

University of Washington
Department of Civil and Environmental Engineering



INSTALLATION, OPERATION AND MAINTENANCE MANUAL FOR BREAKWATER DATA ACQUISITION AND ANALYSIS SYSTEM

Derald R. Christensen



Water Resources Series
Technical Report No. 91
October, 1984

Seattle, Washington
98195

Department of Civil Engineering
University of Washington
Seattle, Washington 98195

**INSTALLATION, OPERATION AND MAINTENANCE MANUAL FOR
BREAKWATER DATA ACQUISITION AND ANALYSIS SYSTEM**

Derald R. Christensen

Water Resources Series
Technical Report No. 91

October, 1984

U.S. ARMY CORPS OF ENGINEERS
FLOATING BREAKWATER PROTOTYPE TEST MONITORING PROGRAM
CONTRACT NO. DACW67-81-C-0196

INSTALLATION, OPERATION AND MAINTENANCE MANUAL FOR
BREAKWATER DATA ACQUISITION AND ANALYSIS SYSTEM

by

Derald R. Christensen

for the

Seattle District Corps of Engineers

INSTALLATION, OPERATION AND MAINTENANCE MANUAL
FOR BREAKWATER DATA ACQUISITION AND ANALYSIS SYSTEM

TABLE OF CONTENTS

	Page
LIST OF FIGURES	iii
ACKNOWLEDGMENTS	
CHAPTER I INTRODUCTION	1
CHAPTER II SITE DESCRIPTION AND PROJECT LAYOUT	
II.1 General Field Project Layout	3
II.2 Anchoring Detail	5
II.3 Transducer Locations	
CHAPTER III INSTRUMENTATION	
III.1 Design criteria and Transducer Summary	16
III.2 Load Cell	17
III.3 Wave Monitoring	20
III.4 Pressure Sensors	23
III.5 Motions	24
a) Accelerations	
b) Relative Motions	
III.6 Wind Speed and Direction	25
III.7 Concrete Strain	25
III.8 Temperature	25
III.9 Current Speed and Direction	25
III.10 Pontoon Interconnection Forces	26
a) Rubber Connector	
b) Rigid Connection	
CHAPTER IV DATA ACQUISITION SYSTEM	
IV.1 Recording System	57
IV.2 Signal Conditioning System	57
IV.3 Operation	57
IV.4 Software/firmware	58

TABLE OF CONTENTS (Continued)

	Page
CHAPTER V ANALYSIS SYSTEM	
V.1 System Description	65
V.2 Design Criteria	65
V.3 Software	65
CHAPTER VI MONITORING SYSTEM RELIABILITY ASSESSMENT	82
CHAPTER VII CONCLUSIONS AND RECOMMENDATIONS	84
APPENDIX I RECORDING AND ELECTRONICS SYSTEM DRAWINGS	
a) Mechanical	
b) Electronic	
c) System Operation Procedures	
APPENDIX II SOFTWARE LISTINGS	
a) RCA Field Data Acquisition System	
b) CompuPro Tape Reading System	
c) CompuPro Analyst System	
APPENDIX III TAPE SUMMARY	
APPENDIX IV SCALE FACTOR SUMMARY	
APPENDIX V BOAT WAKE AND PULL TESTS	
APPENDIX VI LABORATORY TESTING	
a) Anchor Line Tests	
b) Dynamic Rubber Connector Testing	
APPENDIX VII SUPPLEMENTAL DATA	
ATTACHMENT A Manufactures Literature (Supply to CERC only)	

LIST OF FIGURES

Figure		Page
II-1.	Site Location and Tide Data	6
II-2.	Site Location Chart	7
II-3.	Location Map	8
II-4.	Bottom Contour Map	9
II-5.	Isometric Layout of Both Breakwaters	10
II-6.	Isometric Layout of Concrete Breakwater and Wave Array	11
II-7.	Anchor Plan	12
II-8.	Anchor Detail - General	13
II-9.	Anchor Detail - Concrete Breakwater	14
II-10.	Anchor Detail - Pipe Tire Breakwater	15
III-1.	Project Layout	27
III-2.	Monitoring System Diagram	28
III-3.	Load Cell Layout and Connection Detail	29
III-4 a-c.	25 Kip Load Schematic	30
III-5.	Strain Gage Layout and Voltage Output Calculation	33
III-6 a-b.	Wave Buoy Schematic	34
III-7.	Wave Buoy Anchor Attachment Detail	36
III-8.	Wave Buoy Array Anchoring Detail	37
III-9.	Wave Staff Electronic Circuit	38
III-10.	Wave Staff Calibration Curves	39
III-11.	Wave and Accelerometer Data for Spar Buoy Test (Plus and Minus One Foot Motion)	40
III-12.	Wind Wave Data for Spar Buoy and and Stationary Staff	40
III-13.	Roll Response	41
III-14.	Heave Response	41
III-15.	Autospectral Estimate# for Buoy and Staff Gages	42
III-16.	Side Pressure Transducers Mounting Detail	43
III-17.	Side Pressure Transducers Layout	44
III-18 a-b.	Acceleration Transducer Mounting Detail	45
III-19 a-b.	Relative Motion Device Schematic	47
III-20.	Concrete Strain Gage Layout	49
III-21 a-g.	Concrete Strain Gage Detail	53
III-22.	Original Rubber Connector and Load Bolt Detail	55
III-23.	Rigid Connector Load Bolt Detail	55
III-24.	Second Flexible Connector and Load Bolt Detail	56

CHAPTER I

INTRODUCTION

In February, 1981, the U.S. Army Corps of Engineers initiated a Floating Breakwater Prototype Test Program. The test was designed to obtain field information on construction methods and materials, connector systems, and maintenance problems and to measure wave transmission characteristics, anchor loads, and structural forces. The program consisted of constructing two types of floating breakwaters and extensively instrumenting them to measure structural, hydraulic, and environmental parameters which are important in assessing floating breakwater performance. Program planning, engineering, and design work were completed in September 1981, and construction and placement were completed in August 1982. Monitoring and data collection continued until January, 1984, when the last breakwater was removed from the test site.

The two types of prototype breakwaters (concrete pontoon and pipe-tire) were moored at an exposed test site in Puget Sound, Washington. An extensive array of instrumentation was installed to measure and record data which will serve as a basis for establishing and evaluating the fundamental behavior of the two breakwater types. The monitoring system utilized 74 transducers to measure 16 pertinent environmental and structural variables, that may be involved in the ultimate design and mathematical modelling of these and similar structures. Results will be used to verify and modify existing mathematical models.

Results of the prototype test program fall into two categories: field experience and monitoring measurements. The field experience provides information on construction methods and materials, on connector design, and on maintenance of the structures and overall breakwater performance. The monitoring program placed primary emphasis on measuring wave attenuation and forces acting on the structures and anchor systems in natural sea state conditions (wind-generated waves). Because the most severe wave climate impressed on floating breakwaters used in sheltered waters may originate not from wind-generated waves but from boat wakes, a limited number of boat-wake attenuation tests also were performed.

The concrete breakwater was composed of two 75-foot-long units, each 16 feet wide and 5 feet deep (draft of 3.5 feet). The pipe-tire breakwater was composed of nine 16-inch-diameter steel pipes and 1,650 truck tires fastened together with conveyor belting to form a structure which is 45 feet wide and 100 feet long.

Details of the project site, breakwater installation and anchoring system, and transducer locations are given in Chapter

II. Construction details for the breakwaters are presented in a separate report by the Corps of Engineers. Test variables for the boat-wake tests and pull test on the anchoring system are given in Appendix VII.

CHAPTER II

SITE DESCRIPTION AND PROJECT LAYOUT

II.1 General Field Project Layout

The breakwater test site was in Puget Sound off West Point at Seattle, Washington (Figure 1). The site was in an exposed location, assuring that within the short period available for testing, wave conditions would exceed design waves normally associated with sites currently considered suitable for floating breakwaters. Water depth at the site varied between 40 and 50 feet at mean lower low water (MLLW) and bottom materials consisted of gravel and sand. The diurnal tide range at the site was 11.3 feet and the extreme range was 19.4 feet. The breakwater locations are shown on Figure II-1 - II-4.

The concrete structure design was based on field and design experience from numerous floating structures now in use, available model test data, and detailed structural analysis of similar structures (Adee et al, 1976; Carver, 1979; Davidson, 1971; Hales, 1981). The pipe-tire breakwater was based on a Sea Grant funded design by Volker Harms (Harms and Westerink, 1980) and modified based on local site conditions and personal discussions with Professor Harms. Other types of floating breakwaters, such as log bundles, twin pontoons, or A-frames, were considered, but either high construction costs, lack of broad applicability, or overall test program budget limited testing to the box-type concrete float and the pipe-tire mat structures. Based on available design information, the breakwaters were sized to provide acceptable wave attenuation under conditions typical of sites where the future use of floating breakwaters is anticipated (i.e., $H_s = 2$ to 4 feet, $T = 2$ to 4 seconds). However, the structures and anchor systems were designed to withstand the maximum wave predicted for the West Point site ($H_s = 6$ feet, $T = 5$ seconds).

Concrete Breakwater. The basic design of the concrete breakwater was adapted from the structure planned for a Corps of Engineers', Seattle District, project at Friday Harbor, Washington, but with modifications to accommodate the more severe wave climate at the West Point test site. It consisted of two rectangular units or modules, each 75 feet long and 16 feet wide, with a draft of 3.5 feet and a freeboard of 1.5 feet. The units were cast from lightweight, 5,000-pound-per-square-inch (p.s.i.) compressive strength concrete, and reinforced with welded wire mesh. Each module was subdivided into eight compartments by intersecting walls running the length and width of the module interiors. The interior walls were 3 inches thick, and the module sidewalls, top, and bottom were 4.75 inches thick. Interior forming of the walls was accomplished through the use of carefully cut styrofoam blocks, which also provide positive

buoyancy in the event of hull leakage. The 14- and 16-foot end compartments of each module were emptied of foam after the concrete pour and made accessible through a 23-inch water tight manhole in the deck. These compartments were necessary for access during structural modifications of the modules and for providing an area for adding trim ballast to the completed breakwater.

Initially the units were connected together by flexible connectors. Later, they were rigidly connected to form a continuous float of 150-foot length. The purpose of this procedure was to permit field development and investigation of the behavior of the connectors and measurement of design parameters for longer structures which are more representative of some recent breakwater designs. The initial flexible connection was made of extruded, butyl-rubber, marine fenders anchored in the corners of the reinforced concrete module end walls. The rigid module connection was made by using twenty 1 1/4-inch diameter steel bar tendons inserted through the end walls of the modules. The tendons were threaded at both ends to allow proper tensioning of the connection. The concrete pontoon breakwater is shown schematically in Figure II-5.

Pipe-Tire Breakwater. The breakwater was a floating mat of flexibly interconnected used truck tires. Styrofoam filled steel pipes provided stiffness in the direction of wave motion and served as buoyancy chambers. The orientation of pipes with respect to the incident wave train is shown in the artist's conception on Figure 5, with major structural features of the breakwater given in an additional report on this project. The steel pipes used in the breakwater were 45 feet in length and 16 inches in diameter, with a wall thickness of 1/4 inch. Truck tires ranged in size from 9.00-18 to 10.00-20, with an average outside diameter of 40 inches. A 3-ply conveyor belt strip, 5.5 inches wide by 0.5 inch thick (rated breaking strength of 17,000 pounds), served as binding material. A five nylon bolt connection was used to tie each belt into a continuous loop. A basic pipe-tire breakwater module was composed of two parallel pipes anchored 12 feet apart and armored with 66 tires each. Twelve rows with 11 tires in each row were fastened between the pipes with the conveyor belting. The test breakwater consisted of eight modules to produce a breakwater 45 feet wide by 100 feet long. The buoyancy of the pipe-tire breakwater was calculated from the difference between the total buoyant force and the weight of the breakwater with its mooring system attached. The styrofoam filled pipes contributed approximately 35 pounds of buoyant force per linear foot of pipe. The other major source of buoyancy was the foam in the crowns of 880 tires, originally designed to provide 75 pounds per tire under ideal conditions. The weights used for buoyancy calculations were: (1) 15 pounds (submerged weight) per tire for 770 unfoamed tires; (2) 11 pounds per tire for sediment and biofouling; (3) 1,000 pounds for submerged anchor lines, 1,700 pounds for belting; and (4) 14,500 pounds for vertical component of anchor load with design wave loading and wind/tidal current force applied. The buoyant force

and weights combine to produce a net positive buoyancy of about 33,000 pounds.

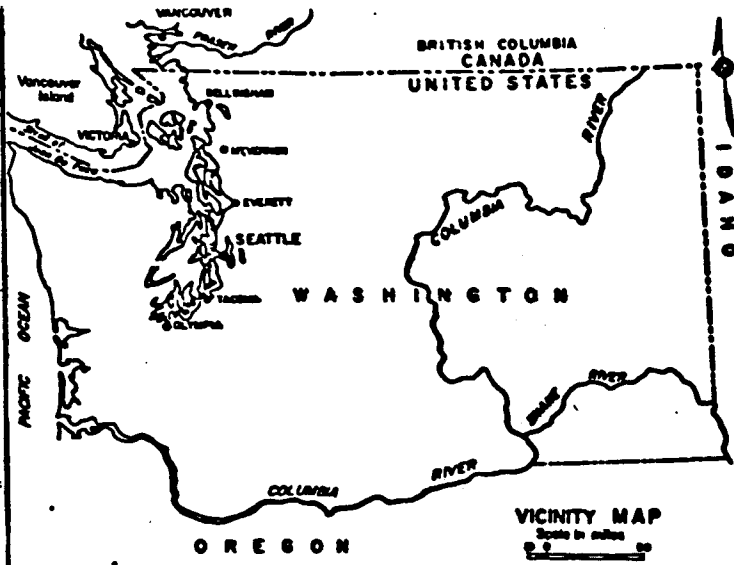
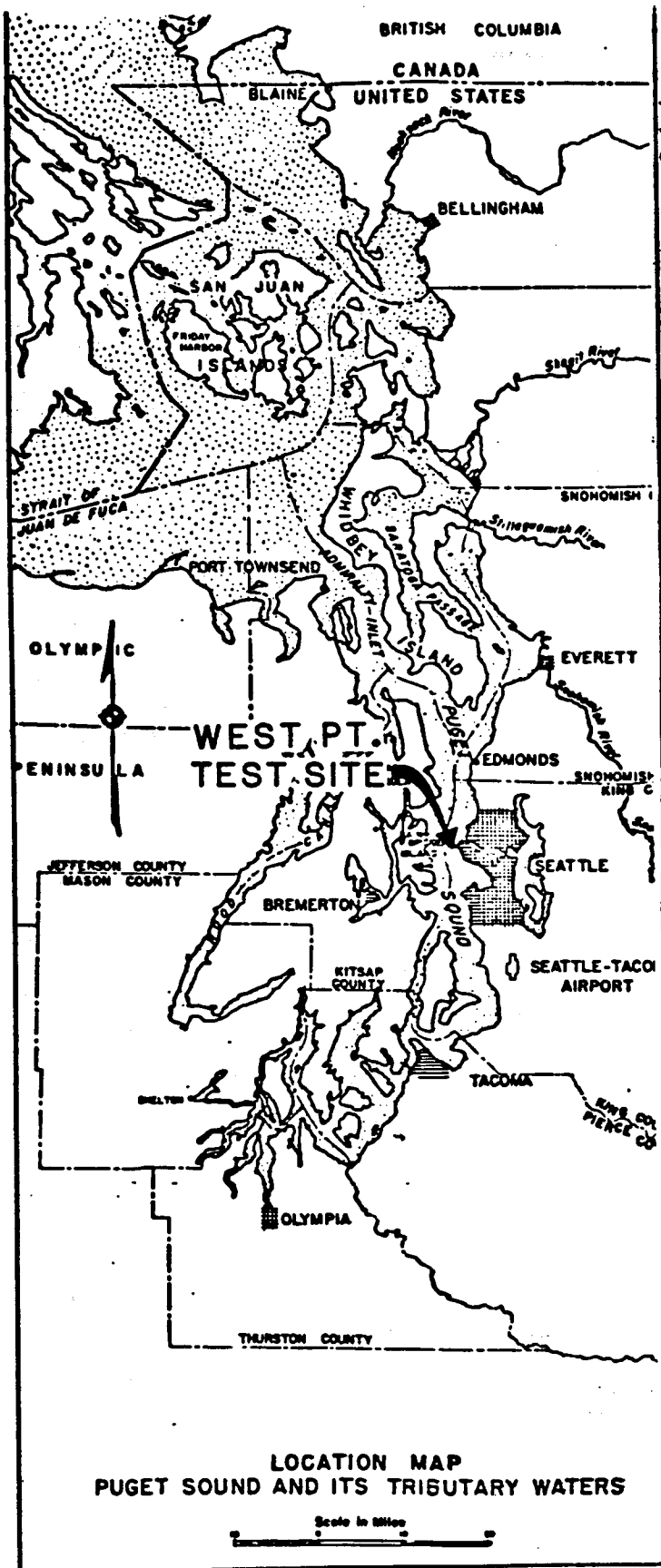
II.2 Anchoring System

The concrete breakwater was anchored in place by using ten 30-foot-long steel H-piles (HP 14 by 102) embedded their full length. Anchor lines consisted of 1-3/8-inch-diameter galvanized bridge rope with 30 feet of 1-1/4-inch stud link chain at each end and a 2,000-pound clump weight attached to minimize lateral displacement of the breakwater. Anchor line lengths were sized to provide a minimum scope of 1 vertical to 4.5 horizontal. Positioning of the four end H-pile anchors at a 5 degree inward angle provided resistance to longitudinal displacement of the breakwater. The bridge rope anchor line was chosen because, when combined with an anodic corrosion protection system (deemed necessary for more permanent structures), it appears to be the most cost effective design when maintenance over a 50-year life is taken into account.

The steel H-piles were considered the most suitable type of anchor for this project because: (1) the cost of steel pile anchors was considerably less than alternatives such as gravity (concrete block) or ship-type (stockless) anchors, (2) the consequences of dragging a gravity or ship-type anchor over either a sewer outfall lying to the north or a cable area to the south were potentially very serious, and (3) the effectiveness of various anchor types was already well documented, and since all anchors depend directly on foundation conditions, any data on anchor efficiency would have been site specific.

The anchors for the pipe-tire breakwater were ten 20-foot long steel H-piles (HP 12 by 53) embedded their full length. Anchor lines consisted of 1-1/4-inch, three strand, nylon rope with 10 feet of 3/4-inch stud link chain at each end. Minimum scope for these anchor lines was about 1 vertical to 4 horizontal. The center and end H-piles had one anchor line each, while the remaining four anchor piles were attached to three anchor lines a piece. The four end pilings were offset at an outward angle to counteract the opposing longitudinal component of force from the adjacent anchor lines.

The anchoring arrangement for the two breakwaters is indicated in plan view on Figure 7. Mooring system details of the concrete and pipe-tire breakwaters are shown on Figures II-8 - II-10, respectively.



