SPECIFICATIONS AND DESIGN OF A FLEXIBLE INFORMATION MANAGEMENT SYSTEM FOR LARGE DATA BASES

by

NICOLA P. SZASZ

S.B., UNIVERSITY OF SAO PAULO
1972

Submitted in partial fulfillment of the requirement for the Degree of Master of Science

at the

Massachusetts Institute of Technology

September, 1974

Signature of Author_____
Department of Ocean Engineering, August 12, 1974

Certified by_____
Thesis Supervisor

Reader_____
Department of Ocean Engineering

Accepted by_____
Chairman, Departmental Committee on Graduate Students

SPECIFICATIONS AND DESIGN OF A FLEXIBLE INFORMATION MANAGEMENT SYSTEM FOR LARGE DATA BASES

NICOLA P. SZASZ

Submitted to the Department of Ocean Engineering on August 12, 1974 in partial fulfillment of the requirements for the degree of Master of Science in Shipping and Shipbuilding Management.

Present trends indicate that large data bases are the viable and efficient solution for the mass storage of large amounts of scientific data collected during physical experiments.

Scientists of Coastal Oceanography are presently engaged in the implementation of an interactive sea scanning system using real time acquisition and display of oceanographic data.

This report presents the concepts involved in the design of an information management system for a large oceanographic data base. Also, results obtained from a preliminary implementation in the M.I.T. Multics system are presented.

Thesis Supervisor: Stuart E. Madnick

Title: Assistant Professor of Management Science

ACKNOWLEDGMENTS

The author expresses thanks to Professor Stuart E. Madnick for his guidance and encouragement as supervisor in this thesis study. He is also grateful to students in the Meteorology Department and staff members of the Cambridge Project, for their valuable discussion on the thesis topic.

I would like to express my thanks to my advisors, at the University of Sao Paulo, for their encouragement in my attending M.I.T. Special thanks go to my course advisor, Professor John W. Devanney III for his tolerance and encouragement during my stay at M.I.T.

Sincere thanks also to my parents, Attila and Ilona Szasz for providing financial and moral support in earlier education.

The implementation of the system described in this report was possible, thanks to the funds provided by the Cambridge Project.

TABLE OF CONTENTS

Table of Contents (continued) Page

List of Figures

Chapter 1

INTRODUCTION

One of the areas in Oceanography that has attracted the attention of many researchers and scientists in the recent past has been the Coastal Oceanography problem area. One of the problems that this area has faced is to obtain better assessments of coastal pollution and offshore activities in order to generate a sufficient understanding of the processes involved in dispersion and transport of pollutants. Once this has been accomplished, it will become easier to predict the consequences of future action, both locally and extensively. Actually, in this problem area there are several complicated features that must be taken into account, as to increase the model predictiveness. The coastal region of the ocean is mostly shallow and the response time to atmospheric input is relatively short. The tendency of pollutants to float at the surface is due to the fact that they are emitted in regions of water with lower density than that of ambient seawater. Wind strongly affects the near surface circulation. The dynamics of the processes are three dimensional and time dependent. There are different scale processes and the zones of activity of all scales are not stationary. Transcient phenomena such as storm passage may

significantly affect these scales and processes. Wind induced currents, transcient upswellings, and storm and run-off induced mixing, which are the processes that determine dispersion of pollutants, all contain unhomogenties of scales from meters to tens of km, lasting from hours to weeks.

Oceanographic measurements have been evolving, from station taking and water samples collection, in the last years, to the use of fixed buoys for longer term observation of physical variables. The use of such information acquisition tools has revealed the existence of fluctuations in water motion, containing energies comparable to the kinetic energy of the mean current systems. The scales and intensities of time dependent ocean dynamics indicate the presence of phenomena of horizontal scales of a few depths. Therefore, the scales of many phenomena in shallow wastal regions are expected to be small.

The tasks of monitoring the state of the ocean and the development and evaluation of predictive models in the coastal and shelf region, generally need systems and techniques that are not available in the present moment. The research on these smaller scale phenomena has been handled by conventional oceanographic and data handling techniques, which have led to several problems. The number of buoys and stations required to determine the dynamics of a local dispersion process is very large and uneconomical. Even if such large efforts are

undertaken, the work is still restricted to a few local areas and the results difficult to interpret since the data would be spatially discontinuous. On the other hand, a big problem is to integrate the information acquired from a number of various sensors on different platforms to arrive at an assessment of the state and the processes controlling pollutant dispersion.

Given that all of the information that is gathered, by oceanographic techniques, is later processed to help in the design of predictive models, careful attention must be given to how the data is handled and processed. Since most of the large amount of data acquired is irrelevant, conventional methods of collecting, sorting, editing and processing raw data are not practical. Existing facilities and data banks are not equipped to handle with the large amounts of data that will be generated in studying areas such as coastal oceanography. Therefore, the data collection process must be continuously assessed in real time to assure that only relevant data is sought and stored. Furthermore, the data should be prepared for storage in a form that is appropriate to shore-based analysis and modeling.

As an attempt to overcome all these mentioned difficulties in the study of problems related to coastal oceanography and to permit further research and development within this area, an interactive data scanning system has been proposed.

The full system would consist of a vessel towing appropriate sensor arrays and maneuverable sensor platforms, with computerized automatic guidance and navigation responsive to real time data assessment, computerized data acquisition and storage, with real time display of processed information for assessment and interpretation, and an off-line research facility for analysis, modeling and data management. The off-line Research Laboratory would consist of graphics terminals, library files, and multiprocessors, coupled to large time-sharing computer facilities for data management, simulation and modeling. The group of scientists, engineers, information theorists and programmers, would then affect the analysis, modeling and simulation, using a data base management system.

In order for such a system (Interactive Data Scanning System) to work properly and make meaningful scientific contribution, it is essential that the on-line real time element of this system be complemented by the shore based research facility.

In the past, the traditional approach to such a Research Laboratory has been of having none. Data used to be collected and stored in a unhomogeneous form and the researchers would utilize means and facilities that were individually available to them. Evidently, there are several drawbacks for this option. Available computation facilities usually

consist of some large computing center which is not oriented towards using large data bases or supplying effective input-output for research which involves large amounts of real data. On the other hand, due to the software barrier, researchers limit very much the data utilization needed to fulfill their objectives.

Given that this approach is highly inefficient and undesirable, the alternative option is to make use of an available major computing center, but to add input-output hardware and problem-oriented software to properly interface the computer with the research, data analysis and data management tasks of IDSS. In this way the Research Laboratory would use the techniques inherent to data base management in order to provide a well-defined and flexible structure that would permit a unique and efficient way of storing and retrieving data.

The above mentioned Research Laboratory is to fulfill the following main functions:

a) Filing and storage of raw and reduced data in such a manner as it is readily accessable and useful to the users.

b) Maintaining a common library of programs so that programs written by one observer or researcher are accessable, documented and readily understandable by other users.

c) Provide hardware and software support for numerical modeling.

The first two items described above reflect very closely what is called today a data base management system.

Once such a system is designed and implemented, the scientist of Coastal Oceanography is provided with powerful tools to analyze his data. By means of directories containing general information on the data stored in the data base, he is able to locate and copy particular sections of data into working temporary files. Once he did this, he may proceed and run analysis on his data using models and/or techniques that are stored in the data base as a common library of programs.

After deciding to interrupt the analysis, the user may, if he wishes, save results in the data base as well as status and general information concerning the results and/or proceedings of his analysis.

It is our belief, that by means of utilizing a data base management system, the research laboratory would provide an efficient tool for scientists to get better acquainted with Coastal Oceanography problems. Such a tool would be used both in the raw data acquisition area as well as the numerical modelling and prediction area.

Chapter 2

RELATED RESEARCH AND LITERATURE

2.1 The data base concept

It has been a general trend in the past, to build special formatted files which could be used by immediately needed programs. Thus, in most information processing centers, when someone asked for a new application using the computer, great thought was given to the preparation and formatting of data into files that would be used by the future programs. The result, unfortunately, has been always the same: after a considerable number of different applications have been implemented, the center found itself with several copies of the same data in different formats.

The natural drawbacks resulting from this procedure are obvious: computer storage waste and inefficient programming.

One might agree that a small number of copies on the same data is a good way to handle integrity. While it is certainly true that all data must have a back-up copy, the problem is that future applications will need new copies of the data, since the data will not be in readily suitable form for the new applications. Presently, with the rapid

advance and development of hardware/software, new applications are very likely to appear and be developed in computer-based systems.

Inefficient programming is a natural result of the several copies of the same data in different formats. Since different formats have different file schemes, the input/output routines, as well as the file manipulating procedures will all be different. Evidently, this is highly undesirable given that a level of standardization is never achieved.

One of the proposed ways of getting around this problem, is to use the data base concept. A data base is just that "a consistent and general conglomerate of data, upon which are built the file interfaces so that different applications programs can use the data which is stored under a unique format.

In such a way, a high level of standardization is achieved, given that all the data is stored in the data base. The files interfaces are considered a different system and are also standardized.

Besides presenting the natural advantage of efficient computer storage usage, the data base concept enables new applications to be developed independently, therefore more rapidly, from the data format.

2.2 On-line conversational interaction

After the data base concept emerged, the natural trend was to use it with the existing background jobs consisting of non-interactive programs.

However, since both the data base and on-line environment concepts have in the last years advanced drastically, providing field for newer applications, the idea of using both together was generated.

While data bases have provided means for efficient storage of data, and therefore fast information retrieval; on-line environments, providing man-computer conversational interaction, have presented a new "doorway" for certain applications.

The whole idea with on-line conversational interaction is to enable the man to direct the computer with regard to which actions the machine should take. This is usually done by using modules, that the computer executes after receiving appropriate instructions and once the machine performed its task, it will display information on the results obtained. After the man has analyzed this displayed information, he is ready to issue a new instruction ordering the machine to perform a new module.

Many programs that once used to run in background mode, are now running under on-line environments more efficiently in terms of performance and cost. The reason for this is as

follows: Often programs have to make decisions during run time as which action to take. While in a background mode these decisions are made by the program itself, based on mostly irregular rules; an interactive and conversational program enables feedback from a human regarding key and difficult to make "beforehand" decisions.

2.3 Related work in Oceanography

The data base concept as described in Section 2.1 has never been attempted before in Coastal Oceanography. The first time the idea was considered was exactly during the preliminary development of the IDSS. As was seen in Chapter 1, the research laboratory is to fulfill several different functions, one of them being the development of a general purpose data base.

In the past, most of the work using computer facilities was done as described in Section 2.1, i.e., different applications and programs had different versions of the same data.

Usually, whenever a problem area is to be covered in Oceanography, the development procedures is as follows: First the physical experiment is established and the variables to be measured are defined. Next the data is acquired in whatever form seems more convenient from the instrumentation point of view. After being gathered, this data is transformed into a compatible computer form, and then the scientist

will usually write a high level language program to run a modeling analysis in his data. Obviously, this program is highly dependent on the data format that was used to store the data gathered during the experiment.

That being the case, whenever a new set of data under a different format is to be used with the same modelling analysis, there are two choices: either reformat the data or create a new version of the program.

On the other hand, sometimes the data acquired for one experiment might be used to run a second and different analysis model. However, given that this program was developed with another data format in mind, once again there is a choice of either reformatting the data or changing the program.

Since a common library of programs is not established and almost no documentation is available,sometimes a user develops programs or routines that have already been developed by another user.

In the data acquisition area using data processing, the Woods Hole Oceanographic Institution provided the development and implementation of a digital recording system as an alternative to the cumbersome and expensive strip chart recording and magnetic tape techniques presently used to collect data from "in-situ" -- marine experiments. The same effort has been developed for Marine Seismic Data

Processing.

In the analysis and modeling area, once more the Woods Hole Oceanographic Institution provided the development of computer solutions for predicting the equilibrium configuration of single point moored surface and subsurface buoy systems set in planar flow.

The ACODAC system, also developed at Woods Hole Oceanographic Institution, has computer programs and techniques to reduce the raw ACODAC ambient data to meaningful graphic plots and statistical information which are representative of the ambient noise data resulting from the deployment of acoustic data capsules during the period of 1971 to 1973. This system was, therefore, an integration between hardware and software, to convert raw ambient noise data into formats that can be used with the appropriate statistical subroutines to obtain the desired acoustic analysis.

The U.S. Navy Electronics Laboratory has conducted experiments in order to study vertical and horizontal thermal structures in the sea and measure factors affecting underwater sound transmission. A detailed temperature structure data in the upper 800 feet of the sea south of Baja, California, was acquired by the U.S. Navy Electrcnics Laboratory using a towed thermistor chain. Data was therefore gathered and later processed by existing software to analyze underwater sound transmission.

In order to start some standardization and begin to establish a data base concept, the people working with Oceanography, have designed and partially implemented the Interactive Data Scanning System tape file system, which will be described in Chapter 3, as well as an interface module, responsible to transfer data from the IDSS tape files to an early version of an oceanographic data base.

The IDSS tape file system represents the first step in the direction of a data base, since it attempts to standardize the format under which data is to be acquired and stored during physical experiments.

2.4 A case study: The Lincoln Lab and the Seismic Project

A good example of an information management system for a large scientific data base is found in the Seismic Project at the Lincoln Laboratory.

Seismic data comes into the Lincoln Lab by means of tape files containing data, that was gathered by different seismic stations located throughout the world.

Whenever a new tape comes in, the first step is to normalize these tape files so that they become consistent with the sesmic data base. The databank consists of a tape library where each tape has an identification number. Next a background job is run in order to append general information, concerning these tape files, to the databank directory.

Each time a scientist wants to run an analysis, he has to find out where the piece of data is that he is interested in. This is accomplished by a conversational on-line program, that asks questions to the user, who is sitting at a console, and expects answers as to which actions it should take. Typically, in this mode the user poses several different queries to the databank directory, until he finally knows the identification number of the tape on which the particular file of data resides. Next the computer operator sets up the tape on an available tape drive and an existing program pulls the data from the tape to a direct-acess device.

Once the data is in a drum/disk, the scientist can run his analysis using programs that were written for seismic directed analysis.

The analysis as implemented in the Lincoln Lab, uses a typewriter console for interactive conversation and a CRT device for graphical displays.

Once the analysis is over, the scientist may, if he wishes, save results on a tape. The system with the user's help will add information into the databank directory concerning saved files.

The data base management system as implemented in the Seismic Project has evidently some limitations. In order to find the piece of data he is interested in, the user has to

ask questions to the databank directory. Given that only the databank directory resides on direct access device, and that this directory contains only general information, it may happen that the user has to set up more than one tape until he finally finds the appropriate section of data to analyze. On the other hand, the analysis is implemented by means of commands that call FORTRAN programs, that are not always as flexible as one might expect. This happens since a software base for data management was not used, and because this project has developed its own graphics software.

Finally, in the performance area, one might mention that the system is using mini computer equipment, thus generating some time and size restrictions.

Chapter 3

USER'S REQUIREMENTS

3.1 General Outline

One of the objectives of the Interactive Data Scanning System is to provide oceanographic researchers with sufficiently powerful tools so that they can analyze the data that was acquired by the off-shore dynamic scanning system. Such an objective, would best be accomplished by a shore based Research Laboratory using a data base management system.

In order to provide conversational interaction with the whole system, so that the scientist can actually interact with the machine, controlling the steps and results of an analysis, such a system should be designed assuming an on-line environment.

In this section, we shall take a general view of what an analysis may consist of, and then we shall describe the general organization of the data base itself. Finally a detailed but somewhat "abstract" description of a possible analysis is given.

In a general form, each time a scientist wants to analyze oceanographic data, he has to go through three distinct procedures:

1 - Considering that all his data is in an on-line environment, the user wants initially to locate and define the logical section of data, he is interested in. Once this has been accomplished, he will copy it into a work file, so that the data base contents remain unaffected.

2 - After having all the data copied into a work file, the user is ready to run the analysis. Basically, the scientist is interested in three blocks of operation: data management (copy, edit, merge, sort), graphical displaying and time series processing.

3 - After the scientist having analyzed his data and obtained the results, he may want to store them for later use. Therefore, the user saves the results of his work in the data base, as well as the status and information on this analysis, so that work can be resumed in the future.

The whole data base management system, from the user's point of view, may be visualized as three distinct blocks:

1-The databank

2-The databank directory

3-The database language and procedures.

The necessity of a global integration between the off-shore real time acquisition system and the shore based Research Laboratory, is stressed in the design of the databank and the databank directory. The raw data, gathered by the on-

line system, is transferred to the database system, by means of tape files consisting of ASCII character records. A typical tape file is divided into master records and data records. The master records contain relevant information on the how, when, why, what and where of the data acquisition. The data records are the ones containing the bulk of the raw-data. The important point is to notice that whenever the how, when, why, what or where of the data drastically change, we need a new set of master records. A combination of master records and data records, giving a tape file, from now on called as a cruise raw file, will be next described.

Master records are always located at the beginning of the file, in a predetermined order, and may not appear anywhere else in the data stream. More than one of any given type may occur and they are in order of appearance in file:

M1) General Information

M2) Attribute table

M3) Synchronous instrumentation geometry

M4) Synchronous instrumentation calibration

M5) Asynchronous instrumentation geometry

M6) Asynchronous instrumentation calibration

M7) System fixed discriptor information

M8) Marker definition

The appended tables (III.1 through III.10) illustrate the typical contents of the master records for a sample cruise. Later, tables III.11 through III.13 illustrate a possible format for the raw data contained in records in the tape file.

It is important to design the databank and databank directory in such a way as to permit an efficient and simple reordering of the cruise raw files, for the appropriate on-line utilization during analysis and modeling sessions.

On the other hand, since the users will most of the time want to save results in order to resume work in the future, a major issue in the design is to enable the scientist to retrieve his results in a simple and efficient way. An interactive mode should be availabe to allow the user an easy and relatively fast way of finding his results.

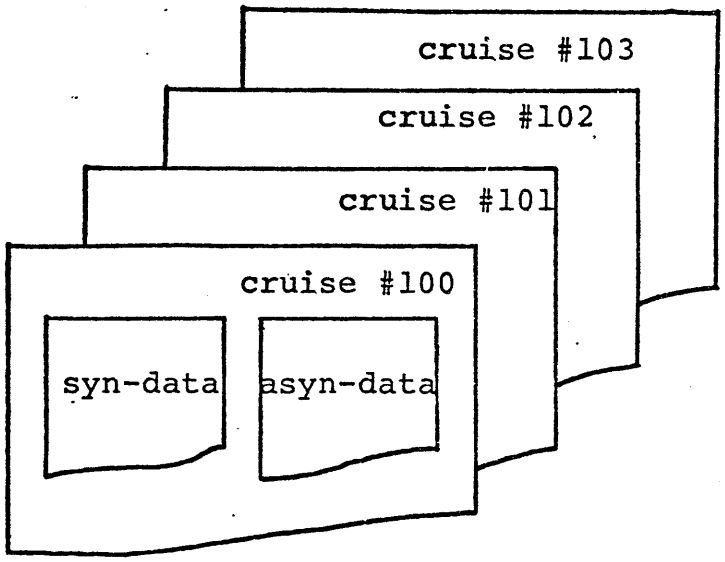
3.2 The Databank

The databank is divided into two logical parts, each part containing a set of files. The first part is the group of files where the acquired raw data is stored. Each different cruise when integrated into the data base generates two files, one containing the synchronous data and the other containing the asynchronous data. The second part of the databank contains the results of a series of well-defined analysis. Each time the scientist finishes an on-line conversational analysis on his data, he saves the results of his

work creating new files in the results databank. Each file, containing either cruise raw data or results data from an analysis, is organized logically by means of entities (observations) and attributes (properties). A file might be visualized as being an m x n matrix where the lines stand for entities (different observations) and the columns for attributes (properties related to the observations).

Figure III.1 depicts the databank format.

At this point, a fundamental difference should be pointed concerning raw files as opposed to results files. The first type has a well defined format and number: two for each cruise; whereas the second needs a wide range of possibilities within the same format. The main reason for this need is that different scientists or even the same scientist will conduct different analysis and might be willing to save the results at different steps involving different values or different attributes. As an example, one might mention the results that are obtained from calculating depth differences for a certain isotherm as opposed to frequency and cumulative distributions of these differences for a certain section of data. In the first case the depth differences are related to time intervals, concerning individual entities of the file, whereas in the second case the attributes are typically related to a group of entities.



