

DTIC FILE COPY

2

AD-A233 544

Texas A&M University

Geochemical and Environmental Research

ANALYSIS OF NEARSHORE BATHYMETRY AND OPTICAL DATA FROM TUPS

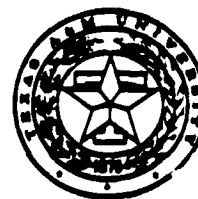
Norman L. Guinasso, Jr. and Denis A. Wiesenburg

**Geochemical and Environmental Research Group
Department of Oceanography
Texas A&M University
Ten South Graham Road
College Station, TX 77840**

**DTIC
SELECTED
MAR 22 1991
S D D**

December 1988

TECHNICAL REPORT # 88-174



REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. Agency Use Only (Leave blank).		2. Report Date. 1988		3. Report Type and Dates Covered. Contract	
4. Title and Subtitle. Analysis of Nearshore Bathymetry and Optical Data from TUPS				5. Funding Numbers. Program Element No 63794N Project No. 01987 Task No. 300 Accession No DN258075	
6. Author(s). Norman L. Guinasso, Jr. and Denis A. Wiesenburg					
7. Performing Organization Name(s) and Address(es). Texas A&M University Technical Support Services Department of Oceanography, College Station, TX				8. Performing Organization Report Number. TR-88-174	
9. Sponsoring/Monitoring Agency Name(s) and Address(es). Naval Oceanographic and Atmospheric Research Laboratory Ocean Science Directorate Stennis Space Center, MS 39529-5004				10. Sponsoring/Monitoring Agency Report Number. CR 020:91	
11. Supplementary Notes. Contract N00014-88-K-6003					
12a. Distribution/Availability Statement. Approved for public release; distribution is unlimited.				12b. Distribution Code.	
13. Abstract (Maximum 200 words). During June 1988, an experiment using NORDA's towed underwater pumping system (TUPS) was conducted in shallow water off Panama City, Florida to gather data in support of the Navy's Airborne Bathymetric Survey System (ABS). Continuous measurements were made of upwelling irradiance, fluorescence, transmissometry, water depth, temperature, and salinity using sensors in TUPS. Ambient light measurements and navigation data were recorded from instruments aboard the research vessel. This report describes the experiment, the cruise itinerary, and the methods used to carry out the measurements. Data are presented in graphical form, along with a description of the data reduction protocols and some preliminary results. Listings of computer programs used for processing the data are included. A description is given of the contents of a magnetic tape (provided to NORDA) containing all programs, raw and processed data. Preliminary analysis of the data indicate that the quality of the light sensor and depth data are good. A significant inverse relationship between depth and upwelled irradiance was observed under some conditions. The relationship holds well in areas of bright sand, but is more complicated in areas of variable bottom type and water clarity. The broad suits of light and water quality sensors aboard TUPS make it an ideal system for providing the data to resolve such problems.					
14. Subject Terms. (U) Bathymetry; (U) Salinity				15. Number of Pages. 133	
				16. Price Code.	
17. Security Classification of Report. Unclassified	18. Security Classification of This Page. Unclassified	19. Security Classification of Abstract. Unclassified	20. Limitation of Abstract. SAR		

Analysis of Nearshore Bathymetry and Optical
Data from TUPS

by

Norman L. Guinasso, Jr.

and

Denis A. Wiesenburg

Geochemical and Environmental Research Group
Department of Oceanography
Texas A&M University
Ten South Graham Road
College Station, Texas 77840

(409) 690-0095

Final Report

Prepared for

Mr. Robert A. Arnone, Scientific Officer
Mapping, Charting and Geodesy Division
Naval Ocean Research and Development Activity
Stennis Space Center, Mississippi 39529.

Accession For	
NTIS	CRA21
DTIC	7AB
Unannounced Justification	
By	
Distribution	
Availability	
Dist	Availability
A-1	

This work was supported by
the Mapping, Charting and Geodesy Division of NORDA
through
ONR Contract No. N00014-88-K-6003
funded by
Oceanographer of the Navy (CNO OP-096)
and
Defense Mapping Agency, HTC

GERG TECHNICAL REPORT 88-174

Abstract

During June 1988, an experiment using NORDA's towed underwater pumping system (TUPS) was conducted in shallow water off Panama City, Florida to gather data in support of the Navy's Airborne Bathymetric Survey System (ABS). Continuous measurements were made of upwelling irradiance, fluorescence, transmissometry, water depth, temperature, and salinity using sensors in TUPS. Ambient light measurements and navigation data were recorded from instruments aboard the research vessel. This report describes the experiment, the cruise itinerary, and the methods used to carry out the measurements. Data are presented in graphical form, along with a description of the data reduction protocols and some preliminary results. Listings of computer programs used for processing the data are included. A description is given of the contents of a magnetic tape (provided to NORDA) containing all programs, raw and processed data.

Preliminary analysis of the data indicate that the quality of the light sensor and depth data are good. A significant inverse relationship between depth and upwelled irradiance was observed under some conditions. The relationship holds well in areas of bright sand, but is more complicated in areas of variable bottom type and water clarity. The broad suite of light and water quality sensors aboard TUPS make it an ideal system for providing the data to resolve such problems.

TABLE OF CONTENTS

	PAGE
TABLE OF CONTENTS.....	ii
LIST OF FIGURES.....	iii
LIST OF TABLES.....	iii
1.0 INTRODUCTION.....	1
1.1 Rationale for the Study	1
1.2 Reasons for Using TUPS	3
2.0 DATA COLLECTION.....	3
2.1 Study Area	3
2.2 Cruise Itinerary	5
2.3 TUPS Sensor Configuration	9
2.4 TESS Sensor Configuration	10
2.5 TUPS At-Sea Data Collection	12
2.6 TESS At-Sea Data Collection	13
2.7 Data Collection Problems	14
3.0 DATA ANALYSIS.....	14
3.1 Inspection of Data Collected	14
3.2 Problems with Collected Data	15
3.3 Recovery of Lost Data	17
3.4 Data Analysis Protocol	18
3.5 Description of Computer Programs	22
4.0 DATA PRESENTATION.....	26
4.1 Time Series Plots	26
4.2 Contour Maps for Each Day	30
4.3 SAS Analysis of Total Data Set	30
4.4 Description of Deliverables	33
5.0 CONCLUSIONS.....	37
6.0 REFERENCES.....	38
7.0 ACKNOWLEDGEMENTS.....	38
APPENDIX A - COMPUTER PROGRAM LISTINGS.....	39
APPENDIX B - CONTOUR MAPS.....	84
APPENDIX C - VAX 9-TRACK TAPE LISTING.....	124

LIST OF FIGURES

Figure 1.	TUPS instrumentation layout as originally configured by NORDA.	 4
Figure 2.	Chart of the study area south of Panama City, FL.	 6
Figure 3.	Flow diagram of data processing protocol showing FORTRAN programs used and data file names.	 20
Figure 4.	Flow diagram of data processing protocol showing SAS programs used and data file names.	... 21
Figure 5.	Time series plot of combined data from 19 June 1988	 27
Figure 6.	Time series plot of combined data from 20 June 1988	 28
Figure 7.	Time series plot of combined data from 21 June 1988	 29
Figure 8.	Plot of inverse of normalized upwelling irradiance vs. depth on 19 June 1988.	 34
Figure 9.	Plot of inverse of normalized upwelling irradiance vs. depth on 20 June 1988.	 35
Figure 10.	Plot of inverse of normalized upwelling irradiance vs. depth on 21 June 1988.	 36

LIST OF TABLES

Table 1.	TUPS Sensor Configuration, June 1988	 11
Table 2.	TESS Sensor Configuration, June 1988.	 12
Table 3.	Position in data record, descriptive name, SAS name and units for data presented in this report and on the data tape supplied.	.. 23
Table 4.	Pearson correlation matrix of TUPS and TESS measurements from the Panama City Experiment.	 31

1.0 INTRODUCTION

1.1 Rationale for the Study

The Naval Ocean Research and Development Activity (NORDA) has been developing and testing an Airborne Bathymetric Survey System (ABS) to measure water depth in coastal areas. The ABS is being developed by NORDA for the Defense Mapping Agency (DMA) and the Oceanographer of the Navy (CNO OP-096) under the NORDA Coastal Hydrographic Techniques Program. The ABS uses a laser system in conjunction with a multispectral scanner which measures upwelled irradiance at several wavelengths. The laser depth measurements and the multispectral scanner data are merged together to provide an algorithm which allows computation of depth from the multi-spectral scanner data alone.

The goal of the work reported here was to gather environmental data for making ground-truth measurement of the ABS. In addition to the ground truth support, this work was undertaken to help improve the present algorithms used by the ABS, by gaining a better understanding of the ocean's optical-environmental parameters. To accomplish these goals, NORDA equipped its towed underwater pumping system (TUPS) with a depth sensor (0-100 meter capable with 2 cm resolution) and upwelling irradiance sensors with fixed wavelength filters of 465 nm, 507 nm and 532 nm. This equipment was installed in TUPS during the spring of 1988.

It was decided to perform a field experiment using TUPS to provide ground-truth coverage for ABS overflights to be conducted during June 1988 off Panama City, Florida in the Gulf of Mexico.

For this experiment, the Geochemical and Environmental Research Group (GERG) of the Department of Oceanography at Texas A&M University was contracted (ONR Contract No. N00014-88-K-6003) to provide the ship and personnel required to tow the NORDA TUPS off Panama City, Florida. This effort included chartering a suitable vessel, installing the TUPS hardware and computers aboard the vessel, operating the system at sea during the cruise, collecting the TUPS data along with navigation and appropriate atmospheric information while at sea, and processing the data upon return to the laboratory.

This report describes the cruise data collected with TUPS off Panama City, Florida, on June 19, 20, 21, 1988. The data analysis procedures are described in detail and copies of FORTRAN computer programs use to process the data from collection to presentation are provided. A complete set of all data and programs is available on a VAX-compatible 9-track magnetic tape which was supplied to the NORDA Scientific Officer, Mr. Robert A. Arnone, along with this report. The complete data will be processed further by NORDA scientists to better understand the relationships between light, water clarity and depth in coastal regions.

1.2 Reasons for Using TUPS

To provide the ground-truth data for the Airborne Bathymetric Survey System (ABS), NORDA chose to use its towed underwater pumping system (TUPS). TUPS was designed and constructed at NORDA for a project to study "Chemical Dynamics in Ocean Frontal Areas". To meet the needs of that project and to provide the capability of performing other surface-water surveys, the TUPS tow vehicle was designed (Rein et al., 1985) to carry a large suite of oceanographic instruments. It was also designed to be easy to modify in order to allow other instruments to be quickly installed and interfaced to its on-board computer. The instrument layout of TUPS, as originally configured, is shown in Figure 1. The tow vehicle configured in this manner had been used successfully to study variability in water optical properties and environmental parameters in the western Mediterranean Sea (Arnone and Wiesenburg, 1988). Its successful use in previous studies and its easy adaptability made it an ideal instrument for studying the optical-depth relationships in coastal waters.

2.0 DATA COLLECTION

2.1 Study Area

The area chosen by NORDA for this study was the coastal area south of Panama City, Florida. This site was chosen because it was an region where different bottom types were available in close proximity to each other, thus minimizing

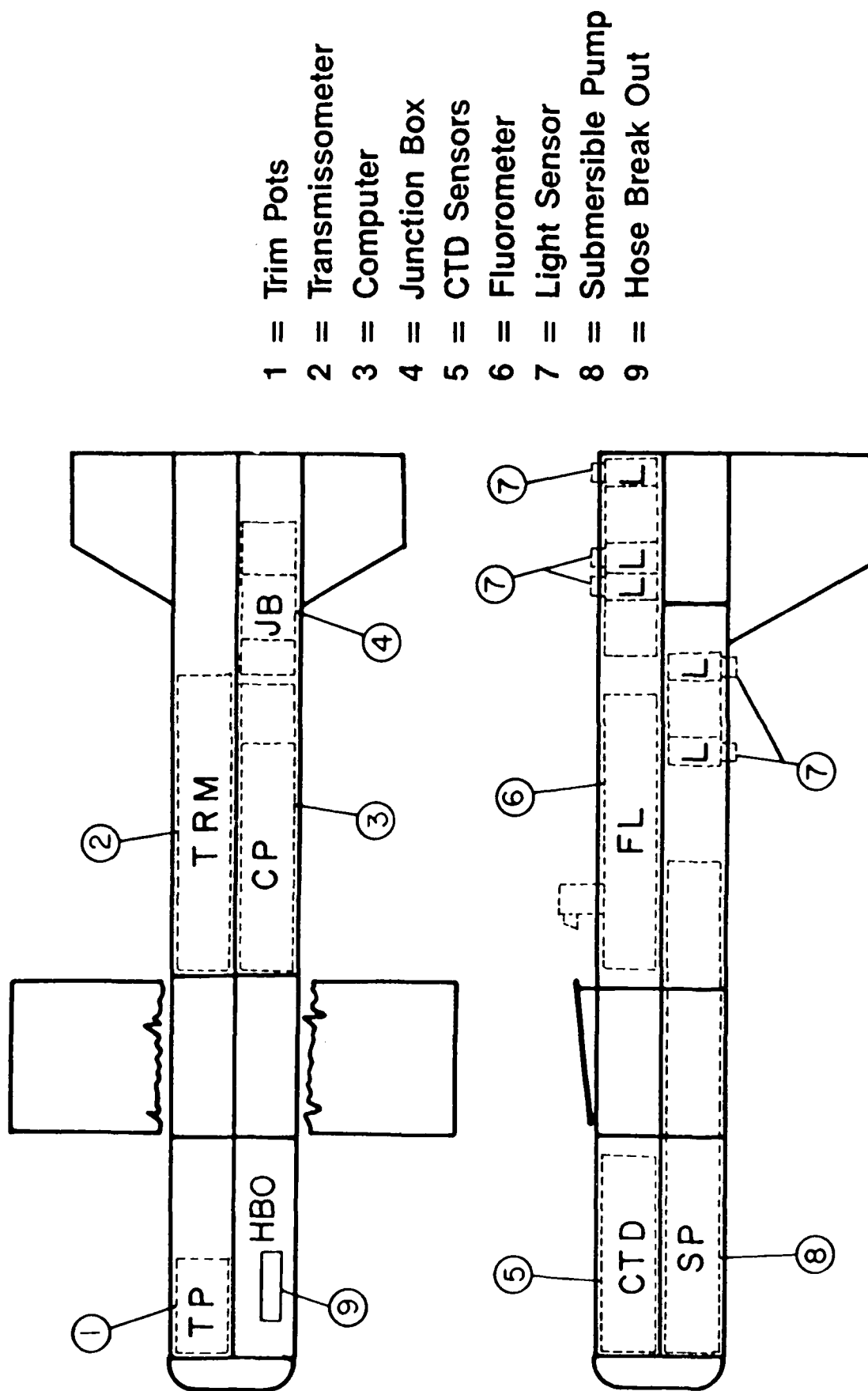


Figure 1. TUPS instrumentation layout as originally configured by NORDA.

the ship travel time required to study several different areas. The Panama City site was also selected for its close proximity to NORDA and the ability of the ABS-equipped P-3 aircraft to operate effectively out of nearby Eglin Air Force Base.

The study area is shown in Figure 2. This map is a 55% reduction (original scale 1:25,000) of a section of NOAA National Ocean Survey Chart No. 11391 (DMA Stock No. 11BHA11391). The map encompasses the area where the TUPS was towed during this experiment. On June 19 and 20, 1988, TUPS was towed in the Gulf of Mexico south of Shell Island. On June 21, 1988 TUPS was towed in the shallow area north of Shell Island.

2.2 Cruise Itinerary

The cruises undertaken during this experiment were conducted aboard the vessel Captain Graydon York, a forty-five (45) foot crew boat. The vessel was loaded with the TUPS and TESS equipment while docked at Sun Harbor Marina on June 18, 1988. TUPS was configured to be towed from the port side of the vessel using a davit that had been constructed at Texas A&M University especially for this experiment.

The Captain Graydon York departed Sun Harbor Marina at 1436Z (0936 CDT) on June 19, 1988 and proceeded to a known reference position (30°02.72'N, 85.41.60'W) to calibrate the TESS LORAN-C system. The LORAN was correct to 0.01' of latitude and longitude. The vessel then proceeded out the

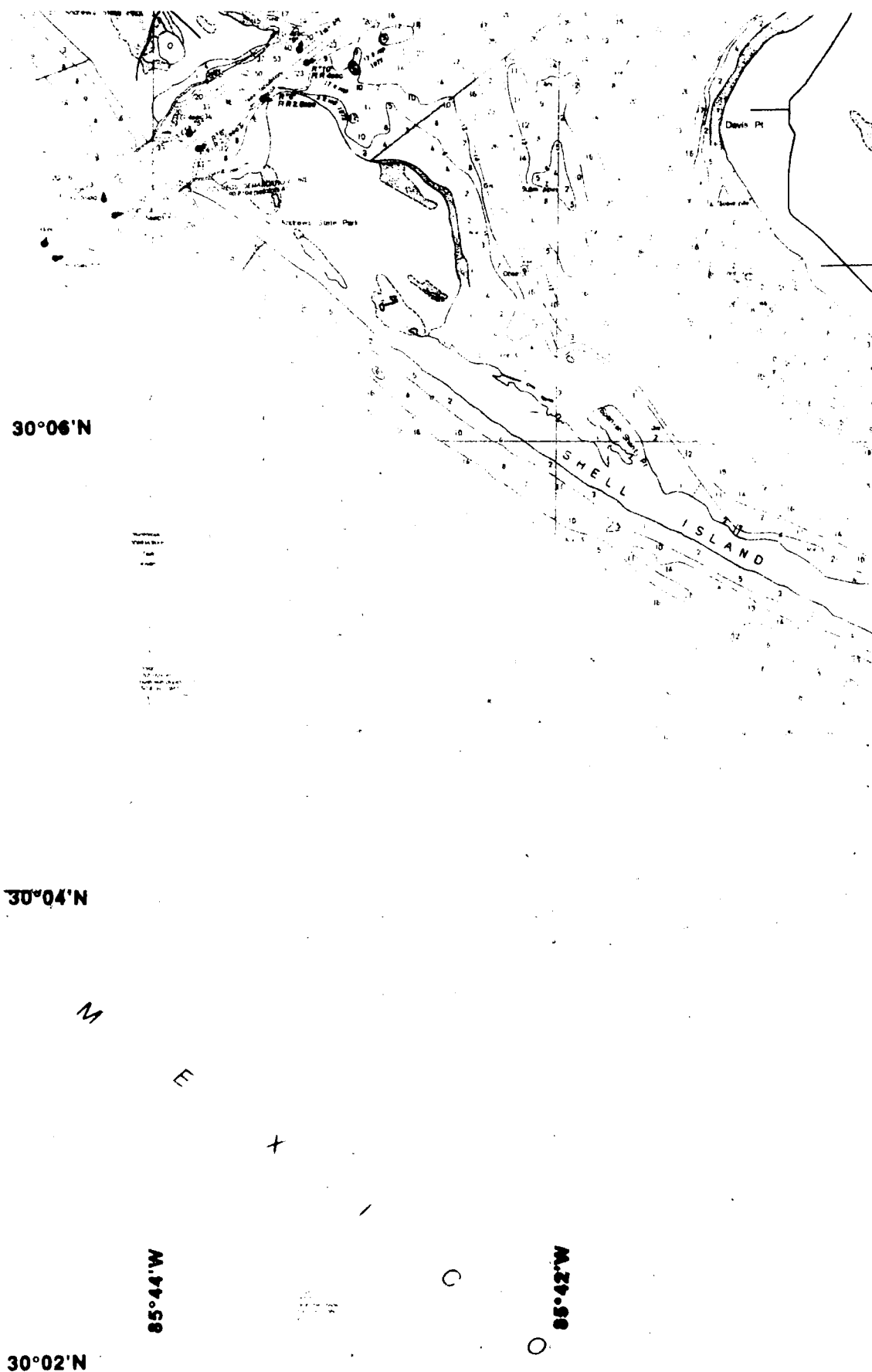


Figure 2. Chart of the study area south of Panama City, FL.

channel and south of Shell Island. TUPS was launched at 1600Z and after proceeding seaward was towed along lines perpendicular to the beach to provide a complete survey of the area. As the vessel approached the beach, we endeavored to go into as shallow water as possible. On two occasions (1847Z and 1854Z) the bow of the boat hit the bottom and severe course adjustments had to be made.

The TUPS survey south of Shell Island was completed at 2016Z. The TUPS tow vehicle remained in the water and collected data while anchor Stations 1 (2017Z-2102Z, 30°05.06N, 85°40.74W) and anchor Station 2 (2114Z-2202Z, 30°04.56N, 85°41.05W) were occupied. The TUPS tow vehicle was hoisted back aboard the vessel at the termination of anchor Station 2 and the vessel returned to port at 2252Z.

On June 20, 1988, the Captain Graydon York departed its dock at 1246Z and again performed a LORAN-C check at 1305Z. LORAN positions were exact with the reference position. The early portions of this day were devoted to a vertical station with the NORDA scanning radiometer system. Station 3 was occupied south of Shell Island at 30°02.15N, 85°45.95W from 1349Z to 1440Z.

After leaving this station, the vessel proceeded shoreward to launch TUPS in shallow water. TUPS was launched at 1504Z (1004 CDT) and a zig-zag pattern was run toward Shell Island with turns at 1542Z, 1552Z, 1558Z, 1605Z, 1611Z, and 1615Z. During the turn at 1615Z, the power generator aboard the Captain Graydon York stopped abruptly. The

generator had overheated due to a failure of the water pump on its cooling system. TUPS was retrieved at 1645Z and the vessel returned to its dock to affect repair of the faulty water pump. No further data were collected on June 20, 1988. Also, the data that had been collected were unreachable at that point as neither the TUPS or TESS data files had been closed properly when power was lost to the computer. The generator was repaired during the evening of June 20, 1988.

On June 21, 1988, the vessel left port at 1105Z (0605 CDT) proceeded to the LORAN reference point and then to a point on the leeward side of Shell Island (30°05.96N, 85°41.31W) where TUPS was launched at 1155Z. TUPS was towed until 1430Z when the vessel anchored for Station 4 at 30°06.27N, 85°41.72W. Station 4 was occupied from 1430Z to 1507Z. Station 5 (30°06.06N, 85°41.48W) was occupied from 1518Z to 1552Z and Station 6 (30°05.85N, 85°41.36W) was occupied from 1602Z to 1623Z. At this station, the TUPS tow vehicle was lowered from its normal tow depth beginning at 1602Z down to near the bottom and then returned to its normal tow depth at 1613Z. Data at Station 6 were collected with TUPS until 1621Z.

The Captain Graydon York then proceeded to Station 7 (30°05.97N, 85°41.08'W) which it occupied from 1630Z to 1762Z. At this station, another vertical cast was made with TUPS. The descent started from the surface at 1643Z and terminated 1.17 m above the bottom at 1651Z. The TUPS tow body was retrieved slowly and arrived back at its normal 1.0

meter tow depth at 1722Z. After the end of Station 7, the vessel proceeded to Station 8 at $30^{\circ}05.57'N$, $85^{\circ}40.61'W$ which it occupied from 1740Z to 1812Z. At the end of this station TUPS was retrieved and placed in its cradle in order to proceed rapidly to a point outside the jetties where it would be launched for a final survey through the Panama City channel.

TUPS was launched again at 1841Z at $30^{\circ}06.79'N$, $85^{\circ}44.13'W$ and the vessel proceeded on a course of $320^{\circ}T$ at 5.0 knots. This run across the channel was conducted from 1841Z to 1845Z. The transect was run about halfway between channel markers 1-2 and 3-4. After the channel crossing the vessel headed south then into the channel (at 1850Z) south of channel markers 1-2. The TUPS was then towed up the Panama City channel to the LORAN-C reference point ($30^{\circ}08.72'N$, $85^{\circ}41.60'W$) where TUPS was retrieved at 1926Z. The Captain Graydon York then returned to its dock, arriving at 1943Z (1443 CDT). The NORDA equipment was off-loaded at that time ending the Panama City field experiment.

2.3 TUPS Sensor Configuration

During the Panama City experiment, the towed underwater pumping system (TUPS) was configured slightly differently than shown in Figure 1. The original configuration included pitch and roll sensors (Trim Pots), had only the capability for two upwelling light sensors and had no depth measuring capability. For this experiment, the trim pots were removed

to provide two extra data channels. One was used for a light sensor and the other was used for an echo sounder.

A list of sensors used on the TUPS during this experiment is shown in Table 1. Three upwelling light sensors at fixed wavelengths (465 nm, 507 nm and 532 nm) were mounted in the lower quadrant of the tow vehicle along with an Ulvertech, Ltd. Model 205 Echo Sounder (Depth Sensor). The echo sounder produced 500 kHz pulses with a power of 150 watts and beam width of four degrees. The unit produces an output voltage of 0-10 VDC which is proportional to 0-100 meters. It has a resolution of 2 cm.

Although the Ulvertech echo sounder was not provided with calibration data, calibrations were made by actual depth measurements (using divers) at several stations during the cruises. The echo sounder reported within 1 cm of the measured depth at 3.1 meters and within 10 cm of the reported depth at 4.4 meters.

2.4 TESS Sensor Configuration

To collect incident light measurements for comparison with the TUPS data, we used NORDA's TUPS Environmental Sensor System (TESS). This system (not to be confused with the Tactical Environmental Satellite System) was used to record total irradiance using two Eppley pyroheliometers as sensors. These measurements are important since clouds passing overhead reduce the amount of light hitting the ocean surface and consequently reduce the amount of upwelling irradiance.

Table 1. TUPS Sensor Configuration, June 1988

Channel*	Sensor	S/N	Calibration ⁺
F0	Temperature sensor, Sea Bird, Inc.	SBE3-638	9-18-87
F1	Conductivity sensor, Sea Bird, Inc.	SBE4-234	9-18-87
A0	465 nm Upwelling Light, Biospherical Instr.	MCP-200H-7129	5-2-88
A1	507 nm Upwelling Light, Biospherical Instr.	MCP-200H-7130	5-2-88
A2	532 nm Upwelling Light, Biospherical Instr.	MCP-200H-7131	5-2-88
A3	Transmissiometer, Sea Tech, Inc.	165	12-31-86
A4	Fluorometer, Sea Mar Tech, Inc. (signal)	6000AR-235	None
A5	Fluorometer, Sea Mar Tech, Inc. (scale)	6000AR-235	None
A6	488 nm Downwelling Light, Biospherical Instr.	QCP-200LM-7101	2-22-86
A7	Echo Sounder, Ulvertech, LTD.	205	unknown

* TUPS on-board computer channel

⁺ Date of last calibration if known

TESS also contained two narrow-band irradiance sensors with fixed wavelengths of 441 and 488 nm. Table 2 lists the sensors used during the Panama City experiment.

Table 2. TESS Sensor Configuration, June 1988

Channel	Sensor
4	Voltage Reference, 2.5V, Analog Devices AD580M
5	Pyroheliometer 1, light bulb, Eppley Instruments, Model 50, S/N 3039
8	Pyroheliometer 2, hemisphere, Eppley Instruments, Model PSP, S/N 8022D1
11	488 nm Light (linear), Biospherical Instruments, Model QCP-200HM-488, S/N 7105
12	441 nm Light (log), Biospherical Instrument, Model QCP-200LM-441, S/N 7102

2.5 TUPS At-Sea Data Collection

All sensors from the TUPS tow vehicle output their signals to an onboard computer. The Sea Bird temperature and conductivity sensors produce a frequency output and all other sensors have analog (voltage) outputs. These signals are converted by the TUPS computer to a hexadecimal code which is sent in ASCII format to the computer aboard the vessel. The shipboard computer used to integrate and collect data from the TUPS computer was a Digital Equipment Corporation Professional 350 (DEC PRO-350) running under the P/OS operating system. The DEC PRO-350 collected a line of data

from the TUPS computer every 5-6 seconds, added a date and time to the data line and wrote the data line to the computer's hard disk. This operation was controlled by a BASIC program (SLOGTUPS) which collects the data, calculates and displays the results in engineering units and writes the raw data to disk.

2.6 TESS At-Sea Data Collection

The data collected from the TUPS environmental sensor system (TESS) include navigation information as well as light data. TESS data are collected with a Zenith Data Systems Model 121 (Z-121) computer running MS/DOS version 2.17. The navigation information is transmitted to the Z-121 computer via an RS-232 interface. The navigation data is collected from an INTERNAV LC-300 LORAN-C which outputs position, time delays and calculated speed and heading every 12 seconds. The LORAN output is used as a trigger for the Z-121 BASIC data logging program (LOGTESS). When the navigation data is sent to the Z-121, the computer records the day and time and samples the A/D board that is receiving data from the TESS light sensors. Each light sensor is read five times and an average is determined. The time and position information along with the average light voltages are displayed on the Z-121 screen and simultaneously written to the floppy disk in ASCII format.

2.7 Data Collection Problems

Relatively few problems occurred during collection of the data. The LORAN-C positions off Panama City, Florida, are very good due to the near-perpendicular crossings of the time delay lines. Most of the TUPS sensors were well-calibrated and produced acceptable data with a few exceptions (e.g. the fluorometer) that could be connected by judicious filtering of the data.

The one major problem did not involve the TUPS or TESS equipment, but rather the loss of ship's power at 1615Z on June 20, 1988. This event is described in section 2.2. The loss of power caused the computer disk files on the DEC-350 (hard disk) and the Z-121 (floppy disk) not to be closed properly. Although the data had been written to the disks immediately after it was collected, it could not be read in the normal fashion because location blocks for the file had not been written to the disk. These data were later recovered using a technique that is described in section 3.3.

3.0 DATA ANALYSIS

3.1 Inspection of Data Collected

Immediately after the Panama City experiment, the TUPS and TESS data files were transferred from the disk on which they were recorded (DEC 350 or Z-121) to the GERG VAX for processing. The files were scanned to make sure each data line was complete and that each file had ended with a complete line of data. Incomplete data lines were either

removed or edited if the error was obvious. Most of the data files were in excellent condition. It was obvious however, that there was a problem with spiking in the fluorometer data that would need attending to.

3.2 Problems with Collected Data

After the initial inspection, the TUPS and TESS files were converted to engineering units. The programs used are described in section 3.5. A quick look at the raw data indicated that there were significant problems with the data from downwelling light sensor in TUPS and the 488 nm TESS (atmospheric) sensor. The downwelling light sensor (looking up) in TUPS was supposed to be a Biospherical Instruments 400-700 nm broadband sensor that measures photosynthetically-active radiation (PAR). Inadvertently, a narrow band 488 nm sensor (Biospherical Instruments, Inc. Model QCP-200LM-7101) was installed in its place. This sensor was intended for the TESS system. The 488 nm sensor in TUPS was a logarithmic sensor intended to operate in air. The calibration for this unit is an air calibration thus the values reported for this sensor are questionable.

An incorrect sensor was also installed in the TESS. Instead of installing the logarithmic 448 nm sensor in TESS, a high gain 488 nm sensor (Biospherical Instruments, Inc. Model QCP-200HM-488-7105) had been installed previously during the Spring of 1988. The calibration for this unit was for use in water only, thus the data from this sensor is also

questionable. In fact, examination of the time series plots indicates that the sensor was saturated almost all the time during this three day experiment.

A more significant problem was the fluorometer data. About half of the data points were unusable. The SeaMarTech fluorometer can produce a noisy signal due to the capacitor discharge when its strobe light is flashing. If the TUPS computer samples at this time, an errant data point will be recorded. We have recommended that NORDA add an electronic filter to their fluorometer to correct this problem in the future. There is also a problem with the fluorometer when it is changing scales. The fluorometer may jump between scales rapidly before it settles on one scale. When the TUPS computer samples during this period of scale change an errant data point is recorded. Since the Sea Mar Tech fluorometer does not change scale very smoothly this process also added to the bad fluorometer data recorded.

There were only a few problems with the other TUPS sensors. When the vessel ran aground, much sediment was stirred-up into the water column. Sensors that work by having water pass through them (conductivity and transmission) are affected by this extra suspended sediment. Both these sensors had spikes in the data lines when the ship ran aground. These spikes were filtered from the data.

3.3 Recovery of Lost data

On 20 June 1988 the generator overheated on the Captain Graydon York causing the unit to shut down. The resultant power failure caused TUPS data files, written on the DEC PRO-350 hard disk, not to be closed. Although the data were written on the disk, the directory entry for the data files was not complete and the files could not be read in the usual fashion. Features for the recovery of the data were not available in the P/OS operating system of the DEC PRO-350 computer. We undertook the job of recovering these lost files so the TUPS data from June 20, 1988 could be saved.

Our strategy for recovering those files was as follows:

a. Boot an RT-11 system from a floppy disk on the PRO-350. RT-11 is another DEC operating system that can be used on the PRO-350. FORTRAN subroutines are available in the RT-11 system library that can bypass normal directory usage and allow individual blocks of data to be read anywhere on the hard disk.

b. On another DEC LSI-11 computer available at TAMUGERG, develop a FORTRAN program that could be operated under RT-11 and could search the disk and recover blocks containing Panama City data. All of the lost TUPS data contained the string "20-JUN-88" so the program was instructed to copy all blocks that contained this string to a new file on a RT-11 floppy disk. The program, named READ350, is listed in Appendix A.

c. The program READ350 was run on the PRO-350 and all data recovered to a RT-11 floppy disk.

d. The data on the RT-11 floppy disk was transferred to the VAX using the DEC LSI-11 computer available at TAMU-GERG.

e. The data was edited to clean up a few bad characters. Essentially all lost data was recovered in this manner.

The TESS data collected on the Zenith 121 computer was recovered by Robert A. Arnone at NORDA using a "Brown Bag" data recovery program. This recovery process provided multiple ASCII files of the data which were slightly overlapping. The recovered data files were transferred to the GERG VAX merged into a single file and edited to remove all duplicate data records produced by the recovery program. This recovery of the TESS data from June 20, 1988 provided a clean TESS file for processing along with the files from the other two days.

3.4 Data Analysis Protocol

TESS data records contain time and voltages in ASCII. TUPS data records contain time and date in ASCII together with voltages and frequencies in a hexadecimal format. Some sensors are inherently noisy. TESS and TUPS data are not collected at exactly the same time. The goal of the data analysis protocol is to convert the raw TESS and TUPS data into a unified data set containing time series of all measured variables in engineering units with erroneous data and noise removed as much as possible.

Analysis proceeds in a number of steps with intermediate results stored as files. The process is shown schematically in Figure 3 and Figure 4. Square boxes indicate data sets and parallelograms represent computer programs.

First, the TESS data are read and converted into engineering units by program READJUN. Formulas used to convert voltages to engineering units are best found in the programs in Appendix A.

Second, TUPS data are converted to engineering units by program CONVERT88.

Third, certain noisy measurements (depth, fluorescence) are filtered using program PRETABLE. Rather than just applying a low pass filter, PRETABLE tries to make intelligent decisions about what data points are noise and removes them from the time series.

Fourth, TESS data are interpolated to the time of the TUPS data and interpolated TESS data and the TUPS data are merged using program MERGETT. This is done for each of the three days. The result is 3 files containing the desired smooth data sets of all data collected on each day of the Panama City experiment.

Fifth, a quick look data plotting program (TTPLOT) was written to read the merged data set and plot each time series on on large poster-sized plot.

Sixth, a SAS program (READ.SAS) was written to read all final data sets and consolidate them into one SAS data set. SAS data sets are self documenting and facilitate the use of

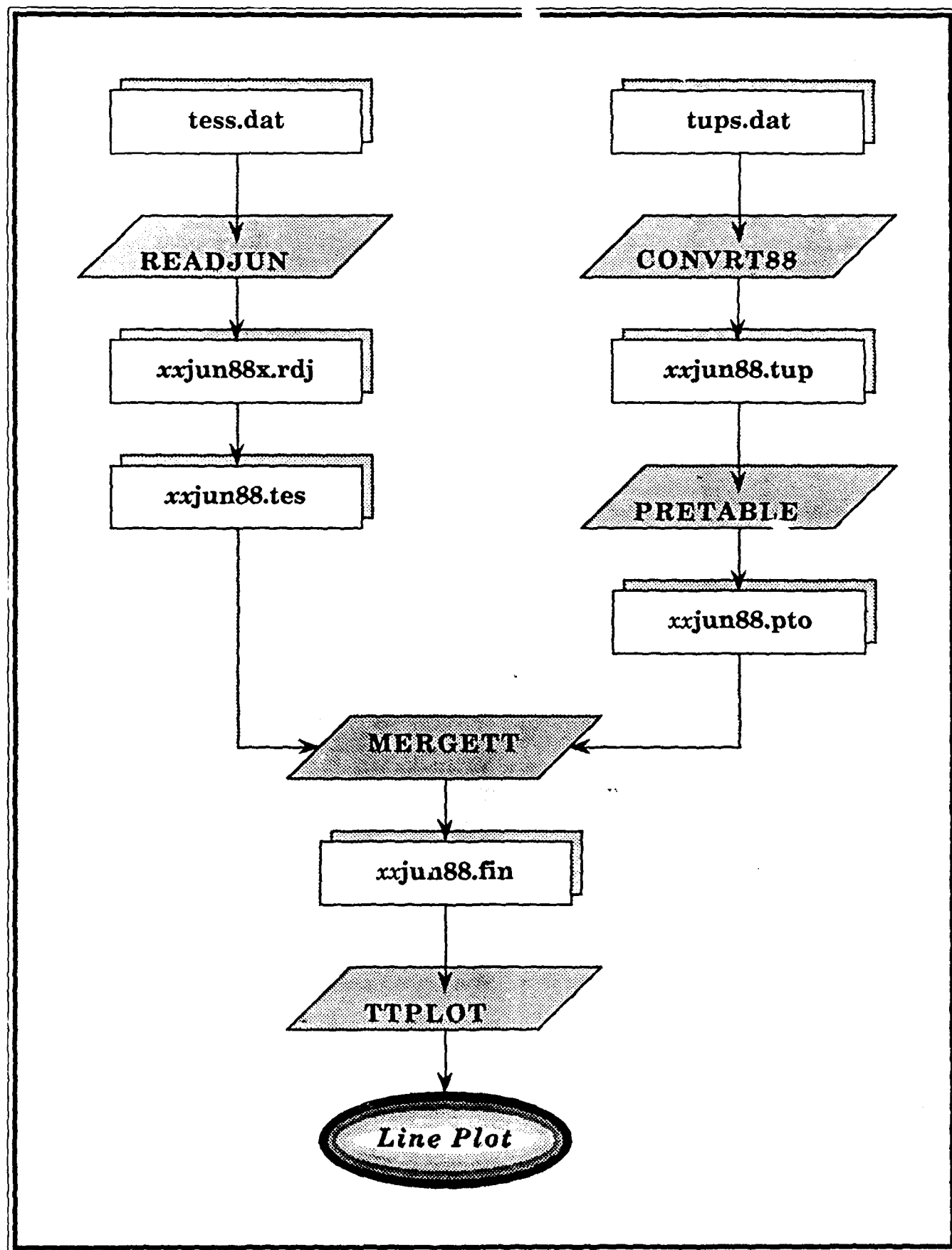


Figure 3. Flow diagram of data processing protocol showing FORTRAN programs used and data file names.