TIDE DATA	
SEATTLE	
HIGHEST TIDE.	14.80 FEET.
MEAN HIGHER HIGH WATER.	11.32
MEAN HIGH WATER.	10.47
MEAN (HALF) TIDE LEVEL.	6.65
MEAN SEA LEVEL.	6.25
MEAN LOW WATER.	2.82
MEAN LOWER LOW WATER.	0.00
LOWEST TIDE.	-4.60 ↓

Figure II-1. Site Location and Tide Data

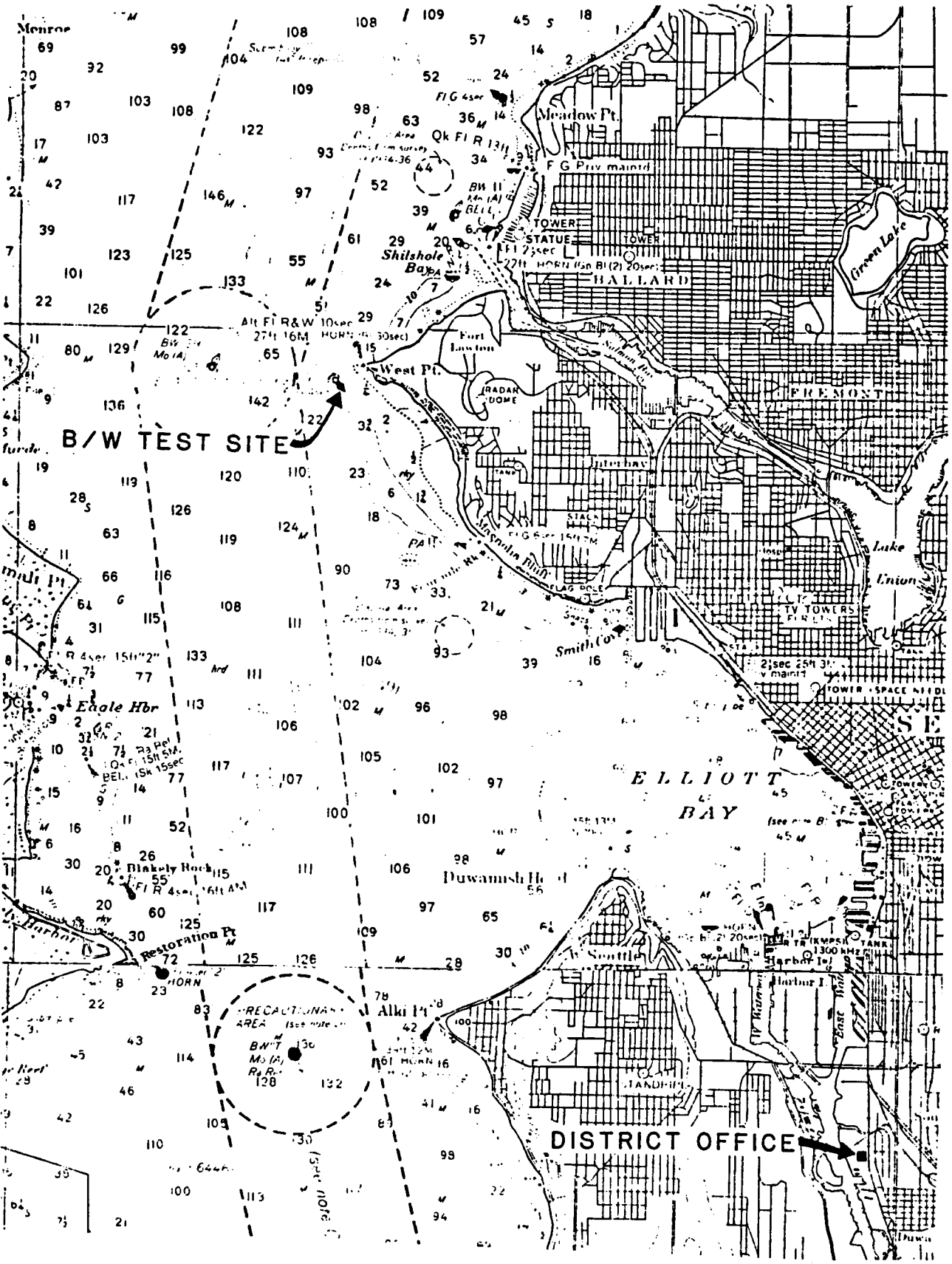


Figure II-2. Site Location Chart

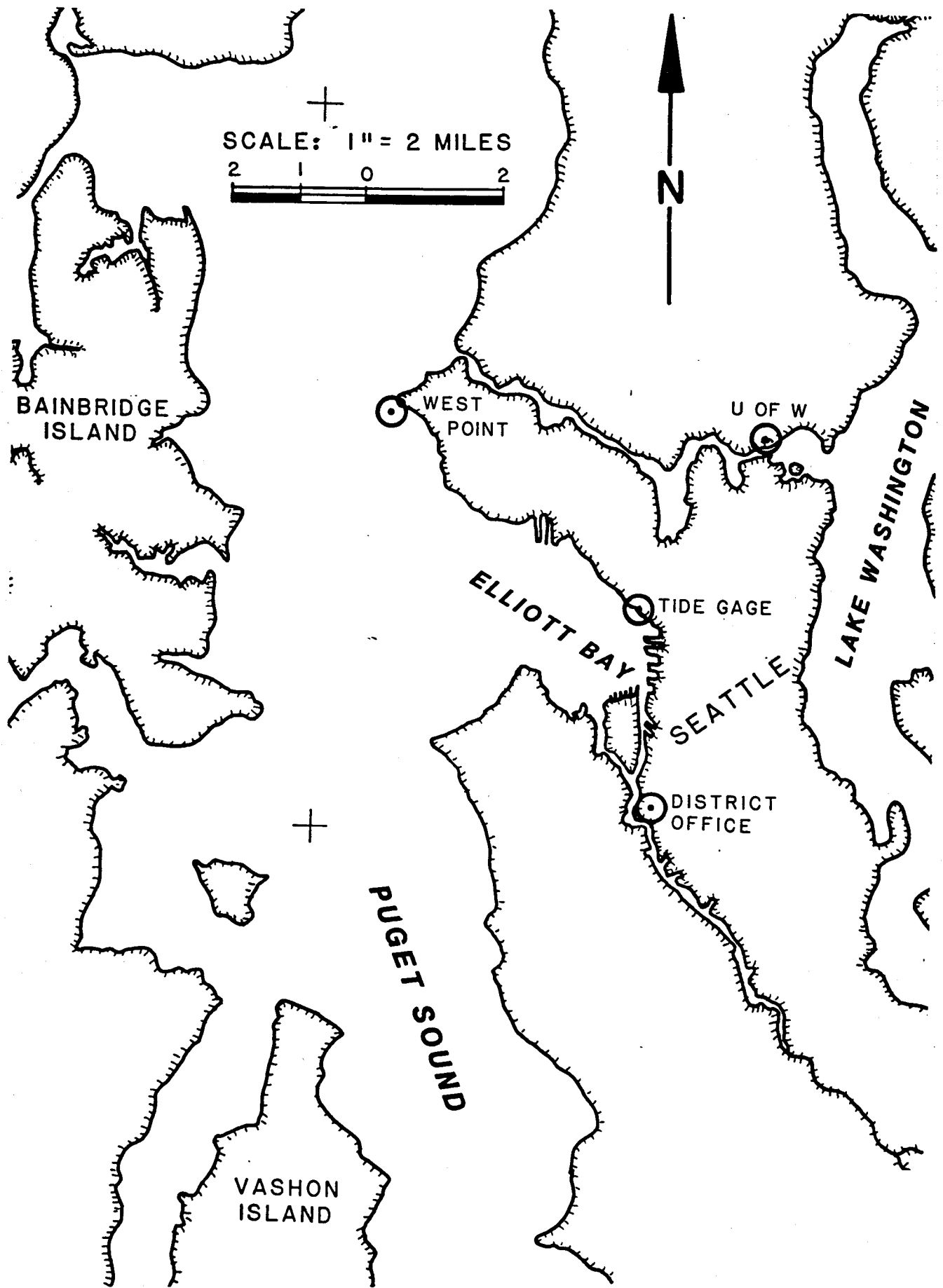


Figure II-3. Location Map

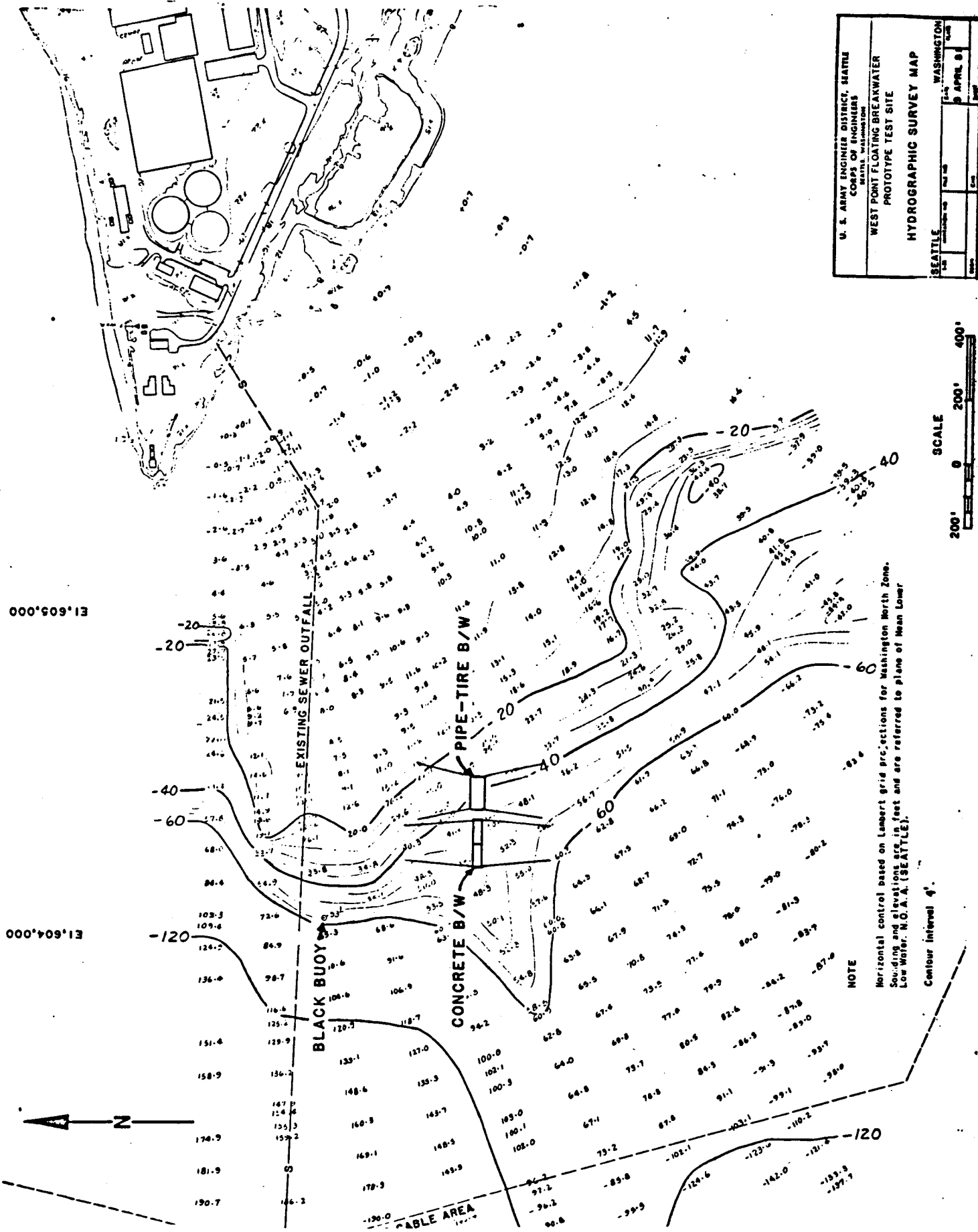


Figure II-4. Bottom Contour Map

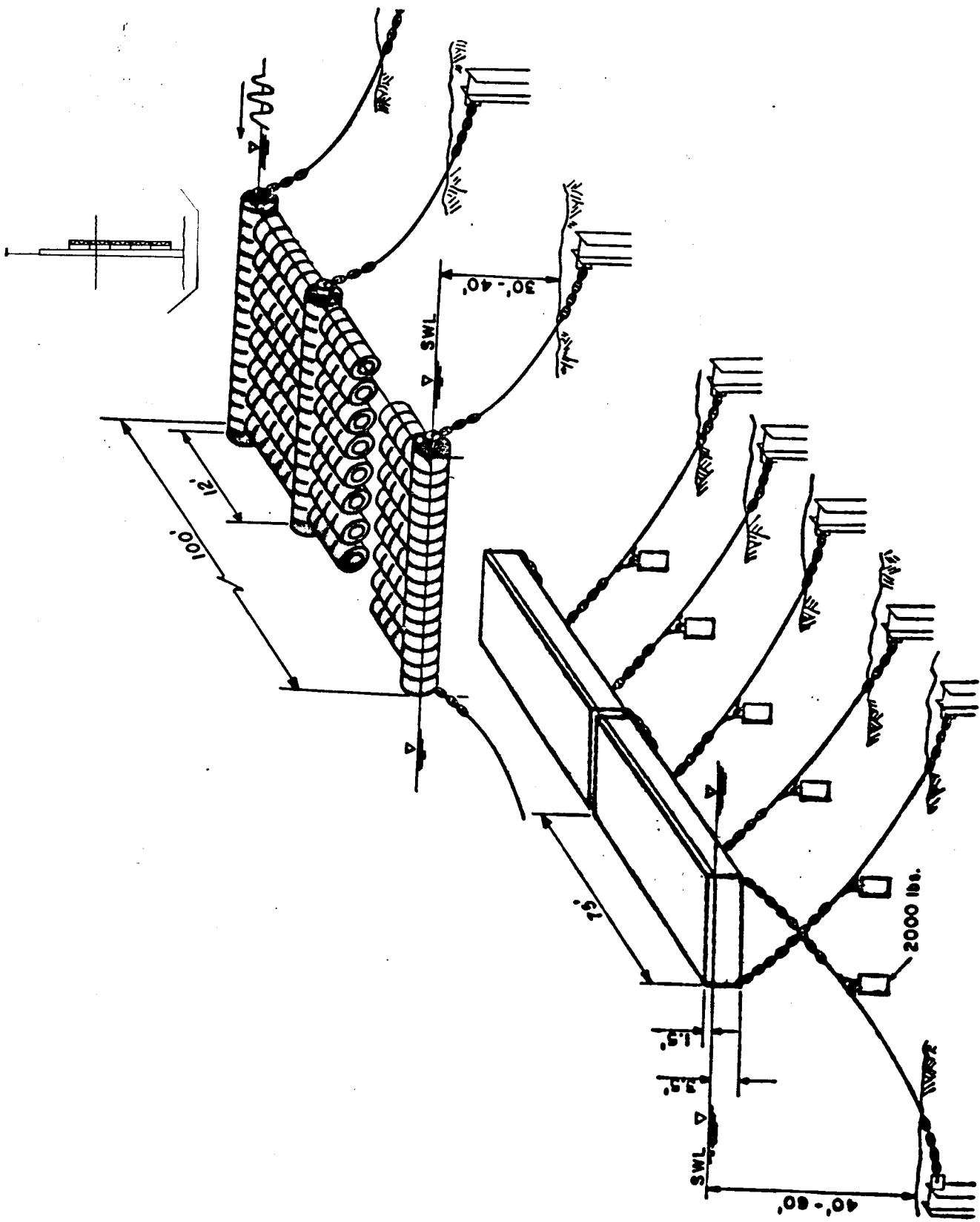


Figure II-5. Isometric Layout of Both Breakwaters

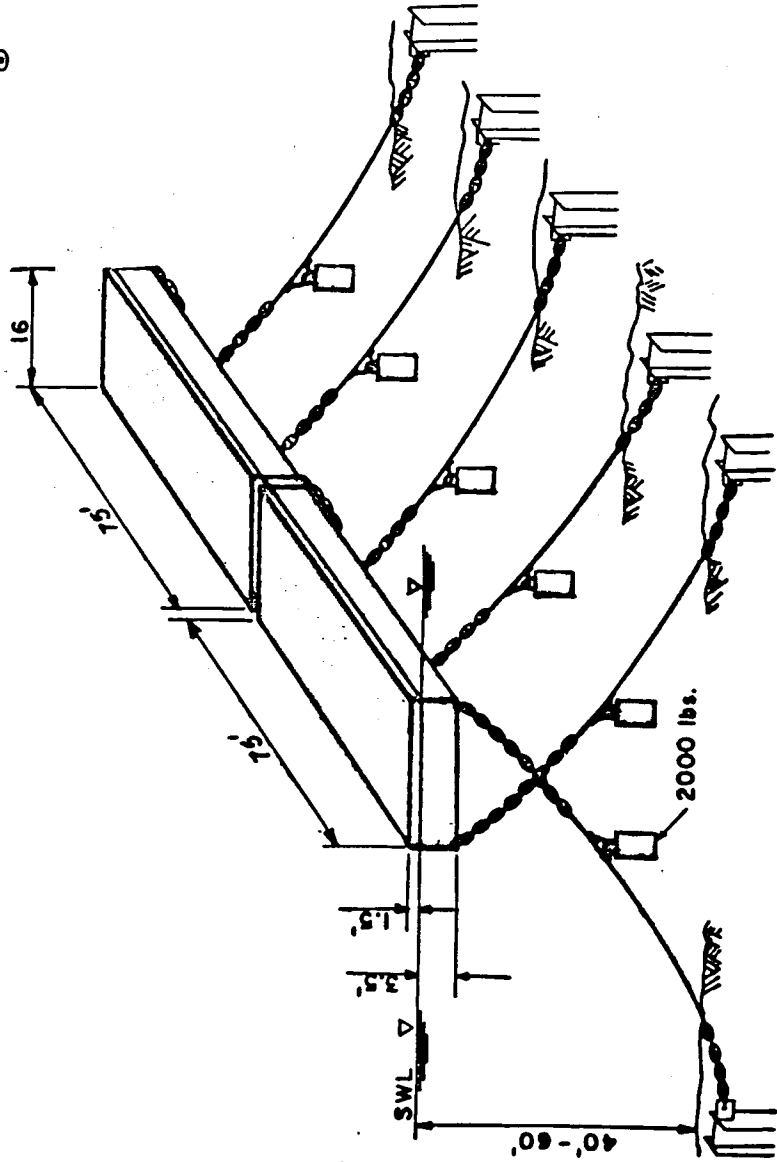
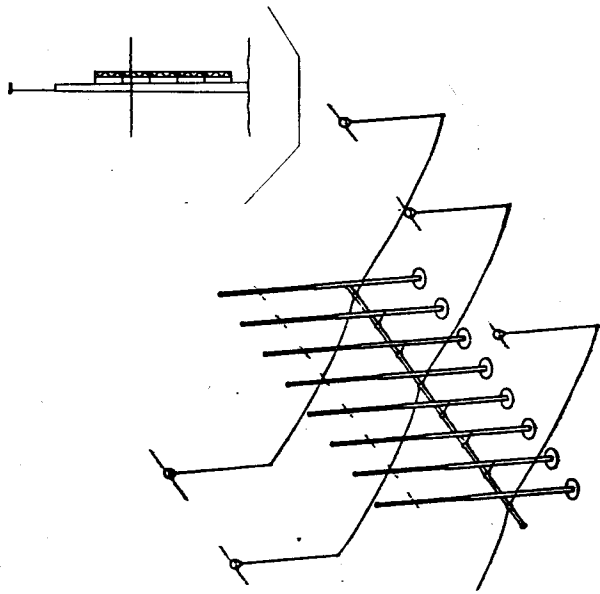