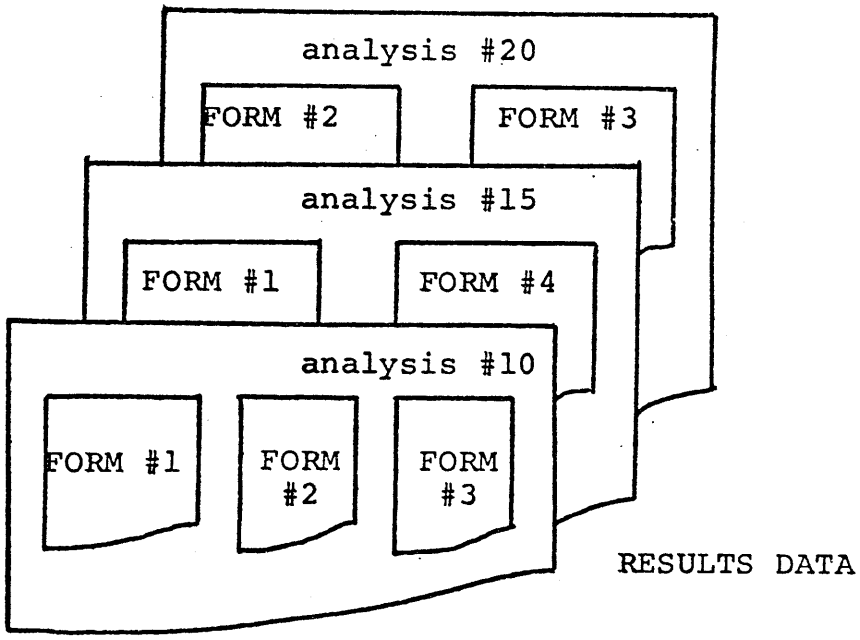


Figure III.1 - DATABANK

The following is a possible format for the raw files:

name → raw_syn_data_cruise_ {cruise_number}

entities → different observations gathered by the real time scanning system.

attributes → a) time
b) latitude
c) longitude
d) ocean_attrib #1 (I,J)
⋮
ocean_attrib #N(I,J)

where ocean_attrib # stands for different oceanographic attributes such as temperature, pressure and salinity; and I and J give a more comprehensive definition of these variables such as temperature in a certain depth I with a certain sensitivity class J.

As mentioned before, the results files may have several different formats. A typical one is shown below:

name → results_data_analysis_ {analysis_number}

entities → a. time
b. analysis_attrib #1 (I,J)
⋮
analysis_attrib #N(I,J)

where analysis_attrib # are typically statistical and mathematical properties of the different observations. The sub-

scripts I and J allow greater flexibility in defining such attributes.

3.3 The databank directory

The databank directory contains all the needed information to keep track of how and what is stored in the databank. Each time a user wants to run an analysis he will find his data by asking questions to the databank directory. In a similar way, the directory stores the status and information on data that has been saved at the end of an analysis session.

The databank directory contains files that are related to the raw data, analysis results data and some other functional files.

Figure III.2 depicts a possible format for this directory. As can be seen by Figure III.2, each cruise has three files stored in the databank directory. These are usually small files that are queried when the scientist already knows the particular cruise he is interested in. The other files are provided for more queries, as will be seen in Section 3.4. The contents and organization of all the databank directory files are fiven as follows:

NAME - raw_general_information

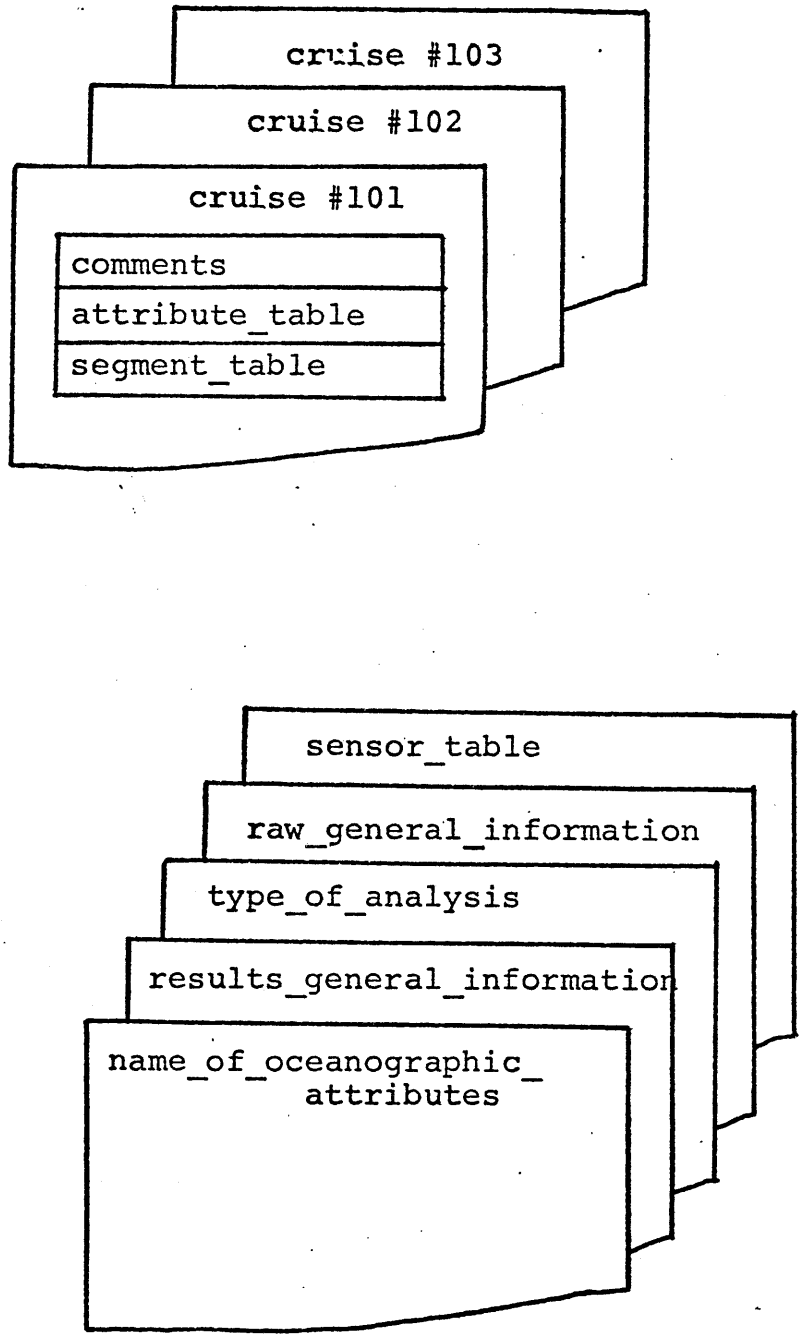


Figure III.2 DATABANK DIRECTORY

This file contains the so-called general information on each cruise that has been run by the off-shore system. The attributes are derived from the master records (tape file) and the system assigns each cruise a unique identifier called cruise_code. The file has information on the following attributes of each cruise:

cruise_code: the actual code number

cruise_date: the date the cruise was run

latitude:
{coordinates of an A PRIORI area of study
longitude:

ship_name: the name of the ship used in the cruise

institution_name: the institution sponsoring the cruise

syn_sensors_num: the number of synchronous sensors

asyn-sensors_num: the number of asynchronous sensors

cable_length: the length of the cable used in the cruise

tim_bet_syn_samples: the sampling time used with the synchronous sensors.

ocean_attrib(I): a flag to inform which oceanographic attributes were sampled.

time_start: the hour a particular cruise started

time_end: the hour a particular cruise ended.

NAME: sensor_table

This file stores information on all sensors, synchronous and asynchronous, used in all cruises, that are stored in the databank. The file keeps information on the following attributes of each sensor:

sensor_num: a code number for each sensor

sensor_type: synchronous/asynchronous

location: the location of the sensor in the towed cable

physical_variable: the physical variable (or oceanographic attribute) being measured

physical_var_units: the units for a particular physical variable

digitized_signal: the digitized signal used to acquire the physical variable.

lsb_dig_signal: the least significant bit of the digital output word from the AID on this sensor

calibration_date: the day the sensor was last calibrated

num_segments: number of linear segments comprising calibration curve

time_bet_asyn_samples: the sampling time used with each asynchronous sensor.

NAME: name_of_oceanographic_attributes

This file keeps information on the oceanographic attributes of interest to scientists. The attributes are:

ocean_attrib_id: a unique identifier for each oceanographic attribute

ocean_attrib_name: a character string representing the oceanographic attribute.

NAME: results-general-information

This file contains the so-called general information on each analysis that has been run by a certain scientist. The following attributes define each analysis within this file:

analysis_code: a unique identifier for each analysis

analysis_date: the date such analysis was performed

scientist_name: the name of the scientist

institution_name: the name of the institution sponsoring the analysis

analysis_type: a code number representing the type of analysis performed

completion_flag: a flag for telling whether the analysis has ended or not

num_saved_files: the number of saved files

basic_raw_code: the code number of the cruise raw data used in the analysis.

NAME: type_of_analysis

This file contains information on each different kind of analysis that the scientists can perform. The attributes of this file are:

analysis_type: the code number for each type of analysis

analysis_description: a brief description of this type of analysis.

NAME: comments_cruise_ {cruise_code}

This file is derived from the contents of the asynchronous raw data records contained in the tape files. During a cruise a scientist will want to store verbal information regarding events. The attributes for this file are:

time: the time the comment was recorded

latitude: } coordinates of the position where the comment was recorded
longitude:

comment: description of the comment

NAME: attribute_table_cruise_ {cruise_code}

This file keeps information on the oceanographic attributes that were recorded during a certain cruise. Attributes are:

ocean_attrib_id: the code number of the physical variable

del_dim_1 } these two attributes define the physical variable matrix acquired. As an example,
del_dim_2

if temperature was recorded for 10 different depths and each depth had 2 different sensitivity recording then

del_dim_1 = 10 and del_dim_2 = 2

NAME: segment_table_cruise_{cruise_code}

This file stores information on how the sensors, both asynchronous and asynchronous were calibrated. Attributes are:

sensor_num: the number of the sensor

sensor_type: asynchronous/synchronous

segment_num: the number of the segment

segment_value(I): the different values assigned for each sensor.

3.4 The data base language and procedures

3.4.1 Introduction

The data base language and procedures are the tools which the system provides to the scientist so that he can communicate and interact with the databank and the databank directory. All systems that have a man-machine interface must have a way to handle such an interface. This might be accomplished by a language consisting of commands which are interpreted by the machine, yielding instructions as to which actions and steps are necessary.

In the beginning of this chapter we mentioned three procedures through which a user, performing oceanographic analysis, might have to pass. Let us now take a closer and

more detailed view of these procedures, trying to build examples of how an "abstract" session would use problem oriented commands and procedures and how these commands would interact with both the databank and databank directory.

Once the researcher has successfully set up a connection with a computer facility, in terms of an on-line mode, and has reached the level of his data base management system, the following functional procedures are the natural path during an analysis.

3.4.2 Interaction

This is the phase when the user interacts with the whole system, in order to determine the piece of data he is interested in. This phase consists of queries and listings of directory files, as well as data files. By imposing restrictions or constraints on cruises and/or results attributes he narrows down and defines the logical section of data he is interested in. During this procedure, the user reads information contained in both the databank and databank directory. Therefore, during the interaction the user does not write on either the databank or the databank directory.

The actual on-line interaction can be best illustrated by examples of simple commands and the action taken by the system when interpreting these commands. An example of such commands and actions is given as follows:

default raw_general_information

action: Tells the system that the following commands will be concerned with information contained in the directory's file raw_general_information.

accept my_cruises = (cruise_date > 03-10-1975 & cruise_date < 05-10-1975) & (ship_name = NEPTUNUS)

action: This command tells the system that the scientist is interested in cruises that satisfy the restrictions given by my_cruises.

count for my_cruises

action: Before the user asks to display attributes on his cruises, he may want to know how many cruises satisfy his restrictions. The command causes the system to display the number of such cruises.

add my_cruises = & (latitude > $36^{\circ}50'$ & latitude < $40^{\circ}20'$) & (longitude > $182^{\circ}45'$ & longitude < $184^{\circ}00'$)

action: This command adds information on the scientist's restrictions. To be used when too many cruises satisfy my_cruises.

subtract my_cruises = (ship_name = NEPTUNUS)

action: This command deletes restrictions for the group of cruises, the scientist is interested in. Thus the number of cruises that satisfy my_cruises may increase. To be used when too few cruises satisfy my_cruises.

```
add my_cruises = & (cable_length > 25) & (time_bet_syn_
                    samples < 5)
```

action: See description above.

```
count for my_cruises
```

action: See description above.

```
add my_cruises = & (syn. sensors_num > 8) & (ocean_attrib =
                    temperature & pressure)
```

action: See description above.

```
display all for my_cruises
```

action: Displays all attributes in directory for the cruises that satisfy the scinetist's constraints. After having better decided the cruises he is interested in, the scientist displays information concerning these cruises.

```
display all in attribute_table_cruise_1873 for all
```

action: Given that cruise #1873 is one of the cruises satisfying my_cruises, the system displays information on the oceanographic attributes existing in the cruise #1873 raw files.

```
display location, calibration_date in sensor_table for
                    cruise_code = 1873
```

action: Displays the location and calibration_data of all sensors used in cruise #1873.

```
add my_cruises = & (calibration_date > 12-20-1974)
```

action: See description above.

display all in segment_table_cruise #1873 for all

action: Displays segment information in all segments used in cruise #1873.

display all in comments_cruise #1873 for time > 20h05min

action: Displays comments generated during the scanning cruise after a certain hour.

check my_cruises

action: The system verifies the results directory to see if someone else has already run an analysis on data satisfying these restrictions.

3.4.3 Definition

Once the scientist determined precisely the quantum of data that he wants to analyze, he will save the information concerning his restrictions in the databank directory. He is advised to do so, for 2 reasons: first, the system may crash while his analysis is under way and he definitely does not want to search and locate his analysis data again. Second, before the user starts running an analysis he may wish to verify if someone else has already worked on data satisfying his constraints.

During this phase the user writes information in the databank directory. The command to accomplish this would be of the form:

append to results_general_information,
analysis_code = 79, analysis_date = 750624,
scientist_name ='JONES', institution_name ='METEØR',
basic_raw_code = 1873

action: the system adds a new "line" to the results_general_information file. The attributes missing will be added later on.

3.4.4 Generation of temporary work files

The next step is to physically create the scientist's work files. By means of simple commands, he copies and/or merges raw and/or results files into his working files. This step is essential if one wants to assure the databank integrity. All the work is thus performed in separate "scratch" files, therefore not affecting the contents of the databank. In order to read raw data files from the databank and write them in a "scratch" work file, the following command could be used:

bring_work_file 1873

action: the command copies the raw data files with cruise_code = 1873

3.4.5 Analysis

In this phase, the scientist having defined his temporary work files, consisting of raw and/or results files, will perform several different operations to obtain results and answers regarding his problem area. This part will involve several different steps using data management, graphical displays and time series processing. Creation and deletion of attributes and entities in existing files, as well as creation of new files will be a normal operation in this phase.

In order to provide us with a feeling of what scientists might be willing to do in this phase, three different oceanographic works were analyzed (5)(8)(18). The following sections give a flavor for what these scientists want to analyze and how the system may help them in doing so.

Lets assume that we have a working file consisting of observations related to a certain cruise in a coastal region. The raw data contained in this file was collected by a thermistor chain, while the boat towing such a chain advanced at a given speed in a predetermined course. Besides having the usual time and portion (latitude, longitude) attributes the working file contains information on oceanographic attributes corresponding to each observation. Thus, the file might look as follows:

attributes: time

latitude

longitude

ocean_attrib #1 (I), ocean_attrib #2(I)

where ocean_attribs stand for physical variables such as temperature, pressure, salinity or density, and I corresponds to the number of depths covered.

A. Raw Data Displays

In the case the file were to contain temperature and salinity, a scientist would like to have a vertical profile on these variables. A possible display of temperature and salinity is depicted in the figure below. The command to request such a plotting might be

vert_profile salinity temperature depth (0,77)
lat(lat_value) long (long_value)

The command above requests a vertical profile for a certain portion (lat,long) of two physical variables: temperature and salinity, in a given range of depth: 0 to 77m.

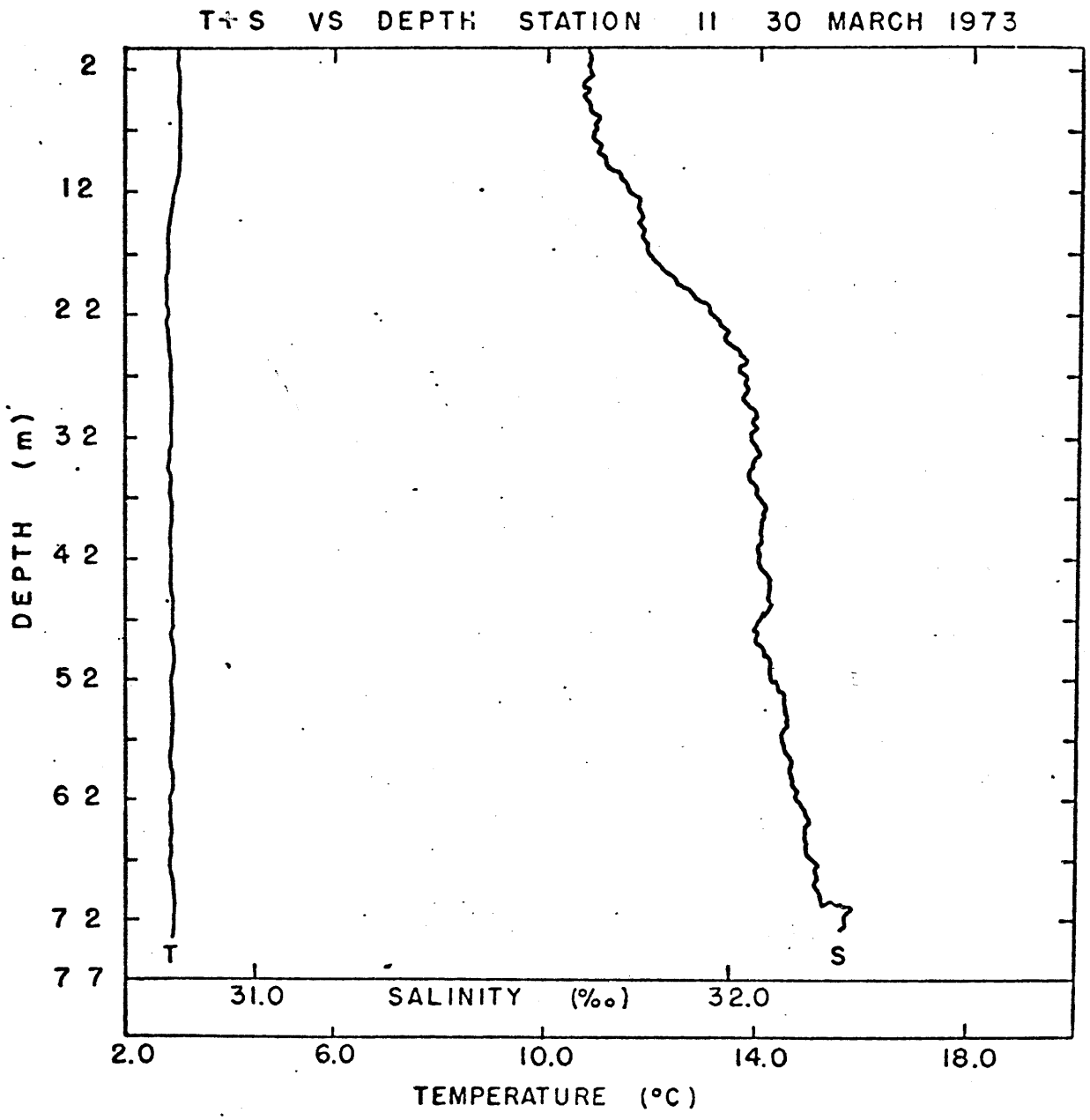


Figure III.3

Salinity and Temperature vs Depth*

---

* graph taken from Manohar-Maharaj thesis, see ref.)