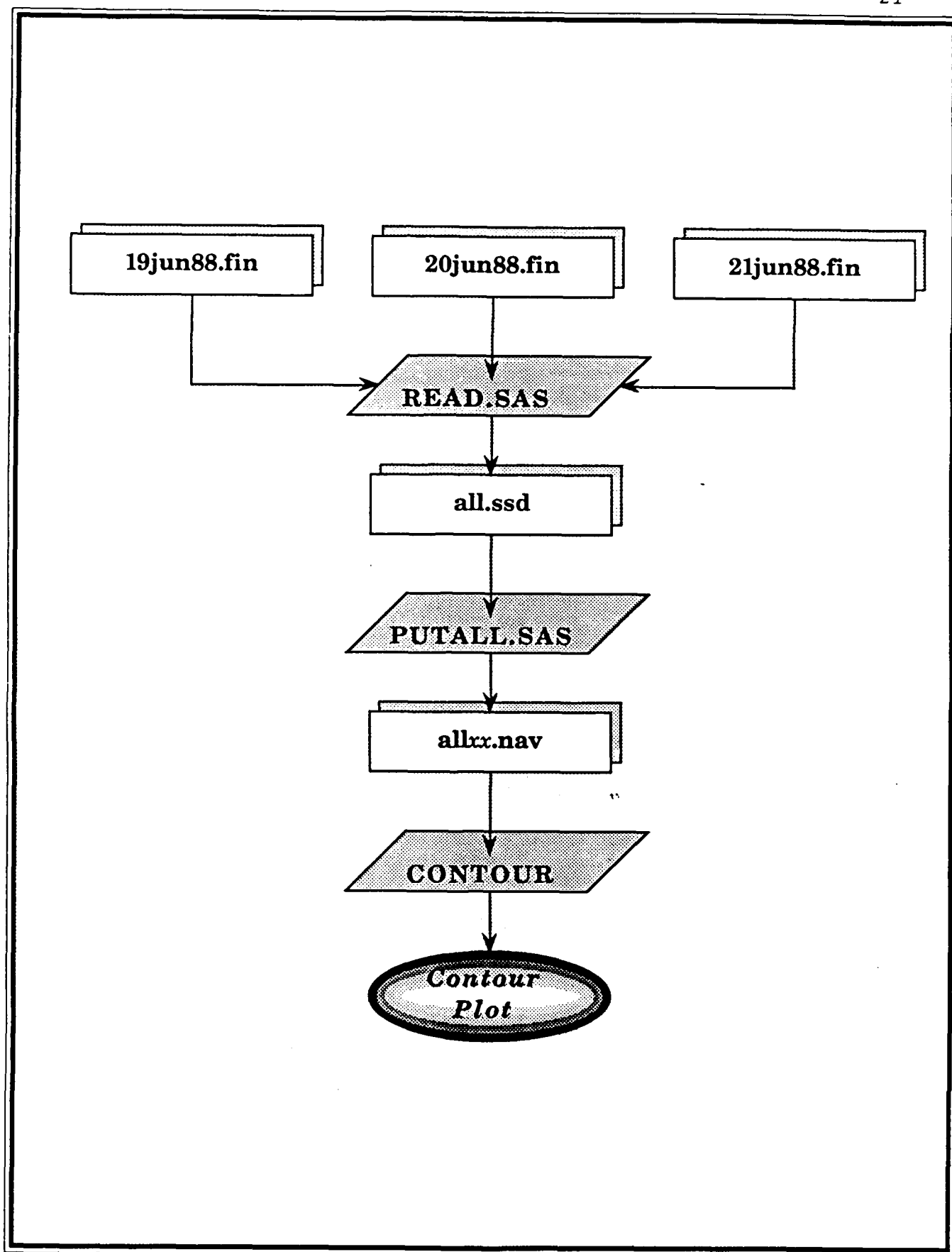


Figure 4. Flow diagram of data processing protocol showing SAS programs used and data file names.

powerful SAS statistical routines. Another SAS program was written to extract subsets of environmentally significant variables for further analysis.

Seventh, a contouring program (CONTOUR) was used to prepare contour plots for each of the three days for all environmentally significant variables.

The smooth data files that we recommend be used for further processing are the 19JUN88.FIN, 20JUN88.FIN and 21JUN88.FIN files produced by the MERGETT program. Table 3 gives a listing of the position in the record (array number) of each of the data items, the descriptive name of the data item, the SAS name used for the correlation analysis and the units of each data item as reported.

3.5 Description of Computer Programs

The following is a brief description of the programs that were used to analyze the data and to prepare the figures in this report. The listings of the programs are presented in Appendix A for information purposes only. The listing does not imply any transfer of ownership or guarantee that the program will operate on any data sets or computers other than those actually used in the course of this work. Some of these programs are based on previously-developed programs. They use subroutines of commercial or public domain origin available on the TAMU-GERG VAX.

Table 3. Position in data record (from XXJUN88.FIN files), descriptive name, SAS name and units for data presented in this report and on the data tape supplied to NORDA.

Position	Descriptive Name	SAS Name	Units
1	Year	-----	-----
2	Month	-----	-----
3	Day	-----	-----
4	Time	-----	HH:MM:SS (UT)
5	Seconds Past Midnight	-----	sec
6	Latitude	-----	deg
7	Longitude	-----	deg
8	Time Delay 1	-----	μ sec
9	Time Delay 2	-----	μ sec
10	Heading	-----	deg
11	Speed	-----	knots
12	Voltage Reference	-----	volts
13	441 nm light (TESS)	TESS441	μ W/cm ² sec nm
14	488 nm light (TESS)	TESS488	μ W/cm ² nm
15	Pyroheliometer 1	TESSPYR1	μ W/cm ² sec*10 ⁻⁴
16	Pyroheliometer 2	TESSPYR2	μ W/cm ² sec*10 ⁻⁴
17	Temperature	TEMPER	°C
18	Salinity	SALIN	PSU
19	% Transmission	PTRANS	%
20	Fluorescence	FLUOR	rel. units
21	465 nm Upwelling Light	L465NM	μ W/CM ² /nm
22	507 nm Upwelling Light	L507NM	μ W/CM ² /nm
23	532 nm Upwelling Light	L532NM	μ W/CM ² /nm
24	Transmissometer Voltage	-----	Volts
25	Fluorometer Signal Voltage	-----	Volts
26	Fluorometer Scale Voltage	-----	Volts
27	488 nm Downselling Light	L488NM	μ W/cm ² sec nm
28	Depth	DEPTH	Meters

1. **CONVERT88** (convert raw TUPS files).

This program reads the raw hexadecimal data file produced by the TUPS as configured during June 1988 and converts data to engineering units. The output files produced by this program have the extension "CNV".

2. **READJUN** (read June {TESS} files).

This program reads the files produced by the TESS program (as configured during June 1988) and produces a clean data file in engineering units. The output files produced by this program have the extension "RDJ".

3. **PRETABLE** (preprocess TUPS file).

This program reads the TUPS files produced by CONVERT88 and filters the data. A special filter (which does not average-in fliers) is used for the fluorescence, depth, and some other data. The output files produced by this program have the extension "PTO".

4. **MERGETT** (merge TESS and TUPS data).

This program reads the output from READJUN (TESS) and PRETABLE (TUPS) and produces a uniform data set containing TESS and TUPS data. TESS data is interpolated to TUPS times. The output files produced by this program have the extension "FIN".

5. **TTPLOT** (TESS TUPS plot).

This program uses NCAR subroutines to plot all elements of the unified data set against time. A large format plot showing all data is output on 32 in by 42 in inch paper.

6. **CONTOUR** (contour 13 TESS-TUPS variables).

This program contours data along the cruise track. The cruise track is shown as a dotted line. Land areas are also displayed. A masking algorithm is used to produce contours only within a certain distance of the cruise track. The same algorithm is used to mask out contours that might be drawn over land.

7. **M1** (map one).

This program is used to convert digitizer (x,y) data to latitude-longitude coordinates. A latitude-longitude grid is first digitized from 20 known positions to produce a calibration field. Then the data locations are digitized. The program does a third-order orthogonal polynomial fit to the calibration points and then uses the resultant coefficients to calculate latitude and longitude for the data locations. The program was used to provide the digitized island shown in the contour maps.

8. **READ.SAS** (read all final data sets)

This program reads final, merged data sets (date.fin) and converts all Panama City data into a SAS data set.

9. **PUTALL.SAS** (extract subset of environmental data for plotting to ASCII file)

This is a SAS program that can use simple statements to make an ASCII file to be used by the plotting programs.

10. **CORR.SAS** (make correlation matrix)

This program produces a correlation matrix of all the Panama City data.

11. **READ350** (read and recover PRO-350 data)

This is an RT-11 program that uses RT-11 system subroutine calls to read the hard disk on the PRO-350 and recover blocks of lost data. These data could not be read in the usual fashion because their directly entry was incomplete due to a power failure while the data were being written to the PRO-350 disk.

12. **PLOT.SAS** (make x vs. y plot)

This SAS program plots the inverse of NUR against depth. Inverse NUR (INORM_UP = TESSPYR1/L507NM) is plotted against depth for each day for all depths less than 12 meters

4.0 DATA PRESENTATION

4.1 Time Series Plots

Time series data for each day were plotted on 42" by 32" paper to facilitate examination of the data. Page size versions of these plots are shown in Figure 5, 6 and 7 for June 19, June 20, and June 21. Each time series is scaled such that it fills its own box. The minimum and maximum value for each box is indicated near the right hand edge. Time starts on even hours. For example, Figure 6 shows 1500Z to 1700Z. It is apparent that the TUPS upwelling irradiance sensors (Panels 7, 8, and 9 from the top) clearly measure the increase in upwelling irradiance in shallow water. Also evident is the general inverse relationship between fluorescence and transmission.

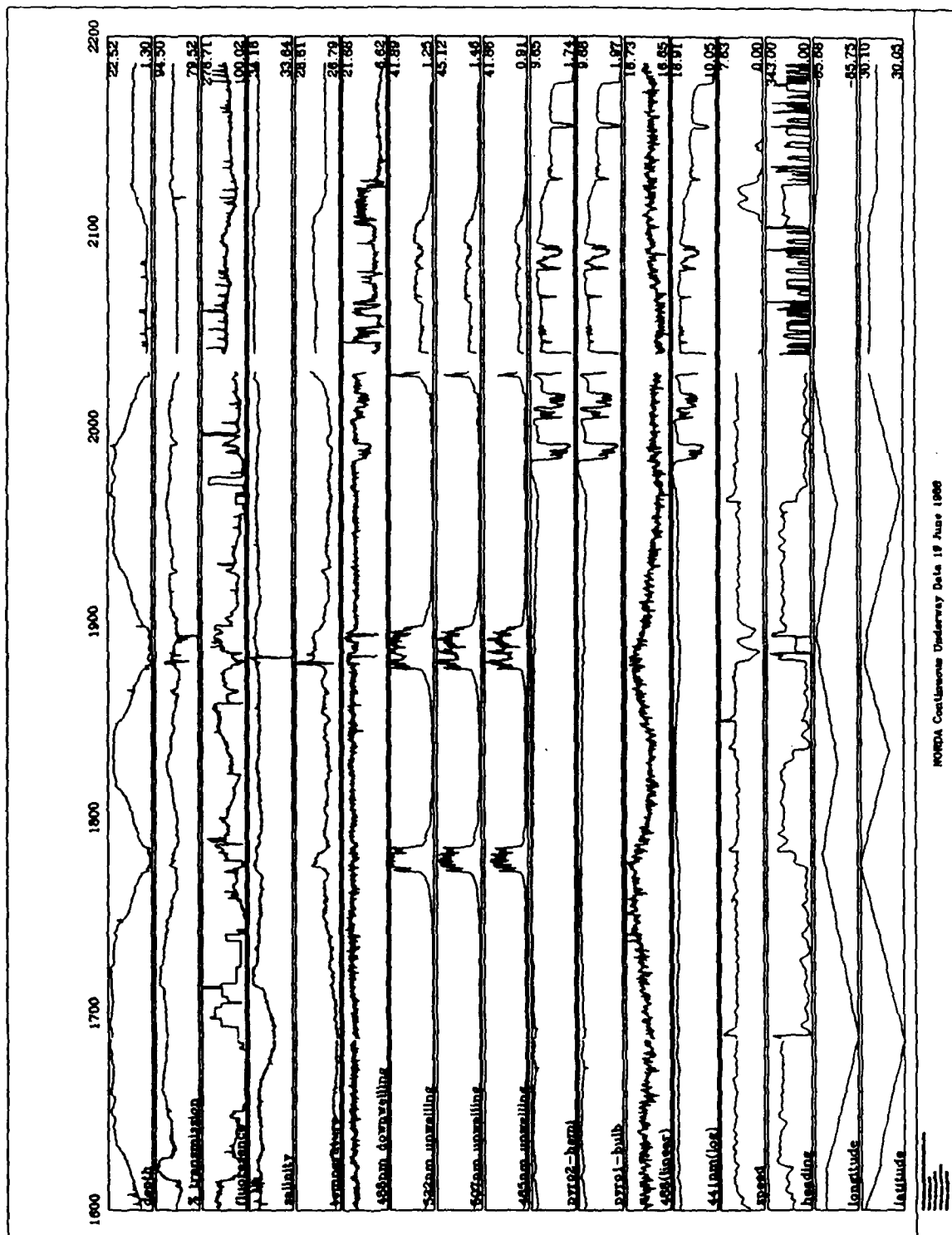


Figure 5. Time series plot of combined data from 19 June 1988.

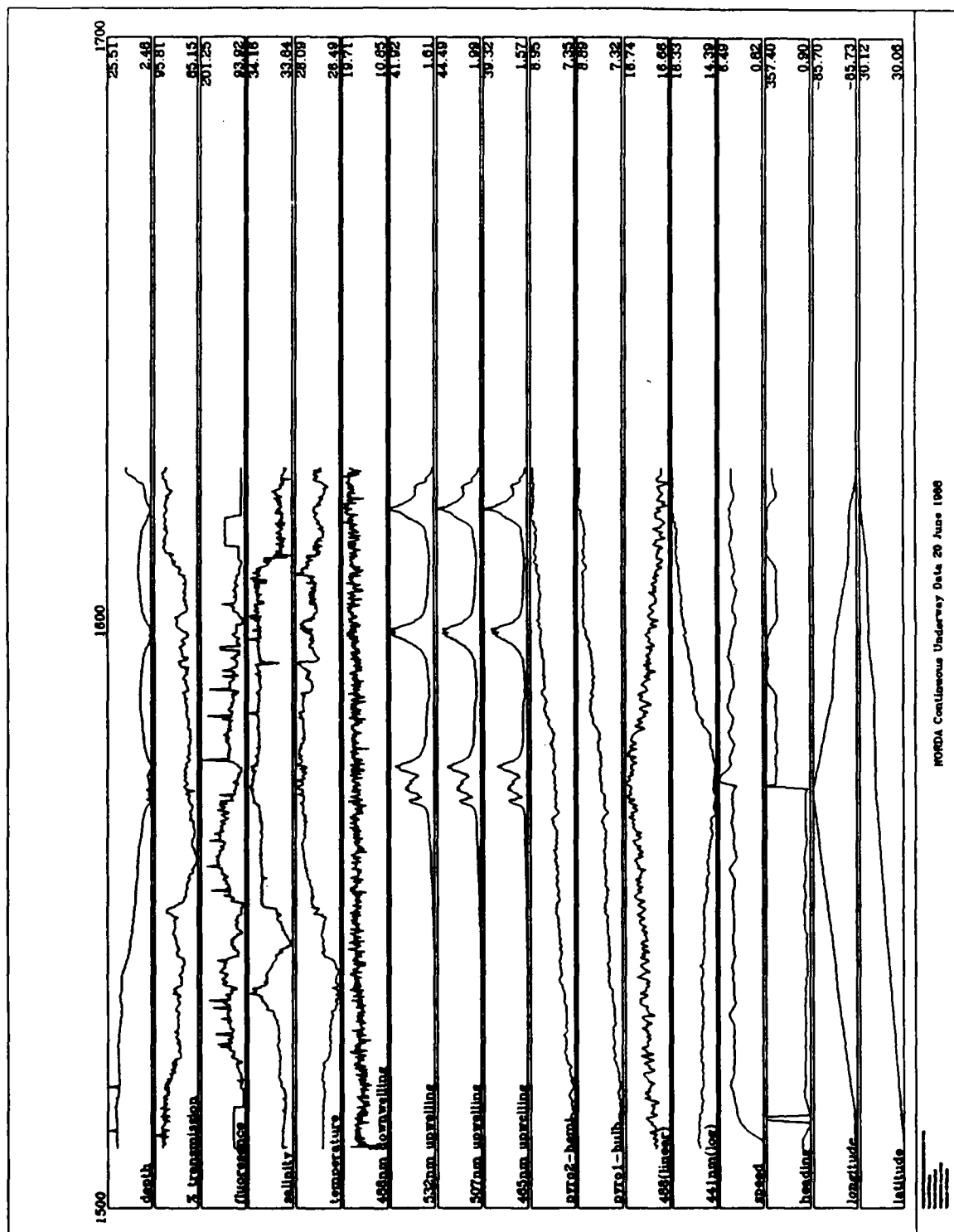


Figure 6. Time series plot of combined data from 20 June 1988.

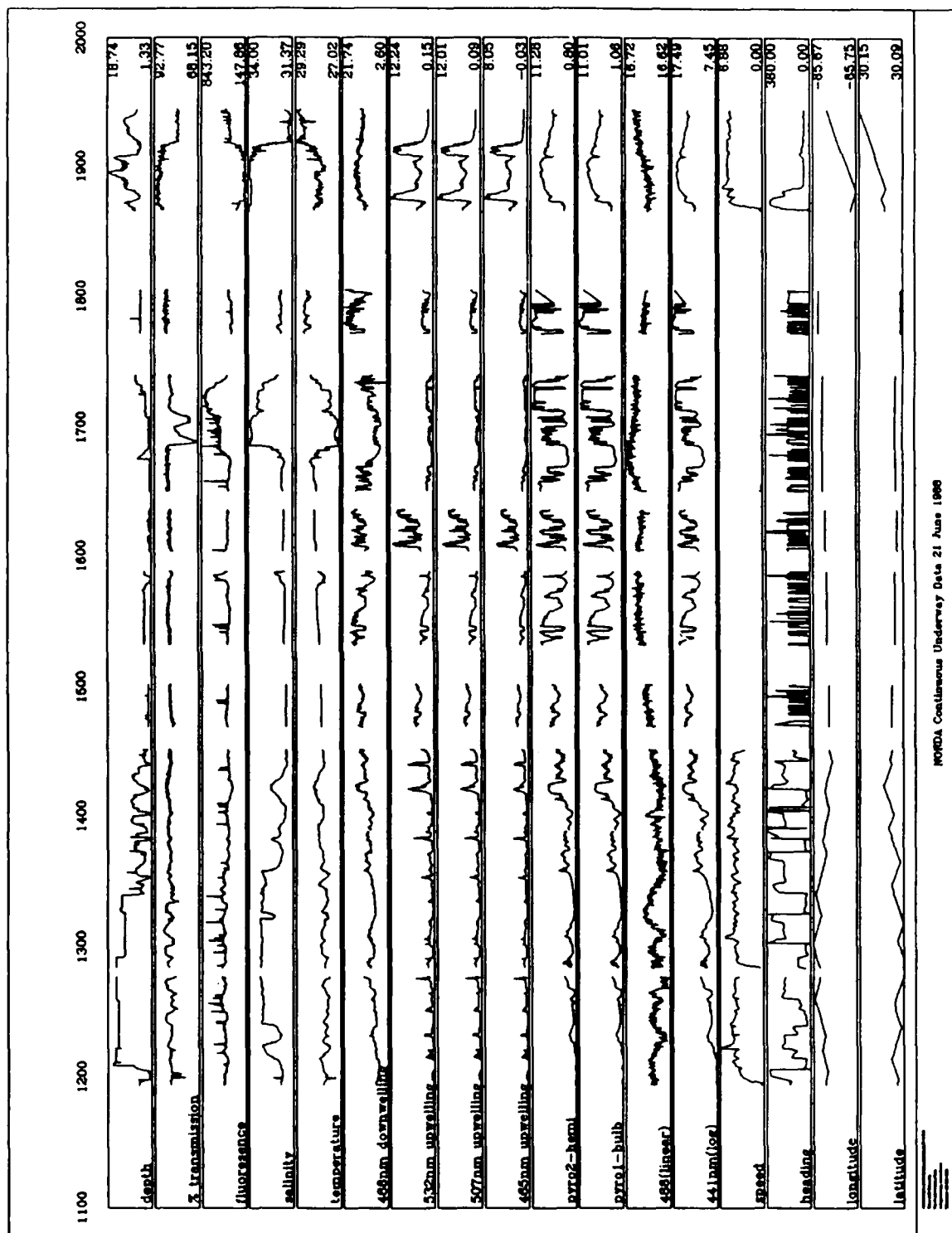


Figure 7. Time series plot of combined data from 21 June 1988.

4.2 Contour Maps for Each Day

Appendix B contains contour plots for 13 environmental variables for each day of the cruise. These are intended to give a quick look at the spatial quality of the data. Where cruise traces are relatively uniformly spaced and there are more than one track as on June 19, the data such as depth contour nicely. Passage of clouds show up dramatically as holes in the light field.

4.3 SAS Analysis of Total Data Set

As mentioned in sections 3.4 and 3.5, all data were placed in a SAS data file to facilitate statistical analysis. The Pearson correlation matrix was calculated for the entire data set (all 3 days) for the following variables:

- TESS light sensors: 441 nm, 448 nm, Pyrometer 1, Pyrometer 2
- TUPS sensors: Temperature, Salinity, Depth, Fluorescence, Percent Transmission
- TUPS light sensors: 465 nm, 488 nm, 507 nm, 532 nm.

This matrix is shown in Table 4. Correlation coefficients are highlighted when their absolute value exceeds 0.60 to draw attention to the most significant correlations.

Three of the four TESS light sensors are highly correlated with correlation coefficients exceeding 0.97. The 488 nm TESS sensor has a low correlation with the other three

Table 4. Pearson correlation matrix of TUPS and TESS measurements from the Panama City Experiment. Correlations with an absolute values over 0.600 are in bold.

VARIABLE	TESS448	TESS441	TESSPYR1	TESSPYR2	TEMPER	SALIN	PTRANS	FLUOR	L465NM	L507NM	L532NM	L488NM	DEPTH
TESS448	1.000												
TESS441	0.460	1.000											
TESSPYR1	0.517	0.978	1.000										
TESSPYR2	0.522	0.978	0.997	1.000									
TEMPER	-0.209	-0.331	-0.353	-0.355	1.000								
SALIN	0.281	0.480	0.499	0.499	-0.744	1.000							
PTRANS	0.163	0.514	0.504	0.504	-0.547	0.623	1.000						
FLUOR	-0.323	-0.640	-0.621	-0.615	0.528	-0.725	-0.791	1.000					
L465NM	0.262	0.364	0.379	0.380	-0.074	0.390	0.261	-0.414	1.000				
L507NM	0.259	0.352	0.367	0.366	-0.011	0.354	0.205	-0.380	0.989	1.000			
L532NM	0.249	0.320	0.336	0.336	0.033	0.310	0.155	-0.334	0.975	0.995	1.000		
L488NM	0.397	0.805	0.809	0.812	-0.340	0.399	0.600	-0.673	0.354	0.320	0.286	1.000	
DEPTH	-0.043	0.182	0.195	0.201	-0.596	0.456	0.561	-0.434	-0.255	-0.329	-0.374	0.350	1.000

TESS light sensors. This provides additional evidence that the TESS 488 nm sensor was not working properly.

Fluorescence is negatively correlated with transmission which implies that the principle light scatterers in the area are phytoplankton. Fluorescence is negatively correlated with the three light sensors which probably reflects the increasing efficiency of plankton fluorescence with decreasing ambient light. Fluorescence is negatively correlated with salinity which implies the fresher bay waters contain more chlorophyll.

Salinity and temperature are negatively correlated; the bay waters are warmer and fresher than the open ocean waters off the beach. Three TUPS light sensors measuring upwelling light are highly correlated.

We made a quick look assessment of the depth and light data to see how well water depth could be predicted by the intensity of upwelling irradiation. First we normalized the upwelling intensity to the downward radiation measured by the on board TESS sensors. Since the three TESS light measurements were highly intercorrelated and since the three TUPS measurements of upwelling irradiation were also highly correlated, we felt justified in choosing one of each for this normalization. We defined *normalized upwelling radiation*, NUR, to be the TUPS 507 nm sensor data divided by the TESS Pyroheliometer 1.

The inverse of NUR is plotted against depth in meters in Figure 8. This is a SAS plot where the letters indicate the

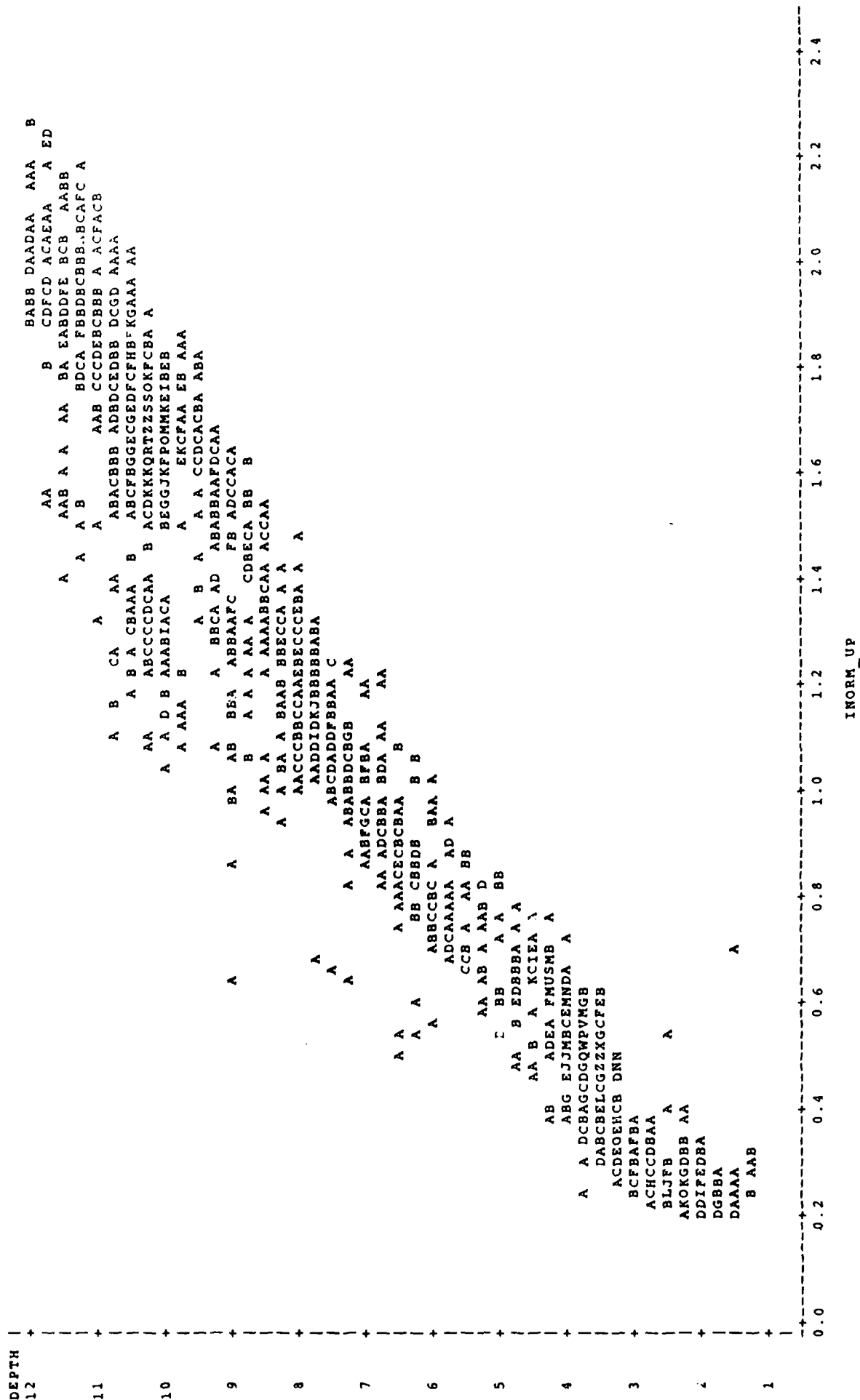
number of data items under each letter. An A represents one data item, a B represents two data items, and so forth. A Z indicates 26 or more data items. Figure 8 shows that for all the data collected on June 19, depth is roughly proportional to the inverse of NUR. Figure 9 show the same for the June 20 data. Again depth seems predictable by the inverse of the NUR.

Figure 10 for June 21 show a very different picture for the data collected behind the barrier islands. No simple relationship exists between NUR and water depth. Bottom types encountered on June 21 included mud, bright sand and grass. Bottom types encountered on June 19 and 20 were mostly bright sand. Bottom reflectance may play a role in distorting the relationship between NUR and water depth. Other environmental variables such as turbidity and pigment concentrations most certainly also play a role. The data, of course, warrant further analysis.

4.4 Description of Deliverables

Provided to NORDA in addition to this report were three 42" x 32" versions of Figures 5, 6 and 7. Also provided was a nine track magnetic tape in VAX BACKUP format that contains all data files described in this report, plus copies of all FORTRAN and SAS programs. Data files generally are ASCII FORTRAN output files with a carriage return line feed pair (CR-LF) at the end of each line. The BACKUP log that contains

PLOT OF DEPTH*INORM_UP LEGEND: A = 1 OBS, B = 2 OBS, ETC.



NOTE: 31 OBS HIDDEN

Figure 8. Plot of inverse of normalized upwelling irradiance versus depth on 19 June 1988.

PLOT OF DEPTH*INORM_UP LEGEND: A = 1 OBS, B = 2 OBS, ETC.

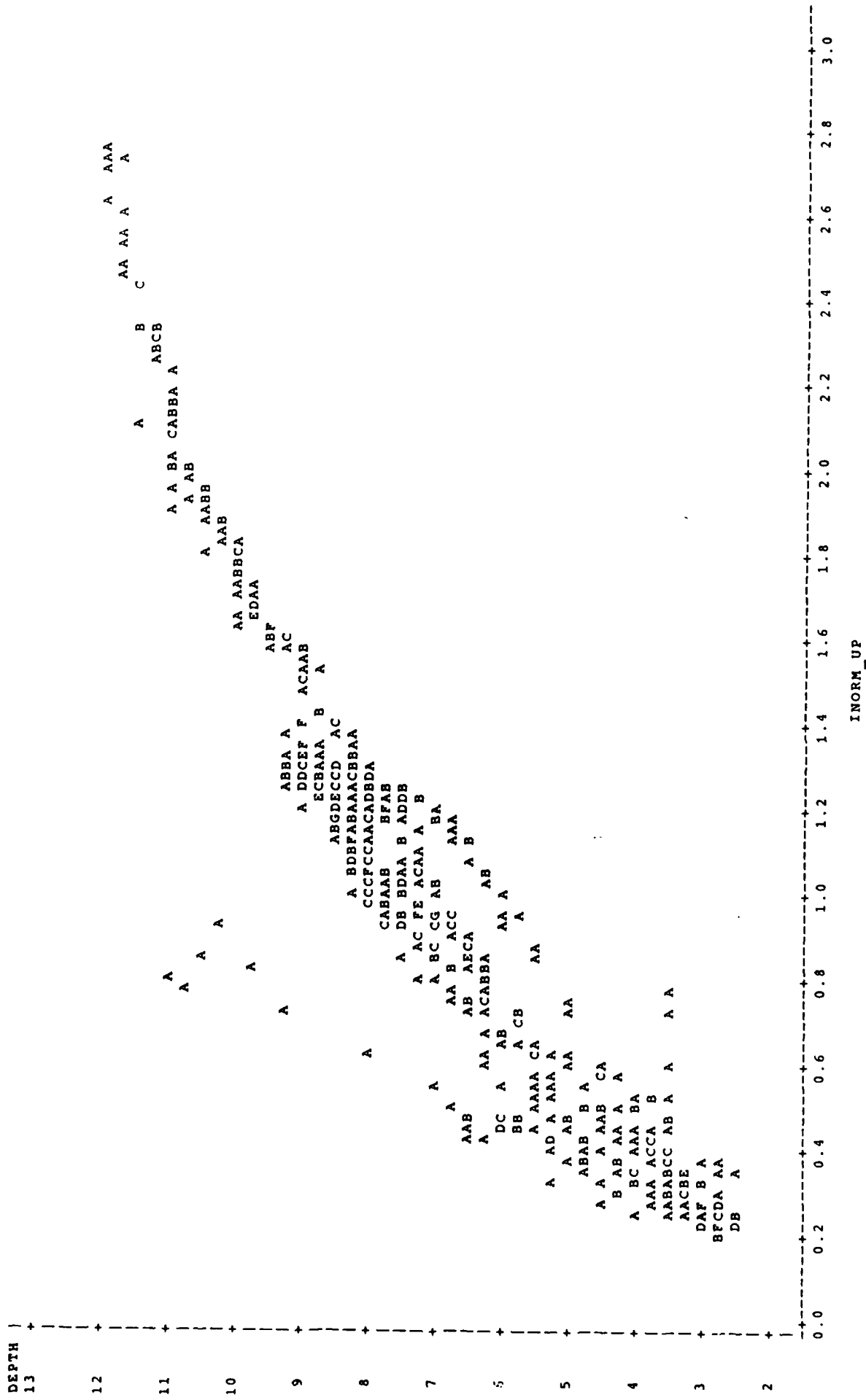
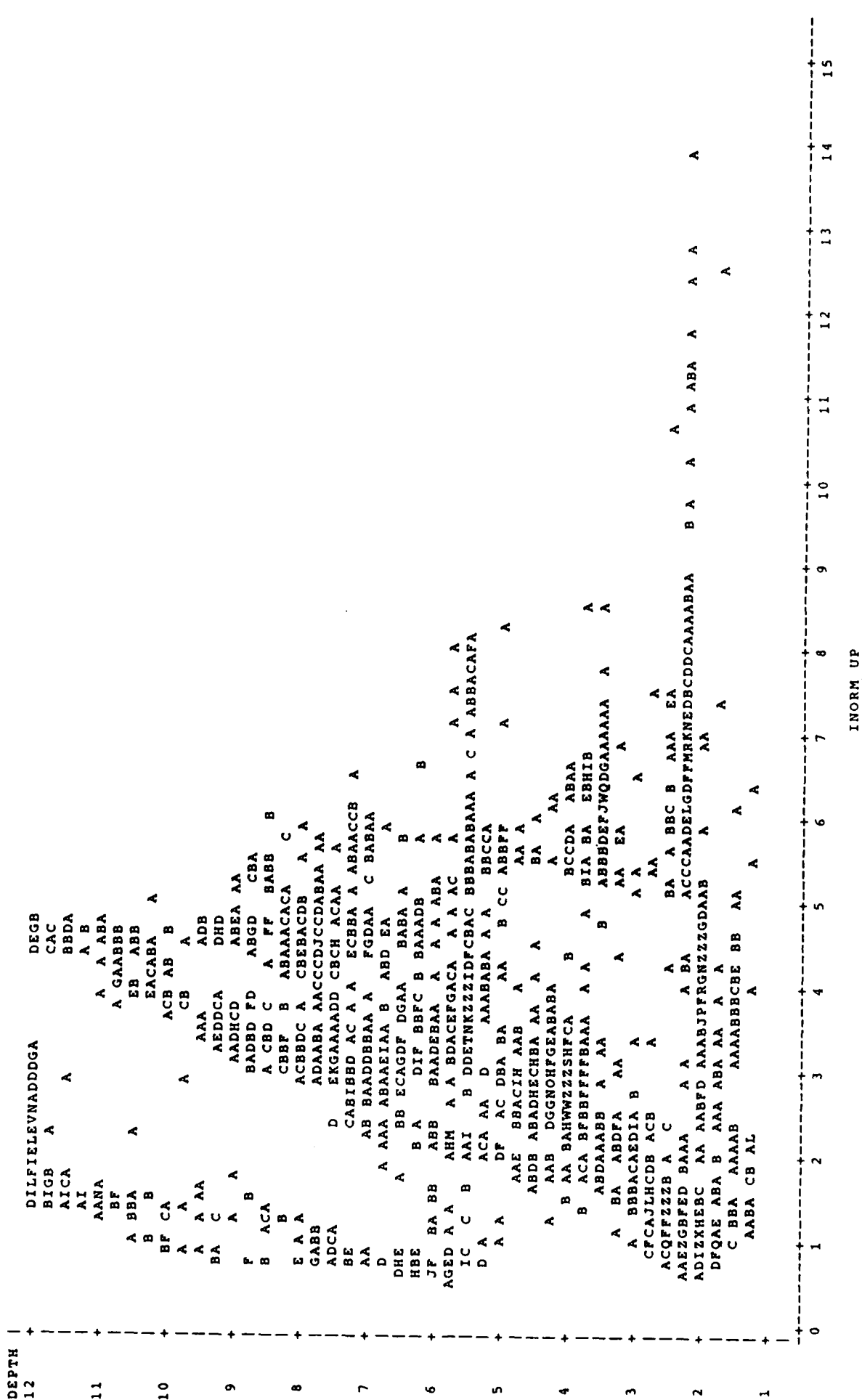


Figure 9. Plot of inverse of normalized upwelling irradiance versus depth on 20 June 1988.

PLOT OF DEPTH*INORM_UP LEGEND: A = 1 OBS, B = 2 OBS, ETC.



NOTE: 410 OBS HIDDEN

Figure 10. Plot of inverse of normalized upwelling irradiance versus depth on 21 June 1988.

a listing of the names of all files on the tape is provided as Appendix C.

5.0 CONCLUSIONS

All TUPS and TESS data collected from the NORDA Panama City experiment has been processed, merged, and placed in a form amenable for further analysis. The data set is relatively complete, as we were able to recover the data "lost" due to a ship power failure on June 20, 1988. From the combined TUPS/TESS data, we produced large-scale time series plots showing all variables. These plots are excellent tools for examining the quality of the data set and for taking a quick look at parameter-parameter relationships. All light data appear to be of high quality, with the possible exception of the 488 nm downwelling light data. The TUPS depth sensor seemed to work exceptionally well.

The TUPS/TESS system appears to be quite useful for examining the optical-depth relationship in shallow water. Water depth can be inferred from measurements of upwelling irradiance under some conditions. Preliminary examination of this data set indicates the relationship holds well in areas of bright sand (e.g. south of Shell Island), but the relation of light to depth is more complicated in variable bottom and water clarity conditions. The broad suite of light and water quality sensors in TUPS makes it ideal for resolving such problems.

6.0 REFERENCES

- Arnone, R.A. and D.A. Wiesenburg. 1988. Upwelling irradiance distribution across frontal zones and implications to ocean processes. Proc. of the Inter. Soc. Opt. Eng. (SPIE), Ocean Optics IX 925: 124-130.
- Rein, C.R., D.A. Wiesenburg and D.M. Lavoie. 1985. A towed instrument vehicle for deep ocean sampling. Naval Ocean Research and Development Activity, NSTL, MS. NORDA Report 90, 48 pp.

7.0 ACKNOWLEDGEMENTS

This study was a joint program with the Mapping, Charting and Geodesy Division of the Naval Ocean Research and Development Activity. The Texas A&M portion of the study was funded by the Mapping, Charting and Geodesy Division of NORDA through Office of Naval Research (ONR) Contract Number N00014-88-K-6003 to the Texas A&M Research Foundation (Robert A. Arnone, NORDA Scientific Officer). The Mapping, Charting and Geodesy Division program was funded by the Oceanographer of the Navy (CNO OP-096) and the Defense Mapping Agency in support of the development of the Airborne Bathymetric System (ABS). We thank Ed Baxter, Mark Hulbert, Jim Goudeau and George Krebs for their assistance in setting up and operating TUPS and TESS aboard the Captain Graydon York. We also thank Dr. Edward Green of ONR for his continued support through the years and Capt. Tex Treadwell (US Navy-Retired, former Oceanographer of the Navy) for his understanding.