Figure II-6. Isometric Layout of Concrete Breakwater and Wave Array

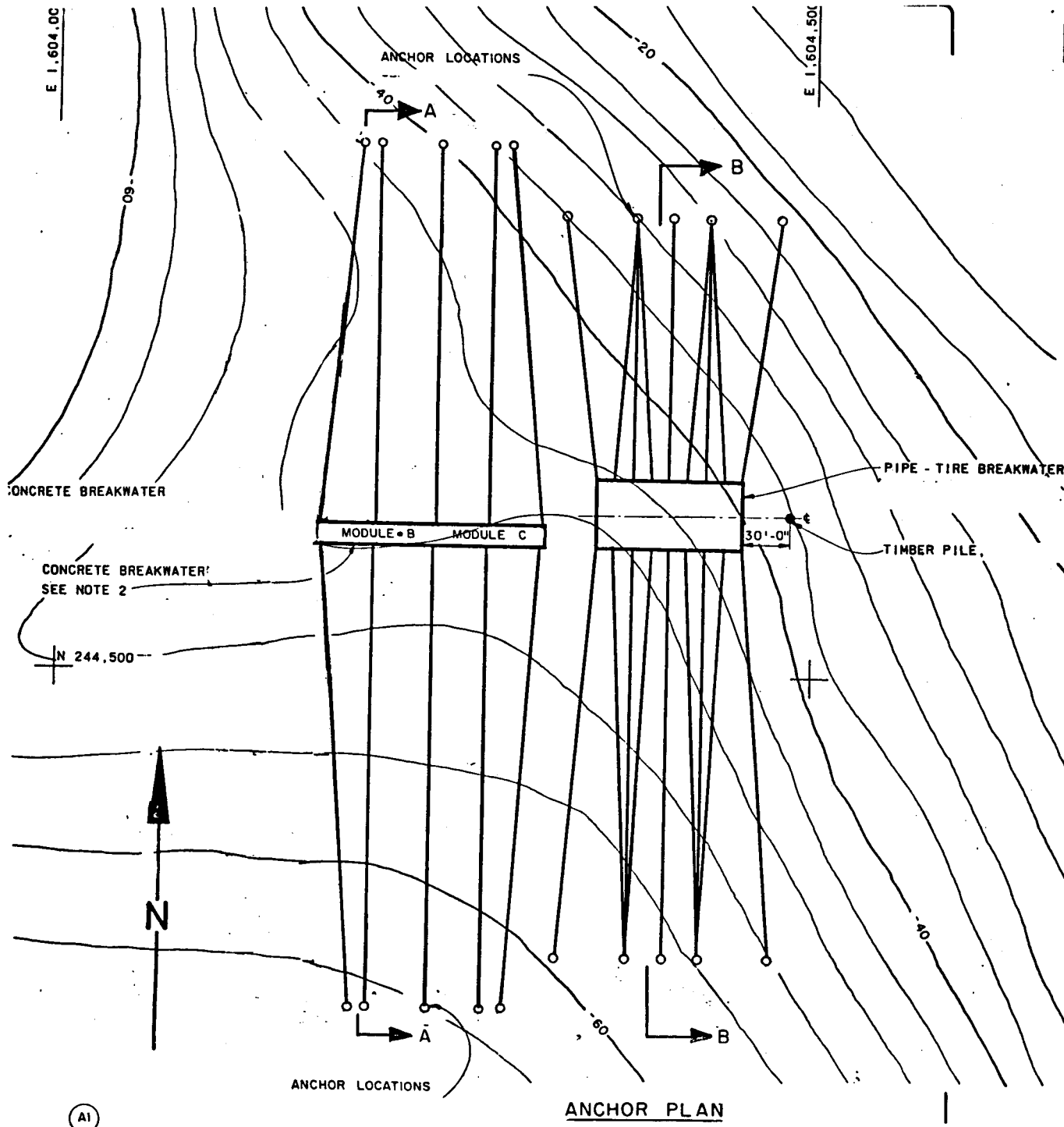


Figure II-7. Anchor Plan

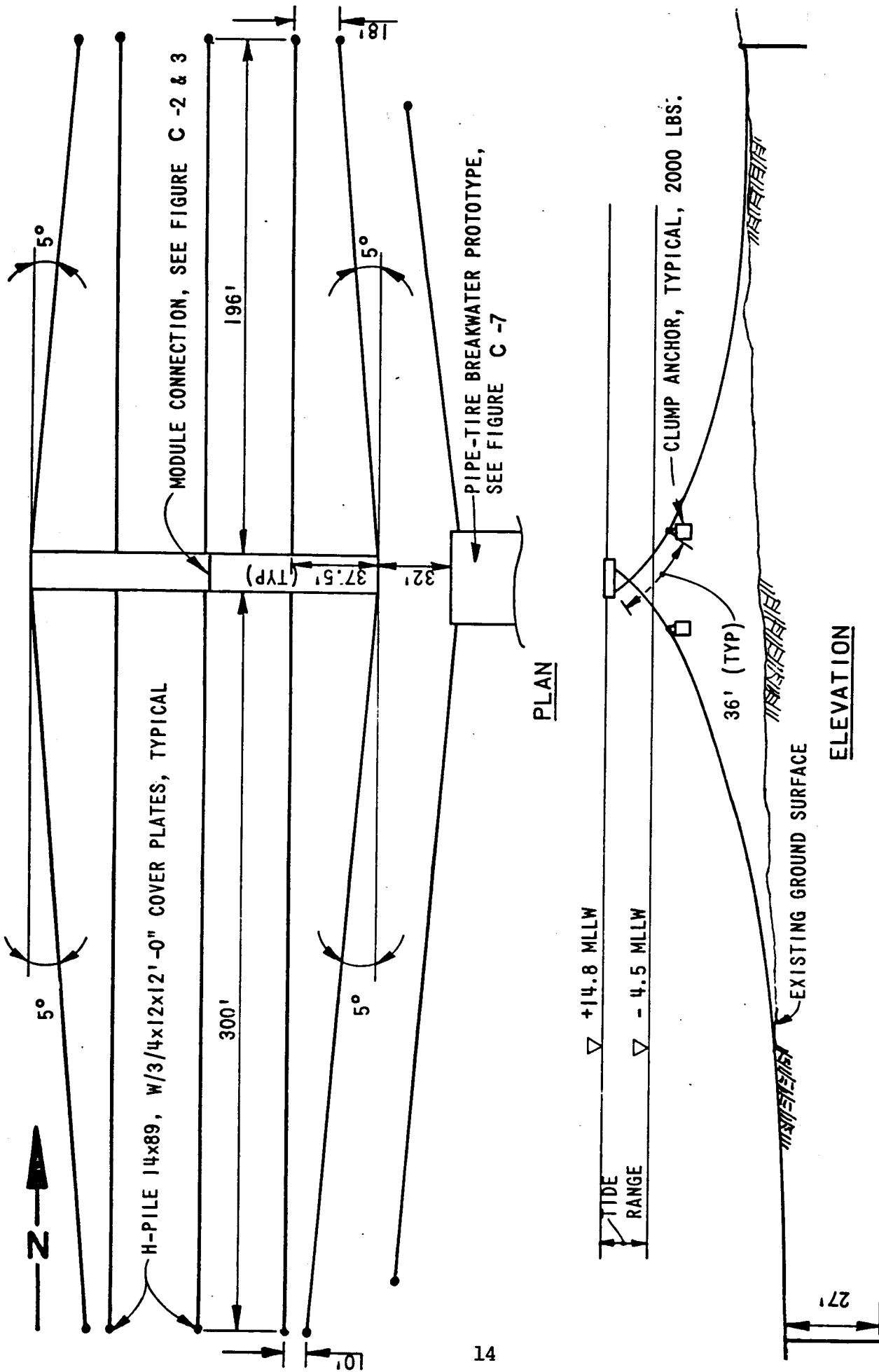


Figure II-9. Anchor Detail - Concrete Breakwater

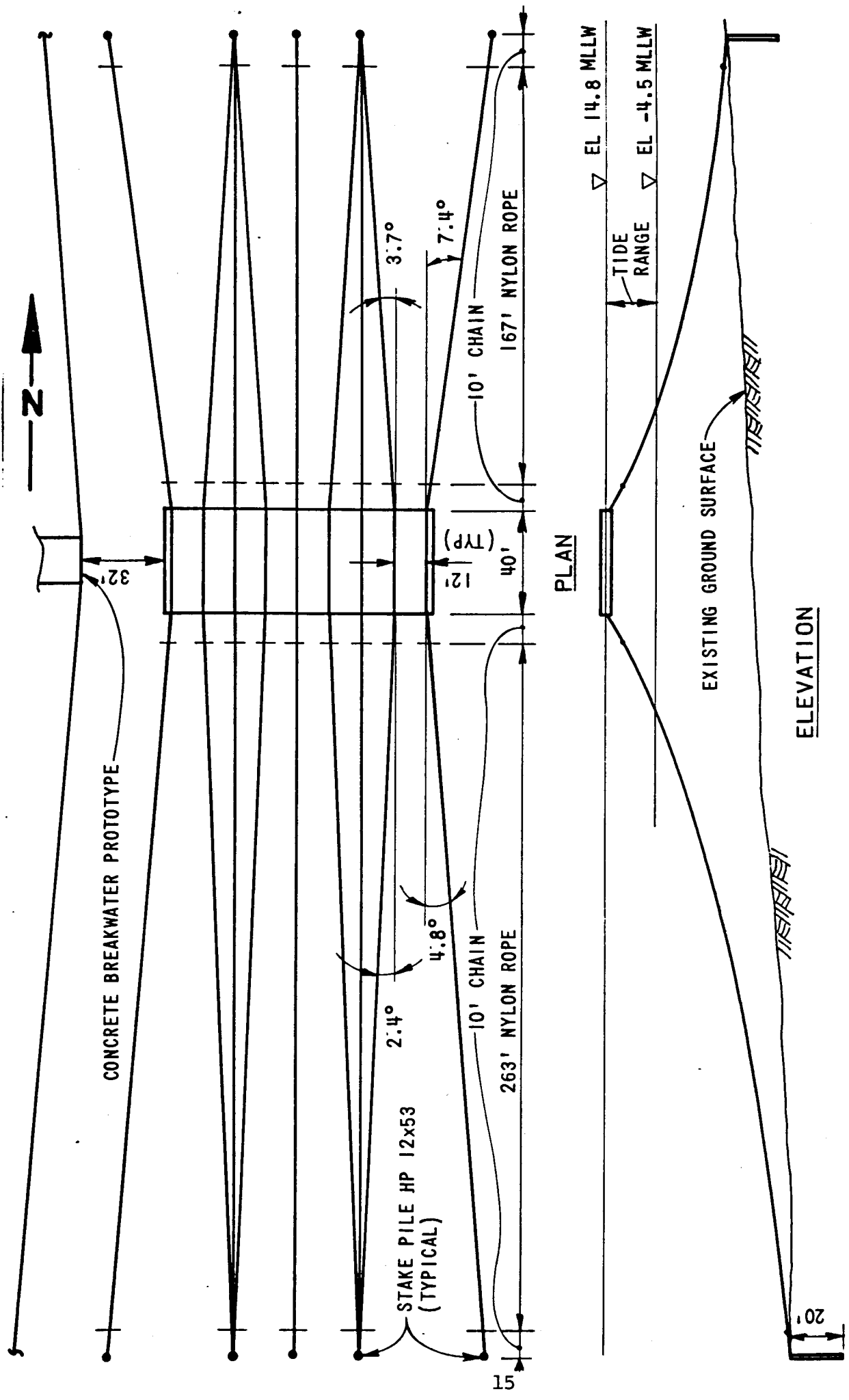


Figure II-10. Anchor Detail - Pipe Tire Breakwater

CHAPTER III

INSTRUMENTATION

III.1 Design Criteria and Transducer Summary

The project monitoring system was installed in the fall of 1982 and removed on 31 January 1984. Details of the individual sensors are described in subsequent paragraphs, however the general purpose of the program was to collect data which will serve as a basis for establishing and evaluating the fundamental behavior of the two breakwater types under study. The system was designed to measure all pertinent environmental and structural variables, that may be involved in the ultimate design and mathematical modeling of these and similar structures. The actual variables to be measured, along with the required individual transducer accuracies and measurement resolutions are summarized in the appendices, and their layout within the floating breakwater test facility is given in the previous section. The monitoring program consisted of four major parts, 1) Equipment installation and maintenance, 2) data collection and storage, 3) remote monitoring, and 4) data reduction and presentation.

The basic criteria used for the development of the monitoring system are as follows:

1. Automatic data recording based on given storm conditions.
2. Complete automation and battery operation.
3. Remote monitoring of transducer outputs and data from a host computer facility and/or remote terminal.
4. On site storage of all data in a retrievable format.
5. Minimum of 1024 samples of each input transducer for each event (an event being defined as hourly sampling of 1024 points of all inputs during any selected storm event).
6. The data reduction consisted of a basic summary of all the pertinent statistical data for each input event. This includes the minimum, maximum and mean values, the standard deviations, and the auto and cross-spectral calculations for all pertinent breakwater response measurements.

The parameters that were measured include: incident and transmitted waves, wind speed and direction, anchor line forces, strains in the concrete units, relative float motions, rotational and linear accelerations, pressure distribution on the concrete

breakwater, water and air temperature, water current speed and direction and connector forces.

Off the shelf transducers for measuring many of the parameters were not available. A major effort was required to design and fabricate underwater anchor force load cells, wave measuring spar buoys, a relative motion sensor, pressure sensors and embedment strain gages. By the end of the monitoring program, approximately 80 transducers had been installed in and around the breakwater. Over 2 miles of underwater electrical cable were required to feed signals to the on-board data acquisition system, which was housed in the hollowed-out interior of the western concrete breakwater unit during the first part of the test program and in a metal house mounted on the deck during the second half of the testing. Using large lead-acid batteries for power, the system was completely self contained. In addition to the input transducers, the system includes a microprocessor controlled data logger and special purpose signal conditioning electronics which were designed and built by the university. The data acquisition system was programmed to sample selected transducers for one minute on an hourly basis. When either wind speed, current speed, anchor force, or significant wave height exceed a present threshold value, a 8.5 minute record of all transducers was made at a sampling rate of 4 hertz. The system was inaugurated on 11 October 1982, but several weeks of debugging were required before reliable data recording actually began.

TABLE

Transducer Layout	Channel Number	Date In	Bits Res.	Transducer Range
Anchor forces-concrete	1-8	10/11/82	12	50 kips
Anchor forces-tire	10-12	10/11/82	12	10 kips
Wave & tide gage	16	10/11/82	12	25 feet
Wave buoys	17-21	*	8	8 feet
Dynamic pressures	22-44	*	8	5 psi
Concrete strains	45-60	10/11/82	8	200 us
Accelerometers	61-66	*	8	1 g
Relative motions	67-72	10/11/83	8	as required
Wind speed	73-74	10/11/82	8	100 mph
Wind direction	75-76	10/11/82	8	0-360 degrees
Current velocity	76-77	*	8	0-5 fps
Unused	79-82			
Voltages	83-88	10/11/82	8	as required
Temperatures	89-92	8/83	8	0-100 of
Wave array	9-12	11/83	12	8 feet
Wave array	*	11/83	8	8 feet
Rubber connector forces	*	*	8	*
Rigid connector forces	*	*	8	*

* Varied throughout project.

III.2 Load Cell

Forces in the anchor lines were measured close to the breakwaters and close to the anchor piles, as shown in Figure III-3. Since the incident waves can approach the test site from both the north and the south, gages needed to be placed in the anchor lines on opposing sides of the breakwater.

The design for the anchor force gage is shown in Figure III-4. It is a University of Washington design, using an O-ring-sealed strain gage load cell, similar in construction to the standard laboratory load cell. The gages were calibrated using standard laboratory test equipment. Four 10-kip load cells with an ultimate strength of 25 kips were placed on the pipe-tire breakwater and eight 50-kip cells with an ultimate strength of 125 kips were placed on the concrete breakwater. Waterproof connectors were used on each anchor force gage. Connectors were capable of being plugged and unplugged underwater.

Each load cell signal conditioning circuit consisted of a load cell bridge power supply and balancing circuit, a high gain precision instrumentation amplifier, and a low pass analog filter. This circuitry is shown in Appendix I.

The strain sensing element of the load cell is a strain gage bridge circuit having four active legs with two strain gages per leg. Mounting of the strain gages in the load cell is illustrated in Figure III-5 as well as the relative position of the gages in the bridge circuit. The stress concentration at the edge of the hole is assumed to be a factor of 3 over the average stress in the corresponding direction. This is only a rough approximation which is nearly true if the hole diameter is very small compared with the tube diameter. However, this is not quite the case here and the approximation serves only to give order of magnitude values for the output voltage from the bridge circuit for design purposes. Because of this unknown factor, it was necessary to calibrate the load cells, (i.e.), measure the output under known applied loads to obtain the actual relationship between load force and output voltage. (See Appendix VI).

The relationship between average strain and output voltage as derived in Figure III-5 and is given as follows:

$$E_o = E_{in} (G.F.) (2) L$$

L = Average longitudinal strain

$G.F.$ = Gage factor or strain gages

E_{in} = Bridge excitation voltage

A bridge excitation voltage of 5 volts was chosen based on the recommended power density level in the strain gages to avoid

$$E_o = \frac{(5)(2.055)(2)(.0005)}{4} = \frac{10.25}{6.7} \text{ millivolts.}$$

The required input to the analog to digital converter for maximum output is 5 volts. Therefore, the required gain of the amplifier is

$$\text{Amplifier Gain} = 5/.0067 = 750$$

Because of the uncertainties noted above and to allow for a reduced output voltage range, the gain of the amplifier was made adjustable from approximately 100 to 1000.

We come now to the question of accuracy of the measurement. While the resolution is very high, the accuracy is not of the same magnitude. There are a number of sources of error in the measurement. The first and most important is the calibration error. Appendix VI shows the calibration curve for the load cells tested. An MTS testing machine was used for these calibrations. The overall accuracy is about 3%.

The principal sources of error in the load cell bridge circuit, the instrumentation amplifier and the A to D converter are the zero-drift, gain stability/linearity, and noise voltages. Since this is to be primarily a dynamic measurement, (although at very low frequencies), zero drift is not as important as the other two sources of error. However, every effort has been made to keep zero drift as low as possible.

If the external pickup noise in the load cell itself and in the leads to the load cell can be reduced to a negligible value, (particularly at the frequencies of interest), the noise voltage at the input to the instrumentation amplifier, (specified by the manufacturer), is 0.8 microvolts peak-to-peak in the frequency range 0.01 to 10 Hz. The one bit recording resolution corresponds to an output voltage change from the load cell of about 1 microvolt and, thus, the noise is of the same order of magnitude as the recording resolution. Because of the high gain of this amplifier, the self-noise of the other components is not significant.

Possible sources of drift are creepage of strain gage backing material and bonding agents, slight differences in temperature coefficient of strain gages and change in gage factor. Since a four-arm active bridge is being used in a very uniform temperature environment, the temperature drift should be very minimal. Zero drift of the instrumentation amplifier is a function of the ambient temperature change and power supply variations. The specified maximum zero drift of the amplifier is + 4 microvolts per degree centigrade and + 3 microvolts for a one per cent change in power supply voltage. The power supply has a temperature stability of +0.05% over a 5°C temperature change and it also has a long term stability of +0.1%. Thus, the amplifier drift due to power supply variation is negligible compared to the drift with temperature. If the temperature of the electronic

recording package is controlled to, say 5°C, the temperature drift at the input to the amplifier, (not counting load cell drift), would be + 20 microvolts. This is equivalent to a +24.4 lb force on the load cell. This, however, would only be a factor when looking at the mean or stationary value over long time periods and would have little effect over the data sample period for a single record. The zero drift in the analog to digital converter is small compared to the drift from the amplifier and can be neglected.

Gain stability and nonlinearity are the most important factors in determining the accuracy of the dynamic measurement. The gain of the bridge circuit will be affected by changes in the gage factor of the strain gages as a function of temperature variation and by power supply changes. The power supply stability as previously stated is +0.15%, (long term stability plus 5°C temperature variation), and the gage factor stability for the bridge circuit is the sum of the two or +.2%. Gain stability of the instrumentation amplifier and the analog to digital converter under the same conditions of power supply and temperature variations are +.01% and +0.02%, respectively. The overall gain stability is then the sum of the stabilities of each component which is +0.23%.

The measured nonlinearity and hysteresis of the load cell is approximately +0.3% of full scale. The maximum nonlinearity of the instrumentation amplifier and the analog to digital converter are specified by the manufacturers as +0.01% and +0.05% of full scale, respectively.

The overall accuracy of the system would be the sum of errors due to gain instability, nonlinearity and calibration errors. Gain instability produces an error based on the magnitude of the reading while nonlinearity produces an error based on full scale. The overall accuracy then is +0.23% of the reading and +0.36% of full scale.

III.3 Wave Monitoring

Incident and transmitted waves were measured for wind waves and during four boat wake tests. Resistance wire wave staffs were used. Detailed descriptions of the design and operation of the wave staff follows.

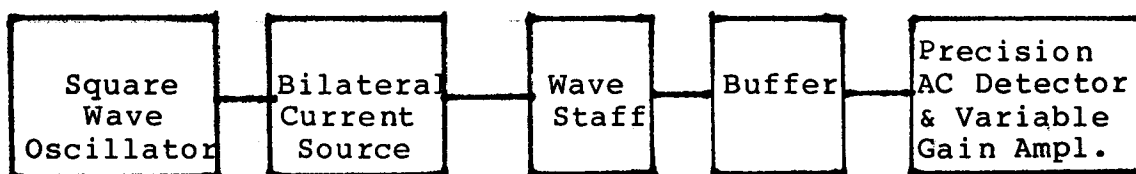
Five wave gages were arranged as shown in Figure III-1. Gages 2 through 5 are spar buoys as shown in Figures III-6, and gage 6 was mounted to a stationary piling and served as a tide gage as well. The spar buoys consist of a 15 feet long, 6 inch diameter polyvinyl chloride, (PVC) pipe with a 2 1/2 foot damper plate, on which is mounted a 12-foot section of 3 1/2 inch PVC pipe with a spirally wound wire to form a resistance gage. The necessary electronics was installed in the top of the upper section. Gage 6 was a 3/4 inch-diameter PVC pipe 25 feet long supported by being attached to a steel cage bolted to the piling. This gage served as both a wave gage and a tide gage, and

therefore, extended from -5' mean lower low water (MLLW) to +20' MLLW. See Figures III-6 - III-8 for more details.

Between mid-October, 1983 and the removal of the breakwater in late January, 1984, there was an eight-gage linear wave buoy array anchored where the rubber tire breakwater was originally located. These gages were at a five-foot-spacing and were 66.7 feet due west of the tide gage. The longitudinal alignment of the array was parallel with the breakwater. See Figure III-8.

During this same period, a Wave Ryder acceleration wave buoy was anchored approximately 150 feet south of the wave array. The data from this buoy were sent directly to the Corps' Coastal Engineering Research Center for analysis.

Wave Staff Design. A block diagram of the wave staff and associated electronic circuits is shown below:



The wave staff itself consists of a length of PVC tubing which is spirally wound with a resistance wire, such that when it is immersed in sea water, the electrical resistance varies in direct proportion to the length of the exposed staff.

The electronic circuits driving the wave staff consists of a fixed frequency square wave oscillator, (having a precisely controlled output amplitude), driving a precision bilateral current source with an output current directly proportional to the input voltage. Thus, the wave staff is driven by a current source of constant magnitude, but one which changes direction with each half cycle of the square wave oscillator. The output of the wave staff then is a square wave voltage with a magnitude, (peak-to-peak), that is directly proportional to the length of exposed wave staff. This output is fed to a high input impedance voltage follower circuit which serves as a buffer between the wave staff and the AC detector circuit. The precision AC detector circuit uses two operational amplifiers in conjunction with two diodes to form a precision full wave rectifier circuit that is capable of operating at very low input voltages. Ordinary diode detector circuits cannot operate on AC signals of peak magnitude less than the forward voltage drop of the diodes and produce large conversion errors unless the signal magnitude is large with respect to the diode voltage drop. A gain control has been incorporated in the detector circuit so that full scale output can be set at any positive value up to +10 volts with a wave staff resistance of 300 ohms up to 3000 ohms.

Alternating current is used to drive the wave staff to avoid

both the corrosion effects that would occur if direct current were used and the DC offset which occurs as a result of the use of dissimilar metals in a conducting solution. The latter is eliminated by use of AC coupling in the output from the wave staff.

Bench tests of the wave staff electronic circuits were made using a 1000 ohm variable precision resistor in place of the wave staff. The circuit was adjusted to produce an output range of 0 to 10 volts with the resistor varied from 0 to 1000 ohms. Linearity was determined to be 0.1% of full scale over this range.

Tests were also made to determine the effect of temperature on sensitivity and zero drift. A decrease in sensitivity was noted with decreasing temperature of about 0.03% of reading per °C over the temperature range of 0 to 24°C. A zero drift of 2 millivolts was also noted over the same temperature range. A +10% change in supply voltage from the nominal +15 volts produced no observable change in output. If we assume an operating temperature range of +5°C, the maximum error in the wave staff electronics due to the combined effects of nonlinearity and sensitivity variations with temperature is +0.2% of reading. Since the primary interest is in a dynamic measurement of waves, the zero drift noted will have negligible effect on the experiment since temperature variations of any appreciable magnitude will only occur over long periods of time compared to the wave periods.

