=SUBSTR(LOWER_DATE(I),1,2)  ;
        L2=SUBSTR(LOWER_DATE(I),4,2)  ;
        L3=SUBSTR(LOWER_DATE(I),7,2)  ;
        U1=SUBSTR(UPPER_DATE(I),1,2)  ;
        U2=SUBSTR(UPPER_DATE(I),4,2)  ;
        U3=SUBSTR(UPPER_DATE(I),7,2)  ;
        LOWER_DATE(I)=L1||L2||L3; UPPER_DATE(I)=U1||U2||U3;
        END;END;
SKIP1:  GET FILE(CARDS) DATA(HOUR_LIMITS);
        FREE     LOWER_HOUR; FREE     UPPER_HOUR;
        IF HOUR_LIMITS=0 THEN BEGIN;
        HOUR_LIMITS=1;
        ALLOCATE LOWER_HOUR; ALLOCATE UPPER_HOUR;
        LOWER_HOUR(1)=0;
        UPPER_HOUR(1)=23;
        PUT DATA(LOWER_HOUR(1),UPPER_HOUR(1))SKIP;
        LOWER_HOUR(1)=1;
        UPPER_HOUR(1)=24;
        GO TO SKIP2;END;
        ALLOCATE LOWER_HOUR; ALLOCATE UPPER_HOUR;
        GET FILE(CARDS) LIST
                ((LOWER_HOUR(I),UPPER_HOUR(I) DO I=1 TO HOUR_LIMITS));
        DO I=1 TO HOUR_LIMITS;
        PUT DATA(LOWER_HOUR(I),UPPER_HOUR(I))SKIP;
        LOWER_HOUR(I)=LOWER_HOUR(I)+1;
        UPPER_HOUR(I)=UPPER_HOUR(I)+1;
        END;
SKIP2:  PUT LIST('MET STATION REQUESTS')SKIP;
        MET_PARAMS=0;
        GET FILE(CARDS) DATA (MET_STATIONS);
        IF MET_STATIONS=0 THEN BEGIN;
        PUT LIST('NO MET STATIONS REQUESTED')SKIP;
        GO TO SKIP3;END;
```

```
        ALLOCATE DUM_MET_PARAM;
        DO I=1 TO MET_STATIONS;
        GET FILE(CARDS) DATA(STATION,PARAMS);
        PUT DATA(STATION)SKIP;
        IF PARAMS=0 THEN BEGIN;
        PUT LIST('NO PARAMETER REQUESTS')SKIP;
        GO TO NEXT;END;
        IF MET_PARAMS+PARAMS>100 THEN BEGIN;
        PUT LIST('NUMBER OF PARAMETERS GREATER THAN 100')SKIP;
        GO TO NEXT;END;
        DO N=1 TO FILE_HEADER.STATIONS;
        IF STATION=FILE_STATION(N).NAME THEN GO TO FIND_N1;
        END;/* N LOOP*/
        PUT LIST('STATION NAME ',STATION,' NOT RECOGNIZED');GO TO NEXT;
FIND_N1:DO J=1 TO PARAMS;
        GET FILE(CARDS) DATA(PARAM,BANDS);
        PUT DATA(PARAM)SKIP;
        IF BANDS>10 THEN BEGIN;
        PUT LIST('NUMBER OF BANDS GREATER THAN 10');
        GO TO NEXT;END;
        DO M=1 TO FILE_STATION (N).PARAMS;
        IF PARAM=FILE_PARAM(FILE_STATION(N).POINTER+M).NAME
        THEN GO TO FIND_M1;
        END;/* M LOOP*/
        PUT LIST('PARAMETER NAME ',PARAM,' NOT RECOGNIZED');GO TO NEXT;
FIND_M1:DUM_MET_PARAM(MET_PARAMS+J).POINTER=FILE_STATION(N).POINTER+M;
        DUM_MET_PARAM(MET_PARAMS+J).BANDS=BANDS;
        PUT LIST('BANDS')SKIP;
        DO K=1 TO BANDS;
        GET FILE(CARDS) LIST(
        DUM_MET_PARAM(MET_PARAMS+J).LOWER_BAND(K),
        DUM_MET_PARAM(MET_PARAMS+J).UPPER_BAND(K));
        PUT LIST(
        DUM_MET_PARAM(MET_PARAMS+J).LOWER_BAND(K),
        DUM_MET_PARAM(MET_PARAMS+J).UPPER_BAND(K));
        END; /* BAND LOOP */
        END; /* PARAM LOOP */
        MET_PARAMS=MET_PARAMS+PARAMS;
        END; /* STATION LOOP */
        FREE     MET_PARAM;
        ALLOCATE MET_PARAM;
        DO J=1 TO MET_PARAMS;
        MET_PARAM(J)=DUM_MET_PARAM(J);
        END;
        FREE DUM_MET_PARAM;
SKIP3:  PUT LIST('OUT STATION REQUESTS')SKIP;
        GET FILE(CARDS) DATA (OUT_STATIONS);
        IF OUT_STATIONS=0 THEN BEGIN;
        PUT LIST('NO OUT STATIONS REQUESTED')SKIP;
        GO TO NEXT;END;
        OUTPUT_HEADER.STATIONS=OUT_STATIONS;
        FREE     OUTPUT_STATION;
        ALLOCATE OUTPUT_STATION;
        OUT_PARAMS=0;
        ALLOCATE DUM_OUT_PARAM;
        ALLOCATE DUM_OUTPUT_PARAM;
        DO I=1 TO OUT_STATIONS;
        GET FILE(CARDS) DATA(STATION,PARAMS);
        PUT DATA(STATION)SKIP;
        IF PARAMS=0 THEN BEGIN;
        PUT LIST('NO PARAMETER REQUESTS')SKIP;
        GO TO NEXT;END;
        IF OUT_PARAMS+PARAMS>100 THEN BEGIN;
        PUT LIST('NUMBER OF PARAMETERS GREATER THAN 100')SKIP;
        GO TO NEXT;END;
        DO N=1 TO FILE_HEADER.STATIONS;
        IF STATION=FILE_STATION(N).NAME THEN GO TO FIND_N2;
        END;/* N LOOP*/
        PUT LIST('STATION NAME ',STATION,' NOT RECOGNIZED');GO TO NEXT;
FIND_N2:OUTPUT_STATION(I).NAME=STATION;
        OUTPUT_STATION(I).POINTER=OUT_PARAMS;
        OUTPUT_STATION(I).PARAMS=PARAMS;
        DO J=1 TO PARAMS;
        DELAY=0;
        GET FILE(CARDS) DATA(PARAM,DELAY);
        PUT DATA(PARAM,DELAY)SKIP;
        DUM_OUTPUT_PARAM(OUT_PARAMS+J).STATION=STATION;
        DUM_OUTPUT_PARAM(OUT_PARAMS+J).NAME=PARAM;
        DO M=1 TO FILE_STATION (N).PARAMS;
        IF PARAM=FILE_PARAM(FILE_STATION(N).POINTER+M).NAME
        THEN GO TO FIND_M2;
```

```
        END;/* M LOOP*/
        PUT LIST('PARAMETER NAME ',PARAM,' NOT RECOGNIZED');GO TO NEXT;
FIND_M2:DUM_OUT_PARAM(OUT_PARAMS+J).POINTER=FILE_STATION(N).POINTER+M;
        DUM_OUT_PARAM(OUT_PARAMS+J).DELAY=DELAY;
        END; /* PARAM LOOP */
        OUT_PARAMS=OUT_PARAMS+PARAMS;
        END; /*STATION LOOP */
        OUTPUT_HEADER.PARAMS=OUT_PARAMS;
        FREE     OUTPUT_PARAM;
        ALLOCATE OUTPUT_PARAM;
        FREE     OUT_PARAM;
        ALLOCATE OUT_PARAM;
        FREE     PRINT_STATION;
        ALLOCATE PRINT_STATION;
        FREE     PRINT_NAME;
        ALLOCATE PRINT_NAME;
        DO J=1 TO OUT_PARAMS;
        OUTPUT_PARAM(J)=DUM_OUTPUT_PARAM(J);
        OUT_PARAM(J)=DUM_OUT_PARAM(J);
        STATION=OUTPUT_PARAM(J).STATION;
        PARAM  =OUTPUT_PARAM(J).NAME    ;
        UNSPEC(PRINT_STATION(J))=UNSPEC(STATION);
        UNSPEC(PRINT_NAME   (J))=UNSPEC(PARAM);
        END;
        FREE DUM_OUTPUT_PARAM;
        FREE DUM_OUT_PARAM;
        CLOSE FILE(CARDS);
        RETURN;
        END INPUT;
SEARCH: PROC;
DCL     HOUR FIXED BINARY, HM1 FIXED BINARY;
DCL     DATE_KEY BIT(1) INITIAL('0'B), HOUR_KEY BIT(1) INITIAL ('0'B);
DCL     DATE_LIMIT FIXED BINARY, HOUR_LIMIT FIXED BINARY;
DCL     DELAY_HOUR FIXED BINARY;
DCL     YEAR CHAR(2), MONTH CHAR(2), DAY CHAR(2);
        DATE_LIMIT=1;
        NUMBER_OBS=0;
        FREE     OUTPUT_DATA;
        ALLOCATE OUTPUT_DATA;
        CLOSE FILE(SCRATCH);
        ON ENDFILE(MASTER)  BEGIN;
        PUT LIST('END OF FILE SENSED ON MASTER')SKIP;
        GO TO WRITE;END;
        READ FILE(MASTER) INTO (TODAY);
        YESTERDAY=TODAY;
DATES:  /* LOOP ON DATES - TEST BETWEEN DATE LIMITS */
        IF TODAY.DATE=LOWER_DATE(DATE_LIMIT) THEN DATE_KEY='1'B;
        IF TODAY.DATE=UPPER_DATE(DATE_LIMIT) THEN BEGIN;
        DATE_KEY='0'B;DATE_LIMIT=DATE_LIMIT+1; GO TO HOURS;
        END;
        IF DATE_KEY='0'B THEN GO TO NEWDATE;
HOURS:  /* LOOP ON HOURS - TEST BETWEEN HOUR LIMITS */
        HOUR_LIMIT=1; HOUR_KEY='0'B;
        DO HOUR=1 TO 24;
        IF HOUR=LOWER_HOUR(HOUR_LIMIT) THEN HOUR_KEY='1'B;
        IF HOUR=UPPER_HOUR(HOUR_LIMIT) THEN BEGIN;
        HOUR_KEY='0'B; HOUR_LIMIT=HOUR_LIMIT+1;GO TO TEST;
        END;
        IF HOUR_KEY='0'B THEN GO TO NEWHOUR;
TEST:   /* TEST PARAMETER BANDS */
        IF MET_PARAMS=0 THEN GO TO STORE;
        DO J=1 TO MET_PARAMS;
        DO K=1 TO MET_PARAM(J).BANDS;
/*      PUT DATA(MET_PARAM(J).LOWER_BAND(K))SKIP;
        PUT DATA(TODAY.PARAM(HOUR,MET_PARAM.POINTER(J)))SKIP;
        PUT DATA(MET_PARAM(J).UPPER_BAND(K))SKIP;          */
        IF TODAY.PARAM(HOUR,MET_PARAM.POINTER(J))>=
        MET_PARAM(J).LOWER_BAND(K)&
           TODAY.PARAM(HOUR,MET_PARAM.POINTER(J))<=
        MET_PARAM(J).UPPER_BAND(K)
        THEN GO TO NEWPAR ;
        END; /* BAND LOOP */
        GO TO NEWHOUR;
NEWPAR: END; /* PARAM LOOP */
STORE:  /* STORE OUPUT PARAMETERS */
        YEAR =SUBSTR(TODAY.DATE,1,2);
        MONTH=SUBSTR(TODAY.DATE,3,2);
        DAY  =SUBSTR(TODAY.DATE,5,2);
        OUTPUT_DATA.DATE=YEAR||'/'||MONTH||'/'||DAY;
        HM1=HOUR-1;
        PUT STRING (OUTPUT_DATA.HOUR) EDIT (HM1 )(F(2),X(2));
```

```
        DO J=1 TO OUT_PARAMS;
        DELAY_HOUR=HOUR-OUT_PARAM(J).DELAY;
        IF DELAY_HOUR<1 THEN BEGIN;
        OUTPUT_DATA.PARAM(J)=
        YESTERDAY.PARAM(24+DELAY_HOUR,OUT_PARAM.POINTER(J));
        GO TO SKIP;END;
        ELSE   BEGIN;
        OUTPUT_DATA.PARAM(J)=
        TODAY.PARAM(DELAY_HOUR,OUT_PARAM.POINTER(J));   END;
 SKIP:  END; /* PARAM LOOP */
        NUMBER_OBS=NUMBER_OBS+1;
/*      PUT DATA(OUTPUT_DATA)SKIP;              */
        WRITE FILE (SCRATCH) FROM (OUTPUT_DATA);
NEWHOUR:IF HOUR_LIMIT>HOUR_LIMITS THEN GO TO NEWDATE;
        END; /* HOUR LOOP */
NEWDATE:IF DATE_LIMIT>DATE_LIMITS THEN GO TO WRITE;
        YESTERDAY=TODAY;
        READ FILE(MASTER) INTO (TODAY);
        GO TO DATES;
WRITE:  /* SEARCH FINISHED - RECORD DATA ON OUTPUT TAPE */
        CLOSE FILE (MASTER);
        IF NUMBER_OBS=0 THEN BEGIN;
        PUT LIST('NO OBSERVATIONS WERE FOUND WITHIN BANDS')SKIP;
        RETURN;END;
        CLOSE FILE(SCRATCH);
        IF PRINT='NO ' THEN GO TO JUMP1;
        FREE      PRINT_DATE; FREE      PRINT_HOUR;
        ALLOCATE PRINT_DATE; ALLOCATE PRINT_HOUR;
        FREE      PRINT_PARAM;
        ALLOCATE PRINT_PARAM;
        IF ALLOCATION(PRINT_PARAM)='0'B THEN BEGIN;
        PUT LIST('INSUFFICIENT STORAGE AVAILABLE FOR DATA ARRAY')PAGE;
        GO TO NEXT; END;
JUMP1:  PUT LIST('DATA STORAGE HAS BEEN ALLOCATED')SKIP;
        WRITE FILE(OUTPUT) FROM (OUTPUT_HEADER);
        WRITE FILE(OUTPUT) FROM (OUTPUT_STATION);
        WRITE FILE(OUTPUT) FROM (OUTPUT_PARAM);
        ALLOCATE WRITE_DATA;
        DO N=1 TO NUMBER_OBS;
        READ FILE (SCRATCH) INTO (OUTPUT_DATA);
        IF PRINT='NO ' THEN GO TO JUMP2;
        UNSPEC(PRINT_DATE(N))=UNSPEC(OUTPUT_DATA.DATE);
        UNSPEC(PRINT_HOUR(N))=UNSPEC(OUTPUT_DATA.HOUR);
JUMP2:  DO M=1 TO OUT_PARAMS;
        IF PRINT='NO ' THEN GO TO JUMP3;
        PRINT_PARAM(M,N)=OUTPUT_DATA.PARAM(M);
JUMP3:  WRITE_DATA(M)=OUTPUT_DATA.PARAM(M);
        END;
        WRITE FILE(OUTPUT) FROM (WRITE_DATA);
        END;
        IF PRINT='NO ' THEN GO TO JUMP4;
/*      CLOSE FILE(SYSPRINT);           */
        CALL TWODPT(PRINT_PARAM(1,1),NUMBER_OBS,OUT_PARAMS,
        PRINT_TITLE(1),PRINT_DATE(1),PRINT_HOUR(1),PRINT_STATION(1),
        PRINT_NAME(1));
JUMP4:  RETURN;
        END SEARCH;
STOP:   END APDATA;
```

2. SEARCH2 Search of Master Tape with Option to Pass Information to FORT

```
      SUBROUTINE LCSSET(OUTDAT,NOBMAX,NPMAX,IERR)
C
      REAL*8 BLK
      COMMON/BLKSTR/BLK(1)
C
      IERR=1
      CALL BULK(125000)
      CALL POINTR(BLK,1,3)
      IF(IPTERR(DUMMY).GT.0)RETURN
      CALL PUTBLK('A       ',NOBMAX*NPMAX,4)
      IF(IPTERR(DUMMY).GT.0)RETURN
      CALL LCS(BLK(IGET('A       ')),OUTDAT,NOBMAX,NPMAX)
      IERR=0
      PRINT 5
    5 FORMAT('0BULK STORAGE HAS BEEN ALLOCATED')
      RETURN
C
      ENTRY RELLCS
C
      CALL FREE
      RETURN
      END
      SUBROUTINE LCS(A,OUTDAT,NOBMAX,NPMAX)
C
      REAL*8 TITLE(1),DATE(1),STA(1),PARAM(1)
      REAL*4 A(NOBMAX,1),OUTDAT(1),HOUR(1)
C
      RETURN
C
      ENTRY PUTLCS(OUTDAT,N)
C
      DO 10 J=1,NPMAX
      A(N,J)=OUTDAT(J)
   10 CONTINUE
      RETURN
C
      ENTRY GETLCS(OUTDAT,N)
C
      DO 20 J=1,NPMAX
      OUTDAT(J)=A(N,J)
   20 CONTINUE
      RETURN
C
      ENTRY PRINT(NOBS,NPARAM,TITLE,DATE,HOUR,STA,PARAM)
C
      CALL TWODPT(A,NOBS,NPARAM,TITLE,DATE,HOUR,STA,PARAM,NOBMAX)
      RETURN
C
      ENTRY FORTRN(NOBS,NPARAM,STA,PARAM)
C
      CALL FORK(NPARAM,NOBS,A,STA,PARAM,NOBMAX)
      RETURN
      END
      SUBROUTINE FORK(NPAR,NOBS,A,SNAM,PNAM,NOBMAX)
      DIMENSION A(1)
      DIMENSION SNAM(1),PNAM(1)
      IWON=1
      IIN=1
      IOUT=1
      DO 222 IPAR=1,NPAR
      DO 111 IOBS=1,NOBS
      A(IOUT)=A(IIN)
      IIN=IIN+1
  111 IOUT=IOUT+1
      IWON=IWON+NOBMAX
  222 IIN=IWON
      CALL FORT(NPAR,NOBS,A,SNAM,PNAM)
      RETURN
      END
      SUBROUTINE TWODPT(ARAY,ICT,JCT,TITLE,ROWT1,ROWT2,COLT1,COLT2,
     1ICTMAX)
C
C                                TWODPT PRINTS A TWO-DIMENSIONAL ARRAY.  IT
C                               PRINTS A TITLE LINE AND TWO LINES OF COLUMN
C                               TITLES, WHICH ARE FOLLOWED BY 50 LINES UP
C                               TO TEN COLUMNS WIDE PLUS THE INDEX COLUMN,
C                               WHICH CONTAINS DATE AND HOUR.
C                               ARGUMENTS ARE -
C
C                                     ARAY - THE TWO-DIMENSIONAL ARRAY
C                                            TO BE PRINTED.
```

```
C                                       ICT - NUMBER OF ROWS IN ARRAY.
C                                       JCT - NUMBER OF COLUMNS IN ARRAY.
C                                     TITLE - PAGE TITLE (10A8).
C                                     ROWT1 - ARRAY DIMENSIONED BY ICT, CON-
C                                             TAINING DATE FOR EACH ROW
C                                             (A8).
C                                     ROWT2 - ARRAY DIMENSIONED BY ICT, CON-
C                                             TAINING HOUR FOR EACH ROW
C                                             (A4, PRINTED AS A2).
C                                     COLT1 - ARRAY DIMENSIONED BY JCT, CON-
C                                             TAINING FIRST LINE OF TITLE
C                                             FOR EACH COLUMN (A8).
C                                     COLT2 - ARRAY DIMENSIONED BY JCT, CON-
C                                             TAINING SECOND LINE OF TITLE
C                                             FOR EACH COLUMN (A8).
C
      REAL*8 TITLE(10),ROWT1(1),COLT1(1),COLT2(1)
      REAL*4 ARAY(ICTMAX,1),ROWT2(1)
      II=1
      IJ=1
      IMAX=0
      JMAX=0
      MAX=0
      IFLAG=0
   20 IEND=JCT
   40 MIN=MAX+1
      MAX=MIN0(MAX+10,IEND)
      IF(IFLAG.NE.0) GO TO 47
   45 NROW=0
   46 CONTINUE
      PRINT 210,(TITLE(I),I=1,10)
   47 PRINT 220
      PRINT 230,(COLT1(JC),JC=MIN,MAX)
      PRINT 230,(COLT2(JC),JC=MIN,MAX)
      PRINT 230
      IMAX=MIN0(II+52,ICT)
      IDIF=IMAX-II+5
      NROW=NROW+IDIF
      JMAX=MIN0(IJ+9,JCT)
      DO 65 I=II,IMAX
      PRINT 240,ROWT1(I),ROWT2(I),(ARAY(I,J),J=IJ,JMAX)
   65 CONTINUE
      IF(IMAX.GE.ICT) GO TO 70
      II=IMAX+1
      GO TO 46
   70 IF(JMAX.GE.JCT) GO TO 130
   80 II=1
      IJ=JMAX+1
      IF((NROW+IDIF-57).GT.0) GO TO 84
   82 IFLAG=1
      GO TO 40
   84 IFLAG=0
      GO TO 40
  130 CONTINUE
      DO 140 I=1,25
      PRINT 150
  150 FORMAT(' ')
  140 CONTINUE
      RETURN
  210 FORMAT(1H1,19X,10A8)
  220 FORMAT(///)
  230 FORMAT(12X,10(4X,A8))
  240 FORMAT(1X,A8,1X,A2,10F12.3)
      END
APDATA: PROC OPTIONS(MAIN);
    DCL DATE_LIMITS FIXED BINARY INITIAL(0),
        LOWER_DATE(DATE_LIMITS) CHAR(8) CONTROLLED,
        UPPER_DATE(DATE_LIMITS) CHAR(8) CONTROLLED;
    DCL HOUR_LIMITS FIXED BINARY INITIAL(0),
        LOWER_HOUR(HOUR_LIMITS) FIXED BINARY CONTROLLED,
        UPPER_HOUR(HOUR_LIMITS) FIXED BINARY CONTROLLED;
DCL     1 FILE_HEADER,
            2 STATIONS FIXED BINARY,
            2 PARAMS FIXED BINARY,
            2 STARTING_DATE CHAR(6),
            2 ENDING_DATE CHAR(6);
DCL     1 FILE_STATION (FILE_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 FILE_PARAM (FILE_HEADER.PARAMS) CONTROLLED,
```

```
            2 NAME CHAR(8);
DCL     1 TODAY      CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (FILE_HEADER.PARAMS);
DCL     1 YESTERDAY CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (FILE_HEADER.PARAMS);
DCL     1 MET_PARAM (MET_PARAMS) CONTROLLED,
            2 POINTER FIXED BINARY,
            2 BANDS FIXED BINARY,
            2 BAND(10),
                3 LOWER_BAND FIXED DECIMAL (10,3),
                3 UPPER_BAND FIXED DECIMAL (10,3);
DCL     1 DUM_MET_PARAM (100)  CONTROLLED,
            2 POINTER FIXED BINARY,
            2 BANDS FIXED BINARY,
            2 BAND(10),
                3 LOWER_BAND FIXED DECIMAL (10,3),
                3 UPPER_BAND FIXED DECIMAL (10,3);
    DCL OUT_STATIONS FIXED BINARY (31,0),
        OUT_PARAMS   FIXED BINARY (31,0);
DCL     1 OUT_PARAM (OUT_PARAMS) CONTROLLED,
            2 POINTER FIXED BINARY,
            2 DELAY FIXED DECIMAL (2);
DCL     1 DUM_OUT_PARAM (100)  CONTROLLED,
            2 POINTER FIXED BINARY,
            2 DELAY FIXED DECIMAL (2);
DCL     1 OUTPUT_HEADER,
            2 LABEL (10) CHAR(8),
            2 STATIONS FIXED BINARY (31,0),
            2 PARAMS FIXED BINARY (31,0),
            2 NUMBER_OBS FIXED BINARY (31,0);
DCL     1 OUTPUT_STATION (OUTPUT_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 OUTPUT_PARAM (OUTPUT_HEADER.PARAMS) CONTROLLED,
            2 STATION CHAR(8),
            2 NAME CHAR(8);
DCL     1 DUM_OUTPUT_PARAM(100)  CONTROLLED,
            2 STATION CHAR(8),
            2 NAME CHAR(8);
DCL       OUTPUT_DATA_DATE CHAR(8), OUTPUT_DATA_HOUR CHAR(4),
          OUTPUT_DATA_PARAM(OUTPUT_HEADER.PARAMS) CONTROLLED;
DCL       PRINT_TITLE (10) FLOAT DECIMAL (16),
          PRINT_DATE (NUMBER_OBS_MAX) CONTROLLED FLOAT DECIMAL (16),
          PRINT_HOUR (NUMBER_OBS_MAX) CONTROLLED FLOAT DECIMAL ( 6),
          PRINT_STATION (OUT_PARAMS) CONTROLLED FLOAT DECIMAL (16),
          PRINT_NAME     (OUT_PARAMS) CONTROLLED FLOAT DECIMAL (16);
DCL       CARD CHAR(80);
DCL       OPT1 CHAR(3), OPT2 CHAR(3), OPT3  CHAR(3), OPTLIST CHAR(50);
          PUT LIST('MASTER FILE HEADER INFORMATION')PAGE;
          READ FILE(MASTER) INTO (FILE_HEADER);
          PUT DATA(FILE_HEADER)SKIP;
          ALLOCATE FILE_STATION;
          READ FILE(MASTER) INTO (FILE_STATION);
          PUT DATA(FILE_STATION)SKIP;
          ALLOCATE FILE_PARAM;
          READ FILE(MASTER) INTO (FILE_PARAM);
          PUT DATA (FILE_PARAM)SKIP;
          ALLOCATE TODAY; ALLOCATE YESTERDAY;
NEXT:     /* BEGIN NEW CASE */
          PUT LIST('CASE STARTED AT TIME',TIME)SKIP;
          CLOSE FILE(MASTER);
          ON ENDFILE(MASTER)  BEGIN;
          PUT LIST('END OF FILE SENSED ON MASTER')SKIP;
          GO TO NEXT ;END;
          READ FILE (MASTER) IGNORE(3);
          CLOSE FILE(CARDS);
GETCARD:  GET EDIT(CARD)(A(80));
          WRITE FILE(CARDS) FROM (CARD);
          IF INDEX(CARD,'END ')¬=0 THEN GO TO END;
          IF INDEX(CARD,'STOP')¬=0 THEN GO TO STOP;
          GO TO GETCARD;
END:      CLOSE FILE(CARDS);
          CALL INPUT;
          CALL SEARCH;
          GO TO NEXT;
INPUT:    PROC;
```

```
DCL       PARAMS FIXED BINARY,BANDS FIXED BINARY,
          DELAY FIXED DECIMAL (2);
     DCL  STATION CHAR(8),PARAM CHAR(8);
     DCL  L1 CHAR(2),L2 CHAR(2),L3 CHAR(2),
          U1 CHAR(2),U2 CHAR(2),U3 CHAR(2);
          ON NAME(CARDS) BEGIN;
          PUT LIST('THE FOLLOWING DATA IS OUT OF ORDER: ',DATAFIELD)SKIP;
          GO TO NEXT;END;
          ON ENDFILE(CARDS)  BEGIN;
          PUT LIST('UNEXPECTED EOF ON CARD INPUT')SKIP;
          GO TO NEXT;END;
          PUT LIST('INPUT DATA')PAGE;
          GET FILE(CARDS) EDIT(LABEL)(10A(8));
          DO I=1 TO 10;
          UNSPEC(PRINT_TITLE(I))=UNSPEC(LABEL(I));
          END;
          PUT EDIT(LABEL)(10A(8))SKIP;
          GET FILE (CARDS) DATA(OPTLIST);
          IF INDEX(OPTLIST,'FILE ')>0 THEN OPT1 ='YES';ELSE OPT1 ='NO ';
          IF INDEX(OPTLIST,'PRINT')>0 THEN OPT2 ='YES';ELSE OPT2 ='NO ';
          IF INDEX(OPTLIST,'FORT ')>0 THEN OPT3 ='YES';ELSE OPT3 ='NO ';
          PUT LIST('FILE=',OPT1,' PRINT=',OPT2,' FORT=',OPT3)SKIP;
          GET FILE(CARDS) DATA(DATE_LIMITS);
          FREE     LOWER_DATE; FREE     UPPER_DATE;
          IF DATE_LIMITS=0 THEN BEGIN;
          DATE_LIMITS=1;
          ALLOCATE LOWER_DATE; ALLOCATE UPPER_DATE;
          LOWER_DATE(1)=STARTING_DATE;
          UPPER_DATE(1)=ENDING_DATE;
          PUT DATA(LOWER_DATE(I),UPPER_DATE(1))SKIP;
          GO TO SKIP1;END;
          ELSE BEGIN;
          ALLOCATE LOWER_DATE; ALLOCATE UPPER_DATE;
          GET FILE(CARDS) LIST
                 ((LOWER_DATE(I),UPPER_DATE(I) DO I=1 TO DATE_LIMITS));
          DO I=1 TO DATE_LIMITS;
          PUT DATA(LOWER_DATE(I),UPPER_DATE(I))SKIP;
          L1=SUBSTR(LOWER_DATE(I),1,2)  ;
          L2=SUBSTR(LOWER_DATE(I),4,2)  ;
          L3=SUBSTR(LOWER_DATE(I),7,2)  ;
          U1=SUBSTR(UPPER_DATE(I),1,2)  ;
          U2=SUBSTR(UPPER_DATE(I),4,2)  ;
          U3=SUBSTR(UPPER_DATE(I),7,2)  ;
          LOWER_DATE(I)=L1||L2||L3; UPPER_DATE(I)=U1||U2||U3;
          END;END;
SKIP1:    GET FILE(CARDS) DATA(HOUR_LIMITS);
          FREE     LOWER_HOUR; FREE     UPPER_HOUR;
          IF HOUR_LIMITS=0 THEN BEGIN;
          HOUR_LIMITS=1;
          ALLOCATE LOWER_HOUR; ALLOCATE UPPER_HOUR;
          LOWER_HOUR(1)=0;
          UPPER_HOUR(1)=23;
          PUT DATA(LOWER_HOUR(1),UPPER_HOUR(1))SKIP;
          LOWER_HOUR(1)=1;
          UPPER_HOUR(1)=24;
          GO TO SKIP2;END;
          ALLOCATE LOWER_HOUR; ALLOCATE UPPER_HOUR;
          GET FILE(CARDS) LIST
                 ((LOWER_HOUR(I),UPPER_HOUR(I) DO I=1 TO HOUR_LIMITS));
          DO I=1 TO HOUR_LIMITS;
          PUT DATA(LOWER_HOUR(I),UPPER_HOUR(I))SKIP;
          LOWER_HOUR(I)=LOWER_HOUR(I)+1;
          UPPER_HOUR(I)=UPPER_HOUR(I)+1;
          END;
SKIP2:    PUT LIST('MET STATION REQUESTS')SKIP;
          MET_PARAMS=0;
          GET FILE(CARDS) DATA (MET_STATIONS);
          IF MET_STATIONS=0 THEN BEGIN;
          PUT LIST('NO MET STATIONS REQUESTED')SKIP;
          GO TO SKIP3;END;
          ALLOCATE DUM_MET_PARAM;
          DO I=1 TO MET_STATIONS;
          GET FILE(CARDS) DATA(STATION,PARAMS);
          PUT DATA(STATION)SKIP;
          IF PARAMS=0 THEN BEGIN;
          PUT LIST('NO PARAMETER REQUESTS')SKIP;
          GO TO NEXT;END;
          IF MET_PARAMS+PARAMS>100 THEN BEGIN;
          PUT LIST('NUMBER OF PARAMETERS GREATER THAN 100')SKIP;
          GO TO NEXT;END;
          DO N=1 TO FILE_HEADER.STATIONS;
```

```
        IF STATION=FILE_STATION(N).NAME THEN GO TO FIND_N1;
        END;/* N LOOP*/
        PUT LIST('STATION NAME ',STATION,' NOT RECOGNIZED');GO TO NEXT;
FIND_N1:DO J=1 TO PARAMS;
        GET FILE(CARDS) DATA(PARAM,BANDS);
        PUT DATA(PARAM)SKIP;
        IF BANDS>10 THEN BEGIN;
        PUT LIST('NUMBER OF BANDS GREATER THAN 10');
        GO TO NEXT;END;
        DO M=1 TO FILE_STATION (N).PARAMS;
        IF PARAM=FILE_PARAM(FILE_STATION(N).POINTER+M).NAME
        THEN GO TO FIND_M1;
        END;/* M LOOP*/
        PUT LIST('PARAMETER NAME ',PARAM,' NOT RECOGNIZED');GO TO NEXT;
FIND_M1:DUM_MET_PARAM(MET_PARAMS+J).POINTER=FILE_STATION(N).POINTER+M;
        DUM_MET_PARAM(MET_PARAMS+J).BANDS=BANDS;
        PUT LIST('BANDS')SKIP;
        DO K=1 TO BANDS;
        GET FILE(CARDS) LIST(
        DUM_MET_PARAM(MET_PARAMS+J).LOWER_BAND(K),
        DUM_MET_PARAM(MET_PARAMS+J).UPPER_BAND(K));
        PUT LIST(
        DUM_MET_PARAM(MET_PARAMS+J).LOWER_BAND(K),
        DUM_MET_PARAM(MET_PARAMS+J).UPPER_BAND(K));
        END; /* BAND LOOP */
        END; /* PARAM LOOP */
        MET_PARAMS=MET_PARAMS+PARAMS;
        END; /* STATION LOOP */
        FREE      MET_PARAM;
        ALLOCATE MET_PARAM;
        DO J=1 TO MET_PARAMS;
        MET_PARAM(J)=DUM_MET_PARAM(J);
        END;
        FREE DUM_MET_PARAM;
SKIP3:  PUT LIST('OUT STATION REQUESTS')SKIP;
        GET FILE(CARDS) DATA (OUT_STATIONS);
        IF OUT_STATIONS=0 THEN BEGIN;
        PUT LIST('NO OUT STATIONS REQUESTED')SKIP;
        GO TO NEXT;END;
        OUTPUT_HEADER.STATIONS=OUT_STATIONS;
        FREE      OUTPUT_STATION;
        ALLOCATE OUTPUT_STATION;
        OUT_PARAMS=0;
        ALLOCATE DUM_OUT_PARAM;
        ALLOCATE DUM_OUTPUT_PARAM;
        DO I=1 TO OUT_STATIONS;
        GET FILE(CARDS) DATA(STATION,PARAMS);
        PUT DATA(STATION)SKIP;
        IF PARAMS=0 THEN BEGIN;
        PUT LIST('NO PARAMETER REQUESTS')SKIP;
        GO TO NEXT;END;
        IF OUT_PARAMS+PARAMS>100 THEN BEGIN;
        PUT LIST('NUMBER OF PARAMETERS GREATER THAN 100')SKIP;
        GO TO NEXT;END;
        DO N=1 TO FILE_HEADER.STATIONS;
        IF STATION=FILE_STATION(N).NAME THEN GO TO FIND_N2;
        END;/* N LOOP*/
        PUT LIST('STATION NAME ',STATION,' NOT RECOGNIZED');GO TO NEXT;
FIND_N2:OUTPUT_STATION(I).NAME=STATION;
        OUTPUT_STATION(I).POINTER=OUT_PARAMS;
        OUTPUT_STATION(I).PARAMS=PARAMS;
        DO J=1 TO PARAMS;
        DELAY=0;
        GET FILE(CARDS) DATA(PARAM,DELAY);
        PUT DATA(PARAM,DELAY)SKIP;
        DUM_OUTPUT_PARAM(OUT_PARAMS+J).STATION=STATION;
        DUM_OUTPUT_PARAM(OUT_PARAMS+J).NAME=PARAM;
        DO M=1 TO FILE_STATION (N).PARAMS;
        IF PARAM=FILE_PARAM(FILE_STATION(N).POINTER+M).NAME
        THEN GO TO FIND_M2;
        END;/* M LOOP*/
        PUT LIST('PARAMETER NAME ',PARAM,' NOT RECOGNIZED');GO TO NEXT;
FIND_M2:DUM_OUT_PARAM(OUT_PARAMS+J).POINTER=FILE_STATION(N).POINTER+M;
        DUM_OUT_PARAM(OUT_PARAMS+J).DELAY=DELAY;
        END; /* PARAM LOOP */
        OUT_PARAMS=OUT_PARAMS+PARAMS;
        END; /*STATION LOOP */
        OUTPUT_HEADER.PARAMS=OUT_PARAMS;
        FREE      OUTPUT_PARAM;
        ALLOCATE OUTPUT_PARAM;
        FREE      OUT_PARAM;
```

```
        ALLOCATE OUT_PARAM;
        FREE     PRINT_STATION;
        ALLOCATE PRINT_STATION;
        FREE     PRINT_NAME;
        ALLOCATE PRINT_NAME;
        DO J=1 TO OUT_PARAMS;
        OUTPUT_PARAM(J)=DUM_OUTPUT_PARAM(J);
        OUT_PARAM(J)=DUM_OUT_PARAM(J);
        STATION=OUTPUT_PARAM(J).STATION;
        PARAM  =OUTPUT_PARAM(J).NAME   ;
        UNSPEC(PRINT_STATION(J))=UNSPEC(STATION);
        UNSPEC(PRINT_NAME   (J))=UNSPEC(PARAM);
        END;
        FREE DUM_OUTPUT_PARAM;
        FREE DUM_OUT_PARAM;
        CLOSE FILE(CARDS);
        RETURN;
        END INPUT;
SEARCH: PROC;
DCL     HOUR FIXED BINARY, HM1 FIXED BINARY;
DCL     DATE_KEY BIT(1) INITIAL('0'B), HOUR_KEY BIT(1) INITIAL ('0'B);
DCL     DATE_LIMIT FIXED BINARY, HOUR_LIMIT FIXED BINARY;
DCL     DELAY_HOUR FIXED BINARY;
DCL     YEAR CHAR(2), MONTH CHAR(2), DAY CHAR(2);
DCL     FORTRN EXT ENTRY, RETCODE FIXED BINARY EXT;
        DATE_LIMIT=1;
        NUMBER_OBS=0;
        FREE OUTPUT_DATA_PARAM; ALLOCATE OUTPUT_DATA_PARAM;
/* DONT SET MAX OBSERVATIONS - AT MOST ONE YEARS 'WORTH'          */
        NUMBER_OBS_MAX=MIN( 250000/OUT_PARAMS,32767);
        PUT DATA(NUMBER_OBS_MAX)SKIP;
        FREE PRINT_DATE    ; ALLOCATE PRINT_DATE;
        FREE PRINT_HOUR    ; ALLOCATE PRINT_HOUR;
        CALL LCSSET(OUTPUT_DATA_PARAM(1),NUMBER_OBS_MAX,OUT_PARAMS,
        IERR);
        IF IERR>0 THEN BEGIN;
        PUT LIST ('UNABLE TO ALLOCATE BULK STORAGE')SKIP;
        RETURN; END;
        ON ENDFILE(MASTER)  BEGIN;
        PUT LIST('END OF FILE SENSED ON MASTER')SKIP;
        GO TO WRITE;END;
        READ FILE(MASTER) INTO (TODAY);
        YESTERDAY=TODAY;
        PUT LIST('SEARCH STARTED  AT TIME ',TIME)SKIP;
DATES:  /* LOOP ON DATES - TEST BETWEEN DATE LIMITS */
        IF TODAY.DATE=LOWER_DATE(DATE_LIMIT) THEN DATE_KEY='1'B;
        IF TODAY.DATE=UPPER_DATE(DATE_LIMIT) THEN BEGIN;
        DATE_KEY='0'B;DATE_LIMIT=DATE_LIMIT+1; GO TO HOURS;
        END;
        IF DATE_KEY='0'B THEN GO TO NEWDATE;
HOURS:  /* LOOP ON HOURS - TEST BETWEEN HOUR LIMITS */
        HOUR_LIMIT=1; HOUR_KEY='0'B;
        DO HOUR=1 TO 24;
        IF HOUR=LOWER_HOUR(HOUR_LIMIT) THEN HOUR_KEY='1'B;
        IF HOUR=UPPER_HOUR(HOUR_LIMIT) THEN BEGIN;
        HOUR_KEY='0'B; HOUR_LIMIT=HOUR_LIMIT+1;GO TO TEST;
        END;
        IF HOUR_KEY='0'B THEN GO TO NEWHOUR;
TEST:   /* TEST PARAMETER BANDS */
        IF MET_PARAMS=0 THEN GO TO STORE;
        DO J=1 TO MET_PARAMS;
        DO K=1 TO MET_PARAM(J).BANDS;
        IF TODAY.PARAM(HOUR,MET_PARAM.POINTER(J))>=
        MET_PARAM(J).LOWER_BAND(K)&
           TODAY.PARAM(HOUR,MET_PARAM.POINTER(J))<=
        MET_PARAM(J).UPPER_BAND(K)
        THEN GO TO NEWPAR ;
        END; /* BAND LOOP */
        GO TO NEWHOUR;
NEWPAR: END; /* PARAM LOOP */
STORE:  /* STORE OUPUT PARAMETERS */
        YEAR =SUBSTR(TODAY.DATE,1,2);
        MONTH=SUBSTR(TODAY.DATE,3,2);
        DAY  =SUBSTR(TODAY.DATE,5,2);
        OUTPUT_DATA_DATE=YEAR||'/'||MONTH||'/'||DAY;
        HM1=HOUR-1;
        PUT STRING (OUTPUT_DATA_HOUR) EDIT (HM1 )(F(2),X(2));
        DO J=1 TO OUT_PARAMS;
        DELAY_HOUR=HOUR-OUT_PARAM(J).DELAY;
        IF DELAY_HOUR<1 THEN BEGIN;
        OUTPUT_DATA_PARAM(J)=
```

```
          YESTERDAY.PARAM(24+DELAY_HOUR,OUT_PARAM.POINTER(J));
          GO TO SKIP;END;
          ELSE    BEGIN;
          OUTPUT_DATA_PARAM(J)=
          TODAY.PARAM(DELAY_HOUR,OUT_PARAM.POINTER(J));    END;
 SKIP:    END; /* PARAM LOOP */
          IF NUMBER_OBS+1>NUMBER_OBS_MAX THEN BEGIN;
          PUT LIST('MAXIMUM NUMBER OF OBSERVATIONS EXCEEDED')SKIP;
          GO TO WRITE; END;
          NUMBER_OBS=NUMBER_OBS+1;
          N=NUMBER_OBS;
          UNSPEC(PRINT_DATE(N))=UNSPEC(OUTPUT_DATA_DATE);
          UNSPEC(PRINT_HOUR(N))=UNSPEC(OUTPUT_DATA_HOUR);
          CALL PUTLCS(OUTPUT_DATA_PARAM(1),N);
NEWHOUR:  IF HOUR_LIMIT>HOUR_LIMITS THEN GO TO NEWDATE;
          END; /* HOUR LOOP */
NEWDATE:  IF DATE_LIMIT>DATE_LIMITS THEN GO TO WRITE;
          YESTERDAY=TODAY;
          READ FILE(MASTER) INTO (TODAY);
          GO TO DATES;
WRITE:    /* SEARCH FINISHED */
          PUT LIST('SEARCH FINISHED AT TIME ',TIME)SKIP;
          CLOSE FILE (MASTER);
          IF NUMBER_OBS=0 THEN BEGIN;
          PUT LIST('NO OBSERVATIONS WERE FOUND WITHIN BANDS')SKIP;
          RETURN;END;
          IF OPT1='NO ' THEN GO TO JUMP1;
          WRITE FILE(OUTPUT) FROM (OUTPUT_HEADER);
          WRITE FILE(OUTPUT) FROM (OUTPUT_STATION);
          WRITE FILE(OUTPUT) FROM (OUTPUT_PARAM);
          DO N=1 TO NUMBER_OBS;
          CALL GETLCS(OUTPUT_DATA_PARAM(1),N);
          WRITE FILE(OUTPUT) FROM (OUTPUT_DATA_PARAM);
          END;
          CLOSE FILE (OUTPUT);
JUMP1:    IF OPT2 ='NO ' THEN GO TO JUMP2;
/*        CLOSE FILE(SYSPRINT);          */
          CALL PRINT(                    NUMBER_OBS,OUT_PARAMS,
          PRINT_TITLE(1),PRINT_DATE(1),PRINT_HOUR(1),PRINT_STATION(1),
          PRINT_NAME(1));
JUMP2:    IF OPT3='NO ' THEN GO TO JUMP3;
          FREE PRINT_DATE; FREE PRINT_HOUR;
          CALL FORTRAN(FORTRN,OB,
          NUMBER_OBS,OUTPUT_HEADER.PARAMS,PRINT_STATION(1),
          PRINT_NAME(1));
          IF RETCODE=1 THEN BEGIN;
          PUT LIST('ERROR IN FORTRN')SKIP;
          GO TO JUMP3; END;
          CALL FIOCLSE;
          PUT LIST('EXIT FROM FORT AT TIME',TIME)SKIP;
JUMP3:    CALL RELLCS;
          RETURN;
          END SEARCH;
STOP:
          PUT LIST('RUN  ENDED   AT TIME',TIME)SKIP;
          END APDATA;
      SUBROUTINE POINTR(BLK,MAXSIZ,JPRINT)
C
C     MANIPULATES POINTERS FOR VARIABLE STORAGE
C     ADJUSTS STORAGE IF STORAGE EXCEEDED
C     MULT=2  NON STANDARD INTEGER
C     MULT=4  STANDARD INTEGER OR REAL
C     MULT=8  NON STANDARD REAL
C
      REAL*8 BLK
      REAL*8 NAME,NAMLST,BLANK
      REAL NAM1,NAM2
      INTEGER ALLOC1,ALLOC2
C
      DIMENSION BLK(1)
      DIMENSION NAM1(2),NAM2(2)
C
      COMMON/LOCATE/ LSTM,MAXM,LSTB,MAXB,INDXM,INDXB,IPRINT
      COMMON/TABLES/NAMLST(100),LENLST(100),IPTLST(100),
     1LEN(100),MLT(100),NNAMS
      COMMON/BFLAGS/ IFLAG(100)
      COMMON/NAMBLK/NAME
      COMMON/PTERR/NPTERR
C
      EQUIVALENCE (NAME,NAM2)
C
```

```
      DATA BLANK/'        '/
      DATA MAXBLK/100000/,JUMP/1/
C
      NNAMS=0
      NPTERR=0
      DO 5 I=1,100
      NAMLST(I)=0
      IPTLST(I)=0
      LENLST(I)=0
      LEN(I)=0
      MLT(I)=0
      IFLAG(I)=0
    5 CONTINUE
      MAXM=MAXSIZ
      ILOC=ALLOC2(IERR,2*MAXM)
      IF(ILOC.EQ.0)GO TO 1040
      ILOCM=ILOC
      JLOC=LOCF(BLK)
      INDX =(ILOC-JLOC)/8
      INDXM=INDX
      LSTM=INDX+1
      MAXFM=MAXM
      MAXM=MAXM+INDX
      MAXB=MAXBLK
      ILOC=0
      IF(MAXB.EQ.0)GO TO 10
      ILOC=ALLOC1(IERR,2*MAXB)
      IF(ILOC.EQ.0.AND.JUMP.EQ.0)GO TO 1060
      IF(ILOC.EQ.0)MAXB=0
   10 CONTINUE
      ILOCB=ILOC
      INDX =(ILOC-JLOC)/8
      INDXB=INDX
      LSTB=INDX+1
      MAXFB=MAXB
      MAXB=MAXB+INDX
      CALL PRGSET(BLK)
      CALL REDSET(BLK)
      IPRINT=JPRINT
      DO 15 I=LSTM,MAXM
      BLK(I)=0
   15 CONTINUE
      IF(IPRINT.LT.2)GO TO 16
      LDUM=LSTM-INDXM
      MDUM=MAXM-INDXM
      PRINT 25,LDUM,MDUM
   16 CONTINUE
      IF(MAXFB.EQ.0)RETURN
      DO 20 I=LSTB,MAXB
      BLK(I)=0
   20 CONTINUE
      IF(IPRINT.LT.2)RETURN
      LDUM=LSTB-INDXB
      MDUM=MAXB-INDXB
      PRINT 25,LDUM,MDUM
   25 FORMAT('1POINTR HAS BEEN INITIALIZED'/
     1'0THE CONTAINER ARRAY HAS BEEN CLEARED FROM LOCATION'I10,
     1' TO LOCATION'I10)
      RETURN
C
      ENTRY BULK(MAXSIZ)
C
C
      JUMP=0
      MAXBLK=MAXSIZ
      RETURN
C
      ENTRY FREE
C
      IF(ILOCM.EQ.0)RETURN
      CALL FREE2(ILOCM,2*MAXFM)
      IF(ILOCB.EQ.0)RETURN
      CALL FREE1(ILOCB,2*MAXFB)
      RETURN
C
      ENTRY PUTPNT(NAM1,LENDUM,MULT)
C
C     PUT ARRAY NAME AND POINTER IN TABLE - ALLOCATE TO MAIN
C
      NAM2(1)=NAM1(1)
      NAM2(2)=NAM1(2)
```

```
      LENGTH=((LENDUM-1)*MULT+8)/8
      IF(LSTM+LENGTH-1.LE.MAXM)GO TO 30
C     STORAGE WILL BE EXCEEDED UNLESS GARBAGE IS CLEARED OUT
      CALL PURGE(DUMMY)
      IF(LSTM+LENGTH.GT.MAXM)GO TO 50
   30 CONTINUE
      IF(NNAMS+1.GT.100)CALL PURGE(DUMMY)
      IF(NNAMS+1.GT.100)CALL PURGEB(DUMMY)
      IF(NNAMS+1.GT.100)GO TO 1020
C     ENOUGH STORAGE - RECORD NAME AND POINTER IN TABLE
      NNAMS=NNAMS+1
      NAMLST(NNAMS)=NAME
      IPTLST(NNAMS)=LSTM
      LENLST(NNAMS)=LENGTH
      LEN(NNAMS)=LENDUM
      MLT(NNAMS)=MULT
      IFLAG(NNAMS)=0
      LSTM=LSTM+LENGTH
      IF(IPRINT.LT.2)RETURN
      IDUM=IPTLST(NNAMS)-INDXM
      PRINT 40,NAME,IDUM          ,LENLST(NNAMS),LEN(NNAMS),MLT(NNAMS),
     1IFLAG(NNAMS)
   40 FORMAT(17H0ENTRY FOR ARRAY A8,35H HAS BEEN MADE IN THE POINTER TAB
     1LE/9H POINTER=I10,9H DBL LEN=I5,10H ORIG LEN=I5,6H MULT=I5 ,
     26H BULK=I5)
      RETURN
C
      ENTRY PUTBLK(NAM1,LENDUM,MULT)
C
C     PUT ARRAY NAME AND POINTER IN TABLE - ALLOCATE TO BULK
C
      NAM2(1)=NAM1(1)
      NAM2(2)=NAM1(2)
      LENGTH=((LENDUM-1)*MULT+8)/8
   50 CONTINUE
      IF(LSTB+LENGTH-1.LE.MAXB)GO TO 60
C     STORAGE WILL BE EXCEEDED UNLESS GARBAGE IS CLEARED OUT
      CALL PURGEB(DUMMY)
      IF(LSTB+LENGTH.GT.MAXB)GO TO 1000
   60 CONTINUE
      IF(NNAMS+1.GT.100)CALL PURGE(DUMMY)
      IF(NNAMS+1.GT.100)CALL PURGEB(DUMMY)
      IF(NNAMS+1.GT.100)GO TO 1020
C     ENOUGH STORAGE - RECORD NAME AND POINTER IN TABLE
      NNAMS=NNAMS+1
      NAMLST(NNAMS)=NAME
      IPTLST(NNAMS)=LSTB
      LENLST(NNAMS)=LENGTH
      LEN(NNAMS)=LENDUM
      MLT(NNAMS)=MULT
      IFLAG(NNAMS)=1
      LSTB=LSTB+LENGTH
      IF(IPRINT.LT.2)RETURN
      IDUM=IPTLST(NNAMS)-INDXB
      PRINT 40,NAME,IDUM          ,LENLST(NNAMS),LEN(NNAMS),MLT(NNAMS),
     1IFLAG(NNAMS)
      RETURN
C
      ENTRY WIPOUT(NAM1)
C
C     RECORD NAME IN PURGE TABLE
C
      CALL GETN(NAM1,N)
      IF(N.EQ.0)RETURN
      NAMLST(N)=BLANK
      IF(IPRINT.LT.2)RETURN
      PRINT 70,NAME
   70 FORMAT(7H0ARRAY A8,35H HAS BEEN PLACED IN THE PURGE TABLE)
      RETURN
C
      ENTRY DUMP(NAM1,LUMP)
C
C     DUMP VARIABLE DIMENSION ARRAY ON STANDARD OUTPUT
C
      IF(IPRINT.EQ.0.OR.IPRINT.EQ.2)RETURN
      CALL GETN(NAM1,N)
      IF(N.EQ.0)RETURN
      IPOINT=IPTLST(N)
      IMINUS=INDXM
      IF(IFLAG(N).EQ.1)IMINUS=INDXB
      IDUM=IPTLST(N)-IMINUS
```