APPENDIX A

COMPUTER PROGRAM LISTINGS

```

C*****
C
C PROGRAM          CONVRT88
C
C PURPOSE          1) reads in raw TUPS data files in hex
C                  2) converts data to engr. units with proper calib.
C
C AUTHOR           D.A.WIESENBURG
C
C DATE             2/9/88
C
C REVISION         NORMAN GUINASSO
C                  DEC. 88
C
C INPUT FILE IS GIVEN AND OUTPUT FILE OF SAME NAME.CNV PRODUCED
C
C*****
C
C CHARACTER*3 MONTH
C CHARACTER*8  CHTIME,HHMMSS
C CHARACTER*9  CHDATE,DDMMYY
C CHARACTER*30 FILEN
C CHARACTER*34 CNVDAT,RAWDAT
C CHARACTER*60 RRAW
C CHARACTER*76 BUFFER
C CHARACTER*4  ANALG(8)
C CHARACTER*6  FRQ(8)
C
C REAL*4 V(8)
C REAL*8 TIME1(10000),TIMEA(10000)
C
C INTEGER*4 STR$FIND FIRST IN SET
C COMMON /DAYS/ IDAY,IMON,IYR,jhr,jmin
C COMMON /DATIM/ CHDATE,CHTIME
C COMMON / RAW / ANLG,FRQ
C COMMON / DAY / DMY,HMS
C COMMON / CNVRT / COND,SAL78,TEMP,VOLT,FREQ,XL4,XL8,PSI,
C                  DEPTH,TRANS,TLIGHT,CHLA,MINUTE,RTIME
C
C oldv5=1.0      ! initialize value used for fluor. sig. cleaning
C
C WRITE(6,100)
100  FORMAT(' ENTER FILENAME FOR CURRENT RUN',/,
1    ' INCLUDING EXTENSION (<= 30 CHARS)',/)
C READ(*,'(a)') FILEN
C rawdat=filen
C call chg_ext(rawdat,cnvdat,'UNF')
C OPEN(UNIT=18,FILE=RAWDAT,STATUS='OLD')
C OPEN(UNIT=19,FILE=CNVDAT,STATUS='NEW',form='unformatted',DISP='DELETE')
C call chg_ext(rawdat,cnvdat,'CNV')
C
C i=0
C do while(.true.)
C     read(18,'(a)',iostat=ios) buffer
C     if(ios.eq.-1) goto 200
C     i=i+1
C     J=STR$find first in set(buffer,':')
C     read(buffer(j-2:j+2),'(i2,1x,i2)') ih,im
C     timel(i) = 60.d0*(60.d0*ih + im)
C     jhr      =ih
C     jmin     =im
C
C the seconds are all 00 on TUPE data files

```

```

j=str$find first in set(buffer,'-')
read(buffer(j-2:j+6),'(i2,1x,a3,1x,i2)') iday,month,iyr
IMON = IDECMONTH(MONTH)
inum=i
RRAW = BUFFER(1:60)
CALL CONVRT(RRAW,i)

enddo

200      call timefix(time1,timea,inum)
      rewind(19)
      CALL DATE(DDMMYY)
      CALL TIME(HHMMSS)
      open(20,name=cnvdat,status='new',
*      RECL=255,carriagecontrol='list')

      WRITE(20,150) CNVDAT,DDMMYY,HHMMSS
150      FORMAT(' FILE: ',A35,5X,' CREATED: ',A9,2X,A8,/)
      WRITE(20,160)
160      FORMAT(1X,4HDATE,5X,5HSTIME,4X,5HZTIME,5X,4HTEMP,3X,5HSAL78,
1      4X,5H%TRAN,3X,6HCHL-FL,5X,5H465NM,3X,5H507NM,3X,5H532NM,
2      3X,6HTRANSV,2X,6HFLSIGV,2X,6HFLSCLV,2X,6H488-UP,3X,5HDEPTH,/)
C      3      2X,7('-'),3X,8('-'),1X,5('-'),2X,5('-'),2X,6('-'),
C      4      2X,5('-'),2X,6('-'),2X,5('-'),2X,5('-'))

      DO WHILE (.TRUE.)
      read(19,end=500)
*      i,IDAY,IMON,IYR,jhr,jmin,TEMP,SAL78,TRANS,CHLA,(V(K),K=1,8)
      write(20,1000,IOSTAT=IOS) iyr,imon,iday,timea(i),
*      jhr,jmin,
*      TEMP,SAL78,TRANS,CHLA,(V(K),K=1,8)

      ENDDO

500      close(20)
      CLOSE(UNIT=18)
      CLOSE(UNIT=19)
1000     FORMAT(I2.2,'/',I2.2,'/',I2.2,1x,f8.1,2x,i2.2,':',i2.2,':00',1x,
*      F6.2,F8.3,F8.2,F9.2,1X,'|',6F8.3,F9.3,1X,F7.2)
      STOP
      END

!
integer*4 function idecmonth(chr)
character *(*) chr
character*3 months(12)
data months / 'JAN','FEB','MAR','APR','MAY','JUN',
*             'JUL','AUG','SEP','OCT','NOV','DEC'/
call str$upcase(chr,chr)
DO I=1,12
      IF(CHR.EQ.MONTHS(I)) THEN
            IDECMONTH=I
            RETURN
      ENDIF
ENDDO
TYPE *, ' IDECMONTH-W-BAD MONTH'
RETURN
END

      SUBROUTINE CONVRT(RRAW,i)

C      CNVRT opens existing TUPS' data files, one at a time, and first
C      divides the strings into frequencies and analogs, then converts
C      the strings to various data parameters, which are written to
C      another output file.

C      modified at Texas A&M Univeristy by Guinasso and Wiesenbunrg

```

```

C June 1988 -- for Panama City Cruise
C
C CALLED FROM COLLECT AND CALLS CNVTLT AND PARSE
C
C CHARACTER*60 RRAW
C CHARACTER*4 ANLG(8)
C CHARACTER*6 FRQ(8)
C CHARACTER*8 CHTIME
C CHARACTER*9 CHDATE
C REAL V(8)
C
C COMMON /DAYS/ IDAY,IMON,IYR,jhr,jmin
C COMMON / DATIM/ CHDATE,CHTIME
C COMMON / CNVRT / COND,SAL78,TEMP,VOLT,FREQ,XL4,XL8,PSI,
1 DEPTH,TRANS,TLIGHT,CHLA,MINUTE,RTIME
C COMMON / RAW / ANLG,FRQ
C
C DATA XLFACT /0.121E17/
C
C -- Blank out the ANLG and FRQ arrays
C
C CALL ZERO
C
C -- Divides TUPS string into frequencies and analogs
C
C CALL DIVIDE(RRAW)
C
C -- Convert data parameters
C
C CALL CNVTMP !temperature - 1st frequency
C CALL CNVCND !conductivity - 2nd frequency
C CALL CNVPSI !pressure - 3rd frequency
C
C FORWARD LIGHT SENSOR IN FIRST CHANNEL 0-5 VOLTS
C CALL CNVVLT(ANLG(1))
C V(1) = VOLT*2.0833 ! Factor for 2.4 volt/5 volt conv.
C V(1) = V(1)*10.9459 ! CAL. FACTOR FOR 465 LIGHT SENSOR 6/88
C ! UNITS ARE uW/cm2/nm
C
C MIDDLE LIGHT SENSOR IN SECOND CHANNEL 0-5 VOLTS
C CALL CNVVLT(ANLG(2))
C V(2) = VOLT*2.0833 ! Factor for 2.4 volt/5 volt conv.
C V(2) = V(2)*10.3644 ! CAL. FACTOR FOR 507 SENSOR 6/88
C ! UNITS ARE uW/cm2/nm
C
C AFT LIGHT SENSOR IN THIRD CHANNEL 0-5 VOLTS
C CALL CNVVLT(ANLG(3))
C V(3) = VOLT*2.0833 ! Factor for 2.4 volt/5 volt conv.
C V(3) = V(3)*9.8568 ! CAL FACTOR FOR 532 LIGHT SENSOR 6/88
C ! UNITS ARE uW/cm2/nm
C
C TRANSMISSOMETER IN FOURTH CHANNEL 0-5 VOLTS
C CALL CNVVLT(ANLG(4))
C V(4) = VOLT*2.0833 ! Factor for 2.4 volt/5 volt conv.
C TRANS=100.*((4.738/4.46)*(V(4)-.001))/5. ! SENSOR #165
C
C FLUOROMETER SIGNAL IS FIFTH CHANNEL 0-10 VOLTS
C CALL CNVVLT(ANLG(5))
C V(5) = VOLT*4.1667 ! Factor for 2.4 volt/10 volt conv.
C
C check for bad fluorometer voltage and replace if bad
C if (v(5) .gt. 9.0) then
C v(5)=oldv5

```

```

end if
oldv5=v(5)

```

```

C FLUOROMETER SCALE IS SIXTH CHANNEL 0-10 VOLTS
CALL CNVVLT(ANLG(6))
V(6) = VOLT*4.1667 ! Factor for 2.4 volt/10 volt conv.

```

```

C CALCULATE REL. FLUORENCE USING SIGNAL AND SCALE VALUES
CHLA=V(5)*(2.0**ININT(V(6))) ! Make scale volts an integer

```

```

C UP LOOKING LIGHT SENSOR IS EIGHTH CHANNEL 0-10 VOLTS
C UNITS OF uW/cm3/nm
CALL CNVVLT(ANLG(7))
V(7) = VOLT*4.1667 ! Factor for 2.4 volt/10 volt conv.
V(7) = EXP(V(7)-0.900)*5.88 ! 448 LOG SENSOR USUALLY IN TESS
C **** THIS IS DIFFERENT THAN ALL OTHER TUPS CRUISES ****

```

```

C ECHO SOUNDER IS SEVENTH CHANNEL 0-10 VOLTS
CALL CNVVLT(ANLG(8))
V(8) = VOLT*4.1667 ! Factor for 2.4 volt/10 volt conv.
V(8) = (V(8)*10.0)+1.0 ! meters=volts*10 + 1.0 m tow depth

```

```

WRITE(19)
* i,IDAY,IMON,IYR,jhr,jmin,TEMP,SAL78,TRANS,CHLA,(V(K),K=1,8)

```

```

RETURN
END

```

```

SUBROUTINE ZERO

```

```

CHARACTER*6 F(8)
CHARACTER*4 A(8)

```

```

COMMON / RAW / A,F

```

```

DO 10 J=1,8
  F(J) = ' '
  A(J) = ' '
10 CONTINUE

```

```

RETURN
END

```

```

SUBROUTINE HEXDEC(F)

```

```

CHARACTER*4 O
CHARACTER*6 F
INTEGER*2 G
CHARACTER*1 CM,NUM

```

```

COMMON / CNVRT / COND,SAL78,TEMP,VOLT,FREQ,XL4,XL8,PSI,
1 DEPTH,TRANS,TLIGHT,CHLA

```

```

CM = F(1:1)
XM = XMULT(CM)

```

```

IF (XM.NE.0.) THEN
  DO 50 I=3,6
    O(I-2:I-2) = F(I:I)
50 CONTINUE

```

50

```
      LENGTH = 4
      XN = 0.
      P = 1.
      DO 100 I=LENGTH,1,-1
        G=ICHAR(0(I:I))
        IF(G.GT.64) THEN
          D=FLOAT(G)-55.
        ELSE
          NUM = 0(I:I)
          IF (NUM.EQ.'0') D = 0.  ! ADDED BY K.D.SAUNDERS 9/25/86
          IF (NUM.EQ.'1') D = 1.
          IF (NUM.EQ.'2') D = 2.
          IF (NUM.EQ.'3') D = 3.
          IF (NUM.EQ.'4') D = 4.
          IF (NUM.EQ.'5') D = 5.
          IF (NUM.EQ.'6') D = 6.
          IF (NUM.EQ.'7') D = 7.
          IF (NUM.EQ.'8') D = 8.
          IF (NUM.EQ.'9') D = 9.
          D=FLOAT(G)
        END IF
        XN=XN+D*P
        P=P*16.
100    CONTINUE
        FREQ=10.0E06*XN/XN
      ELSE
        FREQ=0.
      END IF
C
C*** END OF SUBROUTINE HEXDEC
C
      RETURN
      END
```

```
FUNCTION XMULT(CM)
C
C CHARACTER*1 CM
C
      IF(CM.EQ.'0') THEN
        XMULT=253.
      ELSE IF(CM.EQ.'1') THEN
        XMULT=126.
      ELSE IF(CM.EQ.'2') THEN
        XMULT=84.
      ELSE IF(CM.EQ.'3') THEN
        XMULT=63.
      ELSE IF(CM.EQ.'4') THEN
        XMULT=50.
      ELSE IF(CM.EQ.'5') THEN
        XMULT=42.
      ELSE IF(CM.EQ.'6') THEN
        XMULT=36.
      ELSE IF(CM.EQ.'7') THEN
        XMULT=31.
      ELSE IF(CM.EQ.'8') THEN
        XMULT=28.
      ELSE IF(CM.EQ.'9') THEN
        XMULT=25.
      ELSE IF(CM.EQ.'A') THEN
        XMULT=23.
```

```

ELSE IF(CM.EQ.'B') THEN
  XMULT=21.
ELSE IF(CM.EQ.'C') THEN
  XMULT=19.
ELSE IF(CM.EQ.'D') THEN
  XMULT=18.
ELSE IF(CM.EQ.'E') THEN
  XMULT=16.
ELSE IF(CM.EQ.'F') THEN
  XMULT=15.
ELSE
  XMULT=0.
  WRITE(6,*) CM,' *NO MULTIPLIER CALCULATED, XMULT SET TO ZERO'
END IF

```

```

C
C*** END OF FUNCTION XMULT
C

```

```

RETURN
END

```

SUBROUTINE CNVTMP

```

CHARACTER*4 ANLG(8)
CHARACTER*6 FRQ(8)

```

```

COMMON / RAW / ANLG,FRQ
COMMON / CNVRT / COND,SAL78,TEMP,VOLT,FREQ,XL4,XL8,PSI,
1      DEPTH,TRANS,TLIGHT,CHLA

```

```

SEA BIRD CALIBRATION DATA FOR SENSOR #632

```

```

DATA A /3.67517928E-03/, B /6.01320919E-04/, .. ! SEABIRD SENSOR
1      C /1.59454020E-05/, D /2.59063611E-06/ ! #632
DATA FO /6092.84/, XK /273.15/

```

```

SEA BIRD CALIBRATION DATA FOR SENSOR #638 -- 9-18-87 CAL.

```

```

DATA A /3.67399029E-03/, B /6.01229226E-04/, ! SEABIRD SENSOR
1      C /1.51578463E-05/, D /2.71329547E-06/ ! #638
DATA FO /6337.28/, XK /273.15/

```

```

CALL HEXDEC(FRQ(1))

```

```

IF(FREQ.NE.0.) THEN
  DIV=ALOG(FO/FREQ)
  T1=A+B*DIV
  T2=C*DIV**2
  T3=D*DIV**3
  TEMP=1./(T1+T2+T3)-XK

```

```

ELSE
  TEMP=0.
END IF

```

```

C
C*** END OF SUBROUTINE CNVTMP
C

```

```

RETURN
END

```

SUBROUTINE CNVCND

CHARACTER*6 FRQ(8)
 CHARACTER*4 ANLG(8)
 REAL IEXP

COMMON / RAW / ANLG,FRQ
 COMMON / CNVRT / COND,SAL78,TEMP,VOLT,FREQ,XL4,XL8,PSI,
 DEPTH,TRANS,TLIGHT,CHLA

SEA BIRD CALIBRATION DATA SENSOR #267

DATA W /4.43054668E-10/, X /5.20135966E-01/, ! SEA BIRD SENSOR
 Y /-4.36199682/, Z /2.42254149E-04/ ! SENSOR #267
 DATA IEXP /8.4/

SEA BIRD CALIBRATION DATA SENSOR #234 -- 9-18-87 CAL.

DATA W /8.88229858E-09/, X /5.1687217E-01/, ! SEA BIRD SENSOR
 Y /-4.37109279/, Z /-4.80212277E-05/ ! SENSOR #234
 DATA IEXP /7.2/

CALL HEXDEC(FRQ(2))

F=FREQ/1000.
 C1=W*F**IEXP
 C2=X*F**2
 C3=Y
 C4=Z*TEMP
 COND=C1+C2+C3+C4

CALL SALNTY

END OF SUBROUTINE CNVCND

RETURN
 END

SUBROUTINE SALNTY

COMMON / CNVRT / COND,SAL78,TEMP,VOLT,FREQ,XL4,XL8,PSI,
 DEPTH,TRANS,TLIGHT,CHLA

T=TEMP
 P=2. ! HARD CODED FISH DEPTH FOR SAL CALC.
 SAL78=0.

IF(COND.GE.0.0005) THEN
 DT=T-15.
 R=COND/42.914
 RT35=(((.10031E-08*T-.69698E-06)*T+.0001104259)*T
 +.200564E-01) * T+.6766097
 RT=R/(RT35*(1.+FNC(P)/(FNB(T)+FNA(T)*R)))
 RT=SQRT(ABS(RT))

```

XR=RT
XT=DT
S1=((((2.7081*XR-7.0261)*XR+14.0941)*XR+25.3851)*XR-.1692)*XR
S2=.008
S3=XT/(1+.0162*XT)
S4=((((-0.0144*XR+.0636)*XR-.0375)*XR-.0066)*XR-.0056)
1      *XR+.0005
SAL78=S1+S2+S3*S4
END IF

```

```

C
C*** END OF SUBROUTINE SALNTY
C

```

```

RETURN
END

```

SUBROUTINE CNVPSI

```

CHARACTER*4 ANLG(8)
CHARACTER*6 FRQ(8)

```

```

COMMON / RAW / ANLG,FRQ
COMMON / CNVRT / COND,SAL78,TEMP,VOLT,FREQ,X14,XL8,PSI,
1      DEPTH,TRANS,TLIGHT,CHLA

```

```

1      DATA C /-57083.99/, D /0.03636303/, TO /27.597905/,
1      G /6.894759E7/, H /10.0E-6/, XLAT /36.0/

```

```

CALL HEXDEC(FRQ(3))
F=FREQ

```

```

PSI=G*(C*(1.-(H*TO*F)**2))*(1.-D*(1.-(H*TO*F)**2))

```

```

X=SIN(XLAT/57.29578)
X=X**2

```

```

GRAVITY=9.780318*(1.+(5.2788E-3+2.36E-5*X)*X)+1.092E-6*PSI
DEPTH=(((-1.82E-15*PSI+2.279E-10)*PSI-2.2512E-5)*PSI+9.72659)*PSI
DEPTH=DEPTH/GRAVITY

```

```

RETURN
END

```

FUNCTION FNC(XP)

```

FNC=((0.3989E-14*XP-.637E-09)*XP+.207E-04)*XP

```

```

C
C*** END OF FUNCTION FNC
C

```

```

RETURN
END

```

FUNCTION FNB(XT)

FNB=(.4464E-03*XT+.03426)*XT+1.

END OF FUNCTION FNB

RETURN

END

FUNCTION FNA(XT)

FNA=-.003107*XT+.4215

END OF FUNCTION FNA

RETURN

END

SUBROUTINE CNVVL(T,AN)

CHARACTER*4 O,AN

CHARACTER*1 NUM

INTEGER*2 G

COMMON / CNVRT / COND,SAL78,TEMP,VOLT,FREQ,XL4,XL8,PSI,
DEPTH,TRANS,TLIGHT,CHLA

REFERENCE VOLTAGE -- VREF -- FOR TUPS COMPUTER

DATA VREF /1.23536/ ! ONE COMPUTER OLD

DATA VREF /1.233/ ! OTHER COMPUTER NEW -- USED 6/88

O = AN

LENGTH=4

XN=0.

P=1.

DO 100 I=LENGTH,2,-1

G = ICHAR(O(I:I))

IF(G.GT.64) THEN

D=FLOAT(G)-55.

ELSE

NUM = AN(I:I)

IF (NUM.EQ.'0') D = 0.

! ADDED BY ARNONE 2-5-1988

IF (NUM.EQ.'1') D = 1.

IF (NUM.EQ.'2') D = 2.

IF (NUM.EQ.'3') D = 3.

IF (NUM.EQ.'4') D = 4.

IF (NUM.EQ.'5') D = 5.

IF (NUM.EQ.'6') D = 6.

IF (NUM.EQ.'7') D = 7.

IF (NUM.EQ.'8') D = 8.

IF (NUM.EQ.'9') D = 9.

END IF

XN=XN+D*P

```

P=P*16.
100 CONTINUE
C
VOLT=XN*VREF/2048.
IF (AN(1:1).EQ.'C') VOLT=-VOLT
C
C*** END OF SUBROUTINE CNVVL
C
RETURN
END

SUBROUTINE DIVIDE(RRAW)
C
C****
C****
C
SUBROUTINE DIVIDE SEPARATES THE INPUT STRING FROM TUPS
C INTO FREQUENCY AND ANALOG CHANNEL OUTPUT
C
C VARIABLES:
C
C FRQ - ARRAY TO HOLD FREQUENCY DEFINING STRINGS
C ANLG - ARRAY TO HOLD ANALOG DEFINING STRINGS
C
C CALLING PROGRAM(S) :
C
C PROGRAM CNVRT
C
C****
C****
C
CHARACTER*(*) RRAW
CHARACTER*4 ANLG(8)
CHARACTER*6 FRQ(8)
CHARACTER*80 OUTSTR(12)
LOGICAL*1 ILOG(1600)
EQUIVALENCE (ILOG(1),OUTSTR)
INTEGER LSTR(20)

COMMON / RAW / ANLG,FRQ

CALL PARSE(RRAW,OUTSTR,NSTR)

NFRQ = 0
NANAL= 0
DO I = 1,NSTR
    LSTR(I) = INDEX(OUTSTR(I),' ')-1
    IF(LSTR(I) .GT. 1) THEN
        IF(I.GT.1 .AND. LSTR(I-1) .EQ. 1 )THEN
            NFRQ = NFRQ + 1
            FRQ(NFRQ) = OUTSTR(I-1)(1:1)/// ' '//OUTSTR(I)(1:4)
        ELSE
            NANAL = NANAL + 1
            ANLG(NANAL) = OUTSTR(I)(1:4)
        END IF
    END IF
END DO

```

CCCCCCCCCCCCCCCC

22

52

CC

1

—

References

—

C

—

1

1

```

                                GOTO 1000
                                END IF
                                END DO
1000 CONTINUE
    IF(BLANK) IBLANK=0
    IF(IBLANK.EQ. 0 .AND. ICOMMA.EQ. 0) THEN
        MORE = .FALSE.
    ELSE
        COMMA = .FALSE.
        DPTR = MIN(IBLANK,ICOMMA)
        IF(IBLANK.EQ.0) DPTR =ICOMMA
        IF(ICOMMA.EQ.0) DPTR =IBLANK
        IF(ICOMMA.EQ. DPTR) COMMA = .TRUE.
        N = N + 1
        IF(COMMA .AND. DPTR.EQ. 1) THEN
            OUTSTR(N) = ' '
            PTR=PTR+1
        ELSE
            OUTSTR(N)=INSTRING(PTR+1:PTR+DPTR-1)

            IF(IQUOTE.NE.0) THEN
                OUTSTR(N)=INSTRING(PTR+1+IQUOTE:PTR+NQUOTE-1)
            END IF

            TYPE *,N,IBLANK,ICOMMA,OUTSTR(N)
            PTR=PTR+DPTR
            IF(OUTSTR(N).EQ. ' ') N=N-1
        END IF
    END IF
END DO
RETURN
END

!
subroutine timefix(time,timea,n)
real*8 time(n),timea(n)
real*8 dt
inum=n
do i=1,15
    type *,i,time(i)
    if(time(i).eq.time(i+1) ) then
    else
        kks=i+1
        goto 30
    endif
enddo
30 continue
do ii=1,15
    i=inum-ii+1
    type *,i,time(i)
    if(time(i).eq.time(i-1)) then
    else
        kke=i
        goto 31
    endif
enddo
31 nt = kke - kks

dt= (time(kke) - time(kks))/float(nt)
type *,nt,dt
timea(kks)=time(kks)
do i=kks-1,1,-1
    timea(i)=timea(i+1)-dt

```

```

enddo
do i=kks+1,inum
    timea(i)=timea(i-1)+dt
enddo
return
end

```

```

subroutine sec to hms(ts,ih,im,is)
implicit real*8 (t)
    ih=ts/3600.
    tl=ts-ih*3600.
    im=tl/60.
    tl=tl-im*60.
    is=tl
return
end

```

```

C*****
C
C PROGRAM      READJUN
C
C PURPOSE      1) program readjun to deal with tess files
C              2) place tess file in appropriate structure
C
C AUTHOR       NORMAN GUINASSO
C
C DATE         JUNE 88
C
C REVISION     DEC. 88
C
C*****
C
implicit real*8 (d)
real*8 ttime,flatd,flatm,flats,flond,flom,flons
real*8 d60 /60.d0/
REAL*4 V(11),spd

CHARACTER*21 DATELINE
CHARACTER*80 dataline
character*80 naviline
character*40 logname
character*20 cdate

!
type '(' Program to read and translate TESS files  ' ')'
type '(' Enter input file name => ',,$)'
accept '(a)',logname
cdate=' ' ! put date and time into string cdate
call date(cdate)
call time(cdate(12:))
! open files
open(9,name=logname,readonly,status='old',err=999)
call chg_ext(logname,logname,'rdj')
open(11,carriagecontrol='list',status='new',name=logname,recl=255)
type '(' Output file will be called ',a)',logname

!-----
write(11,'(' TESS data processed by program r djun on ',a)' cdate
write(11,'(' TESS data are 2.5VREF, 441 NM, 488 NM, PYR01,PYR02 --
* FILENAME = ',a)' logname
icnt=0
do while (.true.)
    read(9,'(a)',iostat=ios) dateline
    if(ios.eq.-1) goto 100 ! if end of file
    icnt=icnt+1
    j=str$translate(dateline,dateline,' ','-:')
    read(dateline,*,iostat=jos) mon,iday,iyear,ih,im,is
    if(jos.ne.0) type *, ' read error on dateline string ',jos
    if(iyear.gt.1900) iyear=iyear-1900 ! limit to 20th century
    ttime=d60*d60*(ih +(is/d60 +im)/d60)! seconds since midnight

    if(mod(icnt,50).eq.1) type '(' '+',i5,2X)',icnt! prevent boredom

    read(9,'(a)',iostat=ios) dataline
    read(9,'(a)',iostat=ios) naviline
    j=str$translate(naviline,naviline,' ','NW') ! eliminate NW
    read(naviline,*,iostat=jos) flatd,flatm,flats,flond,flonm,flons,
    * ihdg,spd,td1,td2 ! get the numbers
    if(jos.eq.-1) goto 110 ! if they are not here, stop
    if(jos.ne.0) type *, ' error reading naviline ',jos! signal

```

```

      dlat= flatd +(flats/d60 + flatm)/d60      ! decimal degrees
      dlon= flond +(flons/d60 + flonm)/d60      ! ditto
      dlon= -dlon                                ! occident
      j=str$trim(dateline,dateline,len)         ! remove trailing blanks

C      ROUTINE TO CONVERT VOLTS IN THE DATA LINE TO ENGINEERING UNITS
      READ(DATALINE,*,IOSTAT=KOS)(V(K),K=1,11)
      if(KOS.eq.-1) goto 115 ! if they are not there, stop
      CALL CALCENGR(V)

      write(11,300) iyear,mon,iday,ttime,ih,im,is, ! record for
      *      dlat,dlon,td1,td2,ihdg,spd,V(2),(V(K),K=5,8) ! posterity
      enddo
100    type *,icnt, ' records'
      stop 'end of file'
110    stop 'unexpected end of file reading dataline'
115    stop 'unexpected end of file reading dataline FOR CALCULATION'
120    stop 'unexpected end of file reading naviline'
999    stop 'bad inputfile'
300    * format(i2.2,'/',i2.2,'/',i2.2,1x,f8.1,2x,i2.2,':',i2.2,':',i2.2,1x,
      f11.6,f12.6,2f9.2,2x,i3.3,F5.1,5F7.3)
      end

      SUBROUTINE CALCENGR(V)
      REAL*4 V(11)
C      SUBROUTINE TO CONVERT TESS VOLTAGES TO ENGINEERING UNITS
C      THE VOLTAGE ARRAY "V" IS PASSED BACK AND FORTH

C      V(5) IS 441 LOG LIGHT SENSOR SN: 7102
C      UNITS ARE uW/cm2 sec nm
      V(5)=3.54*EXP(V(5)-0.734)

C      V(6) IS 488 LINEAR LIGHT SENSOR SN: 7105
C      UNITS ARE uW/cm2 sec nm
      V(6)=V(6)/0.25599

C      V(7) IS PYR01 -- THE LIGHT BULB PYROHELIOMETER..
C      UNITS ARE uW/cm2*10^4
      V(7)=(V(7)-0.0)/0.640 ! CONVERT VOLTS TO MILLIVOLTS
      V(7)=V(7)*(0.1202/60.0)*4.184*100.0 ! CALIBRATION AND CONVERSION

C      V(8) IS PYR02 -- THE HEMISPHERE PYROHELIOMETER
C      UNITS ARE uW/cm2*10^4
      V(8)=(V(8)-0.0)/0.9719 ! CONVERT VOLTS TO MILLIVOLTS
      V(8)=V(8)*(0.2232/60.0)*4.184*100.0 ! CALIBRATION AND CONVERSION

C** END OF SUBROUTINE CALCENGR
      RETURN
      END

```

```

C*****
C
C PROGRAM      PRETABLE
C
C PURPOSE      1) reads in tups "cnv" file
C              2) filters some data and writes a binary file
C
C AUTHOR       NORMAN GUINASSO
C
C DATE         JUNE 88
C
C REVISION     DEC. 88
C
C*****
C
      parameter (nn=5000)
      common /fopen name/ f name
      character*60 f_name,filename,outfile
!  datatups variables:
      equivalence (datatups(1, 1), temp(1))
      equivalence (datatups(1, 2), salin(1))
      equivalence (datatups(1, 3), ptrans(1))
      equivalence (datatups(1, 4), fluor(1))
      equivalence (datatups(1, 5), d465nm(1))
      equivalence (datatups(1, 6), d507nm(1))
      equivalence (datatups(1, 7), d532nm(1))
      equivalence (datatups(1, 8), transv(1))
      equivalence (datatups(1, 9), flsigv(1))
      equivalence (datatups(1,10), flsclv(1))
      equivalence (datatups(1,11), d488up(1))
      equivalence (datatups(1,12), dep(1))
!
      real*8 datatups(nn,12)
      real*8 temp(nn),salin(nn),ptrans(nn),transv(nn)
      real*8 fluor(nn),flsigv(nn),flsclv(nn)
      real*8 d465nm(nn),d507nm(nn),d532nm(nn),d488up(nn)
      real*8 dep(nn)
!-----
      real*8 rawdepth(nn)
      real*8 rawfluor(nn)
      real*8 ddint(11)
      real*8 data(nn,11)
      real*8 ttime(nn)
      real*8 out(nn)
      character*150 in
      character*8 ctime,timestr
      character ans
      character*30 timedate,filedate
!-----
      call date(timedate(11:))
      call time(timedate(21:))
      timedate(1:10)= 'Processed'

      type *, ' Preprocessor for TUPS file '
      type *, ' This program filters fluorescence, % trans, depth, salinity'
      type *, ' and writes a binary file'
      call fopenread(10)
      type *, ' '
      i=0
      do while(.true.)
         read(10,'(a)',iostat=ios) in
         if(ios.eq.-1) goto 290

```

```

i=i+1
if(mod(i,100).eq.1) type (('+.',$)
j=str$translate(in,in,' ',':/|')
j=str$trim (in,in,len)
read(in(1:len),*,iostat=ios)
*
*
iyr,imo,iday,ttime(i),ihr,imin,isec,
(datatups(i,j),j=1,12)
if(i.gt.1) then
if( ttime(i) .lt. ttime(i-1) ) then
i=i-1
type *, ' bad time at ', i,ttime(i-1)
type *,in(1:70)
endif
endif

if(ios.ne.0) then
i=i-1
type *,ios,' ',in(1:66)
else
endif

enddo
290 continue
ncnt=i
do i=1,ncnt
rawdepth(i)=dep(i)
rawfluor(i)=fluor(i)
enddo
!
104 write(filedate,104) iyr,imo,iday
format('filedate ',i2,'/',i2.2,'/',i2.2)
!-----
type *,ncnt
type *, ' salinity '
call rfilter(salin,ncnt, .03, 2.1 , 30., 37.)
pause
type *, ' percent trans'
call rfilter(ptrans,ncnt, .1, 6. , 60., 100.)
pause
type *, ' depth '
call rfilter(dep,ncnt, .3, 5. , 0., 30.)
pause
!-----
type *, 'fluorescence'
f=.5
fmin=100.
fmax=850.
call rfilter ( fluor, ncnt, f, 65., fmin, fmax )
!-----
!
!
d
d
d
d
#
open(15,name='plotfluor.out',status='new',carriagecontrol='list')
write(15,('f7.3,3f7.1'))
(ttime(i)/3600.,rawfluor(i),rawdepth(i),fluor(i),i=1,ncnt)
close(15)
call chg_ext(f_name,outfile,'pto')

*
open(unit=14,name=outfile,status='new',iostat=jos,
form='unformatted',carriagecontrol='none')
type *,jos,' '//outfile
write (14) ttime(i),filedate,ncnt
type *, ' ncnt ',ncnt
do i=1,ncnt
write(14) ttime(i),(datatups(i,j),j=1,12)
enddo
close(14)

```

```

100      stop
101      format('+.', $)
102      format(6i3.2, 1x, a, 1x, f8.1, 3(/, 8f10.2))
103      format(f9.4, 2f8.1)
      end

!
      subroutine rfilter(d, n, f, windowmin, fmin, fmax)
! windowing filter
! if new point differs by previous point times f then old point is retained
! also if point out of range fmin, fmax than previous point is retained
! guinasso, circa 1988
      real*8 f, windowmin, fmin, fmax
      real*8 d(n)
      real*4 ave, sdev
      type *, ' rfilter-i-f= ', f, windowmin
      type *, ' rfilter-i-fmin, fmax= ', fmin, fmax
      call stat(d, n, ave, sdev)
      ichanged = 0
      tm1=0
      tm2=0
      do i=2, n
          tm3=tm2
          tm2=tm1
          tm1=t
          t=d(i)
          prev = d(i-1)
          if(prev.lt.fmin .or. prev.gt.fmax) then
              prev = ave
              type *, ' prev replaced with average = ', ave
          endif
          delta = abs(d(i) - prev)
          window = f * prev
          if(window.lt.windowmin) window=windowmin
          if(delta.gt.window .or. d(i).lt.fmin .or. d(i).gt.fmax) then
              d(i) = prev
              TYPE '(1x, 2f8.2, 4f8.1,
                  ' changed ', i5, ' to ', f8.1)',
                  window, delta, tm3, tm2, tm1, t, i, d(i)
              ichanged = ichanged + 1
          else
              endif
      enddo
      return
      end

!
      subroutine table_lookup(x, y, n, nn, m, xx, yy, reset)
! -----
! linear interpolation lookup program
! guinasso circa 1988
! -----
!
! n      dimension of x, y
! nn     number actually used
      real*8 x(n)      ! array of x's must be monotonically increasing
      real*8 y(n, m)    ! array of y's
      real*8 xx         ! x value at point to look up
      real*8 yy(m)      ! y's at xx (returned)
      real*8 f          ! interpolation fraction
      integer*4 il /1/  ! start of interpolation range
      logical reset     ! will reset range looked at to 1 thru n

!
      if (reset) il=1
      if(xx.lt. x(1) ) goto 10      ! not on table
      do i=il, n-1
          if(x(i).ge.x(i+1)) then

```

```

                                stop 'table lookup-f-bad table, not monotonic'
                                else if(xx.ge.x(i) .and.xx.le.x(i+1)) then
                                    n1=i
                                    n2=i+1
                                    i1=n1
                                    goto 20
                                endif
                                enddo
10                                continue
                                type *, 'table lookup-i-xx not on table '
                                type *, xx,x(i),x(n)
                                return
!
20                                f = (xx-x(n1) )/( x(n2)-x(n1) )
                                wf = 1.d0-f
                                do j=1,m
                                    yy(j) = wf*y(n1,j) + f*y(n2,j)
                                enddo
                                return
                                end
!
                                character*8 function timestr(secs)
!
! converts seconds past midnight to HH:MM:SS
! guinasso circa 1988
!
                                ih=secs/3600
                                im=(secs - ih * 3600.)/60.
                                is=nint(secs -ih * 3600. - im * 60.)
                                write (timestr,'(i2.2,':'',i2.2,':'',i2.2)' ) ih,im,is
                                return
                                end
!
                                subroutine stat(y,n,aver,sdev)
! calculates mean and standard deviation of array y
                                real*8 y(n)
                                sum=0.
                                do i=1,n
                                    sum=sum+y(i)
                                enddo
                                aver = sum/float(n)
                                sum=0.
                                do i=1,n
                                    sum=sum+ (aver-y(i))**2
                                enddo
                                var = sum/float(n-1)
                                sdev = sqrt(var)
                                type *, 'stat-i-mean, standard deviation = ', aver,sdev
                                return
                                end
!