Further calibration tests were conducted using actual wave staffs of 1-inch diameter and 20-ft. lengths, and 3 1/2-inch diameter and 8-ft. lengths at various depths of immersion in salt water. These tests were conducted from a dock at Shilshole Bay Marina on Puget Sound. Because of ripples and waves on the water of the order of one inch (peak-to-valley), it was difficult to obtain a highly precise measurement. The output was recorded on a strip chart recorder and it was therefore possible to average these variations to some degree. The readout resolution of the strip chart, (and accuracy), is about +1/4 of a minor division. Full scale across the chart is 50 minor divisions and, thus, the resolution is about 0.5% of full scale. Some nonlinearity is noted near full immersion, (see calibration curve). Some offset was expected because of the finite resistance of the salt water path in the ground return which is not taken into account during initial calibration of the wave staff unit. The initial calibration is made with the wave staff on the dock where full scale and zero are set by making actual contact between the ground wire and the wave staff resistance element at the corresponding ends. However, measurements were made of the resistance of the salt water path to ground in the same location where the wave staffs were immersed and the value of resistance measured, (on the order of 10 ohms), does not account for the offset observed at full immersion. In addition, the offset should occur at all readings and, it does not. Therefore, it is believed that the nonlinearity observed is a result of some other phenomenon as yet undeter-

mined. Both units produced highest accuracy near center scale with decreasing accuracy toward either end. Overall accuracy including end points is about +3%. If the range of operation is reduced so as not to use the last one foot on each end of the wave staff, the accuracy is improved to about +1%.

Spar Buoy Design. Spar buoys were used at four locations because of their advantage in handling and transport and because they minimized the placement difficulties due to navigational hazards, water depth, and tidal conditions. The spar buoys were made of two PVC pipes coupled together near the center of the buoy. The lower section is a 15' x 6" pipe filled with styrofoam. The top section is 12' x 3" wherein the upper 8 feet is wound with a resistance wire which measures wave height. The wave staff electronics are mounted inside the top section, above the water line with the remainder being filled with a foam core to add stiffness. The buoys also have a 2.5 foot diameter damping plate mounted on the bottom and are anchored using dual point mooring system with the anchor lines attached at the center of drag on the buoy to prevent it from being pulled underwater in strong currents.

In 1975, one of these buoys was tested in the Puget Sound just north of Seattle. Its performance exceeded expectations both in terms of minimized response to the waves and accuracy of wave height measurement. Figure III-11 gives a sample of the output from the buoy's wave staff in salt water for a plus and minus one foot excitation of the buoy in heave. This was accomplished by pushing the buoy up and down by hand. Some distortion resulted from this approach which shows up in the output of the accelerometer mounted at the center of buoyancy of the buoy. Figures III-13 and III-14 give samples of the output of the response of the buoy in heave and roll in calm water. The natural periods for heave and roll taken from these plots are approximately 18 and 14 seconds respectively which are well out of the range of maximum wave periods expected at the breakwater site, which is between 3 and 5 seconds. Visual observations of the buoy in waves in excess of 1 1/2 feet indicated no heave or roll motion, but some yaw about the anchor line caused by the current and wind. This motion resulted in less than a one foot variation from the buoy's horizontal position in calm water and appeared to have periods in excess of 30 to 60 seconds. For comparative measurements, the buoy was located about 30 feet from an existing one inch diameter resistance wire wave staff. A comparison of simultaneous output from the two wave staffs, (buoy mounted and stationary), is shown in Figure III-12. The autospectras computed from data obtained from one of the stationary wave staffs and from the spar buoy, in a 25 mph storm with maximum wave heights in excess of 1 1/2 feet, are shown in Figure III-15.

These spectras were computed from simultaneous records of 17 minutes in length.

III.4 Pressure Sensors

Fifteen pressure sensors were mounted on the sides of the concrete pontoons and 7 on the bottom of the west pontoon only. (See Figure III-17.). All of the side mounted units were damaged early in the project and were not replaced until late summer, 1983. The mounting details for the side mounted gages is given in Figure III-16.

The pressure sensors used were a Kulite Model IPT-750, 0-5 psi range. They are a semi conductor, strain gage devices with a flush stainless steel diaphragm. The basic specifications are:

1. 0-5 psi range
2. 0.85% overall accuracy
3. Infinite resolution
4. 75 mv output full scale
5. -40°F to 250°F temperature range

III.5 Motions

a) Accelerometers

Linear accelerometers measuring normal, (to the deck, or heave), and transverse, (or sway), acceleration and an angular accelerometer measuring rotation about the longitudinal axis, (or roll), were employed on each float. Although some change of equipment and repair took place during the life of the project, all accelerometers employed were of the highly accurate servo type. The design incorporates a feedback mechanism whereby motion of the displacement pickoff produces a countering force, (or moment), which accelerates the seismic mass so that it undergoes only a minute displacement from the applied input motion. This feature combined with a flexural suspension system provided for accurate acceleration measurement with minimal nonlinearities and negligible hysteresis. Internal filtering mechanisms provided very clean data, so no additional filtering or detrending is required.

b) Relative Motions

The relative displacement between the two concrete pontoons, while connected together using the later rubber connector design, was monitored using a specially designed articulated mechanical transducer. The unit was constructed of anodized aluminum and a U-joint type fixture at each end and a rotating extendable tubular section between them.

The device was able to measure the rotation about the perpendicular axes, which are in the plane of the end sections on the adjacent pontoons and the relative distance and rotation between the two pontoons. See Figure III-19 for more details. From this configuration all six degrees of freedom associated with the relative motion of the two pontoons can be computed.

The five rotational motions were sensed using Vernitech Model 106 Sine-Cosine potentiometers with an overall accuracy of

0.3%. The single linear measurement used a Model 111 linear potentiometer with a full range of 8 inches and an overall accuracy of 0.015%.

III.6 Wind Speed and Direction

A cup-type anemometer was mounted on a pole atop the piling where wave gage No. 5 was attached. The anemometer was set at a height of approximately 30 feet above MLLW and will be used to measure speed and direction. Duration will be deduced from the anemometer records. The anemometer was operated continuously and was used to turn on the complete monitoring system when the wind speed reached a preselected speed-duration level. At the start of the program, monitoring was initiated when the average wind speed exceeded 15 miles per hour, (m.p.h.) for a one minute period, or if selected anchor force gages measured loads exceeding 2000 pounds above the initial tension.

III.7 Concrete Strain

Measurements of internal concrete strains were made at 12 locations on the west module, using rebar strain gages, (see Figure III-20 and III-21). The gages were built at the university and were constructed using a 3 foot piece of No. 5 rebar which was machined to 1/2" over a 6" section at the center. Four strain gages were attached and wired into a complete bridge circuit. The gages were then sealed using standard strain gage sealants and a self adhesive heat shrink tube was placed over the finished unit for both mechanical protection and as a final sealant. Two sets of gages and lead wires were attached to each unit. Each gage was checked, (calibrated) using an MTS testing machine. They were attached to the rebar in the breakwater using standard wire ties. The electrical leads were also attached to the rebar with wire ties and wire running along the rebar to the instrument compartment.

III.8 Temperature

The temperature sensors used were manufactured by Analog Devices (Model AC2626). They are a laser-calibrated two terminal IC transducer. The electrical output is a current (1 A per K) linearly proportional to absolute temperature, thus eliminating the need for costly linearization and cold junction compensation. Due to the units high impedance current output, it is insensitive to voltage drops over long lines thus enabling remote monitoring with no need for costly transmitters or special wire. The unit is also insensitive to supply voltage changes above 3 volts thus allowing for battery operation. The units used were potted in a stainless steel tube 3/16" by 3" and were placed directly in the sea water. The overall accuracy of each was + 0.8°C.

III.9 Current Speed and Direction

An off the shelf, Series 500 Marsh McBirney electromagnetic

water current meter was used in an attempt to measure the x and y components of water velocity. The Model 512 unit with a 1 1/2-inch probe was used. The current probe was mounted under the center of the pontoon. For a number of reasons, satisfactory current measurements were never obtained.

III.10 Pontoon Interconnection Forces

a) Rubber Connector

Strainert standard internally gaged hex head steel bolts were used to measure the axial forces on the flexible connectors. Two different connectors were used in the breakwater experiment. In both cases four bolts were used to monitor the axial forces between the rubber sections (see Figures III-22 and III-24).

The bolts were instrumented by drilling a $\emptyset.15$ inch hole in the end of the bolt approximately 3 to 4 bolt diameters deep. A complete strain gage bridge is then mounted at the bottom of the hole using an inflatable teflon tube and special adhesives techniques. See Appendix VI for calibration information.

b) Rigid Connection

Standard Dywi-dag bolts were sent to Strainert and gages were placed 17 inches from one end. Four bolts were instrumented and they were placed in the four outer corner positions in the rigid connection. See Figure III-23.

LEGEND

- ② WAVE BUOY
- ⊕ PILE-MOUNTED WAVE GAGE
- ANCHOR FORCE GAGES
- ⊕ ACCELERATION XDCRS
- △ CONNECTOR BOLT FORCE
- ⌈⌋ EQUIPMENT COMPARTMENT
- ⊞ RELATIVE MOTION XDCR