```
      PRINT  80,NAME,IDUM       ,LEN(N)
   80 FORMAT(15HODUMP OF ARRAY A8,9H POINTER=I10,8H LENGTH=I10)
      GO TO ( 90,110,130),LUMP
   90 CONTINUE
      IF(MLT(N).EQ.4)GO TO 100
      CALL PRTI1(BLK(IPOINT),LEN(N))
      RETURN
  100 CONTINUE
      CALL PRTI2(BLK(IPOINT),LEN(N))
      RETURN
  110 CONTINUE
      IF(MLT(N).EQ.8)GO TO 120
      CALL PRTR1(BLK(IPOINT),LEN(N))
      RETURN
  120 CONTINUE
      CALL PRTR2(BLK(IPOINT),LEN(N))
      RETURN
  130 CONTINUE
      IF(MLT(N).EQ.8)GO TO 140
      CALL PRTA1(BLK(IPOINT),LEN(N))
      RETURN
  140 CONTINUE
      CALL PRTA2(BLK(IPOINT),LEN(N))
      RETURN
C
      ENTRY GETPNT(NAM1,IPT)
C
C     GET POINTER CORRESPONDING TO ARRAY NAME
C
      IPT=0
      CALL GETN(NAM1,N)
      IF(N.GT.0)GO TO 190
      RETURN
  190 CONTINUE
      IPT=IPTLST(N)
      IF(IPRINT.LT.2)RETURN
      IMINUS=INDXM
      IF(IFLAG(N).EQ.1)IMINUS=INDXB
      IDUM=IPT-IMINUS
      PRINT 200,NAME,IDUM,IFLAG(N)
  200 FORMAT(19HOPOINTER FOR ARRAY A8,38H HAS BEEN EXTRACTED FROM POINTE
     1R TABLE/9H POINTER=I10,6H BULK=I10)
      RETURN
C
      ENTRY CLEAR(NAM1)
C
C     CLEARS ARRAY CORRESPONDING TO NAME
C
      CALL GETN(NAM1,N)
      IF(N.EQ.0)RETURN
      LENGTH=LENLST(N)
      IPT=IPTLST(N)
      DO 230 I=1,LENGTH
      BLK(IPT+I-1)=0
  230 CONTINUE
      IF(IPRINT.LT.2)RETURN
      PRINT 240,NAME
  240 FORMAT(7HOARRAY A8,17H HAS BEEN CLEARED)
      RETURN
C
C     PRINT ERROR MESSAGES
C
 1000 CONTINUE
      NPTERR=NPTERR+1
      PRINT 1010,NAME,LENGTH
 1010 FORMAT(54HOSTORAGE EXCEEDED DURING REQUEST FOR STORAGE OF ARRAY A8
     1,/8H LENGTH=I10)
      RETURN
 1020 CONTINUE
      NPTERR=NPTERR+1
      PRINT 1030
 1030 FORMAT(49HOMAXIMUM NUMBER OF ENTRIES IN NAME TABLE EXCEEDED)
      RETURN
 1040 CONTINUE
      NPTERR=NPTERR+1
      PRINT 1050
 1050 FORMAT('OMAIN STORAGE CANNOT BE ALLOCATED')
      RETURN
 1060 CONTINUE
      PRINT 1070
      NPTERR=NPTERR+1
```

```
 1070 FORMAT('0BULK STORAGE CANNOT BE ALLOCATED')
      RETURN
      END
      SUBROUTINE GETN(NAM1,N)
C
      REAL*8 NAME,NAMLST
      REAL NAM1,NAM2
C
      DIMENSION NAM1(2),NAM2(2)
C
      COMMON/LOCATE/ LSTM,MAXM,LSTB,MAXB,INDXM,INDXB,IPRINT
      COMMON/TABLES/NAMLST(100),LENLST(100),IPTLST(100),
     1LEN(100),MLT(100),NNAMS
      COMMON/NAMBLK/NAME
C
      EQUIVALENCE (NAME,NAM2)
C
      N=0
      NAM2(1)=NAM1(1)
      NAM2(2)=NAM1(2)
      IF(NNAMS.EQ.0)GO TO 20
      DO 10 N=1,NNAMS
      IF(NAMLST(N).EQ.NAME)RETURN
   10 CONTINUE
   20 CONTINUE
      N=0
      IF(IPRINT.LT.2)RETURN
      PRINT   30,NAME
   30 FORMAT(7H0ARRAY A8,34H CANNOT BE FOUND IN THE NAME TABLE)
      RETURN
      END
      SUBROUTINE REDSET(BLK)
C
C     REDEFINES ARRAY NAME
C
      REAL*8 BLK
      REAL*8 NAME,NAMLST,BLANK,DUMNAM
      REAL NAM1,NAM2
C
      DIMENSION BLK(1)
      DIMENSION NAM1(2),NAM2(2),DUM(2)
C
C
      COMMON/LOCATE/ LSTM,MAXM,LSTB,MAXB,INDXM,INDXB,IPRINT
      COMMON/TABLES/NAMLST(100),LENLST(100),IPTLST(100),
     1LEN(100),MLT(100),NNAMS
      COMMON/BFLAGS/ IFLAG(100)
      COMMON/NAMBLK/NAME
      COMMON/PTERR/NPTERR
C
C
      EQUIVALENCE (NAME,NAM2),(DUMMY,DUM(1))
C
      DATA BLANK/'        '/,DUMNAM/'DUMNAM'/
C
      RETURN
C
      ENTRY REDEF(NAM1,LENDUM,MULT)
C
      CALL GETN(NAM1,N)
      IF(N.GT.0)GO TO 30
C
C     NAME NOT IN TABLE SO PUT IT THERE
C
      CALL PUTPNT(NAME,LENDUM,MULT)
      RETURN
   30 CONTINUE
      LENNEW=((LENDUM-1)*MULT+8)/8
      LENOLD=LENLST(N)
      IF(LENNEW.EQ.LENOLD)RETURN
      IF(LENNEW.GT.LENOLD)GO TO 60
C
C     NEW LENGTH LESS THAN OLD LENGTH
C
      IF(NNAMS+1.LE.100)GO TO 50
      CALL PURGE(DUMMY)
      IF(NNAMS+1.LE.100)GO TO 50
      CALL PURGEB(DUMMY)
      IF(NNAMS+1.LE.100)GO TO 50
      NPTERR=NPTERR+1
      PRINT 40
```

```
   40 FORMAT(49HOMAXIMUM NUMBER OF ENTRIES IN NAME TABLE EXCEEDED)
      RETURN
   50 CONTINUE
      NNAMS=NNAMS+1
      L1=NNAMS-N-1
      DO 55 L=1,L1
      K=NNAMS-L
      NAMLST(K+1)=NAMLST(K)
      IPTLST(K+1)=IPTLST(K)
      LENLST(K+1)=LENLST(K)
      LEN   (K+1)=LEN   (K)
      MLT   (K+1)=MLT   (K)
      IFLAG(K+1)=IFLAG(K)
   55 CONTINUE
      LENLST(N)=LENNEW
      LEN(N)=LENDUM
      MLT(N)=MULT
      NAMLST(N+1   )=BLANK
      IPTLST(N+1   )=IPTLST(N)+LENNEW
      LENLST(N+1   )=LENOLD-LENNEW
      IF(IPRINT.LT.2)RETURN
      IMINUS=INDXM
      IF(IFLAG(N).EQ.1)IMINUS=INDXB
      IDUM=IPTLST(N)-IMINUS
      PRINT 56,NAME,IDUM          ,LENLST(N    ),LEN(N    ),MLT(N    ),
     1IFLAG(N    )
   56 FORMAT(17HOENTRY FOR ARRAY A8,35H HAS BEEN MADE IN THE POINTER TAB
     1LE/9H POINTER=I10,9H DBL LEN=I5,10H ORIG LEN=I5,6H MULT=I5 ,
     26H BULK=I5)
      RETURN
   60 CONTINUE
C
C     NEW LENGTH GREATER THAN OLD LENGTH
C
      NAMLST(N)=DUMNAM
      CALL PUTPNT(NAME,LENDUM,MULT)
      IF(NAMLST(N).EQ.DUMNAM)GO TO 90
      CALL GETN(DUMNAM,N)
      IF(N.GT.0)GO TO 90
      NPTERR=NPTERR+1
      RETURN
   90 CONTINUE
      NAMLST(N)=BLANK
      DO 100 L=1,LENOLD
      LSUB1=IPTLST(N)+L-1
      LSUB2=IPTLST(NNAMS)+L-1
      BLK(LSUB2)=BLK(LSUB1)
  100 CONTINUE
      RETURN
      END
      SUBROUTINE PRGSET(BLK)
C
C     SIFTS STORAGE AND ELIMINATES ARRAYS IN THE PURGE TABLE
C
      REAL*8 BLK
      REAL*8       NAMLST,BLANK
C
      DIMENSION BLK(1)
C
      COMMON/LOCATE/ LSTM,MAXM,LSTB,MAXB,INDXM,INDXB,IPRINT
      COMMON/TABLES/NAMLST(100),LENLST(100),IPTLST(100),
     1LEN(100),MLT(100),NNAMS
      COMMON/BFLAGS/ IFLAG(100)
C
      DATA BLANK/'        '/
C
      IMAIN=LSTM
      IBULK=LSTB
      RETURN
C
      ENTRY PURGE(LSTLOC)
C
      ISTART=IMAIN
      KEY=-1
      MAXBLK=MAXM
      IMINUS=INDXM
      IF(NNAMS.GT.0)GO TO 5
      LSTLOC=LSTM
      RETURN
C
      ENTRY PURGEB(LSTLOC)
```

```
C
      ISTART=IBULK
      KEY=0
      MAXBLK=MAXB
      IMINUS=INDXB
      IF(NNAMS.GT.0)GO TO 5
      LSTLOC=LSTB
      RETURN
C
    5 CONTINUE
      IMOV=ISTART
      INAM=0
      DO 40 N=1,NNAMS
      IF(KEY+ IFLAG(N).NE.0)GO TO 10
      INAM=INAM+1
      NAMLST(INAM)=NAMLST(N)
      IPTLST(INAM)=IPTLST(N)
      LENLST(INAM)=LENLST(N)
      LEN(INAM)=LEN(N)
      MLT(INAM)=MLT(N)
      IFLAG(INAM)=IFLAG(N)
      GO TO 40
   10 CONTINUE
      IF(NAMLST(N).EQ.BLANK     )GO TO 40
C     NAME NOT BLANK            - SO SAVE IT
      IMAX=LENLST(N)
      IF(IMOV.EQ.IPTLST(N))GO TO 30
      DO 20 I=1,IMAX
      ISUB1=IMOV+I-1
      ISUB2=IPTLST(N)+I-1
      BLK(ISUB1)=BLK(ISUB2)
   20 CONTINUE
   30 CONTINUE
      INAM=INAM+1
      NAMLST(INAM)=NAMLST(N)
      IPTLST(INAM)=IMOV
      LENLST(INAM)=IMAX
      LEN(INAM)=LEN(N)
      MLT(INAM)=MLT(N)
      IFLAG(INAM)=IFLAG(N)
      IMOV=IMOV+IMAX
   40 CONTINUE
      NNAMS=INAM
      IF(KEY.EQ.0)GO TO 41
      LSTM=IMOV
      GO TO 42
   41 CONTINUE
      LSTB=IMOV
   42 CONTINUE
      LSTBLK=IMOV
      LSTLOC=LSTBLK-IMINUS
      DO 45 I=LSTBLK,MAXBLK
      BLK(I)=0.
   45 CONTINUE
      IF ((NNAMS+1).GT.100)  GO TO 49
      IF(IPRINT.LT.2)RETURN
   49 CONTINUE
      PRINT 50
   50 FORMAT(  '1STORAGE HAS BEEN SIFTED')
C
      ENTRY STATUS
C
      IF ((NNAMS+1).GT.100)  GO TO 51
      IF(IPRINT.LT.2)RETURN
   51 CONTINUE
      PRINT 55
   55 FORMAT('0CURRENT STATUS OF CONTAINER ARRAY IS AS FOLLOWS'/
     11H07X2HN06X4HNAME3X7HPOINTER3X7HDBL LEN2X8HORIG LEN6X4HMULT
     26X4HBULK)
      DO 70 N=1,NNAMS
      IMINUS=INDXM
      IF(IFLAG(N).EQ.1)IMINUS=INDXB
      IDUM=IPTLST(N)-IMINUS
      PRINT 60,N,NAMLST(N),IDUM      ,LENLST(N),LEN(N),MLT(N),IFLAG(N)
   60 FORMAT(1I10,2XA8,8I10)
   70 CONTINUE
      RETURN
      END
      FUNCTION IGET(NAM1)
C
C     GET POINTER CORRESPONDING TO ARRAY NAME
```

```
C
      REAL*8              NAME,NAMLST,BLANK
      REAL NAM1,NAM2
C
      DIMENSION NAM1(2),NAM2(2)
C
      COMMON/LOCATE/ LSTM,MAXM,LSTB,MAXB,INDXM,INDXB,IPRINT
      COMMON/TABLES/NAMLST(100),LENLST(100),IPTLST(100),
     1LEN(100),MLT(100),NNAMS
      COMMON/BFLAGS/ IFLAG(100)
      COMMON/NAMBLK/NAME
      COMMON/PTERR/NPTERR
C
      EQUIVALENCE (NAME,NAM2)
C
      IPT=0
      CALL GETN(NAM1,N)
      IF(N.GT.0)GO TO 10
      RETURN
   10 CONTINUE
      IGET=IPTLST(N)
      IF(IPRINT.LT.2)RETURN
      IMINUS=INDXM
      IF(IFLAG(N).EQ.1)IMINUS=INDXB
      IDUM=IPTLST(N)-IMINUS
      PRINT 60 ,NAME,IDUM,IFLAG(N)
   60 FORMAT(19H0POINTER FOR ARRAY A8,38H HAS BEEN EXTRACTED FROM POINTE
     1R TABLE/9H POINTER=I10,6H BULK=I10)
      RETURN
      END
      SUBROUTINE PRTI1 (J,LENGTH)
C
C     PRINTS NON STANDARD INTEGER ARRAY
      INTEGER*2 J(1)
C
      PRINT 10,(J(L),L=1,LENGTH)
   10 FORMAT(24I5)
      RETURN
      END
      SUBROUTINE PRTI2 (J,LENGTH)
C
C     PRINTS STANDARD INTEGER ARRAY
C
      DIMENSION J(1)
C
      PRINT 10,(J(L),L=1,LENGTH)
   10 FORMAT(12I10)
      RETURN
      END
      SUBROUTINE PRTR1 (A,LENGTH)
C
C     PRINTS STANDARD REAL ARRAY
C
      DIMENSION A(1)
C
      PRINT 10,(A(L),L=1,LENGTH)
   10 FORMAT(8E15.7)
      RETURN
      END
      SUBROUTINE PRTR2 (A,LENGTH)
C
C     PRINTS NON STANDARD REAL ARRAY
C
      REAL*8 A(1)
C
      PRINT 10,(A(L),L=1,LENGTH)
   10 FORMAT(8E15.7)
      RETURN
      END
      SUBROUTINE PRTA1 (A,LENGTH)
C
C     PRINTS STANDARD HOLLERITH ARRAY
C
      DIMENSION A(1)
C
      PRINT 10,(A(L),L=1,LENGTH)
   10 FORMAT(1H 25A4)
      RETURN
      END
      SUBROUTINE PRTA2 (A,LENGTH)
C
```

```
C       PRINTS NON STANDARD HOLLERITH ARRAY
C
        REAL*8 A(1)
C
        PRINT 10,(A(L),L=1,LENGTH)
   10   FORMAT(1H 12A8)
        RETURN
        END
        FUNCTION IPTERR(DUMMY)
C
C       RETURNS NUMBER OF ERRORS THAT HAVE OCCURED IN POINTER ROUTINES
C
        COMMON/PTERR/NPTERR
C
        IPTERR=NPTERR
        NPTERR=0
        RETURN
        END
        FUNCTION ILAST(DUMMY)
C
C       RETURNS FIRST AVAILABLE LOCATION IN CONTAINER ARRAY
C
        COMMON/LOCATE/ LSTM,MAXM,LSTB,MAXB,INDXM,INDXB,IPRINT
C
        ILAST=LSTM-INDXM
        RETURN
        END
        FUNCTION ILASTB(DUMMY)
C
C       RETURNS FIRST AVAILABLE LOCATION IN CONTAINER ARRAY
C       OF BULK CORE
C
C
        COMMON/LOCATE/ LSTM,MAXM,LSTB,MAXB,INDXM,INDXB,IPRINT
C
        ILASTB=LSTB-INDXB
        RETURN
        END
FORTRAN  START
*
* CALLING SEQUENCE  CALL FORTRAN(FTNRTN,FLAG,A,B,C,...);
*                      FLAG < 0  IF NO PARAMETERS
*                      FLAG = 0  IF NO STRING DATA IN PARAMETER LIST
*                      FLAG = N  IF FIRST N PARAMETERS ARE STRINGS
*
         EXTRN IHEERRA
         ENTRY FIOCLSE
         BC    15,12(15)
         DC    X'07'
         DC    CL7'FORTRAN'
         STM   14,12,12(13)                SAVE REGISTERS
         BALR  12,0                        ESTABLISH BASE REGISTER
         USING *,12
BASE     LR    8,13                        LOAD PL/I SAVE AREA POINTER
         LA    13,SAVE                     FORTRAN'S SAVE AREA POINTER
         ST    13,8(8)                     STORE POINTER IN PL/I SAVE AREA
         ST    8,4(13)                     STORE PL/I SAVE AREA POINTER
*                                           IN FORTRAN'S SAVE AREA
         ST    15,SAVE+16                  STORE ADDRESS OF FORTRAN IN
*                                           FIFTH WORD OF ITS SAVE AREA
*                                           FOR TRACEBACK
*
* CREATE FORTRAN ENVIRONMENT THROUGH CALL TO IBFINT
* IBFINT SAVES REGISTERS IN ITS OWN SAVE AREA BUT EXPECTS REGISTER 13
* TO POINT TO SAVE AREA OF SUPERVISOR -- FORTRAN.
*
         L     15,VIBCOM                   LOAD ADDRESS OF IBCOM
         BAL   14,64(15)                   CALL IBFINT TO INITIALIZE FORT.
         L     13,SAVE+4                   LOAD PL/I SAVE AREA POINTER
         L     1,24(13)                    LOAD PAREMETER LIST ADDRESS
         L     2,0(1)
         L     15,0(2)                     LOAD ADDRESS OF FORTRAN SUBRTN.
         LA    13,SAVE                     LOAD SAVE AREA POINTER
* TEST FLAG FOR ANALYSIS OF PARAMETER LIST
         L     3,4(1)
         L     3,0(3)                      LOAD FLAG
         LTR   3,3                         TEST FLAG FOR PARAMETERS
         BC    NOPARM,CALL                 BRANCH IF FLAG<0  NO PARAMETERS
         LA    1,8(1)                      SET PARAMETER LIST POINTER
         BC    NOSTRG,CALL                 BRANCH IF FLAG=0  NO STRINGS
* OBTAIN STRING ADDRESS FROM 1ST WORD OF STRING DOPE VECTOR -- SDV
```

```
         SR     6,6                          ZERO REGISTER
LOOP     L      4,0(6,1)                     LOAD ADDRESS OF SDV
         L      2,0(4)                       LOAD 1ST WORD OF SDV
         ST     2,0(6,1)                     STORE ADDRESS IN PARAMETER LIST
         AH     6,FOUR                       INCREASE REGISTER FOR NEXT PARM
         BCT    3,LOOP                       BRANCH IF N NOT ZERO
*
CALL     BALR   14,15                        CALL FORTRAN SUBROUTINE
*
         L      5,RCADDR                     LOAD ADDRESS OF RETURN CODE
         MVC    0(4,5),RCNORM                PUT ZERO IN RETURN CODE
* TEST RETURN CODE IN REGISTER 15 FOR TERMINAL ERROR, NORMAL RETURN,
* OR STOP.  ALL OTHERS ASSUMED NORMAL.
         LTR    15,15                        TEST RETURN CODE
         BC     ZERO,RETCODE0                BRANCH IF CODE IS 0
         C      15,TERMERR                   TEST FOR TERMINAL ERROR
         BC     NOERR,CONTINUE               BRANCH IF NOT A TERMINAL ERROR
         MVC    0(4,5),RCERR                 PUT 1 IN RETURN CODE : ERROR
         BC     ALWAYS,CONTINUE
RETCODE0 CLI    12(13),X'FF'                 TEST FOR NORMAL RETURN
         BC     ZERO,CONTINUE                BRANCH IF NORMAL RETURN
         MVC    0(4,5),RCSTOP                PUT -1 IN RETURN CODE : STOP
* RESTORE PL/I INTERRUPT SYSTEM
CONTINUE SPIE   IHEERRA,((1,13),15)
*
         L      13,SAVE+4                    LOAD PL/I SAVE AREA POINTER
         LM     14,12,12(13)                 RESTORE REGISTERS
         BR     14                           RETURN TO PL/I
*
* CLOSE FORTRAN I/O DATA SETS THROUGH CALL TO IBEXIT
* IBEXIT SAVES REGISTERS IN ITS OWN SAVE AREA BUT RETURNS VIA FOURTH
* WORD IN SAVE AREA OF SUPERVISOR -- FORTRAN.  REGISTER 14 IN CALL IS
* USED FOR A RETURN CODE.
*
         CNOP   0,4
FIOCLSE  BC     15,12(15)
         DC     X'07'
         DC     CL7'FIOCLSE'
         STM    14,12,12(13)                 SAVE PL/I REGISTERS
         L      12,POINTER-FIOCLSE(15)       ESTABLISH BASE REGISTER
         ST     13,REG13                     STORE PL/I SAVE AREA POINTER
         MVC    REG14(4),SAVE+12             REPLACE FOURTH WORD OF SAVE
         MVC    SAVE+12(4),ARETURN             AREA BY THE RETURN ADDRESS
         L      15,VIBCOM                    LOAD ADDRESS OF IBCOM
         BAL    14,68(15)                    CALL IBEXIT TO CLOSE I/O D.S.
         DC     AL2(0)                       AL2(RETURN CODE)
         CNOP   0,4
* RESTORE PL/I INTERRUPT SYSTEM
RETURN   SPIE   IHEERRA,((1,13),15)
*
         MVC    SAVE+12(4),REG14             RESTORE 4TH WORD OF SAVE AREA
         L      13,REG13                     LOAD PI/I SAVE AREA POINTER
         LM     14,12,12(13)                 RESTORE PL/I REGISTERS
         BR     14                           RETURN TO PL/I
*
* STORAGE SECTION FOR FORTRAN
SAVE     DC     18F'0'
VIBCOM   DC     V(IBCOM#)
POINTER  DC     A(BASE)
ARETURN  DC     A(RETURN)
RCADDR   DC     A(RETCODE)
TERMERR  DC     XL4'00000010'
REG13    DC     F'0'
REG14    DC     F'0'
RCERR    DC     F'1'
RCNORM   DC     F'0'
RCSTOP   DC     F'-1'
FOUR     DC     H'4'
NOPARM   EQU    4
NOERR    EQU    7
ZERO     EQU    8
NOSTRG   EQU    8
ALWAYS   EQU    15
*
* STORAGE AREA FOR RETURN CODE TO PL/I
* DECLARE RETCODE FIXED BINARY (31,0) EXTERNAL;
RETCODE  CSECT
CODE     DC     F'0'                         INITIALIZE CODE TO ZERO
*
         END
```

3. SEARCH3 Search of Master Tape Omitting Missing Data

```
      SUBROUTINE TWODPT(ARAY,ICT,JCT,TITLE,ROWT1,ROWT2,COLT1,COLT2)
C
C                                TWODPT PRINTS A TWO-DIMENSIONAL ARRAY.  IT
C                              PRINTS A TITLE LINE AND TWO LINES OF COLUMN
C                              TITLES, WHICH ARE FOLLOWED BY 50 LINES UP
C                              TO TEN COLUMNS WIDE PLUS THE INDEX COLUMN,
C                              WHICH CONTAINS DATE AND HOUR.
C                              ARGUMENTS ARE -
C
C                                    ARAY - THE TWO-DIMENSIONAL ARRAY
C                                           TO BE PRINTED.
C                                     ICT - NUMBER OF ROWS IN ARRAY.
C                                     JCT - NUMBER OF COLUMNS IN ARRAY.
C                                   TITLE - PAGE TITLE (10A8).
C                                   ROWT1 - ARRAY DIMENSIONED BY ICT, CON-
C                                           TAINING DATE FOR EACH ROW
C                                           (A8).
C                                   ROWT2 - ARRAY DIMENSIONED BY ICT, CON-
C                                           TAINING HOUR FOR EACH ROW
C                                           (A4, PRINTED AS A2).
C                                   COLT1 - ARRAY DIMENSIONED BY JCT, CON-
C                                           TAINING FIRST LINE OF TITLE
C                                           FOR EACH COLUMN (A8).
C                                   COLT2 - ARRAY DIMENSIONED BY JCT, CON-
C                                           TAINING SECOND LINE OF TITLE
C                                           FOR EACH COLUMN (A8).
C
      REAL*8 TITLE(10),ROWT1(1),COLT1(1),COLT2(1)
      REAL*4 ARAY(ICT,1),ROWT2(1)
      II=1
      IJ=1
      IMAX=0
      JMAX=0
      MAX=0
      IFLAG=0
   20 IEND=JCT
   40 MIN=MAX+1
      MAX=MIN0(MAX+10,IEND)
      IF(IFLAG.NE.0) GO TO 47
   45 NROW=0
   46 CONTINUE
      PRINT 210,(TITLE(I),I=1,10)
   47 PRINT 220
      PRINT 230,(COLT1(JC),JC=MIN,MAX)
      PRINT 230,(COLT2(JC),JC=MIN,MAX)
      PRINT 230
      IMAX=MIN0(II+52,ICT)
      IDIF=IMAX-II+5
      NROW=NROW+IDIF
      JMAX=MIN0(IJ+9,JCT)
      DO 65 I=II,IMAX
      PRINT 240,ROWT1(I),ROWT2(I),(ARAY(I,J),J=IJ,JMAX)
   65 CONTINUE
      IF(IMAX.GE.ICT) GO TO 70
      II=IMAX+1
      GO TO 46
   70 IF(JMAX.GE.JCT) GO TO 130
   80 II=1
      IJ=JMAX+1
      IF((NROW+IDIF-57).GT.0) GO TO 84
   82 IFLAG=1
```

```
    GO TO 40
 84 IFLAG=0
    GO TO 40
130 CONTINUE
    DO 140 I=1,25
    PRINT 150
150 FORMAT(' ')
140 CONTINUE
    RETURN
210 FORMAT(1H1,19X,10A8)
220 FORMAT(///)
230 FORMAT(12X,10(4X,A8))
240 FORMAT(1X,A8,1X,A2,10F12.3)
    END
APDATA: PROC OPTIONS(MAIN);
    DCL DATE_LIMITS FIXED BINARY INITIAL(0),
        LOWER_DATE(DATE_LIMITS) CHAR(8) CONTROLLED,
        UPPER_DATE(DATE_LIMITS) CHAR(8) CONTROLLED;
    DCL HOUR_LIMITS FIXED BINARY INITIAL(0),
        LOWER_HOUR(HOUR_LIMITS) FIXED BINARY CONTROLLED,
        UPPER_HOUR(HOUR_LIMITS) FIXED BINARY CONTROLLED;
DCL     1 FILE_HEADER,
            2 STATIONS FIXED BINARY,
            2 PARAMS FIXED BINARY,
            2 STARTING_DATE CHAR(6),
            2 ENDING_DATE CHAR(6);
DCL     1 FILE_STATION (FILE_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 FILE_PARAM (FILE_HEADER.PARAMS) CONTROLLED,
            2 NAME CHAR(8);
DCL     1 TODAY      CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (FILE_HEADER.PARAMS);
DCL     1 YESTERDAY CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (FILE_HEADER.PARAMS);
DCL     1 MET_PARAM (MET_PARAMS) CONTROLLED,
            2 POINTER FIXED BINARY,
            2 BANDS FIXED BINARY,
            2 BAND(10),
                3 LOWER_BAND FIXED DECIMAL (10,3),
                3 UPPER_BAND FIXED DECIMAL (10,3);
DCL     1 DUM_MET_PARAM (100)  CONTROLLED,
            2 POINTER FIXED BINARY,
            2 BANDS FIXED BINARY,
            2 BAND(10),
                3 LOWER_BAND FIXED DECIMAL (10,3),
                3 UPPER_BAND FIXED DECIMAL (10,3);
    DCL OUT_STATIONS FIXED BINARY (31,0),
        OUT_PARAMS   FIXED BINARY (31,0);
DCL     1 OUT_PARAM (OUT_PARAMS) CONTROLLED,
            2 POINTER FIXED BINARY,
            2 DELAY FIXED DECIMAL (2);
DCL     1 DUM_OUT_PARAM (100)  CONTROLLED,
            2 POINTER FIXED BINARY,
            2 DELAY FIXED DECIMAL (2);
DCL     1 OUTPUT_HEADER,
```

```
            2 LABEL (10) CHAR(8),
            2 STATIONS FIXED BINARY (31,0),
            2 PARAMS FIXED BINARY (31,0),
            2 NUMBER_OBS FIXED BINARY (31,0);
DCL     1 OUTPUT_STATION (OUTPUT_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 OUTPUT_PARAM (OUTPUT_HEADER.PARAMS) CONTROLLED,
            2 STATION CHAR(8),
            2 NAME CHAR(8);
DCL     1 DUM_OUTPUT_PARAM(100)  CONTROLLED,
            2 STATION CHAR(8),
            2 NAME CHAR(8);
DCL     1 OUTPUT_DATA CONTROLLED,
            2 DATE CHAR(8),
            2 HOUR CHAR(4),
            2 PARAM(OUTPUT_HEADER.PARAMS) ;
DCL     1 WRITE_DATA (OUTPUT_HEADER.PARAMS) CONTROLLED;
DCL      PRINT_TITLE (10) FLOAT DECIMAL (16),
         PRINT_DATE (NUMBER_OBS) CONTROLLED FLOAT DECIMAL (16),
         PRINT_HOUR (NUMBER_OBS) CONTROLLED FLOAT DECIMAL (6),
         PRINT_STATION (OUT_PARAMS) CONTROLLED FLOAT DECIMAL (16),
         PRINT_NAME    (OUT_PARAMS) CONTROLLED FLOAT DECIMAL (16),
         PRINT_PARAM (OUT_PARAMS,NUMBER_OBS) CONTROLLED;
DCL      CARD CHAR(80);
DCL      PRINT CHAR(3);
         PUT LIST('MASTER FILE HEADER INFORMATION')PAGE;
         READ FILE(MASTER) INTO (FILE_HEADER);
         PUT DATA(FILE_HEADER)SKIP;
         ALLOCATE FILE_STATION;
         READ FILE(MASTER) INTO (FILE_STATION);
         PUT DATA(FILE_STATION)SKIP;
         ALLOCATE FILE_PARAM;
         READ FILE(MASTER) INTO (FILE_PARAM);
         PUT DATA (FILE_PARAM)SKIP;
         ALLOCATE TODAY; ALLOCATE YESTERDAY;
NEXT:    /* BEGIN NEW CASE */
         CLOSE FILE(MASTER);
         ON ENDFILE(MASTER)  BEGIN;
         PUT LIST('END OF FILE SENSED ON MASTER')SKIP;
         GO TO NEXT ;END;
         READ FILE (MASTER) IGNORE(3);
         CLOSE FILE(CARDS);
GETCARD:GET EDIT(CARD)(A(80));
         WRITE FILE(CARDS) FROM (CARD);
         IF INDEX(CARD,'END ')¬=0 THEN GO TO END;
         IF INDEX(CARD,'STOP')¬=0 THEN GO TO STOP;
         GO TO GETCARD;
END:     CLOSE FILE(CARDS);
         CALL INPUT;
         CALL SEARCH;
         GO TO NEXT;
INPUT:   PROC;
DCL      PARAMS FIXED BINARY,BANDS FIXED BINARY,
         DELAY FIXED DECIMAL (2);
     DCL STATION CHAR(8),PARAM CHAR(8);
     DCL L1 CHAR(2),L2 CHAR(2),L3 CHAR(2),
         U1 CHAR(2),U2 CHAR(2),U3 CHAR(2);
         ON NAME(CARDS) BEGIN;
         PUT LIST('THE FOLLOWING DATA IS OUT OF ORDER: ',DATAFIELD)SKIP;
```

```
        GO TO NEXT;END;
        ON ENDFILE(CARDS)  BEGIN;
        PUT LIST('UNEXPECTED EOF ON CARD INPUT')SKIP;
        GO TO NEXT;END;
        PUT LIST('INPUT DATA')PAGE;
        GET FILE(CARDS) EDIT(LABEL)(10A(8));
        DO I=1 TO 10;
        UNSPEC(PRINT_TITLE(I))=UNSPEC(LABEL(I));
        END;
        PUT EDIT(LABEL)(10A(8))SKIP;
        GET FILE (CARDS) DATA(PRINT);
        PUT DATA(PRINT)SKIP;
        GET FILE(CARDS) DATA(DATE_LIMITS);
        FREE      LOWER_DATE; FREE      UPPER_DATE;
        IF DATE_LIMITS=0 THEN BEGIN;
        DATE_LIMITS=1;
        ALLOCATE LOWER_DATE; ALLOCATE UPPER_DATE;
        LOWER_DATE(1)=STARTING_DATE;
        UPPER_DATE(1)=ENDING_DATE;
        PUT DATA(LOWER_DATE(1),UPPER_DATE(1))SKIP;
        GO TO SKIP1;END;
        ELSE BEGIN;
        ALLOCATE LOWER_DATE; ALLOCATE UPPER_DATE;
        GET FILE(CARDS) LIST
                ((LOWER_DATE(I),UPPER_DATE(I) DO I=1 TO DATE_LIMITS));
        DO I=1 TO DATE_LIMITS;
        PUT DATA(LOWER_DATE(I),UPPER_DATE(I))SKIP;
        L1=SUBSTR(LOWER_DATE(I),1,2) +;
        L2=SUBSTR(LOWER_DATE(I),4,2) +;
        L3=SUBSTR(LOWER_DATE(I),7,2) +;
        U1=SUBSTR(UPPER_DATE(I),1,2) +;
        U2=SUBSTR(UPPER_DATE(I),4,2)  ;
        U3=SUBSTR(UPPER_DATE(I),7,2)  ;
        LOWER_DATE(I)=L1||L2||L3; UPPER_DATE(I)=U1||U2||U3;
        END;END;
SKIP1:  GET FILE(CARDS) DATA(HOUR_LIMITS);
        FREE      LOWER_HOUR; FREE      UPPER_HOUR;
        IF HOUR_LIMITS=0 THEN BEGIN;
        HOUR_LIMITS=1;
        ALLOCATE LOWER_HOUR; ALLOCATE UPPER_HOUR;
        LOWER_HOUR(1)=0;
        UPPER_HOUR(1)=23;
        PUT DATA(LOWER_HOUR(1),UPPER_HOUR(1))SKIP;
        LOWER_HOUR(1)=1;
        UPPER_HOUR(1)=24;
        GO TO SKIP2;END;
        ALLOCATE LOWER_HOUR; ALLOCATE UPPER_HOUR;
        GET FILE(CARDS) LIST
                ((LOWER_HOUR(I),UPPER_HOUR(I) DO I=1 TO HOUR_LIMITS));
        DO I=1 TO HOUR_LIMITS;
        PUT DATA(LOWER_HOUR(I),UPPER_HOUR(I))SKIP;
        LOWER_HOUR(I)=LOWER_HOUR(I)+1;
        UPPER_HOUR(I)=UPPER_HOUR(I)+1;
        END;
SKIP2:  PUT LIST('MET STATION REQUESTS')SKIP;
        MET_PARAMS=0;
        GET FILE(CARDS) DATA (MET_STATIONS);
        IF MET_STATIONS=0 THEN BEGIN;
        PUT LIST('NO MET STATIONS REQUESTED')SKIP;
        GO TO SKIP3;END;
        ALLOCATE DUM_MET_PARAM;
```

```
        DO I=1 TO MET_STATIONS;
        GET FILE(CARDS) DATA(STATION,PARAMS);
        PUT DATA(STATION)SKIP;
        IF PARAMS=0 THEN BEGIN;
        PUT LIST('NO PARAMETER REQUESTS')SKIP;
        GO TO NEXT;END;
        IF MET_PARAMS+PARAMS>100 THEN BEGIN;
        PUT LIST('NUMBER OF PARAMETERS GREATER THAN 100')SKIP;
        GO TO NEXT;END;
        DO N=1 TO FILE_HEADER.STATIONS;
        IF STATION=FILE_STATION(N).NAME THEN GO TO FIND_N1;
        END;/* N LOOP*/
        PUT LIST('STATION NAME ',STATION,' NOT RECOGNIZED');GO TO NEXT;
FIND_N1:DO J=1 TO PARAMS;
        GET FILE(CARDS) DATA(PARAM,BANDS);
        PUT DATA(PARAM)SKIP;
        IF BANDS>10 THEN BEGIN;
        PUT LIST('NUMBER OF BANDS GREATER THAN 10');
        GO TO NEXT;END;
        DO M=1 TO FILE_STATION (N).PARAMS;
        IF PARAM=FILE_PARAM(FILE_STATION(N).POINTER+M).NAME
        THEN GO TO FIND_M1;
        END;/* M LOOP*/
        PUT LIST('PARAMETER NAME ',PARAM,' NOT RECOGNIZED');GO TO NEXT;
FIND_M1:DUM_MET_PARAM(MET_PARAMS+J).POINTER=FILE_STATION(N).POINTER+M;
        DUM_MET_PARAM(MET_PARAMS+J).BANDS=BANDS;
        PUT LIST('BANDS')SKIP;
        DO K=1 TO BANDS;
        GET FILE(CARDS) LIST(
        DUM_MET_PARAM(MET_PARAMS+J).LOWER_BAND(K),
        DUM_MET_PARAM(MET_PARAMS+J).UPPER_BAND(K));
        PUT LIST(
        DUM_MET_PARAM(MET_PARAMS+J).LOWER_BAND(K),
        DUM_MET_PARAM(MET_PARAMS+J).UPPER_BAND(K));
        END; /* BAND LOOP */
        END; /* PARAM LOOP */
        MET_PARAMS=MET_PARAMS+PARAMS;
        END; /* STATION LOOP */
        FREE      MET_PARAM;
        ALLOCATE MET_PARAM;
        DO J=1 TO MET_PARAMS;
        MET_PARAM(J)=DUM_MET_PARAM(J);
        END;
        FREE DUM_MET_PARAM;
SKIP3:  PUT LIST('OUT STATION REQUESTS')SKIP;
        GET FILE(CARDS) DATA (OUT_STATIONS);
        IF OUT_STATIONS=0 THEN BEGIN;
        PUT LIST('NO OUT STATIONS REQUESTED')SKIP;
        GO TO NEXT;END;
        OUTPUT_HEADER.STATIONS=OUT_STATIONS;
        FREE      OUTPUT_STATION;
        ALLOCATE OUTPUT_STATION;
        OUT_PARAMS=0;
        ALLOCATE DUM_OUT_PARAM;
        ALLOCATE DUM_OUTPUT_PARAM;
        DO I=1 TO OUT_STATIONS;
        GET FILE(CARDS) DATA(STATION,PARAMS);
        PUT DATA(STATION)SKIP;
        IF PARAMS=0 THEN BEGIN;
        PUT LIST('NO PARAMETER REQUESTS')SKIP;
        GO TO NEXT;END;
```

```
        IF OUT_PARAMS+PARAMS>100 THEN BEGIN;
        PUT LIST('NUMBER OF PARAMETERS GREATER THAN 100')SKIP;
        GO TO NEXT;END;
        DO N=1 TO FILE_HEADER.STATIONS;
        IF STATION=FILE_STATION(N).NAME THEN GO TO FIND_N2;
        END;/* N LOOP*/
        PUT LIST('STATION NAME ',STATION,' NOT RECOGNIZED');GO TO NEXT;
FIND_N2:OUTPUT_STATION(I).NAME=STATION;
        OUTPUT_STATION(I).POINTER=OUT_PARAMS;
        OUTPUT_STATION(I).PARAMS=PARAMS;
        DO J=1 TO PARAMS;
        DELAY=0;
        GET FILE(CARDS) DATA(PARAM,DELAY);
        PUT DATA(PARAM,DELAY)SKIP;
        DUM_OUTPUT_PARAM(OUT_PARAMS+J).STATION=STATION;
        DUM_OUTPUT_PARAM(OUT_PARAMS+J).NAME=PARAM;
        DO M=1 TO FILE_STATION (N).PARAMS;
        IF PARAM=FILE_PARAM(FILE_STATION(N).POINTER+M).NAME
        THEN GO TO FIND_M2;
        END;/* M LOOP*/
        PUT LIST('PARAMETER NAME ',PARAM,' NOT RECOGNIZED');GO TO NEXT;
FIND_M2:DUM_OUT_PARAM(OUT_PARAMS+J).POINTER=FILE_STATION(N).POINTER+M;
        DUM_OUT_PARAM(OUT_PARAMS+J).DELAY=DELAY;
        END; /* PARAM LOOP */
        OUT_PARAMS=OUT_PARAMS+PARAMS;
        END; /*STATION LOOP */
        OUTPUT_HEADER.PARAMS=OUT_PARAMS;
        FREE      OUTPUT_PARAM;
        ALLOCATE  OUTPUT_PARAM;
        FREE      OUT_PARAM;
        ALLOCATE  OUT_PARAM;
        FREE      PRINT_STATION;
        ALLOCATE  PRINT_STATION;
        FREE      PRINT_NAME;
        ALLOCATE  PRINT_NAME;
        DO J=1 TO OUT_PARAMS;
        OUTPUT_PARAM(J)=DUM_OUTPUT_PARAM(J);
        OUT_PARAM(J)=DUM_OUT_PARAM(J);
        STATION=OUTPUT_PARAM(J).STATION;
        PARAM  =OUTPUT_PARAM(J).NAME   ;
        UNSPEC(PRINT_STATION(J))=UNSPEC(STATION);
        UNSPEC(PRINT_NAME   (J))=UNSPEC(PARAM);
        END;
        FREE DUM_OUTPUT_PARAM;
        FREE DUM_OUT_PARAM;
        CLOSE FILE(CARDS);
        RETURN;
        END INPUT;
SEARCH: PROC;
DCL     HOUR FIXED BINARY, HM1 FIXED BINARY;
DCL     DATE_KEY BIT(1) INITIAL('0'B), HOUR_KEY BIT(1) INITIAL ('0'B);
DCL     DATE_LIMIT FIXED BINARY, HOUR_LIMIT FIXED BINARY;
DCL     DELAY_HOUR FIXED BINARY;
DCL     YEAR CHAR(2), MONTH CHAR(2), DAY CHAR(2);
        DATE_LIMIT=1;
        NUMBER_OBS=0;
        FREE      OUTPUT_DATA;
        ALLOCATE  OUTPUT_DATA;
        CLOSE FILE(SCRATCH);
        ON ENDFILE(MASTER)  BEGIN;
        PUT LIST('END OF FILE SENSED ON MASTER')SKIP;
```

```
         GO TO WRITE;END;
         READ FILE(MASTER) INTO (TODAY);
         YESTERDAY=TODAY;
DATES:   /* LOOP ON DATES - TEST BETWEEN DATE LIMITS */
         IF TODAY.DATE=LOWER_DATE(DATE_LIMIT) THEN DATE_KEY='1'B;
         IF TODAY.DATE=UPPER_DATE(DATE_LIMIT) THEN BEGIN;
         DATE_KEY='0'B;DATE_LIMIT=DATE_LIMIT+1; GO TO HOURS;
         END;
         IF DATE_KEY='0'B THEN GO TO NEWDATE;
HOURS:   /* LOOP ON HOURS - TEST BETWEEN HOUR LIMITS */
         HOUR_LIMIT=1; HOUR_KEY='0'B;
         DO HOUR=1 TO 24;
         IF HOUR=LOWER_HOUR(HOUR_LIMIT) THEN HOUR_KEY='1'B;
         IF HOUR=UPPER_HOUR(HOUR_LIMIT) THEN BEGIN;
         HOUR_KEY='0'B; HOUR_LIMIT=HOUR_LIMIT+1;GO TO TEST;
         END;
         IF HOUR_KEY='0'B THEN GO TO NEWHOUR;
TEST:    /* TEST PARAMETER BANDS */
         IF MET_PARAMS=0 THEN GO TO STORE;
         DO J=1 TO MET_PARAMS;
         DO K=1 TO MET_PARAM(J).BANDS;
/*       PUT DATA(MET_PARAM(J).LOWER_BAND(K))SKIP;
         PUT DATA(TODAY.PARAM(HOUR,MET_PARAM.POINTER(J)))SKIP;
         PUT DATA(MET_PARAM(J).UPPER_BAND(K))SKIP;          */
         IF TODAY.PARAM(HOUR,MET_PARAM.POINTER(J))>=
         MET_PARAM(J).LOWER_BAND(K)&
            TODAY.PARAM(HOUR,MET_PARAM.POINTER(J))<=
         MET_PARAM(J).UPPER_BAND(K)
         THEN GO TO NEWPAR ;
         END; /* BAND LOOP */
         GO TO NEWHOUR;
NEWPAR:  END; /* PARAM LOOP */
STORE:   /* STORE OUPUT PARAMETERS */
         YEAR =SUBSTR(TODAY.DATE,1,2);
         MONTH=SUBSTR(TODAY.DATE,3,2);
         DAY  =SUBSTR(TODAY.DATE,5,2);
         OUTPUT_DATA.DATE=YEAR||'/'||MONTH||'/'||DAY;
         HM1=HOUR-1;
         PUT STRING (OUTPUT_DATA.HOUR) EDIT (HM1 )(F(2),X(2));
         DO J=1 TO OUT_PARAMS;
         DELAY_HOUR=HOUR-OUT_PARAM(J).DELAY;
         IF DELAY_HOUR<1 THEN BEGIN;
         OUTPUT_DATA.PARAM(J)=
         YESTERDAY.PARAM(24+DELAY_HOUR,OUT_PARAM.POINTER(J));
         GO TO SKIP;END;
         ELSE   BEGIN;
         OUTPUT_DATA.PARAM(J)=
         TODAY.PARAM(DELAY_HOUR,OUT_PARAM.POINTER(J));    END;
SKIP:    IF OUTPUT_DATA.PARAM(J)=1.E20 THEN GO TO NEWHOUR;
         END; /* PARAM LOOP */
         NUMBER_OBS=NUMBER_OBS+1;
/*       PUT DATA(OUTPUT_DATA)SKIP;            */
         WRITE FILE (SCRATCH) FROM (OUTPUT_DATA);
NEWHOUR:IF HOUR_LIMIT>HOUR_LIMITS THEN GO TO NEWDATE;
         END; /* HOUR LOOP */
NEWDATE:IF DATE_LIMIT>DATE_LIMITS THEN GO TO WRITE;
         YESTERDAY=TODAY;
         READ FILE(MASTER) INTO (TODAY);
         GO TO DATES;
WRITE:   /* SEARCH FINISHED - RECORD DATA ON OUTPUT TAPE */
         CLOSE FILE (MASTER);
```

```
        IF NUMBER_OBS=0 THEN BEGIN;
        PUT LIST('NO OBSERVATIONS WERE FOUND WITHIN BANDS')SKIP;
        RETURN;END;
        CLOSE FILE(SCRATCH);
        IF PRINT='NO ' THEN GO TO JUMP1;
        FREE      PRINT_DATE; FREE      PRINT_HOUR;
        ALLOCATE  PRINT_DATE; ALLOCATE PRINT_HOUR;
        FREE      PRINT_PARAM;
        ALLOCATE PRINT_PARAM;
        IF ALLOCATION(PRINT_PARAM)='0'B THEN BEGIN;
        PUT LIST('INSUFFICIENT STORAGE AVAILABLE FOR DATA ARRAY')PAGE;
        GO TO NEXT; END;
JUMP1:  PUT LIST('DATA STORAGE HAS BEEN ALLOCATED')SKIP;
        WRITE FILE(OUTPUT) FROM (OUTPUT_HEADER);
        WRITE FILE(OUTPUT) FROM (OUTPUT_STATION);
        WRITE FILE(OUTPUT) FROM (OUTPUT_PARAM);
        ALLOCATE WRITE_DATA;
        DO N=1 TO NUMBER_OBS;
        READ FILE (SCRATCH) INTO (OUTPUT_DATA);
        IF PRINT='NO ' THEN GO TO JUMP2;
        UNSPEC(PRINT_DATE(N))=UNSPEC(OUTPUT_DATA.DATE);
        UNSPEC(PRINT_HOUR(N))=UNSPEC(OUTPUT_DATA.HOUR);
JUMP2:  DO M=1 TO OUT_PARAMS;
        IF PRINT='NO ' THEN GO TO JUMP3;
        PRINT_PARAM(M,N)=OUTPUT_DATA.PARAM(M);
JUMP3:  WRITE_DATA(M)=OUTPUT_DATA.PARAM(M);
        END;
        WRITE FILE(OUTPUT) FROM (WRITE_DATA);
        END;
        IF PRINT='NO ' THEN GO TO JUMP4;
        CLOSE FILE(SYSPRINT);
        CALL TWODPT(PRINT_PARAM(1,1),NUMBER_OBS,OUT_PARAMS,
        PRINT_TITLE(1),PRINT_DATE(1),PRINT_HOUR(1),PRINT_STATION(1),
        PRINT_NAME(1));
JUMP4:  RETURN;
        END SEARCH;
STOP:   END APDATA;
```

APICS USER MANUAL

APPENDIX F

PROGRAMS TO ACCESS DATA FROM A SEARCH
SOURCE LISTINGS

1. FORTRN Retrieval of Search Output to Pass to a FORT Routine