B. Graphical Displays of Isolines

The user may want to have a vertical isocounter of a physical variable within a certain period of time. The following figures, Figures III.4 and III.5, depict what usually are the graphical displays that the scientist expects to see.

Assuming that his raw data was composed of temperature measurements, the command to display the vertical isotherm contours for integer isotherms between 17°C and 19°C, in a depth range of 5 to 35m, from 3PM through 10PM, might look like

```
plot_vert_iso  temp(17,19,1) depth (0,35)  time(15,22)
```

On the other hand, the user may want to have a horizontal isocontour of the variable stored in the file. So that the system can display this isoline, the user has to give additional information regarding the area and the isoline breakdown.

The figure below gives an example of horizontal salinity isocontours in Massachusetts Bay. (Figure III.6)

A possible command for plotting salinity isocontours in a certain latitude-longitude area, ranging from 28.4 to 29.6 with a 0.2 breakdown is:

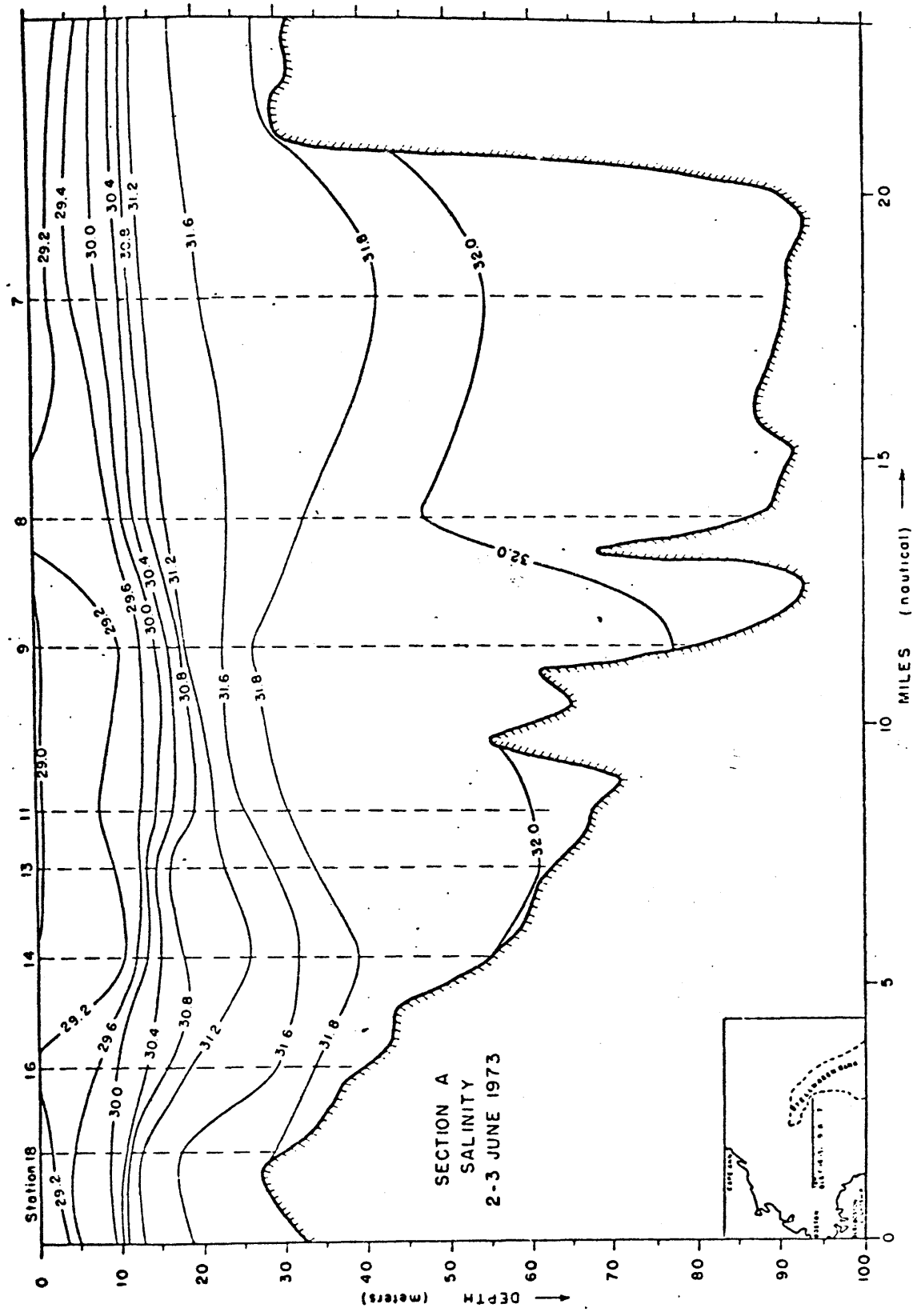
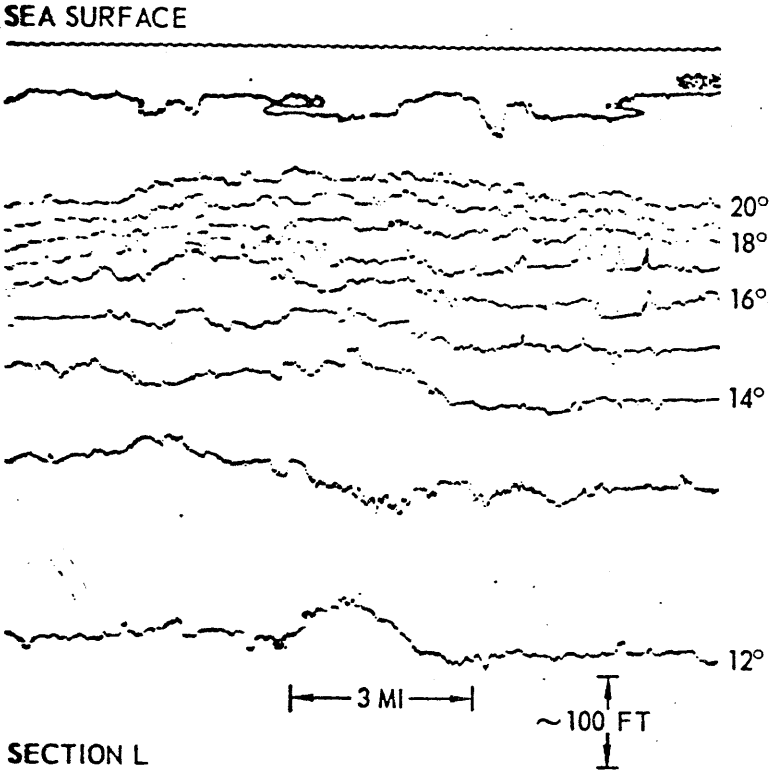


Figure III.4
Vertical salinity isolines*

---

* taken from Manohar-Majaraj thesis, see reference)



A-5

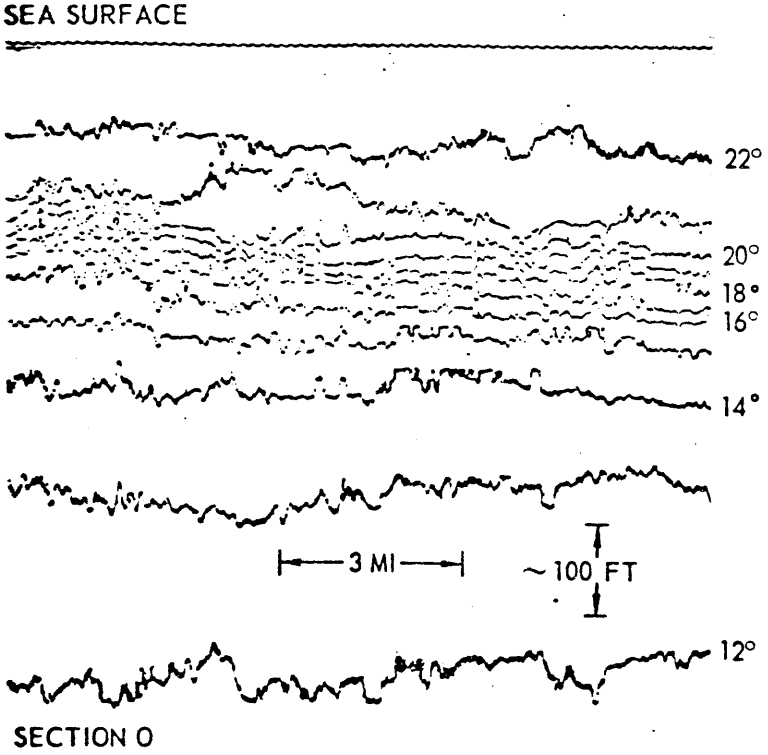


Figure III.5
Vertical temperature isolines*

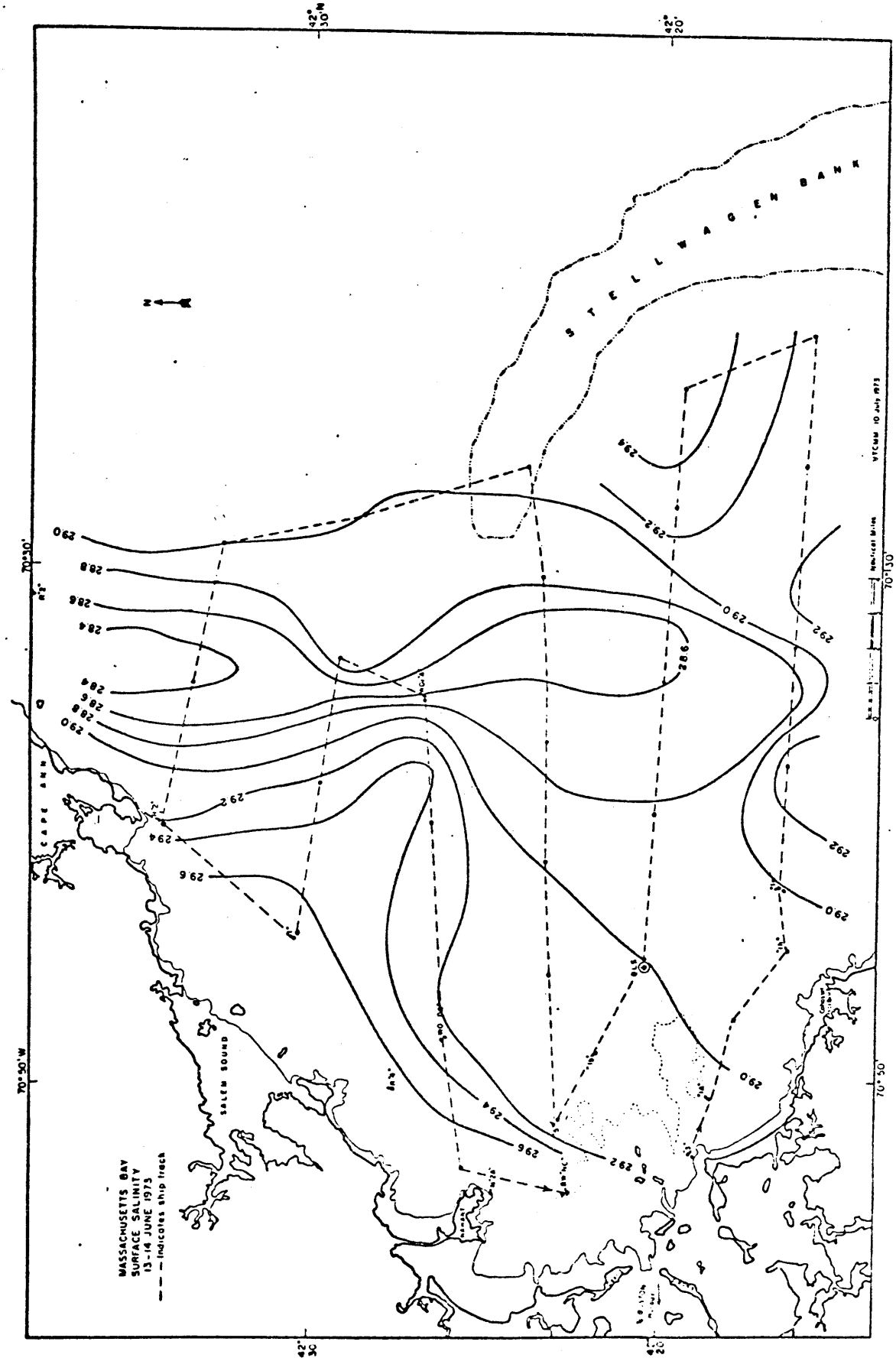


Figure III.6
Horizontal Salinity isolines

```
plot_horiz_iso   salinity (28.4,0.2,29.6)
                 lat (42°10', 42°50')
                 long (70°20', 70°50')
```

The latitude and longitude values denote the area of the present study.

C. Statistical Analysis

Let us suppose that the scientist wants to analyze isotherm variability for a specific isotherm, say 17°C. Assuming that we already have an attribute, in our temporary file, that gives for each observation the depth value for the 17°C isotherm, we may proceed by calculating another attribute, the difference of depth values, between two adjacent observations:

```
depth_dif_17 = depth_17 - depth_17(-1) $
```

Since depth_17 is a vector with as many elements as there are observations, the new vector depth_dif_17 will also be a vector with one element less than the original vector depth_17. The (-1) in the equation above denotes that there is a lag of one element between the two variables in the equation.

Once the depth differences have been calculated, usually the scientist is interested in the frequency and cumulative percentage distributions of differences in depth values for a certain isotherm. The figure below depicts a plot of such variables, identifying the central 50 & 70 percent of data.

The command to be issued asking for such a computation, must include information of the names of files where results are to be stored. The command would be:

distribution depth_dif_17 values_dif_17 freq_dif_17 cum_dif_17

action: frequency and cumulative distributions are computed using the data contained in the vector depth_dif_17. The results are stored in the other 3 files supplied by the user. If the files did not exist yet,they would be created.

To plot the results the command would be:

plot values_dif_17 freq_dif_17 cum_dif_17

In order to store certain values from the distribution computation, such as population quantile estimations, the command to be used would be:

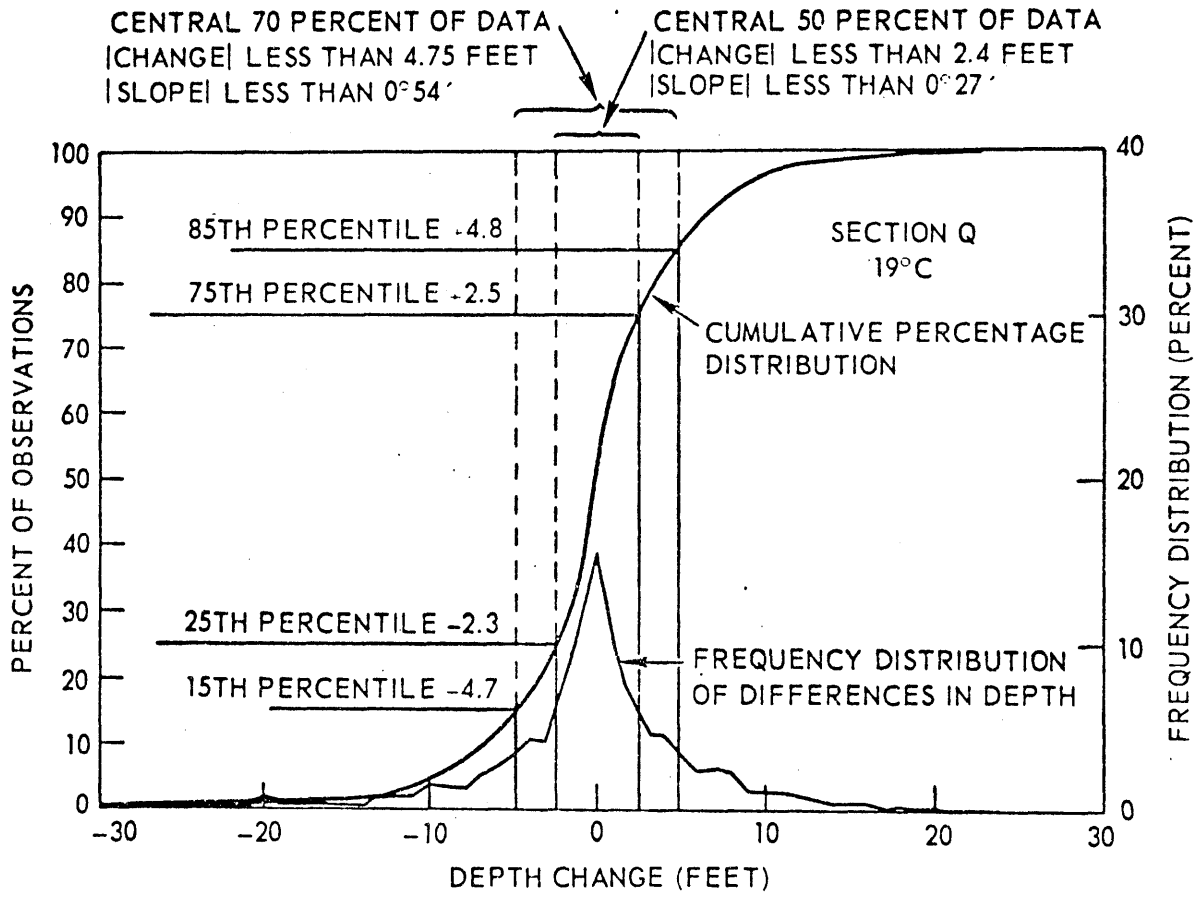

CENTRAL 70 PERCENT OF DATA
|CHANGE| LESS THAN 4.75 FEET
|SLOPE| LESS THAN 0°54′
CENTRAL 50 PERCENT OF DATA
|CHANGE| LESS THAN 2.4 FEET
|SLOPE| LESS THAN 0°27′
85TH PERCENTILE -4.8
75TH PERCENTILE -2.5
25TH PERCENTILE -2.3
15TH PERCENTILE -4.7
SECTION Q
19°C
CUMULATIVE PERCENTAGE DISTRIBUTION
FREQUENCY DISTRIBUTION OF DIFFERENCES IN DEPTH
PERCENT OF OBSERVATIONS
FREQUENCY DISTRIBUTION (PERCENT)
DEPTH CHANGE (FEET)
100
90
80
70
60
50
40
30
20
10
0
40
30
20
10
0
-30
-20
-10
0
10
20
30

Figure III.7

```
percent   depth_dif_17    50   per_50_dif_17
```

action: This command computes and stores under the name "per_50_dif_17" the central 50 percent of data computed from the input vector.

The other possible method of measuring isotherm variability is by means of autocorrelation coefficients. The figure below presents a possible plot of the auto-correlation coefficients against time. The command to be issued, would be

```
auto-correl    depth_17      lags (0,30)
```

action: computes auto correlation coefficients from 0 to 30 lags using the input vector depth_17.