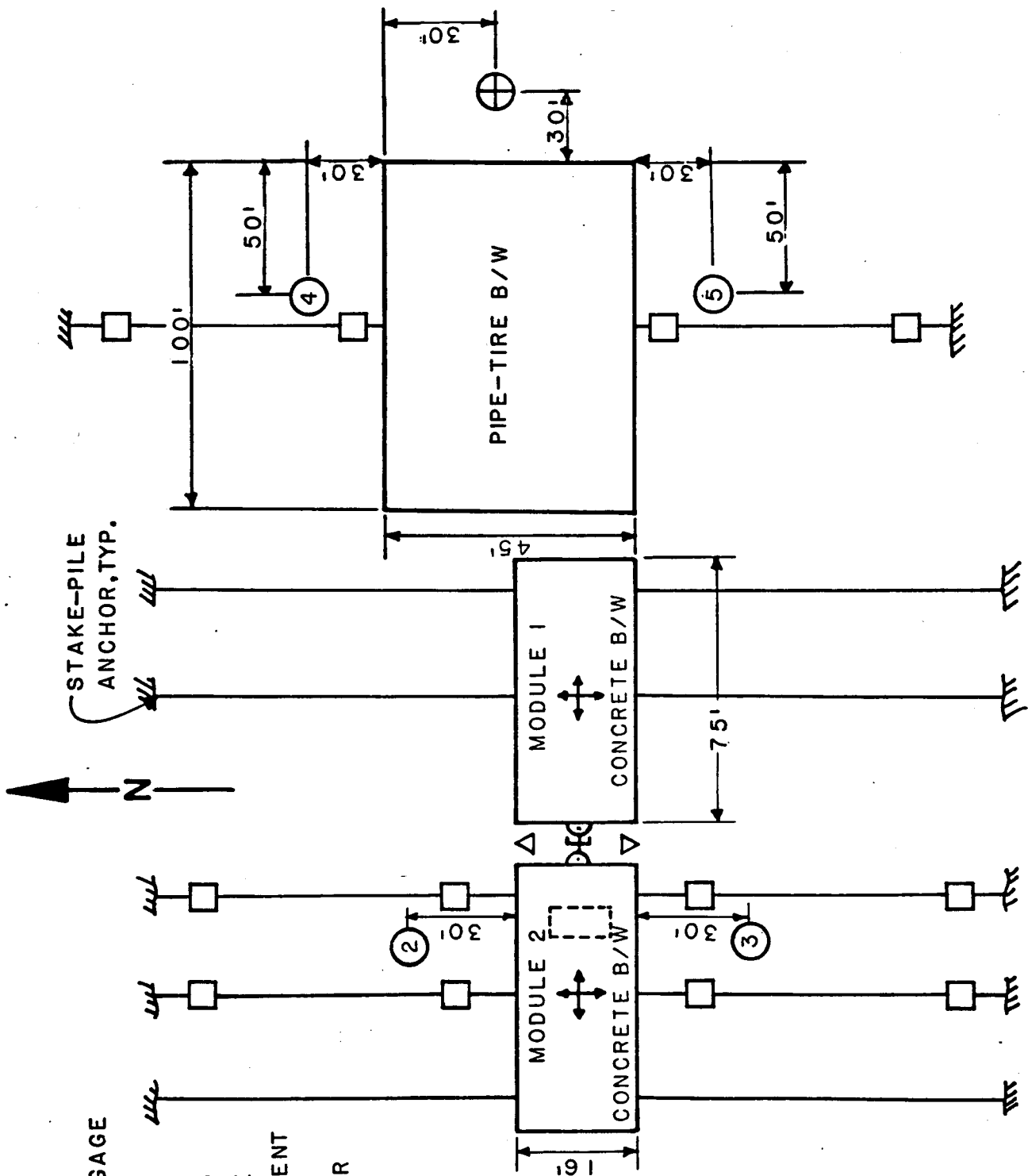


Figure III-1. Project Layout

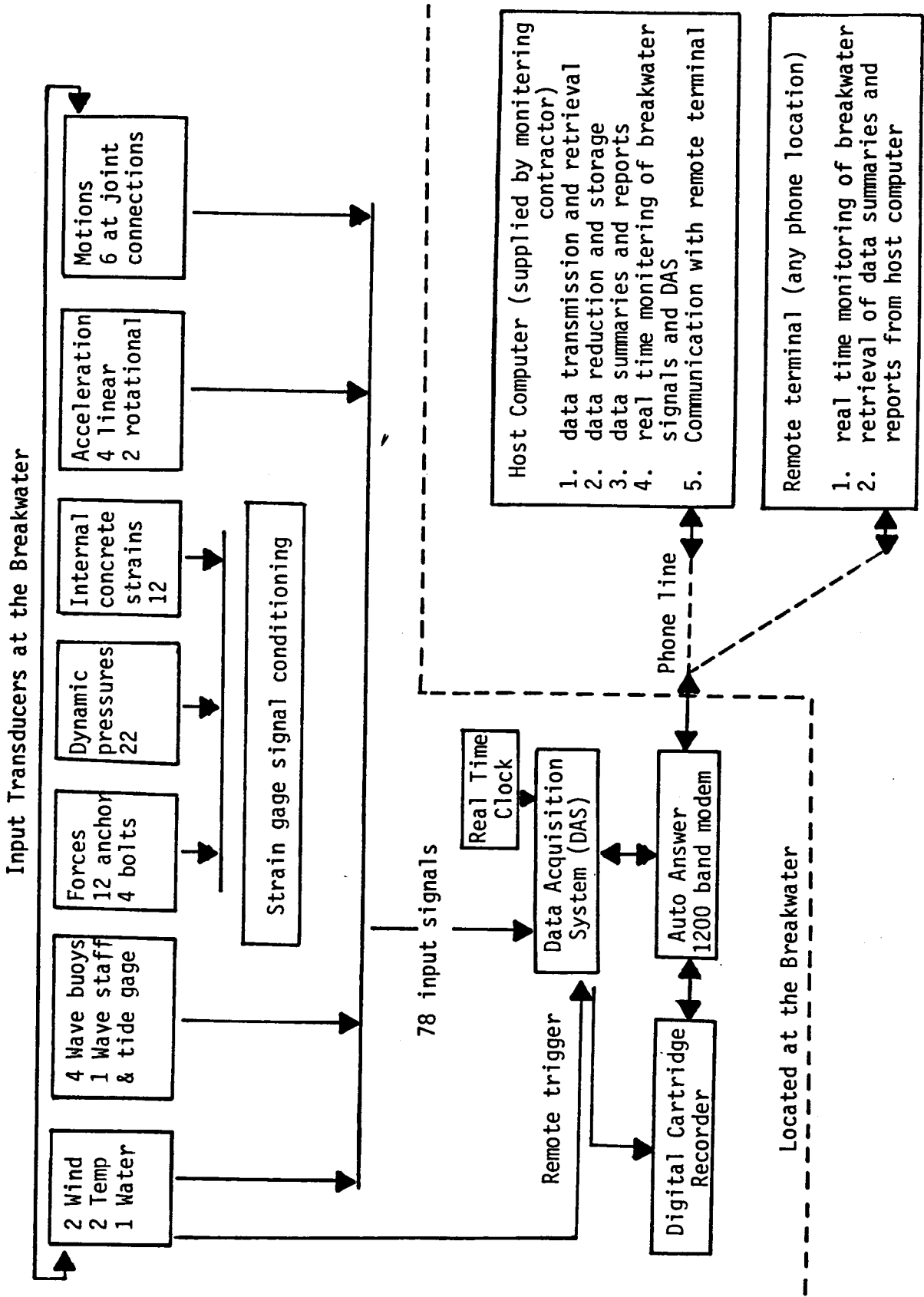


Figure III-2 Monitoring System Diagram

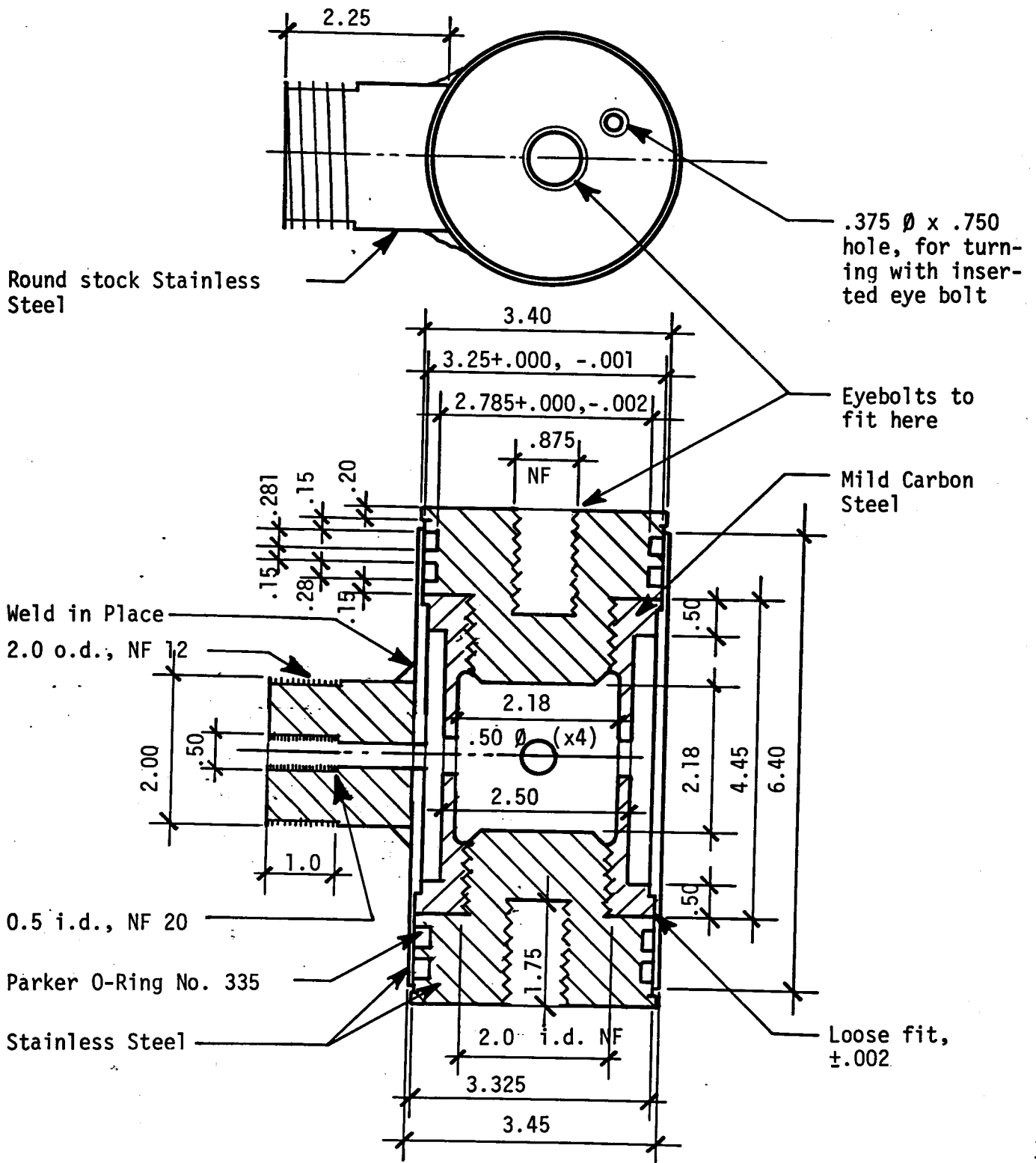


Figure III-4a. 25Kip Load Cell Schematic

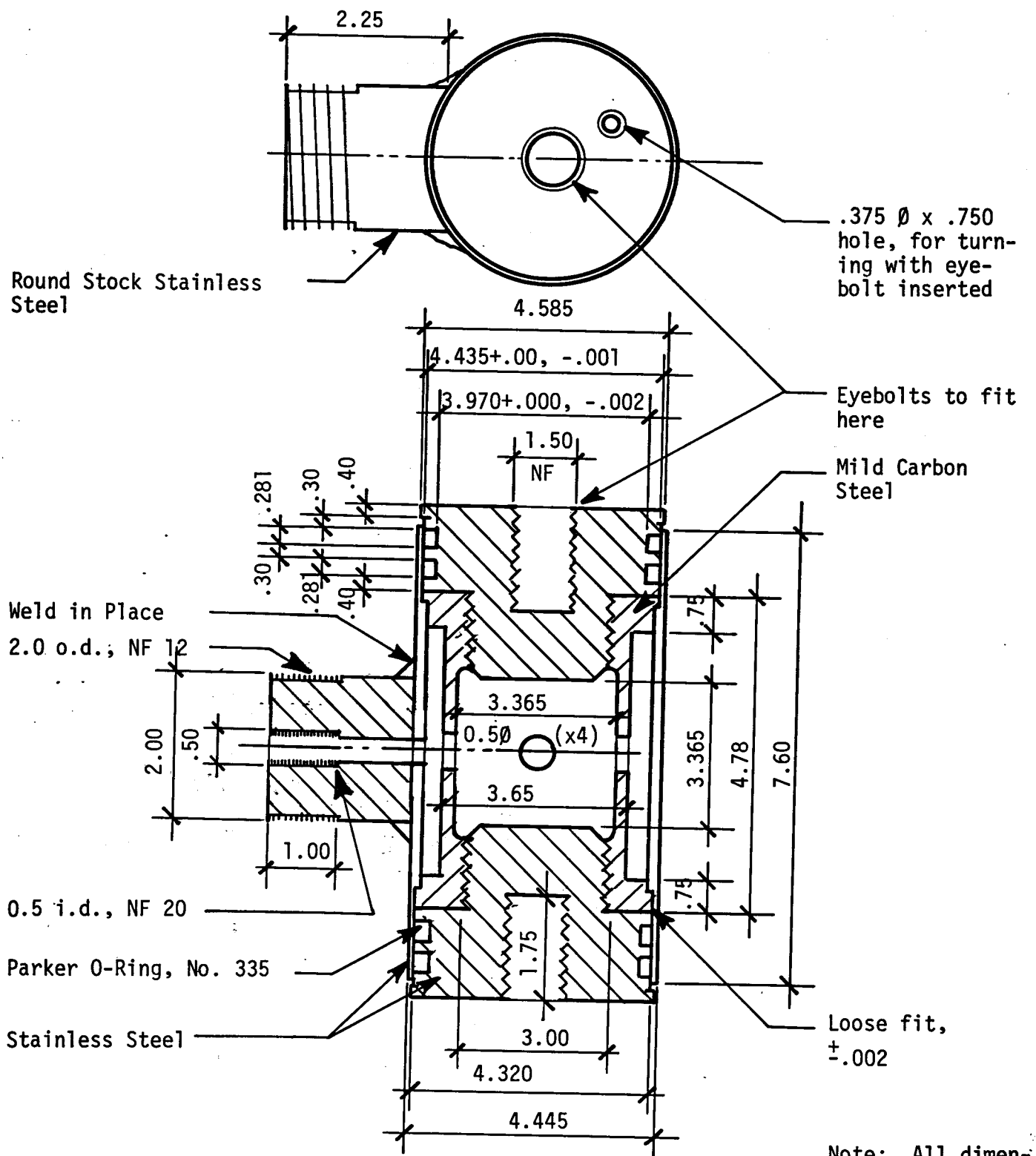


Figure III-4b. 50 Kip Load Cell Schematic

Crouse-Hinds Series B51
underwater connector.
P/N B51F4M(or F)-1

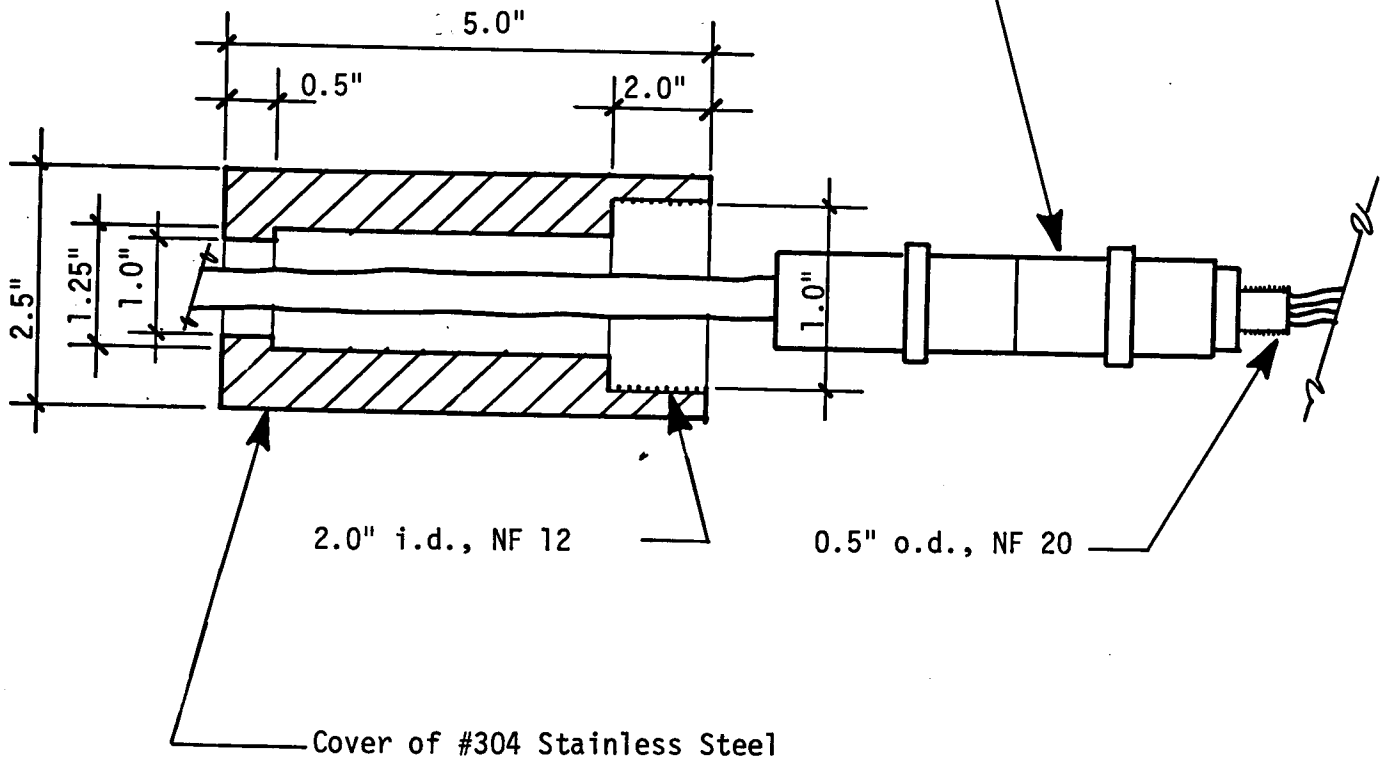
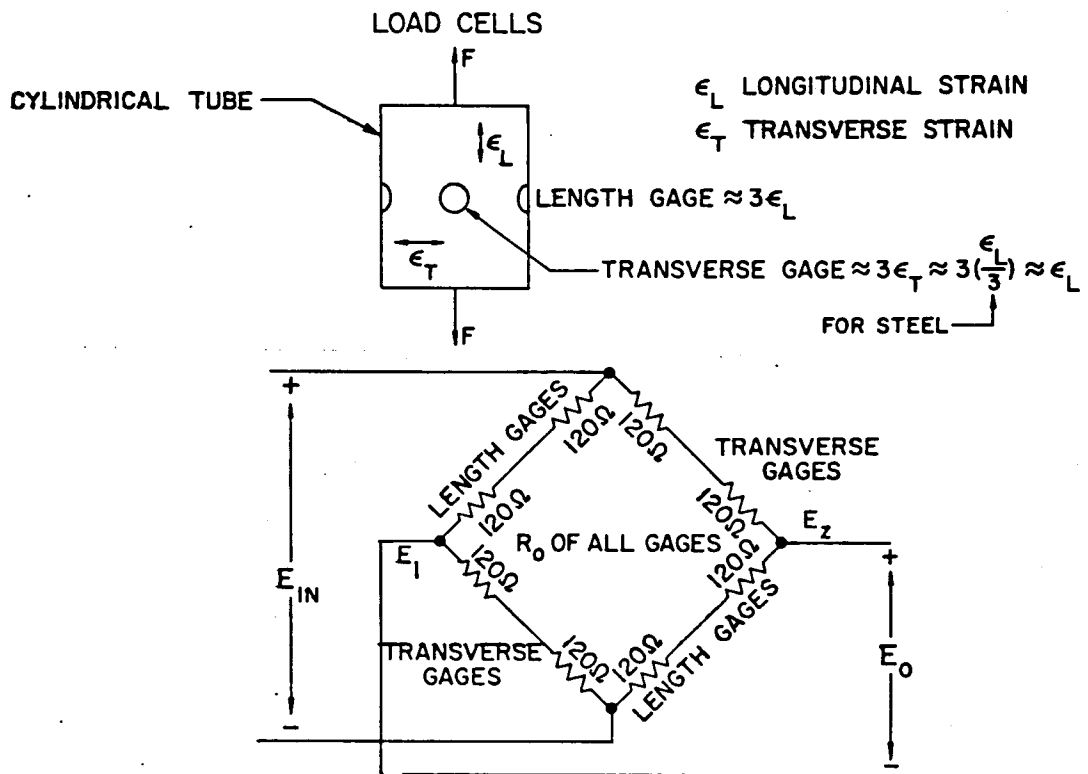


Figure III-4c. Load Cell Electrical Connection



$$E_o = E_2 - E_1 = E_{in} \left[\frac{2R_o + 2\Delta R_L}{4R_o + 2\Delta R_L - 2\Delta R_T} - \frac{2R_o - 2\Delta R_T}{4R_o + 2\Delta R_L - 2\Delta R_T} \right]$$

$$= E_{in} \left[\frac{\Delta R_L + \Delta R_T}{2R_o + \Delta R_L - \Delta R_T} \right]$$

$$E_o \approx E_{in} \left[\frac{\Delta R_L + \Delta R_T}{2R_o} \right] \quad \text{since } 2R_o \gg \Delta R_L - \Delta R_T$$

where ΔR_T = Resistance change of transverse gage due to strain ϵ_T

ΔR_L = Resistance change of length gage due to strain ϵ_L .

In a strain gage, the resistance change (ΔR) is related to the strain (ϵ) by the gage factor (G.F.) as follows:

$$\Delta R = R_o(\text{G.F.})\epsilon$$

Therefore $R_T = R_o(\text{G.F.})\epsilon_L$ and $\Delta R_L = 3R_o(\text{G.F.})\epsilon_L$

$$\text{and } E_o \approx E_{in} \left[\frac{R_o(\text{G.F.})\epsilon_L + 3R_o(\text{G.F.})\epsilon_L}{2R_o} \right] = 2E_{in}(\text{G.F.})\epsilon_L$$

Figure III-5. Strain Gage Layout and Voltage Output Calculation

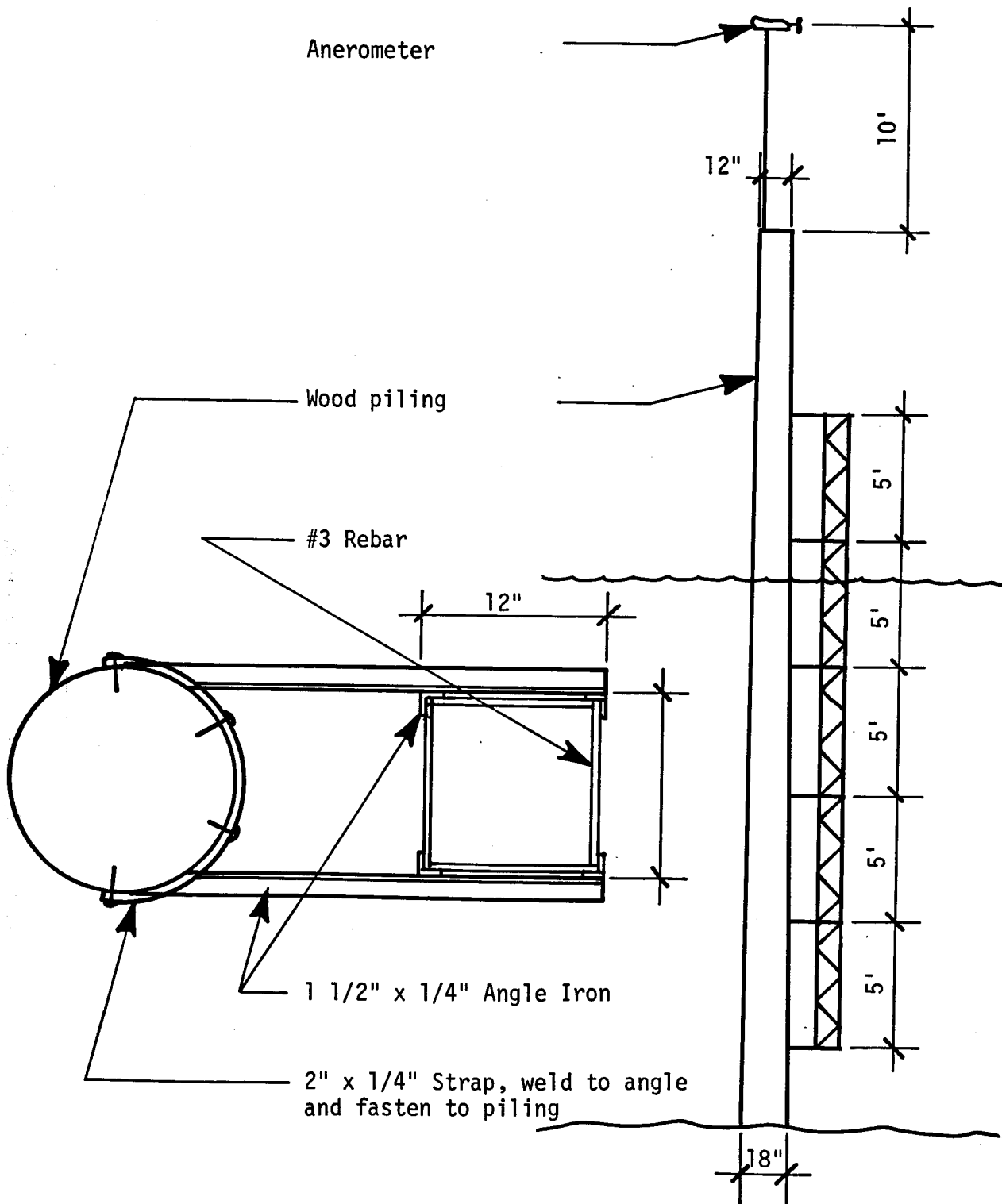


Figure III-6b. Tide Gage Detail

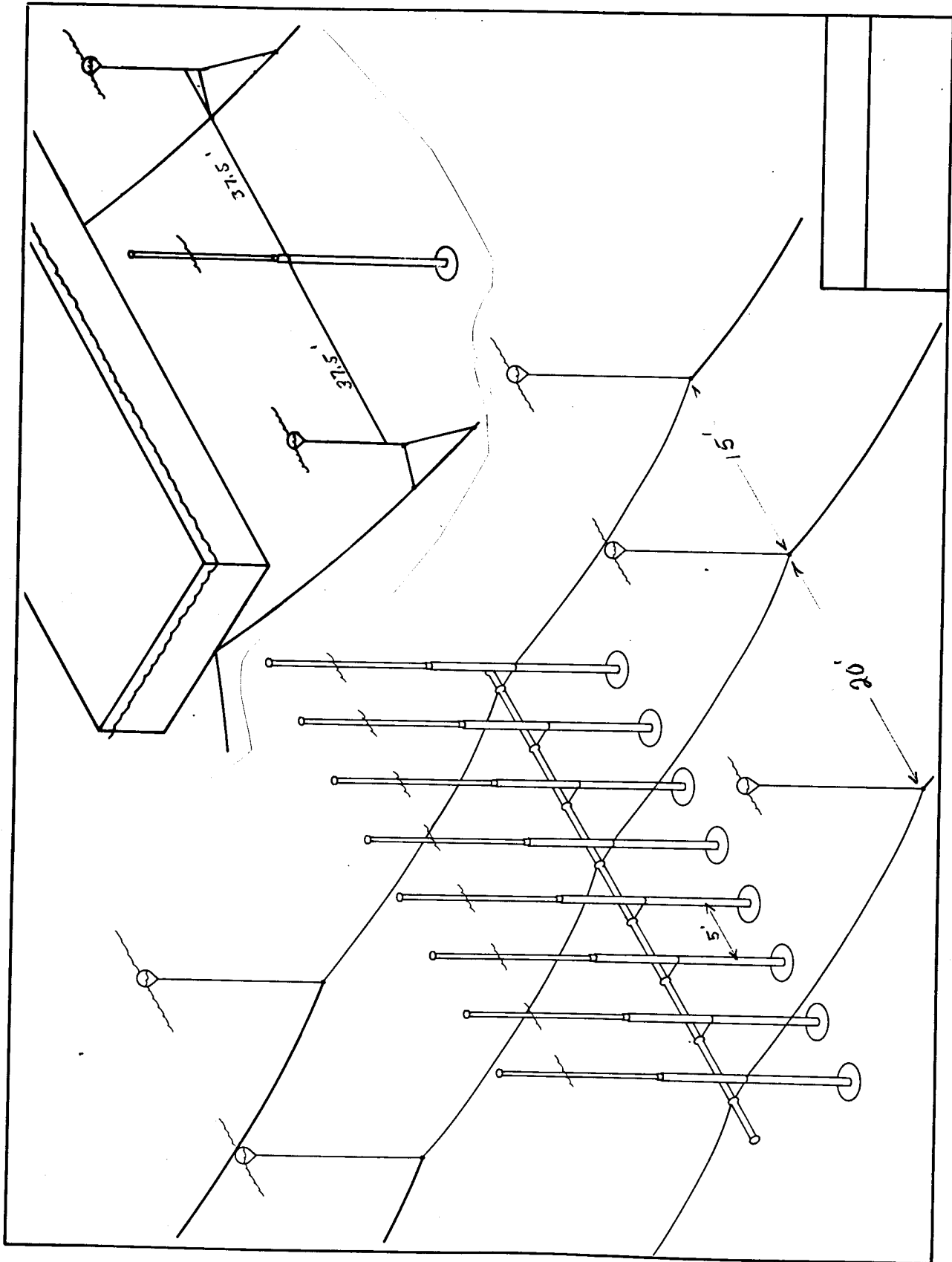
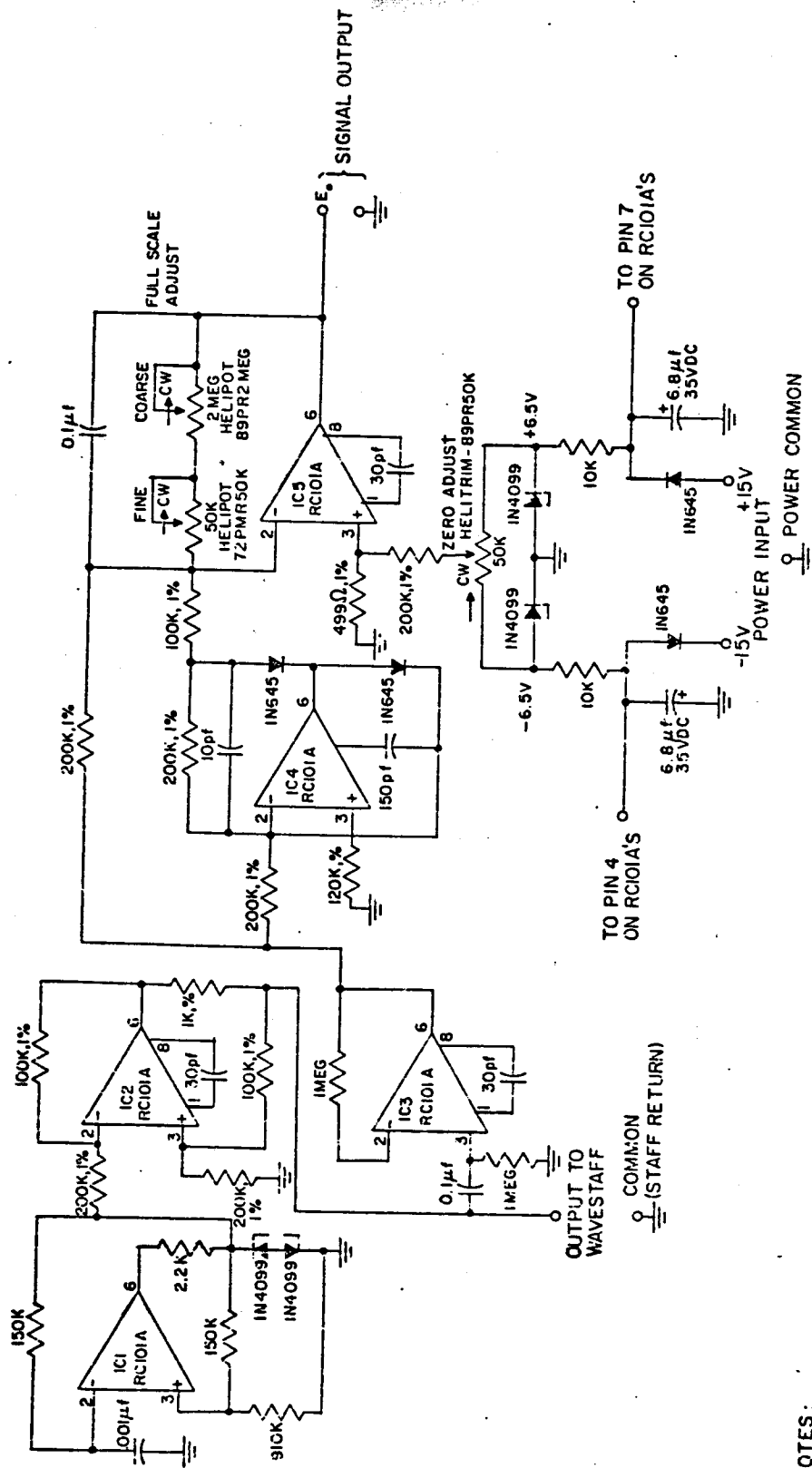


Figure III-8. Wave Buoy Array Anchoring Detail



NOTES:

1. UNLESS OTHERWISE SPECIFIED, ALL RESISTORS ARE 1/4 WATT, 10 %.
2. SELECT DIODES FOR BALANCED OUTPUT (±) AT 2MA DIODE CURRENT.
3. CIRCUIT MUST BE MOUNTED WITH RESISTANCE WIRE WAVESTAFF AND GROUNDED TO SALT WATER AT THE WAVESTAFF.