```
FORTRN: PROC OPTIONS(MAIN);
DCL     1 OUTPUT_HEADER,
            2 LABEL CHAR (80),
            2 STATIONS FIXED BINARY (31,0),
            2 NUMBER_PARAMS FIXED BINARY (31,0),
            2 NUMBER_OBS FIXED BINARY (31,0);
DCL     1 OUTPUT_STATION (OUTPUT_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL    1 OUTPUT_PARAM (NUMBER_PARAMS) CONTROLLED,
            2 STATION CHAR(8),
            2 NAME CHAR(8);
DCL      OUTPUT_DATA (NUMBER_PARAMS) CONTROLLED ;
DCL      REGRES_DATA (NUMBER_PARAMS,NUMBER_OBS) CONTROLLED    ;
DCL      PRINT_STATION(NUMBER_PARAMS) CONTROLLED
         FLOAT DECIMAL (16);
DCL      PRINT_PARAM ( NUMBER_PARAMS) CONTROLLED
         FLOAT DECIMAL (16);
DCL      CLABEL CHAR(80);
DCL          TYPE CHAR (3), STATION CHAR (8),PARAM CHAR (8);
NEXT:    /* BEGIN NEW CASE */
         FREE OUTPUT_STATION, OUTPUT_PARAM, OUTPUT_DATA,REGRES_DATA;
         CLOSE FILE(OUTPUT);
         ON ENDFILE(OUTPUT) BEGIN;
         PUT LIST('ARRAY LABEL NOT FOUND')PAGE;
         GO TO NEXT;END    ;
         GET EDIT(CLABEL)(A(80));
         IF INDEX(CLABEL,'STOP')¬=0 THEN GO TO STOP;
         PUT LIST('TAPE SEARCH BEGUN FOR LABEL: ',CLABEL)PAGE;
READ:    READ FILE(OUTPUT) INTO (OUTPUT_HEADER);
         PUT DATA (OUTPUT_HEADER)SKIP;
         IF CLABEL¬= LABEL THEN BEGIN;
      READ FILE(OUTPUT) IGNORE  (NUMBER_OBS+2);
         GO TO READ;END;
         PUT LIST('ARRAY LABEL FOUND')SKIP;
         ALLOCATE OUTPUT_STATION;
         ALLOCATE OUTPUT_PARAM;
         ALLOCATE OUTPUT_DATA;
         ALLOCATE REGRES_DATA;
         READ FILE(OUTPUT) INTO (OUTPUT_STATION);
         PUT DATA (OUTPUT_STATION)SKIP;
         READ FILE(OUTPUT) INTO (OUTPUT_PARAM);
         PUT DATA (OUTPUT_PARAM  )SKIP;
         DO N=1 TO NUMBER_OBS;
         READ FILE(OUTPUT) INTO (OUTPUT_DATA);
         DO M=1 TO NUMBER_PARAMS;
         REGRES_DATA(M,N)=OUTPUT_DATA(M);
         END; END;
         FREE PRINT_STATION; ALLOCATE PRINT_STATION;
         FREE PRINT_PARAM  ; ALLOCATE PRINT_PARAM  ;
         DO N=1 TO NUMBER_PARAMS;
         UNSPEC(PRINT_STATION(N))=
         UNSPEC(OUTPUT_PARAM(N        ).STATION);
         UNSPEC(PRINT_PARAM  (N))=
         UNSPEC(OUTPUT_PARAM(N        ).NAME);
         END;
         CLOSE FILE(SYSPRINT);
         CALL FORT  (NUMBER_PARAMS,NUMBER_OBS,REGRES_DATA(1,1),
                                  PRINT_STATION(1),PRINT_PARAM(1));
         GO TO NEXT;
STOP:    END FORTRN;
```

2. FORTRNS Retrieval of Search Output Sequentially and Pass n at a Time

```
FORTRN: PROC OPTIONS(MAIN);
DCL RETCODE EXTERNAL FIXED BINARY(31,0);
DCL FORT EXTERNAL ENTRY;
DCL   1 OUTPUT_HEADER,
        2 LABEL CHAR(80),
        2 STATIONS FIXED BINARY(31,0),
        2 NUMBER_PARAMS FIXED BINARY(31,0),
        2 NUMBER_OBS FIXED BINARY(31,0);
DCL   1 OUTPUT_STATION(50),
        2 NAME CHAR(8),
        2 POINTER FIXED BINARY,
        2 PARAMS FIXED BINARY;
DCL   1 OUTPUT_PARAM(300),
        2 STATION CHAR(8),
        2 NAME CHAR(8);
DCL OUTPUT_DATA(300);
DCL REGRES_DATA(NUMBER_PARAMS,MAX_THIS_TIME) CONTROLLED;
DCL PRINT_STATION(300) FLOAT DECIMAL(16),
    PRINT_PARAM(300) FLOAT DECIMAL(16);
DCL CLABEL CHAR(80);
DCL (TYPE,STATION,PARAM) CHAR(8);
DCL (I,M,N,NUMBER_NOW,MAX_THIS_TIME,MAX_OBS_AT_ONE_TIME)
                                FIXED BINARY(31,0);
        GET EDIT(MAX_OBS_AT_ONE_TIME,TYPE)(F(5),X(72),A(3));
        ON RECORD(OUTPUT);
NEW_CASE:
        FREE REGRES_DATA;
        ON ENDFILE(OUTPUT)
        BEGIN;
          PUT LIST('ARRAY LABEL NOT FOUND') PAGE;
/*        CLOSE FILE(OUTPUT);                          */
          GO TO NEW_CASE;
        END;
        GET EDIT(CLABEL) (A(80));
        IF INDEX(CLABEL,'STOP')¬=0 THEN GO TO STOP;
        PUT LIST('TAPE SEARCH BEGUN FOR LABEL: ',CLABEL) PAGE;
READ_LABEL:
        READ FILE(OUTPUT) INTO (OUTPUT_HEADER);
        PUT DATA (OUTPUT_HEADER) SKIP;
        IF CLABEL¬= OUTPUT_HEADER.LABEL THEN
        BEGIN;
          READ FILE(OUTPUT) IGNORE (NUMBER_OBS+2);
          GO TO READ_LABEL;
        END;
        PUT LIST ('ARRAY LABEL FOUND') SKIP;
        MAX_THIS_TIME=MIN(MAX_OBS_AT_ONE_TIME,NUMBER_OBS);
        ALLOCATE REGRES_DATA;
        READ FILE(OUTPUT) INTO (OUTPUT_STATION);
        PUT DATA((OUTPUT_STATION(I) DO I=1 TO OUTPUT_HEADER.STATIONS))
            SKIP;
        READ FILE(OUTPUT) INTO (OUTPUT_PARAM);
        PUT DATA((OUTPUT_PARAM(I) DO I=1 TO
             OUTPUT_HEADER.NUMBER_PARAMS)) SKIP;
          DO N=1 TO NUMBER_PARAMS;
            UNSPEC(PRINT_STATION(N))=UNSPEC(OUTPUT_PARAM(N).STATION);
            UNSPEC(PRINT_PARAM(N))=UNSPEC(OUTPUT_PARAM(N).NAME);
          END;
        NUMBER_NOW=0;
        DO I_AT_OBS=1 TO NUMBER_OBS;
          NUMBER_NOW=NUMBER_NOW+1;
          READ FILE(OUTPUT) INTO (OUTPUT_DATA);
          DO M=1 TO NUMBER_PARAMS;
            REGRES_DATA(M,NUMBER_NOW)=OUTPUT_DATA(M);
          END;
          IF I_AT_OBS¬=NUMBER_OBS&NUMBER_NOW<MAX_THIS_TIME THEN
                GO TO DONT_CALL;
/*        CLOSE FILE(SYSPRINT); */
          I=0;
          CALL FORTRAN(FORT,I,NUMBER_PARAMS,NUMBER_NOW,
             REGRES_DATA(1,1),PRINT_STATION(1),PRINT_PARAM(1));
          NUMBER_NOW=0;
DONT_CALL:
        END;
        NUMBER_NOW=0;
        I=0;
        CALL FORTRAN(FORT,I,NUMBER_PARAMS,NUMBER_NOW,REGRES_DATA(1,1),
             PRINT_STATION(1),PRINT_PARAM(1));
        GO TO NEW_CASE;
STOP:
        NUMBER_NOW=-1;
        I=0;
```

```
          CALL FORTRAN(FORT,I,NUMBER_PARAMS,NUMBER_NOW,REGRES_DATA(1,1),
               PRINT_STATION(1),PRINT_PARAM(1));
          IF RETCODE=0 THEN CALL FIOCLSE;
 END FORTRN;
FORTRAN  START
*
* CALLING SEQUENCE  CALL FORTRAN(FTNRTN,FLAG,A,B,C,...);
*                        FLAG < 0  IF NO PARAMETERS
*                        FLAG = 0  IF NO STRING DATA IN PARAMETER LIST
*                        FLAG = N  IF FIRST N PARAMETERS ARE STRINGS
*
          EXTRN IHEERRA
          ENTRY FIOCLSE
          BC    15,12(15)
          DC    X'07'
          DC    CL7'FORTRAN'
          STM   14,12,12(13)                SAVE REGISTERS
          BALR  12,0                        ESTABLISH BASE REGISTER
          USING *,12
BASE      LR    8,13                        LOAD PL/I SAVE AREA POINTER
          LA    13,SAVE                     FORTRAN'S SAVE AREA POINTER
          ST    13,8(8)                     STORE POINTER IN PL/I SAVE AREA
          ST    8,4(13)                     STORE PL/I SAVE AREA POINTER
*                                             IN FORTRAN'S SAVE AREA
          ST    15,SAVE+16                  STORE ADDRESS OF FORTRAN IN
*                                             FIFTH WORD OF ITS SAVE AREA
*                                             FOR TRACEBACK
*
* CREATE FORTRAN ENVIRONMENT THROUGH CALL TO IBFINT
* IBFINT SAVES REGISTERS IN ITS OWN SAVE AREA BUT EXPECTS REGISTER 13
* TO POINT TO SAVE AREA OF SUPERVISOR -- FORTRAN.
*
          L     15,VIBCOM                   LOAD ADDRESS OF IBCOM
          BAL   14,64(15)                   CALL IBFINT TO INITIALIZE FORT.
          L     13,SAVE+4                   LOAD PL/I SAVE AREA POINTER
          L     1,24(13)                    LOAD PAREMETER LIST ADDRESS
          L     2,0(1)
          L     15,0(2)                     LOAD ADDRESS OF FORTRAN SUBRTN.
          LA    13,SAVE                     LOAD SAVE AREA POINTER
* TEST FLAG FOR ANALYSIS OF PARAMETER LIST
          L     3,4(1)
          L     3,0(3)                      LOAD FLAG
          LTR   3,3                         TEST FLAG FOR PARAMETERS
          BC    NOPARM,CALL                 BRANCH IF FLAG<0  NO PARAMETERS
          LA    1,8(1)                      SET PARAMETER LIST POINTER
          BC    NOSTRG,CALL                 BRANCH IF FLAG=0  NO STRINGS
* OBTAIN STRING ADDRESS FROM 1ST WORD OF STRING DOPE VECTOR -- SDV
          SR    6,6                         ZERO REGISTER
LOOP      L     4,0(6,1)                    LOAD ADDRESS OF SDV
          L     2,0(4)                      LOAD 1ST WORD OF SDV
          ST    2,0(6,1)                    STORE ADDRESS IN PARAMETER LIST
          AH    6,FOUR                      INCREASE REGISTER FOR NEXT PARM
          BCT   3,LOOP                      BRANCH IF N NOT ZERO
*
CALL      BALR  14,15                       CALL FORTRAN SUBROUTINE
*
          L     5,RCADDR                    LOAD ADDRESS OF RETURN CODE
          MVC   0(4,5),RCNORM               PUT ZERO IN RETURN CODE
* TEST RETURN CODE IN REGISTER 15 FOR TERMINAL ERROR, NORMAL RETURN,
* OR STOP.  ALL OTHERS ASSUMED NORMAL.
          LTR   15,15                       TEST RETURN CODE
          BC    ZERO,RETCODEO               BRANCH IF CODE IS 0
          C     15,TERMERR                  TEST FOR TERMINAL ERROR
          BC    NOERR,CONTINUE              BRANCH IF NOT A TERMINAL ERROR
          MVC   0(4,5),RCERR                PUT 1 IN RETURN CODE : ERROR
          BC    ALWAYS,CONTINUE
RETCODEO  CLI   12(13),X'FF'                TEST FOR NORMAL RETURN
          BC    ZERO,CONTINUE               BRANCH IF NORMAL RETURN
          MVC   0(4,5),RCSTOP               PUT -1 IN RETURN CODE : STOP
* RESTORE PL/I INTERRUPT SYSTEM
CONTINUE SPIE IHEERRA,((1,13),15)
*
          L     13,SAVE+4                   LOAD PL/I SAVE AREA POINTER
          LM    14,12,12(13)                RESTORE REGISTERS
          BR    14                          RETURN TO PL/I
*
* CLOSE FORTRAN I/O DATA SETS THROUGH CALL TO IBEXIT
* IBEXIT SAVES REGISTERS IN ITS OWN SAVE AREA BUT RETURNS VIA FOURTH
* WORD IN SAVE AREA OF SUPERVISOR -- FORTRAN.  REGISTER 14 IN CALL IS
* USED FOR A RETURN CODE.
*
```

```
         CNOP   0,4
FIOCLSE  BC     15,12(15)
         DC     X'07'
         DC     CL7'FIOCLSE'
         STM    14,12,12(13)                 SAVE PL/I REGISTERS
         L      12,POINTER-FIOCLSE(15)       ESTABLISH BASE REGISTER
         ST     13,REG13                     STORE PL/I SAVE AREA POINTER
         MVC    REG14(4),SAVE+12             REPLACE FOURTH WORD OF SAVE
         MVC    SAVE+12(4),ARETURN             AREA BY THE RETURN ADDRESS
         L      15,VIBCOM                    LOAD ADDRESS OF IBCOM
         BAL    14,68(15)                    CALL IBEXIT TO CLOSE I/O D.S.
         DC     AL2(0)                       AL2(RETURN CODE)
         CNOP   0,4
* RESTORE PL/I INTERRUPT SYSTEM
RETURN   SPIE   IHEERRA,((1,13),15)
*
         MVC    SAVE+12(4),REG14             RESTORE 4TH WORD OF SAVE AREA
         L      13,REG13                     LOAD PI/I SAVE AREA POINTER
         LM     14,12,12(13)                 RESTORE PL/I REGISTERS
         BR     14                           RETURN TO PL/I
*
* STORAGE SECTION FOR FORTRAN
SAVE     DC     18F'0'
VIBCOM   DC     V(IBCOM#)
POINTER  DC     A(BASE)
ARETURN  DC     A(RETURN)
RCADDR   DC     A(RETCODE)
TERMERR  DC     XL4'00000010'
REG13    DC     F'0'
REG14    DC     F'0'
RCERR    DC     F'1'
RCNORM   DC     F'0'
RCSTOP   DC     F'-1'
FOUR     DC     H'4'
NOPARM   EQU    4
NOERR    EQU    7
ZERO     EQU    8
NOSTRG   EQU    8
ALWAYS   EQU    15
*
* STORAGE AREA FOR RETURN CODE TO PL/I
* DECLARE RETCODE FIXED BINARY (31,0) EXTERNAL;
RETCODE  CSECT
CODE     DC     F'0'                         INITIALIZE CODE TO ZERO
*
         END
```

3. FORTRNK Retrieval of Search Output and Pass n at a Time to a FORT Routine

```
FORTRN: PROC OPTIONS(MAIN);
DCL RETCODE EXTERNAL FIXED BINARY(31,0);
DCL FORT EXTERNAL ENTRY;
DCL   1 OUTPUT_HEADER,
        2 LABEL CHAR(80),
        2 STATIONS FIXED BINARY(31,0),
        2 NUMBER_PARAMS FIXED BINARY(31,0),
        2 NUMBER_OBS FIXED BINARY(31,0);
DCL   1 OUTPUT_STATION(50),
        2 NAME CHAR(8),
        2 POINTER FIXED BINARY,
        2 PARAMS FIXED BINARY;
DCL   1 OUTPUT_PARAM(300),
        2 STATION CHAR(8),
        2 NAME CHAR(8);
DCL OUTPUT_DATA(300);
DCL REGRES_DATA(NUMBER_PARAMS,MAX_THIS_TIME) CONTROLLED;
DCL PRINT_STATION(300) FLOAT DECIMAL(16),
    PRINT_PARAM(300) FLOAT DECIMAL(16);
DCL CLABEL CHAR(80);
DCL (TYPE,STATION,PARAM) CHAR(8);
DCL (I,M,N,NUMBER_NOW,MAX_THIS_TIME,MAX_OBS_AT_ONE_TIME)
                                  FIXED BINARY(31,0);
        GET EDIT(MAX_OBS_AT_ONE_TIME,TYPE)(F(5),X(72),A(3));
        ON RECORD(OUTPUT);
NEW_CASE:
        FREE REGRES_DATA;
        ON ENDFILE(OUTPUT)
        BEGIN;
          PUT LIST('ARRAY LABEL NOT FOUND') PAGE;
          CLOSE FILE(OUTPUT);
          GO TO NEW_CASE;
        END;
        GET EDIT(CLABEL) (A(80));
        IF INDEX(CLABEL,'STOP')¬=0 THEN GO TO STOP;
        PUT LIST('TAPE SEARCH BEGUN FOR LABEL: ',CLABEL) PAGE;
READ_LABEL:
        READ FILE(OUTPUT) INTO (OUTPUT_HEADER);
        PUT DATA (OUTPUT_HEADER) SKIP;
        IF CLABEL¬= OUTPUT_HEADER.LABEL THEN
        BEGIN;
          READ FILE(OUTPUT) IGNORE (NUMBER_OBS+2);
          GO TO READ_LABEL;
        END;
        PUT LIST ('ARRAY LABEL FOUND') SKIP;
        MAX_THIS_TIME=MIN(MAX_OBS_AT_ONE_TIME,NUMBER_OBS);
        ALLOCATE REGRES_DATA;
        READ FILE(OUTPUT) INTO (OUTPUT_STATION);
        PUT DATA((OUTPUT_STATION(I) DO I=1 TO OUTPUT_HEADER.STATIONS))
             SKIP;
        READ FILE(OUTPUT) INTO (OUTPUT_PARAM);
        PUT DATA((OUTPUT_PARAM(I) DO I=1 TO
             OUTPUT_HEADER.NUMBER_PARAMS)) SKIP;
          DO N=1 TO NUMBER_PARAMS;
            UNSPEC(PRINT_STATION(N))=UNSPEC(OUTPUT_PARAM(N).STATION);
            UNSPEC(PRINT_PARAM(N))=UNSPEC(OUTPUT_PARAM(N).NAME);
          END;
        NUMBER_NOW=0;
        DO I_AT_OBS=1 TO NUMBER_OBS;
          NUMBER_NOW=NUMBER_NOW+1;
          READ FILE(OUTPUT) INTO (OUTPUT_DATA);
          DO M=1 TO NUMBER_PARAMS;
            REGRES_DATA(M,NUMBER_NOW)=OUTPUT_DATA(M);
          END;
          IF I_AT_OBS¬=NUMBER_OBS&NUMBER_NOW<MAX_THIS_TIME THEN
                 GO TO DONT_CALL;
/*        CLOSE FILE(SYSPRINT); */
          I=0;
          CALL FORTRAN(FORT,I,NUMBER_PARAMS,NUMBER_NOW,
             REGRES_DATA(1,1),PRINT_STATION(1),PRINT_PARAM(1));
          NUMBER_NOW=0;
DONT_CALL:
        END;
        NUMBER_NOW=0;
        I=0;
        CALL FORTRAN(FORT,I,NUMBER_PARAMS,NUMBER_NOW,REGRES_DATA(1,1),
             PRINT_STATION(1),PRINT_PARAM(1));
        GO TO NEW_CASE;
STOP:
        NUMBER_NOW=-1;
        I=0;
```

```
        CALL FORTRAN(FORT,I,NUMBER_PARAMS,NUMBER_NOW,REGRES_DATA(1,1),
             PRINT_STATION(1),PRINT_PARAM(1));
        IF RETCODE=0 THEN CALL FIOCLSE;
 END FORTRN;
FORTRAN  START
*
* CALLING SEQUENCE  CALL FORTRAN(FTNRTN,FLAG,A,B,C,...);
*                        FLAG < 0  IF NO PARAMETERS
*                        FLAG = 0  IF NO STRING DATA IN PARAMETER LIST
*                        FLAG = N  IF FIRST N PARAMETERS ARE STRINGS
*
         EXTRN IHEERRA
         ENTRY FIOCLSE
         BC    15,12(15)
         DC    X'07'
         DC    CL7'FORTRAN'
         STM   14,12,12(13)              SAVE REGISTERS
         BALR  12,0                      ESTABLISH BASE REGISTER
         USING *,12
BASE     LR    8,13                      LOAD PL/I SAVE AREA POINTER
         LA    13,SAVE                   FORTRAN'S SAVE AREA POINTER
         ST    13,8(8)                   STORE POINTER IN PL/I SAVE AREA
         ST    8,4(13)                   STORE PL/I SAVE AREA POINTER
*                                          IN FORTRAN'S SAVE AREA
         ST    15,SAVE+16                STORE ADDRESS OF FORTRAN IN
*                                          FIFTH WORD OF ITS SAVE AREA
*                                          FOR TRACEBACK
*
* CREATE FORTRAN ENVIRONMENT THROUGH CALL TO IBFINT
* IBFINT SAVES REGISTERS IN ITS OWN SAVE AREA BUT EXPECTS REGISTER 13
* TO POINT TO SAVE AREA OF SUPERVISOR -- FORTRAN.
*
         L     15,VIBCOM                 LOAD ADDRESS OF IBCOM
         BAL   14,64(15)                 CALL IBFINT TO INITIALIZE FORT.
         L     13,SAVE+4                 LOAD PL/I SAVE AREA POINTER
         L     1,24(13)                  LOAD PAREMETER LIST ADDRESS
         L     2,0(1)
         L     15,0(2)                   LOAD ADDRESS OF FORTRAN SUBRTN.
         LA    13,SAVE                   LOAD SAVE AREA POINTER
* TEST FLAG FOR ANALYSIS OF PARAMETER LIST
         L     3,4(1)
         L     3,0(3)                    LOAD FLAG
         LTR   3,3                       TEST FLAG FOR PARAMETERS
         BC    NOPARM,CALL               BRANCH IF FLAG<0  NO PARAMETERS
         LA    1,8(1)                    SET PARAMETER LIST POINTER
         BC    NOSTRG,CALL               BRANCH IF FLAG=0  NO STRINGS
* OBTAIN STRING ADDRESS FROM 1ST WORD OF STRING DOPE VECTOR -- SDV
         SR    6,6                       ZERO REGISTER
LOOP     L     4,0(6,1)                  LOAD ADDRESS OF SDV
         L     2,0(4)                    LOAD 1ST WORD OF SDV
         ST    2,0(6,1)                  STORE ADDRESS IN PARAMETER LIST
         AH    6,FOUR                    INCREASE REGISTER FOR NEXT PARM
         BCT   3,LOOP                    BRANCH IF N NOT ZERO
*
CALL     BALR  14,15                     CALL FORTRAN SUBROUTINE
*
         L     5,RCADDR                  LOAD ADDRESS OF RETURN CODE
         MVC   0(4,5),RCNORM             PUT ZERO IN RETURN CODE
* TEST RETURN CODE IN REGISTER 15 FOR TERMINAL ERROR, NORMAL RETURN,
* OR STOP.  ALL OTHERS ASSUMED NORMAL.
         LTR   15,15                     TEST RETURN CODE
         BC    ZERO,RETCODE0             BRANCH IF CODE IS 0
         C     15,TERMERR                TEST FOR TERMINAL ERROR
         BC    NOERR,CONTINUE            BRANCH IF NOT A TERMINAL ERROR
         MVC   0(4,5),RCERR              PUT 1 IN RETURN CODE : ERROR
         BC    ALWAYS,CONTINUE
RETCODE0 CLI   12(13),X'FF'              TEST FOR NORMAL RETURN
         BC    ZERO,CONTINUE             BRANCH IF NORMAL RETURN
         MVC   0(4,5),RCSTOP             PUT -1 IN RETURN CODE : STOP
* RESTORE PL/I INTERRUPT SYSTEM
CONTINUE SPIE  IHEERRA,((1,13),15)
*
         L     13,SAVE+4                 LOAD PL/I SAVE AREA POINTER
         LM    14,12,12(13)              RESTORE REGISTERS
         BR    14                        RETURN TO PL/I
*
* CLOSE FORTRAN I/O DATA SETS THROUGH CALL TO IBEXIT
* IBEXIT SAVES REGISTERS IN ITS OWN SAVE AREA BUT RETURNS VIA FOURTH
* WORD IN SAVE AREA OF SUPERVISOR -- FORTRAN.  REGISTER 14 IN CALL IS
* USED FOR A RETURN CODE.
*
```

```
         CNOP   0,4
FIOCLSE  BC     15,12(15)
         DC     X'07'
         DC     CL7'FIOCLSE'
         STM    14,12,12(13)              SAVE PL/I REGISTERS
         L      12,POINTER-FIOCLSE(15)    ESTABLISH BASE REGISTER
         ST     13,REG13                  STORE PL/I SAVE AREA POINTER
         MVC    REG14(4),SAVE+12          REPLACE FOURTH WORD OF SAVE
         MVC    SAVE+12(4),ARETURN          AREA BY THE RETURN ADDRESS
         L      15,VIBCOM                 LOAD ADDRESS OF IBCOM
         BAL    14,68(15)                 CALL IBEXIT TO CLOSE I/O D.S.
         DC     AL2(0)                    AL2(RETURN CODE)
         CNOP   0,4
* RESTORE PL/I  INTERRUPT SYSTEM
RETURN   SPIE   IHEERRA,((1,13),15)
*
         MVC    SAVE+12(4),REG14          RESTORE 4TH WORD OF SAVE AREA
         L      13,REG13                  LOAD PI/I SAVE AREA POINTER
         LM     14,12,12(13)              RESTORE PL/I REGISTERS
         BR     14                        RETURN TO PL/I
*
* STORAGE SECTION FOR FORTRAN
SAVE     DC     18F'0'
VIBCOM   DC     V(IBCOM#)
POINTER  DC     A(BASE)
ARETURN  DC     A(RETURN)
RCADDR   DC     A(RETCODE)
TERMERR  DC     XL4'00000010'
REG13    DC     F'0'
REG14    DC     F'0'
RCERR    DC     F'1'
RCNORM   DC     F'0'
RCSTOP   DC     F'-1'
FOUR     DC     H'4'
NOPARM   EQU    4
NOERR    EQU    7
ZERO     EQU    8
NOSTRG   EQU    8
ALWAYS   EQU    15
*
* STORAGE AREA FOR RETURN CODE TO PL/I
* DECLARE RETCODE FIXED BINARY (31,0) EXTERNAL;
RETCODE  CSECT
CODE     DC     F'0'                      INITIALIZE CODE TO ZERO
*
         END
```

APICS USER MANUAL

APPENDIX G

DATA DISPLAY PROGRAMS
SOURCE LISTINGS

1. PRINT Data Display Program

```
      SUBROUTINE TWODPT(ARAY,ICT,JCT,TITLE,DATE,COLT1,COLT2)
C                                 TWODPT PRINTS A TWO-DIMENSIONAL ARRAY.  IT
C                                PRINTS A TITLE LINE AND TWO LINES OF COLUMN
C                                TITLES, WHICH ARE FOLLOWED BY 50 LINES UP
C                                TO TEN COLUMNS WIDE PLUS THE INDEX COLUMN,
C                                WHICH CONTAINS DATE AND HOUR.
C                                ARGUMENTS ARE -
C
C                                      ARAY - THE TWO-DIMENSIONAL ARRAY
C                                             TO BE PRINTED.
C                                       ICT - NUMBER OF ROWS IN ARRAY.
C                                       JCT - NUMBER OF COLUMNS IN ARRAY.
C                                     TITLE - PAGE TITLE (10A8).
C                                     COLT1 - ARRAY DIMENSIONED BY JCT, CON-
C                                             TAINING FIRST LINE OF TITLE
C                                             FOR EACH COLUMN (A8).
C                                     COLT2 - ARRAY DIMENSIONED BY JCT, CON-
C                                             TAINING SECOND LINE OF TITLE
C                                             FOR EACH COLUMN (A8).
C
      REAL*8 TITLE(10),DATE(1) ,COLT1(1),COLT2(1)
      REAL*4 ARAY(ICT,1)
      II=1
      IJ=1
      IMAX=0
      JMAX=0
      MAX=0
      IFLAG=0
   20 IEND=JCT
   40 MIN=MAX+1
      MAX=MIN0(MAX+10,IEND)
      IF(IFLAG.NE.0) GO TO 47
   45 NROW=0
   46 CONTINUE
      PRINT 210,(TITLE(I),I=1,10)
      PRINT 220,DATE
   47 PRINT 230,(COLT1(JC),JC=MIN,MAX)
      PRINT 235,(COLT2(JC),JC=MIN,MAX)
      PRINT 235
      IMAX=MIN0(II+52,ICT)
      IDIF=IMAX-II+5
      NROW=NROW+IDIF
      JMAX=MIN0(IJ+9,JCT)
      DO 65 I=II,IMAX
      IM1=I-1
      PRINT 240     ,            IM1,(ARAY(I,J),J=IJ,JMAX)
   65 CONTINUE
      IF(IMAX.GE.ICT) GO TO 70
      II=IMAX+1
      GO TO 46
   70 IF(JMAX.GE.JCT) GO TO 130
   80 II=1
      IJ=JMAX+1
      IF((NROW+IDIF-57).GT.0) GO TO 84
   82 IFLAG=1
      GO TO 40
   84 IFLAG=0
      GO TO 40
  130 CONTINUE
C     DO 140 I=1,25
C     PRINT 150
  150 FORMAT(' ')
C 140 CONTINUE
      RETURN
  210 FORMAT(1H1,9X,10A8)
  220 FORMAT(10X'DATE:'A8)
  230 FORMAT(1H07X4HHOUR,10(4X,A8))
  235 FORMAT(12X,10(4X,A8))
  240 FORMAT(10XI2,10F12.3)
      END
 APPRINT: PROC OPTIONS (MAIN);
     DCL DATE_LIMITS FIXED BINARY INITIAL(0),
         LOWER_DATE(DATE_LIMITS) CHAR(8) CONTROLLED,
         UPPER_DATE(DATE_LIMITS) CHAR(8) CONTROLLED;
 DCL     1 FILE_HEADER,
             2 STATIONS FIXED BINARY,
             2 PARAMS FIXED BINARY,
             2 STARTING_DATE CHAR(6),
             2 ENDING_DATE CHAR(6);
 DCL     1 FILE_STATION (FILE_HEADER.STATIONS) CONTROLLED
```

```
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 FILE_PARAM (FILE_HEADER.PARAMS) CONTROLLED,
            2 NAME CHAR(8);
DCL     1 TODAY      CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (FILE_HEADER.PARAMS);
DCL     1 YESTERDAY CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (FILE_HEADER.PARAMS);
    DCL OUT_STATIONS FIXED BINARY (31,0),
        OUT_PARAMS   FIXED BINARY (31,0);
DCL     1 OUT_PARAM (OUT_PARAMS) CONTROLLED,
            2 POINTER FIXED BINARY,
            2 DELAY FIXED DECIMAL (2);
DCL     1 DUM_OUT_PARAM (100)  CONTROLLED,
            2 POINTER FIXED BINARY,
            2 DELAY FIXED DECIMAL (2);
DCL     1 OUTPUT_HEADER,
            2 LABEL (10) CHAR(8),
            2 STATIONS FIXED BINARY (31,0),
            2 PARAMS FIXED BINARY (31,0),
            2 NUMBER_OBS FIXED BINARY (31,0);
DCL     1 OUTPUT_STATION (OUTPUT_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 OUTPUT_PARAM (OUTPUT_HEADER.PARAMS) CONTROLLED,
            2 STATION CHAR(8),
            2 NAME CHAR(8);
DCL     1 DUM_OUTPUT_PARAM(100)  CONTROLLED,
            2 STATION CHAR(8),
            2 NAME CHAR(8);
DCL       PRINT_TITLE (10) FLOAT DECIMAL (16),
          PRINT_DATE(1) FLOAT DECIMAL (16),
          PRINT_STATION (OUT_PARAMS) CONTROLLED FLOAT DECIMAL (16),
          PRINT_NAME    (OUT_PARAMS) CONTROLLED FLOAT DECIMAL (16),
          PRINT_PARAM (JDUM,IDUM              ) CONTROLLED;
DCL       IDUM FIXED BINARY (31,0), JDUM FIXED BINARY (31,0);
DCL       CARD CHAR(80);
          PUT LIST('MASTER FILE HEADER INFORMATION')PAGE;
          READ FILE(MASTER) INTO (FILE_HEADER);
          PUT DATA(FILE_HEADER)SKIP;
          ALLOCATE FILE_STATION;
          READ FILE(MASTER) INTO (FILE_STATION);
          PUT DATA(FILE_STATION)SKIP;
          ALLOCATE FILE_PARAM;
          READ FILE(MASTER) INTO (FILE_PARAM);
          PUT DATA (FILE_PARAM)SKIP;
          ALLOCATE TODAY; ALLOCATE YESTERDAY;
NEXT:     /* BEGIN NEW CASE */
          CLOSE FILE(MASTER);
          ON ENDFILE(MASTER)  BEGIN;
          PUT LIST('END OF FILE SENSED ON MASTER')SKIP;
          GO TO NEXT ;END;
          READ FILE (MASTER) IGNORE(3);
          CLOSE FILE(CARDS);
GETCARD:  GET EDIT(CARD)(A(80));
          IF INDEX(CARD,'END ')¬=0 THEN GO TO END;
          IF INDEX(CARD,'STOP')¬=0 THEN GO TO STOP;
          WRITE FILE(CARDS) FROM (CARD);
          GO TO GETCARD;
END:      CLOSE FILE(CARDS);
          CALL INPUT;
          CALL SEARCH;
          GO TO NEXT;
INPUT:    PROC;
DCL       PARAMS FIXED BINARY,BANDS FIXED BINARY,
          DELAY FIXED DECIMAL (2);
     DCL STATION CHAR(8),PARAM CHAR(8);
     DCL L1 CHAR(2),L2 CHAR(2),L3 CHAR(2),
          U1 CHAR(2),U2 CHAR(2),U3 CHAR(2);
          ON NAME(CARDS) BEGIN;
          PUT LIST('THE FOLLOWING DATA IS OUT OF ORDER: ',DATAFIELD)SKIP;
          GO TO NEXT;END;
          ON ENDFILE(CARDS)  BEGIN;
          PUT LIST('UNEXPECTED EOF ON CARD INPUT')SKIP;
          GO TO NEXT;END;
```

```
        PUT LIST('INPUT DATA')PAGE;
        GET FILE(CARDS) EDIT(LABEL)(10A(8));
        DO I=1 TO 10;
        UNSPEC(PRINT_TITLE(I))=UNSPEC(LABEL(I));
        END;
        PUT EDIT(LABEL)(10A(8))SKIP;
        GET FILE(CARDS) DATA(DATE_LIMITS);
        FREE     LOWER_DATE; FREE     UPPER_DATE;
        IF DATE_LIMITS=0 THEN BEGIN;
        DATE_LIMITS=1;
        ALLOCATE LOWER_DATE; ALLOCATE UPPER_DATE;
        LOWER_DATE(1)=STARTING_DATE;
        UPPER_DATE(1)=ENDING_DATE;
        PUT DATA(LOWER_DATE(1),UPPER_DATE(1))SKIP;
        GO TO SKIP1;END;
        ELSE BEGIN;
        ALLOCATE LOWER_DATE; ALLOCATE UPPER_DATE;
        GET FILE(CARDS) LIST
                ((LOWER_DATE(I),UPPER_DATE(I) DO I=1 TO DATE_LIMITS));
        DO I=1 TO DATE_LIMITS;
        PUT DATA(LOWER_DATE(I),UPPER_DATE(I))SKIP;
        L1=SUBSTR(LOWER_DATE(I),1,2)  ;
        L2=SUBSTR(LOWER_DATE(I),4,2)  ;
        L3=SUBSTR(LOWER_DATE(I),7,2)  ;
        U1=SUBSTR(UPPER_DATE(I),1,2)  ;
        U2=SUBSTR(UPPER_DATE(I),4,2)  ;
        U3=SUBSTR(UPPER_DATE(I),7,2)  ;
        LOWER_DATE(I)=L1||L2||L3; UPPER_DATE(I)=U1||U2||U3;
        END;END;
SKIP1:  PUT LIST('OUT STATION REQUESTS')SKIP;
        GET FILE(CARDS) DATA (OUT_STATIONS);
        IF OUT_STATIONS=0 THEN BEGIN;
        PUT LIST('NO OUT STATIONS REQUESTED')SKIP;
        GO TO NEXT;END;
        OUTPUT_HEADER.STATIONS=OUT_STATIONS;
        FREE     OUTPUT_STATION;
        ALLOCATE OUTPUT_STATION;
        OUT_PARAMS=0;
        ALLOCATE DUM_OUT_PARAM;
        ALLOCATE DUM_OUTPUT_PARAM;
        DO I=1 TO OUT_STATIONS;
        PARAMS=0;
        GET FILE(CARDS) DATA(STATION,PARAMS);
        PUT DATA(STATION)SKIP;
        DO N=1 TO FILE_HEADER.STATIONS;
        IF STATION=FILE_STATION(N).NAME THEN GO TO FIND_N2;
        END;/* N LOOP*/
        PUT LIST('STATION NAME ',STATION,' NOT RECOGNIZED');GO TO NEXT;
FIND_N2:OUTPUT_STATION(I).NAME=STATION;
        OUTPUT_STATION(I).POINTER=OUT_PARAMS;
        IF PARAMS=0 THEN BEGIN;
        PARAMS=FILE_STATION(N).PARAMS;
        IF OUT_PARAMS+PARAMS>100 THEN BEGIN;
        PUT LIST('NUMBER OF PARAMETERS GREATER THAN 100')SKIP;
        GO TO NEXT;END;
        ELSE BEGIN   ;
        OUTPUT_STATION(I).PARAMS=PARAMS;
        DELAY=0;
        DO J=1 TO PARAMS;
        DUM_OUTPUT_PARAM(OUT_PARAMS+J).STATION=STATION;
        PARAM=
        FILE_PARAM(FILE_STATION(N).POINTER+J).NAME;
        PUT DATA(PARAM,DELAY)SKIP;
        DUM_OUTPUT_PARAM(OUT_PARAMS+J).NAME=  PARAM;
        DUM_OUT_PARAM(OUT_PARAMS+J).POINTER=FILE_STATION(N).POINTER+J;
        DUM_OUT_PARAM(OUT_PARAMS+J).DELAY=DELAY;
        END; /* PARAM LOOP */
        GO TO NEWSTA; END;        END;
        IF OUT_PARAMS+PARAMS>100 THEN BEGIN;
        PUT LIST('NUMBER OF PARAMETERS GREATER THAN 100')SKIP;
        GO TO NEXT;END;
        OUTPUT_STATION(I).PARAMS=PARAMS;
        DO J=1 TO PARAMS;
        DELAY=0;
        GET FILE(CARDS) DATA(PARAM,DELAY);
        PUT DATA(PARAM,DELAY)SKIP;
        DUM_OUTPUT_PARAM(OUT_PARAMS+J).NAME=PARAM;
        DUM_OUTPUT_PARAM(OUT_PARAMS+J).STATION=STATION;
        DO M=1 TO FILE_STATION (N).PARAMS;
        IF PARAM=FILE_PARAM(FILE_STATION(N).POINTER+M).NAME
        THEN GO TO FIND_M2;
```

```
        END;/* M LOOP*/
        PUT LIST('PARAMETER NAME ',PARAM,' NOT RECOGNIZED');GO TO NEXT;
FIND_M2:DUM_OUT_PARAM(OUT_PARAMS+J).POINTER=FILE_STATION(N).POINTER+M;
        DUM_OUT_PARAM(OUT_PARAMS+J).DELAY=DELAY;
        END; /* PARAM LOOP */
NEWSTA: OUT_PARAMS=OUT_PARAMS+PARAMS;
        END; /*STATION LOOP */
        OUTPUT_HEADER.PARAMS=OUT_PARAMS;
        FREE      OUTPUT_PARAM;
        ALLOCATE OUTPUT_PARAM;
        FREE      OUT_PARAM;
        ALLOCATE OUT_PARAM;
        FREE      PRINT_STATION;
        ALLOCATE PRINT_STATION;
        FREE      PRINT_NAME;
        ALLOCATE PRINT_NAME;
        DO J=1 TO OUT_PARAMS;
        OUTPUT_PARAM(J)=DUM_OUTPUT_PARAM(J);
        OUT_PARAM(J)=DUM_OUT_PARAM(J);
        STATION=OUTPUT_PARAM(J).STATION;
        PARAM  =OUTPUT_PARAM(J).NAME    ;
        UNSPEC(PRINT_STATION(J))=UNSPEC(STATION);
        UNSPEC(PRINT_NAME    (J))=UNSPEC(PARAM);
        END;
        FREE DUM_OUTPUT_PARAM;
        FREE DUM_OUT_PARAM;
        RETURN;
        END INPUT;
SEARCH: PROC;
DCL     HOUR FIXED BINARY;
DCL     DATE_KEY BIT(1) INITIAL('0'B);
DCL     DATE_LIMIT FIXED BINARY;
DCL     DELAY_HOUR FIXED BINARY;
DCL     YEAR CHAR(2), MONTH CHAR(2), DAY CHAR(2);
        IDUM=24; JDUM=OUT_PARAMS;
        DATE_LIMIT=1;
        FREE      PRINT_PARAM;
        ALLOCATE PRINT_PARAM;
        ON ENDFILE(MASTER)  BEGIN;
        PUT LIST('END OF FILE SENSED ON MASTER')SKIP;
        GO TO RETURN; END;
        READ FILE(MASTER) INTO (TODAY);
        YESTERDAY=TODAY;
DATES:  /* LOOP ON DATES - TEST BETWEEN DATE LIMITS */
        IF TODAY.DATE=LOWER_DATE(DATE_LIMIT) THEN BEGIN;
        DATE_KEY='1'B; GO TO HOURS;
        END;
        IF TODAY.DATE=UPPER_DATE(DATE_LIMIT) THEN BEGIN;
        DATE_KEY='0'B;DATE_LIMIT=DATE_LIMIT+1;
        GO TO HOURS;
        END;
        IF DATE_KEY='0'B THEN GO TO NEWDATE;
HOURS:  /* LOOP ON HOURS */
        YEAR =SUBSTR(TODAY.DATE,1,2);
        MONTH=SUBSTR(TODAY.DATE,3,2);
        DAY  =SUBSTR(TODAY.DATE,5,2);
        UNSPEC(PRINT_DATE(1))=UNSPEC(YEAR||'/'||MONTH||'/'||DAY);
        DO HOUR=1 TO 24;
        DO J=1 TO OUT_PARAMS;
        DELAY_HOUR=HOUR-OUT_PARAM(J).DELAY;
        IF DELAY_HOUR<1 THEN BEGIN;
        PRINT_PARAM(J,HOUR)=
        YESTERDAY.PARAM(24+DELAY_HOUR,OUT_PARAM.POINTER(J));
        GO TO SKIP;END;
        ELSE   BEGIN;
        PRINT_PARAM(J,HOUR)=
        TODAY.PARAM(DELAY_HOUR,OUT_PARAM.POINTER(J));   END;
SKIP:   END; /* PARAM LOOP */
        END; /* HOUR LOOP */
/*      CLOSE FILE(SYSPRINT);    */
        CALL TWODPT(PRINT_PARAM(1,1),IDUM,JDUM,PRINT_TITLE(1),
        PRINT_DATE(1),PRINT_STATION(1),PRINT_NAME(1));
NEWDATE:IF DATE_LIMIT>DATE_LIMITS THEN RETURN;
        YESTERDAY=TODAY;
        READ FILE(MASTER) INTO (TODAY);
        GO TO DATES;
RETURN: RETURN;
        END SEARCH;
STOP:   END APPRINT;
```

2. TIMESER Time Series Plotting Program (YOLYPLOT)

```
      SUBROUTINE FORT(NPAR,NOBS,X,SNAM,PNAM)
      DIMENSION X(NOBS,NPAR),A(17544)
      REAL*8 SNAM(1),PNAM(1)
      REAL*8 TITLE1(9)/'                                  TIME SERIES
     X                                    '/
      REAL*8 TITLE2(9)/'                          TAM SO2 VS. WEATHER
     X                                    '/
      IF(NOBS.EQ.0) RETURN
      IF(NOBS.GT.0) GO TO 5
      NCYCLE=-1
      CALL YOLYPLOT(X,Y,XMAX,XPINCH,YMAX,NCYCLE,NOBS,ID,XLABEL,YLABEL,
     YIFST,INCX,INCY,XMIN,YMIN,SYMBOL,TITLE,NN)
      RETURN
    5 TWO=2.0
      CALL TWOTEN(TWO)
      NTAM=8
      DAYSPR=2.0
      SO2MAX=.4
      DX=1./24.
      A(1)=DX
      DO 10 I=2,NOBS
   10 A(I)=A(I-1)+DX
      XMAX=NOBS/24.
      XMIN=0.0
      YMIN=0.0
      DO 20 IP=1,NTAM
   20 CALL YOLYPL(A,X(1,IP),XMAX,DAYSPR,SO2MAX,0,-NOBS,NOBS,
     1SNAM(IP),PNAM(IP),IP,1,1,XMIN,YMIN,1H.,TITLE1,NN)
      IPZ=NTAM+1
      DO 30 IP=IPZ,NPAR
      WMAX=0.0
      WMIN=0.0
   30 CALL YOLYPL(A,X(1,IP),XMAX,DAYSPR,WMAX,0,-NOBS,NOBS,SNAM(IP),
     PPNAM(IP),IP,1,1,XMIN,WMIN,1H.,TITLE1,NN)
C     DO 40 IT=6,NTAM,3
C     IFST=1
C     DO 40 IP=IPZ,NPAR
C     XMAX=0.0
C     XMIN=0.0
C     XPINCH=0.0
C     CALL YOLYPL(X(1,IP),X(1,IT),XMAX,XPINCH,SO2MAX,0,NOBS,IT,
C    2PNAM(IP),PNAM(IT),IFST,1,1,XMIN,YMIN,1H+,TITLE2,NN)
C  40 IFST=IFST+1
      RETURN
      END
      SUBROUTINE YOLYPL(T,S,TMAX,TPINCH,SMAX,NYCYC   ,NPTS,K,/TLABEL/
     S,/SLABEL/,IFSTCALL,INCX,INCY,TMIN,SMIN,SYM,/TITLE/,YFMT)
      REAL*8 $A
      DIMENSION T(100),S(100)
     D                 ,YN(11),YP(11)
      COMMON /W01/ IGRAPH(3081)
      DATA KEYPLOTS/-1/, XGRAPH/6.0/
     D,TEN/10.0/,YGRAPH/0.0/
      NYCYCLES=NYCYC
      N=IABS (NPTS)
      IF(KEYPLOTS.GE.0) GO TO 5
      CALL PLOTS(IGRAPH,12324,49)
      KEYPLOTS=1
    5 IF(NYCYCLES.LT.0) GO TO 25
      NS=1
      NT=1
      IF(IFSTCALL.EQ.1) GO TO 2
      XG=0.0
      YG=  +(TEN+.5)
      IF(YGRAPH-YG.LE.-10.0) GO TO 2
      YGRAPH=YGRAPH-YG
      GO TO 3
    2 XG=XGRAPH
      YG=YGRAPH
      YGRAPH=0.0
    3 SPINCH=(SMAX-SMIN)/TEN
      IF(TMAX.NE.TMIN) GO TO 7
      TMAX=-999.0
      TMIN=+9999.
      I=1-INCX
      NT=0
      DO 6II=1,N
      I=I+INCX
      IF(T(I).GE.9999.) GO TO 6
      IF(T(I).GT.TMAX) TMAX=T(I)
      IF(T(I).LT.TMIN) TMIN=T(I)
```

```
      NT=NT+1
    6 CONTINUE
      CALL DXDY(TMAX,TMIN,TPINCH)
      TMAX=TMIN+TPINCH
      TPINCH=TPINCH/10.0
    7 IF(SMAX.NE.SMIN.OR.NYCYCLES.NE.0) GO TO 9
      SMAX=-999.0
      SMIN=+9999.
      I=1-INCY
      NS=0
      DO 8II=1,N
      I=I+INCY
      IF(S(I).GE.9999.) GO TO 8
      IF(S(I).GT.SMAX) SMAX=S(I)
      IF(S(I).LT.SMIN) SMIN=S(I)
      NS=NS+1
    8 CONTINUE
      CALL DXDY(SMAX,SMIN,DS)
      SMAX=SMIN+DS
      SPINCH=DS/TEN
    9 CALL PLOT(XG,YG,-3)
      IF(XG.NE.0.0)
     XCALL SYMBOL(-1.0,0.5,.14,TITLE,90.0,72)
      C=K
      IF(XG.NE.0.0)
     XCALL NUMBER(-1.5,0.0,.28,C,0.0,0)
      IF(NS*NT.EQ.0) RETURN
      XMAX=(TMAX-TMIN)/TPINCH
C     CALL AXIS(0,0,TLABEL,-8,XMAX,0,TMIN,TPINCH)
      CALL AXIS(0,0,TLABEL,-8,XMAX,0,TMIN,TPINCH,10.0)
      IF(NYCYCLES.EQ.0) GO TO 30
      FCTR=10./NYCYCLES
      NCP1=NYCYCLES+1
      YMAX=ALOG(S  MAX)/2.3025850930
      YMIN=YMAX-NYCYCLES
      YINC=10./NYCYCLES
      INCHES=XMAX/6.+1.6667
      XGRAPH=INCHES*6.
      YN(1)=10.**YMIN
      YP(1)=-.07
      DO 10 IC=1,NYCYCLES
      YN(IC+1)=YN(IC)*10.0
   10 YP(IC+1)=YP(IC)+YINC
      CALL PLOT(0,0,3)
      CALL PLOT(0,10.0,2)
      CALL SYMBOL(-1.5,3.0,.14,SLABEL,90.,8)
      DO 20 IC=1,NCP1
C  20 CALL NUMBER(-1.0,YP(IC),.14,YN(IC),0,YFMT)
      CALL CONVO('(1PE7.1)',$A,0,K$,YN(IC))
   20 CALL SYMBOL(-1.0,YP(IC),.14,$A, 0.0,K$)
   22 CALL PLOT(0,0,3)
      ICX=1-INCX
      ICY=1-INCY
      DO 550 ICC=1,N
      ICX=ICX+INCX
      ICY=ICY+INCY
      X=(T(ICX)-TMIN)/TPINCH
      IF(NPTS.GT.0) X=X-.02
      Y = S(ICY)
      IF (Y .GT. 0.0)
     XY=(ALOG(S  (ICY))/2.3025850930-YMIN)*FCTR
      IF(NPTS.GT.0) Y=Y-.035
      IF(Y.GT.10.00) Y=10.035
      IF(Y.LT.-.035) Y=-.07
      IF(NPTS.GT.0.OR.ICC.EQ.1)
     XCALL SYMBOL(X,Y,.07,SYM,0,1)
      IF(NPTS.LT.0) CALL PLOT(X,Y,2)
  550 CONTINUE
      RETURN
   25 CALL PLOT(XGRAPH,0.0,-3)
      CALL PLOT(0,0,-3)
      CALL PLOT(0,0,999)
      RETURN
   30 CONTINUE
      CALL AXIS(0,0,SLABEL, 8,TEN ,90.0,SMIN,SPINCH,10.0)
      INCHES=XMAX+5.995
      XGRAPH=INCHES
   35 CONTINUE
      ICX=1-INCX
      ICY=1-INCY
      IUP=1
```