The third method of representing isotherm variability is by means of power spectrum analysis. Information to be supplied to the system include the kind of window to be used, its width, the time interval between samples and others.

```
power_spectrum        depth_17        with dt = 10 $
```

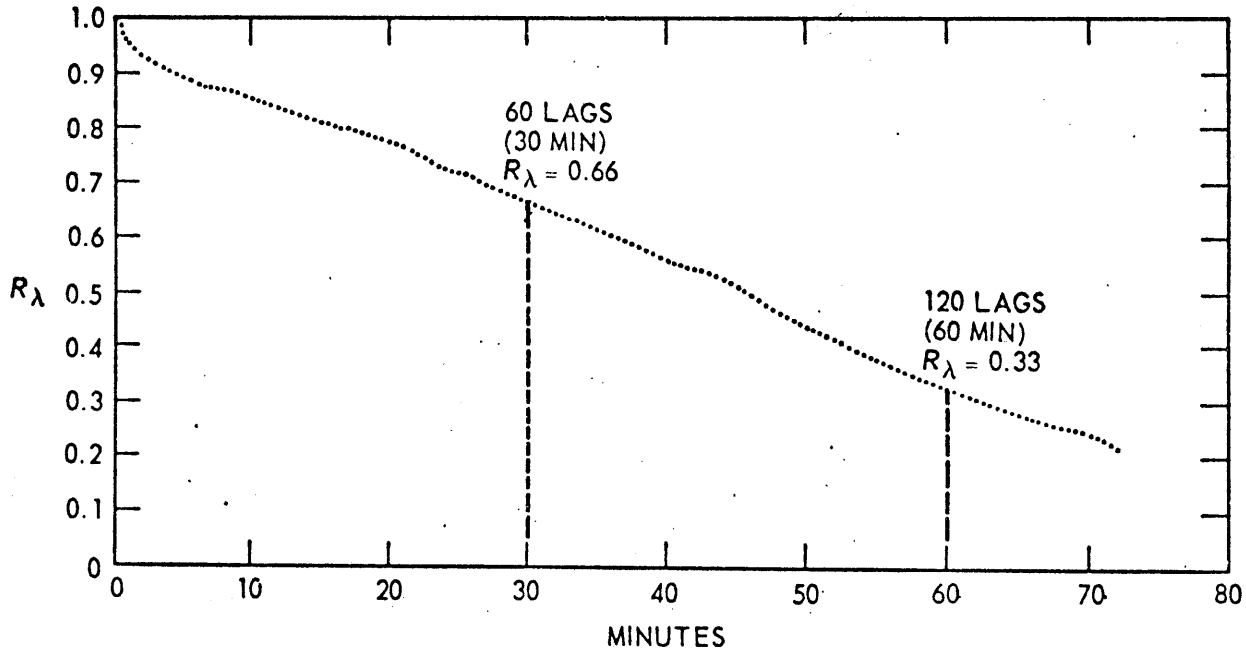

1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
R_λ
60 LAGS
(30 MIN)
R_λ = 0.66
120 LAGS
(60 MIN)
R_λ = 0.33
0
10
20
30
40
50
60
70
80
MINUTES

Figure III.8

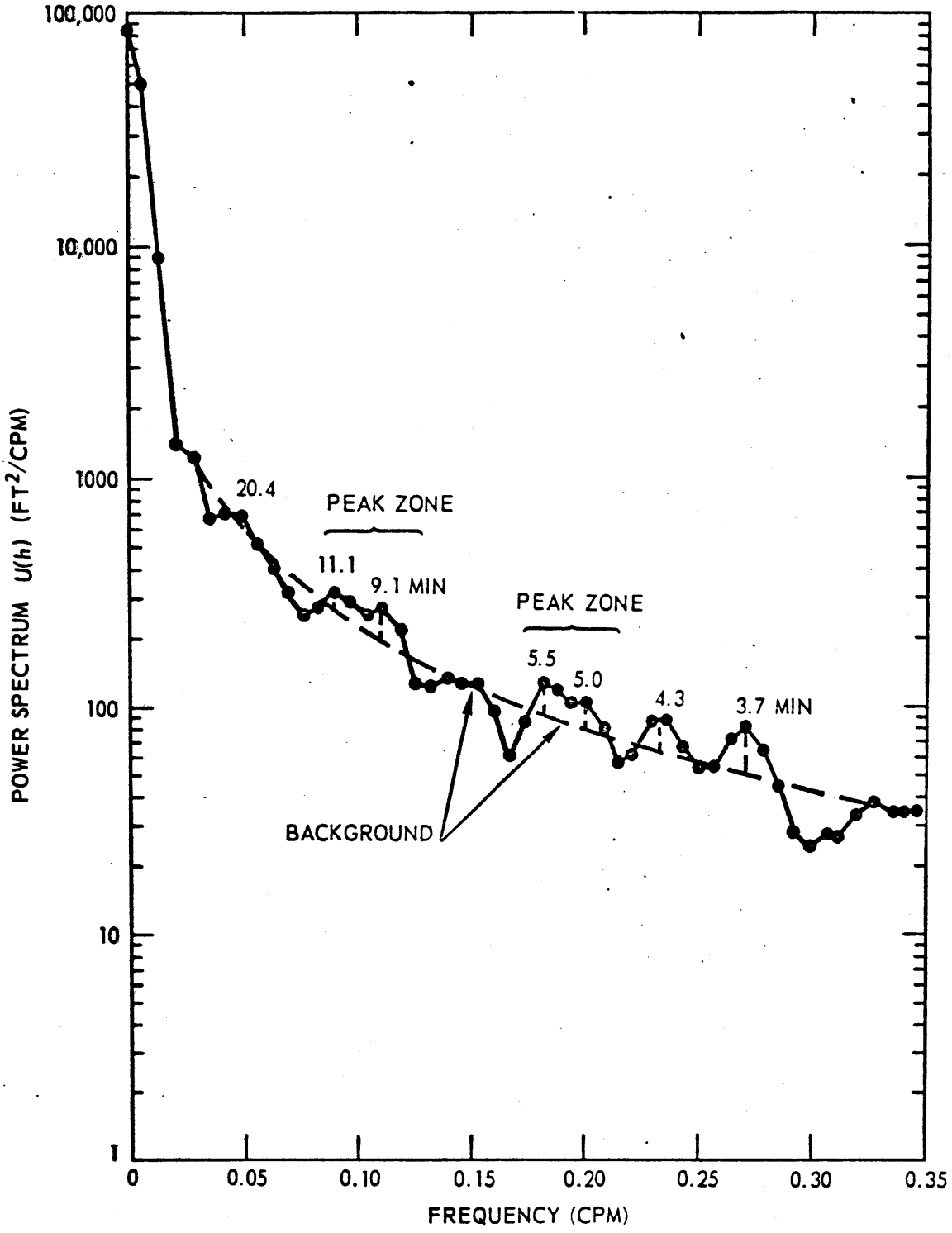

100,000
10,000
1000
100
10
1
POWER SPECTRUM U(h) (FT²/CPM)
0
0.05
0.10
0.15
0.20
0.25
0.30
0.35
FREQUENCY (CPM)
20.4
PEAK ZONE
11.1
9.1 MIN
PEAK ZONE
5.5
5.0
4.3
3.7 MIN
BACKGROUND

Figure III.9

The preceding command runs a complete spectral and cross spectral analysis using the input vector depth_17 and assuming that time between samples is 105.

3.4.6 Back-up Results

Once the scientist feels his results are satisfactory, or he thinks that he might need some off-line analysis time in order to resume work, he may be willing to store the results for his or someone else's further use. This is done in two levels: first he needs to enter information in the directory about the different characteristics of his analysis. Second, he has to copy the results files into the databank.

Given that the user already created a new analysis in the results information file, he now has to complete the attributes, which he did not write during the definition procedure. This might be done by the following commands:

```
alter in results_general_information  for analysis_code = 79,
completion_flag = 1,  num_saved_files = 3,  analysis_type = 5.
```

On the other hand, to save the results files he may use the command

```
save  <file name>
```

Chapter 4

DATA BASE MANAGEMENT TOOLS

The following chapter describes and gives a general overview of the existing software that might be used in data base management systems.

The material covered in this chapter is based on the existing software available at the M.I.T. Multics system. Among the several reasons for having chosen Multics, one might mention the initial goals of the Multics system, which were set out in 1965 by Corbata and Vynotsky:

> "One of the overall design goals of Multics is to create a computing system which is capable of meeting almost all of the requirements of a large computer utility. Such systems must run continously and reliably, being capable of meeting wide service demands: from multiple man-machine interaction to the sequential processing of absentee user jobs, from the use of the system with dedicated languages and subsystems to the programming of the system itself; and from centralized bulk card, tape and printer facilities to remotely located terminals."

Therefore, the reasons for choosing Multics are mainly based on the fact that this system provides a base for software and hardware, both in background and foreground environments that would be unpracticle for one to redesign and reprogram. The Multics system is particularly suited for the implementation of subsystems as will become evident

through the description of the Consistent System in Section 4.2; and has already developed and implemented its own graphics software package.

4.1 Multics

Multics, for Multiplexed Information and Computing Service, is a powerful and sophisticated time-sharing system based on a virtual memory environment provided by the Honeywell 6180. Using Multics, a person can consider his memory space virtually unlimited. In addition, Multics provides an elaborate file system which allows file-sharing on several levels with several modes of limiting access; individual directories, sub-directories and unrestrictive naming conventions. Multics also provides a rich repertoire of compilers and tools. It is a particularly good environment for developing sub-systems and many of its users use only subsystems developed for their field.

One major component of the Multics environment, the virtual memory, allows the user to forget about physical storage of information. The user does not need to be concerned with where his information is or on what device it resides.

The Multics storage system can be visualized as being a "tree-structured" hierarchy of directory segments. The basic unit of information within the storage system is the

segment. In such a way, a segment may store source card images, object card images, or simply data cards. A special type of segment is a directory, which stores information on all segments that are subordinated to a certain directory. The following figure depicts the Multics storage system. At the beginning of the tree is the root directory, from where all other directories and segments emanate. The library directory is a catalog of all the system commands, while the udd (user_directory_directory) is a catalog of all project directories. The same way, each project directory contains entries for each user in that project.

In order to identify a certain segment, a user has to indicate its position in the hierarchy in relation to the root directory. This is done by means of a name, called the pathname. Therefore, to refer to a particular segment or directory, the user must list these names in the proper order. The greater-than symbol (>) is used in Multics to denote hierarchy levels. Thus, to refer to segment alpha, in the figure above, the pathname would be

>udd > Proj A > user 1 > drect 1 > alpha

Each user on Multics functions as though he performs his work from a particular location within the Multics storage system; his working directory. In order to avoid

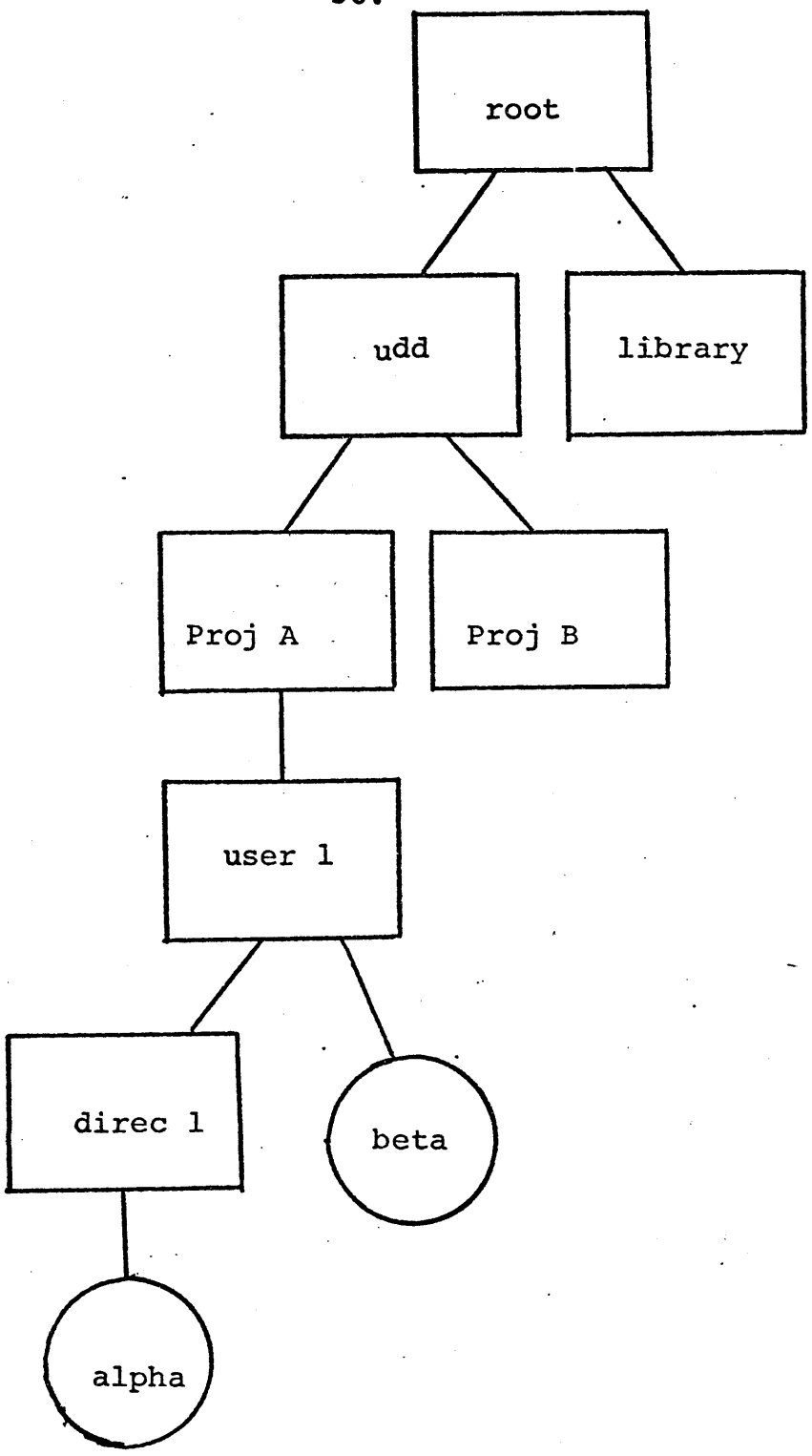


Figure IV.1

Multics hierarchical storage system

the need of always typing absolute pathnames, the user defaults a certain directory as his working directory and is able to reference segments by simple relative pathnames.

On the Multics system, the user is able to share as much or as little of his work with as many other users as he desires. The checking done by the hardware on each memory reference ensures that the access privileges described by the user for each of his segments are enforced

Besides having the universe of commands, which are available to most time-sharing environments, the Multics system provides several additional commands in order to transform the user's work in a clear, "clean" and objective stream of commands.

In order to give the general reader a flavor for what the Multics system provides, let us illustrate some commands and their meanings. Before the user can use these commands, he will have to set up a connection with the Multics system. This is usually done by means of dialing a phone number and setting up a connection between the terminal and the computer.

createdir > udd > ProjA > User 1 > Dir23

This command causes a storage system directory branch of specified name (Dir23) to be created in a specified directory (> udd > ProjA > User 1).

change_wdir > udd > ProjB > User 3 > Myd

this command changes the user's current working directory to the directory specified (> udd > ProjB > User3 > Myd).

listnames > udd > ProjA > User 1

this command prints a list of all the segments and directories in a specified directory (udd > ProjA > User 1)

print alpha

this command prints the contents of the segment alpha, which is assumed to be in the current working directory.

dprint beta

this command causes the system to print out the segment beta, using a high speed printer.

The above commands give an illustration of how the command language works. Actually these commands have powerful options which enable the user to perform various different tasks using the same basic commands. As already mentioned, the system has many more commands that might be used for manipulating directories and segments, for running programs, and perform almost any kind of on-line work.

4.2 Consistent System

The Consistent System (CS) is a subsystem within Multics on the Honeywell 6180 computer at M.I.T. Basically, the CS is a collection of programs for analyzing and manipulating data. The system is intended for scientists who are not programmers in any conventional sense, and is designed to be used interactively.

Programs in the CS can be used either single or in combination with each other. Some CS programs are organized into "subsystems", such as the Janus data handling system and the time-series-processing system (TSP). Compatibility is achieved among all elements of the system through a standardized file system.

The CS tries to let the scientist combine programs and files of data in whatever novel ways his problem seems to suggest, and combine them without getting a programmer to help him. In such an environment, programs of different sorts supplement each other, and each is much more valuable than it would be in isolation.

The foundation for consistency is the description scheme code (DSC) that is attached to each file of data. In this system, a file of data normally includes a machine readable description of the format of the data. Whenever a program is directed to operate on a file of data, it must check the DSC to see whether it can handle that scheme, and

if it cannot, must take some orderly action like an error message.

Presently there are two DSC that are of interest: "char" which is limited to simple files of characters that can be typed on the terminal, and "mnarray" which encompasses multidimensional, rectangular arrays as well as integer arrays).

To keep track of files and programs, the CS maintains directories. In a directory, the name of a file or program is associated with certain attributes, such as its length, its location in the computer, and in the case of a file its DSC.

The user typically has data files of his own, and if he has the skill and interest, he may have programs he has written for his own use. He may make each program or file of data available to all users, or keep it private.

To enter the CS, the following command should be issued from the Multics command level:

cs name

where "name" is the name of a CS directory.

In order to leave the CS, the user should type exit, and this returns the user to Multics command level.

The user operates in the CS by issuing commands from his console. When he gives a command, he types a line that

always begins with the command name, often followed by directions specifying how the command is to operate. Generally, the directions consist of a list of arguments that are separated from each other by blank space or commas. Some arguments are optional,others are mandatory, and some arguments are variables supplied by the user, while others are constants.

Occasionally, the user needs to transfer a Multics file to the CS. If such a file is located in the file system defined by the pathname

udd > ProjA > User 1 > my_segment

it can be brought into the CS in two different ways. First, let us assume that the file represents the data in "character" form. Then, the command to be issued is:

```
bring_char:a > udd > ProjA > Userl > my_segment   my_cs_seg
```

where "my_cs_seg" will be the name of the file within the CS. Let us remember that this file will have DSC "char".

On the other hand, if the Multics file actually contains binary representations of numbers, then the following command should be issued:

```
bring_mn array:a > udd > ProjA > Userl > my_segment  my_cs_seg
```

where my_cs_seg is the name of a "mnarray" file within the CS.

To save files from within the CS to Multics, the export: x command should be used. Such a command exports "mnarray" files into Multics. Files with DSC "char" are transfered by means of the put_char"x command.

There are three programs that display scatterplots, with axes, on a CRT terminal; one giving the option of connecting the points by straight lines. There is also a program that prints scatterplots on a typewriter terminal.

The Reckoner is a loose collection of programs that accept and produce files of DSC "mnarray". They give the user a way of doing computations for which he does not find provisions elsewhere in the system. There are programs that:

-- print an array on the terminal
-- extract or replace a subanay
-- do matrix arithmetic
-- create a new anay

Besides these programs, the CS offers some simple tools to perform statistical analysis. As an example there are programs to calculate frequency and cumulative frequency distributions.

It is possible to issue Multics commands from within the Consistent System. This is a very adequate and powerful

doorway, giving the CS user an almost unlimited flexibility from within the CS.

Finally, there are programs that permit the user to delete and create files, change their names, and establish references to other user's directories.

4.3 Janus

Janus is a data handling and analysis subsystem of the Consistent System. Janus is strongly oriented toward the kind of data generated by surveys, behavioral science, experiments and organizational records.

The long-range objectives of Janus include:

-- To provide a conversational, interactive language interface between users and their data.
-- To perform various common activities associated with data preparation, such as reading, editing, recoding, logical and algebraic transformations, subsetting, and others.
-- To provide a number of typewritten displays, such as labelled listings, ranked listings, means, medians, maxima and minima, cross-tabulations, and others.
-- To permit inspection of several different datasets, whether separately or simultaneously.

The following defines the data model, used in the design of the Janus system:

A dataset is a set of observations on one or more entities, each of which is characterized by one or more attributes. One example of a dataset is the set of responses to a questionnaire survey. The entities are the respondents and the attributes are the questions.

An entity is the basic unit of analysis from the scientist's point of view; it is the class of things about which the scientist draws his final conclusions. Some synonyms for the concept of an entity are: item, unit and observation.

Entities have attributes. More specifically, entities have attribute values assigned to them according to an assignment rule. Conclusions about entities are stated in terms of their assigned attribute values. Therefore, the attributes must be defined in terms of the characteristics of the entities one wishes to discuss. Synonyms for the concept of an attribute include: characteristic, category and property.

A Janus dataset provides the focus for some particular set of questions or some set of interrelated hypothesis. The raw data is read selectively into a Janus dataset by defining and creating attributes. Each user can create his own Janus dataset and analyze the data according to his own point of view.

There are 4 basic types for attributes in Janus: integer, floating-point, text and nominal. The type of an attribute determines the way it is coded in the system and the operations that may be performed on it.

An integer attribute value is a signed number which does not contain any commas or spaces, like a person's age.

A floating-point attribute value is a signed rational number, like the time, in seconds, of a trial run. This number may and is expected to include a decimal point.

A text attribute value is a character string which may include blanks, like a person's name.

Finally, a nominal attribute value is a small, positive integer which represents membership in one of the categories of the attribute, like a person's sex, 1 being for male and 2 for female.