```

```

C*****
C
C PROGRAM      MERGETT
C
C PURPOSE      1) merge norda tess tups files
C
C AUTHOR       NORMAN GUINASSO
C
C DATE        JUNE 88
C
C             DEC. 88
C
C SUBROUTINES
C
C
C
C*****
C
! Geocemical and Environmental Resaearch Group
! Texas A&M University
!XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
!
!   parameter (nn=5000)
!   parameter (nnn=25000)
!
!   common /fopen name/ f_name
!   character*60 f_name,outfile,infile
!   character*150 in
!   real*8 ddint(11)
!   real*8 data(nn,11)
!   real*8 datatups(nnn,12)
!   real*8 attime(nnn)
!   real*8 time(nn),ttime
!   character*8 ctime,timestr
!   character*30 timedate,filedate
!
!   -----
!   call lib$init_timer()
!   type *, ' tess file '
!   call fopenread(9)
!
! for tups file
!   call chg_ext(f_name,infile,'pto')
!   type *, ' Opening TUPS pretable output file ',infile
!   open (14, name=infile,form='unformatted',status='old',iostat=jos)
!   if(jos.ne.0) stop 'file not found'
!
! output file
!   call chg_ext(f_name,outfile,'fin')
!   type *, ' Output file will be called ',outfile
!   type *, ' '
! for tess file
!   do while(.true.)
!     read(9,'(a)',iostat=ios) in
!     if(ios.eq.-1) goto 90
!     i=i+1
!     j=str$translate(in,in,' ','/:')
!     read(in,*,iostat=ios)
!     *   iyr,imo,iday,time(i),ih,im,is,(data(i,j),j=1,11)
!     if(ios.ne.0) then
!       type *,in(1:76)
!       i=i-1
!     endif
!     if(mod(i,100).eq.1) type 100
!   enddo
! enddo

```

```

90      nctess=i
        close(9)
        type *, nctess, ' records in tess file '
        type *,time(1),time(nctess), ' are start stop times'
        call lib$show_timer ( )
        read (14) timedate,filedate,nctup
        type *, nctup,' ',timedate,filedate
        do i=1,nctup
            k = i
            read(14,end=92) attime(i),(datatups(i,j),j=1,12)
        enddo
        goto 93
92      type *,' end of tups file ',k,nctup
        if(k.ne.nctup) type *, 'bad tups file'
93      close(14)
        call lib$show_timer()

!
!
        type *,' starting output phase'
        type *,' '
        open(15,name=outfile,carriagecontrol='list',status='new',
*         recl=256)
        iout=0
        do i=1,nctup
            if(mod(i,100).eq.0) type 100
            ttime = attime(i)
            call table_lookup(time,data,nn,nctess,11,
*             ttime,ddint,.false.)
            ctime=timestr(ttime)
            iout=iout+1
            write(15,101)iy,imo,iday,ctime,
*             ttime,ddint,(datatups(i,j),j=1,12)
        enddo
        call lib$show_timer()
        stop
100      format('+.',$)
101      format(3i3.2,1x,a,1x,f8.1,f10.6,f11.6,2f10.2,f6.1,f5.2,
*       5f7.3, f6.2, f7.3, f6.2, f9.2, 7f7.3, f6.2)

! tess
! F lat,long,td1,td2,heading,speed,2.5 ref,data 1-4
! S temp, sal, ptrans, chlorfl, data 1-7, depth
! end

!
!      subroutine table_lookup (x,y,n,nn,m,xx,yy,reset)
!      -----
!      linear interpolation lookup program
!      guinasso circa 1988
!      -----
!      n      dimension of x,y
!      nn     number actually used
!      real*8 x(n)      ! array of x's
!      real*8 y(n,m)    ! array of y's
!      real*8 xx        ! x to look up
!      real*8 yy(m)     ! y's at xx
!      real*8 f         ! interpolation fraction
!      integer*4 il /1/ ! start of interpolation range
!      logical reset    ! will reset range looked at to 1 thru n

!
!      if (reset) il=1
!      if(xx.lt. x(1) .or. xx.gt.x(nn)) goto 10      ! not on table
!      do i=il,nn-1
!          if(x(i).ge.x(i+1)) then
!              type *, i,x(i),x(i+1)
!              stop 'table_lookup-f-bad table'

```

```

endif
  if(xx.ge.x(i) .and.xx.le.x(i+1)) then
    n1=i
    n2=i+1
    i1=n1
    goto 20
  endif
enddo
10 continue
  type *, 'table_lookup-i-xx not on table '
  type *, xx,x(i),x(nn)
return
!
20 f = (xx-x(n1) )/( x(n2)-x(n1) )
  wf = 1.d0-f
  do j=1,m
    yy(j) = wf*y(n1,j) + f*y(n2,j)
  enddo
  return
end
!
character*8 function timestr(secs)
! -----
! converts seconds past midnight to HH:MM:SS
! guinasso circa 1988
! -----
  ih=secs/3600
  im=(secs-ih*3600.)/60.
  is=nint(secs-ih*3600.-im*60.)
  write (timestr,'(i2.2,':'',i2.2,':'',i2.2)' ) ih,im,is
  return
end

```

C*****

```

C
C PROGRAM          TTPLOT
C
C PURPOSE          1) plots combined tess and tups data
C
C AUTHOR           NORMAN GUINASSO
C
C DATE             JUNE 88
C
C REVISION         DEC. 88
C
C

```

C*****

```

parameter(nn=5000)
common /paper/ xpaper,ypaper,LASER
common /xrange/ x1,x2
LOGICAL LASER /.FALSE./
real*4 time(nn)
real*4 flat(nn)
real*4 flon(nn)
real*4 hdg(nn)
real*4 speed(nn)
real*4 flitel(nn),flite2(nn),flite3(nn),flite4(nn)
real*4 temp(nn),salin(nn)
real*4 ptrans(nn)
real*4 fluor(nn)
real*4 depth(nn)
real*4 up1(nn),up2(nn),up3(nn),down1(nn)
character*4 itime
CHARACTER*300 IN
character*60 t1,t2,t3,t4,t5,t6,t7,t8
character*30 pfile
character*30 filename
character*2 cday
data z,sx /0.,42./

```

!- - - - -1234567890 - - - - -

```

filename='xxjun88.fin'
t1='Geochemical and Environmental Research Group'
t2='Texas A&M University'
t3='Ten South Graham Road'
t4='College Station, Texas 77840'
t5='Telephone: 409 690 0095'
t6='NORDA Continuous Underway Data'
t7=' '
t8='Department of Oceanography'
j=str$trim(t6,t6,len)
type '(' enter day of plot ##> ',,$)'
accept '(a)',cday
t7=cday//' June 1988'
t6=t6(1:len+1)//t7
type *,t6
type *,t7
k=0
filename(1:2)= cday
open(9,name=filename,status='old',iostat=ios)
type *, 'iost t =',ios,' '///filename
if(ios.ne.0) stop
DO WHILE(.TRUE.)
    READ(9,'(A)',END=50) IN
    k=k+1
    j=str$trim(in,in,ilen)
    if(mod(k,100).eq.1) type '('+'.'',,$)'
    j=str$translate(in,in,' ',':')

```

```

d      if(k.lt.10) then
d      type *, '$1'//in(1:150)
d      if(ilen.gt.150) type *, '$2'//in(151:ilen)
d      endif
d      read(in,*,iostat=ios) iyr,imo,iday,ihr,imin,isech,
*      time(k),flat(k),flon(k),td1,td2,hdg(k),speed(k),ref,
*      flitel(k),flite2(k),flite3(k),flite4(k),temp(k),salin(k),
*      ptrans(k),fluor(k),up1(k),up2(k),up3(k),d4,d5,d7,down1(k),depth(k)
d
d      type *,iyr,imo,iday,ihr,imin,isech,
*      time(k),flat(k),flon(k),td1,td2,hdg(k),speed(k),ref,
*      flitel(k),flite2(k),flite3(k),flite4(k),temp(k),salin(k),
*      ptrans(k),fluor(k),up1(k),up2(k),up3(k),d4,d5,d7,down1(k),depth(k)
d
d      if(k.gt.250) goto 60
d
d      enddo
50      continue
60      continue
!
pfile = 'ttplotxxZ.iop'
pfile(7:8)=cday
IF(LASER) THEN
    pfile(9:9)='L'
ELSE
    pfile(9:9)='P'
ENDIF
type *, 'starting plot ' // pfile
    xpaper=41.50
    ypaper=32.5
call iopen(20,pfile)
IF(    LASER)call set_42 (z,sx,z,sx,  z,sx,z,sx,1)
IF(.NOT.LASER)call set_60 (z,sx,z,sx,  z,sx,z,sx,1)
n=k
do i=1,n
    time(i)=time(i)/3600.
enddo
ix1 = time(1)
ix2 = time(n) + 1
x1=ix1
x2=ix2
type *, ' Date to be plotted from ',ix1,' to ', ix2,' hours '
call draw_box(z,z,xpaper,ypaper)
call nlgpwrchg(.2)
y=1.35
is=8
call pwritxx(1.,y,t1,is,0,-1)
y=y-.2
call pwritxx(1.,y,t2,is,0,-1)
y=y-.2
call pwritxx(1.,y,t8,is,0,-1)
y=y-.2
call pwritxx(1.,y,t3,is,0,-1)
y=y-.2
call pwritxx(1.,y,t4,is,0,-1)
y=y-.2
call pwritxx(1.,y,t5,is,0,-1)
call draw_box(z,z,xpaper,1.6)
call pwritxx(21.,.8,t6,30,0,0)
xp=2.
yp1=xp+1.5
call plotavec(xp,yp1,time,flat,n,'latitude')

```

```

xp=xp+1.6
yp1=xp+1.5
call plotavec(xp,yp1,time,flon,n,'longitude')
xp=xp+1.6
yp1=xp+1.5
call plotavec(xp,yp1,time,hdg,n,'heading')
xp=xp+1.6
yp1=xp+1.5
call plotavec(xp,yp1,time,speed,n,'speed')
xp=xp+1.6
yp1=xp+1.5
call plotavec(xp,yp1,time,flite1,n,'441nm(log)')
xp=xp+1.6
yp1=xp+1.5
call plotavec(xp,yp1,time,flite2,n,'488(linear)')
xp=xp+1.6
yp1=xp+1.5
call plotavec(xp,yp1,time,flite3,n,'pyro1-bulb')
xp=xp+1.6
yp1=xp+1.5
call plotavec(xp,yp1,time,flite4,n,'pyro2-hemi')
xp=xp+1.6
yp1=xp+1.5
call plotavec(xp,yp1,time,up1,n,'465nm upwelling')
xp=xp+1.6
yp1=xp+1.5
call plotavec(xp,yp1,time,up2,n,'507nm upwelling')
xp=xp+1.6
yp1=xp+1.5
call plotavec(xp,yp1,time,up3,n,'532nm upwelling')
xp=xp+1.6
yp1=xp+1.5
call plotavec(xp,yp1,time,down1,n,'488nm downwelling')
xp=xp+1.6
yp1=xp+1.5
call plotavec(xp,yp1,time,temp,n,'temperature')
xp=xp+1.6
yp1=xp+1.5
call plotavec(xp,yp1,time,salin,n,'salinity')
xp=xp+1.6
yp1=xp+1.5
call plotavec(xp,yp1,time,fluor,n,'fluoresence')
xp=xp+1.6
yp1=xp+1.5
call plotavec(xp,yp1,time,ptrans,n,'% transmission')
xp=xp+1.6
yp1=xp+1.5
call plotavec(xp,yp1,time,depth,n,'depth')

```

```

do i=ix1,ix2
    ii=i*100
    xx=i
    write(itime,'(i4.4)') ii
    call pwritxx( xx, yp1+.15,itime,36,0,0)
enddo
call frame
stop
end

```

```

subroutine plotavec(yp,ypt,x,y,n,title)
common /paper/ xpaper,ypaper,LASER
LOGICAL LASER ! true if ln03 -- false if hp7585b
common /xrange/ x1,x2
character *(*) title

```

```

real*4 x(n),y(n)
character*10 ranget,rangeb
CHARACTER*60 STRING

isi=36
isil=36
call minmax(y,n,fmin,fmax)
write(ranget,'(f9.2)') fmax
write(rangeb,'(f9.2)') fmin
STRING=TITLE
deltay=fmax-fmin
deltax=x2-x1
yuin =deltay/( ypt-yp )
xuin =deltax/(xpaper-2.)
plin = .2*yuin
x2in = 2.*xuin
x1l  = x1+.15*xuin
x2l  = x2-.15*xuin
type *,STRING
IF (LASER) then
    call set_42(1.,xpaper-1.,yp,ypt,x1,x2,fmin,fmax,1)
    isil=36
elseif (.NOT.LASER) THEN
    call set_60(1.,xpaper-1.,yp,ypt,x1,x2,fmin,fmax,1)
    isil=12
endif
call draw_box(x1, fmin, x2, fmax)
call pwritxx(x1l, fmin+plin , string , isil,0,-1)
call pwritxx(x2l, fmin+plin , rangeb , isil,0, 1)
call pwritxx(x2l, fmax-plin , ranget , isil,0, 1)
call curve(x,y,n)
return
end

subroutine curve(x,y,n)
parameter (hiatus = 20./3600.)
real*4 x(n),y(n)
dx=0.
do i=1,n
    if(i.ne.1) dx = x(i) - x(i-1)
    xx=x(i)
    yy=y(i)
    if(i.eq.1 .or. dx.gt.hiatus) then
        call frstpt(xx,yy)
    else
        call vector(xx,yy)
    endif
enddo
return
end

subroutine minmax(y,n,ymin,ymax)
real*4 y(n)
ymin=1.e30
ymax=-1.e30
do i=1,n
    if(y(i).gt. ymax) ymax=y(i)
    if(y(i).lt. ymin) ymin=y(i)
enddo
return
end

```

```

C*****
C
C PROGRAM          CONTOUR
C
C PURPOSE          1) PROGRAM TO CONTOUR PANAMA CITY TUPS/TESS DATA
C
C AUTHOR           NORMAN GUINASSO
C
C DATE             NOV. 88
C
C REVISION         DEC. 88
C
C*****
C
parameter (nx=80,ny=80)
parameter (n1=6000)
parameter (n2=n1*13)
parameter (n3=31*n1)
parameter (n4=14)
parameter (ires=25)
parameter (ires2=ires**2)

EQUIVALENCE(WK(1),WK1(1))
EQUIVALENCE(WK(1),WK2(1,1))
EQUIVALENCE(ADATA(1,1),DEPTH(1))
EQUIVALENCE(ADATA(1,2),TEMPER(1))
EQUIVALENCE(ADATA(1,3),SALIN(1))
EQUIVALENCE(ADATA(1,4),PTRANS(1))
EQUIVALENCE(ADATA(1,5),FLUOR(1))
EQUIVALENCE(ADATA(1,6),L465NM(1))
EQUIVALENCE(ADATA(1,7),L488NM(1))
EQUIVALENCE(ADATA(1,8),L507NM(1))
EQUIVALENCE(ADATA(1,9),L532NM(1))
EQUIVALENCE(ADATA(1,10),TESS441(1))
EQUIVALENCE(ADATA(1,11),TESS488(1))
EQUIVALENCE(ADATA(1,12),TESSPYR1(1))
EQUIVALENCE(ADATA(1,13),TESSPYR2(1))
EQUIVALENCE(ADATA(1,14),SECMID(1))

COMMON /CONRE1/ IOFFP,SPVAL
COMMON /CONRE4/ SIZEL,SIZEM,SIZEP,NREP,
1          NCRT,ILAB,NULBLL,IOFFD,
2          EXT,IOFFM,ISOLID,NLA,
3          NLM,XLT,YBT,SIDE

common /fopen name/ fname
character*60 fname
common /con_grid/ xmin,xmax,ymin,ymax,nnx,nyy
logical larea(nx,ny)

REAL*4 ADATA(N1,N4)
REAL*4 FLAT(n1),FLON(n1),DEPTH(n1)
real*4 temper(n1),salin(n1),ptrans(n1),fluor(n1)
real*4 L465NM(n1),L488NM(n1),L507NM(n1),L532NM(n1)
REAL*4 TESS441(N1),TESS488(N1),TESSPYR1(N1),TESSPYR2(N1)
real*4 array(3,n1),SECMID(N1)
real*4 wk1(20000),wk2(nx,ny)
real*4 wk(n2),iwk(n3),SCarr(ires2)
real*4 area(nx,ny)

logical aver

character*12 thedate /'XX June 1988'/
character*30 title(14)

```

```
CHARACTER*10 FMT
character*140 in
character*8 ctime
```

```
=====
!
conrec common block
xlt= .11          ! xoffset
side=.78          ! length of square
nrep=3
=====
```

```
nnx=nx
nny=ny
IOFFP=1
SPVAL=99999.
```

```
!--
title(1)='DEPTH'
TITLE(2)='TEMPERATURE'
title(3)='SALINITY'
TITLE(4)='PERCENT TRANSMISSION'
TITLE(5)='FLUORESCENCE'
TITLE(6)='TUPS 465NM'
TITLE(7)='TUPS 488NM'
TITLE(8)='TUPS 507NM'
TITLE(9)='TUPS 532NM'
TITLE(10)='TESS 441NM'
TITLE(11)='TESS 488NM'
TITLE(12)='TESS PYROMETER 1'
TITLE(13)='TESS PYROMETER 2'
TITLE(14)='SECONDS'
aver = .false.
xmin = 1000.
ymin = 1000
ymax = -1000.
xmax = -1000.
CALL FOPENREAD(9)
thedata(1:2)=fname(4:5)
type *, ' file date =', thedate
type '(', ' how many plots ( max 13) ? ', '$)'
accept *, nplots
I=0
```

```
DO WHILE (.TRUE.)
```

```
    M=I+1
    read(9, '(a)', iostat=jos) in
    if(jos.ne.0) goto 10
    call pulloff(in, ctime, len)
    READ(in, *, IOSTAT=IOS)
    *      flat(m), flon(m), depth(m),
    *      temper(M), salin(M), ptrans(M), fluor(M),
    *      1465nm(m), 1488nm(m), 1507nm(m), 1532nm(m),
    *      tess441(m), tess488(m), tesspyr1(m), tesspyr2(m)
```

```
    if(ios.eq.0) then
        i=i+1
    else
        type *, 'bad line'
    endif
```

```
10  enddo
    continue
    k=0
    npts=i
    nret=0
    type *, npts, n1, n4, ' CALLING SAMEPTS '
    call samepts(flat, flon, ADATA, npts, N1, N4, Nret)
    npts=nret
    type *, npts, ' points left after samepts'
```

```

do i=1,npts ! fill lat lon arrays
    array(1,i)=flat(i)
    array(2,i)=flon(i)
    call xyminmaxs(flon(i),flat(i),xmin,xmax,ymin,ymax)
enddo
type *, ' lon min max ',xmin,xmax
type *, ' lat min max ',ymin,ymax
call mapmin(xmin,xmax,ymin,ymax)
type *, ' lon min max ',xmin,xmax
type *, ' lat min max ',ymin,ymax
ioptn=1      ! 1= mask on 0= mask off
ireach=5
call mask_init (.false.,larea)

call mask_conrec(flat,flon,npts,ireach,ioptn,larea)

d call mask_show(larea)

call pcity_land(0,larea)

d call mask_show(larea)

call lib$init_timer()

DO KN=1,nplots
    ! set up array for weaver:
    do i=1,NPTS
        array(3,i)=adata(i,KN)
    enddo

    dx=xmax-xmin
    dy=ymax-ymin
    dxy=dx

    ! weaver parameters
    dist= .075*dxy ! measure of distance to fill ( dist must be <.5*dxy )
    ilr      =0 ! wrap left right
    ibt      =0 ! wrap bottom top
    iweav    =1 ! 0=weave lr first; 1=weave bt first
    idec     =2 ! decimal places
    ifine    =1 ! number of times through emphasize
    centr    =1. ! weight of center
    ichek    =0 ! returned error parameter
    wxmin    = xmin
    wxmax    = xmax
    wymin    = ymin
    wymax    = ymax
    nxw=nx
    nyw=ny

    type *, ' dist, dx, xy = ',dist,dx,dy
    type *, ' nx,ny ',nx,ny
    type *, ' min max ',wxmin,wxmax,wymin,wymax

    call weaver(array,npts,area,wk2,nxw,nyw,
        * wxmin,wxmax,wymin,wymax,dist,
        * ilr,ibt,iweav,idec,ifine,centr,ichek)

    type *, 'after weaver ichek = ',ichek

    call mask_apply(area,larea)

    call conrec(area,nx,nx,ny,0.,0.,0.,0,-1,0)

```

```

j=trim(title(kn),title(kn),len)
type *,title(kn)(1:len)
z0=0.
z1 = xlt
z2 = xlt+ side
z3 = .05 +side
zw=1.
call set (z1,z2,.05,z3,xmin,xmax,ymin,ymax,1)
call dashd('$$$$$',12,5,1)
call curvedh(flon,flat,npts)
call curve(flon,flat,npts)
call pcity_land(1,larea)
call set (z1,z2,z1,z2, z1,z2,z1,z2,1)
call pwritxx(.14,.78,title(kn),12,0,-1)
call pwritxx(.14,.75,theDate,12,0,-1)
iz=10
call map_write_latlon(.5, .05-.02 ,iz,0,0,ymin,'lat')
call map_write_latlon(.5, z3+.02 ,iz,0,0,ymax,'lat')
call map_write_latlon(z1-.02, .5 ,iz,90,0,xmin,'lon')
call map_write_latlon(z2+.02, .5 ,iz,90,0,xmax,'lon')
call frame
call lib$show_timer()
ENDDO
stop
end

```

```

=====
subroutine curvedh(x,y,n)
real*4 x(n),y(n)
logical penup
penup=.true.
thresh = .5/60. ! approx .5 NM
call frstd(x(1),y(1))
do i=2,n
    dx = x(i)-x(i-1)
    dy = y(i)-y(i-1)
    dx =dx*cosd(y(i))
    di = sqrt (dx**2 + dy**2)
    if (di.gt. thresh) then
        type *, 'FRSTD n= ',i
        call lastd
        call frstd(x(i),y(i))
        penup = .false.
    else
        call vectd(x(i),y(i))
        penup = .true.
    endif
enddo
call lastd
return
end

```

```

=====
subroutine mask_apply(area,larea)
! applys conrec mask to weaver output file
! guinasso, circa 1988
common /conrel/ ioffp,spval
common /con_grid/ xmin,xmax,ymin,ymax,nx,ny
logical larea(nx,ny)
real*4 area(nx,ny)
character*50 line

!
type *, ' mask_apply ',nx,ny
do i=1,NX
do j=1,NY
    if(.not.larea(i,j)) area(i,j)=spval

```

```

enddo
enddo
return
end

```

```

=====
subroutine mask init(linit,larea)
common /con_grid/ xmin,xmax,ymin,ymax,nx,ny
logical larea(nx,ny)
logical linit
do j=1,ny
do i=1,nx
    larea(i,j)=linit
enddo
enddo
return
end

```

```

subroutine mask show(larea)
common /con_grid/ xmin,xmax,ymin,ymax,nx,ny
logical larea(nx,ny)
z1 = .05
z2 = .95
dx = (xmax-xmin)/float(nx-1)
dy = (ymax-ymin)/float(ny-1)
call set(z1,z2,z1,z2,xmin,xmax,ymin,ymax,1)
do j=1,ny
    ynode=ymin+(j-1)*dy
do i=1,nx
    xnode=xmin+(i-1)*dx
    if(larea(i,j)) call ticm(xnode,ynode,dx/15.)
enddo
enddo
call frame
end

```

```

subroutine mask conrec(flat,flon,npts,ireach,ioptn,larea)
common /con_grid/ xmin,xmax,ymin,ymax,nx,ny
logical larea(nx,ny)

```

```

! flat flon defines a track
! ioptn= 1 turns on grid pts near track
! ioptn= 0 turns off grid points near track
! xmin,xmax,nx ymin,ymax,ny defines a grid nx by ny
! lflag(i,j) is true if point i,j is within dist1 of any point on track
! this defines a a mask that can be applied to a data grid passed to conrec
! guinasso, circa 1988

```

```

real*4 flat(npts),flon(npts)
logical mask1,mask2
dist(x,y)= (x**2 + y**2)

```

```

if(ioptn.eq.1) then
    mask1= .true.
    inc= 1
else
    mask1= .false.
    inc = -1
endif

```

```

itot=nx*ny
tot=itot
dx=(xmax-xmin)/float(nx-1)
dy=(ymax-ymin)/float(ny-1)
dis = min(dx,dy)
dist1 = (ireach*dis)**2

```

```

TYPE *, 'ENTERING MASK CONREC with ioptn = ', ioptn, inc
type '(', ' dx,dy dist1=', '3f12.4)', ' dx,dy,dist1
type *, ' '
ion=0
do j=1,ny
    ynode=ymin+(j-1)*dy
do i=1,nx
    xnode=xmin+(i-1)*dx
    if(mask1.eq.larea(i,j)) then
    else
        do k=1,npts
            dely = flat(k)-ynode
            delx = flon(k)-xnode
            dist2 = dist(delx,dely)
            if(dist2 .lt. dist1) then
                larea(i,j) = mask1
                ion = ion + inc
                goto 5
            else
            endif
        enddo
    endif
enddo
5 continue
fion=100.*ion/tot
if(mod(i,ny).eq.0 .and. mod(j,10).eq.0) type 109,j,fion
enddo
enddo
type *, ' Mask conrec-i- cells changed, total cells= ', ion, itot
return
109 format('+ j, percent switched = ', i5, f6.1, ' ')
end
#####
subroutine samepts(flat,flon,ADATA,n,nd1,nd2,nret)
! removes data with duplicate locations
! N = NUMBER OF LINES
! MD = DIMENSION OF ADATA
! ND = NUMBER OF DATA VARIABLES
! guinasso, circa 1988
real*4 flat(n),flon(n),ADATA(nd1,nd2)
logical skip(6000)
ireject=0
do i=2,n
    do j=1,i-1
        if(flat(i).eq.flat(j) .and. flon(i).eq.flon(j)) then
            ireject = ireject+1
            skip(i) = .true.
            goto 10
        else
            skip(i)=.false.
        endif
    enddo
enddo
10 k=0
do i=1,n
    IF(SKIP(I)) THEN
    else
        flat(k+1)=flat(i)
        flon(k+1)=flon(i)
        DO KD=1,ND2
            ADATA(k+1,KD)=ADATA(i,KD)
        ENDDO
        k=k+1
        nret=k
    endif
endif

```

```

enddo
type *, ' total, returned, rejected = ', n, nret, ireject
return
end

!
subroutine pcity_land(iflag, larea)
common /con_grid/ xmin, xmax, ymin, ymax, nx, ny
logical larea(nx, ny)
real*4 flat(500), flon(500)

!
open(19, name='pcitymap.f10', status='old', iostat=kos)

if(kos.ne.0) stop 'pcitymap.f10 not found'
ipold=0      ! last line #
npts=0
nlines=0     ! lines plotted
do while (.true.)
    read(19, *, iostat=kos) num, ip, x, y, alat, alon
    alon=-alon
    if(ip.ne.ipold .or. kos.eq.-1) then
        if(nlines.ne.0) then
            if(iflag.eq.1) then
                call tcurve(flon, flat, npts)
            else
                call mask_conrec(flat, flon, npts, 1, 0, larea)
            end if
        end if
        nlines=nlines+1
        npts=1
        flat(1)=alat
        flon(1)=alon
        ipold=ip
        type *, ' pcity_land-i-nlines = ', nlines
    else
        npts=npts+1
        flat(npts)=alat
        flon(npts)=alon
    end if
enddo
end

subroutine mapmin(flon1, flon2, flat1, flat2)
if(flon1.gt.0) stop 'mapmin'
flon1=-flon1
flon2=-flon2
iflat1 = flat1*60.
iflat2 = flat2*60.
iflon1 = flon1*60.
iflon2 = flon2*60.
flat1=float(iflat1)/60.
flat2=float(iflat2)/60.
flon1=-float(iflon1)/60.
flon2=-float(iflon2)/60.
return
end

subroutine tcurve(x, y, n)
external between
common /con_grid/ xmin, xmax, ymin, ymax, nx, ny

```

```

real*4 x(n),y(n)
LOGICAL between
logical penup
penup = .true.
do i=1,n
  if(      between(xmin,xmax, x(i)) .and.
    *      between(ymin,ymax, y(i))
    *      ) then
    if(penup) then
      call frstpt ( x(i),y(i) )
      penup = .false.
    else
      call vector ( x(i),y(i) )
    endif
  else
    penup =.true.
  endif
enddo
return
end

```

```

C*****
C
C PROGRAM          M1
C
C PURPOSE          1) sets up map for land boundary plotting
C
C
C AUTHOR          NORMAN GUINASSO
C
C DATE            circa 1985
C
C REVISION        NOV. 1988
C
C takes calibration data and x,y station data from a *.mca file
C calls map_calib to make a *.cof file from calibration data
C also can call map_make_square to make a *.cof file for a square projection
C also looks for a *.flt auxillary file for data input as lat lon
C then calculates x y or lat long for each data point
C and writes all to a *.flt file
C all files take their * name from the specified input file
C*****
C
      common /mapname/ infile
      common /mapoffsets/ xoff,yoff,fac
      character*70 title
      character ans
! calibration data:
      real*4  x(110),y(110)           ! x,y coordinates in inches
      real*4  flat(110),flon(110)    ! lat and long in decimal degrees
! data points:
      integer*4 ident(900)
      real*4  xx(900),yy(900),xz,yz  ! x,y for data points
      real*4  xl(900),yl(900)        ! long, lat fro data points
      logical square                  ! if square plot set up a simple map
      logical finmark /.FALSE./      ! special handling for finmark
      logical digit_with_calib
      logical to_xy                   /.true./
      logical to_latlon              /.false./

      character*30 infile,outfile,auxfile
      character*60 auxformat
      logical rot_clkwise
      logical flt_file

! =====
! calibrated or square
! x,y points or flt file
!
      fac=1.
      j=lib$init_timer()
      type *, 'Program M1 version 1.1'

      square=.true.
      type '(' calibrated or square? (c/s) -->'',$)
      accept '(a)', ans
      if(ans.eq.'c' .or. ans.eq.'C') square=.false.
      if(square) then
         type *, ' scale? '
         accept *,sqscale
      endif

101  format(' Enter input file name -> ', $)
      type 101
      accept 103,infile

```

```

call chgext(infile,outfile,'f10')
open(10,name=outfile,err=991,status='NEW',
1      carriagecontrol='LIST')

!-----
! if .flt file is found, then assume lat lon is there
! first record contains format
! with each of the following records containing:
! station lat_deg lat_min flat_secs lon_deg lon_min flon_secs
!
call chgext(infile,auxfile,'flt')
open(11,name=auxfile,status='OLD',err=10)
! found aux file:
    digit_with_calib = .false.
    type *, 'Found auxillary file with lat lon data: ',auxfile
    read(11,'(a)') auxformat
    type '(1x,a)', ' ', auxformat
    goto 11
! did not find aux file:
10      digit_with_calib = .true.
!
11      continue
!-----
! start:

flt_file= .not. digit_with_calib

if(.not.(flt_file.and.square)) then
!
!   we need to read calibration data

call chgext(infile,infile,'mca')
OPEN(9,name=infile,err=990,status='old',readonly)
read(9,100) n,title
type 109, n,title
write(10,109) n,title

if (n.gt.110) then
    type *, ' this will not work, n too big'
    stop
endif

! read in calibration data
138 format(' Rotate clockwise? (y/n) --> ', $)
type 138
accept '(a)', ans
rot_clkwise = .false.
if(ans.eq.'y' .or. ans.eq.'Y') rot_clkwise=.true.
do i=1,n
    read(9,*,end=900,err=901) Lat,Long,x(i),y(i)
    if(rot_clkwise) then
        xtemp = x(i)
        x(i) = y(i)
        y(i) = -xtemp
    endif
    TYPE *, 'DENIS MODIFIED ME'
    flat(i) = ghexij(lat)
    flon(i) = ghexij(long)
    TYPE 102,flat(i),flon(i),x(i),y(i)
    write(10,102) flat(i),flon(i),x(i),y(i)

enddo
endif

```

```

if (square) then
    call map_make_square(sqscale,1) ! (scale, 1= write file)
else
    call map_calib(flat,flon,x,y,n)
endif

! -----
if(finnmark) then
    nskip=21
    do k=1,nskip
        read(9,103) title
    enddo
endif

! -----
i=1          ! next point to read
npoints=0    ! points read

if(digit_with_calib) then
    read (9 ,103)  title
    write(10,106)  title
    read station locations from digitizer file as x,y
    do while (.true.)
        read(9,*,end=50,err=50) ident(i),xx(i),yy(i)
        if(rot_clkwise) then
            xtemp = xx(i)
            xx(i) = yy(i)
            yy(i) = -xtemp
        endif
        type 107,ident(i),xx(i),yy(i)
        if(finnmark) ident(i)=ident(i)+2000
        npoints =npoints+1
        xt=xx(i)
        yt=yy(i)
        call map_xform(ft,fg,xt,yt,to_latlon) ! to lat long
        xl(i)=fg
        yl(i)=ft
        write(10,108) i,ident(i),xx(i),yy(i),yl(i),xl(i)
        i = i+1
    enddo

else
    read station locations from auxillary file as lat long
    do while (.true.)
        read(11,auxformat,end=50,err=44) ident(i),
        latdeg,latmin,secslat,londeg,lonmin,secslon
        npoints =npoints+1

        type '(2i4,f6.1,2i4,f6.1)',
        latdeg,latmin,secslat,londeg,lonmin,secslon

! function ghexijf converts degrees minutes and seconds to floating degrees
! degrees and minutes are integers, seconds are floating point

        fg=ghexijf(londeg,lonmin,secslon)
        ft=ghexijf(latdeg,latmin,secslat)
        type *,ft,fg

        type '(i4,2f10.3)', ident(i),ft,fg,ident(i),xz,yz
        call map_xform(ft,fg,xz,yz,to_xy) ! to x,y
        xl(i)=fg ! long
        yl(i)=ft ! lat
        xx(i)=xz

```

```
yy(i)=yz
write(10,108) i,ident(i),xx(i),yy(i),yl(i),xl(i)
i      =      i+1
goto 45
44 type *, ' error while reading on unit 11 ',auxfile
45 enddo
endif! (digit_with_calib)

50 continue

type *, 'number of points = ',npoints
j=lib$show_timer()
stop 'normal ending'

900 stop 'program m1-f-unexpected end of data'
901 stop 'program m1-f-error reading calibration data'
990 stop 'program m1-f-error opening input file unit 9'
991 stop 'program m1-f-error opening output file'

100 format (i3,a) ! read format
102 format(1x,2f10.3,2f10.4)
103 format(a)
106 format(1x,a)
108 format(1x,2I7,4f12.5)
107 format(1x,i4,2f8.2)
109 format(1x,i3,' --->',a)
! square with calibration data and point data
! square without calibration data and with flt file
! calibration data and x,y data
! calibration data and lat long data in flt file
end
```

```

/*****
/*
/* PROGRAM      READ.SAS
/*
/* PURPOSE      1)  sas program to consolidate .fin files
/*
/* AUTHOR       NORMAN GUINASSO
/*
/* DATE        NOV. 88
/*
/* REVISION     DEC. 88
/*
*****/
FILENAME PCITY1 '[DENIS.PCITY]19JUN88.FIN';
LIBNAME NORDA '[DENIS.PCITY]';
DATA temp1;
INFILE PCITY1;
INPUT  YEAR MONTH DAY TIME $ SECS MID
      LAT LON T DELAY1 T DELAY2
      HEADING SPEED VREF TESS441 TES448
      TESSPYR1 TESSPYR2 TEMPER SALIN
      PTRANS FLUOR L465NM L507NM L532NM
      TRANSVLT FL_SIG_V FL_SCL_V L488NM DEPTH;
data temp2;
FILENAME PCITY2 '[DENIS.PCITY]20JUN88.FIN';
INFILE PCITY2;
INPUT  YEAR MONTH DAY TIME $ SECS MID
      LAT LON T DELAY1 T DELAY2
      HEADING SPEED VREF TESS441 TES448
      TESSPYR1 TESSPYR2 TEMPER SALIN
      PTRANS FLUOR L465NM L507NM L532NM
      TRANSVLT FL_SIG_V FL_SCL_V L488NM DEPTH;
data temp3;
FILENAME PCITY3 '[DENIS.PCITY]21JUN88.FIN';
INFILE PCITY3;
INPUT  YEAR MONTH DAY TIME $ SECS MID
      LAT LON T DELAY1 T DELAY2
      HEADING SPEED VREF TESS441 TES448
      TESSPYR1 TESSPYR2 TEMPER SALIN
      PTRANS FLUOR L465NM L507NM L532NM
      TRANSVLT FL_SIG_V FL_SCL_V L488NM DEPTH;
data norda.all;
      set temp1 temp2 temp3;
PROC MEANS;
ENDSAS;

```

```
*****  
/*  
/* PROGRAM PUTALL.SAS  
/*  
/* PURPOSE 1) sas program to make smooth data file  
/*  
/* 2) scans large sas file and outputs  
/* selective records  
/*  
/* AUTHOR NORMAN GUINASSO  
/*  
/* DATE NOV. 88  
/*  
/* REVISION DEC. 88  
/*  
*****  
FILENAME PCITY '[DENIS.PCITY]all21.nav';  
LIBNAME NORDA '[DENIS.PCITY]';  
DATA TEMP;  
    FILE PCITY;  
    SET NORDA.all;  
    IF DAY= 21 THEN PUT  
        TIME 1-10 LAT 10.6 LON 11.6 DEPTH 6.2  
        @40 TEMPER SALIN PTRANS FLUOR L465NM L488NM L507NM L532NM  
        TESS441 TES448 TESSPYR1 TESSPYR2 SECS_MID;  
ENDSAS;
```

```
*****  
/*  
/* PROGRAM      CORR.SAS  
/*  
/* PURPOSE      1) sas program to produce correlation matrix  
/*  
/* AUTHOR       NORMAN GUINASSO  
/*  
/* DATE         DEC. 88  
/*  
/* REVISION     DEC. 88  
/*  
*****  
LIBNAME NORDA '[DENIS.PCITY]';  
data temp;  
    set norda.all;  
    tess448 = tes448;  
    keep tess448 tess441 tesspyr1 tesspyr2  
        temper salin ptrans fluor  
        L465NM L507NM L532NM L488NM DEPTH;  
proc corr data=TEMP;  
ENDSAS;
```

```

C*****
C
C PROGRAM          READ 350
C
C PURPOSE          1) RT-11 FORTRAN_IV program to read disk to recover files
C                  2) modified for NORDA PRO-350 1988
C
C AUTHOR           NORMAN GUINASSO
C
C DATE             1982
C
C REVISION         OCT. 1988
C
C*****
C
integer*2 dblk(4)      ! floppy
integer*2 dblko(4)     ! output file
byte buff(514)         ! block buffer
byte bell              ! ring
byte ans               ! a character
logical octal          ! print to screen in octal
logical ascii          ! print to screen in ascii
logical wtf            ! write to file

c
data bell /7/          ! ^G
data dblk /3RDWO,3R    ,3R    ,3R    /      ! the pro350
data dblko/3RDZ1,3RREA,3RDXX,3ROUT/        ! du0:readxx.out

c
buff(513)=0
buff(514)=0
type *, ' program to use hardware read to recover files'
type *, ' from floppy - - - guinasso, circa 1987'

c
lb      =      1+istop-istart      ! length of output file

c
c options:
c
ascii   =      .false. ! write first 75 characters of block ascii
wtf     =      .false. ! write first 75 characters of block ascii
type *, ' printout block in Ascii, Octal, or None?'
      accept 101,ans
if(ans.eq.'a' .or. ans.eq.'A') ascii   =.true.
if(ans.eq.'o' .or. ans.eq.'O') octal   =.true.
type *, ' write to file? '
      accept 101,ans
if(ans.eq.'y' .or. ans.eq.'Y') wtf     =.true.

c
c get channels
      ich=igetc()      ! channel for floppy
      ichout=igetc()   ! channel for output file

c
c open disk as file:
      if(ifetch(dblk).lt.0) stop 'bad fetch' ! fetch floppy handler
      if(lookup(ich,dblk).lt.0) stop 'lookup failed' ! open floppy

c
      if(.not.wtf) goto 4      ! writing to file
      if(ienter(ichout,dblko,lb).lt.0)
%          stop 'enter failed' ! open output
      type *, ' file opened for output'
      iblko=0      ! first block is 0
c

```

```

4      1start=0
      1stop=11000
      1blk=1start
c
      do 10 i=1,10100
          ierr=0
          continue
5      j=ireadw(256,buff,1blk,ich)
          if(j.eq.-1) goto 11      ! end of file
          if(j.eq.-2) goto 8       ! error
          type *, ' BLOCK = ',    1blk
          1blk =                  1blk+1
          jin=index(buff,'20-JUN-88')
          wtf=jin
          if(wtf)      j =          iwritw(256,buff,1blk,ichout)
          if(wtf) 1blk =          1blk+1
          if(octal)      type 100,(buff(k),k=1,75)
          if(ascii)      type 103,(buff(k),k=1,75)
          if(1blk.gt.1stop) goto 20
          goto 10
c
c hard error handling
c
8      type 102 ,bell,1blk
          ierr=ierr+1
          if(ierr.lt.4) goto 5      ! retry 3 times
          stop ' hardware error'
c
10     continue
      goto 20
11     type *, 'eof, block=', 1blk
20     call iclose(ich)
      call iclose(ichout)
      stop
100    format(20o4)
101    format(a)
102    format(1x,a,' hard error, block=',i4)
103    format(80a1)
      end