```
      DO 40 ICC=1,N
      ICX=ICX+INCX
      ICY=ICY+INCY
      IF(S(ICY).LT.9999.) GO TO 75
      IUP=1
      GO TO 40
   75 X=(T(ICX)-TMIN)/TPINCH
      IF(NPTS.GT.0) X=X-.02
      Y=(S(ICY)-SMIN)/SPINCH
      IF(NPTS.GT.0) Y=Y-.035
      IF(X.GT.XGRAPH) X=XMAX
      IF(Y.GT.TEN) Y=TEN+.035
      IF(Y.LT.-.035) Y=-.07
      IF(NPTS.GT.0.OR.IUP.EQ.1)
     XCALL SYMBOL(X,Y,.07,SYM,0,1)
      IUP=0
      IF(NPTS.LT.0) CALL PLOT(X,Y,2)
   40 CONTINUE
      GO TO 50
   45 IF(NYCYCLES)25,35,22
   50 RETURN
      ENTRY TWOTEN(TWO)
      TEN=TWO
      RETURN
      END
      SUBROUTINE DXDY(YMAX,YMIN,TDY)
      TDY=0.0
      ADY=YMAX-YMIN
      IF(ADY)5,100,5
    5 K1=0
    8 IF(ADY-10.0)10,15,15
   10 ADY=ADY*10.0
      K1=K1+1
      GO TO 8
   15  IF (YMIN)  20,30,20
   20  K=ABS (YMIN)*10.**(K1-1)*2.+.999
       V=K*5.
       IF (YMIN)  22,21,21
   21   V=-V+5.
   22  YMIN=-V/10.**K1
   30  K1=0
      ADY=YMAX-YMIN
   40 CONTINUE
      IF(ADY-10.0)50,50,60
   45 K1=K1-1
      ADY=ADY*10.0
      GO TO 40
   50 IF(ADY-8.0)45,45,55
   55 TDY=10.0
      GO TO 96
   60 IF(ADY-20.0)65,65,70
   65 TDY=20.0
      GO TO 96
   70 IF(ADY-25.0) 72,72,74
   72 TDY=25.0
      GO TO 96
   74 IF(ADY-40.0)75,75,80
   75 TDY=40.0
      GO TO 96
   80 IF(ADY-50.0)85,85,90
   85 TDY=50.0
      GO TO 96
   90 IF(ADY-80.0)95,95,91
   91 K1=K1+1
      ADY=ADY/10.0
      GO TO 40
   95 TDY=80.0
   96 CONTINUE
      TDY=TDY*10.0**K1
  100 RETURN
      END
```

3. CITYMAP Mesometeorological Mapping Program

```
      SUBROUTINE FORT(NP,NH,X,STANAM,PARNAM)
      DIMENSION X(NH,NP)
      REAL*8 STANAM(1),PARNAM(1)
CCC+MIKE MATTHIES,TAMS=1-8,ANL=9,MIDWAY=10,MEIGS=11,GLENVIEW=12,OHARE=13
      COMMON/STNS/C(2,13)
CITY OUTLINE
      COMMON/CITY/CC(2,150)
      DIMENSION A$(5)
      DIMENSION BUFF(1000)
      DATA KEY/0/
      PRINT 10,NH,NP
   10 FORMAT(12I6)
      IF(KEY.NE.0) GO TO 15
C INITIALIZATION
      KEY=1
      CALL PLOTS(BUFF,4000,49)
      PRINT 10,NH,NP
C MOVE 2 STATIONS TO THE RIGHT
      C(1,9)=C(1,9)+1.0                                                  X
      C(1,11)=C(1,11)+1.0                                                X
C N(TYPE)(S OR P) = NUMBER OF STATIONS OR PARAMS OF THAT TYPE
C I(TYPE)(S OR Z) = INDEX OF 1ST STATION OR PARAM OF THAT TYPE
C MAY SET IRRELEVANT ITEMS TO ZERO BUT DO NOT OMIT ANY INITIALIZATIONS
      AMILE=.275                                                         M
      NTAMS=8                                                            N
      NTAMP=3                                                            N
      ICALZ=NTAMS*NTAMP+1
      IWILDZ=ICALZ+4
      NWILDS=1                                                           N
      NWILDP=7                                                           N
      IWILDS=NTAMS+1
      IADDZ=IWILDZ+NWILDS*NWILDP
      NADDP=1                                                            N
      IAIRS=IWILDS+NWILDS
      IAIRZ=IADDZ+NADDP
      NAIRP=7                                                            N
C IWZ=THE INDEX OF THE WIND VELOCITY FOR THE 1ST AIRPORT
      IWZ=IAIRZ+1
CCC IS THE HEIGHT OF THE NUMBERS FOR THE PLOTTED VALUES IN INCHES
      CCC=.10
C
C                        10=PRECIPITATION          (WILDCAT STATIONS      W/A
C                         9=NET RADIATION            OR ADDED TO THE      W/A
C                         8=STABILITY                 FIRST AIRPORT)      W/A
C                                                 5=CLOUD COVER
C   1=TEMPERATURE   |----------------|
C   7=VISIBILITY    |  3=WIND DIRECTION|  2=WIND VELOCITY
C   6=DEW POINT     |----------------|  4=CEILING
C                         1=TAM SO2 (PPM)
C                          =1ST PARAMETER
C                           FOR TAMS ONLY
C
C REPOSITION THE CALCOMP PLOTTING PAPER FOR THE NEXT MAP
   15 CONTINUE
      CALL PLOT(8.5,0,-3)
      IF(NH.NE.0) GO TO 20
C LEAVE EXTRA SPACE IN BETWEEN SEARCH RUNS
      CALL PLOT(8.5,0,-3)
      RETURN
   20 IF(NH.GT.0) GO TO 25
C TERMINATE PLOTTING
      CALL PLOT(12.0,0.0,-3)
      CALL PLOT(0.0,0.0,999)
      RETURN
C BEGIN PLOTTING ANOTHER MAP
   25 CONTINUE
C TAM STATIONS      TAM_N(SO2,WIND_VEL,WIND_DIR,       )
      IW=2
C THE 2ND PARAMETER IS THE WIND VELOCITY FOR THE 1ST TAM STATION
      DO 8 IS=1,NTAMS
      CALL SWD(IS,X(1,IW),X(1,IW+1))
      CALL NUMBER(C(1,IS)-.24,C(2,IS)-.45,.14,X(1,IW-1),0,2)
    8 IW=IW+NTAMP
      CALL KEYS(PARNAM(IAIRZ))
CALENDAR INFORMATION    CALENDAR(HOUR,DAY_MON,MONTH,YEAR)
      IM=X(1,ICALZ+2)
      ID=X(1,ICALZ+1)
      IY=X(1,ICALZ+3)
      IH=X(1,ICALZ)*100.0
C DAY/MONTH/YEAR    AND TIME    IN THE UPPER LEFT CORNER ABOVE 9.5 INCHES
      CALL CONVO('(I2,"/",I2,"/",I2," AT ",I4)',A$,0,K$,IM,ID,IY,IH)
      CALL SYMBOL(0.5,9.5,.14,A$,0,K$)
C MILEAGE SCALE ABOUT .5 INCHES UP FROM CENTER BOTTOM (2.5,.5,.35,.35)
      SMILE=2.5+AMILE
```

```
      DO 3125 MILE=1,3
      CALL SYMBOL(SMILE,.5,AMILE,1HL,90.0,1)
      FILE=MILE
      CALL NUMBER(SMILE,.35,.10,FILE,0,0)
 3125 SMILE=SMILE+AMILE
      CALL SYMBOL(SMILE,.35,.10,5HMILES,0,5)
C WILDCAT STATION
C ANL (9,29-35) TEMP,WS1,WD1,LR4,DEWPT,PRECIP,NETRAD
C WILDCAT WINDS
      IW=IWILDZ+1
      IS=IWILDS
      CALL SWD(IS,X(1,IW),X(1,IW+1))
C OTHER PARAMETERS FOR WILDCAT STATION TO PLOT AT POSITIONS 1,8,6,10,9
      IWILD=IWILDS
      XX=C(1,IWILD)
      Y=C(2,IWILD)
      CALL NUMBER(XX      -.6,Y      +.16,CCC,X(1,29),0,0)
      CALL NUMBER(XX      -.24,Y      +.3,CCC,X(1,32),0,1)
      CALL NUMBER(XX      -.6,Y      -.3,CCC,X(1,33),0,0)
      CALL NUMBER(XX      -.24,Y      +.76,CCC,X(1,34),0,3)
      CALL NUMBER(XX      -.24,Y      +.53,CCC,X(1,35),0,1)
C EXTRA SYMBOL FOR ANL SINCE IT WAS MOVED IN TOWARD THE CITY
      CALL SYMBOL(XX      -1.0,Y      ,.14,11,0,-1)
C ADDED PARAMETER AT THE FIRST AIRPORT WEATHER STATION ONLY
      IS=IAIRS
C STAB FOR MIDWAY(10,36) TURNERB AT POSITION NUMBER 8
      CALL NUMBER(C(1,IS)-.24,C(2,IS)+.3,CCC,X(1,IADDZ),0,0)
C AIRPORT STATIONS
C
C                        10=(-.24,+.76)
C                         9=(-.24,+.53)
C                         8=(-.24,+.30)
C      1=(-.6,+.16)  |----------------|  5=(+.3,+.16)
C      7=(-.6,+.07)  |   HEIGHT=.10   |  2=(+.3,-.07)
C      6=(-.6,-.30)  |----------------|  4=(+.3,-.30)
C                         1=(-.24,-.45)
C                           HEIGHT=.14
C                          FOR SO2 ONLY
C
C MIDWAY(10,37-43) TEMP,VEL,DIR,CEIL,CLDSUM4,DEWPT,VIS
C MEIGS(11,44-50)   GLENVIEW(12,51-57)    OHARE(13,58-64)
C AIRPORT WINDS
      IS=IAIRS
C IWZ=THE INDEX OF THE WIND VELOCITY FOR THE 1ST AIRPORT
      DO 45 IW=IWZ,NP,NAIRP
      CALL SWD(IS,X(1,IW),X(1,IW+1))
   45 IS=IS+1
C EXTRA SYMBOL FOR MEIGS AIRPORT SINCE IT HAD TO BE MOVED OVER
      CALL SYMBOL(C(1,11)-1.0,C(2,11),.14,11,0,-1)
C AIRPORT PARAMETERS OTHER THAN WIND
      IS=IAIRS
      DO 100 IIP=IAIRZ,NP,NAIRP
      IP=IIP-1
      CALL NUMBER(C(1,IS)-.6,C(2,IS)+.16,CCC,X(1,IP+1),0,0)
      CALL NUMBER(C(1,IS)+.3,C(2,IS)-.30,CCC,X(1,IP+4),0,0)
      CALL NUMBER(C(1,IS)+.3,C(2,IS)+.16,CCC,X(1,IP+5),0,1)
      CALL NUMBER(C(1,IS)-.6,C(2,IS)-.3,CCC,X(1,IP+6),0,0)
      CALL NUMBER(C(1,IS)-.6,C(2,IS)-.07,CCC,X(1,IP+7),0,1)
  100 IS=IS+1
CORNER L AT EACH CORNER TO BRACKET AND LINE UP THE MAP
      CALL SYMBOL(0,0,.35,1HL,0,1)
      CALL SYMBOL(7.5,0,.35,1HL,90.0,1)
      CALL SYMBOL(7.5,10.0,.35,1HL,180.0,1)
      CALL SYMBOL(0,10.0,.35,1HL,270.0,1)
CITY OUTLINE (ASSUMES 150 POINTS)
      CALL PLOT(CC(1,1),CC(2,1),3)
      DO 150 ICC=1,150
      CALL PLOT(CC(1,ICC),CC(2,ICC),2)
  150 CONTINUE
      RETURN
      END
      SUBROUTINE SWD(II,SPD,DIR)
      COMMON/STNS/C(2,13)
CCC+MIKE MATTHIES,TAMS=1-8,ANL=9,MIDWAY=10,MEIGS=11,GLENVIEW=12,OHARE=13
      X=C(1,II)
      Y=C(2,II)
      CALL SYMBOL(X-.04,Y-.07,.14,1H+,0,1)
      IF(SPD.NE.0.0) GO TO 40
      CALL SYMBOL(X-.08,Y-.140,.28,1H0,0,1)
      RETURN
   40                 CALL SYMBOL(X-.04,Y-.07,.14,112,0,-1)
      IF(DIR.EQ.0.0.OR.DIR.GT.360.0) RETURN
C CCC
```

```
      CALL SYMBOL(X    ,Y      ,.3,62,180.-DIR,-1)
C CCC
      IF(SPD.GE.999.0) RETURN
      CALL NUMBER(X+.3,Y-.070,.10,SPD,0,0)
      RETURN
      END
      SUBROUTINE KEYS(P)
      REAL*8 P(1)
CCC+MIKE MATTHIES,TAMS=1-8,ANL=9,MIDWAY=10,MEIGS=11,GLENVIEW=12,OHARE=13
      COMMON/STNS/C(2,13)
C (X,Y) IS A CONVENIENT LOCATION FOR THE KEY TO BE OUT OF THE WAY
      X=C(1,12)+3.5
      Y=C(2,12)
C
C                        10=PRECIPITATION          (WILDCAT STATIONS      W/A
C                         9=NET RADIATION            OR ADDED TO THE      W/A
C                         8=STABILITY                  FIRST AIRPORT)     W/A
C   1=TEMPERATURE      |-----------------| 5=CLOUD COVER
C   7=VISIBILITY       | 3=WIND DIRECTION| 2=WIND VELOCITY
C   6=DEW POINT        |-----------------| 4=CEILING
C                         1=TAM SO2 (PPM)
C                          =1ST PARAMETER
C                           FOR TAMS ONLY
C
      CALL SYMBOL(X-.04,Y-.07,.14,1H+,0,1)
      CALL SYMBOL(X-.04,Y-.07,.14,1HO,0,1)
      CALL SYMBOL(X-.8,Y+.165,.07,P(1),0,8)
      CALL SYMBOL(X-.8,Y-.035,.07,P(7),0,8)
      CALL SYMBOL(X-.8,Y-.235,.07,P(6),0,8)
      CALL SYMBOL(X+.14,Y+.165,.07,P(5),0,8)
      CALL SYMBOL(X+.14,Y-.235,.07,P(4),0,8)
      CALL SYMBOL(X+.14,Y-.035,.07,P(2),0,8)
      CALL SYMBOL(X-.14,Y-.45,.14,2HSO,0,2)
      CALL SYMBOL(X+.08,Y-.50,.07,1H2,0,1)
C WILDCAT AND/OR ADDITIONAL PARAMETERS IN POSITIONS 8,9,AND,10          W/A
      CALL SYMBOL(X-.28,Y+.3,.07,9HSTABILITY,0,9)                       W/A
      CALL SYMBOL(X-.28,Y+.515,.07,13HNET RADIATION,0,13)               W/A
      CALL SYMBOL(X-.28,Y+.73,.07,13HPRECIPITATION,0,13)                W/A
      RETURN
      END
      BLOCK DATA
C     COMMON/STNS/S(2,13)
C     TAM-1,....,TAM-8,ANL,MDW,CGX,NBU,ORD
C     I TH COLUMN CONTAINS X- AND Y-COORDINATES OF I TH POINT
C     X(I) = S(1,I)              Y(I) = S(2,I)
      COMMON/STNS/S(26)
      DATA S/2.92,6.45, 4.68,5.88, 5.18,4.32, 5.80,2.58, 4.59,2.62,
     1 5.03,0.88, 3.35,4.55, 3.72,1.95, 0.00,0.92, 3.34,2.72, 5.58,4.08,
     2 2.52,8.42, 1.15,6.25/
      COMMON/CITY/C(300)
C     COMMON/CITY/C(2,150)
C     COORDINATES OF CHICAGO PERIMETER : ANL ON Y-AXIS   AND
C           SOUTH-EAST TIP  OF CHICAGO ON X-AXIS
C     I TH COLUMN CONTAINS X- AND Y-COORDINATES OF I TH POINT
C     X(I) = C(1,I)              Y(I) = C(2,I)
      DIMENSION C1(190),C2(68),C3(21)
      COMPLEX C3
      EQUIVALENCE (C(1),C1(1)),(C(191),C2(1)),(C(259),C3(1))
      DATA C1/6.60,0.00, 6.60,1.58, 6.56,1.92, 6.39,2.01, 6.42,2.10,
     1 6.39,2.15, 6.32,2.12, 6.18,2.30, 6.00,2.54, 6.05,2.58, 5.97,2.64,
     2 5.93,2.82, 5.97,2.85, 5.91,2.89, 5.92,2.99, 5.79,3.20, 5.68,3.41,
     3 5.58,3.61, 5.51,3.92, 5.48,4.17, 5.52,4.17, 5.53,3.99, 5.56,3.95,
     4 5.59,3.99, 5.58,4.15, 5.60,4.20, 5.47,4.22, 5.47,4.62, 5.69,4.65,
     5 5.55,4.67, 5.55,4.69, 5.62,4.69, 5.62,4.75, 5.47,4.74, 5.33,4.91,
     6 5.32,5.03, 5.38,5.10, 5.31,5.16, 5.28,5.32, 5.21,5.59, 5.09,5.88,
     7 5.10,5.96, 5.18,5.98, 5.22,6.00, 5.20,6.06, 5.11,6.08, 5.05,6.14,
     8 5.01,6.30, 4.99,6.45, 4.91,6.52, 4.86,6.84, 4.80,6.93, 4.75,7.17,
     9 4.58,7.17, 4.65,7.08, 4.12,7.08, 4.12,6.69, 3.51,6.69, 3.38,6.78,
     A 3.38,6.86, 3.31,6.86, 3.31,6.78, 3.25,6.82, 3.21,6.95, 3.05,6.95,
     B 3.15,6.81, 3.32,6.70, 3.02,6.70, 2.99,6.74, 2.75,6.74, 2.75,7.09,
     C 2.60,7.09, 2.55,6.44, 2.12,6.52, 2.14,6.42, 2.08,6.41, 2.02,6.23,
     D 2.09,6.21, 2.05,5.92, 2.18,5.90, 2.18,5.82, 2.10,5.81, 2.14,5.60,
     E 2.62,5.60, 2.62,5.51, 2.76,5.51, 2.76,5.05, 3.18,5.05, 3.18,4.21,
     F 3.65,4.21, 3.65,3.28, 3.43,3.18, 3.43,2.95, 2.78,2.91, 2.78,2.45/
      DATA C2/3.60,2.45, 3.60,1.72, 4.42,1.72, 4.42,1.30, 4.28,1.30,
     1 4.28,1.18, 4.01,1.18, 3.98,1.29, 3.89,1.29, 3.89,0.90, 3.60,0.90,
     23.60,0.78, 3.98,0.78, 3.98,0.65, 4.08,0.65, 4.08,0.70, 4.15,0.70,
     3 4.15,0.87, 4.20,0.87, 4.20,1.05, 4.28,1.05, 4.28,0.81, 4.38,0.81,
     4 4.38,0.65, 4.72,0.65, 4.72,0.51, 5.02,0.51, 5.02,0.45, 4.88,0.45,
     5 4.88,0.27, 5.27,0.27, 5.27,0.12, 5.32,0.12, 5.32,0.05/
C REPEAT THE LAST POINT FOR THE REST OF THE 150 POINTS
      DATA C3/21*(6.60,0.00)/
      END
```

4. IHISTJJ Printed Graphical Displays (JJPLOT and HIST)

```
      SUBROUTINE FORT(NPAR,NOBS,X,SNAM,PNAM)
      DIMENSION X(NOBS,NPAR)
      REAL*8 SNAM(1),PNAM(1)
      REAL*8 TITLE (9)/'                                  TAM    SO2     VS.    TEMP.
     X                                  '/
      DIMENSION JPLOT(25,50)
      DIMENSION FREQ(20)
      IF(NOBS.LE.0) RETURN
      NTAM=8
      SO2MAX=1.0
      TMAX=110.0
      TMIN=-10.0
      TITLE(4)=PNAM(1)
      TITLE(6)=PNAM(NTAM+2)
      DO 10 IT=1,NTAM
      TITLE(3)=SNAM(IT)
      CALL JJPLOT(NOBS,TMAX,TMIN,SO2MAX,0.0,X(1,NTAM+2),X(1,IT),12,20,
     JTITLE ,'(7X,10(F6.2,6X))','(2X,F6.3,1X,1H-)',JPLOT)
      DO 60 IF=1,20
      NF=0
      DO 50 I=1,12
   50 NF=NF+JPLOT(I,IF)
   60 FREQ(IF)=NF
      CALL HIST(IT,FREQ,20)
      PRINT 25,TITLE (1),TITLE (2),SNAM(IT),PNAM(IT)
   25 FORMAT(4A8)
      DO 80 IF=1,12
      NF=0
      DO 70 I=1,20
   70 NF=NF+JPLOT(IF,I)
   80 FREQ(IF)=NF
      CALL HIST(IT,FREQ,12)
      PRINT 25,TITLE (1),TITLE (2),SNAM(IT),PNAM(NTAM+2)
   10 CONTINUE
      PMAX=32.0
      PMIN=28.0
      TITLE(3)=SNAM(NTAM+1)
      TITLE(4)=PNAM(NTAM+1)
      CALL JJPLOT(NOBS,TMAX,TMIN,PMAX,PMIN,X(1,NTAM+2),X(1,NTAM+1),12,16
     J,TITLE ,'(7X,10(F6.2,6X))','(2X,F6.3,1X,1H-)',JPLOT)
      RETURN
      END
      SUBROUTINE JJPLOT(NXY,XMAX,XMIN,YMAX,YMIN,X,Y,JX,JY,TITLE,FMTX,
     1 FMTY,JPLOT)
      DIMENSION X(2400),Y(2400),JPLOT(25,50),
     1 XX(30),YY(55),TITLE(15),FMTX(4),FMTY(5)
C       PLOT GRAPH WITH FREQ DIST OF POINTS WITHIN BANDS X,X+DX AND
C        Y,Y+DY WHERE DX=XMAX-XMIN/JX ETC.
C       LIMIT JX TO 20 AND JY TO 50.
C       LIMIT X AND Y ARRAYS TO 2400 ENTRIES EACH.
C
C       FMTX IS THE FOMAT IN COLS 1-16 FOR THE PRINTOUT OF THE
C       X VARIABLE VALUES.  IT MUST BE IN THE FORM  (5X,10(F6._,4X))
C       WHERE THE BLANK IS FILLED ACCORDING TO THE NUMBER OF DECIMAL
C        PLACES(FROM 1 TO 5)        EVERY OTHER X AXIS VALUE IS PRINTED.
C
C      FMTY IS SIMILARLY DEFINED BUT MUST BE O/F THE FORM
C        (2X,F6._,1X,'-')      STARTING IN COL 21.
C
C       ZERO ARRAY JPLOT
      DO 10 M=1,20
      DO 10 N=1,50
   10 JPLOT(M,N)=0
C       SET XX AND YY AXES.
      DX=(XMAX-XMIN)/JX
      DY=(YMAX-YMIN)/JY
      DO 20 N=1,JX
   20 XX(N)=XMIN+(N-1)*DX
      DO 25 N=1,JY
   25 YY(N)=YMIN+(N-1)*DY
      XX(JX+1)=XMAX
      YY(JY+1)=YMAX
      XR=JX/(XMAX-XMIN)
      YR=JY/(YMAX-YMIN)
C       NXY IS THE NUMBER OF X AND Y PAIRS.
      DO 50 I=1,NXY
      MX=XR*(X(I)-XMIN) + 1
      MY=YR*(Y(I)-YMIN) + 1
      IF(X(I).LT.XX(MX))  MX=MX-1
      IF(Y(I).LT.YY(MY)) MY=MY-1
      IF(MX.LE.0) MX=1
```

```
      IF(MY.LE.0) MY=1
      IF(X(I).GE.XMAX)MX=JX
      IF(Y(I).GE.YMAX)MY=JY
   50 JPLOT(MX,MY)=JPLOT(MX,MY)+1
      PRINT 401 , TITLE
  401 FORMAT(1H1,3X,20A4 ///)
      JXX=JX+1
      DO 60 N=1,JY
      PRINT FMTY,YY(JY-N+2)
   60 PRINT 403,(JPLOT(M,JY-N+1),M=1,JX)
      PRINT FMTY,YY(1)
  403 FORMAT(10X,25(1X,I4,1X)          )
      PRINT 404
  404 FORMAT(10X,20('.',5X)             )
      PRINT FMTX,(XX(M),M=1,JXX,2)
      DO510I=1,10
  510 PRINT408
  408 FORMAT(20X,/)
      RETURN
      END
      SUBROUTINE HIST(NU,FREQ,IN)                                       HIST 320
C                                                                       HIST  10
C     ..................................................................HIST  20
C                                                                       HIST  30
C        SUBROUTINE HIST                                                HIST  40
C                                                                       HIST  50
C        PURPOSE                                                        HIST  60
C           PRINT A HISTOGRAM OF FREQUENCIES VERSUS INTERVALS           HIST  70
C                                                                       HIST  80
C        USAGE                                                          HIST  90
C           CALL HIST(NU,FREQ,IN)                                       HIST 100
C                                                                       HIST 110
C        DESCRIPTION OF PARAMETERS                                      HIST 120
C           NU   - HISTOGRAM NUMBER (3 DIGITS MAXIMUM)                  HIST 130
C           FREQ - VECTOR OF FREQUENCIES                                HIST 140
C           IN   - NUMBER OF INTERVALS AND LENGTH OF FREQ (MAX IS 20)   HIST 150
C                  NORMALLY, FREQ(1) CONTAINS THE FREQUENCY SMALLER THANHIST 160
C                  THE LOWER BOUND AND FREQ(IN) CONTAINS THE FREQUENCY  HIST 170
C                  LARGER THAN THE UPPER BOUND                          HIST 180
C                                                                       HIST 190
C        REMARKS                                                        HIST 200
C           FREQUENCIES MUST BE POSITIVE NUMBERS                        HIST 210
C                                                                       HIST 220
C        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED                  HIST 230
C           NONE                                                        HIST 240
C                                                                       HIST 250
C        METHOD                                                         HIST 260
C           THE LARGEST FREQUENCY IS DETERMINED AND SCALING IS USED     HIST 270
C           IF REQUIRED                                                 HIST 280
C                                                                       HIST 290
C     ..................................................................HIST 300
C                                                                       HIST 310
C                                                                       HIST 330
      DIMENSION JOUT(20),FREQ(20)
      DATA K/4H*   /,NOTH/4H    /                                       HIST 340
C                                                                       HIST 350
    1 FORMAT(6H EACH ,A1,8H EQUALS ,I2,7H POINTS,/)
    2 FORMAT(1X,I5,4X,20(4X,A1))                                        HIST 370
    3 FORMAT(9H0INTERVAL,4X,19(I2,3X),I2)                               HIST 380
    4 FORMAT(1H1,47X,11H HISTOGRAM ,I3)                                 HIST 390
    5 FORMAT(10H0FREQUENCY,20I5)                                        HIST 400
    6 FORMAT(6H CLASS)                                                  HIST 410
    7 FORMAT(113H   ----------------------------------------------------)HIST 420
     1-----------------------------------------------------------------HIST 560
C                                                                       HIST 570
C        PRINT TITLE AND FREQUENCY VECTOR                               HIST 580
C                                                                       HIST 590
      WRITE(6,4) NU                                                     HIST 600
      DO 12 I=1,IN                                                      HIST 610
   12 JOUT(I)=FREQ(I)                                                   HIST 620
      WRITE(6,5)(JOUT(I),I=1,IN)                                        HIST 630
      WRITE(6,7)                                                        HIST 640
C                                                                       HIST 650
C        FIND LARGEST FREQUENCY                                         HIST 660
C                                                                       HIST 670
      FMAX=0.0                                                          HIST 680
      DO 20  I=1,IN                                                     HIST 690
      IF(FREQ(I)-FMAX) 20,20,15                                         HIST 700
   15 FMAX=FREQ(I)                                                      HIST 710
   20 CONTINUE                                                          HIST 720
C                                                                       HIST 730
C        SCALE IF NECESSARY
```

```
C
      JSCAL=1                                                           HIST 740
      IF(FMAX-50.0) 40,40,30                                            HIST 750
   30 JSCAL=(FMAX+49.0)/50.0                                            HIST 760
      WRITE(6,1)K,JSCAL                                                 HIST 770
C                                                                       HIST 780
C        CLEAR OUTPUT AREA TO BLANKS                                    HIST 790
C                                                                       HIST 800
   40 DO 50 I=1,IN                                                      HIST 810
   50 JOUT(I)=NOTH                                                      HIST 820
C                                                                       HIST 830
C        LOCATE FREQUENCIES IN EACH INTERVAL                            HIST 840
C                                                                       HIST 850
      MAX=FMAX/FLOAT(JSCAL)                                             HIST 860
      DO 80 I=1,MAX                                                     HIST 870
      X=MAX-(I-1)                                                       HIST 880
      DO 70 J=1,IN                                                      HIST 890
      IF(FREQ(J)/FLOAT(JSCAL)-X) 70,60,60                               HIST 900
   60 JOUT(J)=K                                                         HIST 910
   70 CONTINUE                                                          HIST 920
      IX=X*FLOAT(JSCAL)                                                 HIST 930
C                                                                       HIST 940
C        PRINT LINE OF FREQUENCIES                                      HIST 950
C                                                                       HIST 960
   80 WRITE(6,2)IX,(JOUT(J),J=1,IN)                                     HIST 970
C                                                                       HIST 980
C        GENERATE CONSTANTS                                             HIST 990
C                                                                       HIST1000
      DO 90 I=1,IN                                                      HIST1010
   90 JOUT(I)=I                                                         HIST1020
C                                                                       HIST1030
C        PRINT INTERVAL NUMBERS                                         HIST1040
C                                                                       HIST1050
      WRITE(6,7)                                                        HIST1060
      WRITE(6,3)(JOUT(J),J=1,IN)                                        HIST1070
      WRITE(6,6)                                                        HIST1080
      RETURN                                                            HIST1090
      END                                                               HIST1100
                                                                        HIST1110
```

APICS USER MANUAL

APPENDIX H

COMPUTATIONAL PROGRAMS
SOURCE LISTINGS

1. REGRES Linear Regression Analysis

```
      SUBROUTINE REGRES(NVAR,NOBS,KVAR,NDEP,X,ISAVE,STANAM,PARNAM)
C    1B,STDB,T,BZERO)
      REAL*8 A ,STANAM(1),PARNAM(1)
      REAL*8 TITLE1(9),TITLE2(9),RCTITL
      REAL X(NOBS,1),B(1),STDB(1),T(1),ANS(10)
      INTEGER ISAVE(1)
C
      COMMON/STOR /A(5000)
C
      DATA TITLE1/'SUMS OF ' ,'CROSS PR','ODUCTS O','F DEVIAT',
     1'IONS    ',4*'          '/
      DATA TITLE2/'CORRELAT','ION COEF','FICIENTS',6*'          '/
      DATA RCTITL/'  VAR NO'/
C
      CALL POINTR(A,1,5000,3)
      CALL PUTPNT(8HXBAR     ,NVAR,4)
      CALL PUTPNT(8HSTD      ,NVAR,4)
      CALL PUTPNT(8HRX       ,NVAR*NVAR,4)
      CALL PUTPNT(8HR        ,(NVAR*(NVAR+1))/2+1,4)
      CALL PUTPNT(8HDIAG     ,NVAR,4)
      CALL PUTPNT(8HDUM1     ,NVAR,4)
      CALL PUTPNT(8HDUM2     ,NVAR,4)
      IF(IPTERR(DUM).GT.0)STOP
      CALL CORRE(NOBS,NVAR,1,X,A(IGET(8HXBAR     )),
     1A(IGET(8HSTD      )),A(IGET(8HRX       )),A(IGET(8HR        )),
     1A(IGET(8HDIAG     )),A(IGET(8HDUM1     )),A(IGET(8HDUM2     )))
      CALL PRINT1(A(IGET(8HXBAR     )),A(IGET(8HSTD      )),NVAR)
      CALL TWODPR(A(IGET(8HRX       )),NVAR,NVAR,TITLE1,RCTITL,RCTITL,
     10,0,0,DUM,DUM)
      CALL CLEAR(8HRX       )
      CALL STORE(A(IGET(8HRX       )),A(IGET(8HR        )),NVAR)
      CALL TWODPR(A(IGET(8HRX       )),NVAR,NVAR,TITLE2,RCTITL,RCTITL,
     10,0,0,DUM,DUM)
      CALL REDEF(8HRX       ,KVAR*KVAR,4)
      CALL CLEAR(8HRX       )
      CALL PUTPNT(8HRY       ,KVAR,4)
      IF(IPTERR(DUM).GT.0)STOP
      CALL ORDER(NVAR,A(IGET(8HR        )),NDEP,KVAR,
     1        ISAVE ,    A(IGET(8HRX       )),A(IGET(8HRY       )))
      CALL MINV(A(IGET(8HRX       )),KVAR,DET,A(IGET(8HDUM1     )),
     1A(IGET(8HDUM2     )))
      CALL MULTR(NOBS,KVAR,A(IGET(8HXBAR     )),A(IGET(8HSTD      )),
     1A(IGET(8HDIAG     )),A(IGET(8HRX       )),A(IGET(8HRY       )),
     2        ISAVE,B,STDB,T,ANS)
      CALL PRINT2(B,STDB,T,ANS,ISAVE,NDEP,KVAR,STANAM,PARNAM)
      BZERO=ANS(1)
      DO 100 I=1,25
      PRINT 110
  110 FORMAT(' ')
  100 CONTINUE
      RETURN
      END
      SUBROUTINE PRINT1(XBAR,STD,NVAR)
C
C     PRINTS MEANS AND STANDARD DEVIATIONS
C
      REAL*4 XBAR(1),STD(1)
C
      PRINT 5
    5 FORMAT('1REGRESSION ANALYIS RESULTS')
      PRINT 10
   10 FORMAT(4X6HVAR NO10X5HMEANS20H STANDARD DEVIATIONS)
      DO 30 M=1,NVAR
      PRINT 20,M,XBAR(M),STD(M)
   20 FORMAT(I10,E15.5,5XE15.5)
   30 CONTINUE
      RETURN
      END
      SUBROUTINE PRINT2(B,STDB,T,ANS,ISAVE,NDEP,KVAR,STANAM,PARNAM)
C
      REAL*8 STANAM(1),PARNAM(1)
      REAL B(1),STDB(1),T(1),ANS(1)
      INTEGER ISAVE(1)
C
      PRINT 5
    5 FORMAT('1REGRESSION ANALYIS RESULTS')
      PRINT 10
   10 FORMAT(1H02X7HSTATION1X9HPARAMETER4X6HVAR NO7X8HREG COEF
     12X13HS.D. REG COEF8X7HT VALUE)
      DO 30 K=1,KVAR
      PRINT 20,STANAM(K),PARNAM(K),ISAVE(K),B(K),STDB(K),T(K)
```

```
   20 FORMAT(2XA8,2XA8,1I10,3E15.7)
   30 CONTINUE
      PRINT 40
   40 FORMAT('ODEPENDENT VARIABLE')
      PRINT 20,STANAM(KVAR+1),PARNAM(KVAR+1),ISAVE(KVAR+1),ANS(1),ANS(3)
      PRINT 5
      PRINT 100,ANS(2)
  100 FORMAT('OMULTIPLE CORRELATION COEFFICIENT='E15.7)
      PRINT 110,ANS(4)
  110 FORMAT('OSUM OF SQUARES ATTRIBUTABLE TO REGRESSION(SSAR)='E15.7)
      PRINT 120,ANS(5)
  120 FORMAT('ODEGREES OF FREEDOM ASSOCIATED WITH SSAR='E15.7)
      PRINT 130,ANS(6)
  130 FORMAT('OMEAN SQUARE OF SSAR='E15.7)
      PRINT 140,ANS(7)
  140 FORMAT('OSUM OF SQUARES OF DEVIATIONS FROM REGRESSION(SSDR)='
     1E15.7)
      PRINT 150,ANS(8)
  150  FORMAT('ODEGREES OF FREEDOM ASSOCIATED WITH SSDR='E15.7)
      PRINT 160,ANS(9)
  160 FORMAT('OMEAN SQUARE OF SSDR='E15.7)
      PRINT 170,ANS(10)
  170 FORMAT('OF-VALUE='E15.7)
      RETURN
      END
      SUBROUTINE STORE(RX,R,NVAR)
C
C     STORES R IN RX
C
      REAL*4 RX(NVAR,1),R(1)
C
      DO 10 I=1,NVAR
      DO 10 J=1,NVAR
      CALL LOC(I,J,IJ,NVAR,NVAR,1)
      RX(I,J)=R(IJ)
   10 CONTINUE
      RETURN
      END
      SUBROUTINE DATA
      RETURN
      END
      SUBROUTINE TWODPR(ARAY,ICT,JCT,TITLE,ROWT,COLT,INVRT,IPONE,JPONE,
     1RNUM,CNUM)
C                                TWODPR PRINTS A TWO-DIMENSIONAL ARRAY.  IT
C                               CALLS PAGHED TO GET PAGE HEADING AND PRINTS
C                               A TITLE LINE AND COLUMN TITLES WHICH ARE
C                               FOLLOWED BY 50 LINES UP TO SEVEN COLUMNS
C                               WIDE PLUS THE INDEX COLUMN.  ARGUMENTS ARE-
C
C                                    TITLE - PAGE TITLE (9A8).
C                                     ROWT - LINE TITLE (A8).
C                                     COLT - COLUMN TITLE (A8).
C                                     ARAY - THE TWO-DIMENSIONAL ARRAY
C                                            TO BE PRINTED.
C                                      ICT - NUMBER OF ROWS IN ARRAY.
C                                      JCT - NUMBER OF COLUMNS IN ARRAY.
C                                    INVRT - IF ZERO PRINT ARRAY AS STORED.
C                                            IF NOT ZERO INTERCHANGE ROWS
C                                            AND COLUMNS BEFORE PRINTING.
C                                    IPONE - IF ZERO NUMBER ROWS WITH INTE-
C                                            GERS  1,2,3, . . .,ICT.
C                                            IF NOT ZERO NUMBER ROWS WITH
C                                            MEMBERS OF FLOATING POINT
C                                            ARRAY RNUM(ICT).
C                                    JPONE - IF ZERO NUMBER COLS WITH INTE-
C                                            GERS  1,2,3, . . .,JCT.
C                                            IF NOT ZERO NUMBER COLS WITH
C                                            MEMBERS OF FLOATING POINT
C                                            ARRAY CNUM(JCT).
C                                     RNUM - NUMBERS FOR ROWS IF DESIRED.
C                                     CNUM - NUMBERS FOR COLS IF DESIRED.
C
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*4 ARAY(ICT,1)
      DIMENSION          RNUM(1),CNUM(1),TITLE(9)
    5 FORMAT('1REGRESSION ANALYIS RESULTS')
      II=1
      IJ=1
      IMAX=0
      JMAX=0
```

```
      MAX=0
      IFLAG=0
   10 IF(INVRT.NE.0) GO TO 30
   20 IEND=JCT
      GO TO 40
   30 IEND=ICT
   40 MIN=MAX+1
      MAX=MINO(MAX+7,IEND)
      MDLT=MAX-MIN+1
      IF(IFLAG.NE.0) GO TO 47
   45 NROW=0
   46 CONTINUE
      PRINT 5
      PRINT 230,(TITLE(I),I=1,9)
   47 PRINT 240,ROWT,(COLT,JC=MIN,MAX)
      IF(JPONE.EQ.0) GO TO 49
      PRINT 250,(CNUM(JC),JC=MIN,MAX)
      PRINT 250
      GO TO 50
   49 PRINT 260,(JC,JC=MIN,MAX)
      PRINT 260
   50 IF(INVRT.NE.0) GO TO 90
      IMAX=MINO(II+52,ICT)
      IDIF=IMAX-II+5
      NROW=NROW+IDIF
      JMAX=MINO(IJ+6,JCT)
      DO 65 I=II,IMAX
      IF(IPONE.NE.0) GO TO 62
   61 ID=I
      PRINT 210,ID,(ARAY(I,J),J=IJ,JMAX)
      GO TO 65
   62 FID=RNUM(I)
      PRINT 220,FID,(ARAY(I,J),J=IJ,JMAX)
   65 CONTINUE
      IF(IMAX.GE.ICT) GO TO 70
      II=IMAX+1
      GO TO 46
   70 IF(JMAX.GE.JCT) GO TO 130
   80 II=1
      IJ=JMAX+1
      IF((NROW+IDIF-57).GT.0) GO TO 84
   82 IFLAG=1
      GO TO 40
   84 IFLAG=0
      GO TO 40
   90 IMAX=MINO(II+6,ICT)
      JMAX=MINO(IJ+52,JCT)
      IDIF=JMAX-IJ+5
      NROW=NROW+IDIF
      DO 95 J=IJ,JMAX
      IF(IPONE.NE.0) GO TO 92
   91 ID=J
      PRINT 210,ID,(ARAY(I,J),I=II,IMAX)
      GO TO 95
   92 FID=RNUM(J)
      PRINT 220,FID,(ARAY(I,J),I=II,IMAX)
   95 CONTINUE
      IF(JMAX.GE.JCT) GO TO 110
  100 IJ=JMAX+1
      GO TO 46
  110 IF(IMAX.GE.ICT) GO TO 130
  120 IJ=1
      II=IMAX+1
      IF((NROW+IDIF-57).GT.0) GO TO 124
  122 IFLAG=1
      GO TO 40
  124 IFLAG=0
      GO TO 40
  130 CONTINUE
  210 FORMAT(I8,2X,1P7E15.6)
  220 FORMAT(F10.3,1P7E15.6)
  230 FORMAT(1H0,19X,9A8)
  240 FORMAT(1H0,1X,A8,7(7X,A8))
  250 FORMAT(10X,7(5X,F10.3))
  260 FORMAT(10X,7(5X,I10))
      RETURN
      END
      SUBROUTINE POINTR(BLK,LSTLOC,MAXSIZ,JPRINT)
C                                                                       00000100
C     MANIPULATES POINTERS FOR VARIABLE STORAGE                         00000200
C     ADJUSTS STORAGE IF STORAGE EXCEEDED                               00000300
                                                                        00000400
```

```
C     MULT=2  NON STANDARD INTEGER                                      00000500
C     MULT=4  STANDARD INTEGER OR REAL                                  00000600
C     MULT=8  NON STANDARD REAL                                         00000700
C                                                                       00000800
      REAL*8 BLK                                                        00000900
      REAL*8 NAME,NAMLST,BLANK                                          00001000
      REAL NAM1,NAM2                                                    00001100
C                                                                       00001200
      DIMENSION BLK(1)                                                  00001300
      DIMENSION NAM1(2),NAM2(2)                                         00001400
C                                                                       00001500
      COMMON/LOCATE/LSTBLK,MAXBLK,IPRINT                                00001600
      COMMON/TABLES/NAMLST(100),LENLST(100),IPTLST(100),                00001700
     1LEN(100),MLT(100),NNAMS                                           00001800
      COMMON/NAMBLK/NAME                                                00001900
      COMMON/PTERR/NPTERR                                               00002000
C                                                                       00002100
      EQUIVALENCE (NAME,NAM2)                                           00002200
C                                                                       00002300
      DATA BLANK/'        '/                                            00002400
C                                                                       00002500
      CALL PRGSET(BLK)                                                  00002600
      CALL REDSET(BLK)                                                  00002700
      NNAMS=0                                                           00002800
      NPTERR=0                                                          00002900
      LSTBLK=LSTLOC                                                     00003000
      MAXBLK=MAXSIZ                                                     00003100
      IPRINT=JPRINT                                                     00003200
      DO 10 I=1,100                                                     00003300
      NAMLST(I)=0                                                       00003400
      IPTLST(I)=0                                                       00003500
      LENLST(I)=0                                                       00003600
      LEN(I)=0                                                          00003700
      MLT(I)=0                                                          00003800
   10 CONTINUE                                                          00003900
      DO 20 I=LSTBLK,MAXBLK                                             00004000
      BLK(I)=0                                                          00004100
   20 CONTINUE                                                          00004200
      IF(IPRINT.EQ.0)RETURN                                             00004300
      IF(IPRINT.LT.2)RETURN                                             00004400
      PRINT 25,LSTBLK,MAXBLK                                            00004500
   25 FORMAT('1POINTR HAS BEEN INITIALIZED'/                            00004600
     1'0THE CONTAINER ARRAY HAS BEEN CLEARED FROM LOCATION'I10,         00004700
     1' TO LOCATION'I10)                                                00004800
      RETURN                                                            00004900
C                                                                       00005000
      ENTRY INITL  (BLK)                                                00005100
C                                                                       00005200
      CALL PRGSET(BLK)                                                  00005300
      CALL REDSET(BLK)                                                  00005400
      RETURN                                                            00005500
C                                                                       00005600
      ENTRY PUTPNT(NAM1,LENDUM,MULT)                                    00005700
C                                                                       00005800
C     PUT ARRAY NAME AND POINTER IN TABLE                               00005900
C                                                                       00006000
      NAM2(1)=NAM1(1)                                                   00006100
      NAM2(2)=NAM1(2)                                                   00006200
      LENGTH=((LENDUM-1)*MULT+8)/8                                      00006300
      IF(LSTBLK+LENGTH-1.LE.MAXBLK)GO TO 30                             00006400
C     STORAGE WILL BE EXCEEDED UNLESS GARBAGE IS CLEARED OUT            00006500
      CALL PURGE(DUMMY)                                                 00006600
      IF(LSTBLK+LENGTH.GT.MAXBLK)GO TO 1000                             00006700
   30 CONTINUE                                                          00006800
C     ENOUGH STORAGE - RECORD NAME AND POINTER IN TABLE                 00006900
      NNAMS=NNAMS+1                                                     00007000
      IF(NNAMS.GT.100)GO TO 1020                                        00007100
      NAMLST(NNAMS)=NAME                                                00007200
      IPTLST(NNAMS)=LSTBLK                                              00007300
      LENLST(NNAMS)=LENGTH                                              00007400
      LEN(NNAMS)=LENDUM                                                 00007500
      MLT(NNAMS)=MULT                                                   00007600
      LSTBLK=LSTBLK+LENGTH                                              00007700
      IF(IPRINT.LT.2)RETURN                                             00007800
      PRINT 40,NAME,IPTLST(NNAMS),LENLST(NNAMS),LEN(NNAMS),MLT(NNAMS)   00007900
   40 FORMAT(17H0ENTRY FOR ARRAY A8,35H HAS BEEN MADE IN THE POINTER TAB00008000
     1LE/9H POINTER=I5,9H DBL LEN=I5,10H ORIG LEN=I5,6H MULT=I5)        00008100
      RETURN                                                            00008200
C                                                                       00008300
      ENTRY WIPOUT(NAM1)                                                00008400
C                                                                       00008500
```

```
C     RECORD NAME IN PURGE TABLE
C
      CALL GETN(NAM1,N)
      IF(N.EQ.0)RETURN
      NAMLST(N)=BLANK
      IF(IPRINT.LT.2)RETURN
      PRINT 70,NAME
   70 FORMAT(7H0ARRAY A8,35H HAS BEEN PLACED IN THE PURGE TABLE)
      RETURN
C
      ENTRY DUMP(NAM1,JUMP)
C
C     DUMP VARIABLE DIMENSION ARRAY ON STANDARD OUTPUT
C
      IF(IPRINT.EQ.0.OR.IPRINT.EQ.2)RETURN
      CALL GETN(NAM1,N)
      IF(N.EQ.0)RETURN
      IPOINT=IPTLST(N)
      PRINT  80,NAME,IPTLST(N),LEN(N)
   80 FORMAT(15H0DUMP OF ARRAY A8,9H POINTER=I10,8H LENGTH=I10)
      GO TO ( 90,110,130),JUMP
   90 CONTINUE
      IF(MLT(N).EQ.4)GO TO 100
      CALL PRTI1(BLK(IPOINT),LEN(N))
      RETURN
  100 CONTINUE
      CALL PRTI2(BLK(IPOINT),LEN(N))
      RETURN
  110 CONTINUE
      IF(MLT(N).EQ.8)GO TO 120
      CALL PRTR1(BLK(IPOINT),LEN(N))
      RETURN
  120 CONTINUE
      CALL PRTR2(BLK(IPOINT),LEN(N))
      RETURN
  130 CONTINUE
      IF(MLT(N).EQ.8)GO TO 140
      CALL PRTA1(BLK(IPOINT),LEN(N))
      RETURN
  140 CONTINUE
      CALL PRTA2(BLK(IPOINT),LEN(N))
      RETURN
C
      ENTRY GETPNT(NAM1,IPT)
C
C     GET POINTER CORRESPONDING TO ARRAY NAME
C
      IPT=0
      CALL GETN(NAM1,N)
      IF(N.GT.0)GO TO 190
      RETURN
  190 CONTINUE
      IPT=IPTLST(N)
      IF(IPRINT.LT.2)RETURN
      PRINT 200,NAME,IPT
  200 FORMAT(19H0POINTER FOR ARRAY A8,38H HAS BEEN EXTRACTED FROM POINTE
     1R TABLE/9H POINTER=I10)
      RETURN
C
      ENTRY CLEAR(NAM1)
C
C     CLEARS ARRAY CORRESPONDING TO NAME
C
      CALL GETN(NAM1,N)
      IF(N.EQ.0)RETURN
      LENGTH=LENLST(N)
      IPT=IPTLST(N)
      DO 230 I=1,LENGTH
      BLK(IPT+I-1)=0
  230 CONTINUE
      IF(IPRINT.LT.2)RETURN
      PRINT 240,NAME
  240 FORMAT(7H0ARRAY A8,17H HAS BEEN CLEARED)
      RETURN
C
C     PRINT ERROR MESSAGES
C
 1000 CONTINUE
      NPTERR=NPTERR+1
      PRINT 1010,NAME,LENGTH,LSTBLK
 1010 FORMAT(54H0STORAGE EXCEEDED DURING REQUEST FOR STORAGE OF ARRAY A8
```