Janus automatically maintains entity identification numbers within a Janus dataset. Janus prints out the entity numbers associated with the attribute values when the display command is used. These entity numbers can be used in commands such as display and alter to specify the particular entities to be referenced. Entities can also be referenced in a command by defining a logical condition for an attribute which only certain entities can satisfy. The logical condition specifies a subset of entities to be referenced in a command, such as display or compute.

Attribute values can be referenced in a command by specifying both an attribute name and entity numbers or a logical condition. Logically, the attribute values are being referenced by row (entity) and column (attribute).

4.4 Time Series Processor

The time series processor (TSP) is an interactive computer language for the statistical analysis of time series and cross sectional data. Using a readily understandable language, the user can transform data, run it through regressions or spectral analysis, plot out the results and save the files with results obtained.

Because of the difficulty of programming completely general language interpreters, a feasible program must establish its own syntax. A syntax is made up of a series of conventions that, in a computer language, are quite rigid.

A command is made up of a series of one or more names, numbers or special symbols. The purpose of a command is to communicate to the program a request that some action be taken. It is up to the user to structure the request so that the action taken is meaningful and productive. The program checks only for syntax errors and not at all for the meaningfulness of the request.

The "end" command tells the program to stop processing the stream of typed output and to return to the first com-

mand typed after the last end to begin executing all of the commands just typed in the order they were presented to the program. After all these commands have been executed, the program will again start processing the characters the user types at the console.

The basic unit of data within TSP is the variable. The variable in TSP commands corresponds to the attribute in Janus. An observation in TSP corresponds to an entity in Janus or the Consistent System.

A variable is referred to in TSP by a name assigned to the variable. Name assignments occur by the use of a generation equation. Names assigned in Janus or CS are carried over to TSP if the databank command has been executed.

Whenever a variable is referred to in a command, the program retrieves the required data automatically and supplies it to the executing procedure. The user may specify the subset of observations that are to be used in the execution of a command. This is done by means of the "smpl" command. The subset of observations thus defined will be used for every command until replaced by another "smpl" command.

The user may shift the scale of observations of one variable relative to another. The displacement of the scale of observations is indicated by a number enclosed in parenthesis typed following the variable name in any command to be executed. A lag of one so that the immediately proceding

observation of the variable lagged would be considered along with the current observation of one or more others, would be indicated by A(-1).

The GENR procedure generates new variables by performing arithmetic operations on variables previously loaded or generated. The arithmetic statements used in GENR are very similar to FORTRAN or PL I statements, but a knowledge of these languages is not at all necessary.

Among useful TSP commands, one may include

OLSQ - carries out a ordinary least squares and two stage least squares estimation.

CORREL-prints out a correlation matrix of any set of variables which have previously been loaded or generated.

SPECTR-performs a complete spectral and cross-spectral analysis of a list of one or more variables.

Chapter 5

SYSTEM IMPLEMENTATION

Our objective in this Chapter shall be to closely follow the sequence of topics described in Chapter 3, showing how they might be implemented through the use of the tools and software described in Chapter 4.

5.1 File System

Using the Multics environment and storage system concepts described earlier, Figure V.1 depicts a "tree-structured" hierarchy of our data base file system.

The whole data base is contained in the project OCEAN directory. Under it we have directories related to the data bank directory, the databank itself and as many scientist directories as different oceanographic users exist.

5.1.1 The databank directory

The databank directory is contained under a CS directory labelled as Dtbkdir. It is made up of several Janus datasets and files that are described in the following pages. Whenever a new cruise tape file is loaded into the database, this directory is updated and/or changed accordingly.

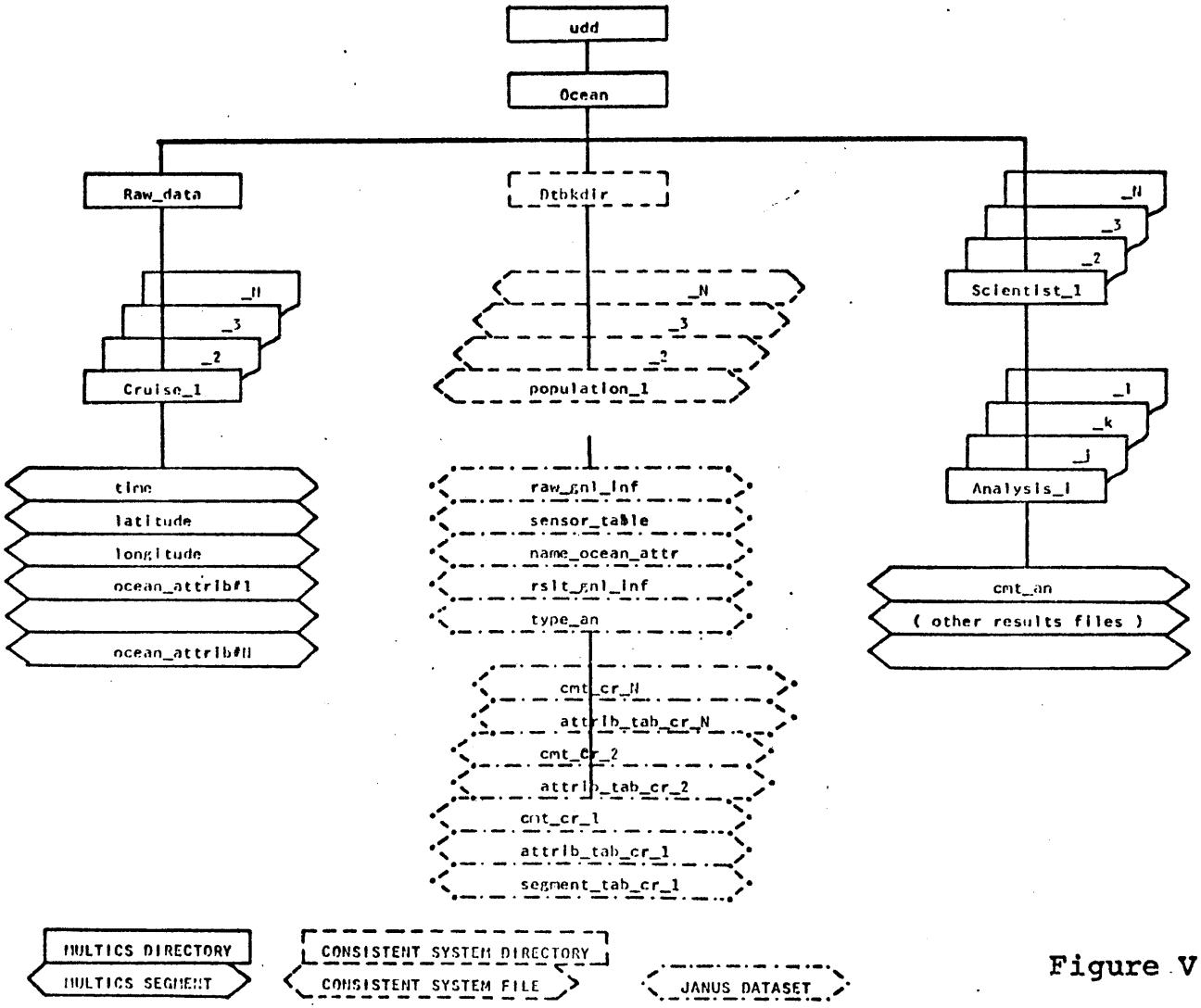


Figure V.1

General Data-Base File System

directory - Dtbkdir

file type - Janus dataset

NAME - raw_gnl_inf

CONTENTS - contains general information on raw data files. Each cruise is assigned an identifier called cruise_code.

ENTITIES - different cruises.

ATTRIBUTES-

name	type	example
cruise_code	integer	173
cruise_data	integer	750611
latitude	float	+45.50
longitude	float	-71.25
ship_name	text	NEPTUNUS
institution_name	text	METEØR
syn_sensors_num	integer	12
asyn_sensors_num	integer	3
cable_length	float	50.0
time_bet_syn_samples	float	1.50
num_columns_raw	integer	120
ocean_attrib(N)	integer	YES/NO (1/0)
time_start	text	9:32:06
time_end	text	14:05:10

directory - Dtbkdir

file type - Janus dataset

NAME - sensor_table

CONTENTS - contains information on the sensors, synchronous and asynchronous that were used during the cruises.
different sensors.

ATTRIBUTES -

name	type	example
cruise_code	integer	187
sensor_num	integer	4
sensor_type	integer	(1/0) ASYN/SYN
location	float	25.0
physical_variable_id	integer	12
physical_var_units	text	DECIBARS
digitized_signal	text	VOLTS
lsb_dig_signal	float	0.005
calibaration_date	integer	750608
time_bet_asyn_samples	float	2.50
num_segments	integer	3

directory - Dtbkdir

file type - Janus dataset

NAME - name_ocean_attr

CONTENTS - each oceanographic attribute is assigned a unique identifier and name

ENTITIES - different oceanographic attributes

ATTRIBUTES -

name	type	example
attrib_id	integer	11
attrib_name	text	TEMPERATURE

directory Dtbkdir

file type - Janus dataset

NAME - rslt_gnl_inf

CONTENTS - contains general information on results data files. Each interactive session is assigned an identifier called analysis_code.

ENTITIES - different analysis sessions.

ATTRIBUTES -

name	type	example
analysis_code	integer	27
analysis_date	integer	150611
scientist_name	text	JONES
institution_name	text	METEOR
analysis_type	integer	4
completion_flag	integer	YES/NO(1/0)
num_saved_files	integer	5
basic_raw_code	integer	187

directory Dtbkdir

file type - Janus dataset

NAME - type_on

CONTENTS - each type of analysis performed by the scientist has an identifier and attached description.

ENTITIES - different types of analysis.

ATTRIBUTES -

name	type	example
analysis_type	integer	4
description	text	SPECTRAL ANALYSIS

directory - Dtbkdir

file type - Janus dataset

NAME - cmt_cr_{cruise_code}

CONTENTS - stores the comments recorded in the asynchronous data records during a certain cruise.

ENTITIES - different comments.

ATTRIBUTES -

name	type	example
time	float	8.15132 {8 hours and $15132/100000$ of hour}
latitude	float	41.52 (same as time)
longitude	float	70.79 (same as time)
comment	text	"PASSING THROUGH THERMAL FRONT"

directory - Dtbkdir

file type - Janus dataset

NAME - attrib_tab_cr {cruise_code}

CONTENTS - stores information on all the oceanographic attributes acquired during a certain cruise.

ENTITIES - different oceanographic attributes.

ATTRIBUTES -

name	type	example
attrib_id	integer	11
del_dim_1	integer	8 (number of rows for attrib_id=11)
del_dim_2	integer	1 (number of cols for attrib_id=11)
field_length	integer	5 (number of digits)
precision	integer	1 (number of digits right to decimal point)

directory - Dtbkdir

file type - CS file with DSC "mnarray"

NAME - population {cruise_code}

CONTENTS - contains the number of entities of the raw data files stored in the databank.

5.1.2 The databank

The databank resides under a Multics directory labeled as Raw-data. This directory contains as many subdirectories as there are different cruise-codes. The files contained within each Cruise-{cruise_code} directory consist of two types: the time, latitude and longitude files are always present, while the ocean_attrib files contain data related to physical variables such as temperature, pressure and salinity, that depend on each cruise. The raw data files are loaded into the data base, whenever a new cruise tape file is processed by an interface program. These files are stored in binary form, thus enabling storage space saving.

At this point, it should be mentioned how certain variables are logically stored. Given that time, latitude and longitude are usually referred to in a "non-decimal" way, like time = 8 hours 6 min 35 seconds, or latitude = 35°N 36' 15", that presents computational problems, it was decided to store them in an equivalent decimal form. As an example:

$$45^{\circ}\text{N } 37' \ 42'' \equiv +45.62851^{\circ}$$

and

$$8 \text{ hours } 37' \ 42'' \equiv 8.66851 \text{ hours.}$$

5.1.3 The Scientist directories

Each active user of the IDSS data base is assigned a Multics directory under the OCEAN directory. Each such directory contains a number of affiliate directories that are related to the different analysis performed by the scientist. This is needed, since different users will perform different analysis and will save different results. The user should refer to Fig. V-1 to understand this point.

5.2 The On-Line Session

The following section illustrates an example of a real session, and follows closely the outline given in Section 3.4 - Data base Language and Procedures.

The Figure below (Fig. V.2) presents the data base as it was structured for the on-line sessions. Basically it is identical to Fig. V.1, the only difference being that during the production sessions, two extra directories were used between the Multics add directory and the project Ocean directory. This was needed since the funds for the on-line sessions came from the Cambridge Project.

The approach used in this section was to divide it in 5 functional modules: interaction, definition, work files generation, analysis and results back-up. Each module

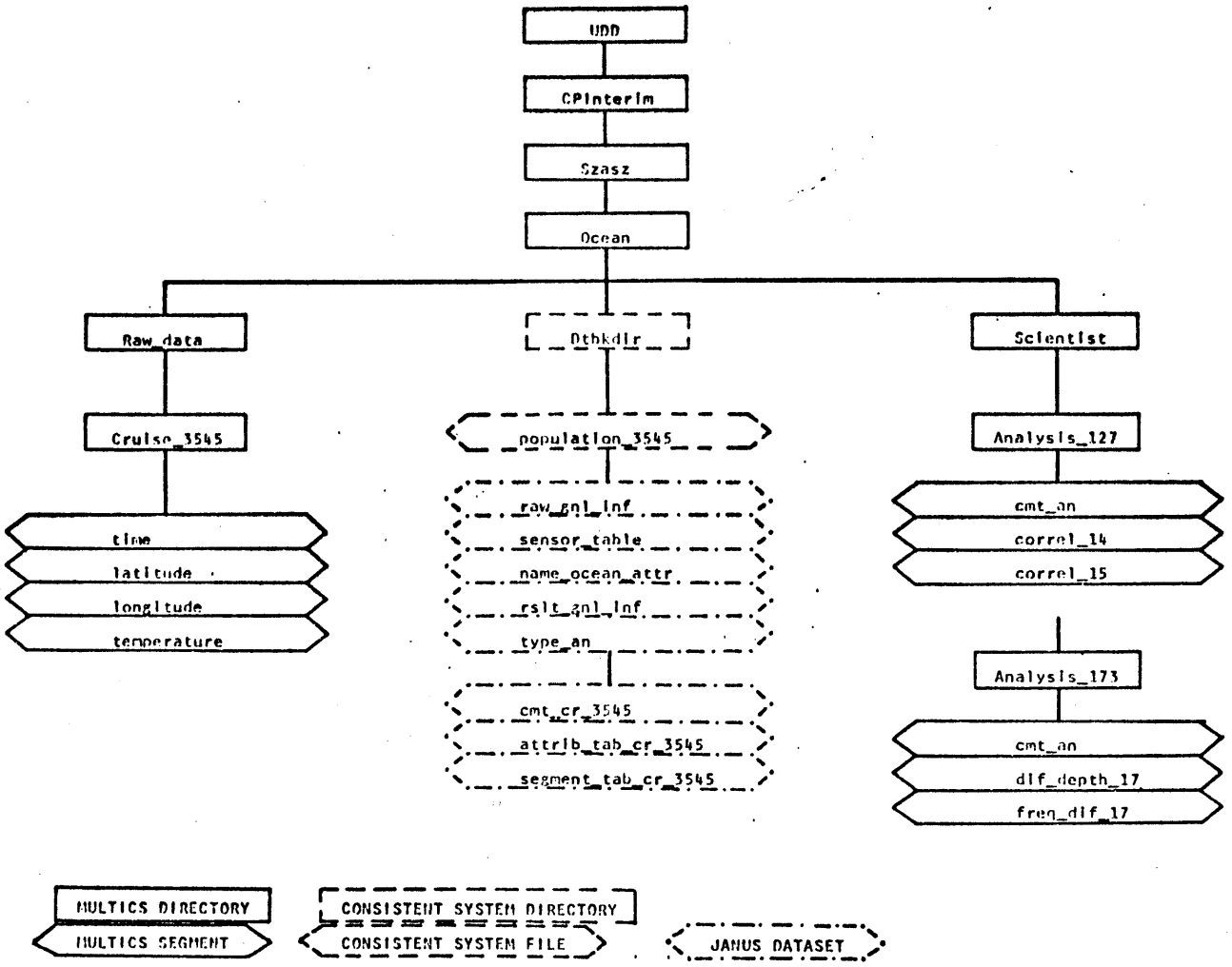


Figure V.2 - Experimental Data-Base File System

consists of two parts: an explanation of the actual commands used and then attached a copy of the working version as implemented on a typewriter console. For clarity and easy understanding, the commands are numbered and explained in the first part.

5.2.1 - Interaction

This phase consists basically of three steps:

1. Queries regarding raw data file.
2. Queries verifying if the analysis, the scientist has in mind, was done before.
3. Listing of directory files related to the specific cruise(s), the scientist is interested in.

Given that the databank directory files are contained in a CS directory, and furthermore are defined within the Janus system, the first step for the scientist is to enter the Janus system.

1 The user presently at Multics command level enters the databank directory Dtbkdir.

2 The user identifies to the CS, the foreign directory 'x'.

3 Enters Janus.

4 Informs the system that subsequent commands are concerned with the dataset raw_gnl_inf.

5 6 7 Places queries to the databank directory, imposing constraints on the raw_gnl_inf file attributes.

8 Assuming the user is interested in raw data files, he asks the system what is the attribute identification for TEMPERATURE.

9 10 11 The user continues his queries.

12 After having only one cruise satisfying his constraints, he displays all information on this cruise.

13 14 15 16 Leaves Janus, exits from CS, goes into the cruise_3545 Multics directory and lists the names of the raw data files contained in this directory.

17 18 19 20 Returns to Janus command level.

21 Lists information concerning the sensors used in Cruise 3545.

22 Verifies if there has been any previous analysis using the raw data files of Cruise 3545.

23 Gets the description of the analysis type used in Analysis-127.

24 25 26 27 Returns to Multics level in directory Analysis-127 and lists the files contained in this directory.

28 Prints out the comments file of this analysis.

29 30 31 32 Returns to Janus.

33 The user mistypes the command and receives an error message.

34 Prints out the comments generated in Cruise 3545.

35 Prints out information on the physical variables acquired during cruise 3545. Only TEMPERATURE is stored (attrib_id=11), and the samples include 8 depths.

5.2.2 - Definition

Assuming the scientist, after analyzing the general contents of a particular raw data file by interacting with the databank directory, wants to proceed and perform an analysis, he has to save information regarding his interests.

Remembering that the user is still within the Janus system, the following shows how to perform this procedure.

1 The user appends a new entity to the results general information file.

2 The user displays the new entity. The character "*" means "missing", these attributes will be modified at the end of the analysis.

3 Leaves the CS.

5.2.3 - Work Files Generation

To proceed with the session, and in order to leave the contents of the databank unchanged, the user copies the raw data files into the Consistent System directory.

This is done since most of the analysis is done within the CS or its aggregated subsystems.

To copy Multics segments into the CS we use the following set of commands. The reader should note that before copying files the user prints out the number of entities in the raw data files, so that he can supply it to the CS commands.

1 displays the number of entities within the raw data files.

2 3 4 5 Brings into the CS the raw data files supplying the number of entities.

5.2.4 - Analysis

The scientist, having created his temporary work files, is now ready to perform the analysis and/or modeling on his data.

The scientist first wants to have a plotting of raw data. Given that he has a temperature array corresponding to a scanning run, where the physical variable temperature was sampled each 10s at 8 different depths, he now wants to plot temperature vs. depth for a given location, defined either by time or by means of the coordinates (lat, long) of a spot

in the ocean.

1 2 Enters Janus.

3 Brings into Janus the files.

4 Determines a certain spot for plotting his raw data: the 13th entity or the 13th line in the temperature anay.

5 Leaves Janus.

6 Gets a vector (the 13th line) of the temperature anay.

7 Builds the depth vector, using information from the sensor-table file.

8 Plots temperature against depth using a CRT device.

Assuming the scientist has produced a vector (depth-17) that contains the depths of the 17°C isotherm, the following command produces a plot on a CRT device of time vs. depth-17.

```
9       plot 3:x  time depth-17    -xscale   10.003
                                             10.069
```

where xscale denotes the range of the variable time the user is interested in. (10.003 → 10.069 hours)

10 11 The user enters TSP and asks for a short ready message.