Figure III-9. Wave Staff Electronic Circuit

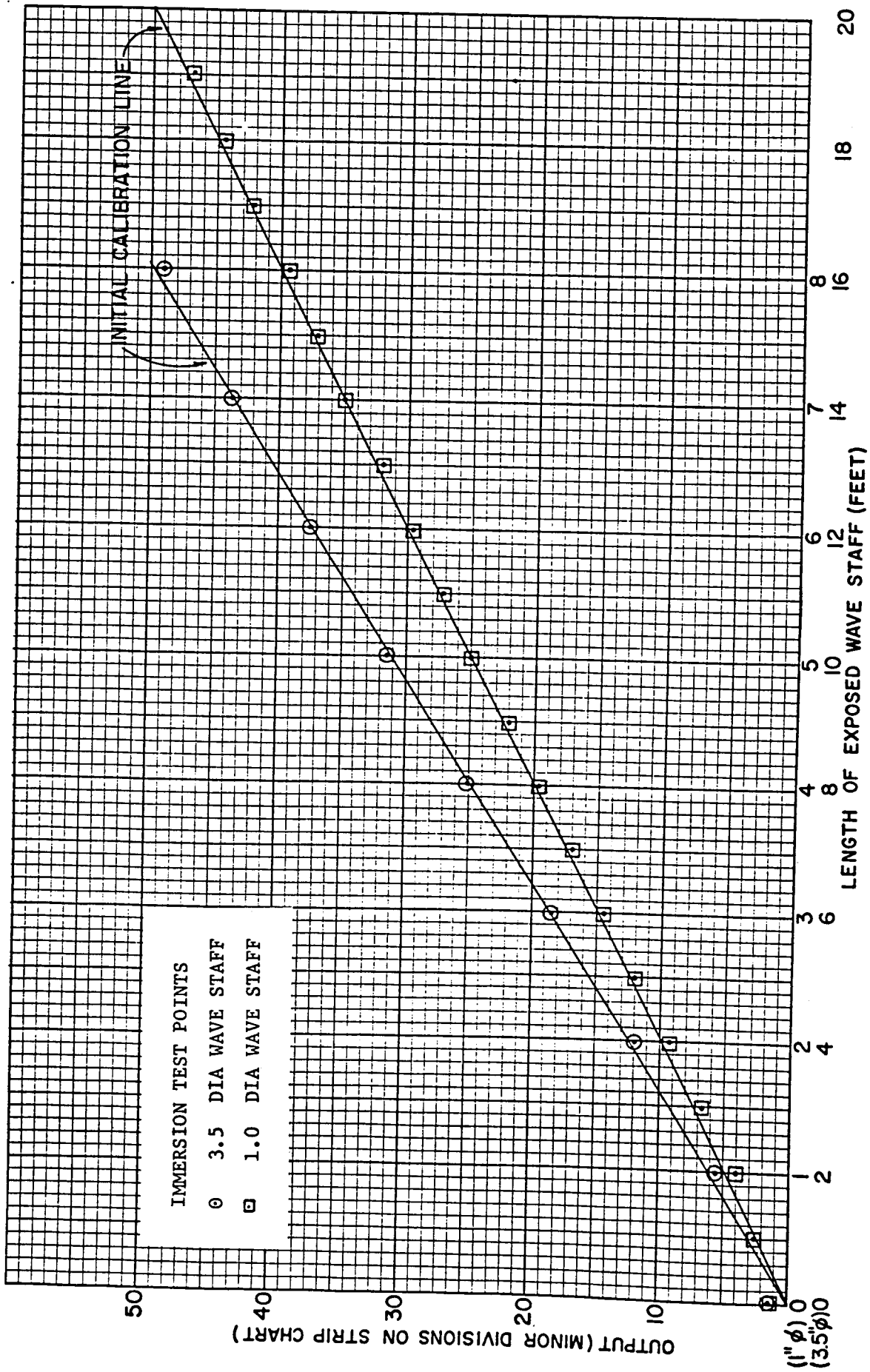


Figure III-10. Wave Staff Calibration Curves

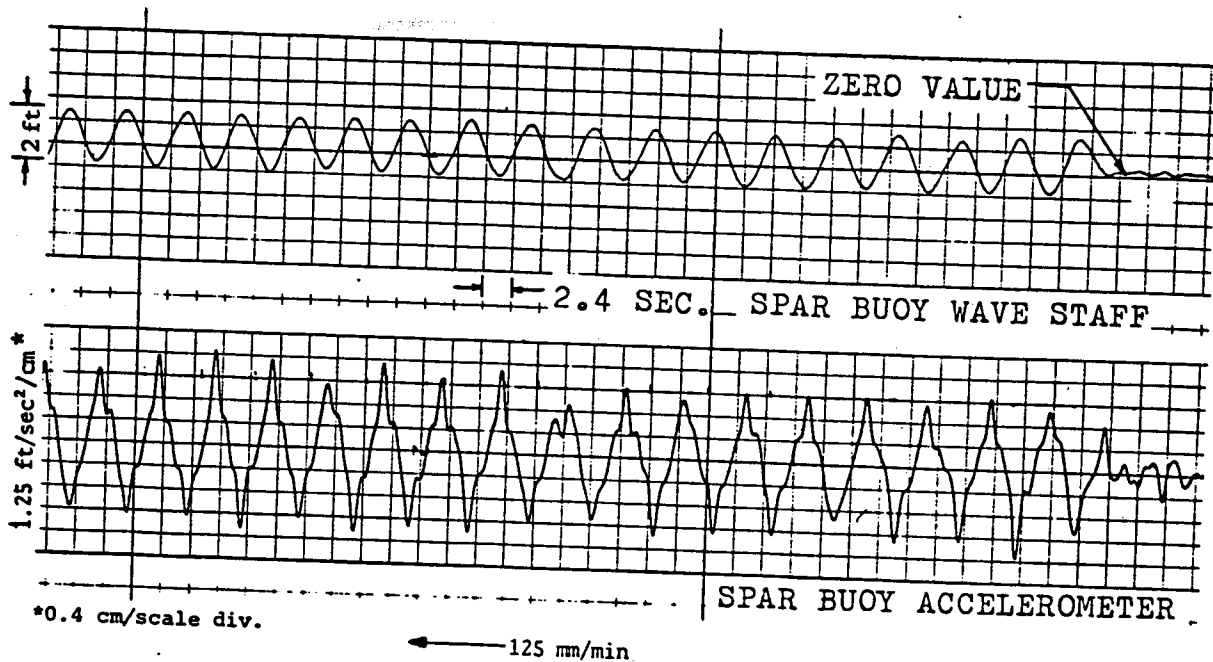


Figure III-11. Wave and Accelerometer Data for Spar Buoy Test (Plus and Minus One Foot Motion)