```

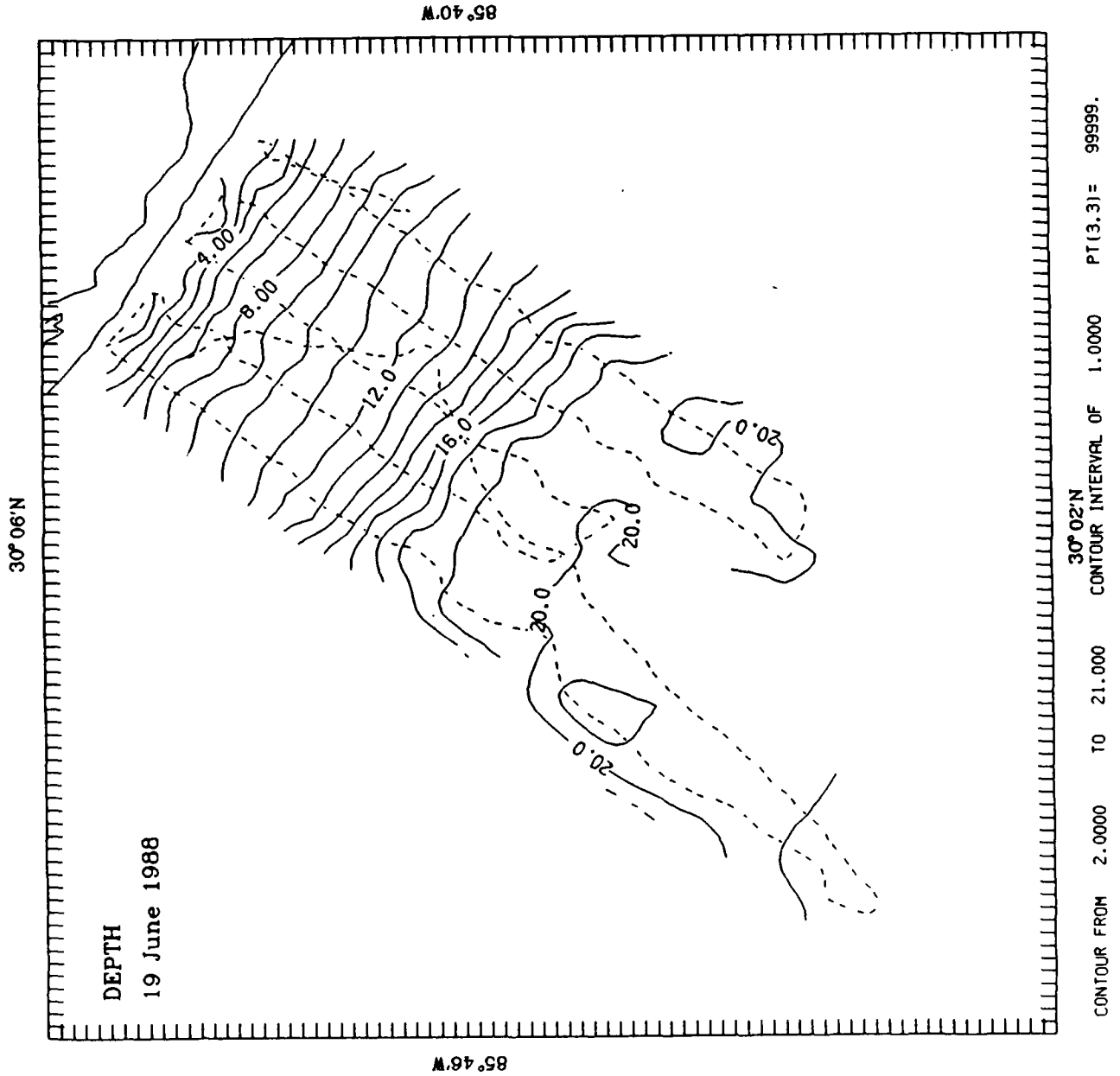
```

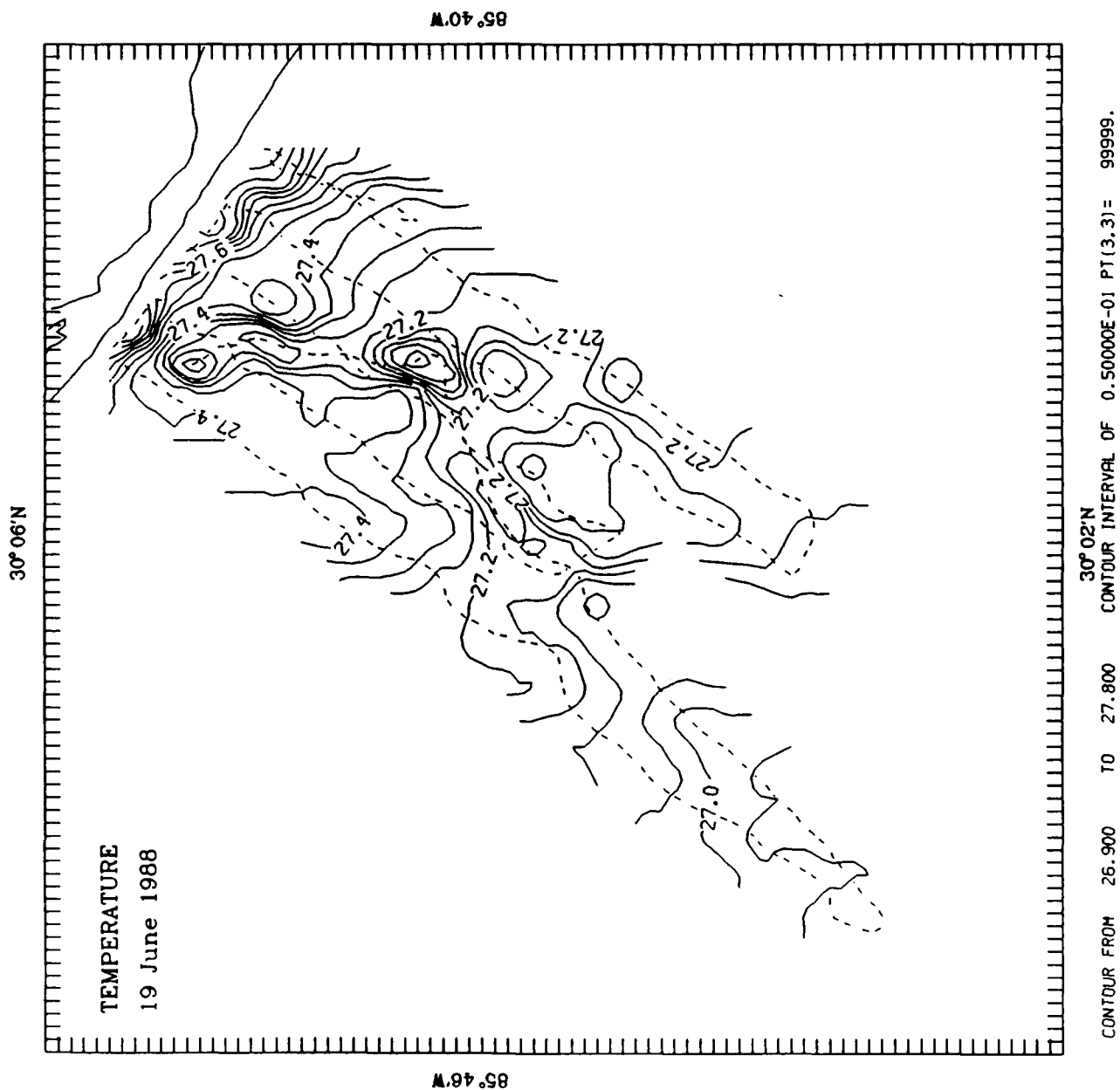
/*****
/*
/* PROGRAM          PLOT.SAS
/*
/* PURPOSE          1) sas program to plot light data
/*
/* AUTHOR           NORMAN GUINASSO
/*
/* DATE             DEC. 88
/*
/* REVISION         DEC. 88
/*
*****/
/* PLOT LIGHT DATA AND DEPTH*/
LIBNAME NORDA '[DENIS.PCITY]';
DATA NORDA.ALLSUB;
    set norda.all;
    tess448 = tes448;
    NORM UP = L507NM/TESSPYR1; /*NORMALIZED UPWELLING L507*/
    INORM UP = 1./NORM UP;
    KEEP
    DAY tess448 tess441 tesspyr1 tesspyr2
    temper salin ptrans fluor NORM UP INORM UP
    L465NM L507NM L532NM L488NM DEPTH;
DATA TEMP;
    SET NORDA.ALLSUB;
    IF DAY=19 AND DEPTH<12.;
PROC PLOT DATA=TEMP;
    PLOT DEPTH*INORM UP;
    TITLE 'INVERSE NORMALIZED 507NM UPWELLING IRRADIANCE';
    TITLE2 '19 JUNE 1988';
DATA TEMP;
    SET NORDA.ALLSUB;
    IF DAY=20 AND DEPTH<12.;
PROC PLOT DATA=TEMP;
    PLOT DEPTH*INORM UP;
    TITLE 'INVERSE NORMALIZED 507NM UPWELLING IRRADIANCE';
    TITLE2 '20 JUNE 1988';
;
DATA TEMP;
    SET NORDA.ALLSUB;
    IF DAY=21 AND DEPTH<12.;
PROC PLOT DATA=TEMP;
    PLOT DEPTH*INORM UP;
    TITLE 'INVERSE NORMALIZED 507NM UPWELLING IRRADIANCE';
    TITLE2 '21 JUNE 1988';
ENDSAS;
;

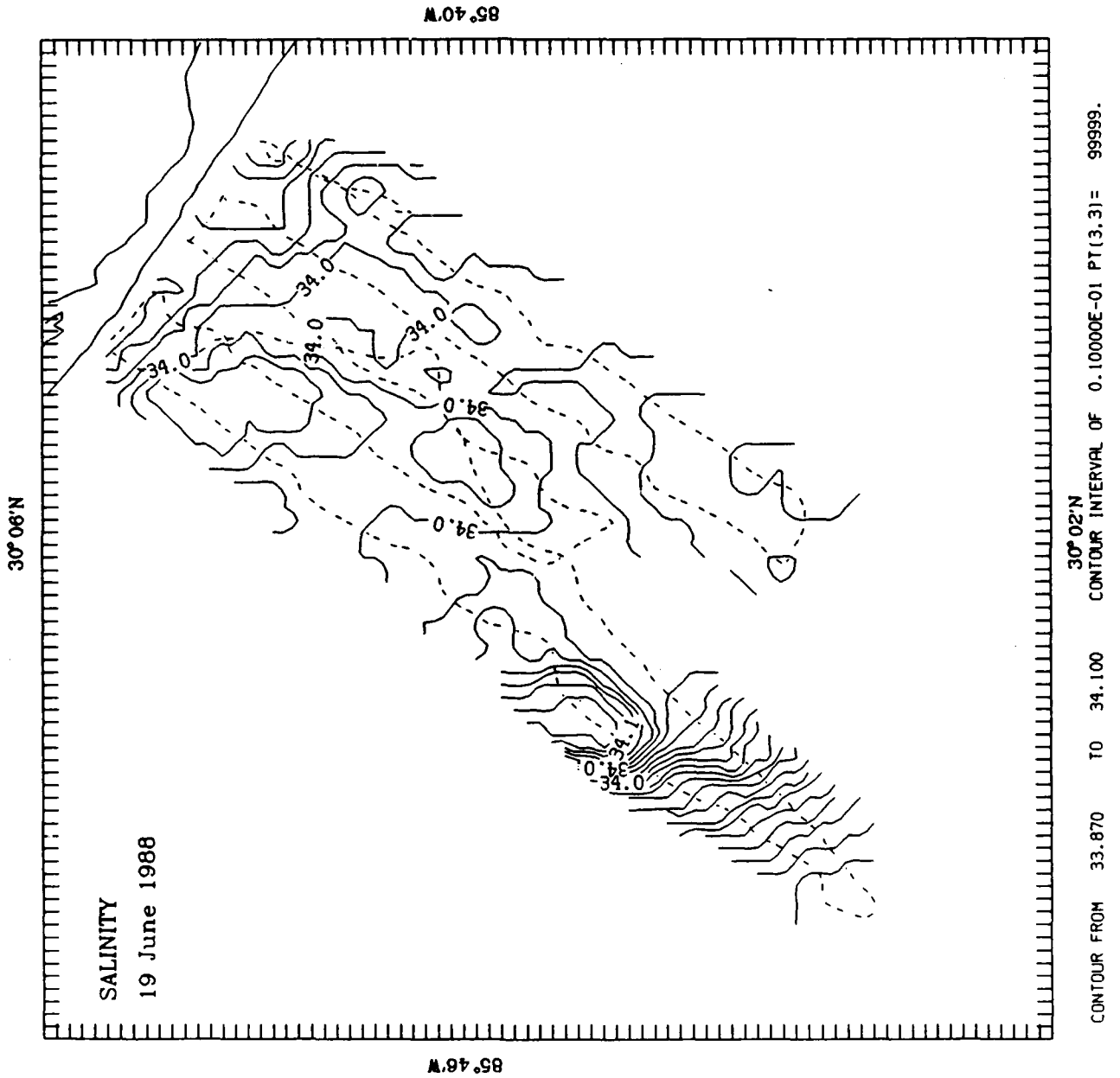
```