12 13 14 Brings into TSP the vector depth-17.

15 produces the vector dif_depth_17, differences in depth from point to point along the 17°C isotherm.

16 17 Saves the vector into the CS and returns to CS command level.

18 Extracts the first element of the vector dif_depth_17.

19 Sorts the vector in increasing order.

20 21 22 23 Calculates and prints population quantile estimations, namely the 50th and 70th central percents.

24 25 Calculates and plots a frequency distribution of the dif_depth_17 vector.

26 Plots the same frequency distribution on a CRT device.

The other two ways of measuring isotherm variability is by means of autocorrelation coefficients and power spectrum analysis. The user once more enters TSP, determines his sample as being entities 1 through 100 and issues the correl command, informing the system the particular values of lags he is interested in.

Next the SPECTR command is used, specifying plots as an option, and the time interval to be considered as 10 sec. As an output the scientist gets several variables resulting from the univariate spectrum analysis of depth-17. Furthermore, the system plots relevant variables against frequency, so that the user may have a graphical view of his results.

27 28 Enters TSP

29 Specifies all print out to be directed to the console.

30 Generates a correlation matrix.

31 32 Performs a spectral analysis.

5.2.5 - Results Back-up

Due to the data base structure, this procedure starts with the creation of a new directory: Analysis-173, directly subordinated to the scientist directory. Next the user creates the comment file (cmt_an) and copies it into the recently created directory. Having decided that the vectors dif_depth_17 and freq_dif_17 might interest him in the future, he saves them, again under the Analysis_173 directory.

The CS files are deleted and the user enters Janus to append information to the databank directory.

Since the analysis performed by this user was not described in the type of analysis file, he creates a new analysis type and stores it in the databank directory.

After having altered information in the results general information file, the user returns to the CS, deletes the temporary work files and enters the Multics directory Analysis-173 to list the files saved. Given that the files were saved, the whole analysis is over and the scientist logs out from the system.

1 Creates a new Multics sub-directory Analysis-173.

2 Loads a comment file into the CS.

3 4 5 Saves the files into Multics, directory Analysis-173.

6 Deletes files.

7 8 9 Enters Janus and defaults the results general information file.

10 11 12 Replaces missing values for num_saved_files and completion_flag in the defaulted data set.

13 14 Creates a new analysis type in the file type of analysis.

15 Replaces missing value for analysis_type.

16 Makes sure everything is all right.

17 18 19 20 Goes into the Multics directory Analysis-173.

21 Lists the name of the files that have been saved.

(1)
```
cs Dtbkdir
CS, Version 2
R 1.027
```

(2)
```
cdd >udd>cpc>P>x
R .255
```

(3)
```
janus:x
J
```

(4)
```
default raw_gni_inf
J
```

(5)
```
display cruise_code, for cruise_date > 751201

cruise_code
#

4    1343
5    2789
6    3545
7    6007
8    9413

J
```

(6)
```
ds cruise_code, for cruise_date > 751201 & ( ship_name = "NEPTUNUS" | ship_name = "MARYSVIL")

cruise_code
#

4    1343
5    2789
6    3545
8    9413
J
```

(7)
```
ds cruise_code, for cruise_date > 751201 & latitude > 40.5 & latitude < 43.0 & longitude > 70.0 & longitude < 74.0

cruise_code
#

6    3545
7    6007
8    9413
J
```

(8)
```
ds attrib_id, in name_ocean_attr for attrib_name = "TEMPERATURE"
```

Figure V.3a

```
attrib_id
#

1   11
J
```

(9) ds cruise_code, for latitude > 40.5 & latitude < 43.0 & longitude > 70.0 & longitude < 74.0 & ocean_attrib(11) = 1

```
cruise_code
#

1      1
2     17
6   3545
7   6007
8   9413

J
```

(10) ds cruise_code, for latitude > 40.5 & latitude < 43.0 & longitude > 70.0 & longitude < 74.0 & ocean_attrib(11) = 1 & & cable_length > 35.0 & syn_sensors_num > 7

```
cruise_code
#

1      1
6   3545
7   6007
J
```

(11) ds cruise_code, for cruise_date > 751201 & latitude > 40.5 & latitude < 43.0 & longitude > 70.0 & longitude < 74.0& & ocean_attrib(11) = 1 & cable_length > 35.0 & syn_sensors_num > 7 & time_bet_syn_samples > 5

```
cruise_code
#

6   3545
J
```

(12) ds all, for cruise_code = 3545

```
                                                                     syn_sensors_num
                                                                  asyn_sensors_num
                 time_start                                  cable_length                       ship_name
#  time_end                              ocean_attrib      time_bet_syn_samples           institution_name        longitude

6   13:14:45   10:50:00   0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0   10.000   45.000   2   8   METEOR   MARYSVIL   71.250

  latitude         cruise_code
#          cruise_date

6   41.250   760708   3545
```

Figure V.3b

```
(13) leave
     R 1.529

(14) exit
     r 1859 10.426 269.166 2669

(15) change_wdir >udd>CPInterim>Szasz>Ocean>Raw_data>Cruise_3545
     r 1859 .153 .600 29

(16) listnames

     Segments= 4, Records= 5.

     temperature
     longitude
     latitude
     time

     r 1859 .380 1.332 17

(17) cwd >udd>CPINterim@ cwd >udd>CPInterim>Szasz
     r 1900 .096 .930 22

(18) cs Dtbkdir
     CS, Version 2
     R .649

(19) cdd >udd>cpc>P>x
     R .171

(20) Janus:x
     J
```

Figure V.3c

(21) ds all, in sensor_table for cruise_code = 3545

```
                time_bet_asyn_samples          physical_var_units        sensor_num
sensor_type          calibration_date   digitized_signal                  location
#   num_segments                  lsb_dig_signal          physical_variable_id     cruise_code

1      0   3   0.0000   760702   .005000   VOLTS   CELSIUS   11   45.000   1   3545
2      0   3   0.0000   760702   .005000   VOLTS   CELSIUS   11   40.000   2   3545
3      0   3   0.0000   760702   .005000   VOLTS   CELSIUS   11   35.000   3   3545
4      0   3   0.0000   760702   .005000   VOLTS   CELSIUS   11   30.000   4   3545
5      0   3   0.0000   760702   .005000   VOLTS   CELSIUS   11   25.000   5   3545

6      0   3   0.0000   760702   .005000   VOLTS   CELSIUS   11   20.000   6   3545
7      0   3   0.0000   760702   .005000   VOLTS   CELSIUS   11   15.000   7   3545
8      0   3   0.0000   760702   .005000   VOLTS   CELSIUS   11   10.000   8   3545
J
```

(22) ds all, in rslt_gnl_inf for basic_raw_code = 3545

```
                 analysis_type
            completion_flag
basic_raw_code      institution_name        analysis_date
#    num_saved_files            scientist_name     analysis_code

1     3545   3   1   4   METEOR   SCIENTIST   760825   127
J
```

(23) ds description, in type_an for analysis_c#type = 4

```
#          description

4   CORRELATION ANALYSIS
J
```

(24) leave
R 17.101

(25) exit
r 2231 19.589 439.104 4554

(26) change_wdir >udd>CPInterim>Szasz>Ocean>Scientist>Analysis_127
r 2231 .173 2.212 44

(27) listnames

Segments= 3, Records= 1.

Figure V.3d

```
cmt_an
correl_15
correl_14

r 2232 .262 2.450 47
```

(28)
```
print cmt_an

                        cmt_an    06/21/74  2232.3 edt Fri


Today's analysis was intended to run a correlation analysis on the 15C
and 14C isotherms. Response time was a loser , therefore decided to re-
sume analysis another day. Files created contain correlation matrixes for
the two isotherms and range from zero to 10 lags.


r 2232 .381 1.722 36
```

(29)
```
cwd >udd>CPInterim>Szasz>#
r 2233 .077 .462 12
```

(30)
```
cs Dtbkdir
CS, Version 2
R .710
```

(31)
```
cdd >udd>cpc>P>x
R .243
```

(32)
```
janus:x
J
```

(33)
```
ds all, in cmt_cr_3545
display:  Check for missing 'for' or 'thru' statement at '
'
J
```

(34)
```
ds all, in cmt_cr_3545 for all

                                        latitude
#              comment                 longitude              time

1   PASSING THROUGH THERMAL FRONT       71.734    41.268    11.520
J
```

(35)
```
ds all , in attrib_tab_cr_3545 for all

          attrib_id
del_dim_2
#   del_dim_1

1    1   8   11
J
```

Figure V.3e

(1)
```
append to rslt_gnl_inf analysis_code 173 / analysis_date 761015 / scientist_name "SCIENTIST" / A
       institution_name = "###"METEOR" / basic_raw_code 3545
J
```

(2)
```
ds all, in rslt_gnl_inf for analysis_code = 173

                 analysis_type
            completion_flag
basic_raw_code      institution_name        analysis_date
#     num_saved_files             scientist_name     analysis_code

2      3545    *    *    *    METEOR    SCIENTIST    761015    173
J
```

(3)
```
leave
R 3.466
```

Figure V.4

```
(1)  print_mnarray:a population_3545
     dims =  1-     1

       100.000


     R .277

(2)  bring_mnarray:a >udd>CPinterim>Szasz>Ocean>Raw_data>Cruise_3545>time time -float
     1 100
     R .372

(3)  bring_mnarray:a >udd>CPinterim>Szasz>Ocean>Raw_data>Cruise_3545>latitude latitude -float
     1 100
     R .164

(4)  bring_mnarray:a >udd>CPinterim>Szasz>Ocean>Raw_data>Cruise_3545>longitude longitude -float
     1 100
     R .167

(5)  bring_mnarray:a >udd>CPinterim>Szasz>Ocean>Raw_data>Cruise_3545>temperature temperature -float
     2 100 8
     R .168
```

Figure V.5

```
(1)     cdd >udd>cnc>P>x
        R .483

(2)     janus:x
        J

(3)     get_mnarray time latitude longitude in analysis_173
        J

(4)     display longitude time, in analysis_173 for latitude < 41.47 & latitude > 41.45

                longitude
          #                time

        13    71.734    10.036
        J

(5)     leave
        R 6.008

(6)     subarray:x temperature 1 1 13, plot_t
        R .774

(7)     make_vector:x 45.0 40.0 35.0 30.0 25.0 20.0 15.0 10.0 depth
        R .908
```

Figure V.6a

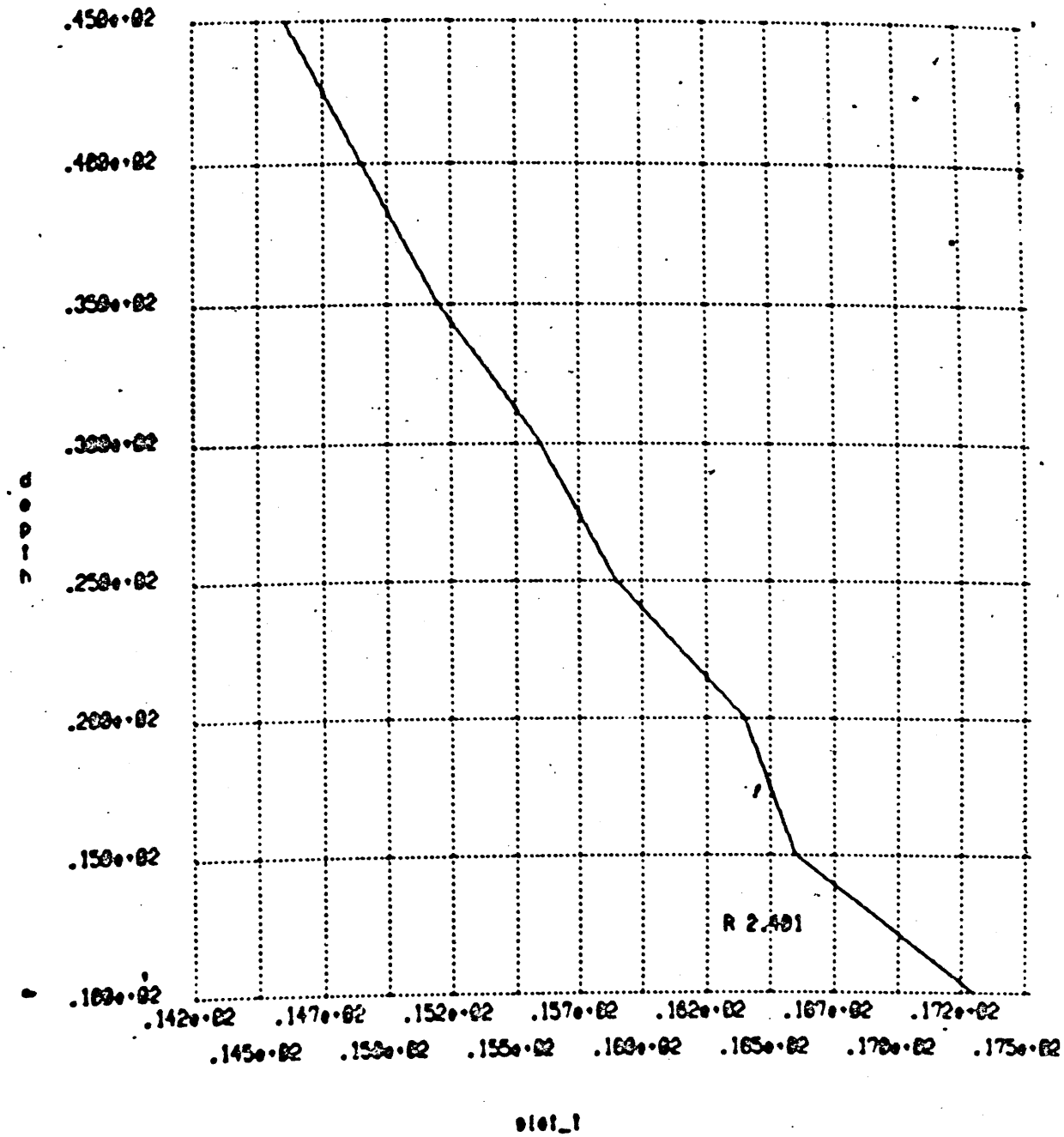

.450e+02
.400e+02
.350e+02
.300e+02
.250e+02
.200e+02
.150e+02
.100e+02
depth
.142e+02
.147e+02
.152e+02
.157e+02
.162e+02
.167e+02
.172e+02
.145e+02
.150e+02
.155e+02
.160e+02
.165e+02
.170e+02
.175e+02
R 2.401
plot_1

Figure V.6b

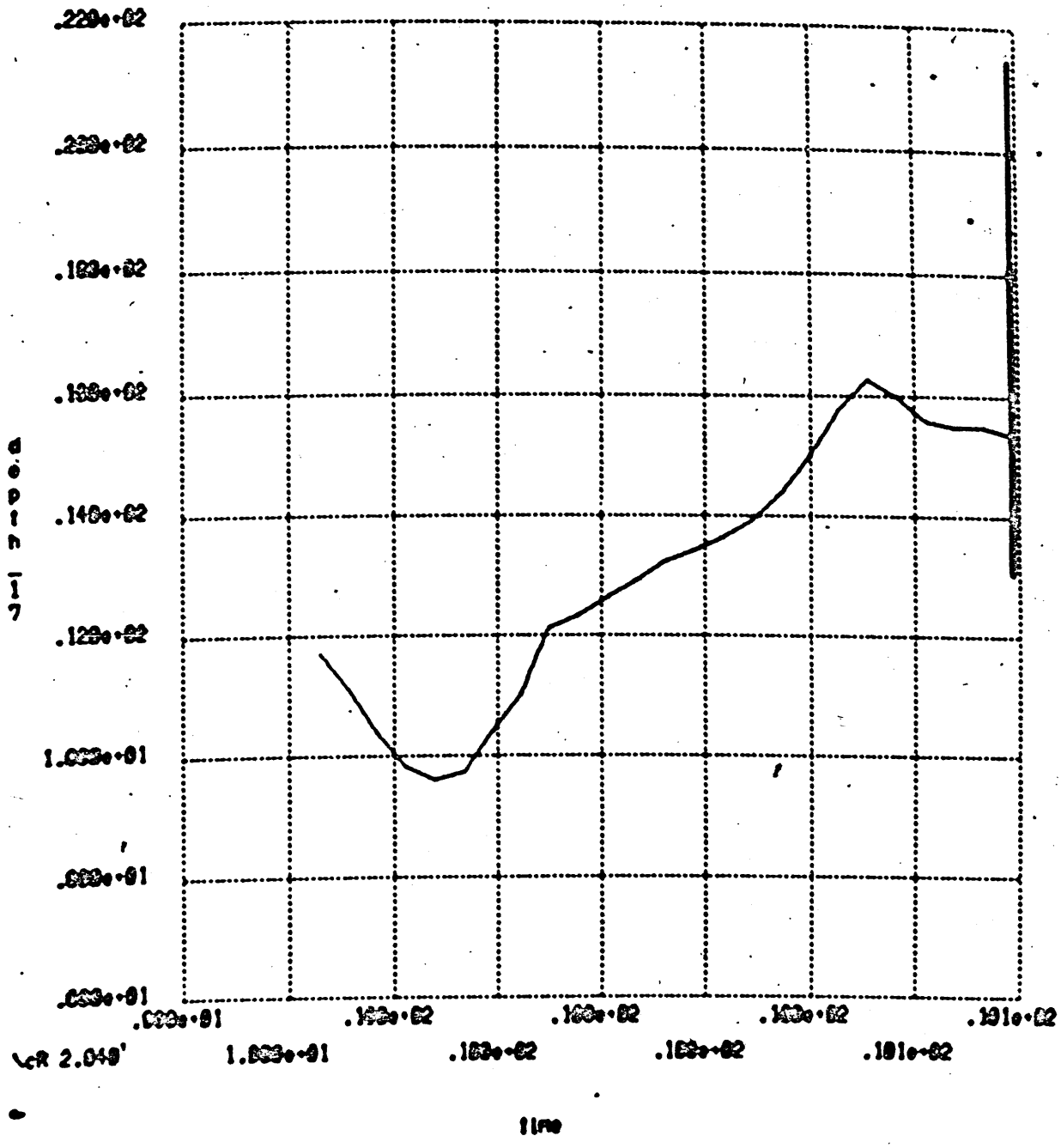


Figure V.6c

```
(10)  tspr:x
      T 20:50   3.669 $0.37
      Please type input line.
(11)  short$ end$
      T 20:51   1.127 $0.12
(12)  databank mnarray$
      T 20:51   0.335 $0.04
(13)  fetch depth_17$ end$
      T 20:51   0.992 $0.10
(14)  smpl 1, 100$
      T 20:51   0.249 $0.03
(15)  genr dif_depth_17 = depth_17 - depth_17(-1)$ end$
      entities vector
         1 to  100
      depth_17
      depth_17
      dif_dept
      T 20:52   1.032 $0.11
(16)  save dif_depth_17$ end$
      T 20:52   0.659 $0.07
(17)  stop$ end$
      R 9.191
(18)  subarray:x dif_depth_17 1 -1 1, new_dif_17
      R .503
(19)  sort:a new_dif_17 sort_dir#f_17
      R .687
(20)  percent:a sort_dif_17 50 per_50_isoth_17
      R .256
(21)  print_mnarray:a per_50_isoth_17
      dims =  1-      2
        0.000  0.000
      R .272
(22)  percent:a sort_dif_17 70 per_70_isoth_17
      R .205
```

Figure V.6d

(23) print_mnarray:a per_70_isoth_17

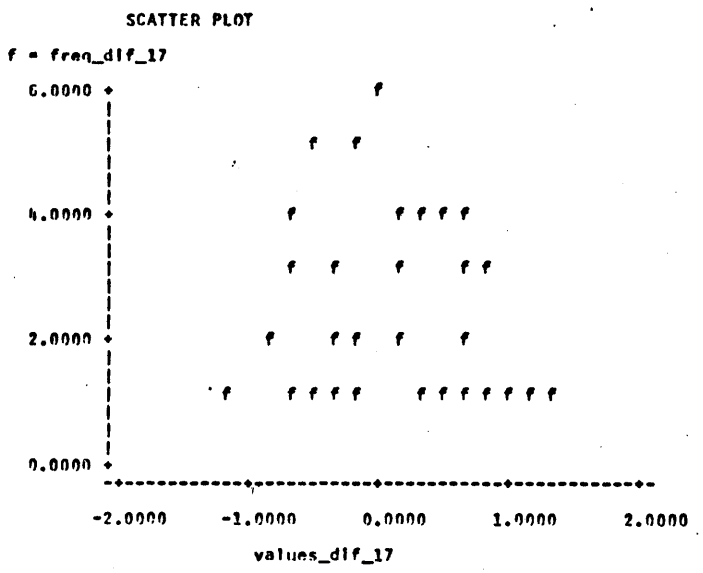
```
dims =  1-     2
  0.300  0.000

R .727
```