```
     1,/8H LENGTH=I10,25H LAST AVAILABLE LOCATION=I10)
      RETURN
 1020 CONTINUE
      NPTERR=NPTERR+1
      PRINT 1030
 1030 FORMAT(49HOMAXIMUM NUMBER OF ENTRIES IN NAME TABLE EXCEEDED)
      RETURN
      END
      SUBROUTINE GETN(NAM1,N)
C
      REAL*8 NAME,NAMLST
      REAL NAM1,NAM2
C
      DIMENSION NAM1(2),NAM2(2)
C
      COMMON/LOCATE/LSTBLK,MAXBLK,IPRINT
      COMMON/TABLES/NAMLST(100),LENLST(100),IPTLST(100),
     1LEN(100),MLT(100),NNAMS
      COMMON/NAMBLK/NAME
C
      EQUIVALENCE (NAME,NAM2)
C
      N=0
      NAM2(1)=NAM1(1)
      NAM2(2)=NAM1(2)
      IF(NNAMS.EQ.0)GO TO 20
      DO 10 N=1,NNAMS
      IF(NAMLST(N).EQ.NAME)RETURN
   10 CONTINUE
   20 CONTINUE
      N=0
      IF(IPRINT.LT.2)RETURN
      PRINT   30,NAME
   30 FORMAT(7HOARRAY A8,34H CANNOT BE FOUND IN THE NAME TABLE)
      RETURN
      END
      SUBROUTINE REDSET(BLK)
C
C     REDEFINES ARRAY NAME
C
      REAL*8 BLK
      REAL*8 NAME,NAMLST,BLANK,DUMNAM
      REAL NAM1,NAM2
C
      DIMENSION BLK(1)
      DIMENSION NAM1(2),NAM2(2),DUM(2)
C
C
      COMMON/LOCATE/LSTBLK,MAXBLK,IPRINT
      COMMON/TABLES/NAMLST(100),LENLST(100),IPTLST(100),
     1LEN(100),MLT(100),NNAMS
      COMMON/NAMBLK/NAME
      COMMON/PTERR/NPTERR
C
C
      EQUIVALENCE (NAME,NAM2),(DUMMY,DUM(1))
C
      DATA BLANK/'        '/,DUMNAM/'DUMNAM'/
C
      RETURN
C
      ENTRY REDEF(NAM1,LENDUM,MULT)
C
      CALL GETN(NAM1,N)
      IF(N.GT.0)GO TO 30
C
C     NAME NOT IN TABLE SO PUT IT THERE
C
      CALL PUTPNT(NAME,LENDUM,MULT)
      RETURN
   30 CONTINUE
      LENNEW=((LENDUM-1)*MULT+8)/8
      LENOLD=LENLST(N)
      IF(LENNEW.EQ.LENOLD)RETURN
      IF(LENNEW.GT.LENOLD)GO TO 60
C
C     NEW LENGTH LESS THAN OLD LENGTH
C
      NNAMS=NNAMS+1
      IF(NNAMS.LE.100)GO TO 50
      NPTERR=NPTERR+1
```

```
      PRINT 40
   40 FORMAT(49HOMAXIMUM NUMBER OF ENTRIES IN NAME TABLE EXCEEDED)
      RETURN
   50 CONTINUE
      L1=NNAMS-N-1
      DO 55 L=1,L1
      K=NNAMS-L
      NAMLST(K+1)=NAMLST(K)
      IPTLST(K+1)=IPTLST(K)
      LENLST(K+1)=LENLST(K)
      LEN    (K+1)=LEN    (K)
      MLT    (K+1)=MLT    (K)
   55 CONTINUE
      LENLST(N)=LENNEW
      LEN(N)=LENDUM
      MLT(N)=MULT
      NAMLST(N+1   )=BLANK
      IPTLST(N+1   )=IPTLST(N)+LENNEW
      LENLST(N+1   )=LENOLD-LENNEW
      IF(IPRINT.LT.2)RETURN
      PRINT 56,NAME,IPTLST(N     ),LENLST(N     ),LEN(N     ),MLT(N     )
   56 FORMAT(17HOENTRY FOR ARRAY A8,35H HAS BEEN MADE IN THE POINTER TAB
     1LE/9H POINTER=I5,9H DBL LEN=I5,10H ORIG LEN=I5,6H MULT=I5)
      RETURN
   60 CONTINUE
C
C     NEW LENGTH GREATER THAN OLD LENGTH
C
      NAMLST(N)=DUMNAM
      CALL PUTPNT(NAME,LENDUM,MULT)
      IF(NAMLST(N).EQ.DUMNAM)GO TO 90
      CALL GETN(DUMNAM,N)
      IF(N.GT.0)GO TO 90
      NPTERR=NPTERR+1
      RETURN
   90 CONTINUE
      NAMLST(N)=BLANK
      DO 100 L=1,LENOLD
      LSUB1=IPTLST(N)+L-1
      LSUB2=IPTLST(NNAMS)+L-1
      BLK(LSUB2)=BLK(LSUB1)
  100 CONTINUE
      RETURN
      END
      SUBROUTINE PRGSET(BLK)
C
C     SIFTS STORAGE AND ELIMINATES ARRAYS IN THE PURGE TABLE
C
      REAL*8 BLK
      REAL*8       NAMLST,BLANK
C
      DIMENSION BLK(1)
C
      COMMON/LOCATE/LSTBLK,MAXBLK,IPRINT
      COMMON/TABLES/NAMLST(100),LENLST(100),IPTLST(100),
     1LEN(100),MLT(100),NNAMS
C
      DATA BLANK/'        '/
C
      RETURN
C
      ENTRY PURGE(LSTLOC)
C
      IF(NNAMS.GT.0)GO TO 5
      LSTLOC=LSTBLK
      RETURN
    5 CONTINUE
      IMOV=1
      INAM=0
      DO 40 N=1,NNAMS
      IF(NAMLST(N).EQ.BLANK     )GO TO 40
   10 CONTINUE
C     NAME NOT BLANK          - SO SAVE IT
      IMAX=LENLST(N)
      IF(IMOV.EQ.IPTLST(N))GO TO 30
      DO 20 I=1,IMAX
      ISUB1=IMOV+I-1
      ISUB2=IPTLST(N)+I-1
      BLK(ISUB1)=BLK(ISUB2)
   20 CONTINUE
   30 CONTINUE
```

```
      INAM=INAM+1
      NAMLST(INAM)=NAMLST(N)
      IPTLST(INAM)=IMOV
      LENLST(INAM)=IMAX
      LEN(INAM)=LEN(N)
      MLT(INAM)=MLT(N)
      IMOV=IMOV+IMAX
   40 CONTINUE
      NNAMS=INAM
      LSTBLK=IMOV
      LSTLOC=LSTBLK
      DO 45 I=LSTBLK,MAXBLK
      BLK(I)=0.
   45 CONTINUE
      IF(IPRINT.LT.2)RETURN
      PRINT 50
   50 FORMAT(  '1STORAGE HAS BEEN SIFTED')
C
      ENTRY STATUS
C
      IF(IPRINT.LT.2)RETURN
      PRINT 55
   55 FORMAT('0CURRENT STATUS OF CONTAINER ARRAY IS AS FOLLOWS'/
     11H07X2HN06X4HNAME3X7HPOINTER3X7HDBL LEN2X8HORIG LEN6X4HMULT)
      DO 70 N=1,NNAMS
      PRINT 60,N,NAMLST(N),IPTLST(N),LENLST(N),LEN(N),MLT(N)
   60 FORMAT(1I10,2XA8,4I10)
   70 CONTINUE
      RETURN
      END
      FUNCTION IGET(NAM1)
C
C     GET POINTER CORRESPONDING TO ARRAY NAME
C
      REAL*8            NAME,NAMLST,BLANK
      REAL NAM1,NAM2
C
      DIMENSION NAM1(2),NAM2(2)
C
      COMMON/LOCATE/LSTBLK,MAXBLK,IPRINT
      COMMON/TABLES/NAMLST(100),LENLST(100),IPTLST(100),
     1LEN(100),MLT(100),NNAMS
      COMMON/NAMBLK/NAME
      COMMON/PTERR/NPTERR
C
      EQUIVALENCE (NAME,NAM2)
C
      IPT=0
      CALL GETN(NAM1,N)
      IF(N.GT.0)GO TO 10
      RETURN
   10 CONTINUE
      IGET=IPTLST(N)
      IF(IPRINT.LT.2)RETURN
      PRINT 60,NAME,IPTLST(N)
   60 FORMAT(19H0POINTER FOR ARRAY A8,38H HAS BEEN EXTRACTED FROM POINTE
     1R TABLE/9H POINTER=I10)
      RETURN
      END
      SUBROUTINE PRTI1 (J,LENGTH)
C
C     PRINTS NON STANDARD INTEGER ARRAY
      INTEGER*2 J(1)
C
      PRINT 10,(J(L),L=1,LENGTH)
   10 FORMAT(24I5)
      RETURN
      END
      SUBROUTINE PRTI2 (J,LENGTH)
C
C     PRINTS STANDARD INTEGER ARRAY
C
      DIMENSION J(1)
C
      PRINT 10,(J(L),L=1,LENGTH)
   10 FORMAT(12I10)
      RETURN
      END
      SUBROUTINE PRTR1 (A,LENGTH)
C
C     PRINTS STANDARD REAL ARRAY
```

```
C
      DIMENSION A(1)
C
      PRINT 10,(A(L),L=1,LENGTH)
   10 FORMAT(8E15.7)
      RETURN
      END
      SUBROUTINE PRTR2 (A,LENGTH)
C
C     PRINTS NON STANDARD REAL ARRAY
C
      REAL*8 A(1)
C
      PRINT 10,(A(L),L=1,LENGTH)
   10 FORMAT(8E15.7)
      RETURN
      END
      SUBROUTINE PRTA1 (A,LENGTH)
C
C     PRINTS STANDARD HOLLERITH ARRAY
C
      DIMENSION A(1)
C
      PRINT 10,(A(L),L=1,LENGTH)
   10 FORMAT(1H 25A4)
      RETURN
      END
      SUBROUTINE PRTA2 (A,LENGTH)
C
C     PRINTS NON STANDARD HOLLERITH ARRAY
C
      REAL*8 A(1)
C
      PRINT 10,(A(L),L=1,LENGTH)
   10 FORMAT(1H 12A8)
      RETURN
      END
      FUNCTION IPTERR(DUMMY)
C
C     RETURNS NUMBER OF ERRORS THAT HAVE OCCURED IN POINTER ROUTINES
C
      COMMON/PTERR/NPTERR
C
      IPTERR=NPTERR
      NPTERR=0
      RETURN
      END
      FUNCTION ILAST(DUMMY)
C
C     RETURNS FIRST AVAILABLE LOCATION IN CONTAINER ARRAY
C
      COMMON/LOCATE/LSTBLK,MAXBLK,IPRINT
C
      ILAST=LSTBLK
      RETURN
      END
APREG:  PROC OPTIONS(MAIN);
DCL     1 OUTPUT_HEADER,
            2 LABEL CHAR (80),
            2 STATIONS FIXED BINARY (31,0),
            2 NUMBER_PARAMS FIXED BINARY (31,0),
            2 NUMBER_OBS FIXED BINARY (31,0);
DCL     1 OUTPUT_STATION (OUTPUT_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL    1 OUTPUT_PARAM (NUMBER_PARAMS) CONTROLLED,
            2 STATION CHAR(8),
            2 NAME CHAR(8);
DCL     OUTPUT_DATA (NUMBER_PARAMS) CONTROLLED ;
DCL     REGRES_DATA (NUMBER_PARAMS,NUMBER_OBS) CONTROLLED   ;
DCL     ISAVE(NVAR+1          ) CONTROLLED FIXED BINARY (31,0);
DCL     PRINT_STATION(NVAR+1) CONTROLLED FLOAT DECIMAL (16);
DCL     PRINT_PARAM  (NVAR+1) CONTROLLED FLOAT DECIMAL (16);
DCL     NVAR FIXED BINARY (31,0), NDEP FIXED BINARY (31,0);
DCL     CLABEL CHAR(80);
DCL         TYPE CHAR (3), STATION CHAR (8),PARAM CHAR (8);
DCL     CARD CHAR(80);
NEXT:   /* BEGIN NEW CASE */
        FREE OUTPUT_STATION, OUTPUT_PARAM, OUTPUT_DATA,REGRES_DATA;
        CLOSE FILE(OUTPUT);
        CLOSE FILE(CARDS);
        NCARDS=0;
        PUT LIST('CARD INPUT LIST')PAGE;
GETCARD:GET EDIT(CARD)(A(80));
```

```
        PUT LIST (CARD)SKIP;
        WRITE FILE(CARDS) FROM (CARD);
        IF INDEX(CARD,'END ')¬=0 THEN GO TO END;
        IF INDEX(CARD,'STOP')¬=0 THEN GO TO STOP;
        NCARDS=NCARDS+1;
        GO TO GETCARD;
END:    CLOSE FILE(CARDS);
        PUT DATA(NCARDS)SKIP;
        CLOSE FILE (OUTPUT);
        ON ENDFILE(OUTPUT) BEGIN;
        PUT LIST('ARRAY LABEL NOT FOUND')PAGE;
        GO TO NEXT;END        ;
        GET FILE(CARDS) EDIT(CLABEL)(A(80));
        PUT LIST('TAPE SEARCH BEGUN FOR LABEL: ',CLABEL)PAGE;
READ:   READ FILE(OUTPUT) INTO (OUTPUT_HEADER);
        PUT DATA (OUTPUT_HEADER)SKIP;
        IF CLABEL¬= LABEL THEN BEGIN;
     READ FILE(OUTPUT) IGNORE  (NUMBER_OBS+2);
        GO TO READ;END;
        PUT LIST('ARRAY LABEL FOUND')SKIP;
        ALLOCATE OUTPUT_STATION;
        ALLOCATE OUTPUT_PARAM;
        ALLOCATE OUTPUT_DATA;
        ALLOCATE REGRES_DATA;
        READ FILE(OUTPUT) INTO (OUTPUT_STATION);
        PUT DATA (OUTPUT_STATION)SKIP;
        READ FILE(OUTPUT) INTO (OUTPUT_PARAM);
        PUT DATA (OUTPUT_PARAM  )SKIP;
        DO N=1 TO NUMBER_OBS;
        READ FILE(OUTPUT) INTO (OUTPUT_DATA);
        DO M=1 TO NUMBER_PARAMS;
        REGRES_DATA(M,N)=OUTPUT_DATA(M);
        END; END;
        NDEP=0;
        NVAR=NCARDS-2;
        IF NVAR<1 THEN BEGIN;
        PUT LIST('NUMBER OF VARIABLES LESS THAN ONE')PAGE;
        GO TO NEXT;END;
        FREE ISAVE; ALLOCATE ISAVE       ;
        ISAVE=0;
        FREE PRINT_STATION; ALLOCATE PRINT_STATION;
        FREE PRINT_PARAM  ; ALLOCATE PRINT_PARAM  ;
        KEY=0;
        DO N=1 TO NVAR+1;
        GET FILE(CARDS) DATA (    TYPE,STATION,PARAM)     ;
        PUT DATA(TYPE,STATION,PARAM)SKIP;
        IF      TYPE='IND'     |      TYPE= 'DEP'      THEN GO TO FIND_VAR;
        PUT LIST('VARIABLE TYPE NOT RECOGNIZED')SKIP;
        GO TO GET_VAR;
FIND_VAR: DO M=1 TO NUMBER_PARAMS;
        IF STATION=OUTPUT_PARAM(M).STATION&PARAM=OUTPUT_PARAM(M).NAME
        THEN GO TO SKIP1;
        END;
        PUT LIST('STATION OR PARAMETER NAME NOT RECOGNIZED');
        GO TO GET_VAR;
SKIP1:  IF      TYPE¬='DEP'      THEN GO TO SKIP2;
        NDEP=M;        KEY=1;
        ISAVE(NVAR+1)=M;
        GO TO GET_VAR;
SKIP2:  ISAVE(N-KEY)=M;
GET_VAR:END;
        IF NDEP=0 THEN BEGIN;
        PUT LIST('DEPENDENT VARIABLE NOT SPECIFIED')PAGE;
        GO TO NEXT;END;
        IF NVAR=1 THEN GO TO SKIP4;
        DO I=1 TO NVAR-1;
        IDUM=ISAVE(I);
        DO J=I+1 TO NVAR;
        IF ISAVE(J)<IDUM THEN BEGIN;
        ISAVE(I)=ISAVE(J); ISAVE(J)=IDUM; IDUM=ISAVE(I);  END;
        END;END;
SKIP4:  DO N=1 TO NVAR+1;
        UNSPEC(PRINT_STATION(N))=
        UNSPEC(OUTPUT_PARAM(ISAVE(N)).STATION);
        UNSPEC(PRINT_PARAM  (N))=
        UNSPEC(OUTPUT_PARAM(ISAVE(N)).NAME);
        END;
        PUT DATA(ISAVE)SKIP;
        PUT DATA(PRINT_STATION)SKIP;
        PUT DATA(PRINT_PARAM)  SKIP;
        CLOSE FILE(SYSPRINT);
        CALL REGRES(NUMBER_PARAMS,NUMBER_OBS,NVAR,NDEP,
        REGRES_DATA(1,1),ISAVE(1),PRINT_STATION(1),PRINT_PARAM(1));
        GO TO NEXT;
STOP:   END APREG;
```

2. STEPMDA Stepwise Multiple Discriminant Analysis

```
      SUBROUTINE SETUP(NPAR,NDATA,X,STANAM,PARNAM)
      REAL*8 VNAME,STANAM,PARNAM
      COMMON/B/ NOBS(20),NVAR,NG
      COMMON/NAMES/ VNAME(2 ,50)
      DIMENSION X(NDATA,NPAR)
      DIMENSION STANAM(1),PARNAM(1)
C
C        FOR EACH CASE SETUP
C              NVAR - NUMBER OF VARIABLES
C              NG - NUMBER OF GROUPS
C        STOP AT END OF ALL CASES
C
C
C        INSERT FORTRAN STATEMENTS HERE
      DATA IND1,IND2/0,0/
      IF ( IND1 .EQ. 1) STOP
      NVAR=5
      NG=2
      IND1=IND1+1
C
C
      RETURN
      ENTRY DATA(XX,ISIZE)
      DIMENSION XX(NG,NVAR,1000)
C
C        FOR EACH CASE SETUP THE FOLLOWING FOR ALL I,J,AND K
C              VNAME(1,I) - STATION NAME FOR I-TH VARIABLE
C              VNAME(2,I) - PARAM NAME FOR I-TH VARIABLE
C              NOBS(J) - NUMBER OF OBSERVATIONS IN J-TH GROUP
C                        (MUST NOT BE ZERO)
C              XX(J,I,K) - K-TH DATA POINT FORI-TH VARIABLE IN J-TH GROUP
C
C        INSERT FORTRAN STATEMENTS HERE
      NOBS(1)=0
      NOBS(2)=0
      DO 10 I=1,NDATA
      IF ( X(I,2) .LT. 266. ) III=1
      IF ( X(I,2) .GE. 266. ) III=2
      MMM=NOBS(III)+1
      DO 5 J=1,5
      VNAME(1,J)=STANAM(J)
      VNAME(2,J)=PARNAM(J)
    5 XX(III,J,MMM)=X(I,J)
      NOBS(III)=MMM
      XX(III,2,MMM)=X(I,4)-8.
   10 VNAME(1,2)=STANAM(4)
      IF (NOBS(1) .EQ. 0  .OR.NOBS(2).EQ.0) GO TO 20
C
C
      CALL PRINT (XX)
      CALL EIDISC(XX)
   20 CONTINUE
      RETURN
      END
      SUBROUTINE BORDER(A,C,IORD)
      DIMENSION A(50,50), C(50)
C     MATRIX A CONTIANS PREV. INVERSE, CONTIANS INVERSE ON RETURN.
C     C IS A DUMMY STORAGE ARRAY IN SEQ. FOR ECONOMY
      NORD=IORD+1
      ALPHA=0.0
      DO 5 J=1,IORD
      C(J)=0.0
      DO 5 I=1,IORD
    5 C(J)=C(J)-A(NORD,I)*A(I,J)
      DO 6 J=1,IORD
    6 A(NORD,J)=C(J)
      DO 10 I=1,IORD
      C(I)=0.0
      DO 9 J=1,IORD
    9 C(I)=C(I)-A(J,NORD)*A(I,J)
   10 ALPHA=ALPHA+A(NORD,I)*A(I,NORD)
      DO 11 I=1,IORD
   11 A(I,NORD)=C(I)
      ANN=A(NORD,NORD)
      ALPHA=ANN-ALPHA
      DO 20 I=1,IORD
      DO 20 J=1,IORD
      A(I,J)=A(I,J)-(A(I,NORD)*A(NORD,J)/ALPHA)
   20 CONTINUE
      DO 30 I=1,IORD
```

```
      A(NORD,I)=A(NORD,I)/ALPHA
      A(I,NORD)=A(I,NORD)/ALPHA
30    CONTINUE
      A(NORD,NORD)=1.0/ALPHA
      RETURN
      END
      SUBROUTINE CORRE (XX,II)                                          ???
      COMMON/B/ NOBS(20),NVAR,NNG                                     XXXXXXX
      DIMENSION XX(NNG,NVAR,1000)                                       XXXXX
      COMMON L(8),XM(50),GN(50),XBAR(50,50),SSD(50,50)
     1/DSC/X(50),SX(50),W(50,50),SUMT(50)
      COMMON/L1/T,NG,ENEK,EK1,MY,KRUNCH,ENTER1
      COMMON/L3/SS,TS,R,SD,KRUNH1
      DIMENSION SS(50,50),TS(50,50),R(50,50),SD(50)
      DIMENSION          TITLE(20),D(50,50)
      EQUIVALENCE (D,R)
      M=NVAR                                                            ???
      T=M
      NG=NOBS(II)
      READ  (5,22)(TITLE(I), I=1,20)
   22 FORMAT (20A4)
      L8=L(8)                                                           0620
  210 ENG=NG                                                            0640
      DO 230  I=1,M                                                     0650
  220 SX(I)=0.0                                                         0660
      DO 230  J=I,M                                                     0670
  230 SS(I,J)=0.0
      DO 260 JJ=1,NG                                                    0700
      DO 260 I = 1,M                                                     XXX
      SX(I)=SX(I)+XX(II,I,JJ)                                           0720
      DO 260 J = I,M                                                   XXXXX
  260 SS(I,J)=SS(I,J)+XX(II,I,JJ)*XX(II,J,JJ)                           0740
C     RAW SUM OF SQUARES AND CROSS-PRODUCTS                             0770
  280 DO 286  I=1,M                                                     0780
      DO 286  J=I,M                                                     0790
      SSD(I,J)=SS(I,J)-SX(I)*SX(J)/ENG                                  0800
      SS(J,I) = SS(I,J)                                                 0810
  286 SSD(J,I)=SSD(I,J)                                                 0820
C     DEVIATION SUM OF SQUARES AND CROSS-PRODUCTS                       0830
      DO 295  I=1,M                                                     0840
      XM(I)=SX(I)/ENG                                                   0850
  295 SD(I)=SQRT(SSD(I,I)/(ENG-1.0))                                    0860
C     MEANS AND STANDARD DEVIATIONS
      WRITE (6,31)(TITLE(I), I=1,20)
   31 FORMAT (26H1CORRELATION ANALYSIS FOR , 20A4)                      0890
      WRITE (6,32)T                                                     0900
   32 FORMAT (23HONUMBER OF VARIABLES = , F3.0)                         0910
      WRITE (6,33)NG                                                    0920
   33 FORMAT (36HONUMBER OF SUBJECTS IN THIS GROUP = , I5)             ********
  370 IF(L8)400,400,375                                                 1080
  375 WRITE (6,376)                                                     1090
  376 FORMAT(38H1RAW SUMS OF SQUARES AND CROSSPRODUCTS)
      CALL MPRNTV (SS, M, 3, 6HRAW SS,0,0,0,0,0)
  400 IF(L8)435,435,405                                                 1160
  405 WRITE (6,406)                                                     1170
  406 FORMAT(44H1DEVIATION SUMS OF SQUARES AND CROSSPRODUCTS)
      CALL MPRNTV ( SSD, M,3,6HDEV SS,0,0,0,0,0)                        1240
  435 DO 441  I=1,M                                                     1250
      DO 441  J=I,M                                                     1260
      D(I,J)=SSD(I,J)/(ENG-1.0)                                         1270
  441 D(J,I)=D(I,J)                                                     1280
C     DISPERSION MATRIX
      DO 452  I=1,M
  452 WRITE(4) (D(I,J),J=1,M)
  448 IF (L8)  475,475,450                                              1300
  450 WRITE (6,451)                                                     1310
  451 FORMAT(27H1VARIANCE-COVARIANCE MATRIX)
      CALL M PRNTV (D, M, 3, 6HD MAT ,0,0,0,0,0)                        1370
  475 DO 486 I = 1,M                                                    1380
      DO 486  J=I,M                                                     1390
      R(I,J) = D (I,J)/(SD(I)* SD(J))
      IF(SD(I).LE.0.0) R(I,J)=0.0
      IF(SD(J).LE.0.0) R(I,J)=0.0                                       1400
  486 R(J,I)=R(I,J)                                                     1410
C     CORRELATION MATRIX                                                1470
  625 IPNO=1
  628 WRITE (6,50)(TITLE (I), I = 1,20), NG, I PNO                      1490
   50 FORMAT (21H1CORRELATION MATRIX, , 20A4,1X2HN= , I4, 6X9HPAGE NO. ,  1500
     1 I2)
      CALL MPRNTV(R,M,1,5HR MAT, XM, 5HMEANS, SD,6HST.DEV,0)            1520
  700 RETURN
```

```
      END
      SUBROUTINE DIRNM
C     SUBROUTINE DIRNM.
C
C     A, M, B, X, AND XL ARE DUMMY NAMES AND MAY BE CHANGED IN THE
C     CALLING STATEMENT.
C
C     SUBROUTINE DIRNM, DIAGONALIZATION OF A REAL NON-SYMMETRIC MATRIX
C     OF THE FORM B-INVERSE*A.  CODED BY P. R. LOHNES, U. N. H.
C     A AND B ARE M BY M INPUT MATRICES.  UPON RETURN VECTOR XL CONTAINS
C     THE EIGENVALUES OF B-1*A, AND MATRIX X CONTAINS THE EIGENVECTORS
C     IN ITS COLUMNS.  SUBROUTINE HDIAG IS REQUIRED.
      COMMON/L3/A,B,X,XL,M
      DIMENSION A(50,50), B(50,50), X(50,50), XL(50)
       CALL HDIAG (B, M, 0, X, NR)
      DO 1  I = 1, M
    1 XL(I)=1.0/SQRT(ABS(B(I,I)))
      DO 2  I = 1, M
      DO 2  J = 1, M
    2 B(I,J) = X(I,J) * XL(J)
      DO 3  I = 1, M
      DO 3  J = 1, M
      X(I,J) = 0.0
      DO 3  K = 1, M
    3 X(I,J) = X(I,J) + B(K,I) * A(K,J)
      DO 4  I = 1, M
      DO 4  J = 1, M
      A(I,J) = 0.0
      DO 4  K = 1, M
    4 A(I,J) = A(I,J) + X(I,K) * B(K,J)
C     A NOW CONTAINS BPRIME*A*B OF THE NOTES.
      CALL HDIAG (A, M, 0, X, NR)
      DO 5  I = 1, M
    5 XL(I) = A(I,I)
      DO 6  I = 1, M
      DO 6  J = 1, M
      A(I,J) = 0.0
      DO 6  K = 1, M
    6 A(I,J) = A(I,J) + B(I,K) * X(K,J)
      DO 9  J = 1, M
      SUMV = 0.0
      DO 7  I = 1, M
    7 SUMV=SUMV+ ((ABS(A(I,J)))**2.0)
      DEN=SQRT(SUMV)
      DO 8  I = 1, M
    8 X(I,J) = A(I,J) / DEN
    9 CONTINUE
C     COLUMNS OF X(I,J) ARE NOW NORMALIZED.
      RETURN
      END
      SUBROUTINE DISCRM(XX)
      REAL*8 VNAME
      COMMON/B/ NOBS(20),NVAR,NNG
      DIMENSION XX(NNG,NVAR,1000)
      COMMON/L123/ LIN,K,M,ENTER,I,KRIT,KG,CNT,RNG,OPT2,OPT1,NTOT
      COMMON L(8),XM(50),GN(50),XBAR(50,50),SSD(50,50)
     1/DSC/X(50),SX(50),W(50,50),SUMT(50)
      COMMON/L1/T,NG,ENEK,EK1,MZ,KRUNCH,ENTER1
      COMMON/L3/SS,TS,R,SD,KRUNH1
      DIMENSION SS(50,50),TS(50,50),R(50,50),SD(50)
      DIMENSION ISUMT(50),D(50,50)
      EQUIVALENCE(SUMT,ISUMT),(D,SSD)
      COMMON/NAMES/VNAME(2,50)
      DIMENSION RNG(2,2)
      MZ=M
      ENTER1=ENTER
      EK = K
      EM = M
      WRITE (6,2)
    2 FORMAT (31HOMULTIPLE DISCRIMINANT ANALYSIS)
      WRITE (6,4)K
    4 FORMAT (17HONO. OF GROUPS = I2)
      WRITE (6,5)M
    5 FORMAT  (20HONO. OF VARIABLES = I2)
  100 N = 0
      DO 6  I = 1,M
      SUMT(I) = 0.0
      DO 6  J = 1,M
      W(I,J) = 0.0
    6 TS(I,J) = 0.0
      DO 8 II = 1, K
```

```
  200 CALL CORRE(XX,II)                                                 XXX
      N = N + NG                                                        0750
      DO 8  I = 1,M                                                     0760
      GN(II)=NG
      XBAR(II,I)=XM(I)
      SUMT(I) = SUMT(I) + SX(I)                                         0770
      DO 8  J = 1,M                                                     0780
      W(I,J) = W(I,J) + SSD(I,J)                                        0790
    8 TS(I,J) = TS(I,J) + SS(I,J)                                       0800
    9 EN = N                                                            0810
      DO 10  I = 1, M                                                   0820
      X(I) = SUMT(I) / EN                                               0850
      SX(I)=SQRT(W(I,I))
C     TOTAL SAMPLE MEANS ARE NOW IN X(I).                               0860
      DO 10  J = 1, M                                                   0870
      SS(I,J) = TS(I,J)                                                 0880
   10 TS(I,J) = SS(I,J) - (SUMT(I)*SUMT(J)/EN)                          0890
C     TS(I,J) IS NOW T, THE TOTAL DEVIATION-SCORES CROSS-PRODUCTS.      0900
      DO 11  I = 1, M                                                   0910
   11 SD(I) = SQRT(TS(I,I)/(EN-1.0))                                    0920
C     TOTAL SAMPLE STANDARD DEVIATIONS ARE NOW IN SD(I).                0930
      IF(L(7).NE.1) GO TO 20                                            0940
      WRITE (6,12)                                                      0950
   12 FORMAT (16H1POOLED W MATRIX)
      CALL MPRNTV (W, M, 3, 6HW MAT  ,0,0,0,0,0)
   14 FORMAT (8F15.4)                                                   0970
      WRITE (6,16)                                                      0980
   16 FORMAT (50H1TOTAL DEVIATION SUMS OF SQUARES AND CROSSPRODUCTS)    0990
      CALL MPRNTV (TS, M, 3, 6HT MAT ,0,0,0,0,0)
 20   DO 191 I = 1,M                                                    1090
      DO 191 J = I,M                                                    1100
      R(I,J) = TS(I,J) / ((EN - 1.0) * SD(I) * SD(J))                   1110
  191 R(J,I) = R(I,J)                                                   1120
C     R(I,J) NOW CONTAINS THE TOTAL R MATRIX.                           1130
      WRITE (6,192) N                                                   1140
  192 FORMAT (25H1TOTAL CORRELATION MATRIX,5X,25H TOTAL NO. OF SUBJECTS
     1=  ,I5)
      CALL MPRNTV (R,M,1,5HTOT R,X,5HMEANS,SD,6HST.DEV,0)
  400 DO 25  I = 1, M                                                   1300
      DO 25  J = 1, M                                                   1310
   25 D(I,J) = TS(I,J) - W(I,J)                                         1320
C     D(I,J) NOW CONTAINS THE A MATRIX.                                 1330
      IF(L(7).NE.1) GO TO 21                                            1340
      WRITE (6,26)                                                      1350
   26 FORMAT (9H1A MATRIX)
      CALL MPRNTV (D, M, 3, 6HA MAT  ,0,0,0,0,0)                        1370
 21   EK1 = EK - 1.0                                                    1380
      ENEK = EN - EK                                                    1390
      WRITE (6,61)EK1, ENEK                                             1400
   61 FORMAT (38H1F RATIOS FOR FOR INDIVIDUAL VARIABLES //
     1 33HONUMBER OF DEGREES OF FREEDOM ARE   F5.0, 1X 3HAND F6.0)      1420
      WRITE (6,63)                                                      1430
   63 FORMAT(57HOAMONG MEAN SQR    WITHIN MEAN SQR    F RATIO    VARIABL 1440
     1E)                                                                1450
      DO 65 I = 1,M                                                     1460
      AMS = D(I,I) / EK1                                                1470
      WMS = W(I,I) / ENEK
      X(I)=AMS /WMS
   65 WRITE (6,67)AMS, WMS, X(I),I,VNAME(1,I),VNAME(2,I)
 67   FORMAT(1X,F11.3,F19.3,F15.2, I8,2X, A8,1H-,A8)
C     VSTAT IS THE STEPWISE PROCEDURE.
      CALL VSTAT
      KRUNH1=KRUNCH
C     USES M.I.T. EIGEN ROUTINES
      CALL DIRNM
C     ROOTS OF W-A ARE NOW IN SD AND VECTORS ARE COLUMNS OFR            1530
C
      KLOBER=K-1
      IF(KRUNCH.LT.KLOBER)KLOBER=KRUNCH                                 1570
      TRACE = 0.0
      DO 35  I = 1, KRUNCH                                              1590
   35 TRACE = TRACE + SD(I)
      DO 36  I = 1, KRUNCH                                              1610
   36 XM(I) = 100.0 * (SD(I)   / TRACE)
C      ZERO ALL VECTORS
      DO 140 I=1,M
      DO 140 J=1,M
 140  SSD(I,J)=0.0
C    REPLACE ZEROS WITH VALID ELEMENTS IF THEY EXIST. DELETED VARS. HAVE
C    ZEROS REMAINING
      DO 141 I=1,KRUNCH
```

```
      IROW=ISUMT(I)
      DO 141 J=1,KRUNCH
 141  SSD(IROW,J)=R(I,J)
C     CONTINUE AS IN DISTRIBUTED VERSION                                1680
      WRITE (6,42)
 42   FORMAT(35H1DISCRIMINANT FUNCTIONS AS COLUMNS      )               1620
      WRITE (6,37)TRACE
   37 FORMAT (24HOTRACE OF W-INVERSE*A = E15.5)
      CALL MPRNTV (SSD,M,2,6HVECTOR,SD,5HROOTS,XM,6HPERCNT,KLOBER)
      DO 44  I = 1, M                                                   1720
      DO 44  J = 1, M                                                   1730
   44 TS(I,J) = SX(I) * SSD(I,J)                                        1740
C     TS(I,J) NOW CONTAINS THE SCALED VECTORS TO SHOW THE RELATIVE      1750
C     CONTRIBUTIONS OF THE VARIABLES TO A FUNCTION.                     1760
      WRITE (6,46)                                                      1770
   46 FORMAT (15H1SCALED VECTORS)                                       1780
      CALL MPRNTV (TS ,M,2,6HVECTOR,SD,5HROOTS,XM,6HPERCNT,KLOBER)
      IF(KRUNCH-K)47,48,48
   47 MD = KRUNCH
      GO TO 49                                                          1830
   48 MD = K - 1                                                        1840
 49   IF(LIN)50,50,50
   50 XLAMB = 1.0                                                       1860
      EM=KRUNCH                                                         1870
      DO 51  I = 1, MD                                                  1880
   51 XLAMB = XLAMB * (1.0/(1.0 + SD(I)))                               1890
      WRITE (6,52)XLAMB
   52 FORMAT (25H1LAMBDA FOR TEST OF H2 = E14.7)
      S=1.
      IF (  (EK-1.) .NE. 1. .AND. EM .NE. 1. )
     1S=SQRT(((EM**2)*((EK-1.0)**2)-4.0)/((EM**2)+((EK-1.0)**2)-5.0))
      Y = XLAMB**(1.0/S)                                                1920
      XM1 = (EN - 1.0) - ((EM + EK)/2.0)                                1930
      XL =-((EM * (EK-1.0)) - 2.0)/4.0                                  1940
      R1 = (EM * (EK-1.0))/2.0                                          1950
      F1 = 2.0 * R1                                                     1960
      F2 = (XM1 * S) +(2.0*XL)                                          1970
      F = ((1.0 - Y)/ Y) * (F2/F1)                                      1980
      WRITE (6,53)F1                                                    1990
   53 FORMAT (6HODF1= F14.7)
      WRITE (6,54)F2                                                    2010
   54 FORMAT (6HODF2= F14.7)
      WRITE (6,55)F                                                     2030
   55 FORMAT (21HOFOR TEST OF H2, F = E14.7)                            2050
C
 999  RETURN
      END
      SUBROUTINE EIDISC(XX)                                            XXXXXXX
C     PROGRAM EIDIS
C     THIS PROGRAM CONVERTED TO THE CDC6400 BY DON MORRISON
C ***********************************************************************
C   PROGRAM EIDISC NU C0068       JANUARY 25, 1967
C
C  PROGRAMMERS DONALD MORRISON AND RICHARD ART
C    THIS PROGRAM IS AVAILABLE THROUGH THE VOGELBACK COMPUTING CENTER OF
C NORTHWESTERN UNIVERSITY
C
C
C ***********************************************************************
C       EIDISC PROGRAM    PERFORMS A MULTIPLE DISCRIMINANT ANALYSIS
C     ON UP TO 50 VARIABLES AND TWENTY GROUPS.  AN UNGROUPED SET OF DATA
C     MAY FOLLOW THE GROUPS UPON WHICH THE DISCRIMINATION IS TO BE BASED
C
C                                                                       0110
C     INPUT.
C     THE FIRST CONTROL CARD IS PUNCHED AS FOLLOWS-
C     COLS. 1-2     K= NUMBER OF GROUPS ON WHICH THE ANALYSIS IS BASED.
C           3-4     M= THE NUMBER OF VARIABLES
C           5       JOBS=1 IF ANOTHER ANALYSIS FOLLOWS THIS ONE. 0 IF NOT
C           6       OPT1= 1   IF THE UNGROUPED SET FOLLOWS THE DISCRIM-
C                             INATING GROUPS. =0 IF NOT
C           7       OPT2 =1   IF A PLOT OF THE DISCRIMINANT SCORES FOR
C                             EACH GROUP IS DESIRED.  =0 IF NOT.
C           9       =1 IF THE W, A, T MATRICES FOR THE TOTAL GROUPS ARE PRINTED
C           10      =1 IF ALL MATRICES ARE TO BE PRINTED .
C           11-14   = CHI SQUARE WITH GROUPS MINUS ONE DEGREES OF
C                       FREEDOM TO LIMIT VARIABLES ENTERING ANALYSIS
C           15-17    =PERCENT OF TRACE OF DESIRED ROOTS GIVEN AS 98. ETC.
C           18-22    =NUMBER IN UNCLASSIFIED GROUP IF COL 6=1
C     THE SECOND TO N-TH CONTROL CARDS FOR THE ANALYSIS CONTIAN THE
C     VARIABLE NAMES  AT 6 COLUMNS APIECE 12 PER CARD AS NEEDED
```

```
C          THE FIRST CONTROL CARD FOR EACH GROUP IS DEFINED AS FOLLOWS.
C     COL.1-2  M = T = NUMBER OF VARIABLES (MAX. OF 50)
C     COL.3-7  NG = ENG = NUMBER OF SUBJECTS IN GROUP
C     COL.15-17  IPROB = PROBLEM NUMBER FOR IDENTIFYING PUNCHED OUTPUT.
C          THE SECOND CARD CONTAINS GROUP TITLE IN COLS. 1-72.
C     CARDS 3-5 CONTAIN THE VARIABLE FORMAT FOR DATA CARDS.  FIRST
C     SCORE CARD IS CARD NUMBER SIX.
C     5) INDIVIDUAL SCORE CARDS.  STUDENT ID IS THE FIRST VARIABLE IN
C     THE CARD, AND IS READ IN A FORMAT AT LEAST A1)
C     PREFACE THE  UNGROUPED SET OF DATA BY TWO FORMAT CARDS)
C     IF THIS GROUP IS NOT PRESENT  INSERT TWO BLANK CARDS NECESSARY
C
C          PRINTED OUTPUT INCLUDES
C     1) PROGRAM TITLE, K, M
C     2) FROM CORREL, FOR EACH GROUP, NUMBER OF GROUP, NUMBER IN GROUP,
C     GROUP MEANS, GROUP STANDARD DEVIATIONS, GROUP DISPERSION, GROUP
C     CORRELATION MATRIX IF REQUESTED
C     3) POOLED W MATRIX
C     4) T MATRIX
C     5) MEANS FOR TOTAL SAMPLE
C     6) STANDARD DEVIATIONS FOR TOTAL SAMPLE
C     7) A MATRIX
C     8) EIGENVALUES OF W-INVERSE * A
C     9) TRACE OF W-INVERSE * A , AND SUM OF EIGENVALUES
C     10) PERCENTAGE WHICH EACH EIGENVALUE IS OF TRACE
C     11) VECTORS OF W-INVERSE * A
C     12) SCALED VECTORS TO SHOW THE RELATIVE CONTRIBUTIONS OF THE
C     VARIABLES TO EACH FUNCTION
C     13) TOTAL SAMPLE SIZE, WILKS LAMBDA, F1, F2,  AND F FOR TEST OF H2
C     PRINTED OUTPUT IS APPROPRIATELY LABELED.
C          PRINTED OUTPUT INCLUDES
C     1) PROGRAM TITLE, KG, M, N
C     2) VECTORS OF WEIGHTS
C     3) ORIGINAL MEAN VECTORS FOR GROUPS
C     4) CENTROIDS OF GROUPS IN REDUCED SPACE
C     5) DISPERSIONS OF GROUPS IN REDUCED SPACE
C     PRINTED OUTPUT IS APPROPRIATELY LABELED.
C     3)THE CENTOURS OF THE CENTROIDS.
C     4)DISCRIMINANT SCORES, CHI SQUARES, AND PROBABILITIES.
C
      REAL*8 VNAME,TITLE                                                XXXXXXX
      COMMON/B/ NOBS(20),NVAR,NG                                         XXXXXX
      DIMENSION XX(NG,NVAR,1000)
      DIMENSION TITLE(10)
      COMMON L(8),XM(50),GN(50),DUMMY(5000)
      COMMON/L123/ LIN,K,M,ENTER,I,KRIT,KG,CNT,RNG,OPT2,OPT1,NTOT
      INTEGER OPT1,OPT2
      DIMENSION RNG(2,2)
      K=NG
      M=NVAR                                                            XXXXX
    1 READ (5,22,END=100) (TITLE(I),I=1,10)
   22 FORMAT (10A8)
      READ (5,1000,END=100)    OPT1,OPT2,LIN,L(7),L(8),ENTER,TROOT,NTOT   XXXX
 1101 LIN=0                                                             XXX
 1000 FORMAT (5X,5I1,F4.2,F3.0,I5)
      REWIND1
      REWIND4
      WRITE (6,1004) (TITLE(I),I=1,10)                                   ???
      CALL DISCRM(XX)
 1004 FORMAT (1H1,10A8)
      ENDFILE1
      ENDFILE4
      REWIND1
      REWIND4
      ROOT=0.0
      DO 20 I=1,M
      ROOT=ROOT+XM(I)
      IF(ROOT-TROOT)20,30,30
   20 CONTINUE
   30 KRIT=0
      WRITE(6,1003)ROOT,I                                               XXXXX
      CALL RCLAS2(XX)
   70 IF(OPT2)90,90,80
   80 CALL HSPLOT
   90 CONTINUE
  100 RETURN
 1003 FORMAT(19H1PERCENT OF TRACE= ,F7.3,6H USING,I4,24H DISCRIMINANT FU
     1NCTIONS.)
      END
      SUBROUTINE FFUNC(F,FVAL)
      COMMON/PARTS/PART5,XNU1,XNU2,PART6
```

```
C   DETERMINE THE NUMBER OF FUNCTIONS FOR 98 PER CENT DISCRIMINATION
C
        FVAL=EXP((ALOG(F)*PART5)-(ALOG(1.+(XNU1*F/XNU2))*PART6))
      RETURN
      END
      SUBROUTINE FORT(NPAR,NDATA,X,STANAM,PARNAM)
      REAL*8        STANAM,PARNAM
      INTEGER*4 ALLOC$,ALLOC1,ALLOC2
      COMMON/B/ NOBS(20),NVAR,NG
      DIMENSION STANAM(1),PARNAM(1)
      DIMENSION X(NDATA,NPAR)
      EXTERNAL DATA
    5 CALL SETUP(NPAR,NDATA,X,STANAM,PARNAM)
      ISIZE=NG*NVAR*1000
      ILOC=ALLOC2(IR,ISIZE)
      IF ( IR .LE. C)  GO TO 999
      CALL ALLOCP(DATA,ILOC,ISIZE)
      CALL FREE2(ILOC,ISIZE)
      GO TO 5
  999 WRITE (6,2000)
 2000 FORMAT ('1 UNABLE TO ALLOCATE ARRAY')
      GO TO 5
      END
      FUNCTION GAUSFQ(F,L,NU1,NU2)
      EXTERNAL FFUNC
      COMMON/PARTS/PART5,XNU1,XNU2,PART6
       XNU1=NU1
      XNU2=NU2
      PART5=.5*(XNU1-2.0)
      PART6=.5*(XNU1+XNU2)
C
      CALL INTEG1(FFUNC,.00001,F,.001,.1,.02,VALU)
C
      PART1=ALGAMA(.5*(XNU1+XNU2))
      PART2=ALGAMA(.5*XNU1)
      PART3=ALGAMA(.5*XNU2)
       PART4=((XNU1/XNU2)**(.5*XNU1))
C
      FIRST=EXP(PART1-PART2-PART3)
      FRONT=FIRST*PART4
      GAUSFQ=FRONT*VALU
       RETURN
      END
      SUBROUTINE   HDIAG (H,N,IEGEN,U,NR)
C4F05.0EIGENVALUES AND EIGENVECTORS OF A REAL SYMMETRIC MATRIX BY THE
C     JACOBI METHOD. (HDIAG).  PROGRAMMED BY F. J. CORBATO AND M. MERWIN
C     OF THE M. I. T. COMPUTATION CENTER.
C     SUBROUTINE HDIAG.
C
C          THIS SUBROUTINE COMPUTES THE EIGENVALUES AND EIGENVECTORS
C     OF A REAL SYMMETRIC MATRIX, H, OF ORDER N ( WHERE N MUST BE LESS
C     THAN 51), AND PLACES THE EIGENVALUES IN THE DIAGONAL ELEMENTS OF
C     THE MATRIX H, AND PLACES THE EIGENVECTORS (NORMALIZED) IN THE
C     COLUMNS OF THE MATRIX U.  IEGEN IS SET AS 1 IF ONLY EIGENVALUES
C     ARE DESIRED, AND IS SET TO 0 WHEN VECTORS ARE REQUIRED.  NR CON-
C     TAINS THE NUMBER OF ROTATIONS DONE.
C
C     H, N, IENGEN, U, AND NR OF THE ARGUMENT LIST ARE DUMMY VARIABLES
C     AND MAY BE NAMED DIFFERENTLY IN THE CALLING OF THE SUBROUTINE.
C      REM 0001 SYM. CARDS DIST.     731 RCV. 08-13-59CORRECTS DIST 705F2
C      REM 0186 SYM. CARDS DIST.     705 RCV. 07-14-59FORTRAN PROGRAM  F2
C     MIHDI3, FORTRAN II DIAGONALIZATION OF A REAL SYMMETRIC MATRIX BY
C        THE JACOBI METHOD.
C     MAY 19, 1959
C     CALLING SEQUENCE FOR DIAGONALIZATION
C           CALL  HDIAG( H, N, IEGEN, U, NR)
C              WHERE H IS THE ARRAY TO BE DIAGONALIZED.
C     N IS THE ORDER OF THE MATRIX, H.
C
C     IEGEN MUST BE SET UNEQUAL TO ZERO IF ONLY EIGENVALUES ARE
C              TO BE COMPUTED.
C     IEGEN MUST BE SET EQUAL TO ZERO IF EIGENVALUES AND EIGENVECTORS
C              ARE TO BE COMPUTED.
C
C     U IS THE UNITARY MATRIX USED FOR FORMATION OF THE EIGENVECTORS.
C
C     NR IS THE NUMBER OF ROTATIONS.
C
C     A DIMENSION STATEMENT MUST BE INSERTED IN THE SUBROUTINE.
C     DIMENSION H(N,N), U(N,N), X(N), IQ(N)
C
```

```
C     SUBROUTINE PLACES COMPUTER IN THE FLOATING TRAP MODE
C
C     THE SUBROUTINE OPERATES ONLY ON THE ELEMENTS OF H THAT ARE TO THE
C             RIGHT OF THE MAIN DIAGONAL.   THUS, ONLY A TRIANGULAR
C             SECTION NEED BE STORED IN THE ARRAY H.
C
C
C
C
      DIMENSION H(50,50),U(50,50),X(50),IQ(50)
      IF (IEGEN) 15,10,15
 10   DO 14 I=1,N
      DO 14 J=1,N
      IF(I-J)12,11,12
 11   U(I,J)=1.0
      GO TO 14
 12   U(I,J)=0.0
 14   CONTINUE
C
 15   NR = 0
      IF (N-1) 1000,1000,17
C
C     SCAN FOR LARGEST OFF DIAGONAL ELEMENT IN EACH ROW
C     X(I) CONTAINS LARGEST ELEMENT IN ITH ROW
C     IQ(I) HOLDS SECOND SUBSCRIPT DEFINING POSITION OF ELEMENT
C
 17   NMI1=N-1
      DO 30 I=1,NMI1
      X(I) = 0.0
      IPL1=I+1
      DO 30 J=IPL1,N
      IF ( X(I) - ABS( H(I,J))) 20,20,30
 20   X(I)=ABS(H(I,J))
      IQ(I)=J
 30   CONTINUE
C
C     SET INDICATOR FOR SHUT-OFF.RAP=2**-27,NR=NO. OF ROTATIONS
      RAP=.745058059E-08
      HDTEST=1.0E38
C
C     FIND MAXIMUM OF X(I) S FOR PIVOT ELEMENT AND
C     TEST FOR END OF PROBLEM
C
 40   DO  70  I=1,NMI1
      IF (I-1) 60,60,45
 45   IF ( XMAX- X(I)) 60,70,70
 60   XMAX=X(I)
      IPIV=I
      JPIV=IQ(I)
 70   CONTINUE
C
C     IS MAX. X(I) EQUAL TO ZERO, IF LESS THAN HDTEST, REVISE HDTEST
      IF ( XMAX) 1000,1000,80
 80   IF (HDTEST) 90,90,85
 85   IF (XMAX - HDTEST) 90,90,148
 90   HDIMIN = ABS( H(1,1) )
      DO 110  I= 2,N
      IF (HDIMIN- ABS( H(I,I))) 110,110,100
 100  HDIMIN=ABS(H(I,I))
 110  CONTINUE
C
      HDTEST=HDIMIN*RAP
C
C     RETURN IF MAX.H(I,J)LESS THAN(2**-27)ABSF(H(K,K)-MIN)
      IF (HDTEST- XMAX) 148,1000,1000
 148  NR = NR+1
C
C     COMPUTE TANGENT, SINE AND COSINE,H(I,I),H(J,J)
 150  TANG=SIGN(2.0,(H(IPIV,IPIV)-H(JPIV,JPIV)))*H(IPIV,JPIV)/(ABS(H(IPI
     1V,IPIV)-H(JPIV,JPIV))+SQRT((H(IPIV,IPIV)-H(JPIV,JPIV))**2+4.0*H(IP
     2IV,JPIV)**2))
      COSINE=1.0/SQRT(1.0+TANG**2)
      SINE=TANG*COSINE
      HII=H(IPIV,IPIV)
      H(IPIV,IPIV)=COSINE**2*(HII+TANG*(2.0*H(IPIV,JPIV)+TANG*H(JPIV,JPI
     1V)))
      H(JPIV,JPIV)=COSINE**2*(H(JPIV,JPIV)-TANG*(2.0*H(IPIV,JPIV)-TANG*H
     1 II))
      H(IPIV,JPIV)=0.0
C
C       PSEUDO RANK THE EIGENVALUES
```