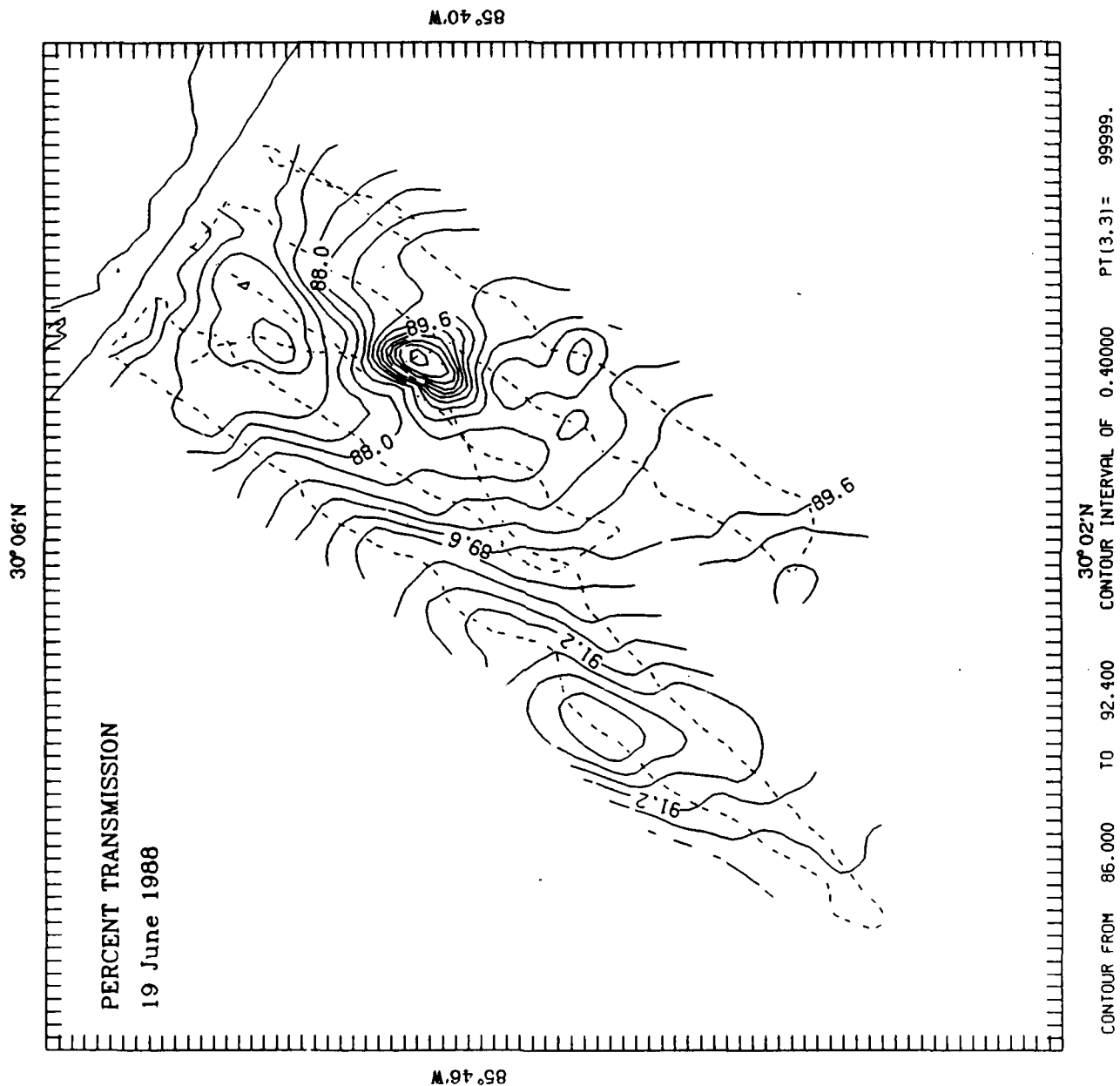
APPENDIX B

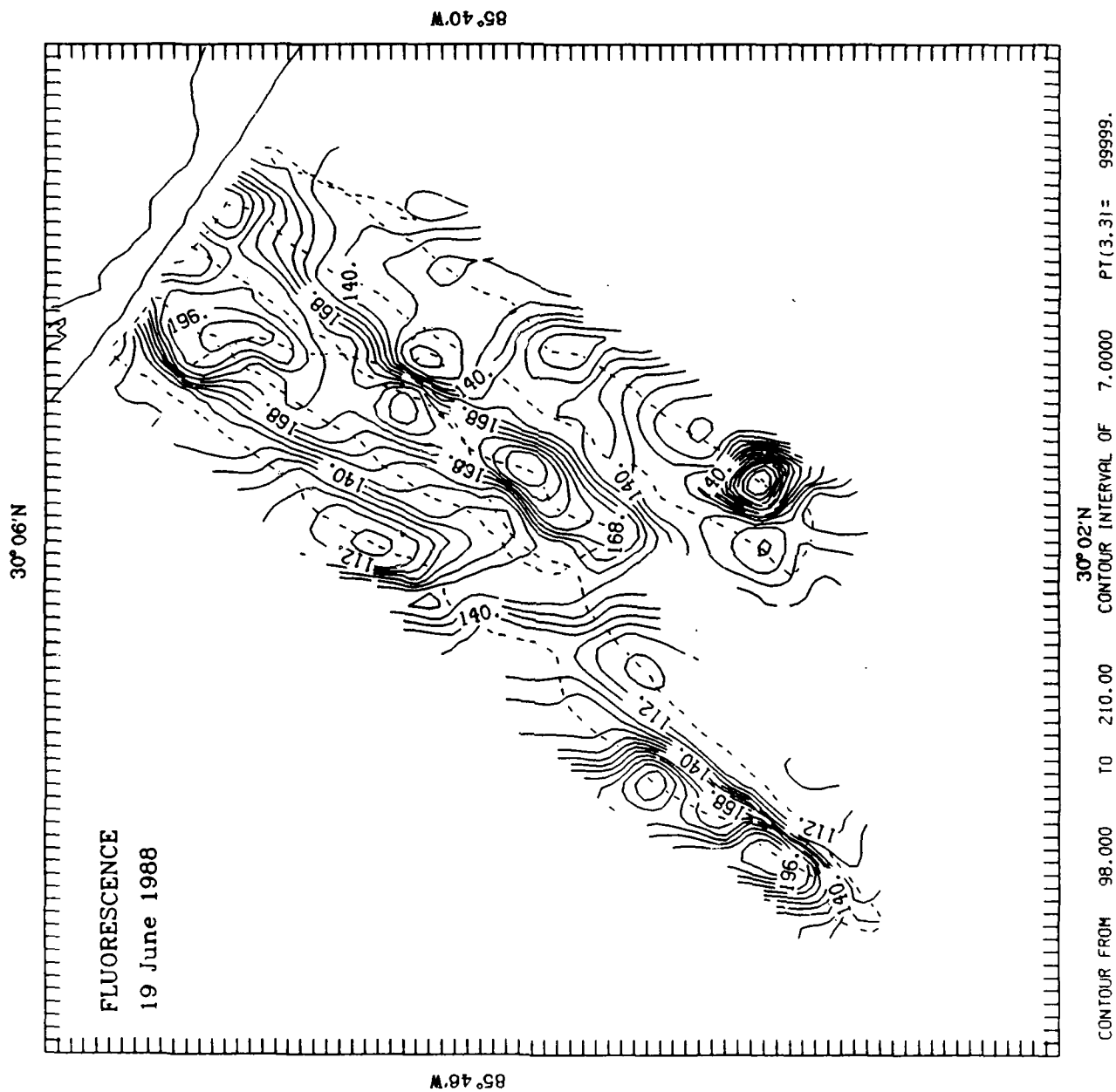
CONTOUR MAPS

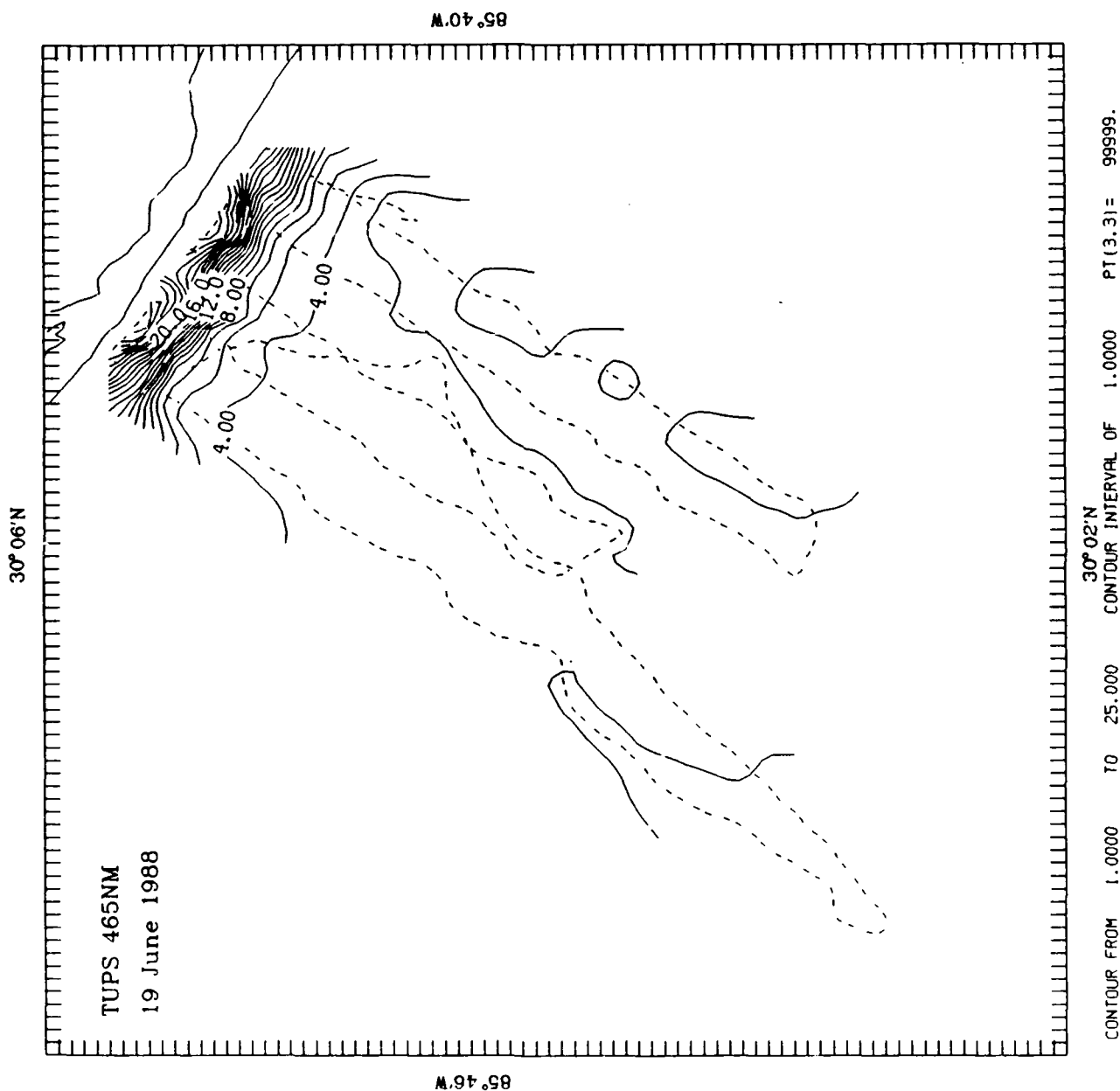


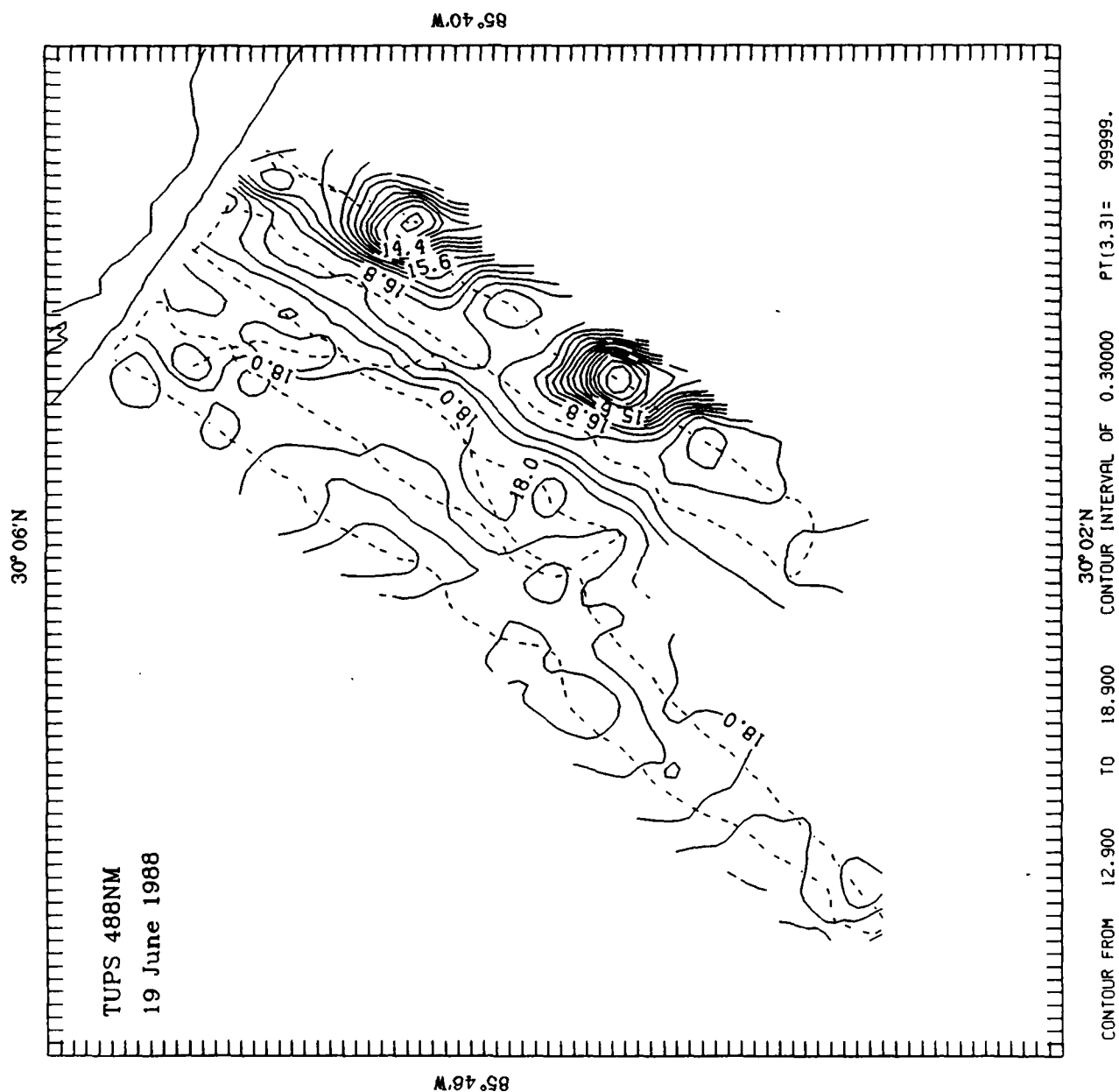


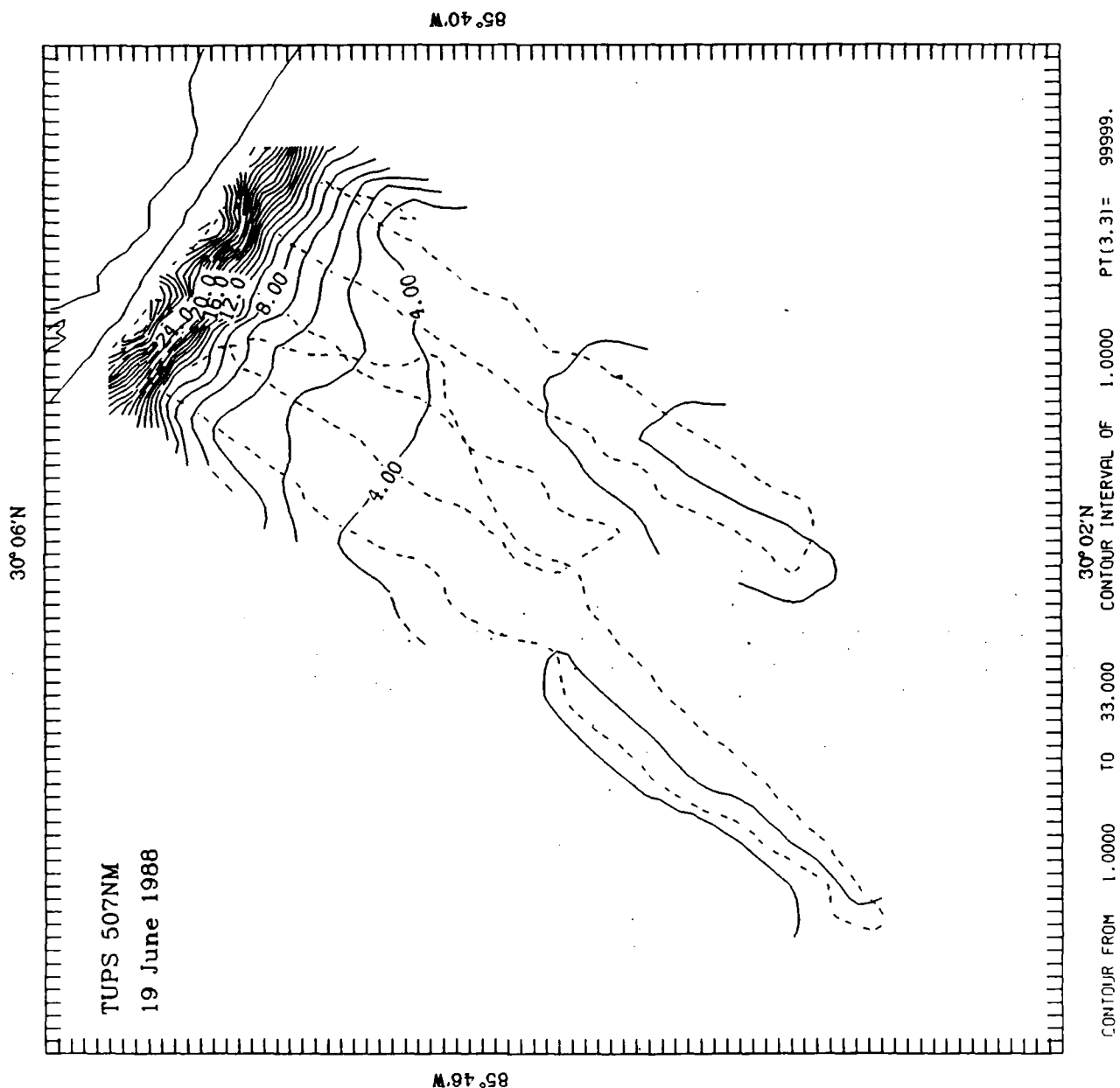


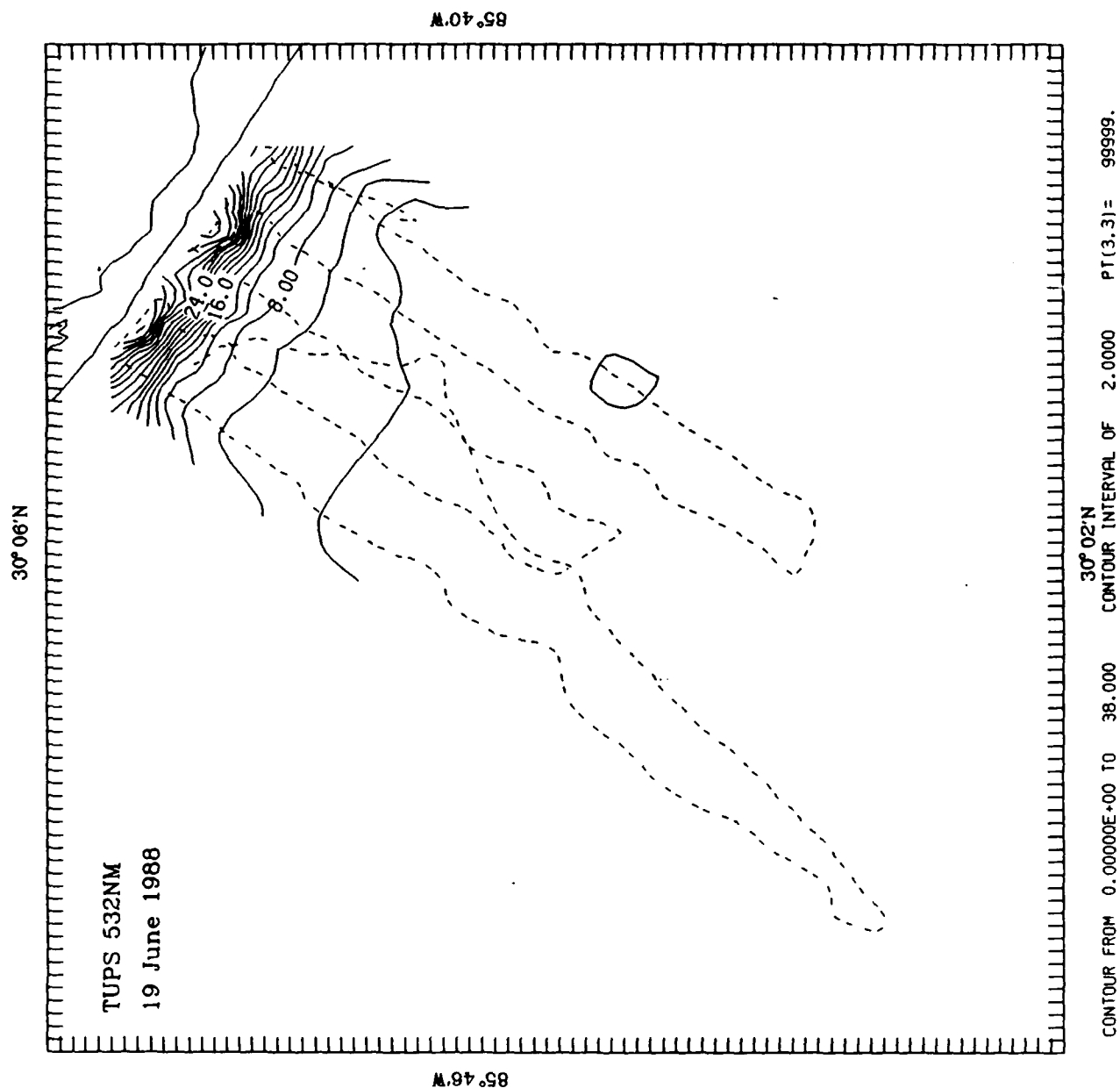


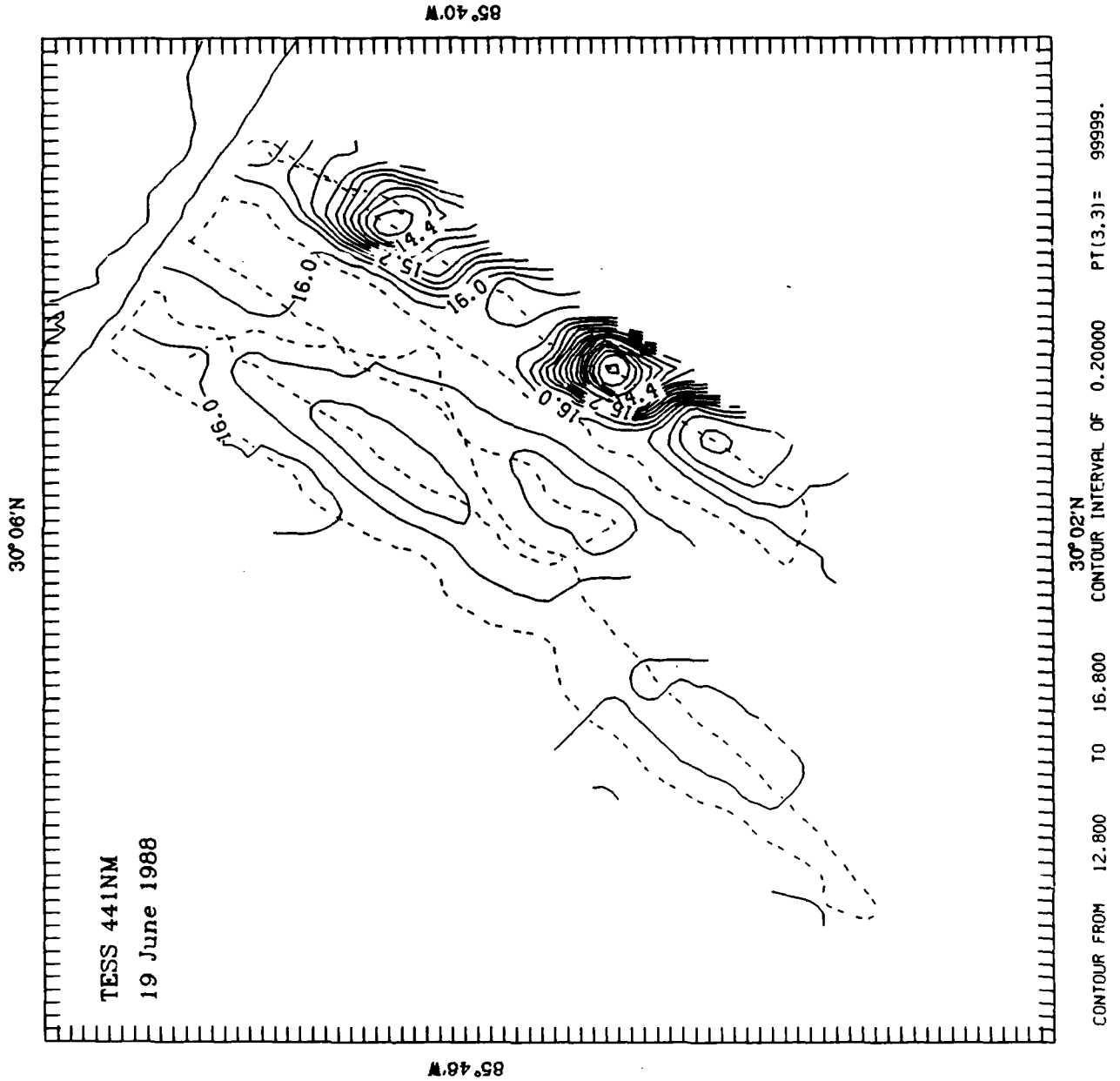


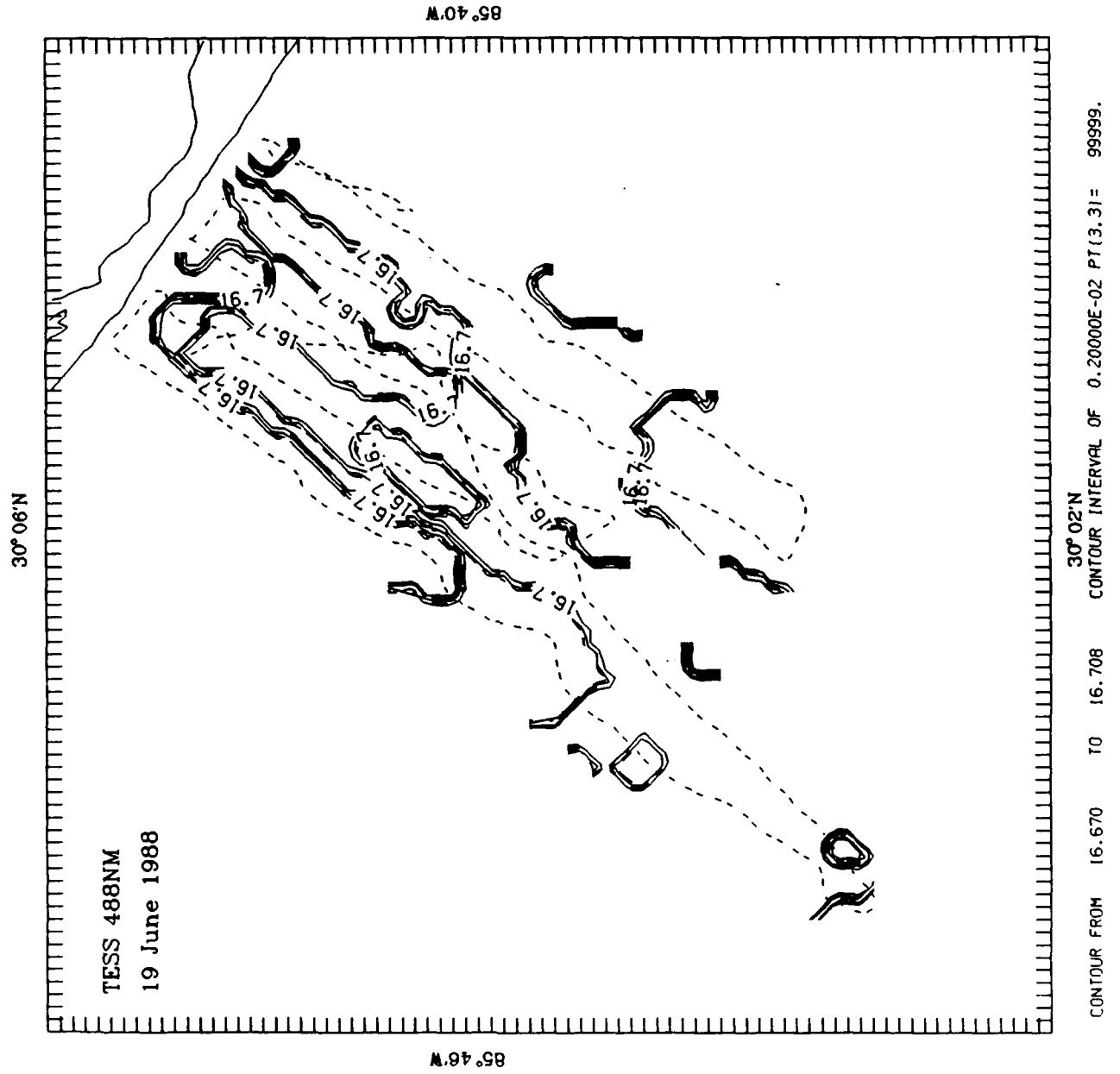


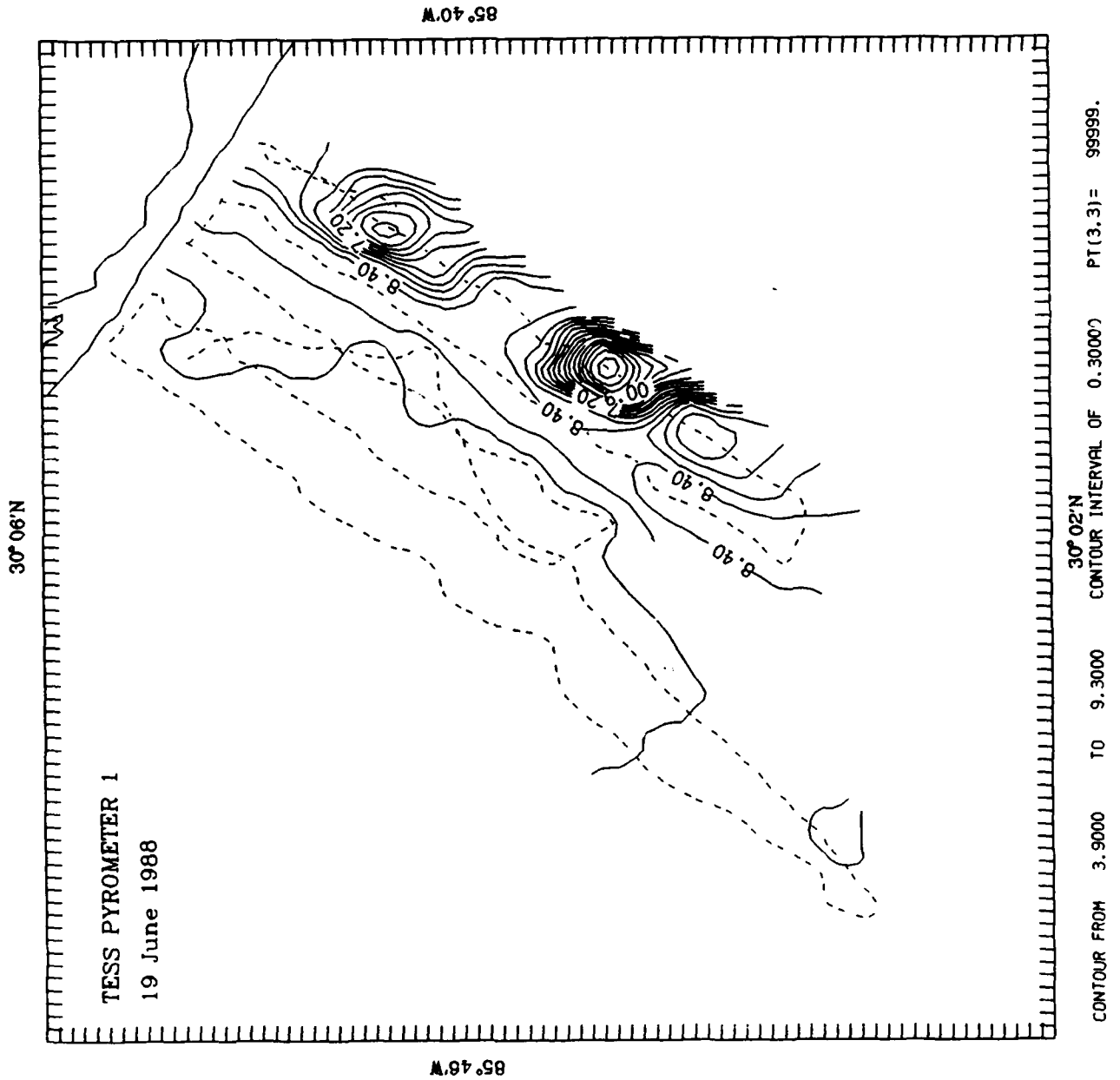


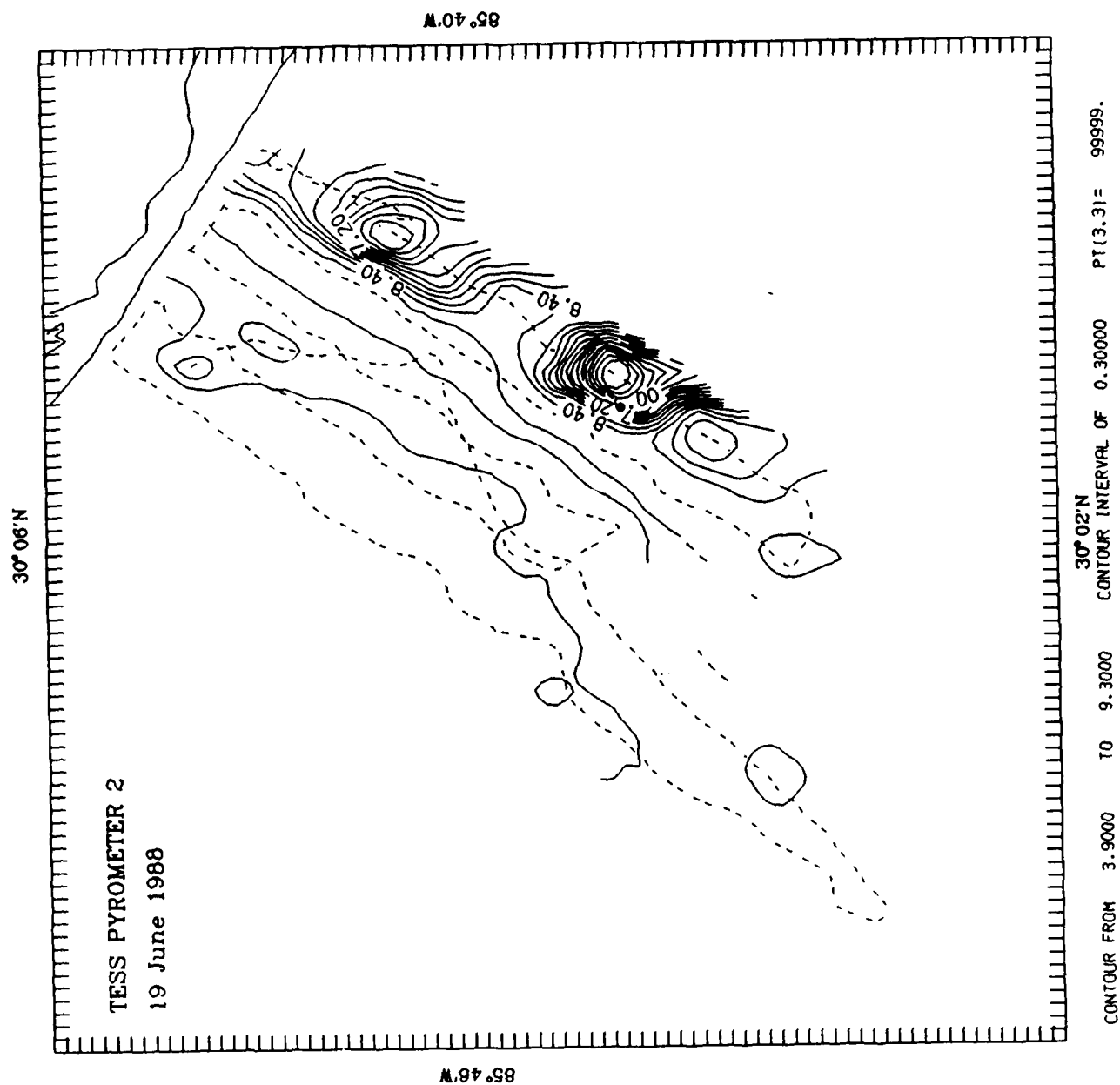


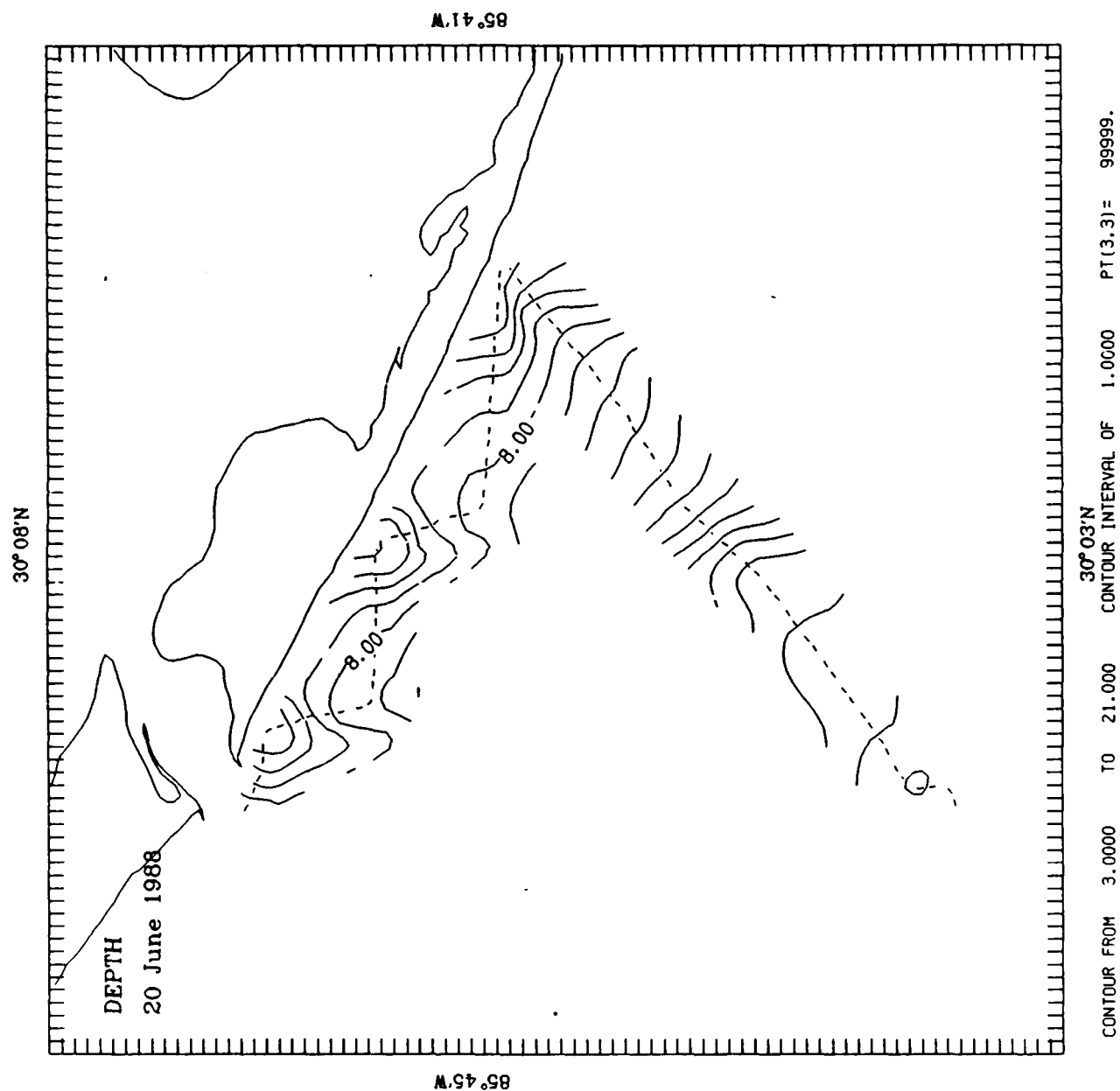


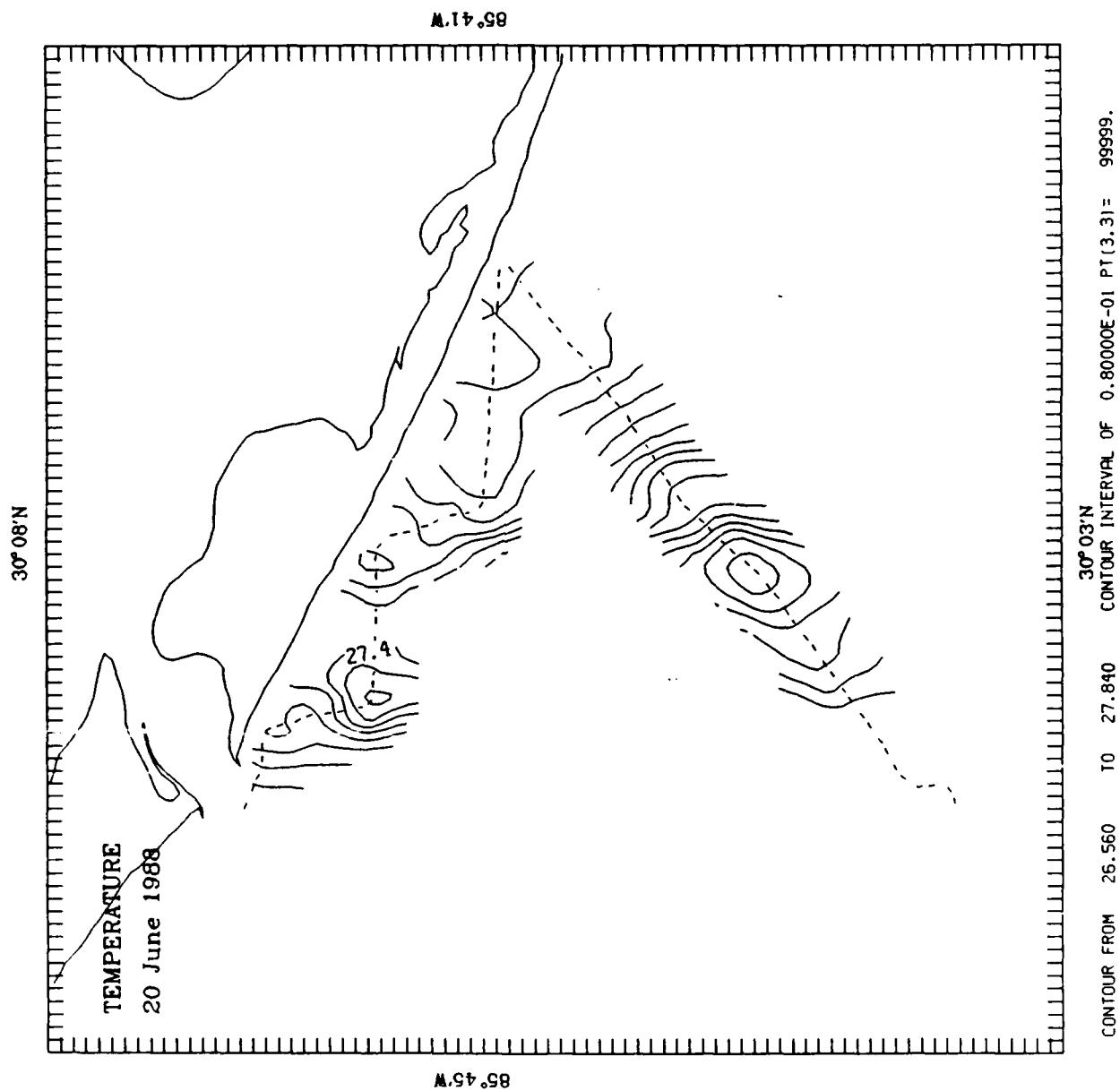


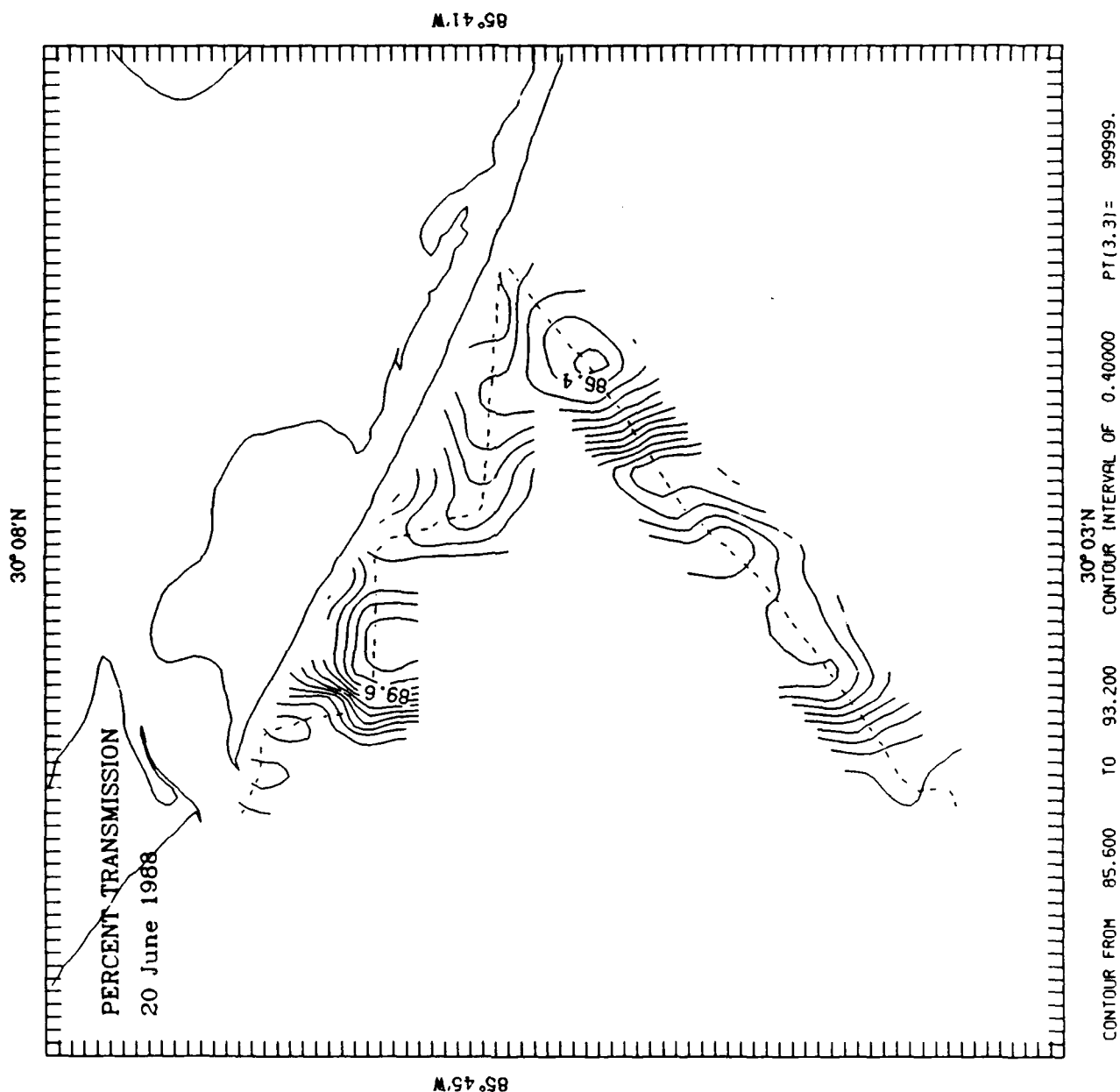


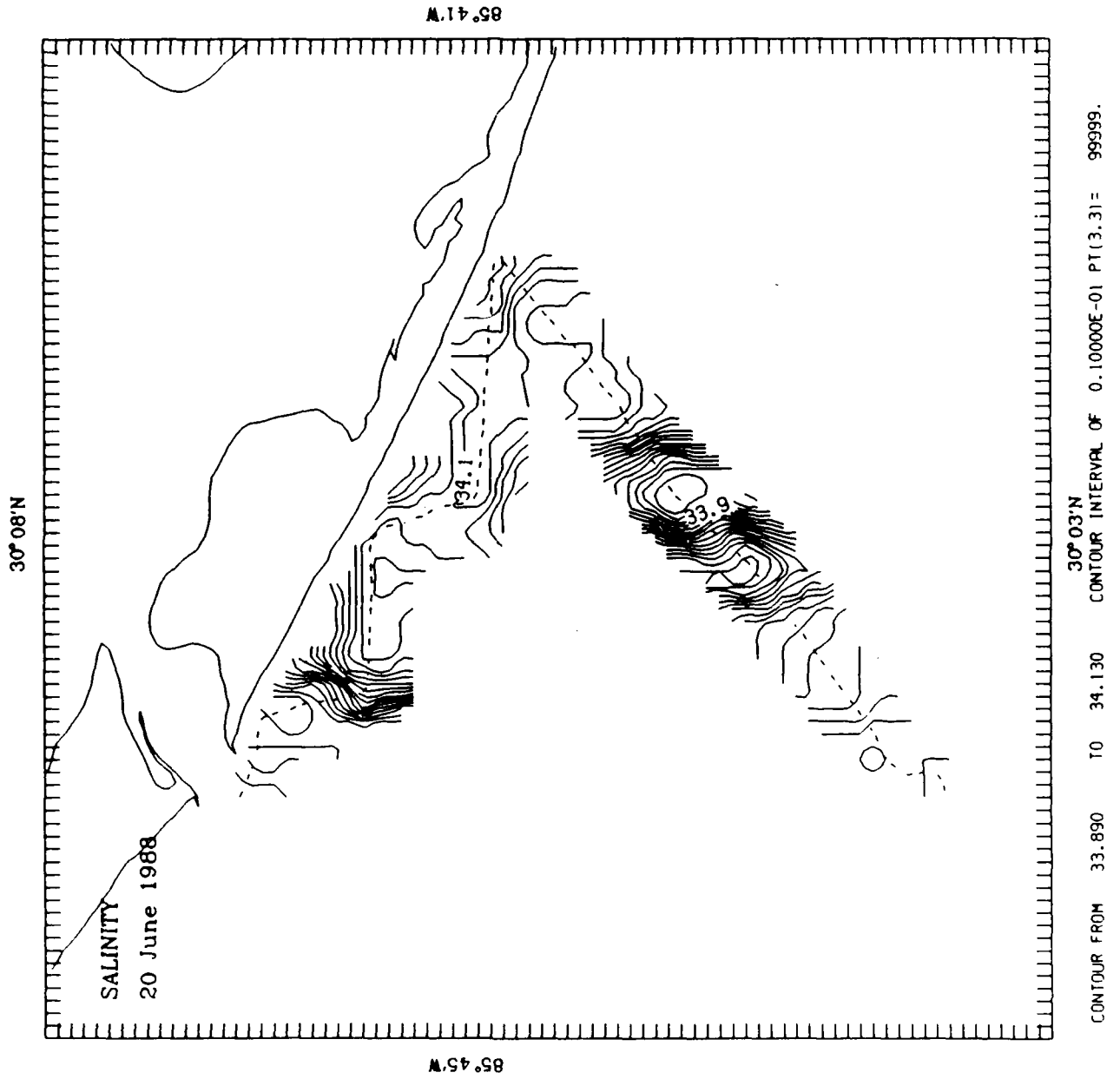


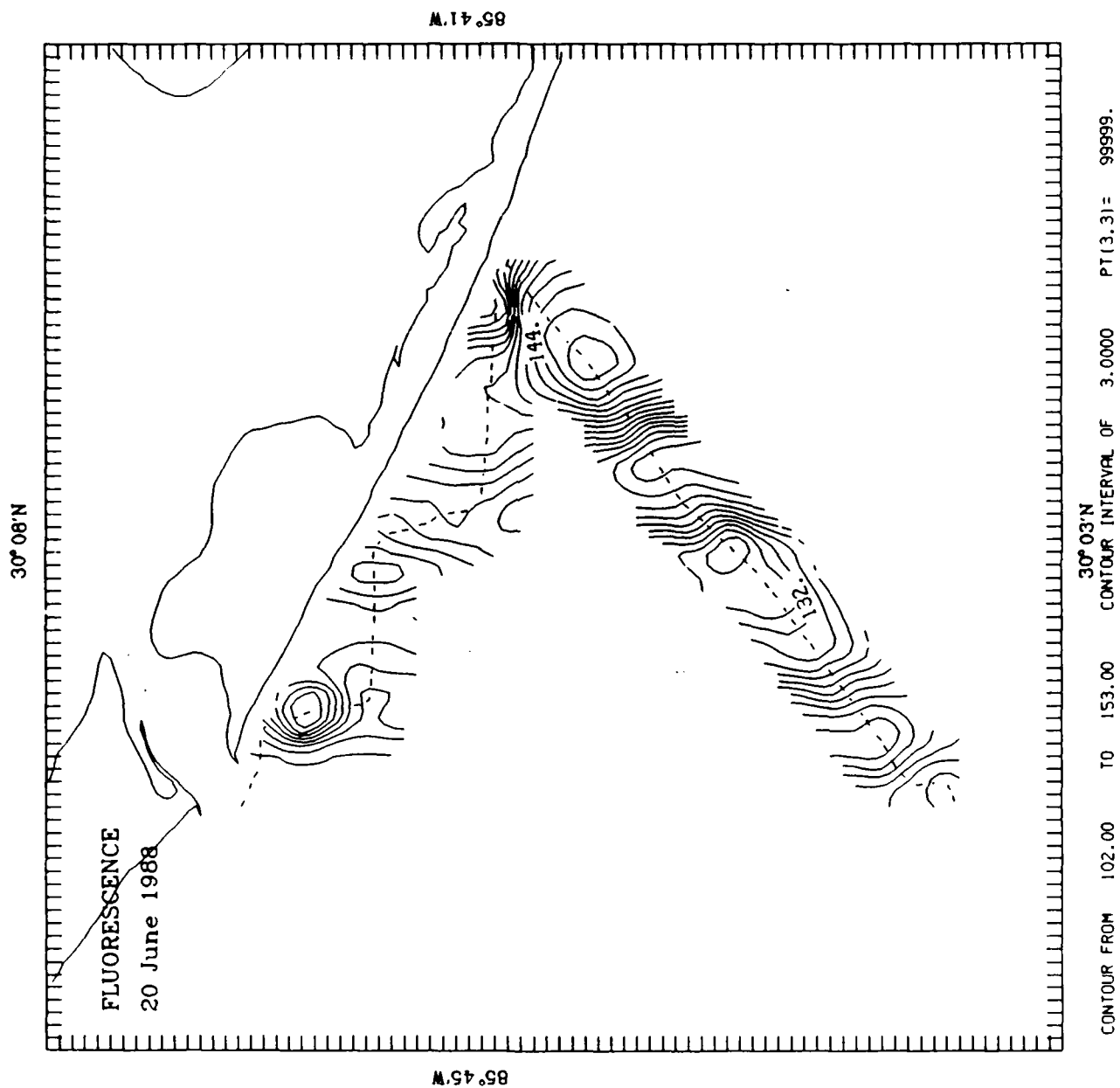


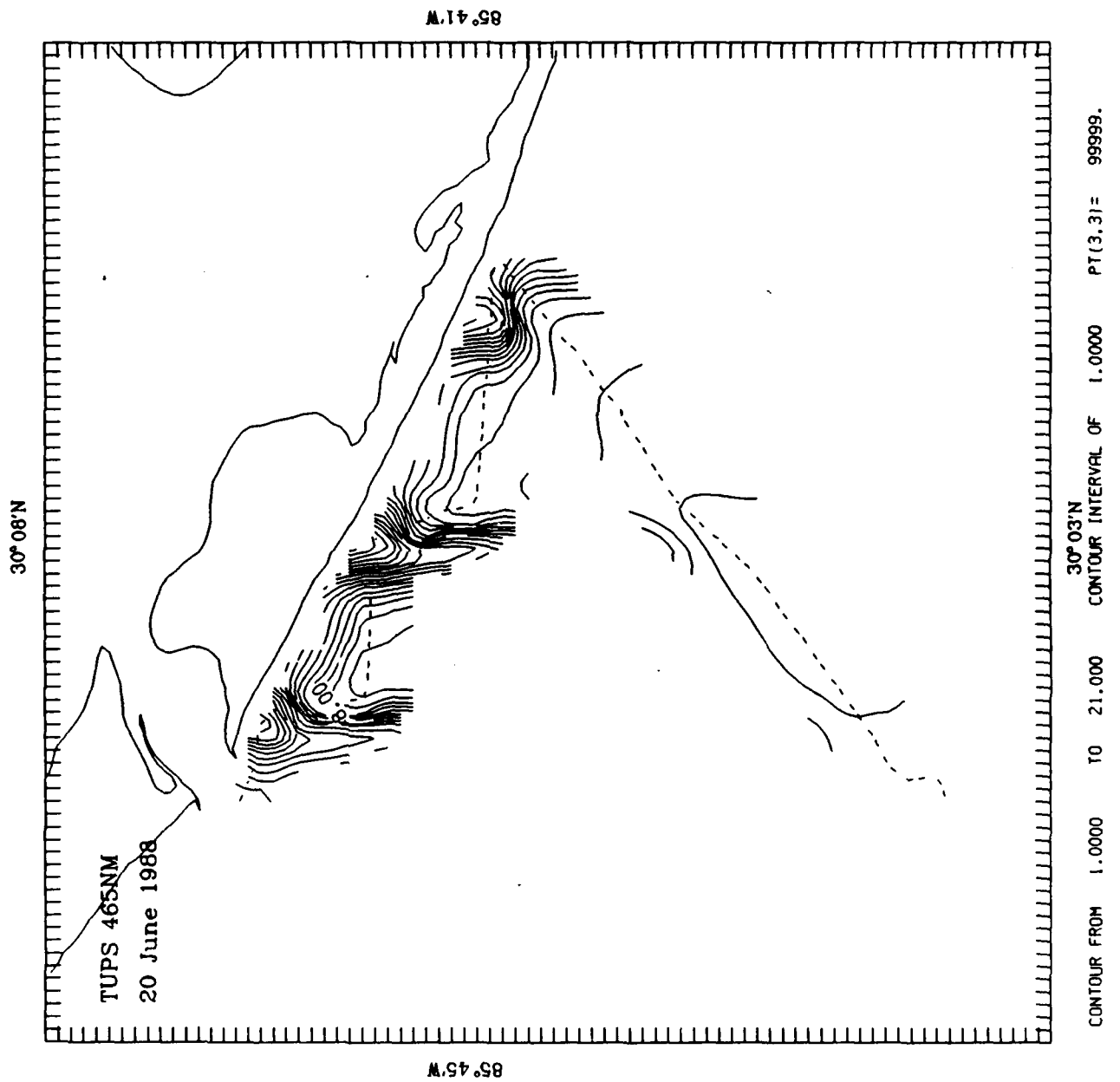




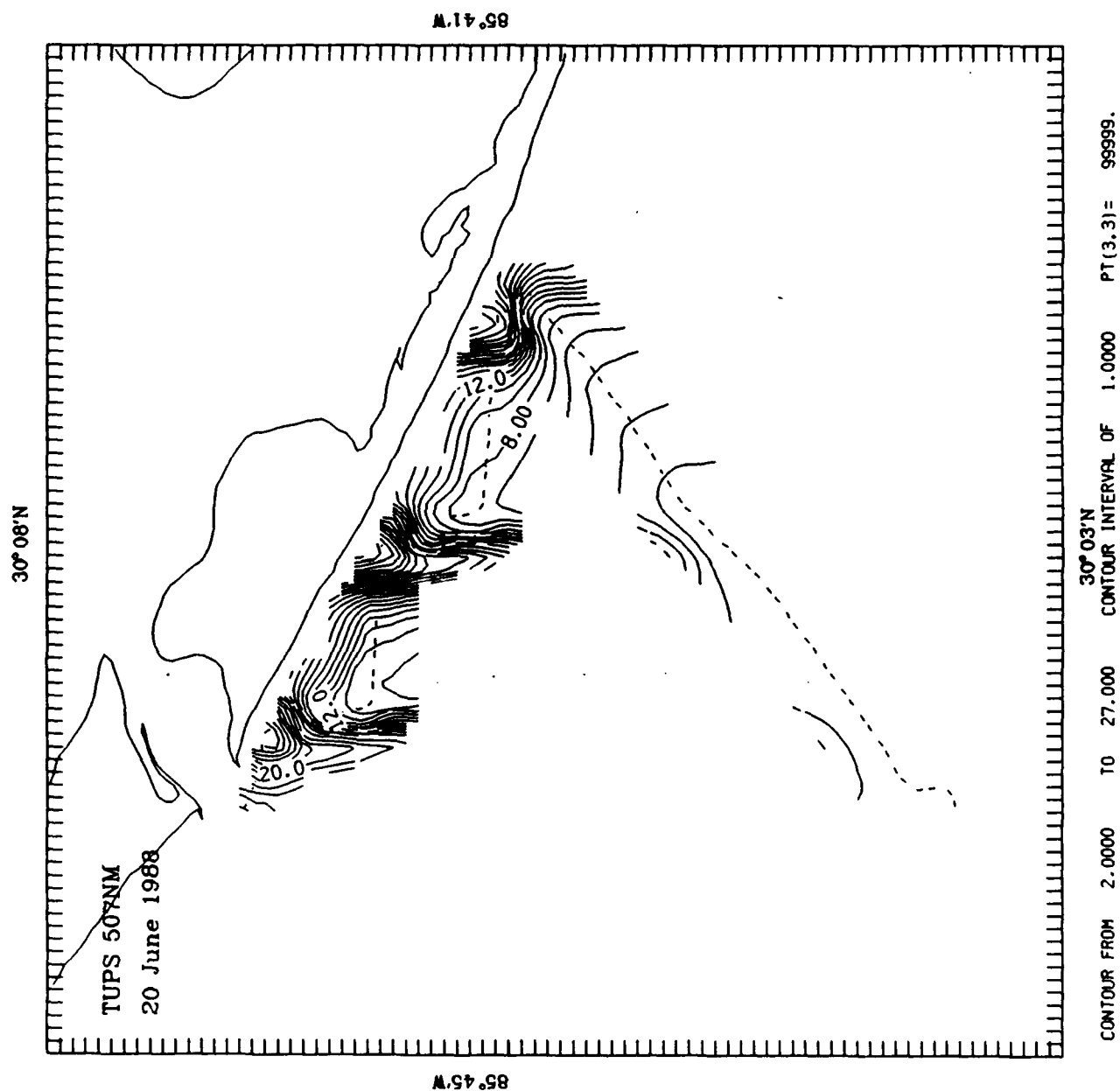


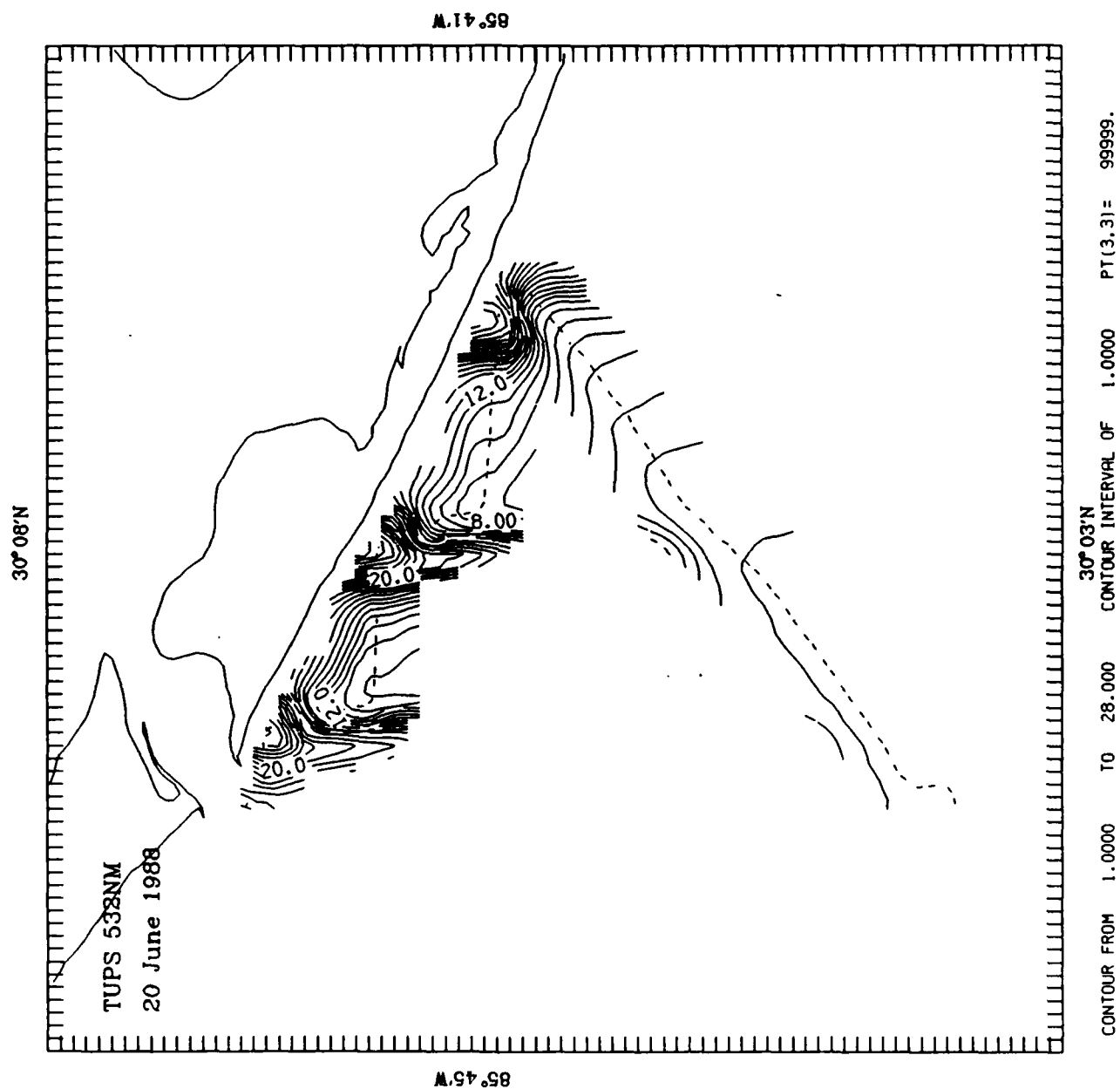


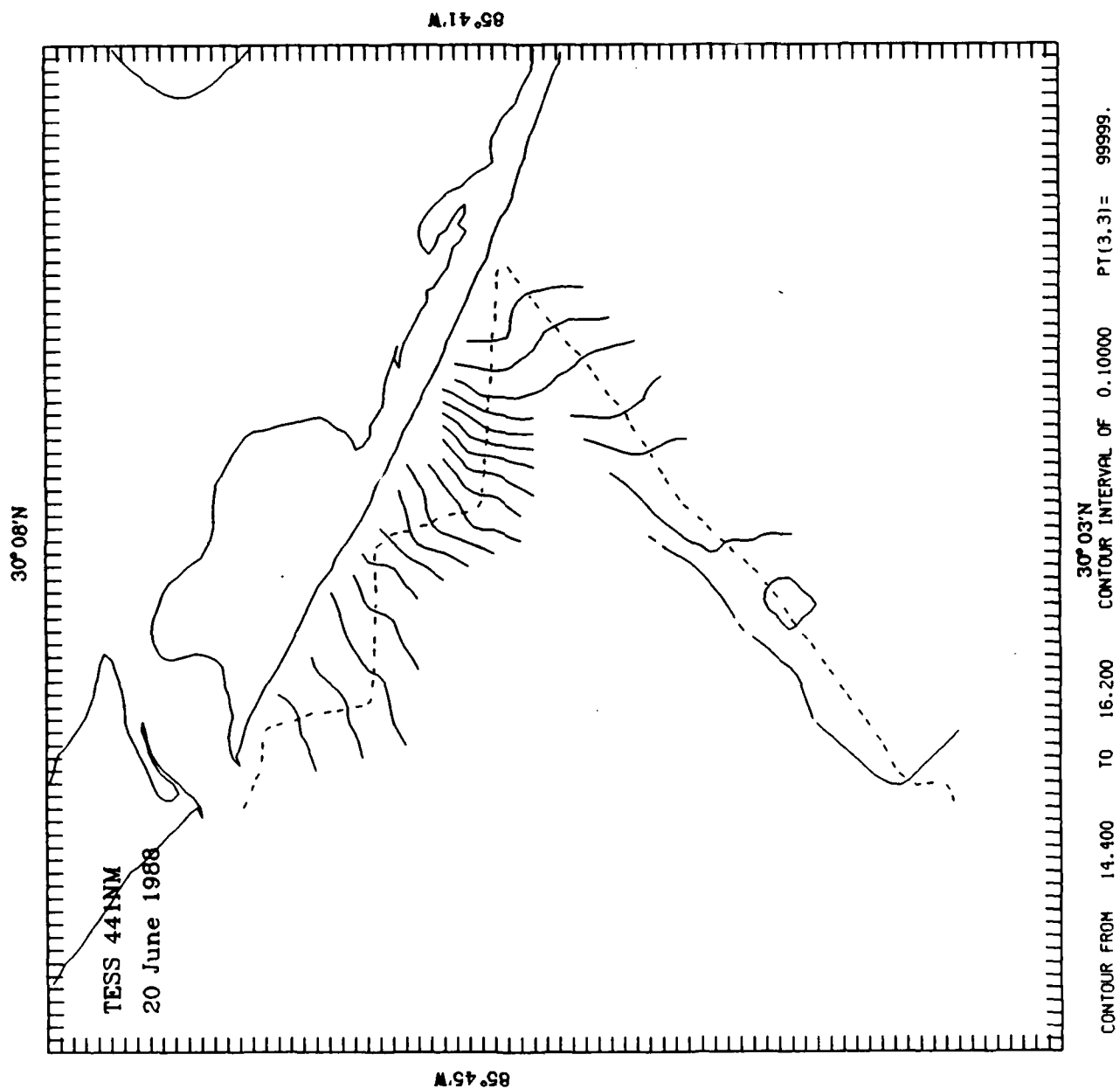


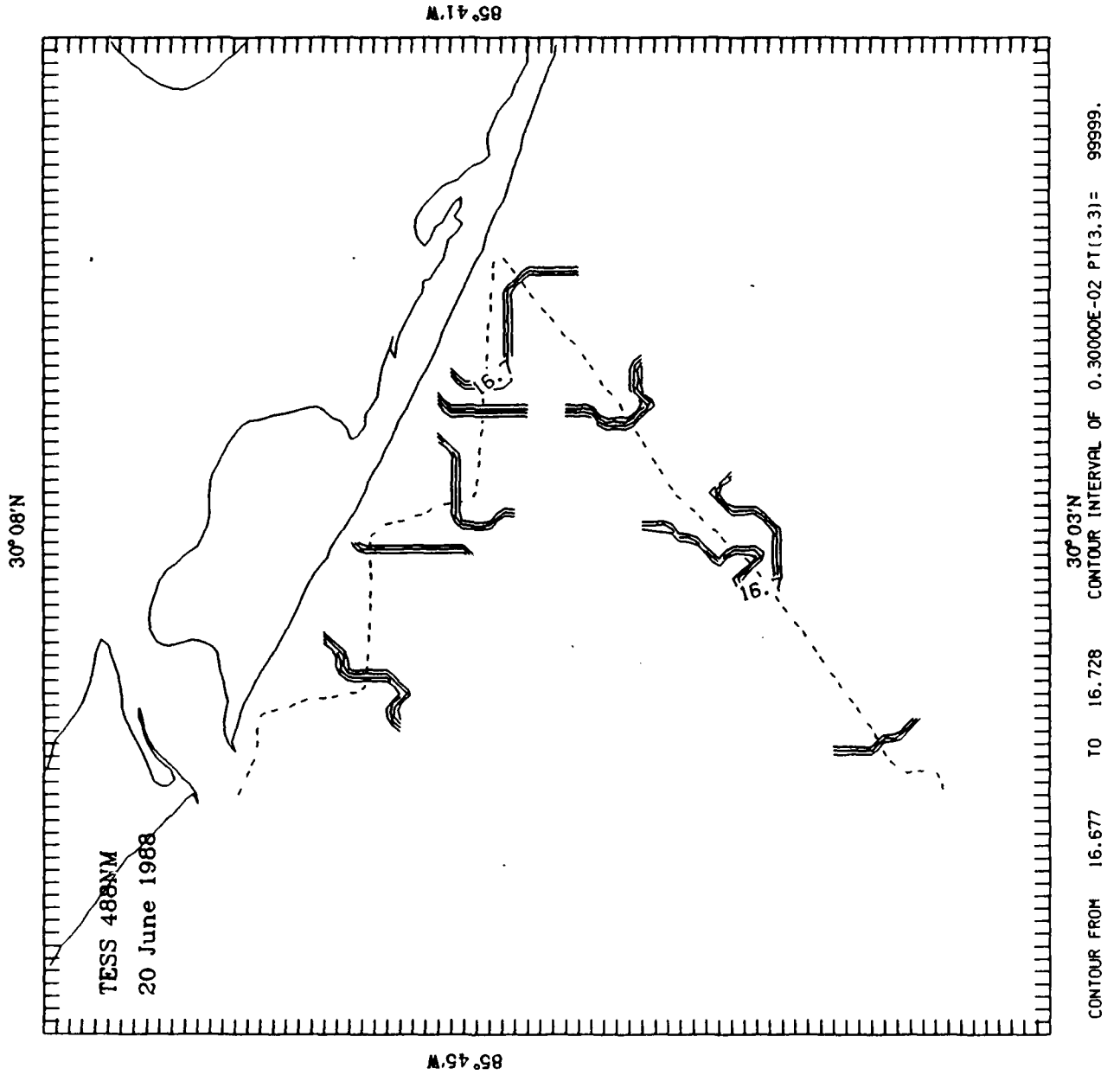


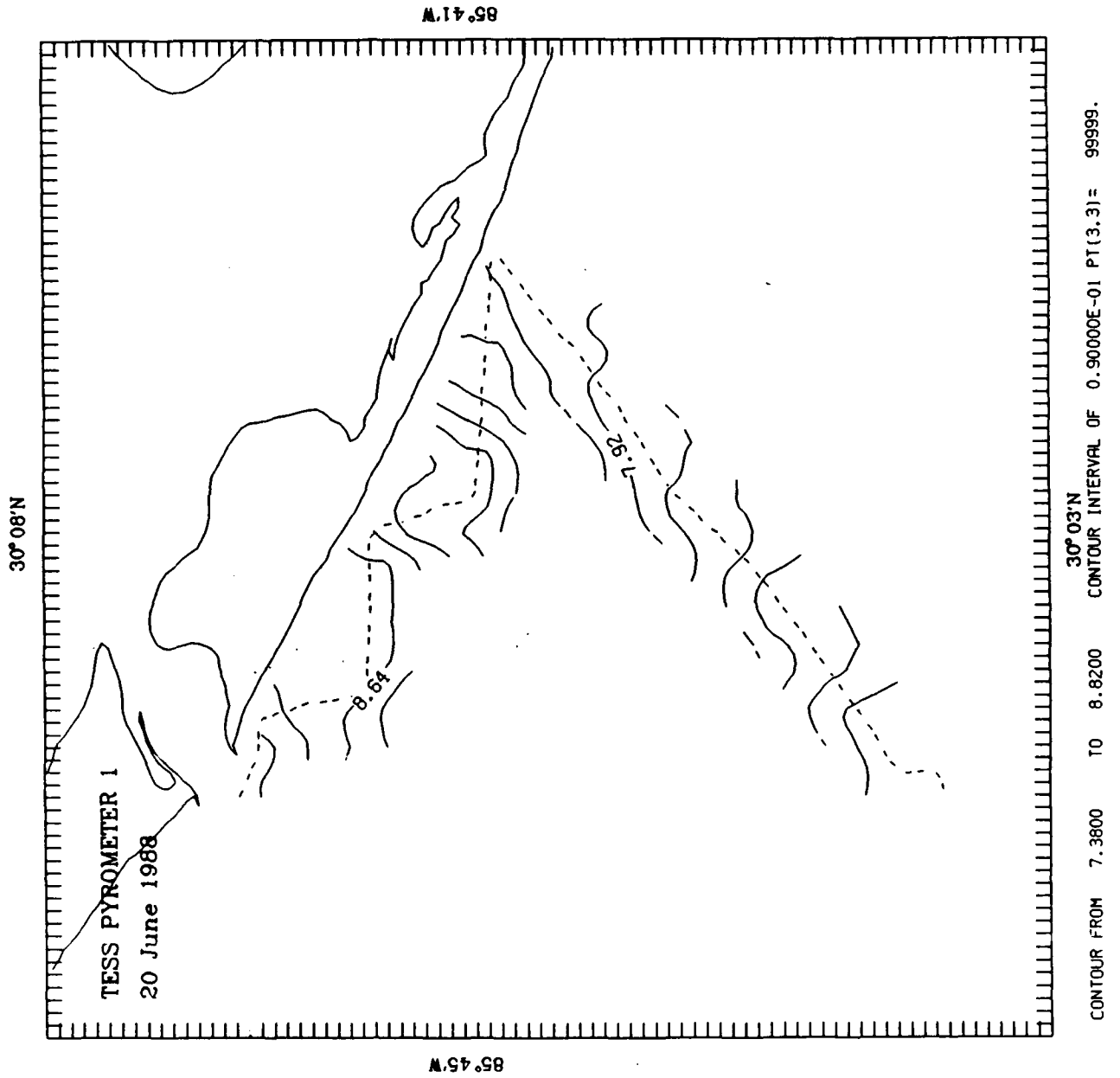


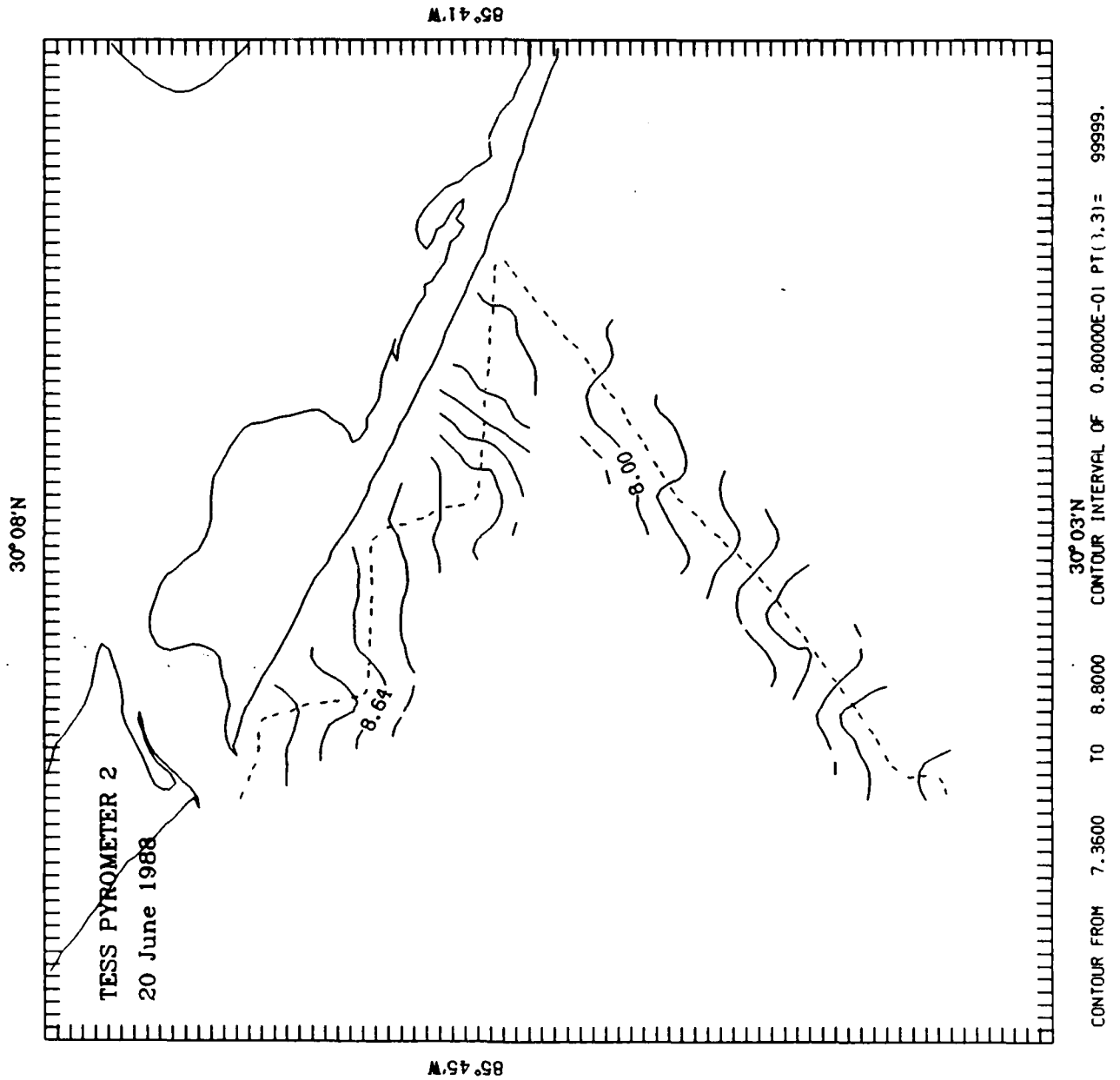


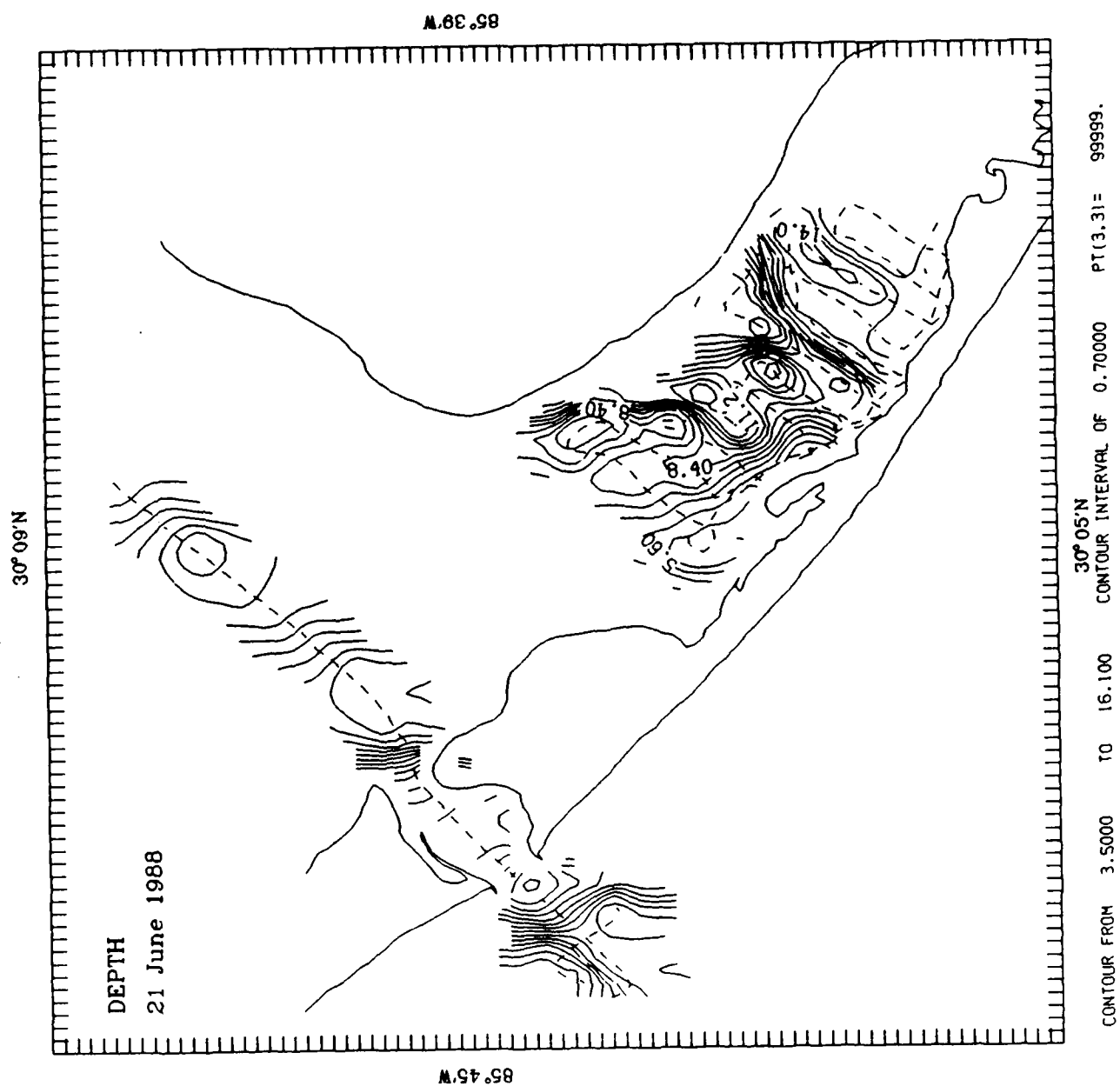


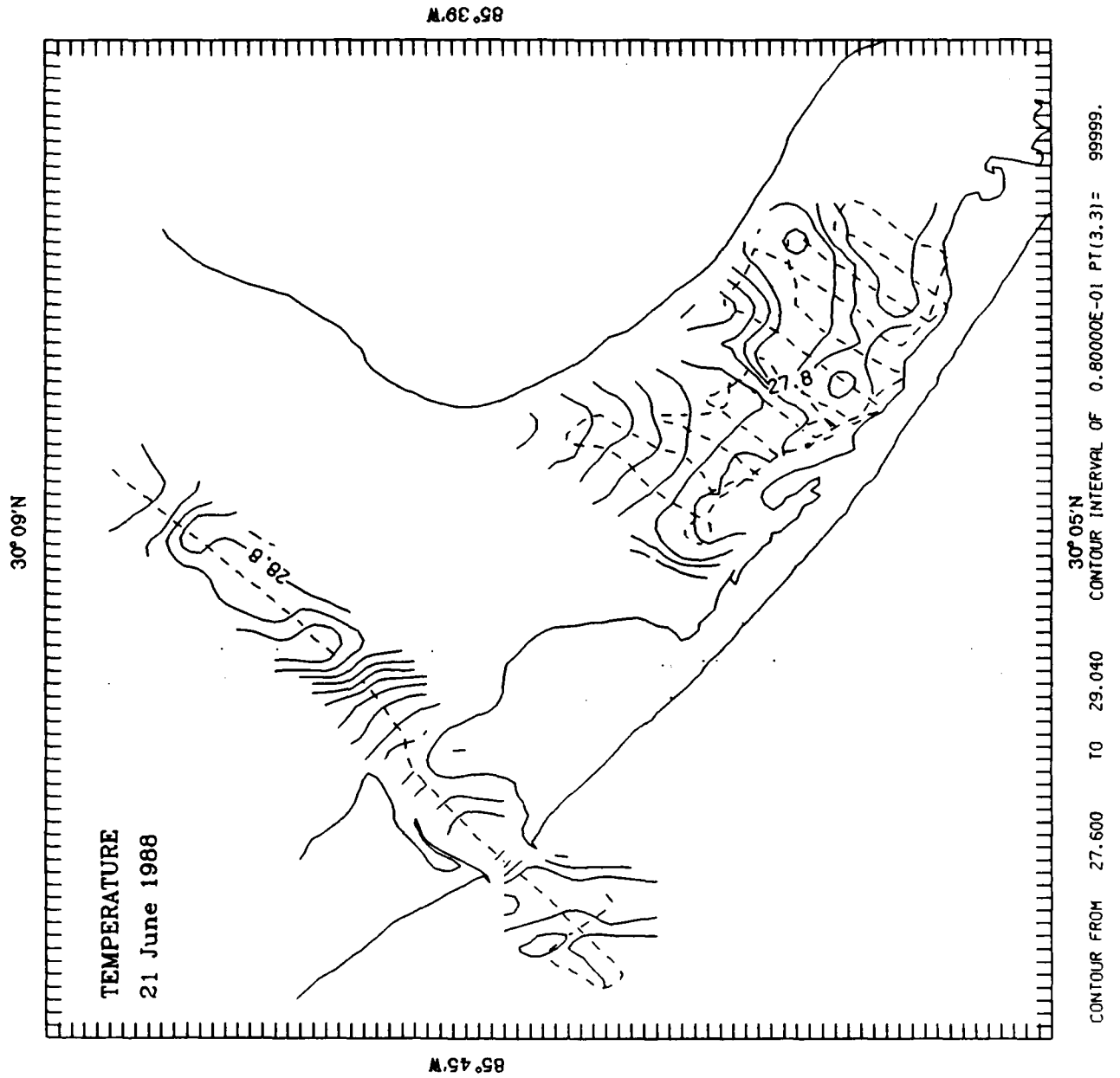


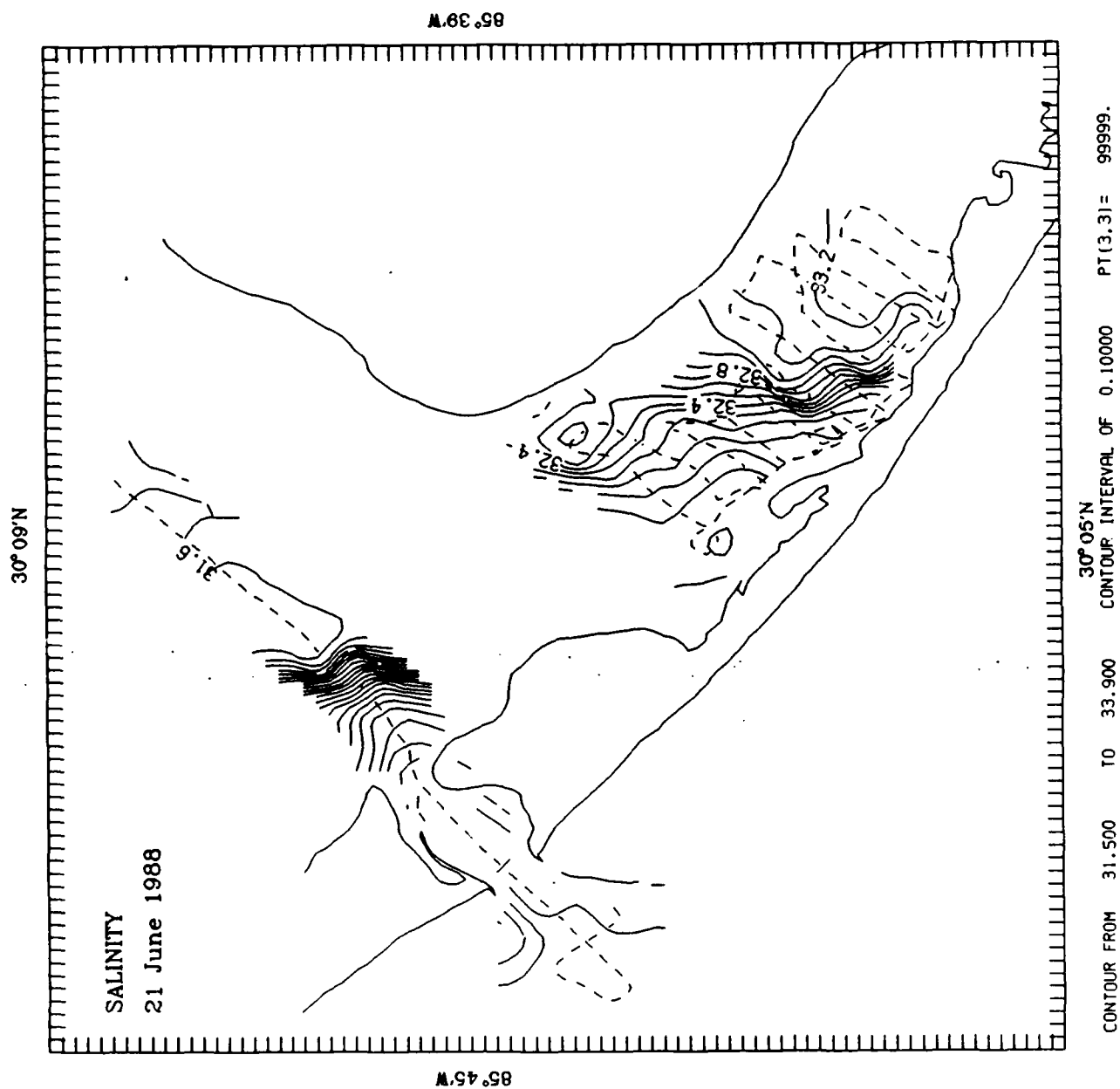


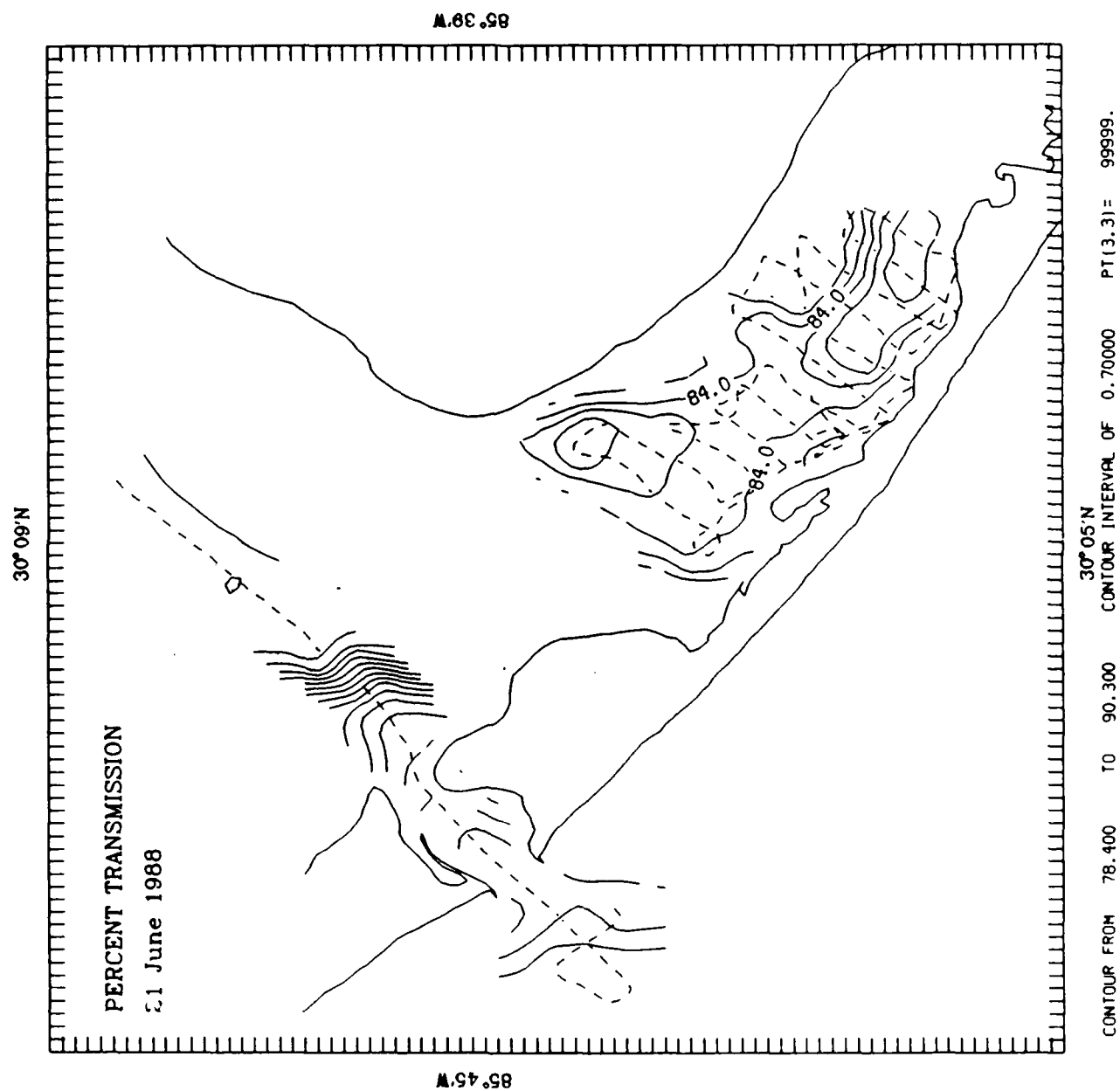


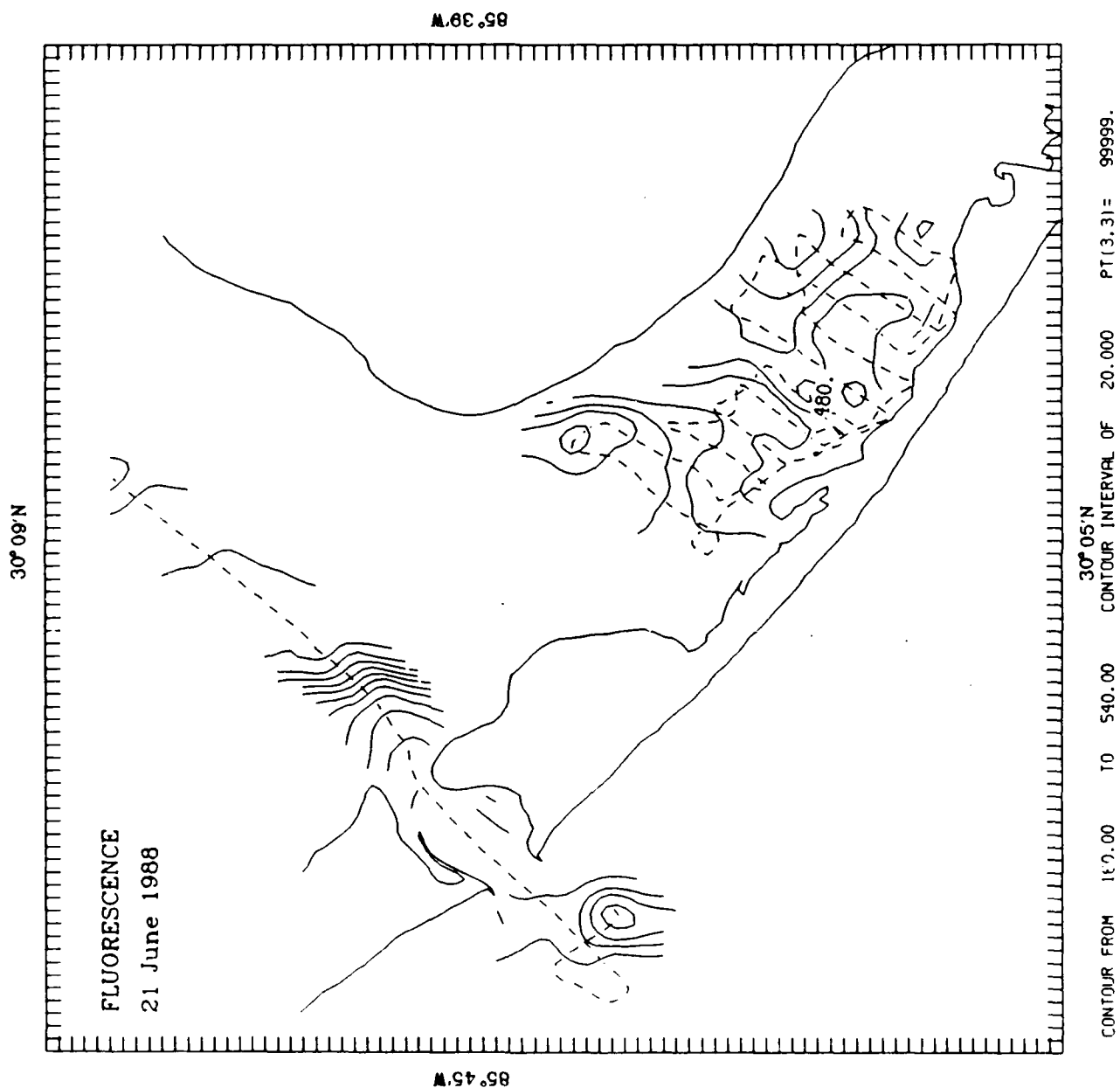


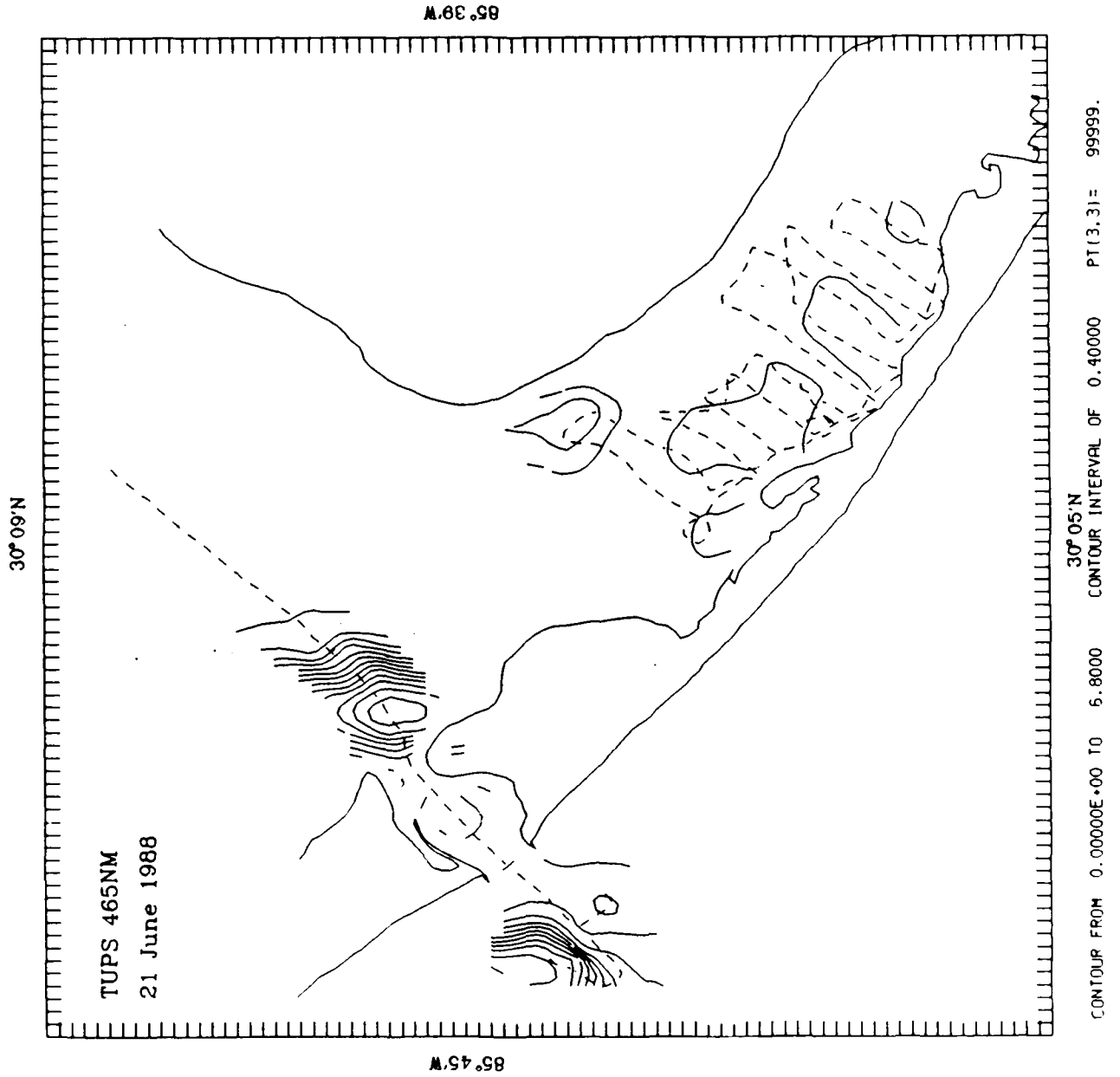


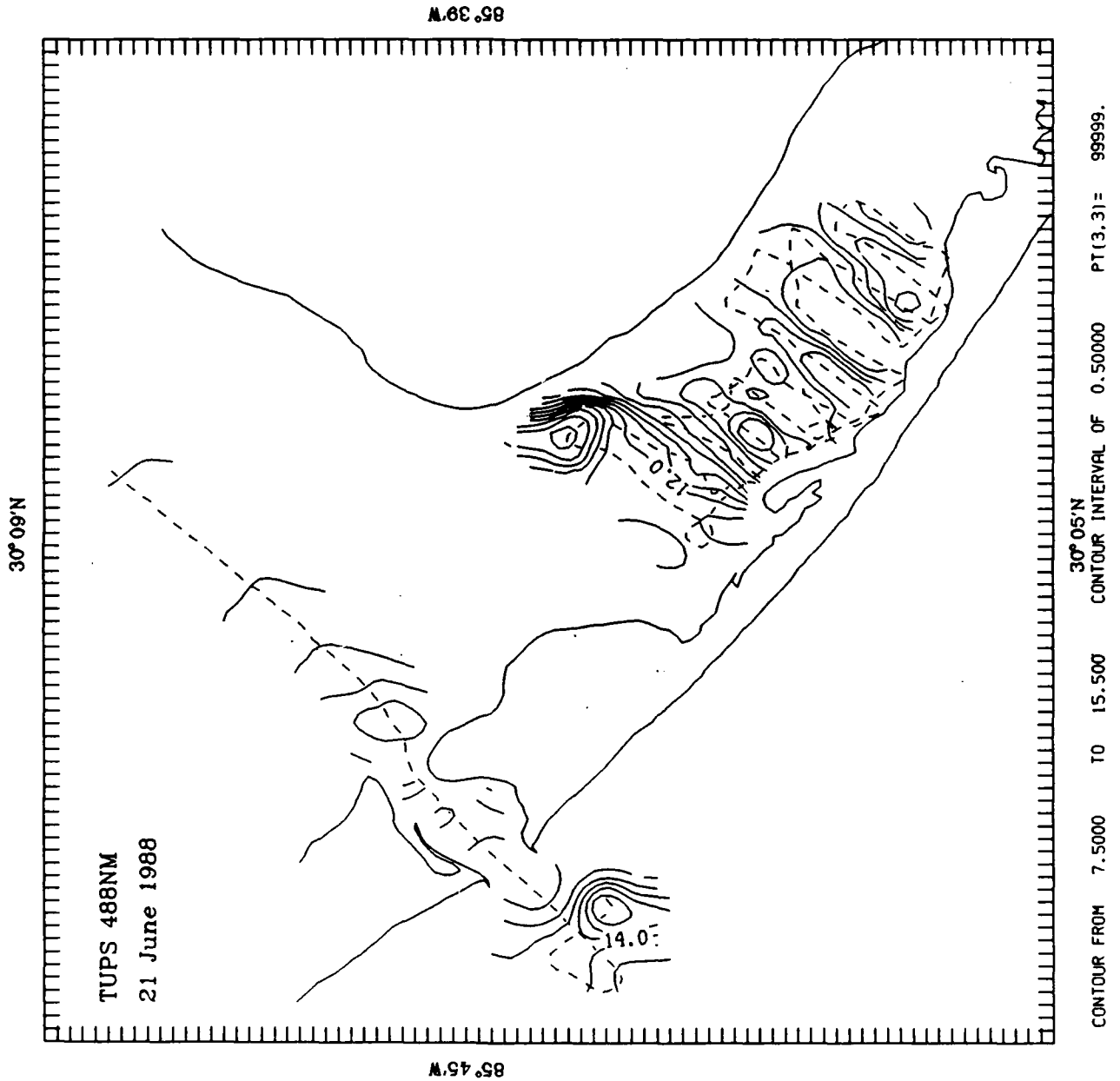


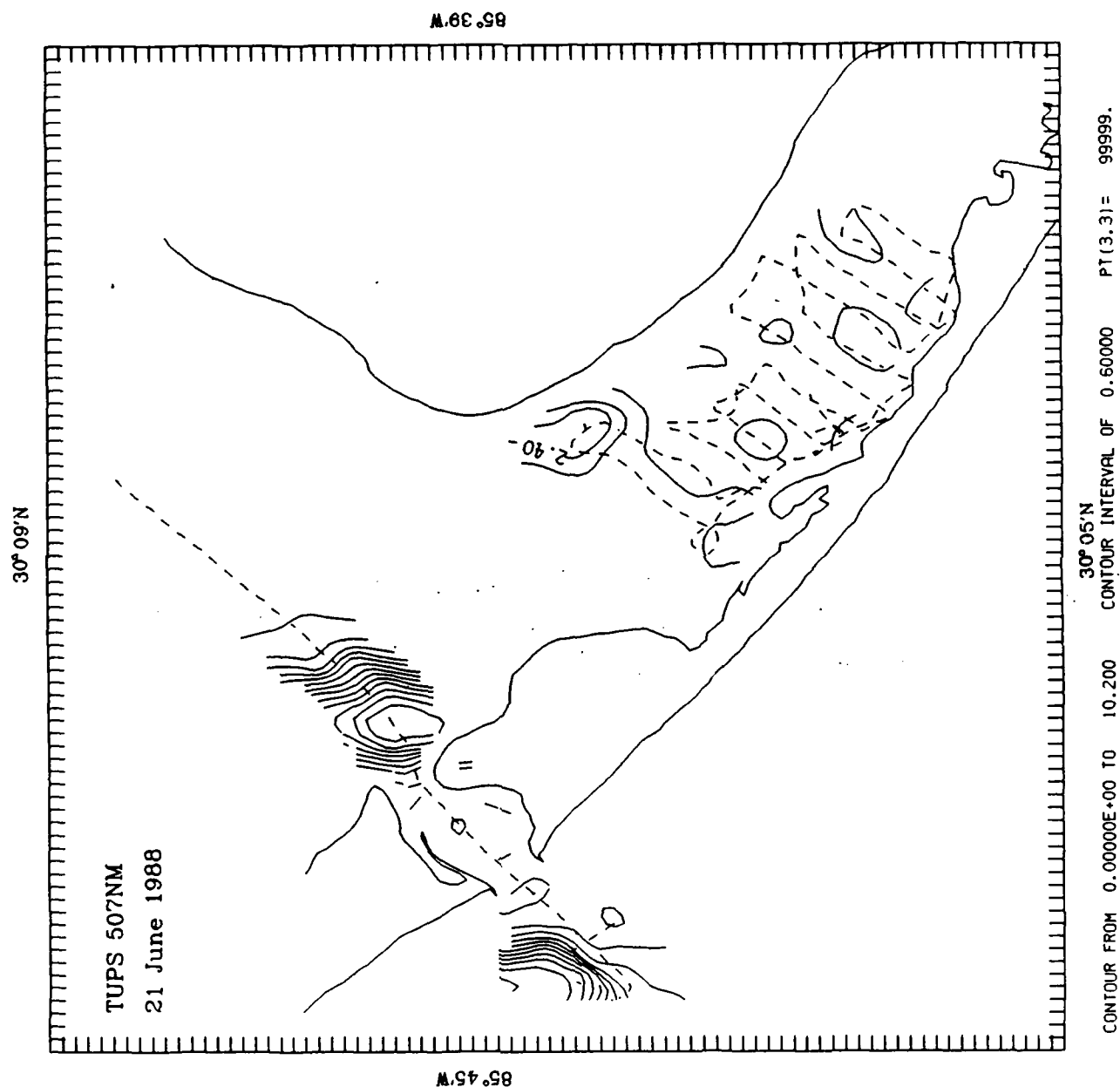


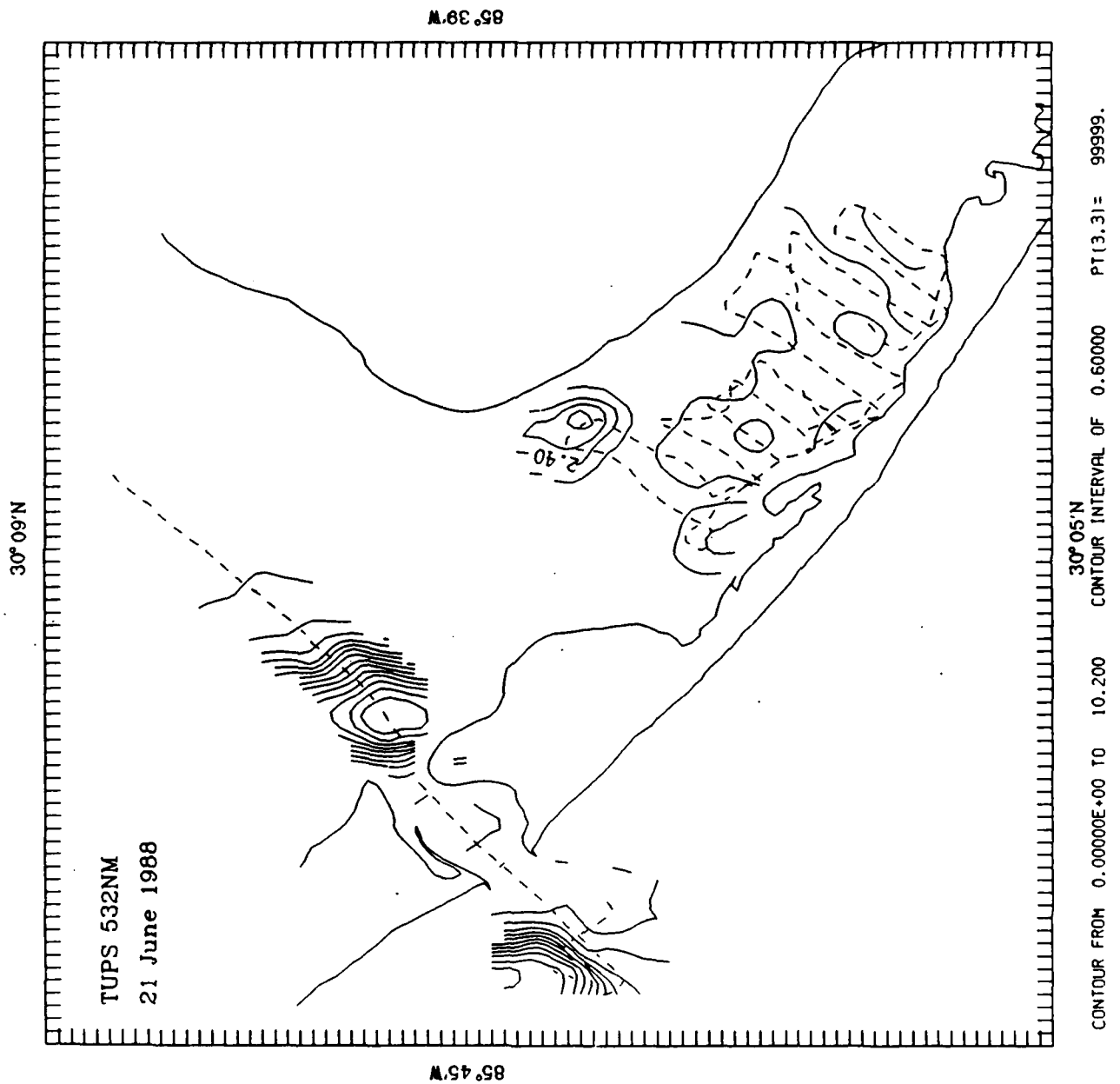


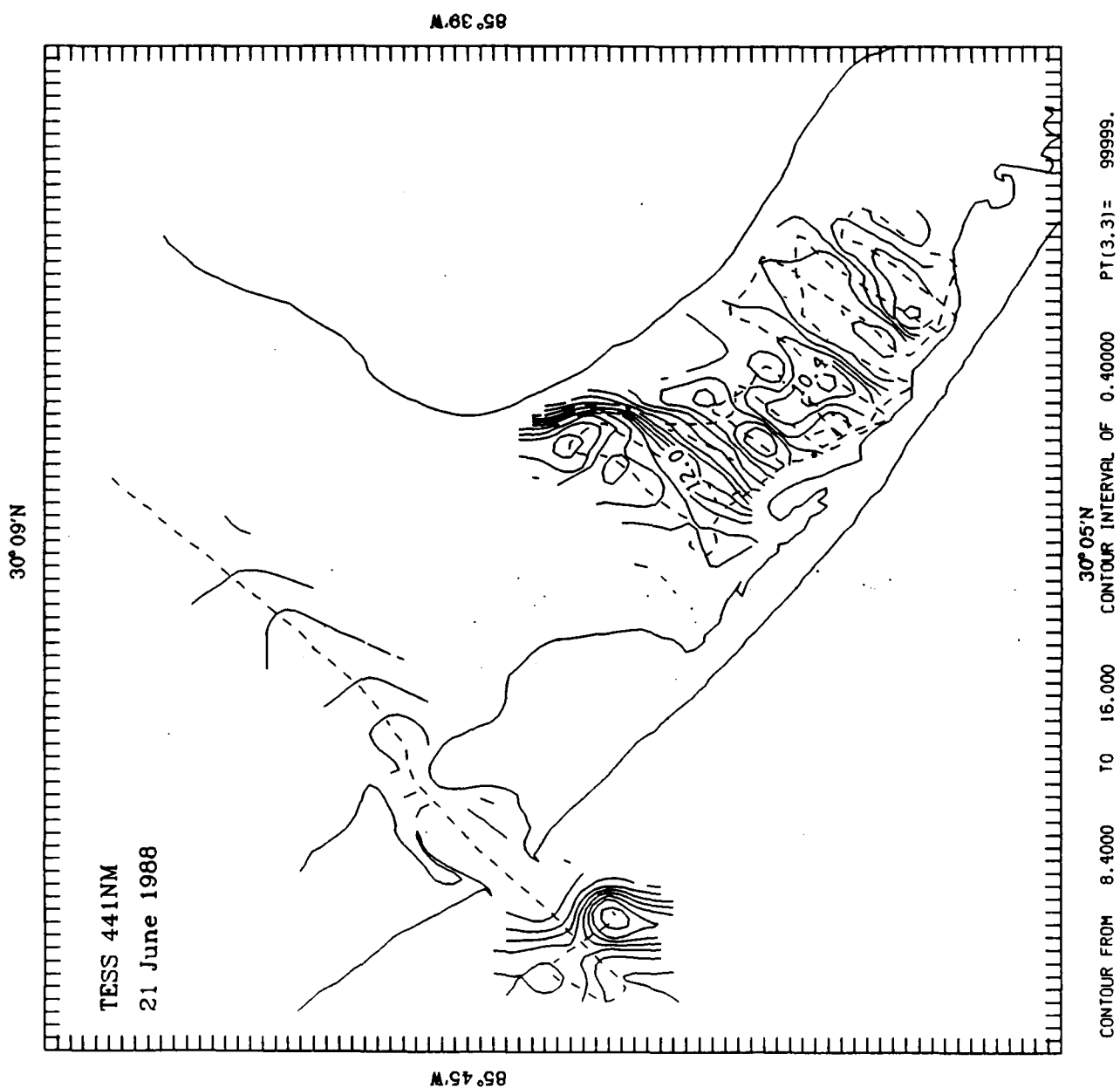


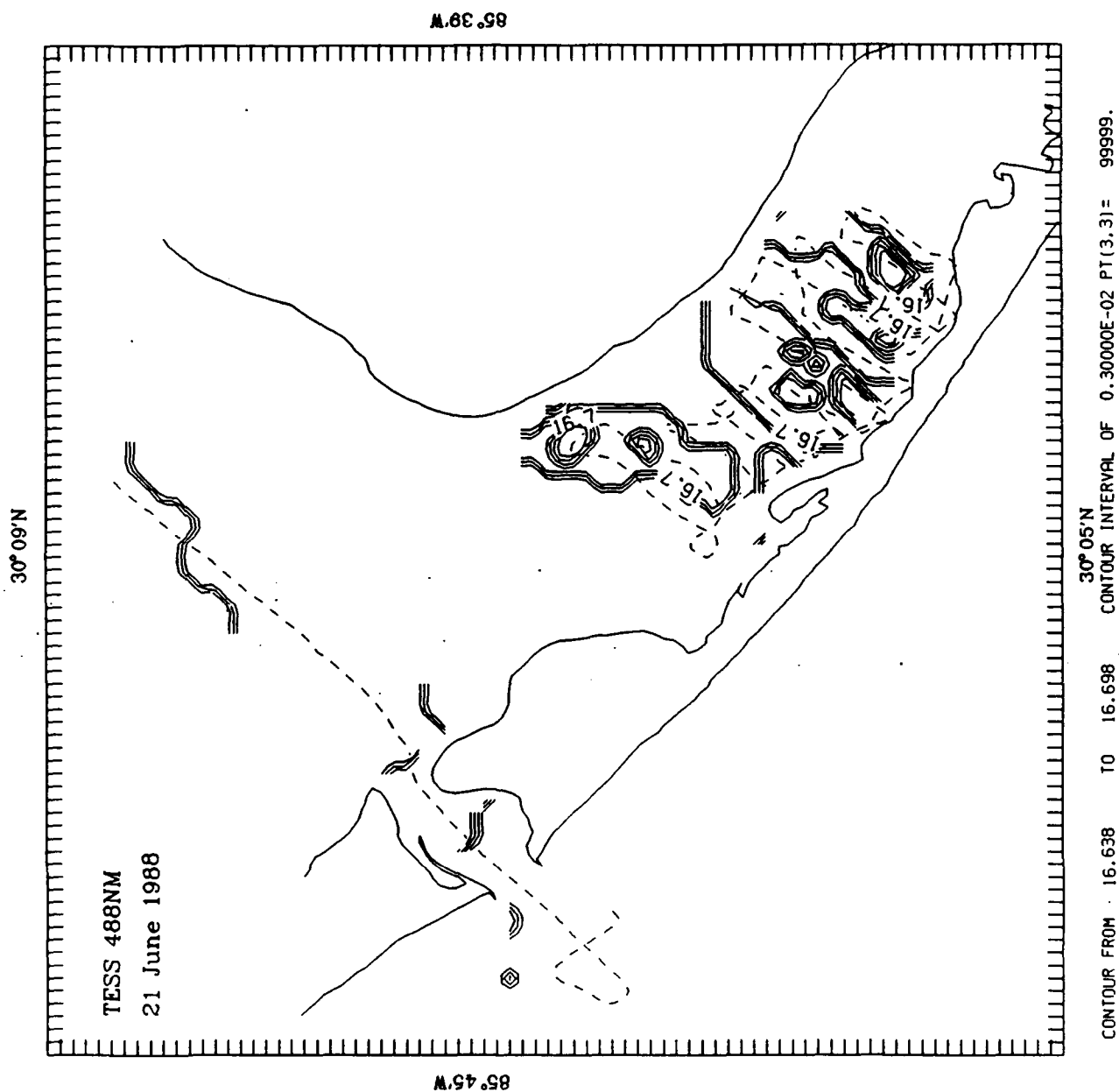


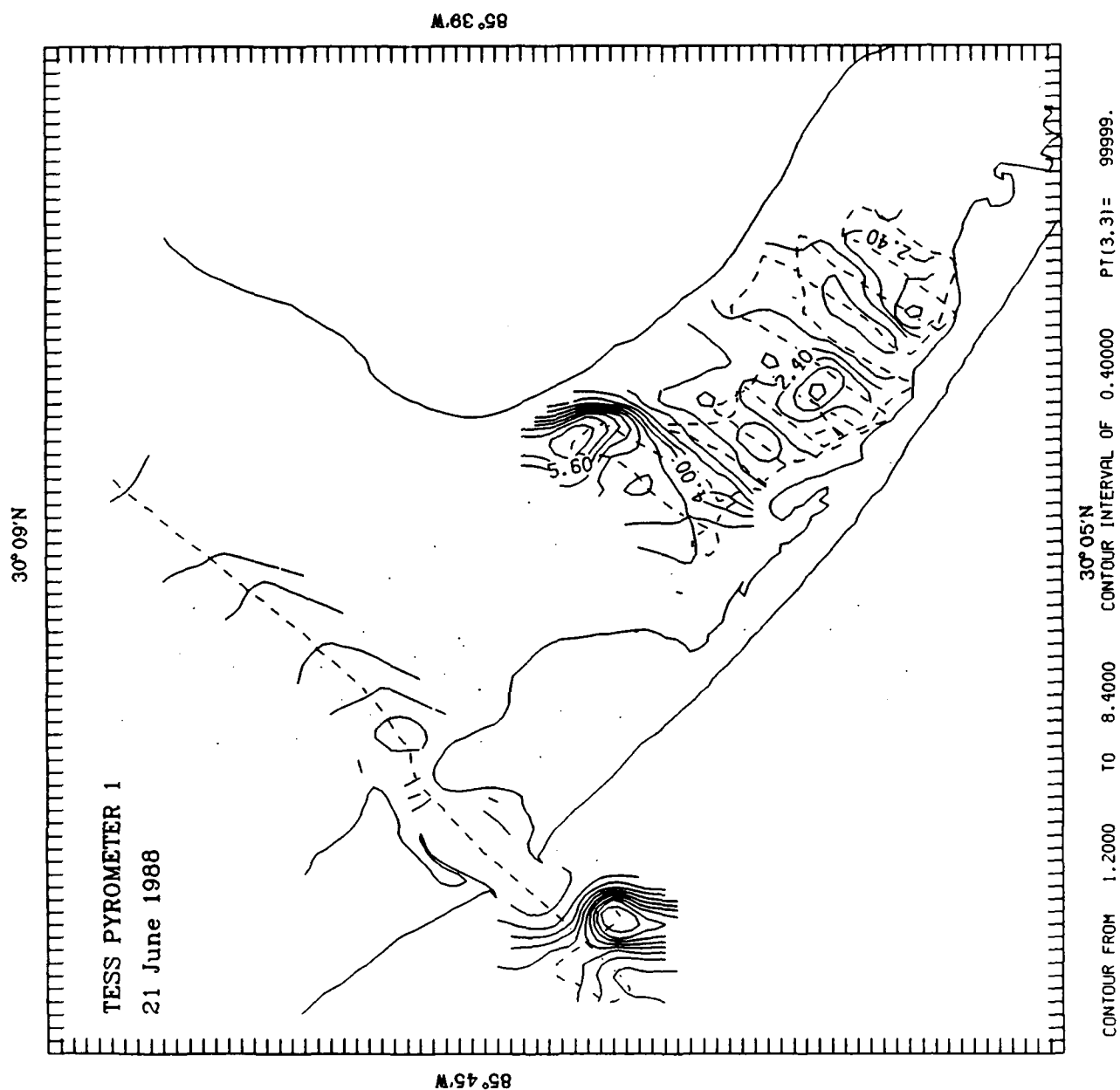


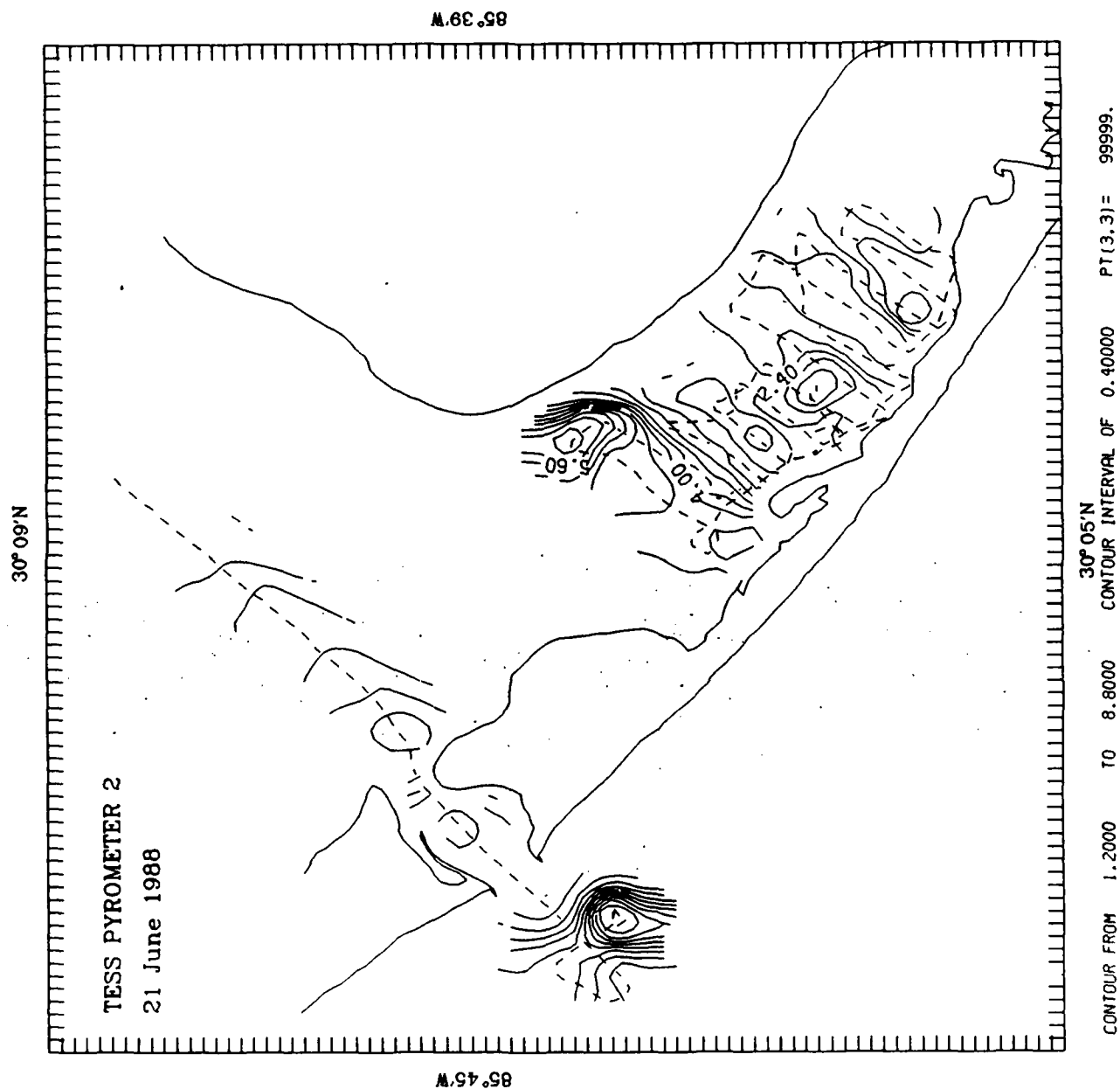












APPENDIX C

VAX 9-TRACK TAPE LISTING

arnone.lis

Thu, Dec 28, 1989

1

Listing of save set(s)

Save set: PCITY.BCK
 Written by: DENIS
 UIC: [000200,000007]
 Date: 28-DEC-1988 14:30:50.31
 Command: BACKUP [DENIS.PCITY...]/LIST=ARNONE.LIS MUB0:PCITY.BCK/SAVE_SET
 Operating system: VAX/VMS version V4.7
 BACKUP version: V4.7
 CPU ID register: 08000000
 Node name: _GERGA::
 Written on: _GERGA\$MUB0:
 Block size: 8192
 Group size: 10
 Buffer count: 3

[DENIS.PCITY]19J1600.CNV;3	166	12-JUL-1988	12:09
[DENIS.PCITY]19J1600.TIM;2	31	29-JUN-1988	08:41
[DENIS.PCITY]19J1652.CNV;3	655	12-JUL-1988	12:13
[DENIS.PCITY]19J2022.CNV;3	287	12-JUL-1988	12:14
[DENIS.PCITY]19JUN88.FIN;1	1727	28-NOV-1988	11:41
[DENIS.PCITY]19JUN88.PTO;1	933	28-NOV-1988	11:39
[DENIS.PCITY]19JUN88.TES;1	565	11-JUL-1988	11:00
[DENIS.PCITY]19JUN88.TUP;3	1106	12-JUL-1988	12:34
[DENIS.PCITY]19JUN881.RDJ;1	386	11-JUL-1988	09:51
[DENIS.PCITY]19JUN882.RDJ;1	179	11-JUL-1988	09:54
[DENIS.PCITY]20J1505.CNV;1	226	3-NOV-1988	09:19
[DENIS.PCITY]20JUN88.FIN;2	353	28-NOV-1988	11:18
[DENIS.PCITY]20JUN88.PTO;1	191	28-NOV-1988	11:17
[DENIS.PCITY]20JUN88.TES;1	247	11-JUL-1988	11:05
[DENIS.PCITY]20JUN88.TUP;1	226	3-NOV-1988	09:26
[DENIS.PCITY]20JUN881.RDJ;1	247	11-JUL-1988	10:01
[DENIS.PCITY]21J1155.CNV;1	158	12-JUL-1988	12:39
[DENIS.PCITY]21J1250.CNV;1	196	12-JUL-1988	12:40
[DENIS.PCITY]21J1353.CNV;1	118	12-JUL-1988	12:40
[DENIS.PCITY]21J1441.CNV;1	63	12-JUL-1988	12:41
[DENIS.PCITY]21J1519.CNV;1	106	12-JUL-1988	12:41
[DENIS.PCITY]21J1602.CNV;1	61	12-JUL-1988	12:42
[DENIS.PCITY]21J1630.CNV;1	170	12-JUL-1988	12:42
[DENIS.PCITY]21J1742.CNV;1	63	12-JUL-1988	12:43
[DENIS.PCITY]21J1839.CNV;1	149	12-JUL-1988	12:43
[DENIS.PCITY]21JUN88.FIN;1	1681	28-NOV-1988	12:01
[DENIS.PCITY]21JUN88.PTO;1	908	28-NOV-1988	11:59
[DENIS.PCITY]21JUN88.TES;3	577	18-JUL-1988	14:03
[DENIS.PCITY]21JUN88.TUP;1	1077	12-JUL-1988	12:55
[DENIS.PCITY]21JUN881.RDJ;1	112	11-JUL-1988	10:02
[DENIS.PCITY]21JUN882.RDJ;1	77	11-JUL-1988	10:03
[DENIS.PCITY]21JUN883.RDJ;1	136	11-JUL-1988	10:04
[DENIS.PCITY]21JUN884.RDJ;1	140	18-JUL-1988	13:33
[DENIS.PCITY]21JUN885.RDJ;1	116	18-JUL-1988	13:35
[DENIS.PCITY]ADDNUM.EXE;3	6	2-DEC-1988	08:32
[DENIS.PCITY]ADDNUM.LIS;4	5	2-DEC-1988	08:32
[DENIS.PCITY]ALL.SSD;2	4224	29-NOV-1988	11:00
[DENIS.PCITY]ALL19.NAV;1	1075	30-NOV-1988	16:40
[DENIS.PCITY]ALL20.NAV;1	220	12-DEC-1988	09:01