(24) freq_val:a sort_dif_17 values_dif_17 freq_dif_7#17

```
R .927
```

(25) plot:x values_dif_17 freq_dif_17




```
R 2.018
```

Figure V.6e

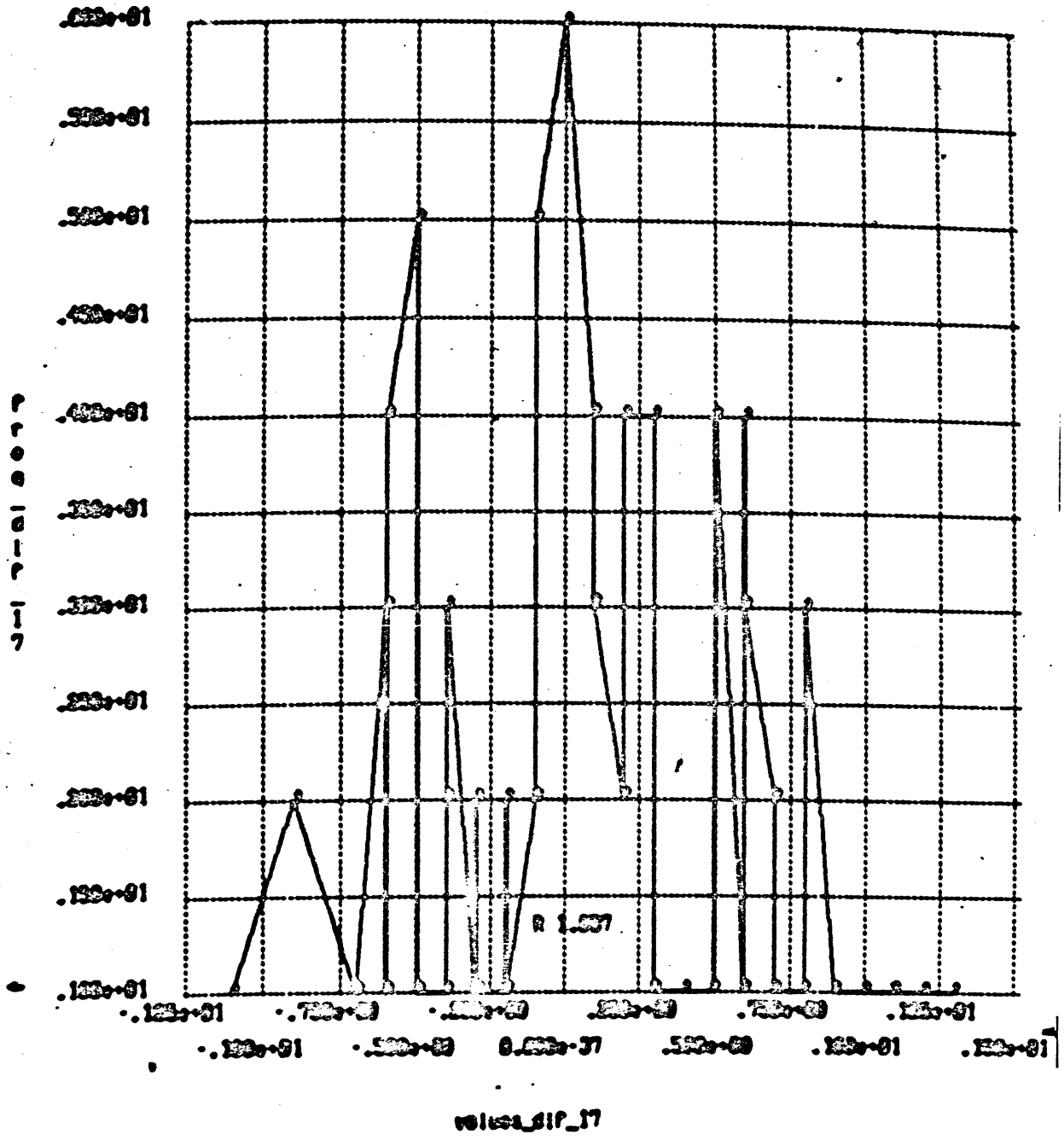


Figure V.6f

(27) tsprix

T 20:58 6.814 $0.76

(28) smpl 1, 100$

T 20:59 0.296 $0.04

(29) online fuu#11$ end$

entities vector
1 to 100

T 20:59 0.307 $0.03

(30) correl depth_17 depth_17 depth_17(-1) depth_17(-2) depth_17(-3) depth_17(-5) depth_17(-10)$ end$

correlation output
****

		mean	standard deviation
		mean	standard deviation
	depth_17	16.2711	2.2440
depth_17		16.2711	2.2440
	depth_17	16.2711	2.2440
depth_17		16.2711	2.2440
	depth_17	16.2200	2.2812
depth_17		16.2200	2.2812
	depth_17	16.1733	2.3217
depth_17		16.1733	2.3217
	depth_17	16.1211	2.3838
depth_17		16.1211	2.3838
	depth_17	16.0267	2.5330
depth_17		16.0267	2.5330
	depth_17	15.8467	2.7940
depth_17		15.8467	2.7940

correlation matrix

col row	depth_17(0) 1	depth_17(0) 2	depth_17(-1) 3	depth_17(-2) 4	depth_17(-3) 5	depth_17(-5) 6	depth_17(-10) 7
1	1.0000	1.0000	.97484	.91124	.82472	.63343	.17095
2	1.0000	1.0000	.97484	.91124	.82472	.63343	.17095
3	.97484	.97484	1.0000	.97600	.91514	.74616	.28532
4	.91124	.91124	.97600	1.0000	.97643	.84384	.39764
5	.82472	.82472	.91514	.97643	1.0000	.92565	.50873
6	.63343	.63343	.74616	.84384	.92565	1.0000	.70963
7	.17095	.17095	.28532	.39764	.50873	.70963	1.0000

T 21: 2 2.366 $0.28

Figure V.6g.

105.

(31) smpl 1, 100$ end$

entities vector
1 to 100

T 23:53 0.382 $0.05

(32) spectr depth_17 with plots with dt=10.0$ end$

analysis of the univariate spectrum of depth_17

frequency range		cosine transform	sine transform	complex transform		power spectrum
from	to	(real)	(imaginary)	amplitude	phase	
0.0000	0.0009	1340.019	-.113597	1340.019	-0.00	25269.0342
0.0019	0.0028	-182.542	113.3247	214.8586	148.17	390.501419
0.0037	0.0046	-135.050	78.43110	156.1728	149.85	164.631392
0.0056	0.0065	28.46171	139.3220	142.1905	78.45	108.613296
0.0074	0.0083	52.55189	73.12532	90.05006	54.30	48.4859147
0.0093	0.0102	72.79383	11.27087	73.66275	8.81	27.9692342
0.0111	0.0120	38.94346	-18.0046	42.90408	-24.81	10.9280013
0.0130	0.0139	-20.8123	-4.67521	21.33096	-167.34	3.44341779
0.0148	0.0157	-22.9513	35.93464	42.63870	122.57	10.2682173
0.0167	0.0176	8.066285	57.48638	58.04954	82.01	17.1913667
0.0185	0.0194	29.69817	36.80242	47.29058	51.10	11.4620774
0.0204	0.0213	38.24280	10.13256	39.56236	14.84	8.29023239
0.0222	0.0231	24.61054	-4.94546	25.10252	-11.36	3.30493161
0.0241	0.0250	.2002351	-11.4173	11.41904	-89.00	.821435153
0.0259	0.0269	-9.01290	11.54983	14.65029	127.97	1.21880075
0.0278	0.0287	-.547807	26.31069	26.31639	91.19	3.52731425
0.0296	0.0306	17.11054	28.54358	33.27922	59.06	5.78068435
0.0315	0.0324	32.29449	12.59528	34.63114	21.17	6.05643886
0.0333	0.0343	25.16267	-8.02297	26.41375	-17.68	3.58185366
0.0352	0.0361	9.107793	-12.7028	15.63054	-54.36	1.38668673
0.0370	0.0380	-3.83304	-1.99967	4.277960	-153.64	.178774500
0.0389	0.0398	-2.72129	14.72999	14.97925	100.47	1.16802052
0.0407	0.0417	11.76827	21.07351	24.13680	60.82	3.03910029
0.0426	0.0435	25.35352	11.60728	27.88447	24.60	3.98257625
0.0444	0.0454	26.94906	-5.14856	27.43647	-10.82	3.87693793
0.0463	0.0472	9.504310	-7.93118	12.37383	-39.84	1.53235544

Figure V.6h

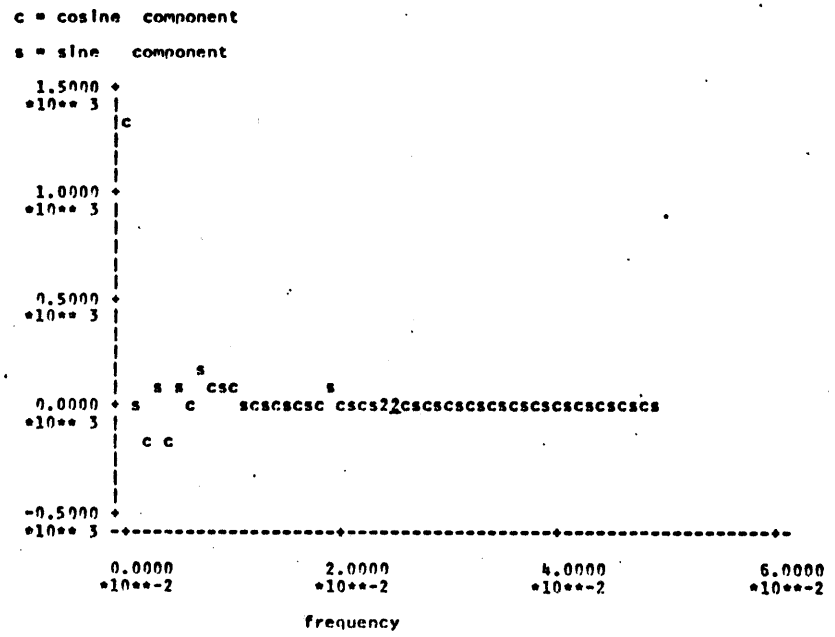

c = cosine component
s = sine component
1.5000 *10** 3
1.0000 *10** 3
0.5000 *10** 3
0.0000 *10** 3
-0.5000 *10** 3
0.0000 *10**-2
2.0000 *10**-2
4.0000 *10**-2
6.0000 *10**-2
Frequency

Figure V-6i

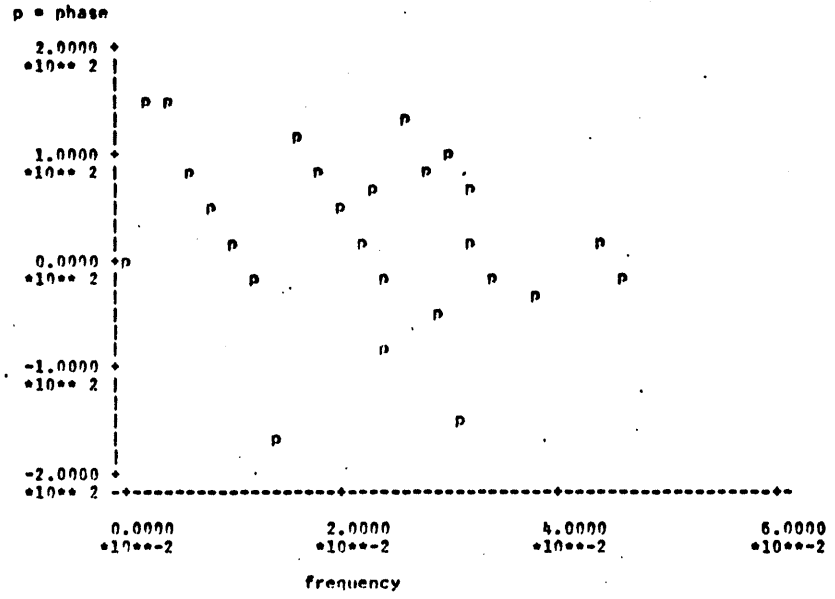

p = phase
2.0000
*10** 2
1.0000
*10** 2
0.0000
*10** 2
-1.0000
*10** 2
-2.0000
*10** 2
0.0000
*10**-2
2.0000
*10**-2
4.0000
*10**-2
6.0000
*10**-2
frequency

Figure V.6j

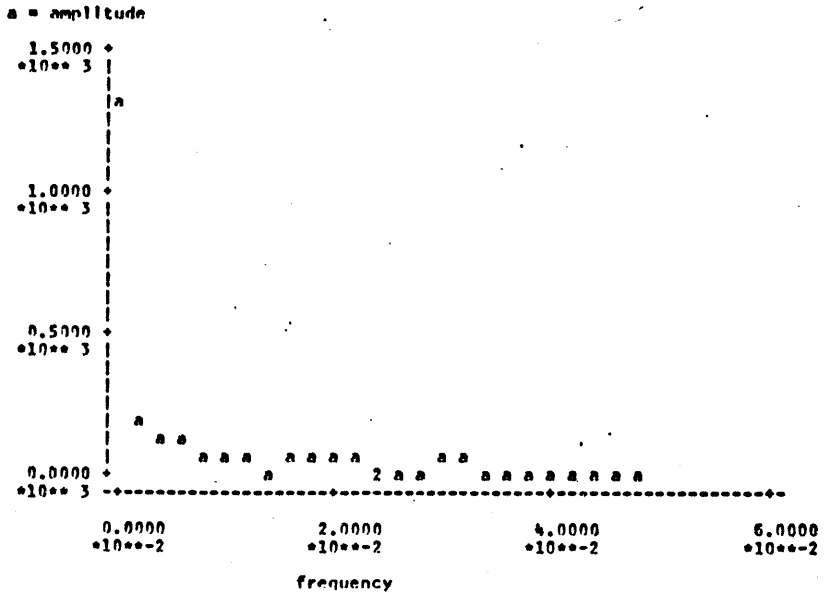

a = amplitude
1.5000
*10** 3
1.0000
*10** 3
0.5000
*10** 3
0.0000
*10** 3
0.0000
*10**-2
2.0000
*10**-2
4.0000
*10**-2
6.0000
*10**-2
frequency

Figure V.6k

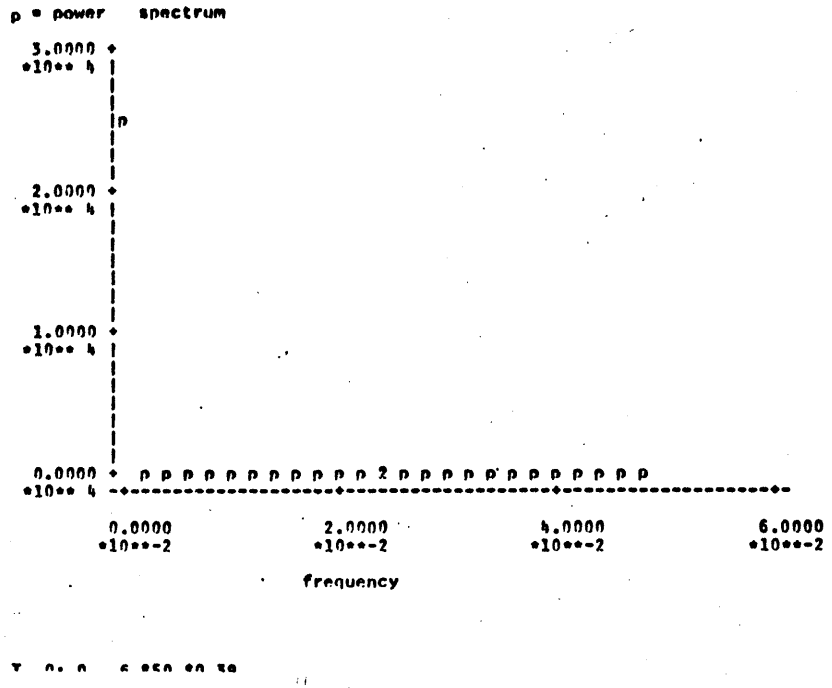

p = power spectrum
3.0000
•10•• 4
2.0000
•10•• 4
1.0000
•10•• 4
0.0000
•10•• 4
p p p p p p p p p p p 2 p p p p p p p p p p p
0.0000
•10••-2
2.0000
•10••-2
4.0000
•10••-2
6.0000
•10••-2
frequency

Figure V.61

(1)
```
exmul createdir >udd>CPInterim>Szasz>Ocean>Scientist>Analysis_173
R .222
```

(2)
```
read_char:a cmt_an
The analysis run today used the temperature raw data from cruise number 3545. Results
stored include two files. The first stores the differences in depth between two sucessive
entities of the 17 celsius isotherm. The other file stores the frequency distribution of
these differences in depth.
.
R .446
```

(3)
```
put_char:x cmt_an >udd>CPInterim>Szasz>Ocean>Scientist>Analysis_173>cmt_an
R .369
```

(4)
```
export:x dif_depth_17 1 10 5 >udd>CPInterim>Szasz>Ocean>Scientist>Analysis_173>dif_depth_17
export:x- no conversion - content space
R .358
```

(5)
```
export:x freq_dif_17 1 10 5 >udd>CPInterim>Szasz>Ocean>Scientist>Abalysis_173>freq_dif_17
R .629
```

(6)
```
delete:a cmt_an dif_depth_17 freq_dif_17 per_50_isoth_17 per_70_isoth_17
R .384
```

(7)
```
cdd >udd>cpc>P>x
R .413
```

(8)
```
janus:x
J
```

(9)
```
default rslt_n*gnl_inf
J
```

Figure V.7a

(10)
```
display all, for analysis_code = 173

                analysis_type
            completion_flag
basic_raw_code      institution_name        analysis_date
#     num_saved_files           scientist_name     analysis_code

2      3545    *    *    *    METEOR    SCIENTIST    761015    173
J
```

(11)
```
alter num_saved_files for entity 2 3
J
```

(12)
```
alter completion_flag for entity 2 1
J
```

(13)
```
display all, in type_an for all

#                  description     analysis_type

1    RAW DATA DISPLAY                        1
2    VERTICAL AND HORIZONTAL ISOLINES DISPLAY    2
3    FREQUENCY AND CUMULATIVE DISTRIBUTIONS      3
4    CORRELATION ANALYSIS                    4
5    SPECTRAL ANALYSIS                       5

6    RAW DATA AND ISOLINES DISPLAY                6
J
```

(14)
```
append to type_an analysis_type 7 / description "COMBINATION OF ANALYSIS TYPES 12# 2 3 4 5"
J
```

(15)
```
alter analysis_type for entity 2 7
J
```

(16)
```
display all, for analysis_code = 173

                analysis_type
            completion_flag
basic_raw_code      institution_name        analysis_date
#     num_saved_files           scientist_name     analysis_code

2      3545    3    1    7    METEOR    SCIENTIST    761015    173
J
```

Figure V.7b

```
(17)  leave
      R .934

(18)  delete:a time latitude longitude temperature
      R .490

(19)  exit
      r 2346 3.290 92.284 753

(20)  change_wdir >udd>CPInterim>Szasz>Ocean>Scientist>Analysis_173
      r 2346 .125 .650 28

(21)  listnames

      Segments= 3, Records= 1.

      freq_dif_17
      dif_depth_17
      cmt_an

      r 2346 .181 .462 2
```

Figure V.7c

Chapter 6

CONCLUSIONS AND RECOMMENDATIONS

The complete design, development and implementation of a data base management system as described in Chapter III is a long and difficult project. Besides the need, in such a project, of the integration of several different subsystems that satisfy user's requirements, there is a large and constant need of an intelligent and helpful feedback from the users' part. In the past, there has been a general trend to develop information systems with sophisticated objectives, with either small or non-informative feedback from users. Many times whole information systems were designed based on users specifications established at the beginning of the project. Evidently, by the time the information system was ready to work, usually the users' specifications have changed or the users claimed that it was not what they had in mind. They had set up a complete list of specifications at the beginning, about which they were not sure since it is hard to establish detailed specifications on complex projects, without seeing a preliminary version of it.