```
C       ADJUST SINE AND COS FOR COMPUTATION OF H(IK) AND U(IK)
        IF ( H(IPIV,IPIV) -  H(JPIV,JPIV)) 152,153,153
  152   HTEMP = H(IPIV,IPIV)
        H(IPIV,IPIV) = H(JPIV,JPIV)
        H(JPIV,JPIV) = HTEMP
C         RECOMPUTE SINE AND COS
        HTEMP = SIGN(1.0, -SINE) * COSINE
        COSINE = ABS(SINE)
        SINE = HTEMP
  153   CONTINUE
C
C       INSPECT THE IQS BETWEEN I+1 AND N-1 TO DETERMINE
C       WHETHER A NEW MAXIMUM VALUE SHOULD BE COMPUTED SINCE
C       THE PRESENT MAXIMUM IS IN THE I OR J ROW.
C
        DO 350 I=1,NM11
        IF(I-IPIV)210,350,200
  200   IF(I-JPIV)210,350,210
  210   IF(IQ(I)-IPIV)230,240,230
  230   IF(IQ(I)-JPIV)350,240,350
  240   K=IQ(I)
  250   HTEMP=H(I,K)
        H(I,K)=0.0
        IPL1=I+1
        X(I) =0.0
C
C       SEARCH IN DEPLETED ROW FOR NEW MAXIMUM
C
        DO 320 J=IPL1,N
        IF ( X(I)- ABS( H(I,J)) ) 300,300,320
  300   X(I) = ABS(H(I,J))
        IQ(I)=J
  320   CONTINUE
        H(I,K)=HTEMP
  350   CONTINUE
C
        X(IPIV) =0.0
        X(JPIV) =0.0
C
C       CHANGE THE OTHER ELEMENTS OF H
C
        DO 530 I=1,N
C
        IF(I-IPIV)370,530,420
  370   HTEMP = H(I,IPIV)
        H(I,IPIV) = COSINE*HTEMP + SINE*H(I,JPIV)
        IF ( X(I) -  ABS( H(I,IPIV)) )380,390,390
  380   X(I) = ABS(H(I,IPIV))
        IQ(I) = IPIV
  390   H(I,JPIV) = -SINE*HTEMP + COSINE*H(I,JPIV)
        IF ( X(I) -  ABS( H(I,JPIV)) ) 400,530,530
  400   X(I) = ABS(H(I,JPIV))
        IQ(I) = JPIV
        GO TO 530
C
  420   IF(I-JPIV)430,530,480
  430   HTEMP = H(IPIV,I)
        H(IPIV,I) = COSINE*HTEMP + SINE*H(I,JPIV)
        IF ( X(IPIV) -  ABS( H(IPIV,I)) ) 440,450,450
  440   X(IPIV) = ABS(H(IPIV,I))
        IQ(IPIV) = I
  450   H(I,JPIV) = -SINE*HTEMP + COSINE*H(I,JPIV)
        IF ( X(I) -  ABS( H(I,JPIV)) ) 400,530,530
C
  480   HTEMP = H(IPIV,I)
        H(IPIV,I) = COSINE*HTEMP + SINE*H(JPIV,I)
        IF ( X(IPIV) -  ABS( H(IPIV,I)) ) 490,500,500
  490   X(IPIV) = ABS(H(IPIV,I))
        IQ(IPIV) = I
  500   H(JPIV,I) = -SINE*HTEMP + COSINE*H(JPIV,I)
        IF ( X(JPIV) -  ABS( H(JPIV,I)) ) 510,530,530
  510   X(JPIV) = ABS(H(JPIV,I))
        IQ(JPIV) = I
  530   CONTINUE
C
C       TEST FOR COMPUTATION OF EIGENVECTORS
C
        IF(IEGEN)40,540,40
  540   DO 550 I=1,N
        HTEMP=U(I,IPIV)
        U(I,IPIV)=COSINE*HTEMP+SINE*U(I,JPIV)
```

```
550   U(I,JPIV)=-SINE*HTEMP+COSINE*U(I,JPIV)                          2000
      GO TO 40                                                        2010
1000  RETURN                                                          2020
      END
      SUBROUTINEINTEG1(FFUNC,XL,XU,DXINT,EPS,AMAG,VALU)                  2
C                                                                        2
C      THIS PROGRAM HAS BEEN TRANSLATED FOR THE     360/50              2
C      WITH RELEASE 1-A OF THE MOD-50 TRANSDECK                         2
C                                                          JDB           2
C                                                                        2
C     ROUTINE TO EVALUATE INTEGRAL OF F(X) BETWEEN                       2
C     THE LIMITS XL AND XU BY VARIABLE SIMPSONS                          2
C     RULE.                                                              2
C                                                                        2
      VALU=0.                                                            3
      XZERO=XL                                                           4
      DX=DXINT                                                           5
      CALLFFUNC(XZERO,FZERO)                                             6
   10 CONTINUE                                                           7
      IF(XU-(XZERO+4.*DX))20,30,30                                       8
   20 CONTINUE                                                           9
      DX=(XU-XZERO)/4.                                                  10
   30 CONTINUE                                                          11
      CALLFFUNC(XZERO+DX,F1)                                            12
      CALLFFUNC(XZERO+2.*DX,F2)                                         13
      CALLFFUNC(XZERO+3.*DX,F3)                                         14
      CALLFFUNC(XZERO+4.*DX,F4)                                         15
      S1=2.*DX*(FZERO+4.*F2+F4)/3.                                      16
      S2=DX*(FZERO+4.*F1+2.*F2+4.*F3+F4)/3.                             17
      RATIO=ABS(S2-S1)/(AMAX1(ABS(S2),AMAG*1.E-4)*EPS)
      IF(RATIO-1.)60,40,40                                              19
   40 CONTINUE                                                          20
C                                                                       20
C     CYCLE REJECTED. REDUCE DX AND TRY AGAIN                           20
C                                                                       20
      DX=DX/1.5                                                         21
      GOTO30                                                            22
   60 CONTINUE                                                          23
C                                                                       23
C     CYCLE ACCEPTED.                                                   23
C                                                                       23
      ADD=S2+(S2-S1)/15.                                                24
      VALU=VALU+ADD                                                     25
      XZERO=XZERO+4.*DX                                                 26
      FZERO=F4                                                          27
      IF(XZERO-XU)80,70,70                                              28
   70 CONTINUE                                                          29
      RETURN                                                            30
   80 CONTINUE                                                          31
      IF(RATIO-.5)100,90,90                                             32
   90 CONTINUE                                                          33
      DX=DX/1.5                                                         34
      GOTO10                                                            35
  100 CONTINUE                                                          36
      IF(RATIO-.01)110,10,10                                            37
  110 DX=DX*1.5                                                         38
      GOTO10                                                            39
      END                                                               41
      SUBROUTINE MATINV(A,N,B,M,DETERM)
CMATINV
C     SUBROUTINE MATINV.
C
C          PROGRAMMED BY BURTON S. GARBOW, ARGONNE NATIONAL LABORATORY,
C     AND REPORTED IN IBM 704-709 SHARE LIBRARY AS AN F402.
C          THIS SUBROUTINE COMPUTES THE INVERSE AND DETERMINANT OF
C     MATRIX A, OF ORDER N, BY THE GAUSS-JORDAN METHOD. A-INVERSE
C     REPLACES A, AND THE DETERMINANT OF A IS PLACED IN DETERM. IF
C     M = 1 THE VECTOR B CONTAINS THE CONSTANT VECTOR WHEN MATINV IS
C     CALLED, AND THIS IS REPLACED WITH THE SOLUTION VECTOR.  IF M = 0,
C     NO SIMULTANEOUS EQUATION SOLUTIONS ARE CALLED FOR, AND B IS NOT
C     PERTINENT. N IS NOT TO EXCEED 50.
C     A, N, B, M, AND DETERM IN THE ARGUMENT LIST ARE DUMMY VARIABLES.
C
C
      DIMENSION IPIVOT(50), A(50,50), B(50,1), INDEX(50,2), PIVOT(50)
      EQUIVALENCE (IROW,JROW), (ICOLUM,JCOLUM), (AMAX, T, SWAP)
C     INITIALIZATION
   10 DETERM=1.0
   15 DO 20 J=1,N
   20 IPIVOT(J)=0
   30 DO 550 I=1,N
```

```
C       SEARCH FOR PIVOT ELEMENT
   40   AMAX=0.0
   45   DO 105 J=1,N
   50   IF (IPIVOT(J)-1) 60, 105, 60
   60   DO 100 K=1,N
   70   IF (IPIVOT(K)-1) 80, 100, 740
   80   IF (ABS (AMAX)-ABS (A(J,K))) 85, 100, 100
   85   IROW=J
   90   ICOLUM=K
   95   AMAX=A(J,K)
  100   CONTINUE
  105   CONTINUE
  110   IPIVOT(ICOLUM)=IPIVOT(ICOLUM)+1
C       INTERCHANGE ROWS TO PUT PIVOT ELEMENT ON DIAGONAL
  130   IF (IROW-ICOLUM) 140, 260, 140
  140   DETERM=-DETERM
  150   DO 200 L=1,N
  160   SWAP=A(IROW,L)
  170   A(IROW,L)=A(ICOLUM,L)
  200   A(ICOLUM,L)=SWAP
  205   IF(M) 260, 260, 210
  210   DO 250 L=1, M
  220   SWAP=B(IROW,L)
  230   B(IROW,L)=B(ICOLUM,L)
  250   B(ICOLUM,L)=SWAP
  260   INDEX(I,1)=IROW
  270   INDEX(I,2)=ICOLUM
  310   PIVOT(I)=A(ICOLUM,ICOLUM)
  320   DETERM=DETERM*PIVOT(I)
C       DIVIDE PIVOT ROW BY PIVOT ELEMENT
  330   A(ICOLUM,ICOLUM)=1.0
  340   DO 350 L=1,N
  350   A(ICOLUM,L)=A(ICOLUM,L)/PIVOT(I)
  355   IF(M) 380, 380, 360
  360   DO 370 L=1,M
  370   B(ICOLUM,L)=B(ICOLUM,L)/PIVOT(I)
C       REDUCE NON-PIVOT ROWS
  380   DO 550 L1=1,N
  390   IF(L1-ICOLUM) 400, 550, 400
  400   T=A(L1,ICOLUM)
  420   A(L1,ICOLUM)=0.0
  430   DO 450 L=1,N
  450   A(L1,L)=A(L1,L)-A(ICOLUM,L)*T
  455   IF(M) 550, 550, 460
  460   DO 500 L=1,M
  500   B(L1,L)=B(L1,L)-B(ICOLUM,L)*T
  550   CONTINUE
C       INTERCHANGE COLUMNS
  600   DO 710 I=1,N
  610   L=N+1-I
  620   IF (INDEX(L,1)-INDEX(L,2)) 630, 710, 630
  630   JROW=INDEX(L,1)
  640   JCOLUM=INDEX(L,2)
  650   DO 705 K=1,N
  660   SWAP=A(K,JROW)
  670   A(K,JROW)=A(K,JCOLUM)
  700   A(K,JCOLUM)=SWAP
  705   CONTINUE
  710   CONTINUE
  740   RETURN
        END
        SUBROUTINE MPRNTV(R,M,L, TI, V1, TJ, V2, TK,NOLO)
C       SUBPROGRAM MATRIX PRINTOUT AND VECTORS
C
C       R IS THE SQUARE MATRIX OF ORDER M TO BE PRINTED
C       TI IS A SIX LETTER DESIGNATE OF THE MATRIX
C       V1 IS A VECTOR TO BE PRINTED UNDER THE MATRIX
C       TJ IS A SIX LETTER DESIGNATE OF THE VECTOR
C       V2 IS ANOTHER SUCH VECTOR(EACH MAX. OF 50)
C       TK IS THE DESIGNATE
C       FORMAT OF THE PRINTOUT IS 10F11.2
        REAL*8 TI,TJ,TK,VNAME
        DIMENSION R(50,50), J(50), V1(50), V2(50)
        COMMON/NAMES/VNAME(2,50)
        J1 = 0
        J2 = 0
        JSEC = 0
      9 J1 = J2 + 1
        J2 = J1 + 9
        IF ( J2-M)  13,13,12
     12 J2 = M
```

```
   13 JSEC = JSEC + 1                                                   0230
      IF((L.EQ.2).AND.(J2.GT.NOLO))J2=NOLO
      IF ( JSEC-1) 18,18, 19                                            0240
   18 WRITE (6,17)TI, JSEC                                              0250
   17 FORMAT(1H0, A6, 9H  SECTIONI3)                                    0260
      GO TO 201                                                         0270
   19 WRITE (6,20)TI, JSEC                                              0280
   20 FORMAT(1H1, A6 , 9H  SECTIONI3)                                   0290
  201 IF(L.EQ.2) GO TO 21
      WRITE (6,22) (VNAME(1,I),I=J1,J2)
   22 FORMAT (6H0  VAR,19X,10(A8,3X))
      WRITE (6,23) (VNAME(2,I),I=J1,J2)
   23 FORMAT (25X,10(A8,3X))
      IF(L.EQ.3) GO TO 27
   21 WRITE (6,26)TJ , (V1(K), K= J1,J2)
   26 FORMAT(1H , A6,15X,1P10E11.4)
      WRITE (6,26)TK, (V2(K), K= J1,J2)
   27 WRITE (6,33)                                                      0370
      DO 29  I=1,M
   29 WRITE (6,30) VNAME(1,I),VNAME(2,I),I,(R(I,K),K=J1,J2)
   30 FORMAT (1H ,A8,1H-,A8,1X,I2,1X,1P10E11.4)
   31 IF((L.EQ.2).AND.(J2.EQ.NOLO))RETURN                               0400
      IF ( J2- M) 9,32,32
   33 FORMAT (1H0)                                                      0410
   32 RETURN
      END
      SUBROUTINE ORDER(A,M,KK)
      DIMENSION A(50),KK(50)
      II=1
      BIG=10.**27
      LITTLE=1
    1 IBIG=LITTLE
      DO 5 I=1,M
      IF((A(I).GT.A(IBIG)).AND.(A(I).LT.BIG)) IBIG=I
      IF(A(I).LT.A(LITTLE))LITTLE=I
    5 CONTINUE
      DO 10 I=1,M
      IF(A(I).NE.A(IBIG))GO TO 10
      KK(II)=I
      II=II+1
      IF(II.EQ.(M+1)) GO TO 30
   10 CONTINUE
      BIG=A(IBIG)
      GO TO 1
   30 RETURN
      END
      SUBROUTINE PRINT(XX)
      REAL*8 VNAME
      COMMON/B/ NOBS(20),NVAR,NG
      COMMON/NAMES/VNAME(2,50)
      DIMENSION XX(NG,NVAR,1000)
      IR=45
      IC=10
      DO 20 K=1,NG
      IL=-IC+1
    5 IL=IL+IC
      IU=NVAR
      IF((IU-IL) .GE. IC) IU=IL+IC-1
      JL=-IR+1
   10 JL=JL+IR
      JU=NOBS(K)
      IF ((JU-JL).GE.IR) JU=JL+IR-1
      WRITE (6,1000) K,(VNAME(1,I),I=IL,IU)
      WRITE (6,1001)    (VNAME(2,I),I=IL,IU)
      WRITE (6,1003)
      DO 15 J=JL,JU
   15 WRITE (6,1002) J,(XX(K,I,J),I=IL,IU)
      IF(JU.LT.NOBS(K)) GO TO 10
      IF(IU.LT. NVAR ) GO TO 5
   20 CONTINUE
 1000 FORMAT (6H1GROUP,I3,5H DATA//3X,5HCOUNT,10(4X,A8))
 1001 FORMAT(8X,10(4X,A8))
 1002 FORMAT (I8,1P10E12.3)
 1003 FORMAT (1X)
      RETURN                                                            XXX
      END
      SUBROUTINE RCLAS2(Y)                                              XXXXXXX
      INTEGER OPT1,OPT2                                                    XXX
      COMMON/B/ NOBS(20),NVAR,NG
      DIMENSION  Y(NG,NVAR,1000)
      DIMENSION IRW1(20,20),IRW2(20,20)
```

```
      DIMENSION FMT(36),RNG(2,2),RATIO2(50),P12(50),PROB2(50)
      COMMON L(8),B(50),GN(50),XBAR(50,50),V(50,50)
     1/RCL2/ DIF(20),CHI(20),CHISQ(20),P1(20),PROB(20),DIS(50,50),
     2        CENT(20,20),DG(50,50),D(20,20,20),RATIO(20),X(50),DISC(20)
      COMMON/L123/LIN,KG,M,ENTER,N,KRIT,MG,CNT,RNG,OPT2,OPT1,NTOT
      DATA  ADD/6H000001/, ADD10/6H000010/
      DATA  DATAID/6HGRP.01/
      MGG=0
      ICNT=CNT                                                          XXX
      IF(KRIT)300,1000,235
 1000 WRITE (6,1)                                                       0630
    1 FORMAT(43HOCENTROIDS AND DISPERSIONS IN REDUCED SPACE)            0640
      WRITE (6,3)KG                                                     0690
    3 FORMAT (17HONO. OF GROUPS = I2)                                   0700
      WRITE (6,4)M                                                      0710
    4 FORMAT (25HONO. OF TEST VARIABLES = I2)                           0720
      WRITE (6,5)N                                                      0730
    5 FORMAT (30HONO. OF LINEAR COMBINATIONS = I2)                      0740
   10 FORMAT (1P8E15.8)
      DO 14  I = 1, KG                                                  0900
      DO 14  J = 1, N                                                   0910
      CENT(I,J) = 0.0                                                   0920
      DO 14  K = 1, M                                                   0930
   14 CENT(I,J) = CENT(I,J) + XBAR(I,K) * V(K,J)                        0940
      WRITE (6,15)                                                      0950
   15 FORMAT(47HOCENTROIDS OF GROUPS IN REDUCED SPACE, ROW-WISE)        0970
      DO 16   I = 1, KG                                                 0980
16    WRITE(6,12) I,(CENT(I,J),J=1,N)
   12 FORMAT (6HOGROUP,I3,(1P8E15.8))
      DO 234 LL= 1, KG
      WRITE (6,18)LL                                                    1020
   18 FORMAT (39HODISPERSION IN REDUCED SPACE FOR GROUP I2)             1030
      DO 19  I = 1, M                                                   1040
19    READ(4)(DG(I,J),J=1,M)
      DO 20  I = 1, N                                                   1060
      DO 20  J = 1,  M                                                  1070
      DIS (I,J) = 0.0                                                   1080
      DO 20  K = 1,  M                                                  1090
   20 DIS (I,J) = DIS (I,J) + V(K,I)* DG(K,J)                           1100
      DO 21  I = 1, N                                                   1110
      DO 21  J = 1, N                                                   1120
      DG(I,J) = 0.0                                                     1130
      DO 21  K = 1, M                                                   1140
   21 DG(I,J) =  DG(I,J) + DIS(I,K)* V(K,J)                             1150
      DO 23  I = 1, N                                                   1170
   23 WRITE (6,10)(DG(I,J), J=1,N)                                      1180
      CALL MATINV(DG, N, B, 0, DETERM)                                  1190
      DO 232  I=1,N                                                     1200
      DO 232 J= 1,N                                                     1210
  232 D(I,J,LL)= DG(I,J)
      PRINT 1107,DETERM
 1107 FORMAT(1H ,E20.7)
      RATIO2(LL)=1./SQRT(ABS(DETERM))
  234 RATIO (LL)=GN(LL)/SQRT(ABS(DETERM))
      REWIND 4
      DO 401 J=1,2
      RNG(J,1)=10.**35
401   RNG(J,2)=-RNG(J,1)
      KK = 1                                                            1260
      XID=DATAID
  211 DO 212 I = 1,N                                                    1270
  212 DISC(I) = CENT(KK,I)                                              1280
      WRITE(6,99)KK
99    FORMAT(57HOCHI SQUARES AND GROUP MEMBERSHIP PROBABILITIES FOR GROU
     1P    , I3,1X,8HCENTROID    )
      GO TO 25                                                          1290
  300 ISUB=ICNT-CNT+1
      DO 305 JJJ=1,M
  305 X(JJJ)=Y(MGG,JJJ,ISUB)
      XID=ISUB
      GO TO 237
  235 READ (5,7001) XID,(X(I),I=1,M)
 7001 FORMAT (10E20.6)
  237 DO 24 J=1,N                                                       1310
      DISC(J) = 0.0                                                     1320
      DO 24 I=1,M                                                       1330
   24 DISC(J) = DISC(J) +  (X(I)*V(I,J))                                1340
      NZ=1
      WRITE (6,7022) NZ,XID,(DISC(J),J=1,N)
   25 DO 32 K=1, KG
      DO 28  I = 1, N                                                   1370
```

```
 28 DIF(I) = DISC(I) - CENT(K,I)                                        1380
    DO 30 J = 1,N                                                       1390
    CHI(J) = 0.0                                                        1400
    DO 30 I = 1,N                                                       1410
 30 CHI(J) = CHI(J)+(DIF(I)*D(I,J,K))                                   1420
    CHISQ(K) = 0.0                                                      1430
    DO 31 I = 1,N                                                       1440
 31 CHISQ(K) = CHISQ(K)+(DIF(I)*CHI(I))                                 1450
    IF(CHISQ(K).LT.0.0) CHISQ(K)=0.0
 32 IF(CHISQ(K).GT.1400.0) CHISQ(K)=1400.0
    NZ=2
    WRITE (6,7023) NZ,(CHISQ(K),K=1,KG)
 33 P2 = 0.0
    P3=0.0
    DO 34 K = 1,KG                                                      1480
    P1(K) = RATIO (K) * EXP(-CHISQ(K)/2.0)                              1490
    P12(K)=RATIO2(K)*EXP(-CHISQ(K)/2.0)
    P3=P3+P12(K)                                                        1500
 34 P2 = P2 + P1(K)
    PBMAX1=0.
    PBMAX2=0.
    DO 36 K=1,KG
    PROB2(K)=P12(K)/P3
    PROB(K)=P1(K)/P2
    IF ( PBMAX1 .GE. PROB(K)) GO TO 7002
    PBMAX1=PROB(K)
    MAXGP1=K
7002 IF (PBMAX2 .GE. PROB2(K) ) GO TO 36
    PBMAX2=PROB2(K)
    MAXGP2=K
 36 CONTINUE
    IF (MGG .LE. 0 ) GO TO 7013
    IRW1(MGG,MAXGP1)=IRW1(MGG,MAXGP1)+1
    IRW2(MGG,MAXGP2)=IRW2(MGG,MAXGP2)+1
7013 NZ=3
    WRITE (6,7021) NZ,MAXGP1,PBMAX1,(PROB(K),K=1,KG)
    NZ=NZ+1
    WRITE (6,7021) NZ,MAXGP2,PBMAX2,(PROB2(K),K=1,KG)
7021 FORMAT (I2,I9,7F15.4/(26X,6F15.4))
7022 FORMAT (I2,2X,F4.0,18X,6F15.4/(26X,6F15.4))
7023 FORMAT (I2,24X,6F15.4/(26X,6F15.4))
    IF(KRIT+1)37,38,38
 37 WRITE(4) XID,PROB(MGG),(DISC(IJ),IJ=1,2)
    DO 400 J=1,2
    RNG(J,1)=AMIN1(RNG(J,1),DISC(J))
400 RNG(J,2)=AMAX1(RNG(J,2),DISC(J))
38  IF (KG-KK) 500,500 , 39                                             1550
 39 KK = KK + 1
    XID=XID+ADD
    IF(KK.EQ.10) XID=DATAID+ADD10                                       1560
    GO TO 211
500 CNT=CNT-1.0
    IF(KRIT)501,502,503
501 IF(CNT)510,510,300
 502 KRIT=-2
 510 MGG=MGG +1
    IF ( MGG .GT. 1 ) GO TO 7004
    DO 7005 K=1,KG
    DO 7005 KZ=1,KG
    IRW1(K,KZ)=0.
7005 IRW2(K,KZ)=0.
7004 IF ( MGG .LE. KG) GO TO 7007
    WRITE (6,7008) (K,K=1,KG)
    DO 7006 K=1,KG
7006 WRITE (6,7009) K,(IRW1(K,KZ),KZ=1,KG)
    WRITE (6,7010) (K,K=1,KG)
    DO 7011 K=1,KG
7011 WRITE (6,7009) K,(IRW2(K,KZ),KZ=1,KG)
    GO TO 1059
7007 CONTINUE
7009 FORMAT (/I6,3X,18I6/(9X,18I6))
7008 FORMAT (1H1,12X,23HCLASSIFICATION MATRIX 1/9H   GROUP ,18I6/
    1 (9X,18I6))
7010 FORMAT(////13X,23HCLASSIFICATION MATRIX 2/9H   GROUP ,18I6/(9X,
    1     18I6))
    CNT=GN(MGG)
    ICNT=CNT
    WRITE(6,1001)
    WRITE(6,1002)MGG,KG
1001 FORMAT(67H1DISCRIMINANT SCORES, CHI SQUARES, AND MEMBERSHIP PROBAB
    1ILITIES FOR)
```

```
 1002 FORMAT(7H GROUP ,I2,4H OF ,I2)
      GO TO 300
 1059 REWIND 1
      REWIND 4
      IF(OPT1.LE.0) RETURN
      KRIT=1
      CNT=NTOT
      WRITE(6,1001)
      WRITE (6,1005)
 1005 FORMAT(19H UNCLASSIFIED GROUP)
      KK= 99
      GO TO 235
  503 IF(CNT.GT.0) GO TO 235
      RETURN
      END
      SUBROUTINE SPUNCH (X, M, XID, N)                                  0150
C     SUBROUTINE SPUNCH                                                 0010
C          THIS SUBROUTINE IS USED FOR PUNCHING OUT SCORE VECTORS,      0020
C     TOGETHER WITH INDIVIDUAL IDENTIFICATION CODE.                     0030
C     DEFINITIONS OF SUBROUTINE ARGUMENTS.                              0040
C          X = VECTOR TO BE PUNCHED                                     0050
C          M = NUMBER OF ELEMENTS IN VECTOR                             0060
C          XID = INDIVIDUAL IDENTIFICATION NUMBER (7 DIGIT MAXIMUM)     0070
C          N = ONE INTEGER INDICATING VECTOR TYPE                       0080
C     OUTPUT.                                                           0090
C          COL. 1 = N                                                   0100
C          COL. 2-9 = XID IN F8.0                                       0110
C          COL. 10 = SECTION NUMBER OF CARD (7 ELEMENTS/CARD)           0120
C          COL. 11-80 = DISCRIMINANT SCORES, CLASSIFICATION CHI SQUARES, 0130
C     GROUP MEMBERSHIP PROBABILITIES, FACTOR SCORES, ETC. IN 7F10.4     0140
      DIMENSION X(20)                                                   0160
      J1 = 0                                                            0180
      J2 = 0                                                            0190
      JSEC = 0                                                          0200
   11 J1 = J2 + 1                                                       0210
      J2 = J1 + 6                                                       0220
      IF (J2 - M) 16, 14, 14                                            0230
   14 J2 = M                                                            0240
   16 JSEC = JSEC + 1                                                   0250
      WRITE (6,18)N, XID, JSEC, (X(J), J = J1, J2)
   18 FORMAT(1H ,I1,2X,F4.0,2X,I1,7F15.4)
      IF (J2 - M) 11, 20, 20                                            0280
   20 RETURN                                                            0290
      END
      SUBROUTINE VSTAT
      REAL*8 VNAME
      COMMON /NAMES/VNAME(2,50)
      COMMON L(8),XM(50),GN(50),XBAR(50,50),SSD(50,50)
     1/DSC/X(50),SX(50),W(50,50),ISUMT(50)
      COMMON/L3/SS,TS,R,VS,KRUNH1
      DIMENSION SS(50,50),TS(50,50),R(50,50),VS(50)
      COMMON/L1/T,NG,ENEK,EK1,M,KRUNCH,ENTER1
      DIMENSION  WORK(50),JJ(50),KK(50),D(50,50)
      EQUIVALENCE(WORK,JJ),(SSD,D)
      ENTER=ENTER1
      KRSW=0
      K1=EK1
      ENK=ENEK
      NK=ENK
      CALL ORDER(X,M,KK)
      K=KK(1)
      TS(1,1)=ENK/W(K,K)
      SS(1,1)=D(K,K)
      ICNT=1
      IREM=M
      WRITE(6,100)
      WRITE(6,101)
      WRITE(6,102)
      V=TS(1,1)*SS(1,1)
      GO=GAUSFQ(X(K),1,K1,NK)
      GV=GAUSFQ(V/EK1,3,K1,120)
      GV=1.0-GV
      GVD=GV
      VDIFF=V
      WRITE (6,103) K,VNAME(1,K),VNAME(2,K),V,VDIFF,GV,GVD,GO
      ISUMT(1)=K
C     FOLLOWING COMPUTES ALL POSSIBLE V'S IN REMAINING VARS.
    5 KLIP=ICNT+1
      VPREV=V
      DO 35 II=2,IREM
      KRUD=II-1
```

```
      K=KK(II)
      VS(KRUD)=0.0
      DO 10 I=1,ICNT
      DO 10 J=1,ICNT
 10   R(I,J)=TS(I,J)
      DO 20 J=1,ICNT
      I=ISUMT(J)
      R(KLIP,J)=W(K,I)/ENK
      R(J,KLIP)=R(KLIP,J)
      SS(KLIP,J)=D(K,I)
      SS(J,KLIP)=SS(KLIP,J)
 20    CONTINUE
      R(KLIP,KLIP)=W(K,K)/ENK
      SS(KLIP,KLIP)=D(K,K)
      CALL BORDER(R,WORK,ICNT)
      DO 30 I=1,KLIP
      DO 30 J=1,KLIP
 30   VS(KRUD)=VS(KRUD)+R(I,J)*SS(I,J)
 35   CONTINUE
C     SECTION COMPUTES THE LARGEST V COMPUTED IN PASS
      KRUD=IREM-1
      CALL ORDER(VS,IREM-1,JJ)
      K=JJ(1)
      V=VS(K)
      DO 40 I=1,KRUD
      J=JJ(I)
 40   JJ(I)=KK(J+1)
      DO 50 I=1,KRUD
 50   KK(I)=JJ(I)
      K=KK(1)
      VDIFF=V-VPREV
      IF ( VDIFF .LE. 0. ) GO TO 52
      GVD=GAUSFQ(VDIFF/EK1,1,K1,120)
      GVD=1.0-GVD
      EKLIPS=KLIP
      GV=GAUSFQ(V/(EK1*EKLIPS),3,K1*KLIP,120)
      GV=1.0-GV
      GO=GAUSFQ(X(K),1,K1,NK)
      IF((VDIFF.GT.ENTER).OR.(KRSW.NE.0))   GO TO  55
   52 KRUNCH=ICNT
      KRSW=1
      WRITE(6,104)
 55   ISUMT(KLIP)=K
      WRITE (6,103) K,VNAME(1,K),VNAME(2,K),V,VDIFF,GV,GVD,GO
      DO 60 I=1,KLIP
      J=ISUMT(I)
      SS(KLIP,I)=D(K,J)
      TS(KLIP,I)=W(K,J)/ENK
      TS(I,KLIP)=TS(KLIP,I)
 60   SS(I,KLIP)=SS(KLIP,I)
      CALL BORDER(TS,WORK,ICNT)
      ICNT=ICNT+1
      IREM=IREM-1
      IF(KRSW.EQ.1) GO TO 1313
      IF(IREM.GT.1) GO TO 5
      IF(KRSW.NE.1) KRUNCH =M
      WRITE(6,104)
1313  DO 70 I=1,KRUNCH
      IROW=ISUMT(I)
      DO 70 J=1,KRUNCH
      JROW=ISUMT(J)
 70   TS(I,J)=W(IROW,JROW)
      RETURN
  100 FORMAT (1H1,12X,48H VARIABLE  RAOS V    V ADDED     SIGNIFICANCE O
     1F )
  101 FORMAT(13X,57H ENTERED  FOR TOTAL  BY VAR.  TOTAL SET  ENTERED VAR
     1IABLE  )
  102 FORMAT(1H ,25X,43HSET      ENTERED                AS ADDED ALONE )
  103 FORMAT (I3,1X,A8,1H-,A8,2(F9.2,1X),1PE11.4,2E9.2)
  104 FORMAT(52H0VARIABLES ENTERED TO HERE WILL BE USED FOR ANALYSIS)
      END
      SUBROUTINE HSPLOT(K,NOFNS,RNG)
      REAL*8 VNAME
C     HISTOGRAM AND PLOTTING ROUTINE FOR NUDISC.
      DIMENSION DISC(2),RNG(2,2),RNY(4),RNX(4)
      COMMON L(8),XM(50),GN(50),IBUFF1(102),IBUFF2(102),IFLOW(102,2)
      DATA K1/1HX/,K2/1HS/,K3/1H /
      IF(NOFNS-1)999,500,10
   10 WRITE(6,1000)
 1000 FORMAT(8H1NO PLOT)
  999 RETURN
```

```
C       HISTOGRAM SECTION OF HSPLOT SCORES ON TAPE 4
  500   RANGE=RNG(1,2)-RNG(1,1)
        SCALE1=ALOG10(RANGE)
        SCALE2=AINT(SCALE1)
        REM=SCALE1-SCALE2
        FRAC=10.**REM
        IF(FRAC-2.)510,510,520
  510   STEP=2.*(10.**(SCALE2-2.))
        GO TO 550
  520   IF(FRAC-5.)530,530,540
  530   STEP=5.0*(10.**(SCALE2-2.))
        GO TO 550
  540   STEP=10.**(SCALE2-1.)
  550   BOT=(AINT(RNG(1,1)/STEP))*STEP
        IF(RNG(1,1))555,560,560
  555   BOT=BOT-STEP
  560   CONTINUE
        DO 650 I=1,K
        DO 565 J=1,102
        IBUFF1(J)=0
  565   IBUFF2(J)=K3
        ICNT=GN(I)
        DO 575 J=1,ICNT
        READ(4)XID,PROB,DISC
        JUMP=INT((DISC(1)-BOT)/STEP)+1
        IBUFF1(JUMP)=IBUFF1(JUMP)+1
  575   CONTINUE
        ISEL=50
        IFLCT=1
        DO 585 J=1,101
        IF(IBUFF1(J)-ISEL)585,585,582
  582   IFLOW(IFLCT,1)=J
        IFLOW(IFLCT,2)=IBUFF1(J)
        IBUFF2(J)=K2
        IFLCT=IFLCT+1
  585   CONTINUE
        WRITE(6,600)I
  600   FORMAT(19H1HISTOGRAM OF GROUP,I4,///)
        WRITE(6,601)IBUFF2
  601   FORMAT(11H OVERFLOW I,102A1)
  605   DO 610 J=1,101
        IF(IBUFF1(J)-ISEL)610,606,606
  606   IBUFF2(J)=K1
  610   CONTINUE
        IF (MOD(ISEL,10))615,612,615
  612   WRITE(6,602)ISEL,IBUFF2
  602   FORMAT(1H ,3X,I2,4X,1H+,102A1)
        GO TO 616
  615   WRITE(6,603)IBUFF2
  603   FORMAT(1H ,9X,1HI,102A1)
  616   ISEL=ISEL-1
        IF(ISEL)620,620,605
  620   WRITE(6,621)
  621   FORMAT(1H ,8X,1H0,10(10H+---------),3H+--)
        DO 625 J=1,10
  625   IBUFF2(J)=J*10
  630   WRITE(6,631)(IBUFF2(J),J=1,10)
  631   FORMAT(12H INTERVAL  1,10(7X,I3))
        DO 635 J=1,11
        BUMP=J
  635   XM(J)=BOT+(10.*(BUMP-1.)*STEP)
        WRITE(6,636)(XM(J),J=1,11,2)
  636   FORMAT(1H ,7X,6(E11.4,9X))
        WRITE(6,637)(XM(J),J=2,10,2)
  637   FORMAT(1H ,17X,5(E11.4,9X),///)
        IF(IFLCT-1)650,650,638
  638   WRITE(6,639)
  639   FORMAT(19H OVERFLOW INTERVALS)
        WRITE(6,640)
        WRITE(6,640)((IFLOW(J,N),N=1,2),J=1,IFLCT)
  640   FORMAT(1H ,20(I3,I5))
  650   CONTINUE
        RETURN
        END
Z071    TITLE 'AN Z071S  FORTRAN CORE ALLOCATOR  ** VERSION 1.0 **'
********                    **********                          ********
********          THIS ROUTINE ALLOWS A FORTRAN PROGRAM TO DYNAMICLY
********     ALLOCATE AND FREE BOTH MAIN (HIGH-SPEED) AND BULK (LOW-
********     SPEED) MEMORY DURING PROGRAM EXECUTION.
```

```
********          TO ALLOCATE CORE:                                    00000800
********                                                               00000900
********                   ALLOC$                EITHER BULK OR MAIN   00001000
********              ILOC = ALLOC1 (R,N)        BULK ONLY             00001100
********                   ALLOC2                MAIN ONLY             00001200
********                                                               00001300
********              THE ADDRESS OF ALLOCATED MEMORY IS THE VALUE RETURN 00001400
********              ED BY THE FUNCTION; ZERO IS RETURNED IF NO MEMORY 00001500
********              IS ALLOCATED.                                    00001600
********                                                               00001700
********              THE PARAMETERS ARE:                              00001800
********                   R -  THE RETURN CODE IS PLACED IN R         00001900
********                        VALUES ARE 1 (BULK)  2 (MAIN)          00002000
********                        R IS NEGATIVE IF NO ALLOCATION WAS MADE 00002100
********                   N -  THE NUMBER OF 4-BYTE WORDS TO BE       00002200
********                        ALLOCATED.                             00002300
********                                                               00002400
********          TO FREE CORE:                                        00002500
********                                                               00002600
********                   FREE1                 BULK                  00002700
********              CALL FREE2 (ILOC,N)        MAIN                  00002800
********                                                               00002900
********              THE PARAMETERS ARE:                              00003000
********                   ILOC -  THE STARTING ADDRESS OF PREVIOUSLY  00003100
********                           ALLOCATED MEMORY                    00003200
********                   N    -  THE NUMBER OF 4-BYTE WORDS IN THE   00003300
********                           BLOCK                               00003400
********                                                               00003500
********              *NOTE*     PARAMETERS TO FREE MAY BE GROUPED,E.G., 00003600
********                      CALL FREE1 (ILOC(1),N(1),ILOC(2),N(2))   00003700
********              *NOTE*     RESTRICTIONS:                         00003800
********                      1. THE TYPE OF CORE FREED MUST BE THE TYPE 00003900
********                         OF CORE ALLOCATED,I.E., TYPE 1 CORE (BULK 00004000
********                         ) CANNOT BE FREED BY A TYPE 2 CALL.   00004100
********                      2. INVALID CONDITIONS IN A FREE CALL WILL BE 00004200
********                         DETECTED BY THE OPERATING SYSTEM.     00004300
********                      3. ALLOCATION IS DONE VIA SINGLE ELEMENT, 00004400
********                         CONDITIONAL REQUESTS.                 00004500
********                      4. ALLOCATION OF TYPE1 (BULK) CORE IS ALWAYS 00004600
********                         IN A MULTIPLE OF 2048 BYTES (512 WORDS). 00004700
********                      5. ALLOCATION OF TYPE2 (MAIN) CORE IS ALWAYS 00004800
********                         IN A MULTIPLE OF 8 BYTES (2 WORDS).   00004900
********                                                               00005000
********                         **********                   ********  00005100
********  SPACE 2                                                      00005200
********        PROGRAMMER            JOHN A. OHDE                     00005300
********        DATE                  MAY 1, 1968                      00005400
          GBLC  &BULK                                                  00005500
&BULK     SETC  'YES'                                                  00005600
********                         **********                   ********  00005700
          EJECT                                                        00005800
ALLOC$    CSECT                                                        00005900
          AIF   ('&BULK' NE 'YES').GOA                                 00006000
          ENTRY ALLOC1,FREE1                                           00006100
.GOA      ANOP                                                         00006200
          ENTRY ALLOC2,FREE2,ALLOCP                                    00006300
R0        EQU   0          REGISTER DEFINITIONS                        00006400
R1        EQU   1                                                      00006500
R2        EQU   2                                                      00006600
R3        EQU   3                                                      00006700
R4        EQU   4                                                      00006800
R5        EQU   5                                                      00006900
R6        EQU   6                                                      00007000
R7        EQU   7                                                      00007100
R8        EQU   8                                                      00007200
R9        EQU   9                                                      00007300
RA        EQU   10                                                     00007400
RB        EQU   11                                                     00007500
RC        EQU   12                                                     00007600
RD        EQU   13                                                     00007700
RE        EQU   14                                                     00007800
RF        EQU   15                                                     00007900
          SPACE 3                                                      00008000
          SAVE  (14,12),,ALLOC$      SAVE REGISTERS                    00008100
ALLOC$A   BALR  RA,0                 SET BASE                          00008200
          USING *,RA                                                   00008300
          BAL   R9,LINKSVS           SET NEW SAVE AREA                 00008400
          BAL   R9,FETCHAD           GET PARMS                         00008500
          AIF   ('&BULK' NE 'YES').GO1                                 00008600
          MVI   ALLOC$SW+1,X'F0'      SET TWO-TRY SWITCH ON            00008700
          B     ALLOC1A                START WITH BULK                 00008800
```

```
         AGO    .GO2                                                    00008900
.GO1     ANOP                                                           00009000
         B      ALLOC2A                GO TO ALLOCATE MAIN CORE         00009100
.GO2     ANOP                                                           00009200
         AIF    ('&BULK' NE 'YES').GO3                                  00009300
         EJECT                                                          00009400
ALLOC1   EQU    *                 ALLOCATE BULK CORE ROUTINE           00009500
         SAVE   (14,12),,ALLOC1        SAVE REGISTERS                   00009600
ALLOC$B  BALR   RA,0                                                    00009700
         LA     R9,ALLOC$B-ALLOC$A                                      00009800
         SLR    RA,R9                  KEEP SAME BASE                   00009900
         BAL    R9,LINKSVS             SET NEW SAVE AREA                00010000
         BAL    R9,FETCHAD             GET PARMS                        00010100
         MVI    ALLOC$SW+1,X'00'       TURN TWO-TRY SWITCH OFF          00010200
ALLOC1A  EQU    *                                                       00010300
         ST     R4,BULKSIZE            SAVE REQUEST SIZE (BYTES)        00010400
         LA     R1,BULKLIST                                             00010500
         GETBULK     MF=(E,(1))        ALLOCATE                         00010600
         SPACE 2                                                        00010700
         LA     R5,1                   SET RETURN CODE                  00010800
         LTR    RF,RF                  TEST RETURN CODE                 00010900
         BNZ    NOALLOC1                NO ALLOCATION                   00011000
         L      R0,BULKOUT             SET ADDRESS                      00011100
.GO3     ANOP                                                           00011200
         SPACE 6                                                        00011300
EXIT     EQU    *                                                       00011400
         ST     R5,0(R2)               STORE IT                         00011500
EXIT1    L      RB,4(RD)                                                00011600
EXIT2    LR     RD,RB                  RESET SAVE AREA POINTER          00011700
         LM     14,15,12(13)           RESTORE ALL REGISTERS            00011800
         LM     1,12,24(13)              EXCEPT 0 (RETURN VALUE)        00011900
         MVI    12(13),X'FF'           SIGNAL RETURN                    00012000
         SR     RF,RF                                                   00012100
         BR     14                     RETURN                           00012200
         AIF    ('&BULK' NE 'YES').GO4                                  00012300
         SPACE 3                                                        00012400
NOALLOC1 LNR    R5,R5                  SET NO ALLOCATION                00012500
         SR     R0,R0                                                   00012600
ALLOC$SW NOP    ALLOC2A                TWO-TRY SWITCH: NOP|BRANCH       00012700
         B      EXIT                                                    00012800
.GO4     ANOP                                                           00012900
         EJECT                                                          00013000
ALLOC2   EQU    *                 ALLOCATE MAIN CORE ROUTINE           00013100
         SAVE   (14,12),,ALLOC2        SAVE REGISTERS                   00013200
ALLOC$C  BALR   RA,0                                                    00013300
         LA     R9,ALLOC$C-ALLOC$A                                      00013400
         SLR    RA,R9                  KEEP SAME BASE                   00013500
         BAL    R9,LINKSVS             SET NEW SAVE AREA                00013600
         BAL    R9,FETCHAD             GET PARMS                        00013700
ALLOC2A  EQU    *                                                       00013800
         ST     R4,MAINSIZE            SAVE REQUEST SIZE (BYTES)        00013900
         LA     R1,MAINLIST                                             00014000
         GETMAIN     MF=(E,(1))        ALLOCATE                         00014100
         SPACE 2                                                        00014200
         LA     R5,2                   SET RETURN CODE                  00014300
         LTR    RF,RF                  TEST RETURN CODE                 00014400
         BNZ    NOALLOC2                NO ALLOCATION                   00014500
         L      R0,MAINOUT             ADDRESS                          00014600
         B      EXIT                                                    00014700
         SPACE 3                                                        00014800
NOALLOC2 LNR    R5,R5                  SET NO ALLOCATION                00014900
         SR     R0,R0                                                   00015000
         B      EXIT                                                    00015100
         EJECT                                                          00015200
         AIF    ('&BULK' NE 'YES').GO5                                  00015300
FREE1    EQU    *                 FREE BULK CORE ROUTINE               00015400
         SAVE   (14,12),,FREE1         SAVE REGISTERS                   00015500
ALLOC$D  BALR   RA,0                                                    00015600
         LA     R9,ALLOC$D-ALLOC$A                                      00015700
         SLR    RA,R9                  KEEP SAME BASE                   00015800
         BAL    R9,LINKSVS             SET NEW SAVE AREA                00015900
FREE1A   EQU    *                                                       00016000
         BAL    R9,FETCHAD             GET PARMS                        00016100
         MVC    BULKIN(4),0(R2)        ADDRESS OF CORE                  00016200
         ST     R4,BULKCT              SIZE OF BLOCK (BYTES)            00016300
         LA     R1,BULKFREE                                             00016400
         FREEBULK    MF=(E,(1))        FREE IT                          00016500
         SPACE 2                                                        00016600
         LTR    R3,R3                                                   00016700
         BM     EXIT1                  LAST OF PARMS - EXIT             00016800
         LA     R6,8(R6)                                                00016900
```

```
         LR    R1,R6                                                    00017000
         B     FREE1A                GET NEXT SEQUENCE                  00017100
         EJECT                                                          00017200
.G05     ANOP                                                           00017300
FREE2    EQU   *                 FREE MAIN CORE ROUTINE                 00017400
         SAVE  (14,12),,FREE2        SAVE REGISTERS                     00017500
ALLOC$E  BALR  RA,0                                                     00017600
         LA    R9,ALLOC$E-ALLOC$A                                       00017700
         SLR   RA,R9                 KEEP SAME BASE                     00017800
         BAL   R9,LINKSVS            SET NEW SAVE AREA                  00017900
FREE2A   EQU   *                                                        00018000
         BAL   R9,FETCHAD            GET PARMS                          00018100
         MVC   MAININ(4),0(R2)       ADDRESS OF CORE                    00018200
         ST    R4,MAINCT             SIZE OF BLOCK (BYTES)              00018300
         LA    R1,MAINFREE                                              00018400
         FREEMAIN    MF=(E,(1))      FREE IT                            00018500
         SPACE 2                                                        00018600
         LTR   R3,R3                                                    00018700
         BM    EXIT1                 LAST OF PARMS - EXIT               00018800
         LA    R6,8(R6)                                                 00018900
         LR    R1,R6                                                    00019000
         B     FREE2A                GET NEXT PARMS                     00019100
         EJECT                                                          00019200
ALLOCP   EQU   *                                                        00019300
         SAVE  (14,12),,ALLOCP       SAVE REGISTERS                     00019400
ALLOC$F  BALR  RA,0                                                     00019500
         LA    R9,ALLOC$F-ALLOC$A                                       00019600
         SLR   RA,R9                                                    00019700
*                                                                       00019800
         LR    R2,R1                                                    00019900
         GETMAIN R,LV=72                                                00020000
         LR    RB,RD                 GET AND                            00020100
         LR    RD,R1                         CHAIN                    00020200
         ST    RD,8(RB)                            SAVE                 00020300
         ST    RB,4(RD)                                  AREAS          00020400
         ST    R1,TEMP1              SAVE AREA ADDRESS                  00020500
*                                                                       00020600
         LR    R3,R2                                                    00020700
GETLAST  LA    R3,4(R3)                                                 00020800
         TM    0(R3),X'80'           TEST FOR LAST PARM                 00020900
         BNO   GETLAST                NO                                00021000
         LR    R4,R3                  YES                              00021100
         SR    R4,R2                                                    00021200
         LA    R0,0(R4)                                                 00021300
         ST    R0,TEMP2              SAVE AREA SIZE                     00021400
         GETMAIN R,LV=(0)            GET NEW P-LIST AREA                00021500
*                                                                       00021600
         ST    R1,TEMP3              SAVE AREA ADDRESS                  00021700
         L     RF,0(R2)                                                 00021800
         L     RF,0(RF)              GET BRANCH ADDRESS (1ST PARM)      00021900
         LA    R2,4(R2)              RESET LIST POINTER                 00022000
         LR    R7,R1                                                    00022100
         LA    R4,8                  SET UP FOR BXLE BELOW              00022200
         LR    R5,R3                                                    00022300
MOVELIST L     R6,0(R2)              MOVE THE CONTENTS                  00022400
         L     R6,0(R6)                 TO THE NEW P-LIST               00022500
         ST    R6,0(R7)                                                 00022600
         MVC   4(4,R7),4(R2)         ADCON FOR WORD COUNT               00022700
         LA    R7,8(R7)              BUMP TARGET ADDRESS                00022800
         BXLE  R2,R4,MOVELIST                                           00022900
         MVI   0(R3),X'80'           SET LAST                           00023000
*                                                                       00023100
         BALR  RE,RF                 BRANCH                             00023200
*                                                                       00023300
         L     R0,TEMP2                                                 00023400
         L     R1,TEMP3                                                 00023500
         FREEMAIN  R,LV=(0),A=(1)    FREE THE PARM LIST AREA            00023600
*                                                                       00023700
         L     RD,4(RD)                                                 00023800
         L     R1,TEMP1                                                 00023900
         FREEMAIN  R,LV=72,A=(1)      FREE THE SAVE AREA USED           00024000
*                                                                       00024100
         B     EXIT2                                                    00024200
         EJECT                                                          00024300
********     SUBROUTINES                                        ******** 00024400
         SPACE 2                                                        00024500
LINKSVS  EQU   *                 CHAIN SAVE AREAS                       00024600
         LR    RB,RD                                                    00024700
         LA    RD,SAVEAREA                                              00024800
         ST    RD,8(RB)                                                 00024900
         ST    RB,4(RD)                                                 00025000
```