[DENIS.PCITY]ALL21.NAV;1	1038	12-DEC-1988	09:09
[DENIS.PCITY]ALLSUB.SSD;2	2416	22-DEC-1988	08:40
[DENIS.PCITY]ARNONE.LIS;1	3	28-DEC-1988	14:25
[DENIS.PCITY]BEND.OUT;59	2	19-JUL-1988	22:20
[DENIS.PCITY]BEND.OUT;58	2	19-JUL-1988	22:20

arnone.lis

Thu, Dec 28, 1989

2

[DENIS.PCITY] BEND.OUT; 57	2	19-JUL-1988	22:20
[DENIS.PCITY] BSAS.COM; 6	1	23-NOV-1988	07:42
[DENIS.PCITY] CONTOUR.COM; 3	1	20-DEC-1988	15:26
[DENIS.PCITY] CONTOUR.COM; 2	1	20-DEC-1988	15:18
[DENIS.PCITY] CONTOUR.COM; 1	1	9-DEC-1988	08:41
[DENIS.PCITY] CONTOUR.EXE; 121	202	21-DEC-1988	14:29
[DENIS.PCITY] CONTOUR.FOR; 159	22	21-DEC-1988	11:24
[DENIS.PCITY] CONTOUR.IOP; 9	29	29-NOV-1988	08:27
[DENIS.PCITY] CONTOUR.LIS; 122	75	21-DEC-1988	14:28
[DENIS.PCITY] CORR.CLN; 2	4	21-DEC-1988	15:37
[DENIS.PCITY] CORR.CLN; 1	4	21-DEC-1988	15:35
[DENIS.PCITY] CORR.LIS; 2	12	21-DEC-1988	15:33
[DENIS.PCITY] CORR.LOG; 4	2	21-DEC-1988	15:30
[DENIS.PCITY] DAT.DIR; 1	2	28-NOV-1988	09:49
[DENIS.PCITY.DAT] 19J1600.DAT; 1	98	24-JUN-1988	14:15
[DENIS.PCITY.DAT] 19J1652.DAT; 1	389	11-JUL-1988	14:40
[DENIS.PCITY.DAT] 19J2022.DAT; 1	170	24-JUN-1988	14:36
[DENIS.PCITY.DAT] 19JUN881.DAT; 1	586	25-JUN-1988	14:03
[DENIS.PCITY.DAT] 19JUN882.DAT; 1	272	24-JUN-1988	13:55
[DENIS.PCITY.DAT] 20J1505.DAT; 1	134	3-NOV-1988	09:14
[DENIS.PCITY.DAT] 20JUN881.DAT; 1	373	30-JUN-1988	11:11
[DENIS.PCITY.DAT] 21J1155.DAT; 1	94	24-JUN-1988	14:49
[DENIS.PCITY.DAT] 21J1250.DAT; 1	117	24-JUN-1988	14:53
[DENIS.PCITY.DAT] 21J1353.DAT; 1	70	24-JUN-1988	14:58
[DENIS.PCITY.DAT] 21J1441.DAT; 1	37	24-JUN-1988	15:01
[DENIS.PCITY.DAT] 21J1519.DAT; 1	63	24-JUN-1988	15:02
[DENIS.PCITY.DAT] 21J1602.DAT; 1	36	24-JUN-1988	15:05
[DENIS.PCITY.DAT] 21J1630.DAT; 1	101	24-JUN-1988	15:06
[DENIS.PCITY.DAT] 21J1742.DAT; 1	38	24-JUN-1988	15:11
[DENIS.PCITY.DAT] 21J1839.DAT; 1	88	24-JUN-1988	15:12
[DENIS.PCITY.DAT] 21JUN881.DAT; 1	169	24-JUN-1988	14:34
[DENIS.PCITY.DAT] 21JUN882.DAT; 1	116	24-JUN-1988	14:59
[DENIS.PCITY.DAT] 21JUN883.DAT; 1	206	24-JUN-1988	15:15
[DENIS.PCITY.DAT] 21JUN884.DAT; 1	211	18-JUL-1988	13:01
[DENIS.PCITY.DAT] 21JUN885.DAT; 1	175	18-JUL-1988	13:16
[DENIS.PCITY.DAT] CSRT.DAT; 1	1	3-NOV-1988	09:13
[DENIS.PCITY.DAT] FOR023.DAT; 1	2	4-NOV-1988	13:35
[DENIS.PCITY.DAT] FOR029.DAT; 1	1	4-NOV-1988	13:50
[DENIS.PCITY.DAT] FRAG.DAT; 1	6	7-NOV-1988	11:57
[DENIS.PCITY.DAT] FRAGMENT.DAT; 3	139	3-NOV-1988	09:06
[DENIS.PCITY.DAT] FRAGMENT.DAT; 2	134	3-NOV-1988	09:12
[DENIS.PCITY.DAT] IOP020.DAT; 1	127	4-NOV-1988	13:35
[DENIS.PCITY.DAT] TRANTEST.DAT; 1	6	7-NOV-1988	11:59
[DENIS.PCITY] DECLIST.TXT; 2	1	21-DEC-1988	10:34
[DENIS.PCITY] DECLIST.TXT; 1	1	21-DEC-1988	10:32
[DENIS.PCITY] FOR011.DAT; 2	6	1-DEC-1988	12:24
[DENIS.PCITY] FOR014.DAT; 3	17	13-DEC-1988	08:45
[DENIS.PCITY] FOR023.DAT; 27	2	22-DEC-1988	15:01
[DENIS.PCITY] FOR029.DAT; 3	1	21-DEC-1988	13:45
[DENIS.PCITY] FOR097.DAT; 65	77	21-DEC-1988	11:34
[DENIS.PCITY] FRAG.CNV; 1	10	7-NOV-1988	11:59
[DENIS.PCITY] GETMCA.COM; 3	1	1-DEC-1988	15:50
[DENIS.PCITY] IOP020.DAT; 41	130	21-DEC-1988	12:25
[DENIS.PCITY] LASER.TEK; 20	1291	21-DEC-1988	17:26
[DENIS.PCITY] LASER.TEK; 19	1348	21-DEC-1988	17:15
[DENIS.PCITY] LASER.TEK; 18	468	21-DEC-1988	15:10
[DENIS.PCITY] LN03PLUS.TXT; 1	1	20-DEC-1988	13:09
[DENIS.PCITY] LPLOT.COM; 3	1	21-DEC-1988	08:54
[DENIS.PCITY] LPLOT.COM; 2	1	21-DEC-1988	08:40
[DENIS.PCITY] LPLOT.COM; 1	1	21-DEC-1988	08:38

arnone.lis

Thu, Dec 28, 1989

3

[DENIS.PCITY]M1.EXE;2	43	2-DEC-1988	12:30
[DENIS.PCITY]M1.FOR;3	12	2-DEC-1988	12:30
[DENIS.PCITY]M1.LIS;2	28	2-DEC-1988	12:30
[DENIS.PCITY]MAK.DAT;1	6	2-DEC-1988	14:44
[DENIS.PCITY]NICE1.OUT;17	7	27-DEC-1988	14:03
[DENIS.PCITY]PCITY.INFO;4	2	28-NOV-1988	11:27
[DENIS.PCITY]PCITY.INFO;3	1	28-NOV-1988	09:55
[DENIS.PCITY]PCITY.INFO;2	1	28-NOV-1988	09:49
[DENIS.PCITY]PCITY.INFO;1	1	28-NOV-1988	09:48
[DENIS.PCITY]PCITYMAP.COF;2	1	2-DEC-1988	12:30
[DENIS.PCITY]PCITYMAP.DAT;4	64	2-DEC-1988	09:17