The other way around is to design a part of the system, trying not to reduce its generality, that can be implemented

in a short period of time and that enables an efficient feedback from the users.

In order to be coherent with the whole idea of an integrated and flexible data base, it is important to choose a consistent environment for programming and design additions that will occur in the future. A major reason for having chosen the consistent system is that it provides a consistent and very general base for these additions. As an example, we might mention that if a user wants to add a command which performs computations not yet provided in the C.S., he may do so by writing a program in a high level language and cataloging it into the C.S. Thus, the consistent system provides us with a very powerful base. It has in it several tools which are fundamental for us, such as a general file system, graphics software, data management and time series processing.

Evidently, the C.S. does not have and was not meant for having oceanographic oriented software. This is a work that has to be done gradually using constant feedback from scientists. As it was shown in Chapter 3, the commands "accept my cruises", "add my cruises", "subtract my cruises", etc. are not available in the C.S. They would have to be implemented in terms of programs and added to the C.S. However, there are several ways around these commands: the user would have to do more typing and would need a greater

and deeper knowledge of the system. To further illustrate this reasoning, the system, after being used and implemented, could be used by people with almost no knowledge of computers or data base systems. The system itself would teach the user and check for the correct use of commands.

Further, we would like to stress the point that the user himself must act as a manager of his resources. This means that the system cannot make decisions, regarding several trade-offs, that are inherent to the system, such as saving results of an analysis versus having to reprocess the data in the future. Once the data base system is working and the scientist wants to use it, he needs to have an overall picture of what and how efficiently his system can do. This once again points out the importance of an on-line environment, where machine and man combined can lead to an efficient use of a system.

While our desire in Chapter 5 has been to "simulate" as closely as possible what was described in Chapter 3, in some instances, we were not able to pursue this objective--the reason being that most of the software available in the Consistent System is still available only in a prototype form. This is particularly true for TSP, where the prototype does not permit the user, yet, to save results in a convenient form. Fortunately, these seem to be minor problems, given that the Cambridge Project staff, responsible

for the Consistent System, will release new versions of both JANUS and TSP in the near future.

A word should be said about graphical display devices. Supposedly, if and when such a system, as proposed in this report, would come to operate, most of the work would be carried out using CRT terminals. While our initial objective was meant to include examples of graphical displays, this was only partly possible given that at the present time there is not a single CRT device, at Cambridge Project, connected with the Multics System, that is attached to a hard copy device. Instead, we have taken POLAROID pictures from the graphical displays obtained at a CRT device in the department of Architecture.

In this report we have attempted to design and implement a preliminary and first version of an information management system for a large oceanographic data base. While in this early stage, the data was supposed to reside in direct-access devices, this will present problems as the data base size increases. Possible alternatives include the establishment of a tape library, just as described in the Seismic Project, Chapter 2.

With regard to performance, we must remember that this is a subsystem of the Multics System. Therefore the response time and efficiency of our system are directly related to those of the host system: Multics. While in general Multics

presents a reasonably good response time for on-line users, it seems that the Consistent System can present slow response time during hours of peak system utilization.

The Consistent System was designed and developed for a wide range of applications, therefore reducing efficiency and flexibility on a particular set of applications. Future developments and extensions to this report should be directed in the area of providing more flexible and powerful interaction with the databank directory and providing more powerful and a larger number of tools for scientists analyzing oceanographic data.

In the present version, the scientist needs to do much more typing than is convenient in order to locate the section of data he is interested in. Possible improvements in the area of interaction would be to implement the commands described in Chapter 3: accept, add and subtract my cruises.

On the other hand, several commands for efficient and flexible displaying and time-series processing of data are not available within the C.S. Even the ones that are available, are far too general to give a desirable output for the scientist. In this area, there is a need to develop problem oriented software.

Finally, as seen before, the data base size in itself is a limitation for on-line activities. Even if the whole data base resides on direct access devices, there is a

question about the size of the data to be used during an analysis. Even though, TSP or the CS in general, is supposed to handle normally files with 50,000 entities, the response time with commands that involve such quantities of data may prove to be unpractical. A possible solution for this problem would be to mix background jobs with on-line interactive work.

MASTER RECORD #1 - GENERAL INFORMATION

Table III.1

FIELD	BYTE LOCATION(S)		REMARKS
RECORD TYPE	1-2	(2)	'M1'
NEXT RECORD TYPE	3-4	(2)	'M2'
DATE OF CRUISE	5-12	(8)	MM:DD:YYYY
LATTITUDE (to nearest minute)	13-17	(5)	DDD:MM A priora Area of Study (This info
LONGITUDE (to nearest minute)	18-23	(5)	DDD:MM will come from scientist, not raydist)
CRUISE #	24-31	(8)	'USE DATE'
SHIP NAME	32-46	(15)	'R.V. GOODLUCK'
INSTITUTION	47-62	(16)	'MIT DEPT OF MET'
ATTRIBUTE #1	63-66	(4)	'1' (towed therm chain) Attributes may be: physical variable types, sensor classification and in general, anything that is defined in the attribute table.
ATTRIBUTE #2	67-70	(4)	'4' (BT cast)
ATTRIBUTE #3	71-74	(4)	'11' (temperature)
ATTRIBUTE #4	75-78	(4)	'12' (pressure)
UNUSED	79-256	(178)	THIS RECORD CAN CONTAIN FIELDS AS SHOWN FOR UP TO 48 ATTRIBUTES.

MASTER RECORD #2 - ATTRIBUTE TABLE

Table III.2

FIELD	BYTE LOCATION(S)		REMARKS
RECORD TYPE	1-2	(2)	'M2'
NEXT RECORD TYPE	3-4	(2)	'M2'
ATTRIBUTE #1	5-20	(16)	'TOWED THERM CHAIN'
ATTRIBUTE #2	21-36	(16)	'HYDROGLIDER PORP'
ATTRIBUTE #3	37-52	(16)	'HYDROGLIDER FIXED'
#4	53-68	(16)	'BT CAST'
#5	69-84	(16)	'CTD CAST'
#6	85-100	(16)	'VELOCIMETER'
#7	101-116	(16)	'AUTOANALYSER'
#8	117-132	(16)	'ANEMOMETERS'
#9	133-148	(16)	'#40 PLRNKTN NET'
#10	149-164	(16)	'BOTTOM SAMPLES'
#11	165-180	(16)	'TEMPERATURE'
#12	181-196	(16)	'PRESSURE'
#13	197-212	(16)	'CONDUCTIVITY'
#14	213-228	(16)	'SALINITY'
#15	229-244	(16)	'PHOSPHATE'
UNUSED	245-256	(12)	

MASTER RECORD #3 - CONTINUATION OF ATTRIBUTE TABLE

Table III.3

FIELD	BYTE LOCATION(S)		REMARKS
RECORD TYPE	1-2	(2)	'M2'
NEXT RECORD TYPE	3-4	(2)	'M3'
ATTRIBUTE #16	5-20	(16)	'NITRITE'
#17	21-36	(16)	'NITRITE'
#18	37-52	(16)	'VELOCITY AIR'
#19	53-68	(16)	'VELOCITY WATER'
#20	69-84	(16)	'E. COLI'
#21	85-100	(16)	'AMPHORA OVALIS'
UNUSED			EXPANSION OF ATTRIBUTE TABLE TO ARBITRARY # OF RECORDS IS POSSIBLE, AS THE NEED ARISES.

MASTER RECORD #4 - SYNCHRONOUS INSTRUMENTATION GEOMETRY

Table III.4

FIELD	BYTE LOCATION(S)		REMARKS	
RECORD TYPE	1-2	(2)	'M3'	
NEXT RECORD TYPE	3-4	(2)	'M4'	
NSENS	5-6	(2)	'9'	Number of synchronous sensors
Δt	7-10	(4)	'100'	Time difference between samples from a sensor: (SS.SS - Seconds to the nearest hundredth, so '100' = 1 second)
NSAMPS	11-14	(4)	'6'	Number of samples per sensor in an SD record
LENGTH OF CABLE OVER-BOARD (METERS)	15-18	(4)	'50'	As measured between two well-defined points, e.g. from the shackle on the V-fin to a pre-measured and marked spot on the cable, which is at the surface.
SD SENSOR #1				
SENSOR	19-22	(4)	'P1'	Alphameric description
BYTE LENGTH (of sample in SD record)	23-24	(2)	'4'	Allows for varying resolutions for different types of sensors.
PRECISION	25-26	(2)	'1'	# of bytes to right of (implied) decimal point
PHYSICAL LOCATION	27-30	(4)	'1'	Distance from end of cable (meters)

MASTER RECORD #4 (CONT.) - SYNCHRONOUS INSTRUMENTATION GEOMETRY(CONT.)

Table III.5

	FIELD	BYTE LOCATION(S)		REMARKS	
SD SENSOR #2	SENSOR	31-34	(4)	'T1'	Alphameric description
	BYTE LENGTH (of sample in SD record)	35-36	(2)	'4'	
	PRECISION	37-38	(2)	'1'	# of bytes to right of (implied) decimal point
	PHYSICAL LOCATION	39-42	(4)	'5'	Distance from end of cable (meters)
SD SENSOR #3	SENSOR	43-46	(4)	'T2'	
	BYTE LENGTH	47-48	(2)	'4'	
	PRECISION	49-50	(2)	'1'	
	LOCATION	51-54	(4)	'10'	
	FIELDS FOR SENSOR #4	55-66	(12)		
	FIELDS FOR SENSOR #5	67-78	(12)		
	FIELDS FOR SENSOR #6	79-90	(12)		
	FIELDS FOR SENSOR #7	91-102	(12)		
	FIELDS FOR SENSOR #8	103-114	(12)		
	FIELDS FOR SENSOR #9	115-126	(12)	This record can contain fields as shown for up to 19 synchronous sensors. For more than 19 sensors, another 'M3' type record would follow this one in the file, and would be treated as a continuum.	
	UNUSED	127-256	(130)		

MASTER RECORDS 5-13 - SYNCHRONOUS INSTRUMENTATION CALIBRATION
(ONLY THE FIRST ONE FOR SYNCHRONOUS SENSOR #1 IS SHOWN)

Table III.6

FIELD		BYTE LOCATION(S)		REMARKS
RECORD TYPE		1-2	(2)	'M4'
NEXT RECORD TYPE		3-4	(2)	'M5'
SENSOR		5-8	(4)	'P1' Same as sensor #1 field on M3 record.
PHYSICAL VARIABLE		9-12	(4)	'12' (Pressure) Uses attribute code from M2 records
PHYSICAL VARIABLE UNITS		13-20	(18)	'DECIBARS' Alphameric description
DIGITIZED SIGNAL		31-40	(10)	'VOLTS' Alphameric description
LSB OF DIGITIZED SIGNAL		41-50	(10)	'.005' e.g. the least significant bit of the digital output word from the A/D on this sensor is worth .005 volts.
DATE OF CALIBRATION		51-58	(8)	MM:DD:YYYY
NSEGS		59-62	(4)	'3' Number of linear segments comprising calibration curve. NSEGS=0→the next field contains a proportionality constant
	V1	63-72	(10)	'5.0000'
SEGMENT 1	P1	73-82	(10)	'0.0000'
	V2	83-92	(10)	'10.0500'
SEGMENT 2	P2	93-102	(10)	'22.7850'
	V3	103-112	(10)	'15.0000'
SEGMENT 3	P3	113-122	(10)	'34.8260'
	V4	123-132	(10)	'20.0000' Implied four decimal place accuracy
	P4	133-142	(10)	'80.0000'
UNUSED				

MASTER RECORD #14 - ASYNCHRONOUS INSTRUMENTATION GEOMETRY

(THIS RECORD DESCRIBES A BT)

Table III.7

FIELD	BYTE LOCATION(S)		REMARKS	
RECORD TYPE	1-2	(2)	'M5'	
NEXT RECORD TYPE	3-4	(2)	'M6'	
NSENS	5-6	(2)	'2'	Number of asynchronous sensors (BT → 2: Pressure, Temperature)
SENSOR	7-10	(4)	'P1'	Alphameric description
Δt	11-16	(6)	'100'	Time difference between samples from the sensor, SSSS.SS, so '100' = 1 second.
NSAMPS	17-20	(4)	'27'	Number of samples from this sensor in an AID record.
BYTE LENGTH (of sample in AID record)	21-22	(2)	'4'	Number of bytes to right
PRECISION	23-24	(2)	'1'	Number of bytes to right of (implied) decimal point
SENSOR	25-28	(4)	'T1'	
Δt	29-34	(6)	'100'	
NSAMPS	35-38	(4)	'27'	
BYTE LENGTH	39-40	(2)	'4'	
PRECISION	41-42	(2)	'1'	
UNUSED	13-256	(214)		

MASTER RECORDS #15-16 ASYNCHRONOUS INSTRUMENTATION CALIBRATION
(ONLY THE FIRST ONE FOR ASYNCHRONOUS SENSOR #1,
THE PRESSURE XDUCER ON THE BT, IS SHOWN)

Table III.8

FIELD		BYTE LOCATION(S)		REMARKS
RECORD TYPE		1-2	(2)	'M6'
NEXT RECORD TYPE		3-4	(2)	'M6'
SENSOR		5-8	(4)	'P1' Same as sensor #1 field on the M5 record.
PHYSICAL VARIABLE		9-12	(4)	'12' (Pressure) Users attribute code from M2 records.
PHYSICAL VARIABLE UNITS		13-30	(18)	'DECIBARS' Alphameric description.
DIGITIZED SIGNAL		31-40	(10)	'VOLTS' Alphameric description.
LSB OF DIGITIZED SIGNAL		41-50	(10)	'.005' e.g. The least significant bit of the digital output word from the A/D on this sensor is worth .005 volts.
DATE OF CALIBRATION		51-58	(8)	MM:DD:YYYY
NSEGS		59-62	(4)	'3' Number of linear segments comprising calibration curve. NSEGS = 0 → the next field contains proportionality constant.
	V1	63-72	(10)	'5.0000'
SEGMENT 1	P1	73-82	(10)	'0.0000'
	V2	83-92	(10)	'10.0500'
SEGMENT 2	P2	93-102	(10)	'22.7850' Implied four decimal place accuracy
	V3	103-112	(10)	'15.0000'
SEGMENT 3	P3	113-122	(10)	'34.8260'
	V4	123-132	(10)	'20.0000'
	P4	133-142	(10)	'80.0000'
UNUSED		143-256	(114)	

MASTER RECORD #17 - SYSTEM FIXED DESCRIPTION INFORMATION

Table III.9

FIELD	BYTE LOCATION(S)		REMARKS
RECORD TYPE	1-2	(2)	
NEXT RECORD TYPE	3-4	(2)	
TO BE USED BY DRAPER TO DESCRIBE LAYOUT OF AID SYSTEM STATUS RECORDS THAT APPEAR IN DATA STREAM	5-256	(252)	

MASTER RECORD #18 - MARKER DEFINITION

(ONLY ONE IS SHOWN, BUT THERE CAN BE n)

Table III.10

FIELD	BYTE LOCATION(S)		REMARKS
RECORD TYPE	1-2	(2)	'M8'
NEXT RECORD TYPE	3-4	(2)	
MARKER	5-6	(2)	'$'
DEFINITION	7-256	(250)	'PASSING THROUGH THERMAL FRONT'

SD RECORD FORMAT - ONE CHARACTER PER BYTE

Table III.11

FIELD	BYTE LOCATION(S)		REMARKS
RECORD TYPE	1-2	(2)	'SD'
NEXT RECORD TYPE	3-4	(2)	Ø IF THIS IS LAST RECORD IN FILE
FRAGMENTATION FLAG	5-6	(2)	NOT APPLICABLE TO SD RECORDS
TIME of 1st sample	7-12	(6)	FORMAT IS: HH:MM:SS, GMT.
LATITUDE in	13-22	(10)	FORMAT IS: ±DD:MM:SS:TH
LONGITUDE record	23-32	(10)	FORMAT IS: ±DDD:MM:SS:TH + = North, West - = South, East Location to .01 seconds
UNUSED	33-40	(8)	SPARE LOCATIONS FOR FUTURE EXPANSION

SD RECORD FORMAT(cont.)

Table III.12

FIELD	BYTE LOCATION(S)		REMARKS
PRESSURE 1 (t1)	41-44	(4)	FORMAT IS: PPP.P (ACCURACY TO TENTHS OF DECIBARS)
THERM 1 (t1)	45-48	(4)	FORMAT IS: TTT.T (ACCURACY TO TENTHS OF A DEGREE C)
THERM 2 (t1)	49-52	(4)	
THERM 3 (t1)	53-56	(4)	
THERM 4 (t1)	67-60	(4)	
THERM 5 (t1)	61-64	(4)	
THERM 6 (t1)	65-68	(4)	
THERM 7 (t1)	69-72	(4)	
THERM 8 (t1)	73-76	(4)	
PRESSURE 1 (t2)	77-80	(4)	
.	.		REPEAT SAMPLING OF SENSORS
.	.		EVERY ΔT SECONDS
.	.		
THERM 8 (t2)	99-102	(4)	
SAMPLES AT t3	103-138	(36)	

SD RECORD FORMAT (CONT.)

Table III.13

FIELD	BYTE LOCATION(S)		REMARKS
SAMPLES AT t4	139-174	(36)	
SAMPLES AT t5	175-210	(36)	
SAMPLES AT t6	211-246	(36)	
UNUSED	247-256	(10)	

REFERENCES

Berteaux, H.O. and N.K. Chhabra, 1973 "Computer Programs for the Static Analysis of Single Point Moored Surface Buoy Systems", Woods Hole Oceanographic Institution, Unpublished Manuscript.

Brode, John, 1972 "Time Series Processor Users Manual" Reference Guide.

Cambridge Project, 1973 "Handbook of Programs and Data" Reference Guide.

Dorman, Craig E. and Erik Mollo-Christensen, 1973 "Observation of the Structure on Moving Gust Patterns over a Water Surface", M.I.T., Department of Meteorology.

Karpen, Joseph, 1973 "Dissolved Nutrient-Se-water Density Correlations and the Circulation in Boston Harbor and Vicinity" S.M. Thesis, M.I.T.

Klensin, John, 1973 "The interim plotting programs" Cambridge Project report.

Lange, Caroline, 1973 "Beginner's Manual for the Consistent System", Cambridge Project report.

Manohar-Maharaj, Veshpati, 1973 "Spring runn off into Massachusetts Bay, 1973" S.M. Thesis, M.I.T., Cambridge, Massachusetts.

Martin, James, 1972 "Introduction to Teleprocessing", Prentice-Hall.

M.I.T. and Project Mac, 1972 "The Multics Programmers' Manual: Graphics Users' Supplement", Reference Guide.

M.I.T. and Project Mac, 1973 "The Multiplexed Information and Computing Service: Programmers' Manual", Reference Guide.

Mollo-Christensen, Erik and Pior Koziol, 1972 "Underway and Scanning Measurement of Air-Sea Surface Layer Properties", M.I.T., Department of Meteorology.

Nolan, R.L., 1973 "Computer data bases: the future is now" Harvard Business Review".

Shuford, Dorothy and Jeffrey Stamen, 1973 "User's Manual for the Janus Prototype System" Cambridge Project draft.

Stamen, Jeffrey P. and R.M. Wallace, 1973 "Janus: a Data Management Analysis System for the behavioral sciences" Cambridge Project report.

Theriault, Kenneth B., 1973 "Some significant papers in Marine Seismic Data Processing", WHOI, Unpublished Manuscript.

Tollows, Constantine D., 1973 "The ACODAC data processing system", Woods Hole Oceanographic Institution", Unpublished Manuscript.

U.S. Navy Electronics Laboratory, 1966 "Vertical and Horizontal Thermal Structures in the Sea" Research and Development Report.