```
         BR     R9                                                      00025100
         SPACE  6                                                       00025200
FETCHAD  EQU    *                GET PARAMETER ADDRESSES               00025300
         LM     R2,R3,0(R1)           TWO PER CALL                     00025400
         L      R4,0(R3)               SIZE IN WORDS                   00025500
         SLA    R4,2                   MULTIPLY BY FOUR                00025600
         LR     R6,R1                 SAVE LIST POINTER                00025700
         BR     R9                                                      00025800
         EJECT                                                          00025900
********         CONSTANTS                                       ******** 00026000
         SPACE  2                                                       00026100
SAVEAREA DC     18F'0'                                                  00026200
TEMP1    DC     F'0'                                                    00026300
TEMP2    DC     F'0'                                                    00026400
TEMP3    DC     F'0'                                                    00026500
*                                                                       00026600
MAINLIST GETMAIN EC,LV=2048,A=MAINOUT,MF=L,SP=2                         00026700
*                                                                       00026800
         AIF    ('&BULK' NE 'YES').G06                                  00026900
BULKLIST GETBULK EC,LV=2048,A=BULKOUT,MF=L,SP=1                         00027000
*                                                                       00027100
.G06     ANOP                                                           00027200
MAINFREE FREEMAIN     LV=2048,A=MAININ,MF=L,SP=2                        00027300
*                                                                       00027400
         AIF    ('&BULK' NE 'YES').G07                                  00027500
BULKFREE FREEBULK     LV=2048,A=BULKIN,MF=L,SP=1                        00027600
*                                                                       00027700
.G07     ANOP                                                           00027800
MAINOUT  DC     F'0'                                                    00027900
MAININ   DC     F'0'                                                    00028000
MAINSIZE EQU    MAINLIST                                                00028100
MAINCT   EQU    MAINFREE                                                00028200
         AIF    ('&BULK' NE 'YES').G08                                  00028300
BULKOUT  DC     F'0'                                                    00028400
BULKIN   DC     F'0'                                                    00028500
BULKCT   EQU    BULKFREE                                                00028600
BULKSIZE EQU    BULKLIST                                                00028700
.G08     ANOP                                                           00028800
         SPACE  5                                                       00028900
         END                                                            00029000
```

3. TABPRED Tabulation Prediction Program

```
      SUBROUTINE FORT(NVAR,NOBS,X,SN,PN)
C     NVAR          NUMBER OF VARIABLES
C     NOBS          NUMBER OF OBSERVATIONS
C     X             ARRAY OF VALUES
C     SUBSCRIPT ORDER FOR VARIABLES
C      TAM_1 SO2, TAM_1 WIND_VEL, TAM_1 WIND_DIR,
C      TAM_2 SO2, TAM_2 WIND_VEL, TAM_2 WIND_DIR,
C        .          .
C      TAM_ITAM SO2, TAM_ITAM WIND_VEL, TAM_ITAM WIND_DIR,
C      HOUR, DAY, MONTH, YEAR,
C      CEIL, TEMP,
C      LIQ_PCP1, LIQ_PCP2,
C      FRO_PCP1, FRO_PCP2, FRO_PCP3
      REAL*8 TITLE,PERIOD,SN(1),PN(1)
      DIMENSION XX(500),YY(500)
      REALX(NOBS,1)
      DIMENSION CEILB(20),WINDB(20),TEMPB(20),BOUND(2,4),NB(100)
      COMMON /A/                IBOUND(2,4),J1,XXP
      COMMON /C/ TITLE(7),PERIOD(3)
      COMMON /D/ NHR(100)
      COMMON/F/ XT(30),PLOTT(25,50),HMAX(50),HMIN(50),MYSAVE(50),JX,JY,K
      READ (5,1000) TITLE,PERIOD
      READ (5,1001) ITAM,ICEIL,IWIND,ITEMP
      N=ICEIL+1
      READ (5,1002) (CEILB(I),I=1,N)
      N=IWIND+1
      READ (5,1002) (WINDB(I),I=1,N)
      N=ITEMP+1
      READ (5,1002) (TEMPB(I),I=1,N)
      READ (5,1001) (NHR(I),I=1,ITAM)
      READ (5,1001) (NB(I),I=1,ITAM)
      READ (5,1002) PRECIP
 1000 FORMAT (10A8)
 1001 FORMAT (8I8)
 1002 FORMAT (10F8.0)
      DO 30 I=1,25
   30 XT(I)=(I-1)*.05
      JX=24
      I6=3*ITAM+1
      I7=I6+1
      I3=3*ITAM+5
      I5=I3+1
      K1=I5+1
      K2=K1+1
      K3=K1+2
      K4=K1+3
      K5=K1+4
      DO 600 K=1,NOBS
      X(K,I7)=0.
      IF (X(K,K1).GT.0..OR.X(K,K2).GT.0..OR.X(K,K3).GT.0..OR.X(K,K4).GT.
     1      0..OR.X(K,K5).GT.0. ) X(K,I7)=1.0
  600 CONTINUE
      DO 100 K=1,ITAM
      I1=1+(K-1)*3
      I2=I1+2
      I4=I1+1
      JT1=NB(K)
      XXP=JT1*4.*6.*12.
      DO 100 J1=1,JT1
      DO 100 J2=1,ICEIL
      BOUND(1,2)=CEILB(J2)
      BOUND(2,2)=CEILB(J2+1)
      IBOUND(1,2)=BOUND(1,2)
      IBOUND(2,2)=BOUND(2,2) -1.
      DO 100 J3=1,IWIND
      BOUND(1,3)=WINDB(J3)
      BOUND(2,3)=WINDB(J3+1)
      IBOUND(1,3)=BOUND(1,3)
      IBOUND(2,3)=BOUND(2,3)  -1.
      DO 100 J4=1,ITEMP
      BOUND(1,4)=TEMPB(J4)
      BOUND(2,4)=TEMPB(J4+1)
      IBOUND(1,4)=BOUND(1,4)
      IBOUND(2,4)=BOUND(2,4) -1.
      N=0
      DO 50 I=1,NOBS
      IF ( X(I,I7) .NE. PRECIP ) GO TO 50
      IF (X(I,I1) .LT. .001 .OR. X(I,I1) .GT.  2. ) GO TO 50
      IF(X(I,I3).LT.BOUND(1,2)*.01.OR.X(I,I3).GE.BOUND(2,2)*.01)GO TO 50
      IF (X(I,I4) .LT. BOUND(1,3) .OR. X(I,I4) .GE. BOUND(2,3)) GO TO 50
      IF (X(I,I5) .LT. BOUND(1,4) .OR. X(I,I5) .GE. BOUND(2,4)) GO TO 50
```

```
      CALL WIND (X(I,I2),K,J1,IND)
      IF (IND .EQ. 0 ) GO TO 50
      N=N+1
      XX(N)=X(I,I1)
      YY(N)=X(I,I6)
   50 CONTINUE
      IF (N .EQ. 0 ) GO TO 100
      CALL JPLOT1 (N,XX,YY)
      CALL TWODFD
  100 CONTINUE
      RETURN
      END
      SUBROUTINE WIND ( W,K,J1,IND)
C     W - WIND DIRECTION
C     K - TAM STATION NUMBER
C     J1 - BAND NUMBER
C     IND - RETURN INDICATOR
C         IND=1  DATA POINT IN BAND
C         IND=0  DATA POINT NOT IN BAND
      GO TO (31,32,33,34,35,36,37,38),K
   31 GO TO (141,142,143,144,145,146,147,148),J1
  141 IF ( W .GE. 360. .OR.  W .LE. 45. ) GO TO 70
      GO TO 50
  142 IF ( W .GE. 45. .AND. W .LE. 90. ) GO TO 70
      GO TO 50
  143 IF ( W .GE. 90. .AND. W .LE. 135. ) GO TO 70
      GO TO 50
  144 IF ( W .GE. 135. .AND. W .LE. 180. ) GO TO 70
      GO TO 50
  145 IF ( W .GE. 180. .AND. W .LE. 225. ) GO TO 70
      GO TO 50
  146 IF ( W .GE. 225. .AND. W .LE. 270. ) GO TO 70
      GO TO 50
  147 IF ( W .GE. 270. .AND. W .LE. 315. ) GO TO 70
      GO TO 50
  148 IF ( W .GE. 315. .AND. W .LE. 360. ) GO TO 70
      GO TO 50
   32 GO TO (241,242,243,244,245,246,247,248),J1
  241 IF ( W .GE. 360. .OR.  W .LE. 45. ) GO TO 70
      GO TO 50
  242 IF ( W .GE. 45. .AND. W .LE. 90. ) GO TO 70
      GO TO 50
  243 IF ( W .GE. 90. .AND. W .LE. 135. ) GO TO 70
      GO TO 50
  244 IF ( W .GE. 135. .AND. W .LE. 180. ) GO TO 70
      GO TO 50
  245 IF ( W .GE. 180. .AND. W .LE. 225. ) GO TO 70
      GO TO 50
  246 IF ( W .GE. 225. .AND. W .LE. 270. ) GO TO 70
      GO TO 50
  247 IF ( W .GE. 270. .AND. W .LE. 315. ) GO TO 70
      GO TO 50
  248 IF ( W .GE. 315. .AND. W .LE. 360. ) GO TO 70
      GO TO 50
   33 GO TO (341,342,343,344,345,346,347,348),J1
  341 IF ( W .GE. 360. .OR.  W .LE. 45. ) GO TO 70
      GO TO 50
  342 IF ( W .GE. 45. .AND. W .LE. 90. ) GO TO 70
      GO TO 50
  343 IF ( W .GE. 90. .AND. W .LE. 135. ) GO TO 70
      GO TO 50
  344 IF ( W .GE. 135. .AND. W .LE. 180. ) GO TO 70
      GO TO 50
  345 IF ( W .GE. 180. .AND. W .LE. 225. ) GO TO 70
      GO TO 50
  346 IF ( W .GE. 225. .AND. W .LE. 270. ) GO TO 70
      GO TO 50
  347 IF ( W .GE. 270. .AND. W .LE. 315. ) GO TO 70
      GO TO 50
  348 IF ( W .GE. 315. .AND. W .LE. 360. ) GO TO 70
      GO TO 50
   34 GO TO (441,442,443,444,445,446,447,448),J1
  441 IF ( W .GE. 360. .OR.  W .LE. 45. ) GO TO 70
      GO TO 50
  442 IF ( W .GE. 45. .AND. W .LE. 90. ) GO TO 70
      GO TO 50
  443 IF ( W .GE. 90. .AND. W .LE. 135. ) GO TO 70
      GO TO 50
  444 IF ( W .GE. 135. .AND. W .LE. 180. ) GO TO 70
      GO TO 50
  445 IF ( W .GE. 180. .AND. W .LE. 225. ) GO TO 70
```

```
      GO TO 50
  446 IF ( W .GE. 225. .AND. W .LE. 270. ) GO TO 70
      GO TO 50
  447 IF ( W .GE. 270. .AND. W .LE. 315. ) GO TO 70
      GO TO 50
  448 IF ( W .GE. 315. .AND. W .LE. 360. ) GO TO 70
      GO TO 50
   35 GO TO (541,542,543,544,545,546,547,548),J1
  541 IF ( W .GE. 360. .OR.  W .LE. 45. ) GO TO 70
      GO TO 50
  542 IF ( W .GE. 45. .AND. W .LE. 90. ) GO TO 70
      GO TO 50
  543 IF ( W .GE. 90. .AND. W .LE. 135. ) GO TO 70
      GO TO 50
  544 IF ( W .GE. 135. .AND. W .LE. 180. ) GO TO 70
      GO TO 50
  545 IF ( W .GE. 180. .AND. W .LE. 225. ) GO TO 70
      GO TO 50
  546 IF ( W .GE. 225. .AND. W .LE. 270. ) GO TO 70
      GO TO 50
  547 IF ( W .GE. 270. .AND. W .LE. 315. ) GO TO 70
      GO TO 50
  548 IF ( W .GE. 315. .AND. W .LE. 360. ) GO TO 70
      GO TO 50
   36 GO TO (641,642,643,644,645,646,647,648),J1
  641 IF ( W .GE. 360. .OR.  W .LE. 45. ) GO TO 70
      GO TO 50
  642 IF ( W .GE. 45. .AND. W .LE. 90. ) GO TO 70
      GO TO 50
  643 IF ( W .GE. 90. .AND. W .LE. 135. ) GO TO 70
      GO TO 50
  644 IF ( W .GE. 135. .AND. W .LE. 180. ) GO TO 70
      GO TO 50
  645 IF ( W .GE. 180. .AND. W .LE. 225. ) GO TO 70
      GO TO 50
  646 IF ( W .GE. 225. .AND. W .LE. 270. ) GO TO 70
      GO TO 50
  647 IF ( W .GE. 270. .AND. W .LE. 315. ) GO TO 70
      GO TO 50
  648 IF ( W .GE. 315. .AND. W .LE. 360. ) GO TO 70
      GO TO 50
   37 GO TO (741,742,743,744,745,746,747,748),J1
  741 IF ( W .GE. 360. .OR.  W .LE. 45. ) GO TO 70
      GO TO 50
  742 IF ( W .GE. 45. .AND. W .LE. 90. ) GO TO 70
      GO TO 50
  743 IF ( W .GE. 90. .AND. W .LE. 135. ) GO TO 70
      GO TO 50
  744 IF ( W .GE. 135. .AND. W .LE. 180. ) GO TO 70
      GO TO 50
  745 IF ( W .GE. 180. .AND. W .LE. 225. ) GO TO 70
      GO TO 50
  746 IF ( W .GE. 225. .AND. W .LE. 270. ) GO TO 70
      GO TO 50
  747 IF ( W .GE. 270. .AND. W .LE. 315. ) GO TO 70
      GO TO 50
  748 IF ( W .GE. 315. .AND. W .LE. 360. ) GO TO 70
      GO TO 50
   38 GO TO (841,842,843,844,845,846,847,848),J1
  841 IF ( W .GE. 360. .OR.  W .LE. 45. ) GO TO 70
      GO TO 50
  842 IF ( W .GE. 45. .AND. W .LE. 90. ) GO TO 70
      GO TO 50
  843 IF ( W .GE. 90. .AND. W .LE. 135. ) GO TO 70
      GO TO 50
  844 IF ( W .GE. 135. .AND. W .LE. 180. ) GO TO 70
      GO TO 50
  845 IF ( W .GE. 180. .AND. W .LE. 225. ) GO TO 70
      GO TO 50
  846 IF ( W .GE. 225. .AND. W .LE. 270. ) GO TO 70
      GO TO 50
  847 IF ( W .GE. 270. .AND. W .LE. 315. ) GO TO 70
      GO TO 50
  848 IF ( W .GE. 315. .AND. W .LE. 360. ) GO TO 70
   50 IND=0
      RETURN
   70 IND=1
      RETURN
      END
      SUBROUTINE JPLOT1 (NXY,X,Y)
      COMMON/F/ XX(30),PLOTT(25,50),HMAX(50),HMIN(50),MYSAV (50),JX,JY,K
```

```
      DIMENSION XSUM(30),XXSUM(30)
      COMMON/B/ XMEAN(30),XSD(30)
      COMMON /D/ JJY(100)
      DIMENSION X(NXY),Y(NXY)
      JY=JJY(K)
      XMIN=XX(1)
      XMAX=XX(JX+1)
      DO 10 M=1,25
      DO 10 N=1,50
   10 PLOTT(M,N)=0.
C       SET XX AND YY AXES.
      DO 25 N=1,JY
      XSUM(N)=0.
      XXSUM(N)=0.
      XMEAN(N)=0.
      XSD(N)=0.
      HMAX(N)=0.
      HMIN(N)=0.
   25 MYSAV(N)=0
      XR=JX/(XMAX-XMIN)
C       NXY IS THE NUMBER OF X AND Y PAIRS.
      DO 50 I=1,NXY
      MX=XR*(X(I)-XMIN) + 1
      IF(X(I).LT.XX(MX))  MX=MX-1
      IF(MX.LE.0) MX=1
      IF(X(I).GE.XMAX)MX=JX
      MY=IHOUR(Y(I),K)
      XSUM(MY)=X(I)+XSUM(MY)
      XXSUM(MY)=X(I)*X(I)+XXSUM(MY)
      MYSAV(MY)=MYSAV(MY)+1
 26   IF(MYSAV(MY).GT.1) GO TO 27
         HMAX(MY)=X(I)
         HMIN(MY)=X(I)
   27 IF(X(I).GT.HMAX(MY))HMAX(MY)=X(I)
      IF(X(I).LT.HMIN(MY))HMIN(MY)=X(I)
 50   PLOTT(MX,MY)=PLOTT(MX,MY)+1.
      DO 60 N=1,JY
      IF (MYSAV (N) .EQ. 0) GO TO 60
      XMEAN(N)=XSUM(N)/MYSAV (N)
      XSD(N)=XXSUM(N)/MYSAV (N)-XMEAN(N)*XMEAN(N)
      IF (XSD(N) .LT. 0. ) XSD(N)=0.
   60 XSD(N)=SQRT(XSD(N))
      RETURN
      END
      FUNCTION IHOUR(Y,K)
C     Y - HOUR OF DAY FOR DATA POINT
C     K - TAM STATION NUMBER
C     IHOUR - HOUR BAND NUMBER
      MY=0
      GO TO (31,32,33,34,35,36,37,38),K
   31 IF ( Y .GE. 7. .AND. Y .LT. 11. ) MY=1
      IF ( Y .GE. 11..AND. Y .LT. 17. ) MY=2
      IF ( Y .GE. 17..AND. Y .LT. 23. ) MY=3
      IF ( MY .EQ. 0 ) MY=4
      GO TO 50
   32 IF ( Y .GE. 7. .AND. Y .LT. 11. ) MY=1
      IF ( Y .GE. 11..AND. Y .LT. 17. ) MY=2
      IF ( Y .GE. 17..AND. Y .LT. 23. ) MY=3
      IF ( MY .EQ. 0 ) MY=4
      GO TO 50
   33 IF ( Y .GE. 7. .AND. Y .LT. 11. ) MY=1
      IF ( Y .GE. 11..AND. Y .LT. 17. ) MY=2
      IF ( Y .GE. 17..AND. Y .LT. 23. ) MY=3
      IF ( MY .EQ. 0 ) MY=4
      GO TO 50
   34 IF ( Y .GE. 7. .AND. Y .LT. 11. ) MY=1
      IF ( Y .GE. 11..AND. Y .LT. 17. ) MY=2
      IF ( Y .GE. 17..AND. Y .LT. 23. ) MY=3
      IF ( MY .EQ. 0 ) MY=4
      GO TO 50
   35 IF ( Y .GE. 7. .AND. Y .LT. 11. ) MY=1
      IF ( Y .GE. 11..AND. Y .LT. 17. ) MY=2
      IF ( Y .GE. 17..AND. Y .LT. 23. ) MY=3
      IF ( MY .EQ. 0 ) MY=4
      GO TO 50
   36 IF ( Y .GE. 7. .AND. Y .LT. 11. ) MY=1
      IF ( Y .GE. 11..AND. Y .LT. 17. ) MY=2
      IF ( Y .GE. 17..AND. Y .LT. 23. ) MY=3
      IF ( MY .EQ. 0 ) MY=4
      GO TO 50
   37 IF ( Y .GE. 7. .AND. Y .LT. 11. ) MY=1
```

```
      IF ( Y .GE. 11..AND. Y .LT. 17. ) MY=2
      IF ( Y .GE. 17..AND. Y .LT. 23. ) MY=3
      IF ( MY .EQ. 0 ) MY=4
      GO TO 50
   38 IF ( Y .GE. 7. .AND. Y .LT. 11. ) MY=1
      IF ( Y .GE. 11..AND. Y .LT. 17. ) MY=2
      IF ( Y .GE. 17..AND. Y .LT. 23. ) MY=3
      IF ( MY .EQ. 0 ) MY=4
   50 IHOUR=MY
      RETURN
      END
      SUBROUTINE TWODFD
      REAL*8 TITLE,PERIOD
      COMMON /A/                IBOUND(2,4),J1,XXP
      COMMON/B/ XMEAN(30),XSD(30)
      COMMON /C/ TITLE(7),PERIOD(3)
      COMMON/F/XX(30),PLOT(25,50),HMAX(50),HMIN(50),ISU(50),MAXI,MAXJ,K
      DIMENSION CUMSM(25,50),PCTL(10),XPCTL(55,8),DIFF1(50),
     1      DIFF2(50),DIFF3(50)
      DATA PCTL/10.,25.,50.,75.,90.,95.,98.,99./
      DATA NPCTLS/8/
      DATA KK/0/
      IF ( K.EQ. KK)  GO TO 602
      LINE=60
      EK=K
  602 KK=K
  604 DO 652 J=1,MAXJ
      IF ( ISU(J) .LT. 1 ) GO TO 652
      DO 651 I=1,MAXI
  675 CUMSM(I,J)=0.
       PLOT(I,J)= PLOT(I,J)*100./ ISU(J)
      IF(I.EQ.1)GO TO 653
      CUMSM(I,J)= PLOT(I,J)+CUMSM(I-1,J)
      GO TO 651
  653 CUMSM(1,J)= PLOT(1,J)
  651 CONTINUE
  652 CONTINUE
      DO 594J=1,MAXJ
      DO 594N=1,NPCTLS
  594 XPCTL(J,N)=0.
      DO 699 J=1,MAXJ
      IF (ISU(J) .LT. 1 ) GO TO 692
      DO 698 N=1,NPCTLS
      DO 697 I=1,MAXI
      IF(PCTL(N).GT.CUMSM(I,J)) GO TO 697
      IF(I.EQ.1)GO TO 696
      XPCTL(J,N)=(((PCTL(N)-CUMSM(I-1,J))/(CUMSM(I,J)-CUMSM(I-1,J)))*
     1      (XX(I+1)-XX(I)))+XX(I)
      GO TO 695
  696 XPCTL(J,N)=((PCTL(N)/CUMSM(1,J))*(XX(2)-XX(1)))+XX(1)
  695 IF(XPCTL(J,N).GT.HMAX(J))XPCTL(J,N)=HMAX(J)
      IF(XPCTL(J,N).LT.HMIN(J))XPCTL(J,N)=HMIN(J)
      GO TO 698
  697 CONTINUE
  698 CONTINUE
  692 DIFF1(J)=XPCTL(J,4)-XPCTL(J,2)
      DIFF2(J)=XPCTL(J,6)-XPCTL(J,4)
      DIFF3(J)=HMAX(J)-XPCTL(J,4)
  699 CONTINUE
C     PRINT OUT DATA
      DO 785 J=1,MAXJ
      IF (LINE .LT. 60 ) GO TO 780
      WRITE (6,8001) TITLE,K,PERIOD
      WRITE (6,8002)
      WRITE (6,8004)
      LINE=9
  780 WRITE(6,8003)J1,((IBOUND(KP,LP),KP=1,2),LP=2,4),J,HMIN(J),
     1 (XPCTL(J,KP),KP=2,8),HMAX(J),DIFF1(J),DIFF2(J),XMEAN(J),XSD(J),
     2   ISU(J)
      LINE=LINE+1
  785 CONTINUE
 8001 FORMAT (1H1,47X,'TABULATION PREDICTION TECHNIQUE'//1X,7A8,' STATIO
     1N: TAM    ',I1,'    PERIOD OF DATA: ',3A8,//1X,132('-'))
 8002 FORMAT(' WIND DIR| CEILING |WIND SP|  TEMP   |   HOUR   |',11X,
     1        'PERCENTILE VALUES OF SO2(PPM) CONCENTRATIONS',11X,
     2        '| MEAN | ST DEV |FREQ'/    ' DEGREES |  FEET    |  KT    |',
     3         ' DEG F |  CST    | MIN     25     50     75     90     95     98
     4     99    MAX  75-25  95-75 |          |          |'/1X,132('-'))
 8003 FORMAT(2X,'BAND',I2,' |',I4,'-',I4,'|' ,I3,'-',I2,' |',I3,'-',I3,
     1    ' | BAND',I2,' |',F5.2,10F6.2,' |',F5.2,' |',F7.4,' |',I3)
 8004 FORMAT (2(9X,'|'),7X,'|',8X,'|',8X,'|',66X,'|',6X,'|',8X,'|')
```

4. PUFF Atmospheric Diffusion Calculations Dispersion Model

```
      SUBROUTINEIPLUME(EMIS,HEFF)
C CCC EMIS AND HEFF NOT NOW USED.  ARRAYS XR,YR,ZR,RECDAT,SOBS ARE USED.
      REAL*4 EMIS(12,4,1),HEFF(12,4,1)
      REAL*4 XS(1),YS(1),ZS(4,1),QSO2(  1),QHEAT(  1),XR(1),YR(1),ZR(1),
     1 RECDAT(1),WDA(1),WSA(1),PERCENT(4,1)  ,SOBS(1)
C
C       COORDINATE SYSTEM IS +X LEFT,+Y DOWN.
      DIMENSION CHI(50),XRR(50),YRR(50)
C     UP TO 50 RECEPTOR POINTS.
C     DATA XRR/5.,7.,48*C./
C     DATA YRR/-3.,+3.,48*C./
C     NRE =2
C    RECEPTOR POINTS NUMBERED 1-NREC IDENTIFIED BY DATA CARDS. XR,YR
C                                                          IN MILES.
C       CALCULATES GROUND LEVEL CONCENTRATIONS AT EACH RECEPTOR
C       DUE TO AN ASSEMBLAGE OF POINT SOURCES.   ASSUMES STEADY
C       STATE PLUME EQUATION HOLDS FOR EACH HOUR.
      COMMON/MET/WD,WS,HMIX,H,JSTAB,DELH,EFFHGT
      COMMON/COORD/Y,Z,NF
      COMMON/SIG/SY,SZ
      DIMENSION KTAM(8)
      COMMON/RISE/ZSS
     ,,WSMPS
      MMM=0
      RETURN
C       CODE COORDINATES,PLUS FOR X LEFT AND Y DOWN
      ENTRY    PLUMES(IR,TEMP,WSA,WDA,STAB,HLID,NSRC,XS,YS,ZS,
     1 PERCENT,QSO2,QHEAT,
     1NREC,XR,YR,ZR,RECDAT,MM,KTAM,SOBS)
C     SOBS IS THE ACTUAL OBSERVED SO2.  USED HERE ONLY IN PUNCHED
C        OUTPUT FOR STATISTICAL COMPARISON  WITH CALCULATED VALUES
C     UNITS ARE F,DEG,MPH,,FT,,MI,MI,FT,FRACTION,LB/HR,TH/HR,,MI,MI,FT,,
C     WDA,WSA,          ARRAYS ARE NUMBERED SEQUENTIALLY FROM 1
C          UP TO NREC IN PL1.  THUS IF WANT TO STUDY TAM 3 ONLY,
C          WILL FIND WD(TAM3) AS WDA(1),ETC.
C     TAM WIND IS SPECIFIED BY THE RECEPTOR INPUT CARD
C
      NRE=NREC
      DO 10 NR=1,NRE
   10 CHI(NR)=0.
      JSTAB=STAB
      DO 50 NR=1,NRE
      DO 45 NS=1,NSRC
      DO 45 K=1,4
      J=NS
      QPOL=QSO2(  J)*PERCENT(K,J)
      IF(QPOL.LT.1.) GO TO 45
      QHT=QHEAT(  J)*PERCENT(K,J)
      WS=WSA(1)
      WD=WDA(1)
      U=WS*SIN(WD*.01745)
      V=WS*COS(WD*.01745)
      ZSS=ZS(K,J)
      KSTAB=0
      IF(HLID.LT.ZSS)KSTAB=1
C       IF KSTAB=1,HAVE STACK ABOVE LID.USE JSTAB=5.
      WSS=WS+.1
      IF(KSTAB.GT.0) GO TO 45
C    CODE NEGLECTS STACKS HIGHER THAN LID
      HEF=ZSS+AMIN1(PRISE(WSS,JSTAB,QHT),HLID-ZSS)
    9 CONTINUE
      EFFHGT=HEF
      WS=WSMPS*3600./1609.3
C     NOW WS IS CORRECTED FOR WIND SPEED AT STACK HEIGHT
      H=ZSS
      Z=0.
      DELH=HEF-H
      HMIX=HLID
      XSS=XS(NS)
      YSS=-YS(NS)
C       SOURCE DATA IS STORED IN PLANTSIM WITH +Y UP.
C*****************************************************************
      CALL CRDNT(XR (NR), YR (NR), XSS,YSS,WD,X,Y)
      IF(X.LE.0.)GOTO 45
      SY=SIGY(X)
      SZ=SIGZ(X)
      CALL REFLCT(X)
      IF(WS.EQ.0.)  WS=1.
      CHIPT=TRANSS(X)*QPOL*.3812E-4/WS
C
C     IF(CHIPT.LE..0001) GOTO406
C     MMM=MMM+1
C     IF(MMM.GT.10)GOTO406
```

```
C       PRINT 405,QPOL,QHT,WS,WD,ZSS,HLID,XSS,YSS,XRR(NR),YRR(NR),X,Y,SY,
C      1SZ,CHIPT
C   405 FORMAT(10X,'QPOL,QHT,WS,WD,ZSS,HLID,XSS,YSS,XRR(NR),YRR(NR),X,Y,SY
C      1,SZ,CHIPT',/6E10.2,/9E10.2/)
C
  406 CONTINUE
      CHI(NR)=CHI(NR) +CHIPT
   45 CONTINUE
      RECDAT(NR)=CHI(NR)
   50 CONTINUE
      WS=WSA(1)
C       PRINT OUT WS,WD,HLID,IR,AND CHI(NR),NR=1,NRE
      PRINT 401,WS,WD,HLID,IR
  401 FORMAT(        ///,10X,'WS, WD, HLID,HOUR =' 3F10.1,I5/ )
      PRINT 402
  402 FORMAT(11X,'RECEPTOR        SO2 CONCENTRATIONS(PPM)'  )
      PRINT 403,(NR,CHI(NR),NR=1,NRE )
  403 FORMAT(10X,I4,3X,F10.3)
      RETURN
      END
      FUNCTION PRISE(WS,JSTAB,QHEAT)
C
      REAL A(7)
      REAL XP(7)
      COMMON/RISE/ZSS
     ,,WSMPS
      DATA A/2.65,2.65,2.65,1.08,.68,.68,.68/
      DATA XP/.3,.3,.2,.3,.4,.5,.6/
      DATA KPR/1/
      JST=JSTAB
      IF(ZSS.GT.200..AND.JST.LE.3)  JST=4
C       STABILITY CONDITIONS HAVE BEEN LIMITED TO NEUTRAL AND STABLE
C           FOR STACKS ABOVE 200 FEET(THIS INCLUDES ALL CECO PLANTS)
C
C
      WSMPS=WS*1609.3/3600.
      WSMPS=WSMPS*(ZSS/50.)**XP(JST)
C       A WIND SPEED PROFILE HAS BEEN APPROXIMATED BY POWER LAW
C           AFTER BROOKHAVEN WORK
C           ALL TAMS ARBITRARILY ASSUMED AT 50 FEET<  CORRECTION
C           SHOULD BE MADE FOR TAM 3 AT 120 FEET
C
C     NO DOWNWASH EFFECTS OR LOOPING HAVE BEEN INCLUDED.
C
      IF(WSMPS.LT.2.0)  WSMPS=2.0
C     TVA DATA INDICATES CARSON MOSES AND OTHER  FORMULAE
C           OVERPREDICT PLUME RISE FOR WS .LT.2 OR 3 M/S
C           THUS A MINIMUM WINDSPEED OF 2 M/SEC HAS BEEN SET FOR PRISE
      QKCS=QHEAT*7.0
C       QHEAT IN TH/HR
      PRISE=A(JST  )*5.35*3.281*SQRT(QKCS)/WSMPS
C     PRISE IS IN FEET
      IF (KPR.GT.0) GO TO 500
      KPR=KPR+1
      WMPS=WSMPS
      PRINT 401, JST,WMPS,QHEAT,PRISE
  401 FORMAT(15X,'IN PRISE.  JST,WMPS,QHEAT,PRISE=    ', I2,5X,3E12.2)
  500 CONTINUE
      RETURN
      END
      FUNCTION TRANSS(T)
      COMMON/MET/WD,WS,HMIX,H,JSTAB,DELH,EFFHGT
      COMMON/SIG/SY,SZ
      COMMON/COORD/Y,Z,NF
C     CALCULATES (CHI*WS/QSO2)/(.38E-4)
C   GROUND LEVEL CONCENTRATIONS ONLY
C  CONSIDERS REFLECTION FROM GROUND AND LID
C
C
      TRANSC=0.
      TRANSD=0.
      TRANSE=0.
      TRANSF=0.
      ZKEEP=0.
      IF(EFFHGT.GT.HMIX) DELH=HMIX-H
      IF(NF.EQ.0) GO TO 5
      IF(NF.EQ.1) GO TO 6
      IF(NF.EQ.2) GO TO 7
      IF(NF.EQ.3) GO TO 10
    7 Z=
     1((2.*HMIX+H+DELH)/5280.)-ZKEEP
      TRANSE=TPLUME(T)
      Z=((4.*HMIX-H-DELH)/5280.)-ZKEEP
```

```
      TRANSF=TPLUME(T)
    6 Z=((2.*HMIX-H-DELH)/5280.)-ZKEEP
      TRANSC=TPLUME(T)
      Z=((2.*HMIX+H+DELH)/5280.)-ZKEEP
      TRANSD=TPLUME(T)
    5 Z=(H+DELH)/5280.-ZKEEP
       TRANSA=2.*TPLUME(T)
      Z=(2.*HMIX-(H+DELH))/5280.-ZKEEP
      TRANSB=TPLUME(T)
      TRANSS=TRANSA+TRANSB+TRANSC+TRANSD+TRANSE+TRANSF
      Z=ZKEEP
      RETURN
   10 CONTINUE
      HM=HMIX/5280.
      BOT=SQRT(6.284)*SY*HM
   11 EXPON=(-.5*(Y/SY)**2)
      IF(ABS(EXPON).GT.20.) GO TO 20
   12 TOP=EXP(EXPON)
      TRANSS=TOP/BOT
      RETURN
   20 TRANSS=0.
      RETURN
      END
      FUNCTION TPLUME(T)
C
      COMMON/SIG/SY,SZ
      COMMON/COORD/Y,Z,NF
C
      DATA COEF/6.284/
C
      T1=0.
      T2=Y/SY
      T3=Z/SZ
      EXPON=-(T1*T1+T2*T2+T3*T3)*.5
      IF(ABS(EXPON).GT.20)GO TO 2
      TOP=EXP(EXPON)
      BOT=COEF*   SY*SZ
      FT=TOP/BOT
      GO TO 5
    2 CONTINUE
      FT=0.
    5 CONTINUE
      TPLUME=FT
      RETURN
      END
      FUNCTIONSIGY(X)
C     DISPERSION COEFFICIENTS BASED ON TURNER WORKBOOK.
C     J=1,2,3,4,5 ARE CLASSES B,C,D,E,F
C     X AND SIGX ARE IN MILES
      DIMENSION A(5),B(5)
      COMMON/MET/WD,WS,HMIX,H,JSTAB,DELH,EFFHGT
      DATA A/155.,100.,68.,50.,34./,    B/.91,.92,.93,.90,.93/
      XX=X*1.609
       J=JSTAB
      SIGY=A(J)*XX**B(J)
      SIGY=SIGY*.000889
C     .000889=(1/1609.)*(6**.2)
C     WHERE THE SIGMAS GIVEN IN THE WORKBOOK FOR
C     TEN MINUTE SAMPLINGS ARE SCALED TO 1 HOUR
C     AVERAGES BY A FACTOR OF 1.43
      RETURN
      END
      FUNCTION SIGZ(X)
C     DISPERSION COEF. BASED ON TURNERS WORKBOOK
C     J=1,2,3,4,5 ARE CLASSES B,C,D,E,F
C     X AND SIGZ ARE IN MILES
      DIMENSION A(3,5),B(3,5)
      COMMON/MET/WD,WS,HMIX,H,JSTAB,DELH,EFFHGT
      DATA A/110.,110.,110.,60.,60.,60.,33.,33.,40.,21.5,21.5,36.,14.,
     114.,23.5/
      DATA B/1.,1.09,1.09,.92,.92,.92,.80,.61,.53,.77,.56,.35,.78,.53,
     1.30/
C     A(I,J),B(I,J).I=1,2,3 FOR X IN (0,1.),(1.,10.),(10.,100.)
      XX=X*1.609
      I=1
      IF(XX.GT.1.)I=2
      IF(XX.GT.10.) I=3
      J=JSTAB
      SIGZ=(A(I,J)*XX**B(I,J))*.000889
C     .000889=(1//1609)*(60/10)**.2
      RETURN
      END
      SUBROUTINE REFLCT(X)
```

```
      DIMENSION D(3),XC(3)
      COMMON/COORD/Y,Z,NF
      COMMON/MET/WD,WS,HMIX,H,JSTAB,DELH,EFFHGT
      FACT=1./(5280.*2.15)
      D(1)=(2.*HMIX-EFFHGT)*FACT
      D(2)=(2.*HMIX+EFFHGT)*FACT
      D(3)=(4.*HMIX-EFFHGT)*FACT
      NF=0
C      CALCULATE DISTANCES AT WHICH NEW IMAGES
C            BECOME IMPORTANT
      DO 39 N=1,3
C*********************************************************************
      IF(D(N).LE.0.) GOTO 39
      GO TO (31,32,33,34,35),JSTAB
C     B STABILITY
   31 XC(N)=(D(N)/(110.*.000889))**(1/1.09)
      GO TO 39
C     C STABILITY
   32 XC(N)=(D(N)/( 60.*.000889))**(1/.92)
      GO TO 39
C     D STABILITY
   33 XC(N)=(D(N)/( 33.*.000889))**(1/.61)
      GO TO 39
C     E STABILITY
   34 XC(N)=(D(N)/(21.5*.000889))**(1/.56)
      GO TO 39
C     F STABILITY
   35 XC(N)=(D(N)/( 14.*.000889))**(1/.53)
   39 CONTINUE
      IF(X.GE.XC(1)) NF=1
      IF(X.GE.XC(2)) NF=2
      IF(X.GE.XC(3)) NF=3
      RETURN
      END
      SUBROUTINE CRDNT(XR,YR,XS,YS,T,X,Y)
C     T IS WIND DIRECTION(DEG)
C      COORD SYSTEM IS +X LEFT AND +Y DOWN.
      TT=T
      SI=270.-T
      T=T*(3.1416/180.)
      SI=SI*(3.1416/180.)
      XR=-XR
      XS=-XS
      YR=-YR
      YS=-YS
      XRR=XR-XS+.01
      YRR=YR-YS+.01
      H=SQRT (XRR**2+YRR**2)
      ARG= ABS(YRR/XRR)
      PHI= ATAN(ARG)
      IF(XRR)1,2,2
    1 IF(YRR)3,4,4
    2 IF(YRR)5,6,6
    5 PHI=-PHI
      GO TO 6
    4 PHI=3.1416-PHI
      GO TO 6
    3 PHI=3.1416+PHI
    6 CONTINUE
C
      THETA=PHI-SI
      COST= COS(THETA)
      SINT= SIN(THETA)
      X=H* COS(THETA)
      XR=-XR
      XS=-XS
      Y= ABS(H* SIN(THETA))
      T=TT
      YR=-YR
      YS=-YS
      RETURN
      END
      SUBROUTINE LCSSET(OUTDAT,NOBMAX,NPMAX,NSRC,IERR)
C
      REAL*8 BLK
      COMMON/BLKSTR/BLK(1)
C
      IERR=1
      CALL BULK(125000)
      CALL POINTR(BLK,24*8*NSRC,3)
      CALL PUTBLK('A        ',NOBMAX*NPMAX,4)
      CALL PUTBLK('A        ')
      IF(IPTERR(DUMMY).GT.0)CALL ABEND
      CALL LCS(BLK(IGET('A        ')),OUTDAT,NOBMAX,NPMAX)
```

```
      IF(IPTERR(DUMMY).GT.0)RETURN
      CALL PUTPNT('EMIS      ',24*4*NSRC,4)
      CALL PUTPNT('HEFF      ',24*4*NSRC,4)
      IF(IPTERR(DUMMY).GT.0)RETURN
      CALL IPLUME(BLK(IGET('EMIS      ')),BLK(IGET('HEFF      ')))
      IERR=0
      RETURN
C
      ENTRY RELLCS
C
      CALL FREE
      RETURN
      END
      SUBROUTINE LCS(A,OUTDAT,NORMAX,NPMAX)
      REAL*8 TITLE(1),DATE(1),STA(1),PARAM(1)
      REAL*4 A(NOPMAX,1),OUTDAT(1),HOUR(1)
C
      RETURN
C
      ENTRY PUTLCS(OUTDAT,N)
C
      DO 10 J=1,NPMAX
      A(N,J)=OUTDAT(J)
   10 CONTINUE
      RETURN
C
      ENTRY GETLCS(OUTDAT,N)
C
      DO 20 J=1,NPMAX
      OUTDAT(J)=A(N,J)
   20 CONTINUE
      RETURN
C
      ENTRY PRINT(NOBS,NPARAM,TITLE,DATE,HOUR,STA,PARAM)
C
      CALL TWOOPT(A,NOBS,NPARAM,TITLE,DATE,HOUR,STA,PARAM,NOBMAX)
      RETURN
      END
      SUBROUTINE TWOOPT(ARAY,ICT,JCT,TITLE,ROWT1,ROWT2,COLT1,COLT2,
     1ICTMAX)
C                                TWOOPT PRINTS A TWO-DIMENSIONAL ARRAY.  IT
C                               PRINTS A TITLE LINE AND TWO LINES OF COLUMN
C                               TITLES, WHICH ARE FOLLOWED BY 50 LINES UP
C                               TO TEN COLUMNS WIDE PLUS THE INDEX COLUMN,
C                               WHICH CONTAINS DATE AND HOUR.
C                               ARGUMENTS ARE -
C
C                                     ARAY - THE TWO-DIMENSIONAL ARRAY
C                                            TO BE PRINTED.
C                                      ICT - NUMBER OF ROWS IN ARRAY.
C                                      JCT - NUMBER OF COLUMNS IN ARRAY.
C
C                                    TITLE - PAGE TITLE (10A8).
C                                    ROWT1 - ARRAY DIMENSIONED BY ICT, CON-
C                                            TAINING DATE FOR EACH ROW
C                                            (A8).
C                                    ROWT2 - ARRAY DIMENSIONED BY ICT, CON-
C                                            TAINING HOUR FOR EACH ROW
C                                            (A4, PRINTED AS A2).
C                                    COLT1 - ARRAY DIMENSIONED BY JCT, CON-
C                                            TAINING FIRST LINE OF TITLE
C                                            FOR EACH COLUMN (A8).
C                                    COLT2 - ARRAY DIMENSIONED BY JCT, CON-
C                                            TAINING SECOND LINE OF TITLE
C                                            FOR EACH COLUMN (A8).
      REAL*8 TITLE(10),ROWT1(1),COLT1(1),COLT2(1)
      REAL*4 ARAY(ICTMAX,1),ROWT2(1)
      II=1
      IJ=1
      IMAX=0
      JMAX=0
      MAX=0
      IFLAG=0
   20 IEND=JCT
   40 MIN=MAX+1
      MAX=MINO(MAX+10,IEND)
      IF(IFLAG.NE.0) GO TO 47
   45 NROW=0
   46 CONTINUE
      PRINT 210,(TITLE(I),I=1,10)
   47 PRINT 220
      PRINT 230,(COLT1(JC),JC=MIN,MAX)
```

```
      PRINT 230,(COLT2(JC),JC=MIN,MAX)
      PRINT 230
      IMAX=MINC(II+52,ICT)
      IDIF=IMAX-II+5
      NROW=NROW+IDIF
      JMAX=MINO(IJ+9,JCT)
      DO 65 I=II,IMAX
      PRINT 240,ROWT1(I),ROWT2(I),(ARAY(I,J),J=IJ,JMAX)
   65 CONTINUE
      IF(IMAX.GE.ICT) GO TO 70
      II=IMAX+1
      GO TO 46
   70 IF(JMAX.GE.JCT) GO TO 130
   80 II=1
      IJ=JMAX+1
      IF((NROW+IDIF-57).GT.0) GO TO 84
   82 IFLAG=1
      GO TO 40
   84 IFLAG=0
      GO TO 40
  130 CONTINUE
      DO 140 I=1,25
      PRINT 150
  150 FORMAT(' ')
  140 CONTINUE
      RETURN
  210 FORMAT(1H1,19X,10A8)
  220 FORMAT(///)
  230 FORMAT(12X,10(4X,A8))
  240 FORMAT(1X,A8,1X,A2,10F12.3)
      END
APDATA: PROC OPTIONS(MAIN);
    DCL DATE_LIMITS FIXED BINARY INITIAL(0),
        LOWER_DATE(DATE_LIMITS) CHAR(8) CONTROLLED,
        UPPER_DATE(DATE_LIMITS) CHAR(8) CONTROLLED;
DCL     1 FILE_HEADER,
            2 STATIONS FIXED BINARY,
            2 PARAMS FIXED BINARY,
            2 STARTING_DATE CHAR(6),
            2 ENDING_DATE CHAR(6);
DCL     1 FILE_STATION (FILE_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 FILE_PARAM (FILE_HEADER.PARAMS) CONTROLLED,
            2 NAME CHAR(8);
DCL     1 TODAY      CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (FILE_HEADER.PARAMS);
DCL     1 YESTERDAY CONTROLLED,
            2 DATE CHAR(6),
            2 HOUR(24),
                3 PARAM (FILE_HEADER.PARAMS);
DCL      MET_POINTER(5) FIXED BINARY;
DCL      TAM_POINTER (8,3) FIXED BINARY;
DCL      KTAM (8);
DCL      NALT (8) INITIAL (7,3,2,3,8,4,1,5);
DCL      SOURCES          FIXED BINARY;
DCL     1 SOURCE_HEADER,
            2 TITLE (10) CHAR(8),
            2 STATIONS FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL     1 SOURCE_STATION CONTROLLED,
            2 NAME (SOURCE_HEADER.STATIONS) CHAR(8),
            2 XS        (SOURCE_HEADER.STATIONS),
            2 YS        (SOURCE_HEADER.STATIONS),
            2 ZS        (SOURCE_HEADER.STATIONS,4),
            2 PERCENT (SOURCE_HEADER.STATIONS,4);
DCL     1 SOURCE_PARAM (SOURCE_HEADER.PARAMS) CONTROLLED,
            2 STA_NAME CHAR(8),
            2 NAME CHAR(8);
DCL     1 SOURCE_DATA CONTROLLED,
            2 DATE CHAR(8),
            2 HOUR (24),
                3 VALUE (SOURCE_HEADER.PARAMS);
DCL      Q_SO2 (SOURCES) CONTROLLED, Q_HEAT (SOURCES) CONTROLLED;
DCL      RECEPTORS     FIXED BINARY;
DCL     1 REC_DATA CONTROLLED,
            2 STATION (RECEPTORS) CHAR(8),
            2 XR (RECEPTORS),
            2 YR (RECEPTORS),
            2 ZR (RECEPTORS),
```

```
            2 SO2 (RECEPTORS);
DCL      OUT_STATIONS FIXED BINARY,
         OUT_PARAMS   FIXED BINARY;
DCL      1 OUTPUT_HEADER,
            2 LABEL (10) CHAR(8),
            2 STATIONS FIXED BINARY,
            2 PARAMS FIXED BINARY,
            2 NUMBER OBS;
DCL      1 OUTPUT_STATION (OUTPUT_HEADER.STATIONS) CONTROLLED,
            2 NAME CHAR(8),
            2 POINTER FIXED BINARY,
            2 PARAMS FIXED BINARY;
DCL      1 OUTPUT_PARAM (OUTPUT_HEADER.PARAMS) CONTROLLED,
            2 STATION CHAR(8),
            2 NAME CHAR(8);
DCL      1 DUM_OUTPUT_PARAM(100)  CONTROLLED,
            2 STATION CHAR(8),
            2 NAME CHAR(8);
DCL      1 OUT_PARAM (OUT_PARAMS) CONTROLLED,
            2 POINTER FIXED BINARY,
            2 DELAY FIXED DECIMAL (2);
DCL      1 DUM_OUT_PARAM (100)  CONTROLLED,
            2 POINTER FIXED BINARY,
            2 DELAY FIXED DECIMAL (2);
DCL      OUTPUT_DATA_DATE CHAR(8), OUTPUT_DATA_HOUR CHAR(4),
         OUTPUT_DATA_PARAM(OUTPUT_HEADER.PARAMS) CONTROLLED;
DCL      PRINT_TITLE (10) FLOAT DECIMAL (16),
         PRINT_DATE (NUMBER_OBS_MAX) CONTROLLED FLOAT DECIMAL (16),
         PRINT_HOUR (NUMBER_OBS_MAX) CONTROLLED FLOAT DECIMAL ( 6),
         PRINT_STATION (OUTPUT_HEADER.PARAMS) CONTROLLED
         FLOAT DECIMAL (16),
         PRINT_NAME    (OUTPUT_HEADER.PARAMS) CONTROLLED
         FLOAT DECIMAL (16);
DCL      CARD CHAR(80);
DCL      OPT1 CHAR(3), OPT2 CHAR(3), OPT3  CHAR(3), OPTLIST CHAR(50);
         READ FILE(MASTER) INTO (FILE_HEADER);
         ALLOCATE FILE_STATION;
         READ FILE(MASTER) INTO (FILE_STATION);
         ALLOCATE FILE_PARAM;
         READ FILE(MASTER) INTO (FILE_PARAM);
         ALLOCATE TODAY; ALLOCATE YESTERDAY;
         READ FILE (SOURCE) INTO (SOURCE_HEADER);
         SOURCES=SOURCE_HEADER.STATIONS;
         ALLOCATE SOURCE_STATION;
         READ FILE (SOURCE) INTO (SOURCE_STATION);
         ALLOCATE SOURCE_PARAM;
         READ FILE (SOURCE) INTO (SOURCE_PARAM);
         ALLOCATE SOURCE_DATA; ALLOCATE Q_SO2; ALLOCATE Q_HEAT;
NEXT:    /* BEGIN NEW CASE */
         CLOSE FILE(MASTER);
         ON ENDFILE(MASTER)  BEGIN;
         PUT LIST('END OF FILE SENSED ON MASTER')SKIP;
         GO TO NEXT ;END;
         READ FILE (MASTER) IGNORE(3);
         CLOSE FILE (SOURCE);
         ON ENDFILE(SOURCE) BEGIN;
         PUT LIST('END OF FILE SENSED ON SOURCE')SKIP;
         GO TO NEXT; END;
         READ FILE (SOURCE) IGNORE(3);
         CLOSE FILE(CARDS);
GETCARD: GET EDIT(CARD)(A(80));
         WRITE FILE(CARDS) FROM (CARD);
         IF INDEX(CARD,'END ')¬=0 THEN GO TO END;
         IF INDEX(CARD,'STOP')¬=0 THEN GO TO STOP;
         GO TO GETCARD;
END:     CLOSE FILE(CARDS);
         CALL INPUT;
         CALL SEARCH;
         GO TO NEXT;
INPUT:   PROC;
DCL      PARAMS FIXED BINARY, POINT FIXED BINARY;
DCL      DELAY FIXED DECIMAL (2);
DCL      RECEPTOR CHAR(8), SOURCE CHAR(8);
    DCL  STATION CHAR(8),PARAM CHAR(8);
    DCL  L1 CHAR(2),L2 CHAR(2),L3 CHAR(2),
         U1 CHAR(2),U2 CHAR(2),U3 CHAR(2);
         ON NAME(CARDS) BEGIN;
         PUT LIST('THE FOLLOWING DATA IS OUT OF ORDER: ',DATAFIELD)SKIP;
         GO TO NEXT;END;
         ON ENDFILE(CARDS)  BEGIN;
         PUT LIST('UNEXPECTED EOF ON CARD INPUT')SKIP;
         GO TO NEXT;END;
         PUT LIST('INPUT DATA')PAGE;
```