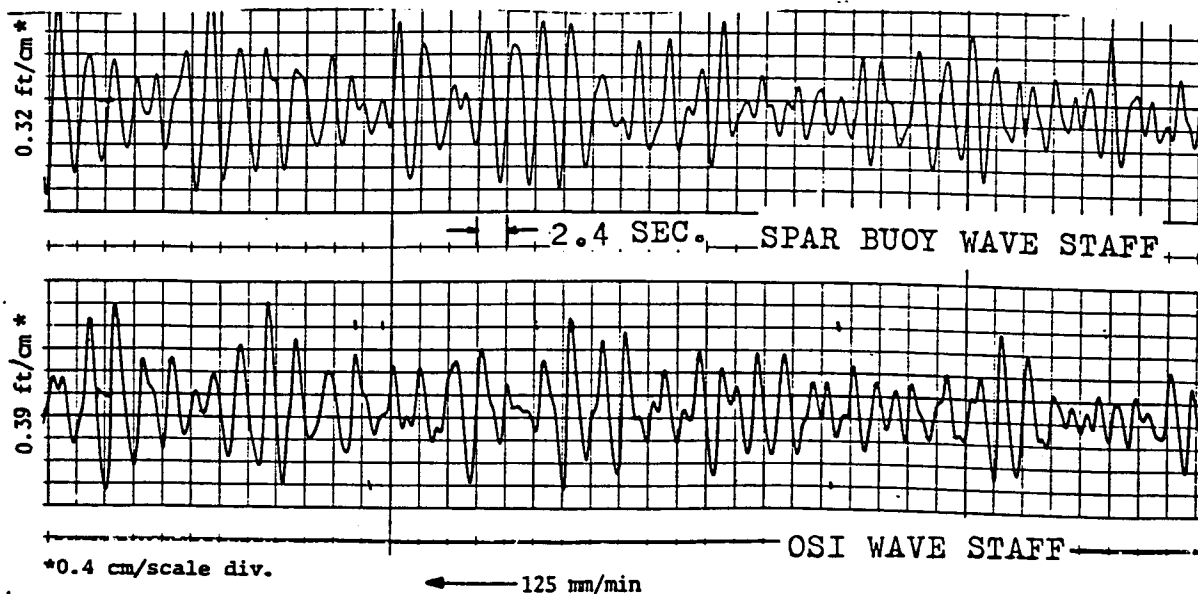


Figure III-12. Wind Wave Data for Spar Buoy and Stationary Staff

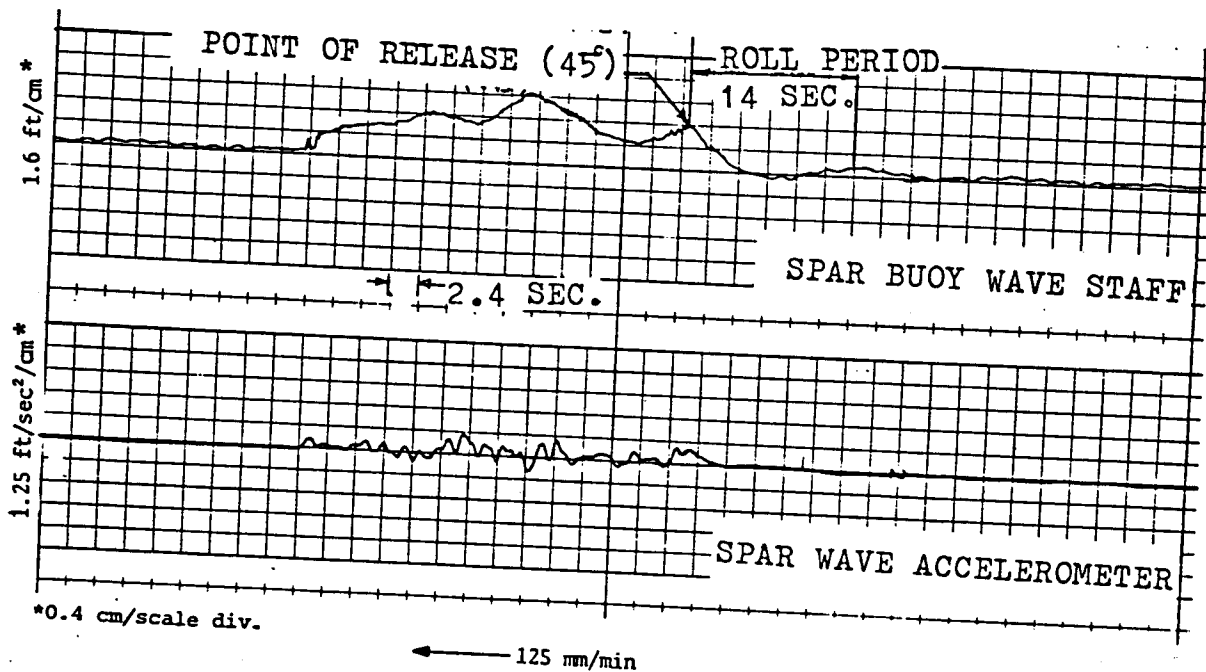


Figure III-13. Roll Response

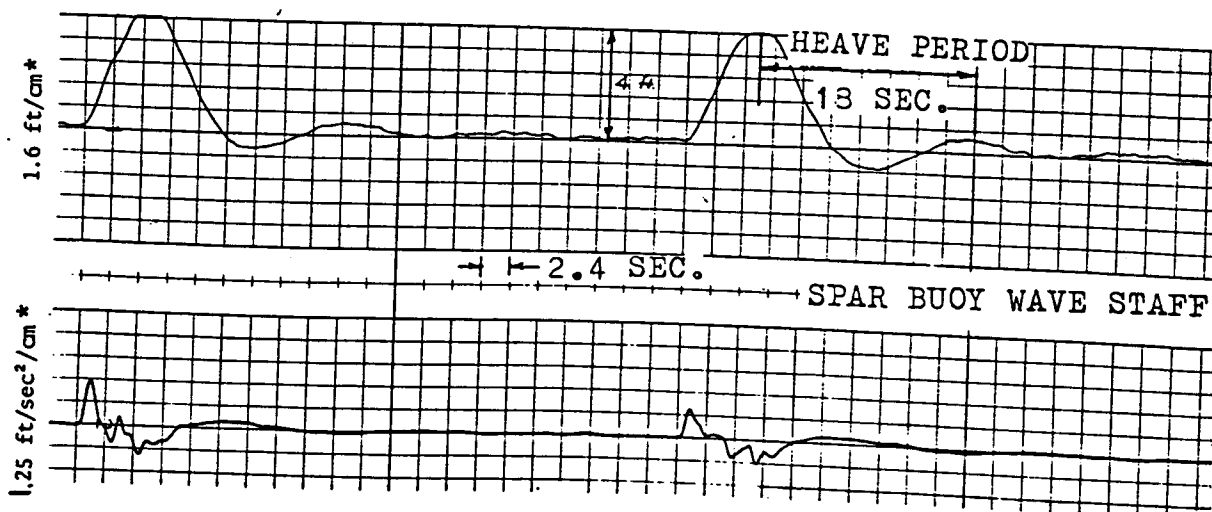


Figure III-14. Heave Response

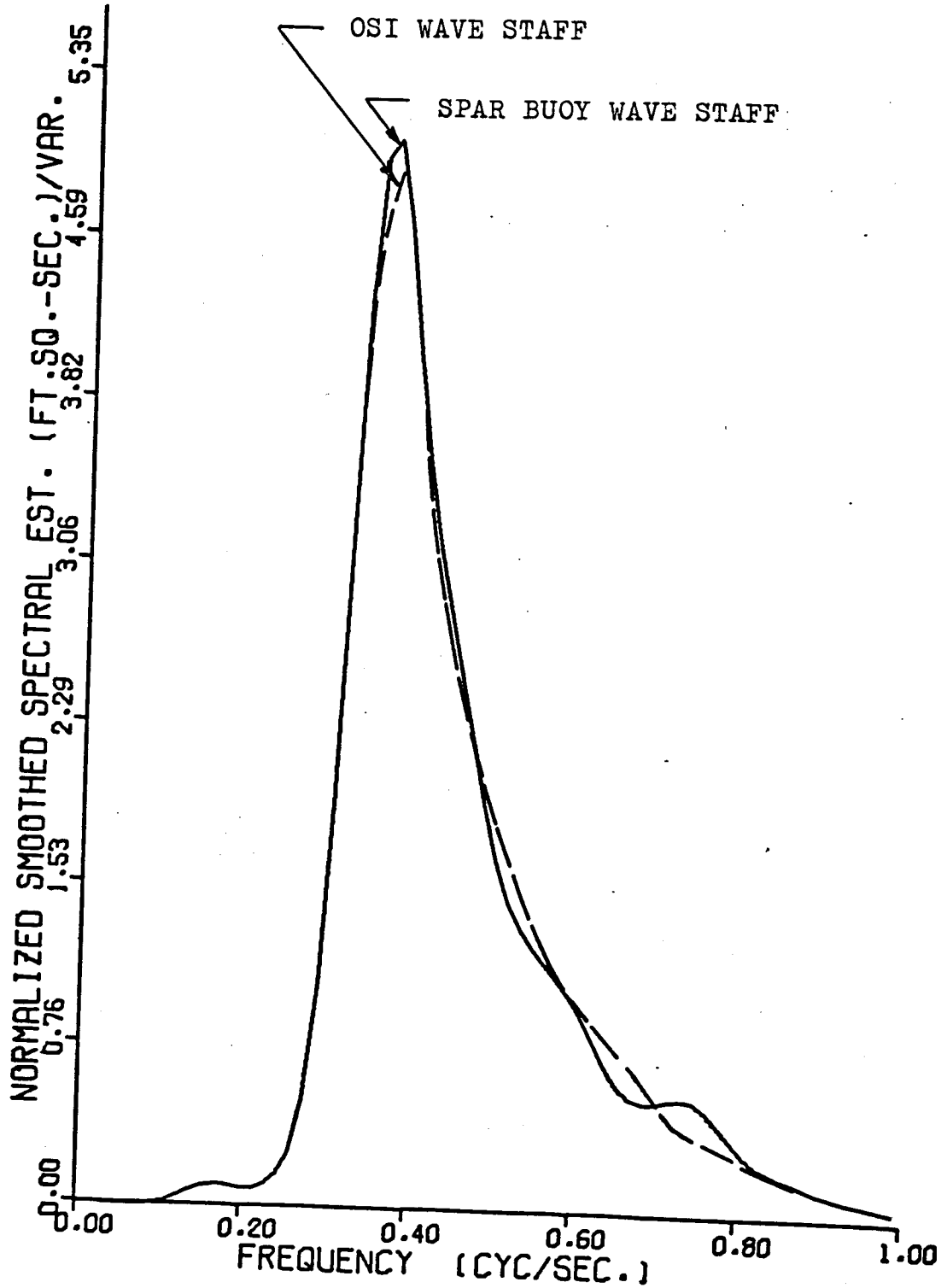


Figure III-15. Autospectral Estimate for Buoy and Staff Gages

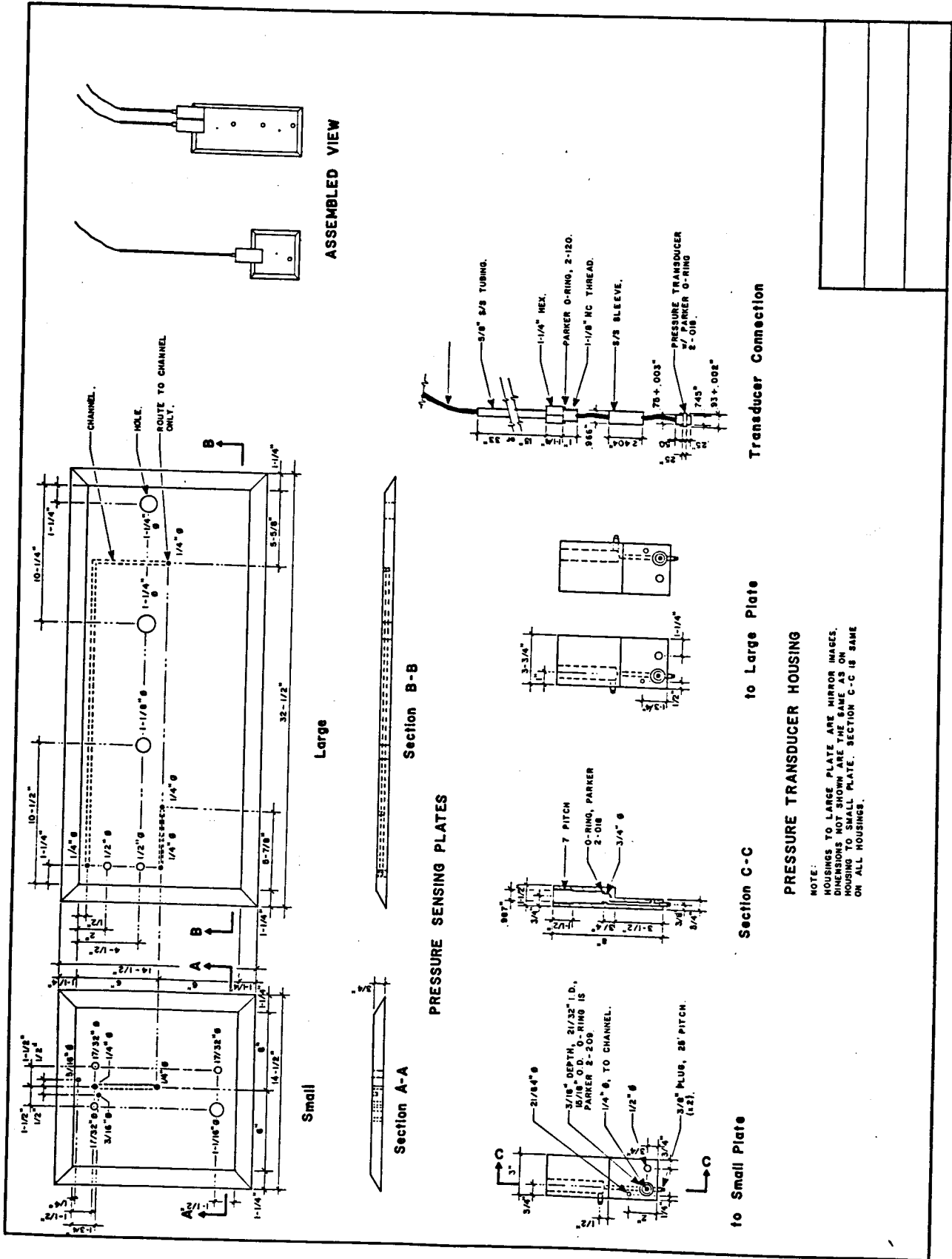


Figure III-16. Side Pressure Transducers Mounting Detail

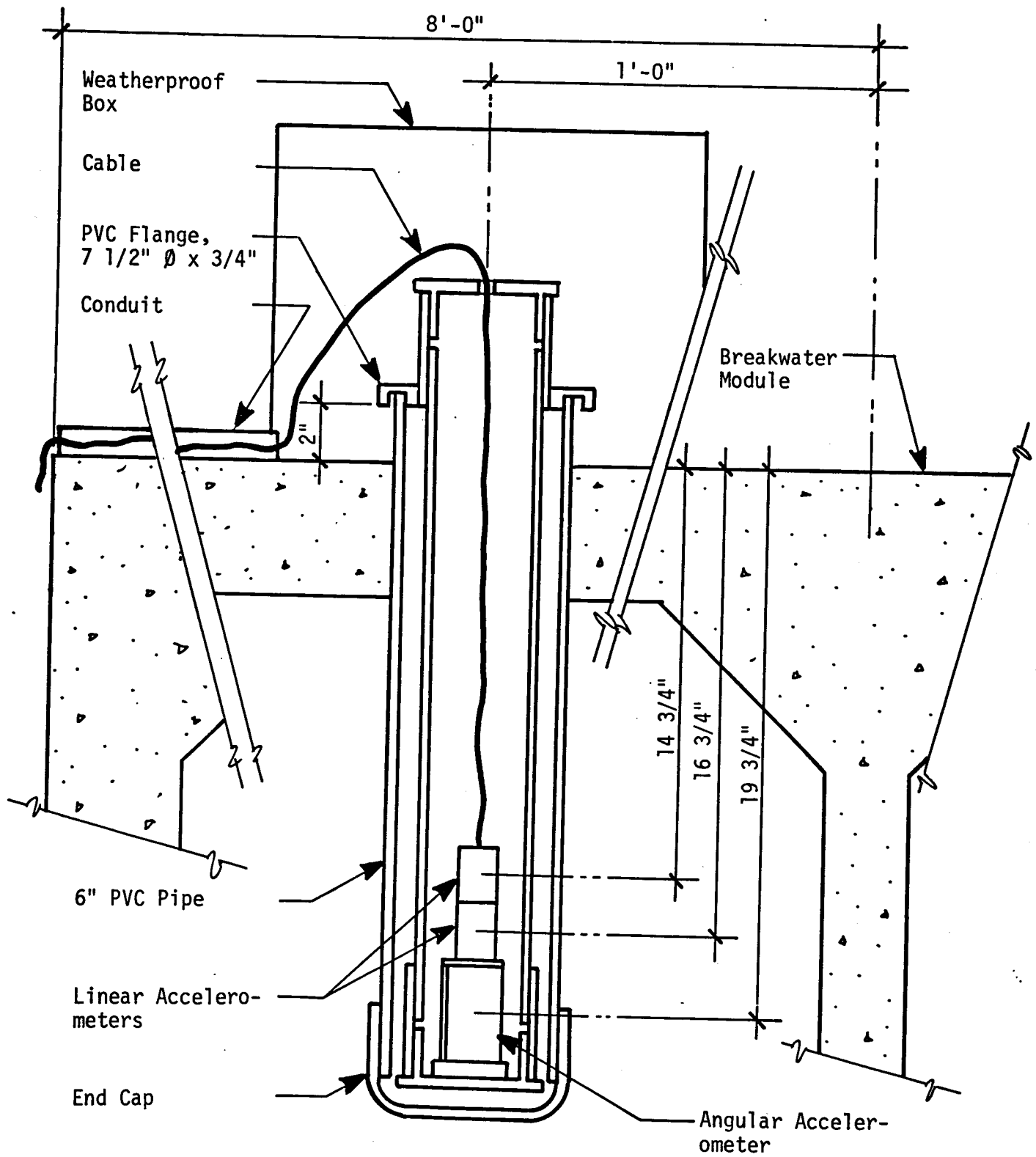


Figure III-18. Acceleration Transducer Mounting Detail

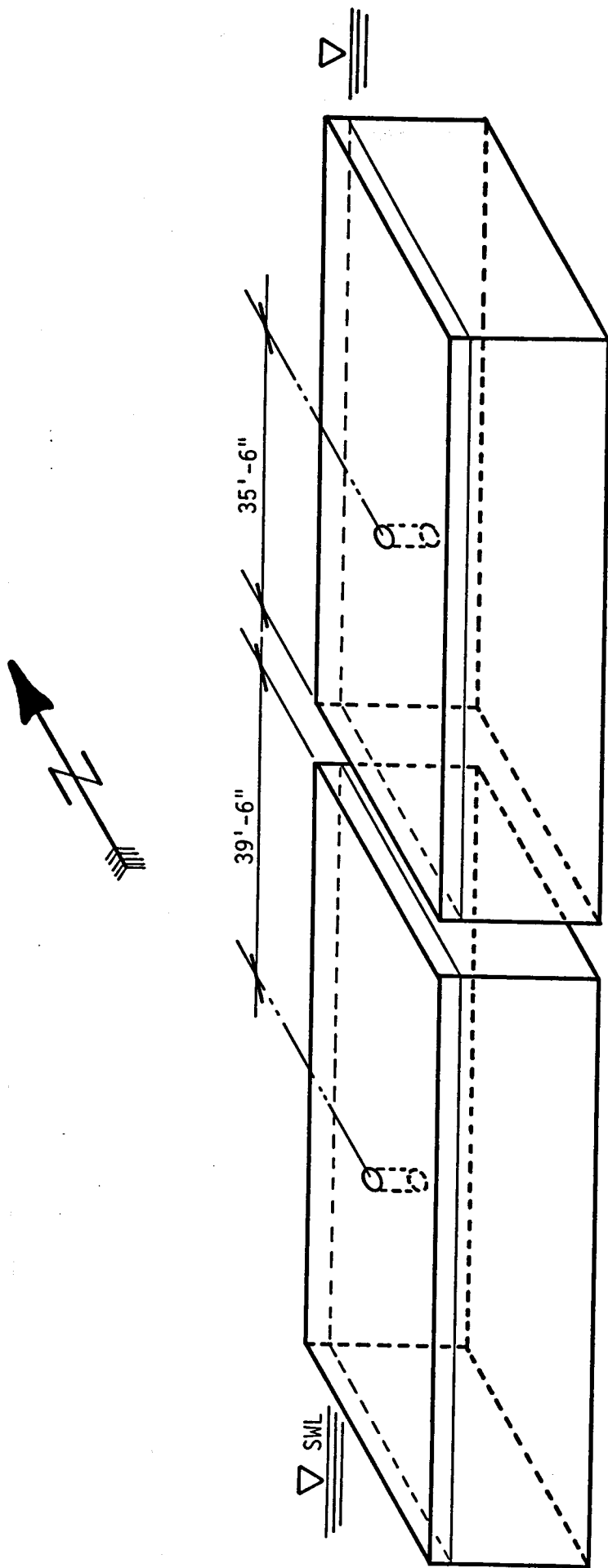


Figure III-18.b. Acceleration Transducer Layout

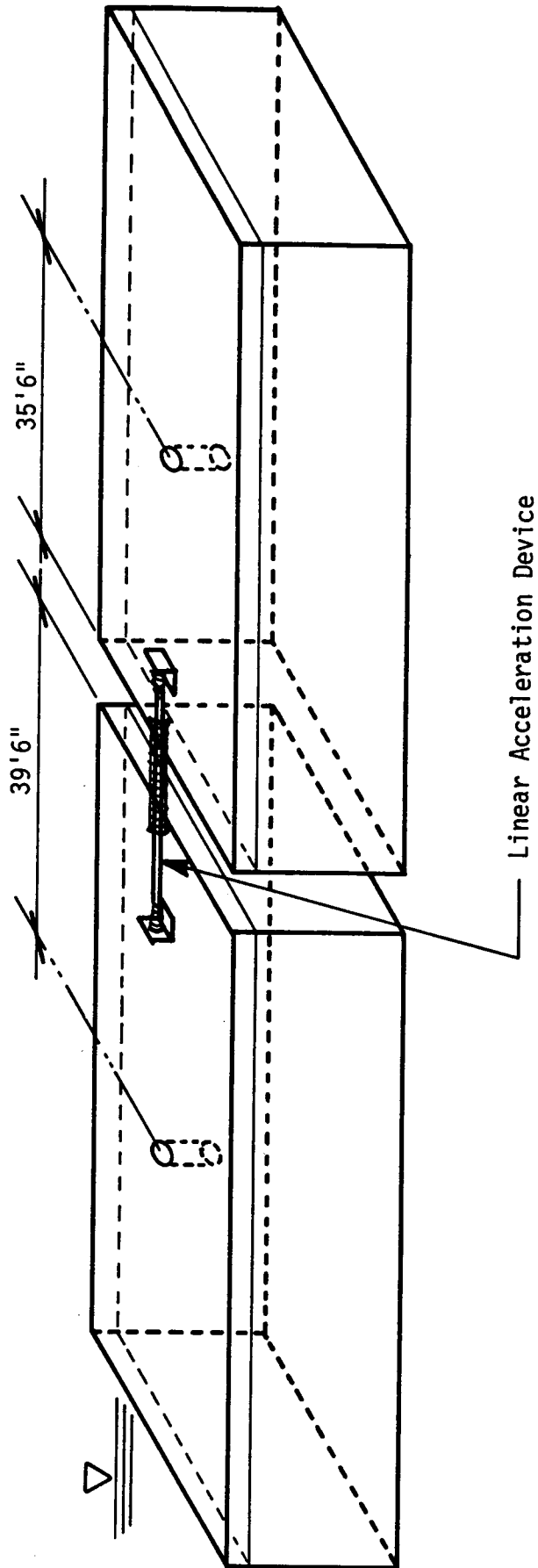
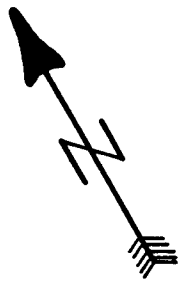
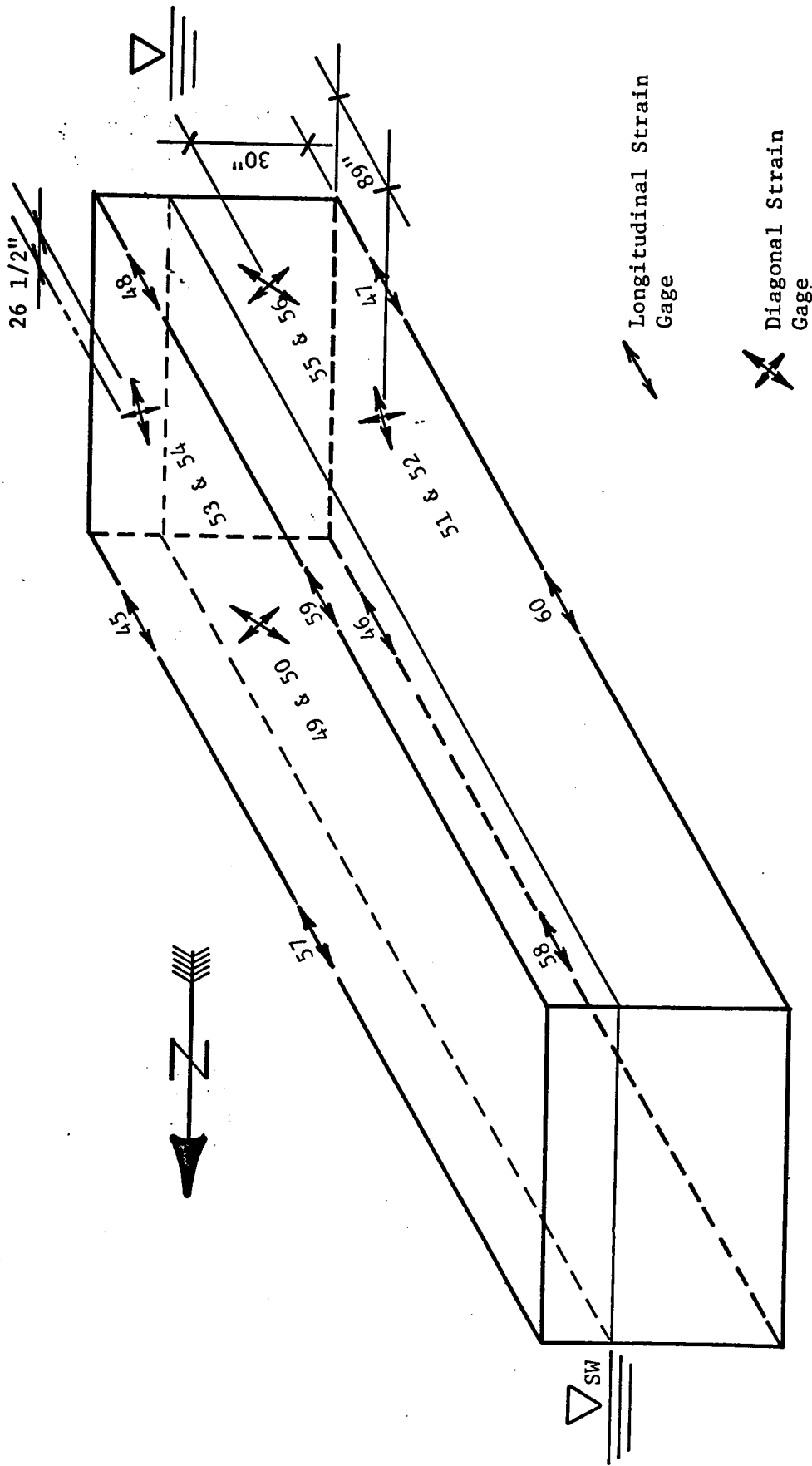