[DENIS.PCITY]PCITYMAP.F10;4	73	13-DEC-1988	14:05
[DENIS.PCITY]PCITYMAP.MCA;4	64	2-DEC-1988	12:21
[DENIS.PCITY]PGMS.DIR;1	3	28-NOV-1988	09:37
[DENIS.PCITY.PGMS]ADDNUM.FOR;1	1	2-DEC-1988	08:31
[DENIS.PCITY.PGMS]CNVTLT.FOR;1	2	13-AUG-1986	14:41
[DENIS.PCITY.PGMS]COLLCT.FOR;1	6	5-FEB-1988	12:51
[DENIS.PCITY.PGMS]CONREC.FOR;1	84	1-DEC-1988	13:32
[DENIS.PCITY.PGMS]CONTOUR.FOR;161	23	27-DEC-1988	12:07
[DENIS.PCITY.PGMS]CONTOUROLD.FOR;1	18	8-DEC-1988	15:19
[DENIS.PCITY.PGMS]CONVRT.EXE;1	20	28-JUN-1988	16:01
[DENIS.PCITY.PGMS]CONVRT.FOR;1	21	5-FEB-1988	13:01
[DENIS.PCITY.PGMS]CONVRT88.EXE;1	22	12-JUL-1988	12:09
[DENIS.PCITY.PGMS]CONVRT88.FOR;2	37	27-DEC-1988	11:52
[DENIS.PCITY.PGMS]CONVRT88.FOR;1	37	12-JUL-1988	12:08
[DENIS.PCITY.PGMS]CONVRTTEST.FOR;1	22	2-JUL-1986	11:28
[DENIS.PCITY.PGMS]CORR.SAS;1	2	27-DEC-1988	13:45
[DENIS.PCITY.PGMS]GRAPH.FOR;1	3	15-AUG-1986	16:33
[DENIS.PCITY.PGMS]JULIAN.FOR;1	1	22-JUN-1985	13:48
[DENIS.PCITY.PGMS]M1.FOR;1	13	27-DEC-1988	13:42
[DENIS.PCITY.PGMS]MERGETT.EXE;3	11	28-NOV-1988	11:15
[DENIS.PCITY.PGMS]MERGETT.FOR;5	9	27-DEC-1988	11:45
[DENIS.PCITY.PGMS]MERGETT.FOR;4	8	28-NOV-1988	11:14
[DENIS.PCITY.PGMS]NICE1.OUT;11	4	27-DEC-1988	14:46
[DENIS.PCITY.PGMS]OPTGRAPH.FOR;1	4	15-AUG-1986	16:35
[DENIS.PCITY.PGMS]PARSE.FOR;1	5	21-DEC-1984	08:43
[DENIS.PCITY.PGMS]PLOT.SAS;2	3	27-DEC-1988	14:45
[DENIS.PCITY.PGMS]PLOT.SAS;1	2	27-DEC-1988	13:50
[DENIS.PCITY.PGMS]PRETABLE.EXE;2	14	28-NOV-1988	11:02
[DENIS.PCITY.PGMS]PRETABLE.FOR;3	14	27-DEC-1988	12:01
[DENIS.PCITY.PGMS]PRETABLE.FOR;2	13	28-NOV-1988	11:01
[DENIS.PCITY.PGMS]PUTALL.SAS;1	2	27-DEC-1988	13:22
[DENIS.PCITY.PGMS]READ.SAS;1	3	27-DEC-1988	13:03
[DENIS.PCITY.PGMS]READ1.SAS;1	1	29-NOV-1988	15:23
[DENIS.PCITY.PGMS]READ350.FOR;1	6	27-DEC-1988	14:00
[DENIS.PCITY.PGMS]READJUN.EXE;1	10	9-JUL-1988	11:01
[DENIS.PCITY.PGMS]READJUN.FOR;2	8	27-DEC-1988	11:48
[DENIS.PCITY.PGMS]READJUN.FOR;1	8	9-JUL-1988	11:01
[DENIS.PCITY.PGMS]SET_42.FOR;1	2	21-DEC-1988	13:00
[DENIS.PCITY.PGMS]SUMMARY.SAS;1	1	29-NOV-1988	14:22
[DENIS.PCITY.PGMS]TESSPLOT.FOR;1	14	22-FEB-1988	16:18
[DENIS.PCITY.PGMS]TIMETST.EXE;1	8	29-JUN-1988	08:41
[DENIS.PCITY.PGMS]TIMETST.FOR;1	3	29-JUN-1988	09:30
[DENIS.PCITY.PGMS]TTPLOT.EXE;5	96	4-NOV-1988	11:43
[DENIS.PCITY.PGMS]TTPLOT.FOR;6	13	27-DEC-1988	14:40
[DENIS.PCITY.PGMS]TTPLOT.FOR;5	12	22-DEC-1988	14:05
[DENIS.PCITY.PGMS]TUPSPLOT.FOR;1	19	24-MAR-1988	16:36
[DENIS.PCITY]PLOT.LIS;3	21	22-DEC-1988	10:51
[DENIS.PCITY]PLOT.LOG;7	4	22-DEC-1988	10:50
[DENIS.PCITY]PLOTFLUOR.OUT;1	53	4-NOV-1988	11:03

arnone.lis

Thu, Dec 28, 1989

4

[DENIS.PCITY] PRETABLE.OUT;1	191	4-NOV-1988	11:04
[DENIS.PCITY] PROGRAMS.COM;1	1	25-OCT-1988	13:42
[DENIS.PCITY] PUTALL.LOG;6	3	12-DEC-1988	09:09
[DENIS.PCITY] READ.LOG;5	7	29-NOV-1988	10:51
[DENIS.PCITY] READ1.LOG;3	2	28-NOV-1988	13:30
[DENIS.PCITY] READ350.FOR;2	5	19-DEC-1988	13:23
[DENIS.PCITY] RSAS.COM;1	1	11-NOV-1988	11:49
[DENIS.PCITY] SAS.COM;2	1	30-NOV-1988	16:06
[DENIS.PCITY] SAS.COM;1	0	30-NOV-1988	16:05
[DENIS.PCITY] SET_42.FOR;1	2	21-DEC-1988	12:59
[DENIS.PCITY] SET_42.LIS;1	7	21-DEC-1988	13:00
[DENIS.PCITY] SUMMARY.LIS;1	2	27-DEC-1988	13:46
[DENIS.PCITY] SUMMARY.LOG;2	2	27-DEC-1988	13:45
[DENIS.PCITY] TEMP.LIS;1	1	8-DEC-1988	13:45
[DENIS.PCITY] TEMP.MASK;4	6	2-DEC-1988	17:58
[DENIS.PCITY] TEMP.TEMP;2	6	2-DEC-1988	17:20
[DENIS.PCITY] TEMP.TEMP;1	0	2-DEC-1988	17:14
[DENIS.PCITY] TTPLOT.EXE;20	107	22-DEC-1988	14:05
[DENIS.PCITY] TTPLOT.FOR;19	12	22-DEC-1988	13:26
[DENIS.PCITY] TTPLOT.INFO;1	1	27-DEC-1988	14:14
[DENIS.PCITY] TTPLOT.LIS;16	37	22-DEC-1988	14:05
[DENIS.PCITY] TTPLOT19.IOP;1	409	21-DEC-1988	17:08
[DENIS.PCITY] TTPLOT19P.IOP;2	392	22-DEC-1988	14:49
[DENIS.PCITY] TTPLOT20.IOP;8	150	21-DEC-1988	15:09
[DENIS.PCITY] TTPLOT20P.IOP;2	139	22-DEC-1988	14:17
[DENIS.PCITY] TTPLOT21.IOP;1	403	21-DEC-1988	17:23
[DENIS.PCITY] TTPLOT21P.IOP;1	384	22-DEC-1988	15:01

Total of 191 files, 33358 blocks
End of save set