```
        GET FILE(CARDS) EDIT(LABEL)(10A(8));
        DO I=1 TO 10;
        UNSPEC(PRINT_TITLE(I))=UNSPEC(LABEL(I));
        END;
        PUT EDIT(LABEL)(10A(8))SKIP;
        GET FILE (CARDS) DATA(OPTLIST);
        IF INDEX(OPTLIST,'FILE ')>0 THEN OPT1 ='YES';ELSE OPT1 ='NO ';
        IF INDEX(OPTLIST,'PRINT')>0 THEN OPT2 ='YES';ELSE OPT2 ='NO ';
        IF INDEX(OPTLIST,'FORT ')>0 THEN OPT3 ='YES';ELSE OPT3 ='NO ';
        PUT LIST('FILE=',OPT1,' PRINT=',OPT2,' FORT=',OPT3)SKIP;
        GET FILE(CARDS) DATA(DATE_LIMITS);
        FREE      LOWER_DATE; FREE      UPPER_DATE;
        IF DATE_LIMITS=0 THEN BEGIN;
        DATE_LIMITS=1;
        ALLOCATE LOWER_DATE; ALLOCATE UPPER_DATE;
        LOWER_DATE(1)=STARTING_DATE;
        UPPER_DATE(1)=ENDING_DATE;
        PUT DATA(LOWER_DATE(1),UPPER_DATE(1))SKIP;
        GO TO SKIP1;END;
        ELSE BEGIN;
        ALLOCATE LOWER_DATE; ALLOCATE UPPER_DATE;
        GET FILE(CARDS) LIST
                ((LOWER_DATE(I),UPPER_DATE(I) DO I=1 TO DATE_LIMITS));
        DO I=1 TO DATE_LIMITS;
        PUT DATA(LOWER_DATE(I),UPPER_DATE(I))SKIP;
        L1=SUBSTR(LOWER_DATE(I),1,2)  ;
        L2=SUBSTR(LOWER_DATE(I),4,2)  ;
        L3=SUBSTR(LOWER_DATE(I),7,2)  ;
        U1=SUBSTR(UPPER_DATE(I),1,2)  ;
        U2=SUBSTR(UPPER_DATE(I),4,2)  ;
        U3=SUBSTR(UPPER_DATE(I),7,2)  ;
        LOWER_DATE(I)=L1||L2||L3; UPPER_DATE(I)=U1||U2||U3;
        END;END;
SKIP1:
        STATION='MIDWAY'; PARAM='WIND_DIR'; CALL FIND;
        IF POINT=0 THEN GO TO NEXT;
        MET_POINTER(1)=POINT;
        STATION='MIDWAY'; PARAM='WIND_VEL'; CALL FIND;
        IF POINT=0 THEN GO TO NEXT;
        MET_POINTER(2)=POINT;
        STATION='TURNER'; PARAM='TURNER_C'; CALL FIND;
        MET_POINTER(3)=POINT;
        IF POINT=0 THEN GO TO NEXT;
        STATION='NORCO '; PARAM='MIX_HT  '; CALL FIND;
        IF POINT=0 THEN GO TO NEXT;
        MET_POINTER(4)=POINT;
        STATION='MIDWAY'; PARAM='TEMP    '; CALL FIND;
        IF POINT=0 THEN GO TO NEXT;
        MET_POINTER(5)=POINT;
DCL     ICHAR (8) CHAR (1) INITIAL ('1','2','3','4','5','6','7','8');
        DO I=1 TO 8;
        STATION='TAM_'||ICHAR(I);
        PARAM='WIND_DIR';
        CALL FIND;
        IF POINT=0 THEN GO TO NEXT;
        TAM_POINTER(I,1)=POINT;
        PARAM='WIND_VEL';
        CALL FIND;
        IF POINT=0 THEN GO TO NEXT;
        TAM_POINTER(I,2)=POINT;
        PARAM='SO2';
        CALL FIND;
        IF POINT=0 THEN GO TO NEXT;
        TAM_POINTER(I,3)=POINT;
        END; /* TAM LOOP */
        PUT LIST('RECEPTOR REQUESTS')SKIP;
        GET FILE(CARDS) DATA(RECEPTORS);
        IF RECEPTORS=0 THEN BEGIN;
        PUT LIST('NO RECEPTOR REQUESTS')SKIP;
        GO TO NEXT; END;
        FREE REC_DATA; ALLOCATE REC_DATA; KTAM=0;
        DO J=1 TO RECEPTORS;
        GET FILE(CARDS) DATA(RECEPTOR,X,Y,Z);
        PUT DATA(RECEPTOR,X,Y,Z )SKIP;
DCL     ICHA CHAR (1);
        ICHA =SUBSTR(RECEPTOR,5,1);
        I=ICHA;
        PUT DATA (J,ICHA)SKIP;
        KTAM(I)=1;
        REC_DATA.STATION(J)=RECEPTOR;
        REC_DATA.XR(J)=X;
        REC_DATA.YR(J)=Y;
        REC_DATA.ZR(J)=Z;
```

```
        END;
        PUT DATA (KTAM)SKIP;
        PUT LIST('OUT STATION REQUESTS')SKIP;
        GET FILE(CARDS) DATA (OUT_STATIONS);
        OUTPUT_HEADER.STATIONS=OUT_STATIONS+RECEPTORS;
        FREE     OUTPUT_STATION;
        ALLOCATE OUTPUT_STATION;
        OUT_PARAMS=0;
        IF OUT_STATIONS=0 THEN BEGIN;
        PUT LIST('NO OUT STATIONS REQUESTED')SKIP;
        GO TO SKIP2; END;
        ALLOCATE DUM_OUT_PARAM;
        ALLOCATE DUM_OUTPUT_PARAM;
        DO I=1 TO OUT_STATIONS;
        GET FILE(CARDS) DATA(STATION,PARAMS);
        PUT DATA(STATION)SKIP;
        IF PARAMS=0 THEN BEGIN;
        PUT LIST('NO PARAMETER REQUESTS')SKIP;
        GO TO NEXT;END;
        IF OUT_PARAMS+PARAMS>100 THEN BEGIN;
        PUT LIST('NUMBER OF PARAMETERS GREATER THAN 100')SKIP;
        GO TO NEXT;END;
        OUTPUT_STATION(I).NAME=STATION;
        OUTPUT_STATION(I).POINTER=OUT_PARAMS;
        OUTPUT_STATION(I).PARAMS=PARAMS;
        DO J=1 TO PARAMS;
        DELAY=0;
        GET FILE(CARDS) DATA(PARAM,DELAY);
        PUT DATA(PARAM,DELAY)SKIP;
        DUM_OUTPUT_PARAM(OUT_PARAMS+J).STATION=STATION;
        DUM_OUTPUT_PARAM(OUT_PARAMS+J).NAME=PARAM;
        CALL FIND;
        IF POINT=0 THEN GO TO NEXT;
        DUM_OUT_PARAM(OUT_PARAMS+J).POINTER=POINT;
        DUM_OUT_PARAM(OUT_PARAMS+J).DELAY=DELAY;
        END; /* PARAM LOOP */
        OUT_PARAMS=OUT_PARAMS+PARAMS;
        END; /*STATION LOOP */
SKIP2:  OUTPUT_HEADER.PARAMS=OUT_PARAMS+RECEPTORS;
        FREE     OUTPUT_PARAM;
        ALLOCATE OUTPUT_PARAM;
        FREE     OUT_PARAM;
        ALLOCATE OUT_PARAM;
        FREE     PRINT_STATION;
        ALLOCATE PRINT_STATION;
        FREE     PRINT_NAME;
        ALLOCATE PRINT_NAME;
        DO J=1 TO OUT_PARAMS;
        OUTPUT_PARAM(J)=DUM_OUTPUT_PARAM(J);
        OUT_PARAM(J)=DUM_OUT_PARAM(J);
        STATION=OUTPUT_PARAM(J).STATION;
        PARAM  =OUTPUT_PARAM(J).NAME    ;
        UNSPEC(PRINT_STATION(J))=UNSPEC(STATION);
        UNSPEC(PRINT_NAME    (J))=UNSPEC(PARAM);
        END;
        FREE DUM_OUTPUT_PARAM;
        FREE DUM_OUT_PARAM;
        DO I=1 TO RECEPTORS;
        STATION=REC_DATA.STATION(I);
        PARAM='SO2';
        OUTPUT_STATION(OUT_STATIONS+I).NAME=STATION;
        OUTPUT_STATION(OUT_STATIONS+I).POINTER=OUT_PARAMS+I-1;
        OUTPUT_STATION(OUT_STATIONS+I).PARAMS=1;
        OUTPUT_PARAM(OUT_PARAMS+I).STATION=STATION;
        OUTPUT_PARAM(OUT_PARAMS+I).NAME=PARAM;
        UNSPEC(PRINT_STATION(OUT_PARAMS+I))=UNSPEC(STATION);
        UNSPEC(PRINT_NAME    (OUT_PARAMS+I))=UNSPEC(PARAM  );
        END;
        CLOSE FILE(CARDS);
        RETURN;
FIND:   PROC;
        DO N=1 TO FILE_HEADER.STATIONS;
        IF STATION=FILE_STATION(N).NAME THEN BEGIN;
        DO M=1 TO FILE_STATION(N).PARAMS;
        IF PARAM=FILE_PARAM(FILE_STATION(N).POINTER+M).NAME
        THEN GO TO FINDIT;
        END;/*M LOOP*/ END;/* BEGIN BLOCK*/
        END; /*N LOOP*/
        PUT LIST('THE COMBINATION STATION NAME ',STATION,
        ' AND PARAMETER NAME ',PARAM,' CANNOT BE FOUND')SKIP;
        POINT=0; RETURN;
FINDIT: POINT=FILE_STATION(N).POINTER+M;
        PUT DATA(STATION,PARAM,POINT)SKIP;
```

```
        RETURN;
        END FIND;
        END INPUT;
SEARCH: PROC;
DCL     HOUR FIXED BINARY, HM1 FIXED BINARY;
DCL     DATE_KEY BIT(1) INITIAL('0'B), HOUR_KEY BIT(1) INITIAL ('0'B);
DCL     DATE_LIMIT FIXED BINARY, HOUR_LIMIT FIXED BINARY;
DCL     DELAY_HOUR FIXED BINARY;
DCL     YEAR CHAR(2), MONTH CHAR(2), DAY CHAR(2);
DCL     PLUMES EXT ENTRY, RETCODE FIXED BINARY EXT;
        DATE_LIMIT=1;
        NUMBER_OBS=0;
        FREE OUTPUT_DATA_PARAM; ALLOCATE OUTPUT_DATA_PARAM;
/*      SET MAX OBSERVATIONS - AT MOST ONE YEARS 'WORTH'          */
        NUMBER_OBS_MAX=MIN (250000/OUTPUT_HEADER.PARAMS,9000);
        PUT DATA(NUMBER_OBS_MAX)SKIP;
        FREE PRINT_DATE    ; ALLOCATE PRINT_DATE;
        FREE PRINT_HOUR    ; ALLOCATE PRINT_HOUR;
        CALL LCSSET(OUTPUT_DATA_PARAM(1),NUMBER_OBS_MAX,
        OUTPUT_HEADER.PARAMS,SOURCES,IERR);
        IF IERR>0 THEN BEGIN;
        PUT LIST ('UNABLE TO ALLOCATE BULK STORAGE')SKIP;
        RETURN; END;
        ON ENDFILE(MASTER)  BEGIN;
        PUT LIST('END OF FILE SENSED ON MASTER')SKIP;
        GO TO WRITE;END;
        PUT LIST('SEARCH STARTED  AT TIME ',TIME)SKIP;
        READ FILE(MASTER) INTO (TODAY);
        YESTERDAY=TODAY;
DATES:  /* LOOP ON DATES - TEST BETWEEN DATE LIMITS */
        IF TODAY.DATE=LOWER_DATE(DATE_LIMIT) THEN BEGIN;
        M=0; DATE_KEY='1'B; END;
        IF TODAY.DATE=UPPER_DATE(DATE_LIMIT) THEN BEGIN;
        DATE_KEY='0'B;DATE_LIMIT=DATE_LIMIT+1; GO TO HOURS;
        END;
        IF DATE_KEY='0'B THEN GO TO NEWDATE;
HOURS:  /* LOOP ON HOURS */
READ:   READ FILE (SOURCE) INTO (SOURCE_DATA);
        YEAR =SUBSTR(TODAY.DATE,1,2);
        MONTH=SUBSTR(TODAY.DATE,3,2);
        DAY  =SUBSTR(TODAY.DATE,5,2);
        OUTPUT_DATA_DATE=YEAR||'/'||MONTH||'/'||DAY;
        IF SOURCE_DATA.DATE¬=OUTPUT_DATA_DATE THEN GO TO READ;
        DO HOUR=1 TO 24;
        M=M+1;
        STAB        =TODAY(HOUR).PARAM(MET_POINTER(3));
        HMIX        =TODAY(HOUR).PARAM(MET_POINTER(4));
        TEMP        =TODAY(HOUR).PARAM(MET_POINTER(5));
        REC_DATA.SO2=0.;
        DO J=1 TO SOURCES;
        Q_SO2(J)=SOURCE_DATA.VALUE(HOUR,2*(J-1)+1);
        Q_HEAT(J)=SOURCE_DATA.VALUE(HOUR,2*(J-1)+2);
        END;
DCL     WS (8), WD (8),  SOBS  (8);
        WD=0.; WS=0.;            SOBS=0.;
        J=0;
        DO I=1 TO 8;
        IF KTAM(I)¬=0 THEN BEGIN;
        J=J+1;
        SOBS(J)=TODAY(HOUR).PARAM(TAM_POINTER(I,3));
        WD(J)=TODAY(HOUR).PARAM(TAM_POINTER(I,1));
        IF WD(J)=1.E20 THEN
        WD(J)=TODAY(HOUR).PARAM(TAM_POINTER(NALT(I),1));
        IF WD(J)=1.E20 THEN
        WD(J)=TODAY(HOUR).PARAM(MET_POINTER(1));
        WS(J)=TODAY(HOUR).PARAM(TAM_POINTER(I,2));
        IF WS(J)=1.E20 THEN
        WS(J)=TODAY(HOUR).PARAM(TAM_POINTER(NALT(I),2));
        IF WS(J)=1.E20 THEN
        WS(J)=TODAY(HOUR).PARAM(MET_POINTER(2));
        END;
        END; /* RECEPTOR LOOP */
        CALL FORTRAN(
        PLUMES,DP,HOUR,TEMP,WS(1),WD(1),STAB,HMIX,
        SOURCES,XS(1),YS(1),ZS(1,1),
        PERCENT(1,1),Q_SO2(1),Q_HEAT(1),RECEPTORS,XR(1),YR(1),ZR(1),
        SO2(1),M,KTAM(I),SOBS(1));
        IF RETCODE=1 THEN BEGIN;
        PUT LIST('ERROR IN CCPUFF')SKIP;
        RETURN; END;
        /* STORE OUPUT PARAMETERS */
        HM1=HOUR-1;
        PUT STRING (OUTPUT_DATA_HOUR) EDIT (HM1 )(F(2),X(2));
```

```
        DO J=1 TO OUT_PARAMS;
        DELAY_HOUR=HOUR-OUT_PARAM(J).DELAY;
        IF DELAY_HOUR<1 THEN BEGIN;
        OUTPUT_DATA_PARAM(J)=
        YESTERDAY.PARAM(24+DELAY_HOUR,OUT_PARAM.POINTER(J));
        GO TO SKIP;END;
        ELSE   BEGIN;
        OUTPUT_DATA_PARAM(J)=
        TODAY.PARAM(DELAY_HOUR,OUT_PARAM.POINTER(J));    END;
 SKIP:  END; /* PARAM LOOP */
        DO I=1 TO RECEPTORS;
        OUTPUT_DATA_PARAM(OUT_PARAMS+I)=SO2(I);
        END; /* RECEPTOR LOOP */
        IF NUMBER_OBS+1>NUMBER_OBS_MAX THEN BEGIN;
        PUT LIST('MAXIMUM NUMBER OF OBSERVATIONS EXCEEDED')SKIP;
        GO TO WRITE; END;
        NUMBER_OBS=NUMBER_OBS+1;
        N=NUMBER_OBS;
        UNSPEC(PRINT_DATE(N))=UNSPEC(OUTPUT_DATA_DATE);
        UNSPEC(PRINT_HOUR(N))=UNSPEC(OUTPUT_DATA_HOUR);
        CALL PUTLCS(OUTPUT_DATA_PARAM(1),N);
        END; /* HOUR LOOP */
NEWDATE:IF DATE_LIMIT>DATE_LIMITS THEN GO TO WRITE;
        YESTERDAY=TODAY;
        READ FILE(MASTER) INTO (TODAY);
        GO TO DATES;
WRITE:  /* SEARCH FINISHED */
        PUT LIST('SEARCH FINISHED AT TIME ',TIME)SKIP;
        CLOSE FILE (MASTER);
        IF NUMBER_OBS=0 THEN BEGIN;
        PUT LIST('NO OBSERVATIONS WERE FOUND WITHIN BANDS')SKIP;
        RETURN;END;
        IF OPT1='NO ' THEN GO TO JUMP1;
        WRITE FILE(OUTPUT) FROM (OUTPUT_HEADER);
        WRITE FILE(OUTPUT) FROM (OUTPUT_STATION);
        WRITE FILE(OUTPUT) FROM (OUTPUT_PARAM);
        DO N=1 TO NUMBER_OBS;
        CALL GETLCS(OUTPUT_DATA_PARAM(1),N);
        WRITE FILE(OUTPUT) FROM (OUTPUT_DATA_PARAM);
        END;
        CLOSE FILE (OUTPUT);
JUMP1:  IF OPT2 ='NO ' THEN GO TO JUMP2;
/*      CLOSE FILE(SYSPRINT);          */
        CALL PRINT(NUMBER_OBS,OUTPUT_HEADER.PARAMS,
        PRINT_TITLE(1),PRINT_DATE(1),PRINT_HOUR(1),PRINT_STATION(1),
        PRINT_NAME(1));
JUMP2:  IF OPT3='NO ' THEN GO TO JUMP3;
        FREE PRINT_DATE; FREE PRINT_HOUR;
        PUT LIST('FORT OPTION NOT AVAILIBLE ')SKIP;
JUMP3:  CALL RELLCS;
        RETURN;
        END SEARCH;
STOP:   END APDATA;
```

```
      SUBROUTINE POINTR(BLK,MAXSIZ,JPRINT)
C
C     MANIPULATES POINTERS FOR VARIABLE STORAGE
C     ADJUSTS STORAGE IF STORAGE EXCEEDED
C     MULT=2  NON STANDARD INTEGER
C     MULT=4  STANDARD INTEGER OR REAL
C     MULT=8  NON STANDARD REAL
C
      REAL*8 BLK
      REAL*8 NAME,NAMLST,BLANK
      REAL NAM1,NAM2
      INTEGER ALLOC1,ALLOC2
C
      DIMENSION BLK(1)
      DIMENSION NAM1(2),NAM2(2)
C
      COMMON/LOCATE/ LSTM,MAXM,LSTB,MAXB,INDXM,INDXB,IPRINT
      COMMON/TABLES/NAMLST(100),LENLST(100),IPTLST(100),
     1LEN(100),MLT(100),NNAMS
      COMMON/BFLAGS/ IFLAG(100)
      COMMON/NAMBLK/NAME
      COMMON/PTERR/NPTERR
C
      EQUIVALENCE (NAME,NAM2)
C
      DATA BLANK/'        '/
      DATA MAXBLK/100000/,JUMP/1/
C
      NNAMS=0
      NPTERR=0
      DO 5 I=1,100
      NAMLST(I)=0
      IPTLST(I)=0
      LENLST(I)=0
      LEN(I)=0
      MLT(I)=0
      IFLAG(I)=0
    5 CONTINUE
      MAXM=MAXSIZ
      ILOC=ALLOC2(IERR,2*MAXM)
      IF(ILOC.EQ.0)GO TO 1040
      ILOCM=ILOC
      JLOC=LOCF(BLK)
      INDX =(ILOC-JLOC)/8
      INDXM=INDX
      LSTM=INDX+1
      MAXFM=MAXM
      MAXM=MAXM+INDX
      MAXB=MAXBLK
      ILOC=0
      IF(MAXB.EQ.0)GO TO 10
      ILOC=ALLOC1(IERR,2*MAXB)
      IF(ILOC.EQ.0.AND.JUMP.EQ.0)GO TO 1050
      IF(ILOC.EQ.0)MAXB=0
   10 CONTINUE
      ILOCB=ILOC
      INDX =(ILOC-JLOC)/8
      INDXB=INDX
      LSTB=INDX+1
      MAXFB=MAXB
      MAXB=MAXB+INDX
      CALL PRGSET(BLK)
      CALL REDSET(BLK)
      IPRINT=JPRINT
      DO 15 I=LSTM,MAXM
      BLK(I)=0
   15 CONTINUE
      IF(IPRINT.LT.2)GO TO 16
      LDUM=LSTM-INDXM
      MDUM=MAXM-INDXM
      PRINT 25,LDUM,MDUM
   16 CONTINUE
      IF(MAXFB.EQ.0)RETURN
      DO 20 I=LSTB,MAXB
      BLK(I)=0
   20 CONTINUE
      IF(IPRINT.LT.2)RETURN
      LDUM=LSTB-INDXB
      MDUM=MAXB-INDXB
      PRINT 25,LDUM,MDUM
   25 FORMAT('1POINTR HAS BEEN INITIALIZED'/
     1'0THE CONTAINER ARRAY HAS BEEN CLEARED FROM LOCATION'I10,
     1' TO LOCATION'I10)
```

```
      RETURN
C
      ENTRY BULK(MAXSIZ)
C
C
      JUMP=0
      MAXBLK=MAXSIZ
      RETURN
C
      ENTRY FREE
C
      IF(ILOCM.EQ.0)RETURN
      CALL FREE2(ILOCM,2*MAXFM)
      IF(ILOCB.EQ.0)RETURN
      CALL FREE1(ILOCB,2*MAXFB)
      RETURN
C
      ENTRY PUTPNT(NAM1,LENDUM,MULT)
C
C     PUT ARRAY NAME AND POINTER IN TABLE - ALLOCATE TO MAIN
C
      NAM2(1)=NAM1(1)
      NAM2(2)=NAM1(2)
      LENGTH=((LENDUM-1)*MULT+8)/8
      IF(LSTM+LENGTH-1.LE.MAXM)GO TO 30
C     STORAGE WILL BE EXCEEDED UNLESS GARBAGE IS CLEARED OUT
      CALL PURGE(DUMMY)
      IF(LSTM+LENGTH.GT.MAXM)GO TO 50
   30 CONTINUE
      IF(NNAMS+1.GT.100)CALL PURGE(DUMMY)
      IF(NNAMS+1.GT.100)CALL PURGEB(DUMMY)
      IF(NNAMS+1.GT.100)GO TO 1020
C     ENOUGH STORAGE - RECORD NAME AND POINTER IN TABLE
      NNAMS=NNAMS+1
      NAMLST(NNAMS)=NAME
      IPTLST(NNAMS)=LSTM
      LENLST(NNAMS)=LENGTH
      LEN(NNAMS)=LENDUM
      MLT(NNAMS)=MULT
      IFLAG(NNAMS)=0
      LSTM=LSTM+LENGTH
      IF(IPRINT.LT.2)RETURN
      IDUM=IPTLST(NNAMS)-INDXM
      PRINT 40,NAME,IDUM          ,LENLST(NNAMS),LEN(NNAMS),MLT(NNAMS),
     1IFLAG(NNAMS)
   40 FORMAT(17H0ENTRY FOR ARRAY A8,35H HAS BEEN MADE IN THE POINTER TAB
     1LE/9H POINTER=I10,9H ORIG LEN=I5,10H ORIG LEN=I5,6H MULT=I5 ,
     26H BULK=I5)
      RETURN
C
      ENTRY PUTBLK(NAM1,LENDUM,MULT)
C
C     PUT ARRAY NAME AND POINTER IN TABLE - ALLOCATE TO BULK
C
      NAM2(1)=NAM1(1)
      NAM2(2)=NAM1(2)
      LENGTH=((LENDUM-1)*MULT+8)/8
   50 CONTINUE
      IF(LSTB+LENGTH-1.LE.MAXB)GO TO 60
C     STORAGE WILL BE EXCEEDED UNLESS GARBAGE IS CLEARED OUT
      CALL PURGEB(DUMMY)
      IF(LSTB+LENGTH.GT.MAXB)GO TO 1000
   60 CONTINUE
      IF(NNAMS+1.GT.100)CALL PURGE(DUMMY)
      IF(NNAMS+1.GT.100)CALL PURGEB(DUMMY)
      IF(NNAMS+1.GT.100)GO TO 1020
C     ENOUGH STORAGE - RECORD NAME AND POINTER IN TABLE
      NNAMS=NNAMS+1
      NAMLST(NNAMS)=NAME
      IPTLST(NNAMS)=LSTB
      LENLST(NNAMS)=LENGTH
      LEN(NNAMS)=LENDUM
      MLT(NNAMS)=MULT
      IFLAG(NNAMS)=1
      LSTB=LSTB+LENGTH
      IF(IPRINT.LT.2)RETURN
      IDUM=IPTLST(NNAMS)-INDXB
      PRINT 40,NAME,IDUM          ,LENLST(NNAMS),LEN(NNAMS),MLT(NNAMS),
     1IFLAG(NNAMS)
      RETURN
C
      ENTRY WIPOUT(NAM1)
C
```

```
C     RECORD NAME IN PURGE TABLE
C
      CALL GETN(NAM1,N)
      IF(N.EQ.0)RETURN
      NAMLST(N)=BLANK
      IF(IPRINT.LT.2)RETURN
      PRINT 70,NAME
   70 FORMAT(7H0ARRAY A8,35H HAS BEEN PLACED IN THE PURGE TABLE)
      RETURN
C
      ENTRY DUMP(NAM1,LUMP)
C
C     DUMP VARIABLE DIMENSION ARRAY ON STANDARD OUTPUT
C
      IF(IPRINT.EQ.0.OR.IPRINT.EQ.2)RETURN
      CALL GETN(NAM1,N)
      IF(N.EQ.0)RETURN
      IPOINT=IPTLST(N)
      IMINUS=INDXM
      IF(IFLAG(N).EQ.1)IMINUS=INDXB
      IDUM=IPTLST(N)-IMINUS
      PRINT  80,NAME,IDUM      ,LEN(N)
   80 FORMAT(15H0DUMP OF ARRAY A8,9H POINTER=I10,8H LENGTH=I10)
      GO TO ( 90,110,130),LUMP
   90 CONTINUE
      IF(MLT(N).EQ.4)GO TO 100
      CALL PRTI1(BLK(IPOINT),LEN(N))
      RETURN
  100 CONTINUE
      CALL PRTI2(BLK(IPOINT),LEN(N))
      RETURN
  110 CONTINUE
      IF(MLT(N).EQ.8)GO TO 120
      CALL PRTR1(BLK(IPOINT),LEN(N))
      RETURN
  120 CONTINUE
      CALL PRTR2(BLK(IPOINT),LEN(N))
      RETURN
  130 CONTINUE
      IF(MLT(N).EQ.8)GO TO 140
      CALL PRTA1(BLK(IPOINT),LEN(N))
      RETURN
  140 CONTINUE
      CALL PRTA2(BLK(IPOINT),LEN(N))
      RETURN
C
      ENTRY GETPNT(NAM1,IPT)
C
C     GET POINTER CORRESPONDING TO ARRAY NAME
C
      IPT=0
      CALL GETN(NAM1,N)
      IF(N.GT.0)GO TO 190
      RETURN
  190 CONTINUE
      IPT=IPTLST(N)
      IF(IPRINT.LT.2)RETURN
      IMINUS=INDXM
      IF(IFLAG(N).EQ.1)IMINUS=INDXB
      IDUM=IPT-IMINUS
      PRINT 200,NAME,IDUM,IFLAG(N)
  200 FORMAT(19H0POINTER FOR ARRAY A8,38H HAS BEEN EXTRACTED FROM POINTE
     1R TABLE/9H POINTER=I10,6H BULK=I10)
      RETURN
C
      ENTRY CLEAR(NAM1)
C
C     CLEARS ARRAY CORRESPONDING TO NAME
C
      CALL GETN(NAM1,N)
      IF(N.EQ.0)RETURN
      LENGTH=LENLST(N)
      IPT=IPTLST(N)
      DO 230 I=1,LENGTH
      BLK(IPT+I-1)=0
  230 CONTINUE
      IF(IPRINT.LT.2)RETURN
      PRINT 240,NAME
  240 FORMAT(7H0ARRAY A8,17H HAS BEEN CLEARED)
      RETURN
C
C     PRINT ERROR MESSAGES
C
```

```
 1000 CONTINUE
      NPTERR=NPTERR+1
      PRINT 1010,NAME,LENGTH
 1010 FORMAT(54HOSTORAGE EXCEEDED DURING REQUEST FOR STORAGE OF ARRAY
     1,/8H LENGTH=I10)
      RETURN
 1020 CONTINUE
      NPTERR=NPTERR+1
      PRINT 1030
 1030 FORMAT(48HOMAXIMUM NUMBER OF ENTRIES IN NAME TABLE EXCEEDED)
      RETURN
 1040 CONTINUE
      NPTERR=NPTERR+1
      PRINT 1050
 1050 FORMAT('0MAIN STORAGE CANNOT BE ALLOCATED')
      RETURN
 1060 CONTINUE
      PRINT 1070
      NPTERR=NPTERR+1
 1070 FORMAT('0BULK STORAGE CANNOT BE ALLOCATED')
      RETURN
      END
      SUBROUTINE GETN(NAM1,N)
C
      REAL*8 NAME,NAMLST
      REAL NAM1,NAM2
C
      DIMENSION NAM1(2),NAM2(2)
C
      COMMON/LOCATE/ LSTM,MAXM,LSTB,MAXB,INDXM,INDXB,IPRINT
      COMMON/TABLES/NAMLST(100),LENLST(100),IPTLST(100),
     1LEN(100),MLT(100),NNAMS
      COMMON/NAMBLK/NAME
C
      EQUIVALENCE (NAME,NAM2)
C
      N=0
      NAM2(1)=NAM1(1)
      NAM2(2)=NAM1(2)
      IF(NNAMS.EQ.0)GO TO 20
      DO 10 N=1,NNAMS
      IF(NAMLST(N).EQ.NAME)RETURN
   10 CONTINUE
   20 CONTINUE
      N=0
      IF(IPRINT.LT.2)RETURN
      PRINT   30,NAME
   30 FORMAT(7H0ARRAY A8,34H CANNOT BE FOUND IN THE NAME TABLE)
      RETURN
      END
      SUBROUTINE REDSET(BLK)
C
C     REDEFINES ARRAY NAME
C
      REAL*8 BLK
      REAL*8 NAME,NAMLST,BLANK,DUMNAM
      REAL NAM1,NAM2
C
      DIMENSION BLK(1)
      DIMENSION NAM1(2),NAM2(2),DUM(2)
C
C
      COMMON/LOCATE/ LSTM,MAXM,LSTB,MAXB,INDXM,INDXB,IPRINT
      COMMON/TABLES/NAMLST(100),LENLST(100),IPTLST(100),
     1LEN(100),MLT(100),NNAMS
      COMMON/BFLAGS/ IFLAG(100)
      COMMON/NAMBLK/NAME
      COMMON/PTERR/NPTERR
C
C
      EQUIVALENCE (NAME,NAM2),(DUMMY,DUM(1))
C
      DATA BLANK/'        '/,DUMNAM/'DUMNAM'/
C
      RETURN
C
      ENTRY REDEF(NAM1,LENDUM,MULT)
C
      CALL GETN(NAM1,N)
      IF(N.GT.0)GO TO 30
C
C     NAME NOT IN TABLE SO PUT IT THERE
C
```

```
      CALL PUTPNT(NAME,LENDUM,MULT)
      RETURN
   30 CONTINUE
      LENNEW=((LENDUM-1)*MULT+8)/8
      LENOLD=LENLST(N)
      IF(LENNEW.EQ.LENOLD)RETURN
      IF(LENNEW.GT.LENOLD)GO TO 60
C
C     NEW LENGTH LESS THAN OLD LENGTH
C
      IF(NNAMS+1.LE.100)GO TO 50
      CALL PURGE(DUMMY)
      IF(NNAMS+1.LE.100)GO TO 50
      CALL PURGEB(DUMMY)
      IF(NNAMS+1.LE.100)GO TO 50
      NPTERR=NPTERR+1
      PRINT 40
   40 FORMAT(49HOMAXIMUM NUMBER OF ENTRIES IN NAME TABLE EXCEEDED)
      RETURN
   50 CONTINUE
      NNAMS=NNAMS+1
      L1=NNAMS-N-1
      DO 55 L=1,L1
      K=NNAMS-L
      NAMLST(K+1)=NAMLST(K)
      IPTLST(K+1)=IPTLST(K)
      LENLST(K+1)=LENLST(K)
      LEN   (K+1)=LEN   (K)
      MLT   (K+1)=MLT   (K)
      IFLAG(K+1)=IFLAG(K)
   55 CONTINUE
      LENLST(N)=LENNEW
      LEN(N)=LENDUM
      MLT(N)=MULT
      NAMLST(N+1   )=BLANK
      IPTLST(N+1   )=IPTLST(N)+LENNEW
      LENLST(N+1   )=LENOLD-LENNEW
      IF(IPRINT.LT.2)RETURN
      IMINUS=INDXM
      IF(IFLAG(N).EQ.1)IMINUS=INDXB
      IDUM=IPTLST(N)-IMINUS
      PRINT 56,NAME,IDUM          ,LENLST(N    ),LEN(N    ),MLT(N    ),
     1IFLAG(N    )
   56 FORMAT(17HOENTRY FOR ARRAY AR,35H HAS BEEN MADE IN THE POINTER TAB
     1LE/9H POINTER=I10,9H ORL LEN=I5,10H ORIG LEN=I5,6H MULT=I5 ,
     26H BULK=I5)
      RETURN
   60 CONTINUE
C
C     NEW LENGTH GREATER THAN OLD LENGTH
C
      NAMLST(N)=DUMNAM
      CALL PUTPNT(NAME,LENDUM,MULT)
      IF(NAMLST(N).EQ.DUMNAM)GO TO 90
      CALL GETN(DUMNAM,N)
      IF(N.GT.0)GO TO 90
      NPTERR=NPTERR+1
      RETURN
   90 CONTINUE
      NAMLST(N)=BLANK
      DO 100 L=1,LENOLD
      LSUB1=IPTLST(N)+L-1
      LSUB2=IPTLST(NNAMS)+L-1
      BLK(LSUB2)=BLK(LSUB1)
  100 CONTINUE
      RETURN
      END
      SUBROUTINE PRGSET(BLK)
C
C     SIFTS STORAGE AND ELIMINATES ARRAYS IN THE PURGE TABLE
C
      REAL*8 BLK
      REAL*8       NAMLST,BLANK
C
      DIMENSION BLK(1)
C
      COMMON/LOCATE/ LSTM,MAXM,LSTB,MAXB,INDXM,INDXB,IPRINT
      COMMON/TABLES/NAMLST(100),LENLST(100),IPTLST(100),
     1LEN(100),MLT(100),NNAMS
      COMMON/BFLAGS/ IFLAG(100)
C
      DATA BLANK/'        '/
C
```

```
      IMAIN=LSTM
      IBULK=LSTB
      RETURN
C
      ENTRY PURGE(LSTLOC)
C
      ISTART=IMAIN
      KEY=-1
      MAXBLK=MAXM
      IMINUS=INDXM
      IF(NNAMS.GT.0)GO TO 5
      LSTLOC=LSTM
      RETURN
C
      ENTRY PURGEB(LSTLOC)
C
      ISTART=IBULK
      KEY=0
      MAXBLK=MAXB
      IMINUS=INDXB
      IF(NNAMS.GT.0)GO TO 5
      LSTLOC=LSTB
      RETURN
C
    5 CONTINUE
      IMOV=ISTART
      INAM=0
      DO 40 N=1,NNAMS
      IF(KEY+ IFLAG(N).NE.0)GO TO 10
      INAM=INAM+1
      NAMLST(INAM)=NAMLST(N)
      IPTLST(INAM)=IPTLST(N)
      LENLST(INAM)=LENLST(N)
      LEN(INAM)=LEN(N)
      MLT(INAM)=MLT(N)
      IFLAG(INAM)=IFLAG(N)
      GO TO 40
   10 CONTINUE
      IF(NAMLST(N).EQ.BLANK     )GO TO 40
C     NAME NOT BLANK          - SO SAVE IT
      IMAX=LENLST(N)
      IF(IMOV.EQ.IPTLST(N))GO TO 30
      DO 20 I=1,IMAX
      ISUB1=IMOV+I-1
      ISUB2=IPTLST(N)+I-1
      BLK(ISUB1)=BLK(ISUB2)
   20 CONTINUE
   30 CONTINUE
      INAM=INAM+1
      NAMLST(INAM)=NAMLST(N)
      IPTLST(INAM)=IMOV
      LENLST(INAM)=IMAX
      LEN(INAM)=LEN(N)
      MLT(INAM)=MLT(N)
      IFLAG(INAM)=IFLAG(N)
      IMOV=IMOV+IMAX
   40 CONTINUE
      NNAMS=INAM
      IF(KEY.EQ.0)GO TO 41
      LSTM=IMOV
      GO TO 42
   41 CONTINUE
      LSTB=IMOV
   42 CONTINUE
      LSTBLK=IMOV
      LSTLOC=LSTBLK-IMINUS
      DO 45 I=LSTBLK,MAXBLK
      BLK(I)=0.
   45 CONTINUE
      IF ((NNAMS+1).GT.100)  GO TO 49
      IF(IPRINT.LT.2)RETURN
   49 CONTINUE
      PRINT 50
   50 FORMAT(  '1STORAGE HAS BEEN SIFTED')
C
      ENTRY STATUS
C
      IF ((NNAMS+1).GT.100)  GO TO 51
      IF(IPRINT.LT.2)RETURN
   51 CONTINUE
      PRINT 55
   55 FORMAT('CCURRENT STATUS OF CONTAINER ARRAY IS AS FOLLOWS'/,
     11H07X2HND6X4HNAME3X7HPOINTER3X7HDBL
```

```
     26X4HBULK)
      DO 70 N=1,NNAMS
      IMINUS=INDXM
      IF(IFLAG(N).EQ.1)IMINUS=INDXB
      IDUM=IPTLST(N)-IMINUS
      PRINT 60,N,NAMLST(N),IDUM      ,LENLST(N),LEN(N),MLT(N),IFLAG(N)
   60 FORMAT(1I10,2X8,8I10)
   70 CONTINUE
      RETURN
      END
      FUNCTION IGET(NAM1)
C
C     GET POINTER CORRESPONDING TO ARRAY NAME
C
      REAL*8              NAME,NAMLST,BLANK
      REAL NAM1,NAM2
C
      DIMENSION NAM1(2),NAM2(2)
C
      COMMON/LOCATE/ LSTM,MAXM,LSTB,MAXB,INDXM,INDXB,IPRINT
      COMMON/TABLES/NAMLST(100),LENLST(100),IPTLST(100),
     1LEN(100),MLT(100),NNAMS
      COMMON/BFLAGS/ IFLAG(100)
      COMMON/NAMBLK/NAME
      COMMON/PTERR/NPTERR
C
      EQUIVALENCE (NAME,NAM2)
C
      IPT=0
      CALL GETN(NAM1,N)
      IF(N.GT.0)GO TO 10
      RETURN
   10 CONTINUE
      IGET=IPTLST(N)
      IF(IPRINT.LT.2)RETURN
      IMINUS=INDXM
      IF(IFLAG(N).EQ.1)IMINUS=INDXB
      IDUM=IPTLST(N)-IMINUS
      PRINT 60 ,NAME,IDUM,IFLAG(N)
   60 FORMAT(19H0POINTER FOR ARRAY A8,38H HAS BEEN EXTRACTED FROM POINTE
     1R TABLE/9H POINTER=I10,6H BULK=I10)
      RETURN
      END
      SUBROUTINE PRTI1 (J,LENGTH)
C
C     PRINTS NON STANDARD INTEGER ARRAY
      INTEGER*2 J(1)
C
      PRINT 10,(J(L),L=1,LENGTH)
   10 FORMAT(24I5)
      RETURN
      END
      SUBROUTINE PRTI2 (J,LENGTH)
C
C     PRINTS STANDARD INTEGER ARRAY
C
      DIMENSION J(1)
C
      PRINT 10,(J(L),L=1,LENGTH)
   10 FORMAT(12I10)
      RETURN
      END
      SUBROUTINE PRTR1 (A,LENGTH)
C
C     PRINTS STANDARD REAL ARRAY
C
      DIMENSION A(1)
C
      PRINT 10,(A(L),L=1,LENGTH)
   10 FORMAT(8E15.7)
      RETURN
      END
      SUBROUTINE PRTR2 (A,LENGTH)
C
C     PRINTS NON STANDARD REAL ARRAY
C
      REAL*8 A(1)
C
      PRINT 10,(A(L),L=1,LENGTH)
   10 FORMAT(8E15.7)
      RETURN
      END
      SUBROUTINE PRTA1 (A,LENGTH)
```

```
C
C     PRINTS STANDARD HOLLERITH ARRAY
C
      DIMENSION A(1)
C
      PRINT 10,(A(L),L=1,LENGTH)
   10 FORMAT(1H 25A4)
      RETURN
      END
      SUBROUTINE PRTA2 (A,LENGTH)
C
C     PRINTS NON STANDARD HOLLERITH ARRAY
C
      REAL*8 A(1)
C
      PRINT 10,(A(L),L=1,LENGTH)
   10 FORMAT(1H 12A8)
      RETURN
      END
      FUNCTION IPTERR(DUMMY)
C
C     RETURNS NUMBER OF ERRORS THAT HAVE OCCURED IN POINTER ROUTINES
C
      COMMON/PTERR/NPTERR
C
      IPTERR=NPTERR
      NPTERR=0
      RETURN
      END
      FUNCTION ILAST(DUMMY)
C
C     RETURNS FIRST AVAILABLE LOCATION IN CONTAINER ARRAY
C
      COMMON/LOCATE/ LSTM,MAXM,LSTB,MAXB,INDXM,INDXB,IPRINT
C
      ILAST=LSTM-INDXM
      RETURN
      END
      FUNCTION ILASTB(DUMMY)
C
C     RETURNS FIRST AVAILABLE LOCATION IN CONTAINER ARRAY
C     OF BULK CORE
C
      COMMON/LOCATE/ LSTM,MAXM,LSTB,MAXB,INDXM,INDXB,IPRINT
C
      ILASTB=LSTB-INDXB
      RETURN
      END
```

```
FORTRAN  START
*
* CALLING SEQUENCE  CALL FORTRAN(FTNRTN,FLAG,A,B,C,...);
*                        FLAG < 0  IF NO PARAMETERS
*                        FLAG = 0  IF NO STRING DATA IN PARAMETER LIST
*                        FLAG = N  IF FIRST N PARAMETERS ARE STRINGS
*
         EXTRN IHEERRA
         ENTRY FIOCLSE
         BC    15,12(15)
         DC    X'07'
         DC    CL7'FORTRAN'
         STM   14,12,12(13)              SAVE REGISTERS
         BALR  12,0                      ESTABLISH BASE REGISTER
         USING *,12
BASE     LR    8,13                      LOAD PL/I SAVE AREA POINTER
         LA    13,SAVE                   FORTRAN'S SAVE AREA POINTER
         ST    13,8(8)                   STORE POINTER IN PL/I SAVE AREA
         ST    8,4(13)                   STORE PL/I SAVE AREA POINTER
*                                          IN FORTRAN'S SAVE AREA
         ST    15,SAVE+16                STORE ADDRESS OF FORTRAN IN
*                                          FIFTH WORD OF ITS SAVE AREA
*                                          FOR TRACEBACK
*
* CREATE FORTRAN ENVIRONMENT THROUGH CALL TO IBFINT
* IBFINT SAVES REGISTERS IN ITS OWN SAVE AREA BUT EXPECTS REGISTER 13
* TO POINT TO SAVE AREA OF SUPERVISOR -- FORTRAN.
*
         L     15,VIBCOM                 LOAD ADDRESS OF IBCOM
         BAL   14,64(15)                 CALL IBFINT TO INITIALIZE FORT.
         L     13,SAVE+4                 LOAD PL/I SAVE AREA POINTER
         L     1,24(13)                  LOAD PARAMETER LIST ADDRESS
         L     2,0(1)
         L     15,0(2)                   LOAD ADDRESS OF FORTRAN SUBRTN.
         LA    13,SAVE                   LOAD SAVE AREA POINTER
* TEST FLAG FOR ANALYSIS OF PARAMETER LIST
         L     3,4(1)
         L     3,0(3)                    LOAD FLAG
         LTR   3,3                       TEST FLAG FOR PARAMETERS
         BC    NOPARM,CALL               BRANCH IF FLAG<0  NO PARAMETERS
         LA    1,8(1)                    SET PARAMETER LIST POINTER
         BC    NOSTRG,CALL               BRANCH IF FLAG=0  NO STRINGS
* OBTAIN STRING ADDRESS FROM 1ST WORD OF STRING DOPE VECTOR -- SDV
         SR    6,6                       ZERO REGISTER
LOOP     L     4,0(6,1)                  LOAD ADDRESS OF SDV
         L     2,0(4)                    LOAD 1ST WORD OF SDV
         ST    2,0(6,1)                  STORE ADDRESS IN PARAMETER LIST
         AH    6,FOUR                    INCREASE REGISTER FOR NEXT PARM
         BCT   3,LOOP                    BRANCH IF N NOT ZERO
*
CALL     BALR  14,15                     CALL FORTRAN SUBROUTINE
*
         L     5,RCADDR                  LOAD ADDRESS OF RETURN CODE
         MVC   0(4,5),RCNORM             PUT ZERO IN RETURN CODE
* TEST RETURN CODE IN REGISTER 15 FOR TERMINAL ERROR, NORMAL RETURN,
* OR STOP.  ALL OTHERS ASSUMED NORMAL.
         LTR   15,15                     TEST RETURN CODE
         BC    ZERO,RETCODEO             BRANCH IF CODE IS 0
         C     15,TERMERR                TEST FOR TERMINAL ERROR
         BC    NOERR,CONTINUE            BRANCH IF NOT A TERMINAL ERROR
         MVC   0(4,5),RCERR              PUT 1 IN RETURN CODE : ERROR
         BC    ALWAYS,CONTINUE
RETCODEO CLI   12(13),X'FF'              TEST FOR NORMAL RETURN
         BC    ZERO,CONTINUE             BRANCH IF NORMAL RETURN
         MVC   0(4,5),RCSTOP             PUT -1 IN RETURN CODE : STOP
* RESTORE PL/I INTERRUPT SYSTEM
CONTINUE SPIE  IHEERRA,((1,13),15)
*
         L     13,SAVE+4                 LOAD PL/I SAVE AREA POINTER
         LM    14,12,12(13)              RESTORE REGISTERS
         BR    14                        RETURN TO PL/I
*
* CLOSE FORTRAN I/O DATA SETS THROUGH CALL TO IBEXIT
* IBEXIT SAVES REGISTERS IN ITS OWN SAVE AREA BUT RETURNS VIA FOURTH
* WORD IN SAVE AREA OF SUPERVISOR -- FORTRAN.  REGISTER 14 IN CALL IS
* USED FOR A RETURN CODE.
*
         CNOP  0,4
FIOCLSE  BC    15,12(15)
         DC    X'07'
         DC    CL7'FIOCLSE'
         STM   14,12,12(13)              SAVE PL/I REGISTERS
         L     12,POINTER-FIOCLSE(15)    ESTABLISH BASE REGISTER
```

```
         ST     13,REG13                    STORE PL/I SAVE AREA POINTER
         MVC    REG14(4),SAVE+12            REPLACE FOURTH WORD OF SAVE
         MVC    SAVE+12(4),ARETURN            AREA BY THE RETURN ADDRESS
         L      15,VIBCOM                   LOAD ADDRESS OF IBCOM
         BAL    14,68(15)                   CALL IBEXIT TO CLOSE I/O D.S.
         DC     AL2(0)                      AL2(RETURN CODE)
         CNOP   0,4
* RESTORE PL/I INTERRUPT SYSTEM
RETURN   SPIE   IHEERRA,((1,13),15)
*
         MVC    SAVE+12(4),REG14            RESTORE 4TH WORD OF SAVE AREA
         L      13,REG13                    LOAD PI/I SAVE AREA POINTER
         LM     14,12,12(13)                RESTORE PL/I REGISTERS
         BR     14                          RETURN TO PL/I
*
* STORAGE SECTION FOR FORTRAN
SAVE     DC     18F'0'
VIBCOM   DC     V(IBCOM#)
POINTER  DC     A(BASE)
ARETURN  DC     A(RETURN)
RCADDR   DC     A(RETCODE)
TERMERR  DC     XL4'C0000C10'
REG13    DC     F'0'
REG14    DC     F'0'
RCERR    DC     F'1'
RCNORM   DC     F'0'
RCSTOP   DC     F'-1'
FOUR     DC     H'4'
NOPARM   EQU    4
NOERR    EQU    7
ZERO     EQU    9
NOSTRC   EQU    8
ALWAYS   EQU    15
*
* STORAGE AREA FOR RETURN CODE TO PL/I
* DECLARE RETCODE FIXED BINARY (31,0) EXTERNAL;
RETCODE  CSECT
CODE     DC     F'0'                        INITIALIZE CODE TO ZERO
         END
1
```

3 4444 00031550 7