Figure III-19b. Relative Motion Device Schematic Layout



↔ Longitudinal Strain Gage

✕ Diagonal Strain Gage

60 Channel Number

Figure III-20a. Concrete Strain Gage Layout

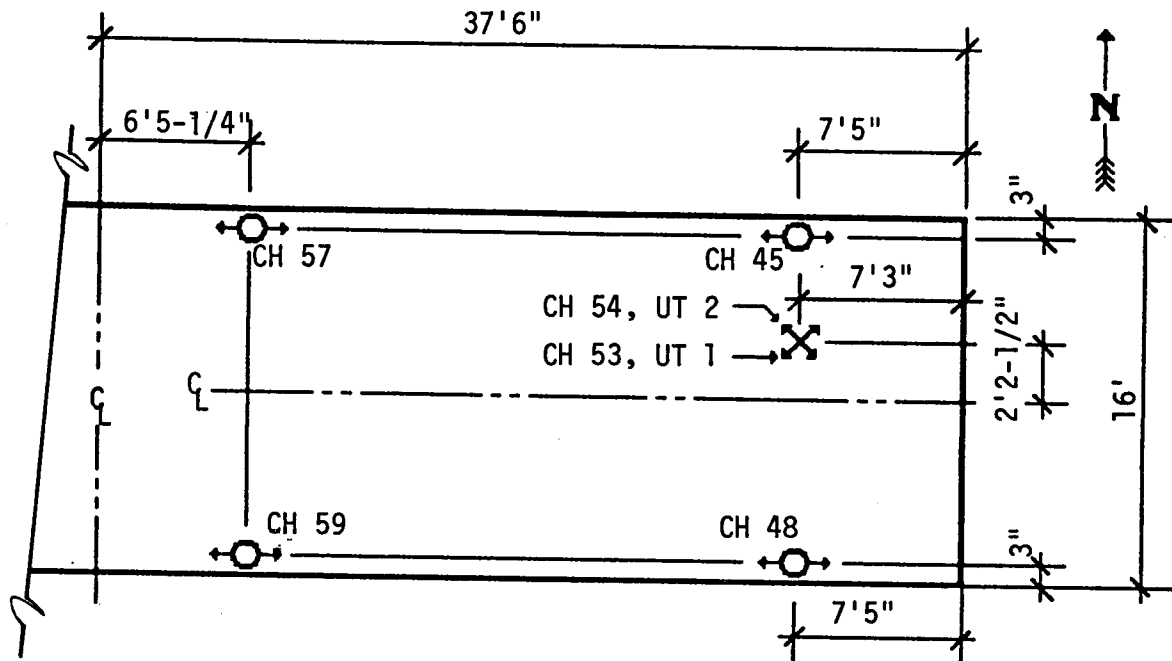


Figure III-20b. Concrete Strain Gage Layout-Top Plan

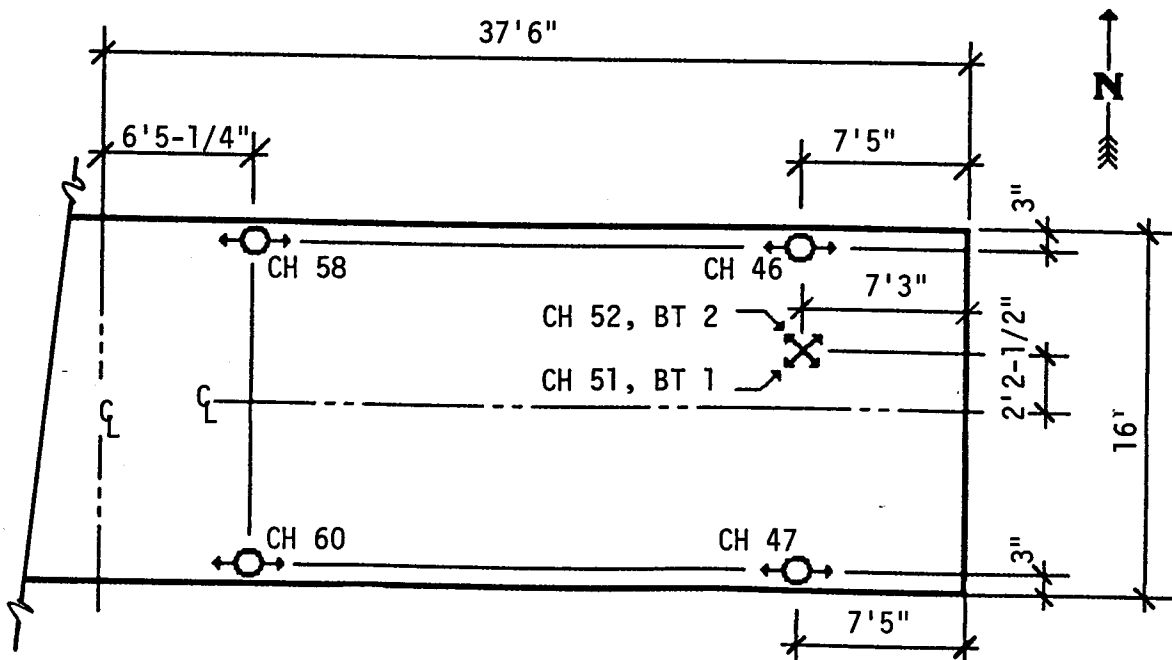


Figure III-20c. Concrete Strain Gage Layout-Bottom Plan

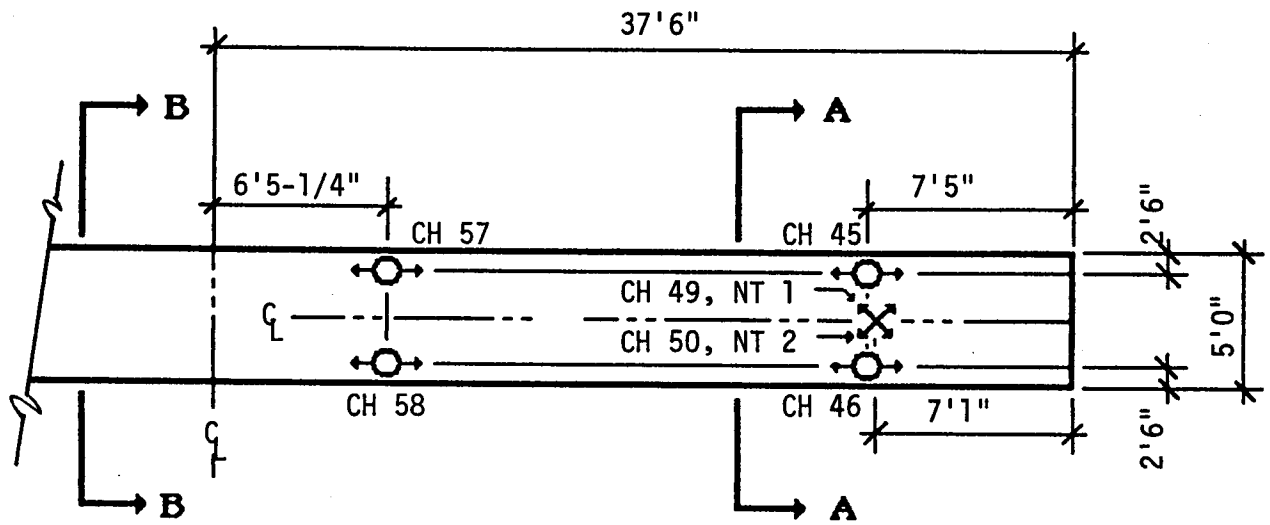


Figure III-20d. Concrete Strain Gage Layout-North Elevation

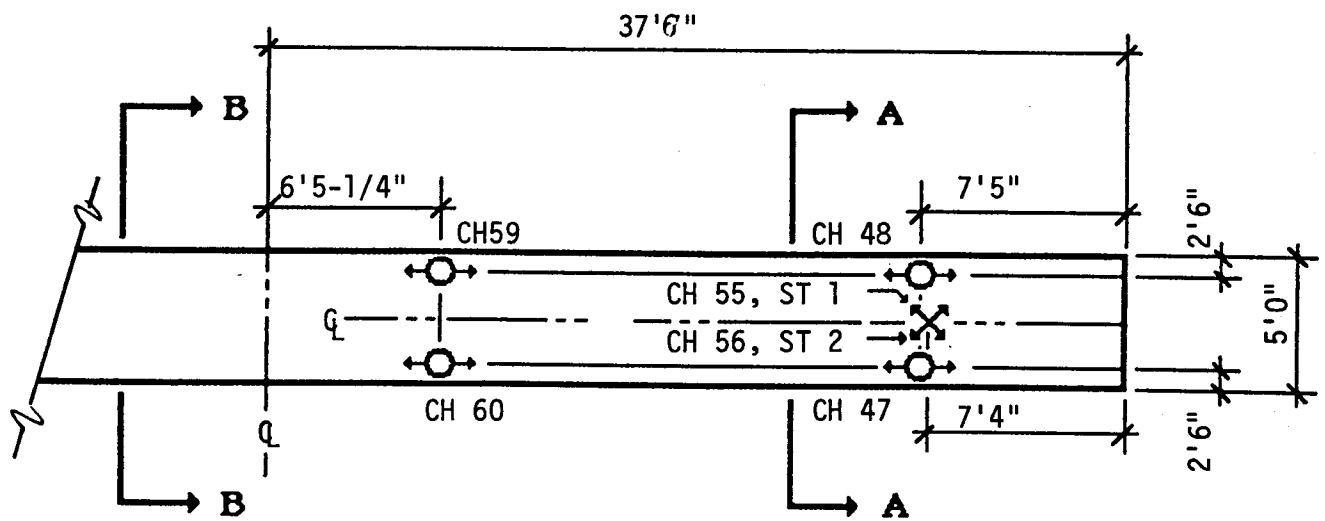


Figure III-20e Concrete Strain Gage Layout-South Elevation

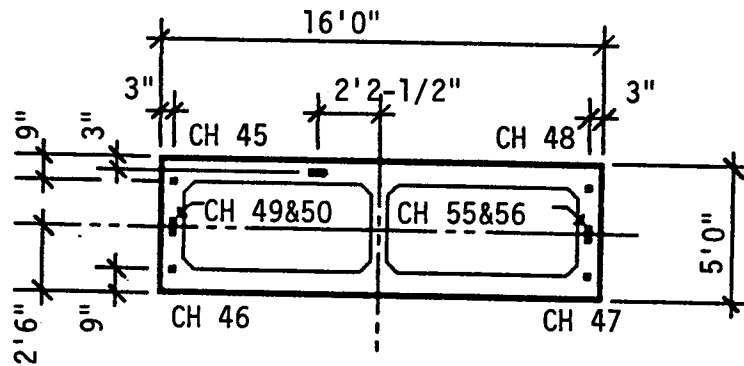


Figure III-20f. Concrete Strain Gage Layout-Section A-A

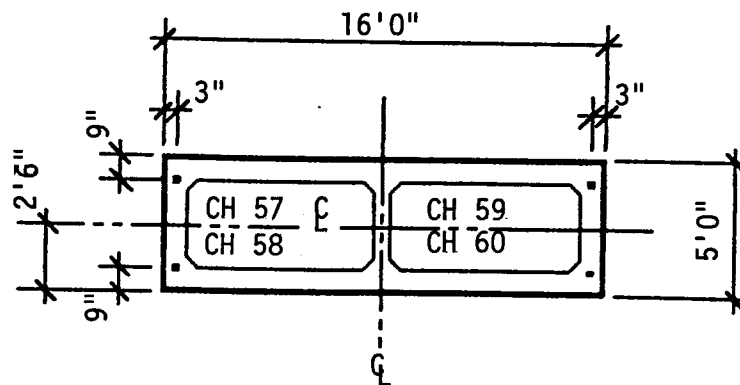


Figure III-20g. Concrete Strain Gage Layout-Section B-B

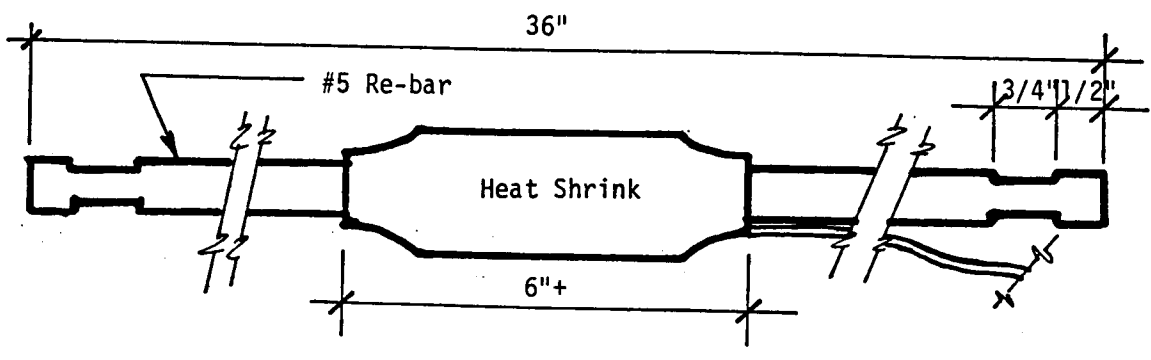
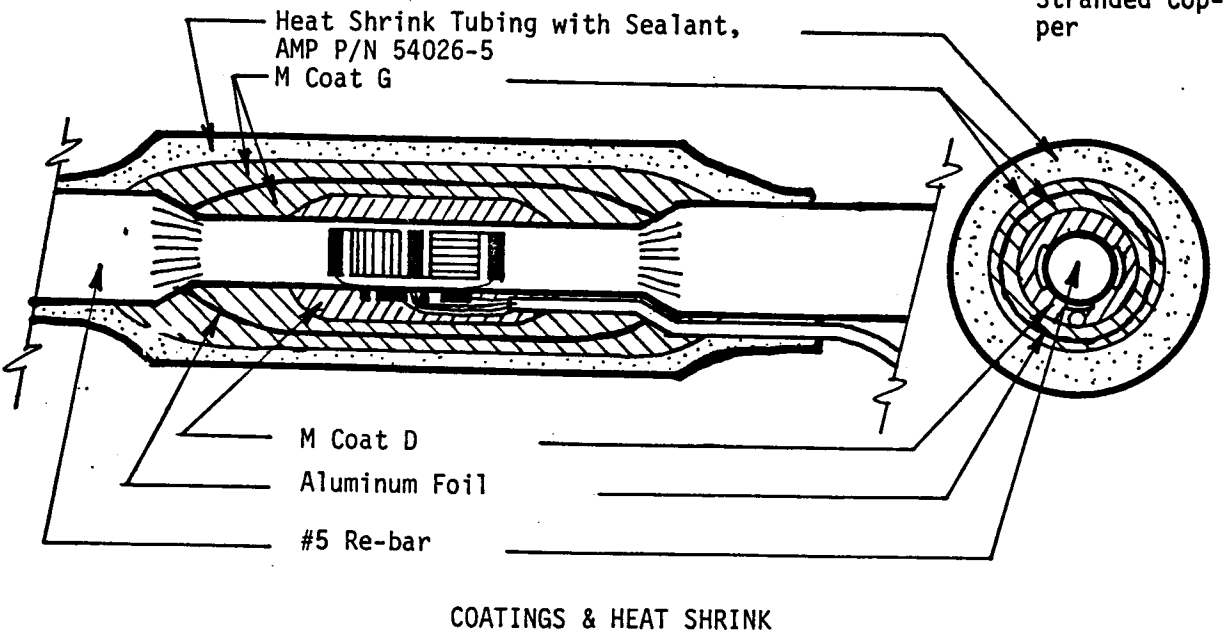
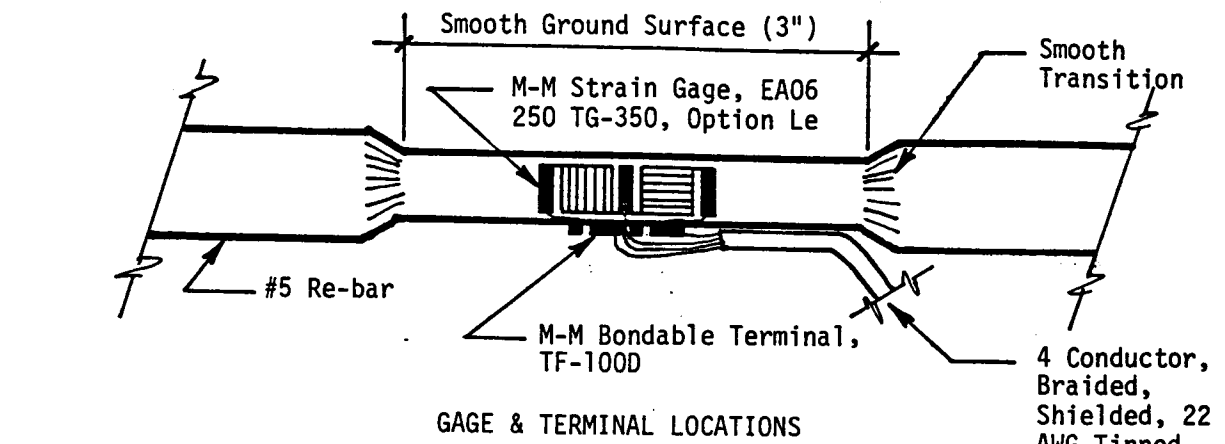
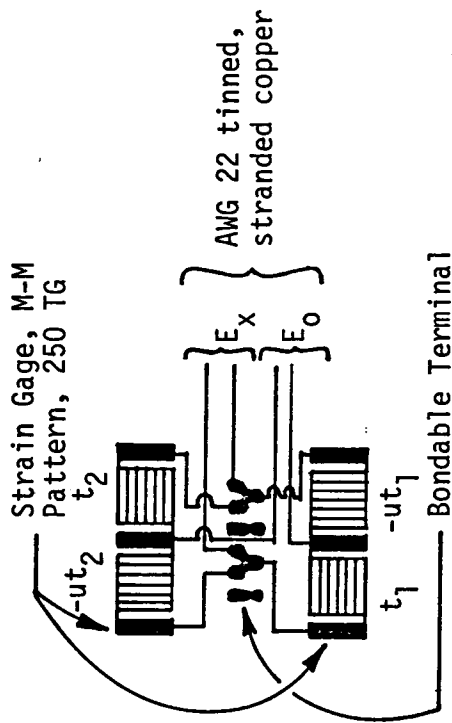
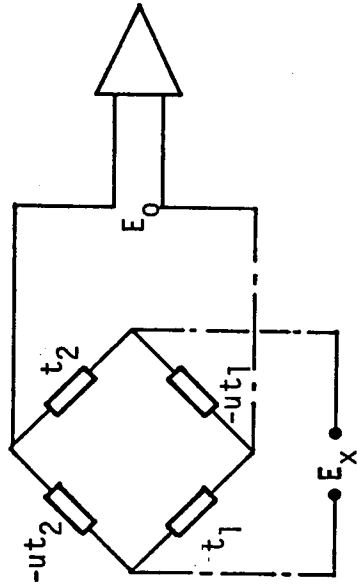


Figure III-21a. Concrete Strain Gage Mechanical Detail



Longitudinal Axis of Bridge

WIRING DIAGRAM FOR STRAIN MEASURING DEVICE



E_o = Excitation voltage

E_x = Output voltage

t = Strain on the axis parallel to the longitudinal bridge axis

$-ut$ = Strain on the axis normal to the longitudinal axis

SCHEMATIC DIAGRAM FOR STRAIN MEASURING DEVICE

Figure III-21b. Concrete Strain Gage Electronic Detail

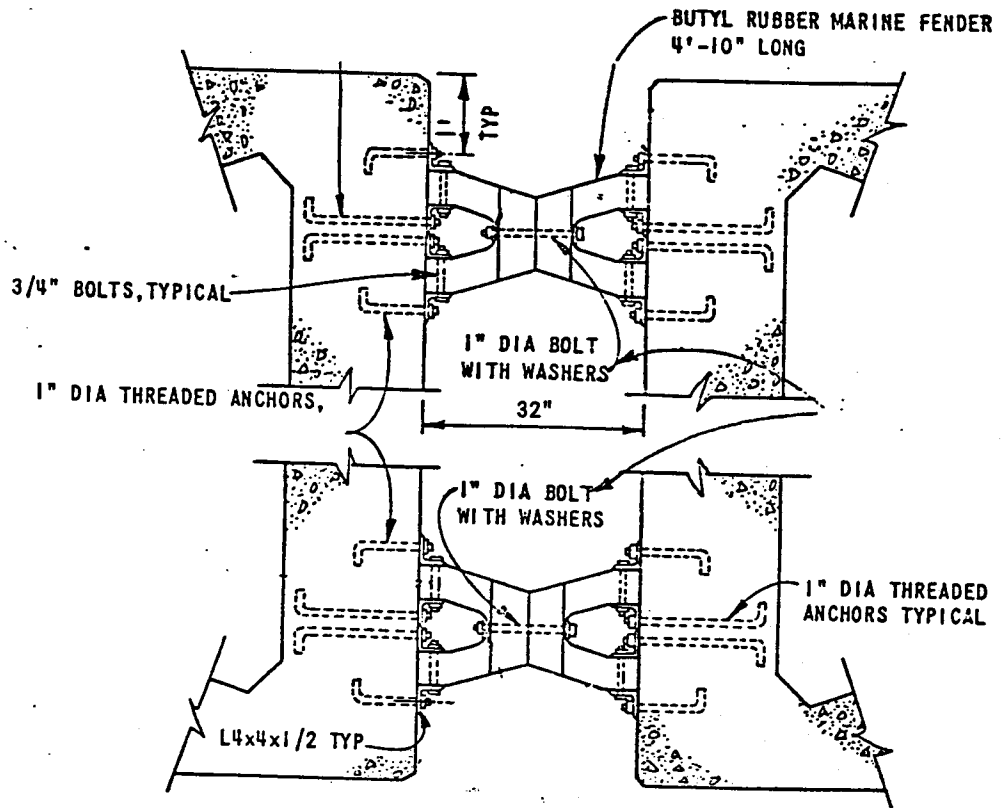


Figure III-22. Original Rubber Connector and Load Bolt Detail

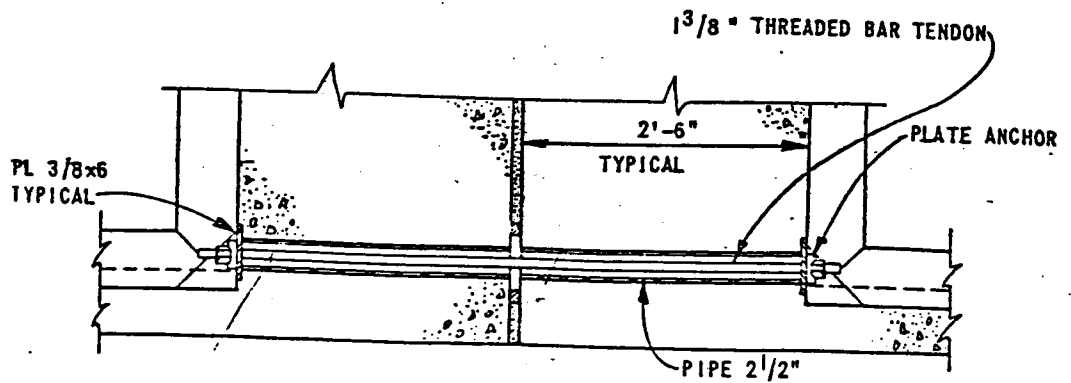


Figure III-23. Rigid Connector Load Bolt Detail

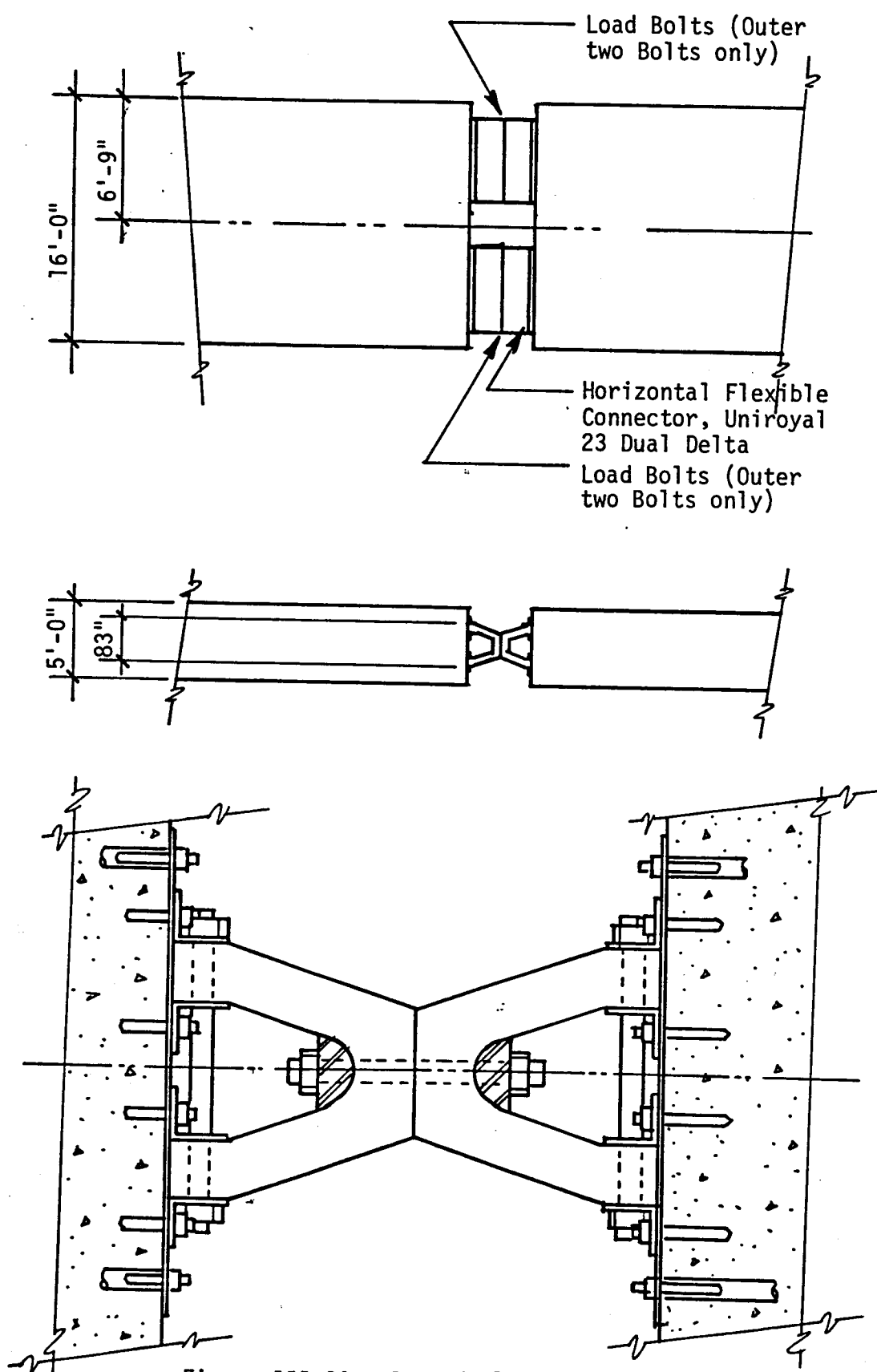


Figure III-24. Second Flexible Connector and Load Bolt Detail

CHAPTER IV

DATA ACQUISITION SYSTEM

IV.1 Recording System

The major factors in making the initial decisions associated with the selection of a data acquisition were:

- a) Low power on site battery operated system.
- b) Size and space was minimal. The system had to fit through a 24-inch diameter hatch cover and fit inside a four-foot high compartment.
- c) On site data storage of a minimum of 10 Mega bytes of data storage in a compact removable format.
- d) Hardwired connections to all transducers. To do otherwise would have been cost prohibitive.
- e) Off the shelf Hardware wherever possible.
- f) Hold cost to a minimum.

Based on this and the current state-of-the-art at the time (Fall 1981) the RCA microboard system was chosen. This was the only complete board level, low power CMOS system available at that time. All the boards or components associated with the data acquisition system, other than some (see Appendix IIA) minor signal logic and interfacing hardware, were manufactured by RCA. The software/firmware required for this application, however, had to be developed in-house. The basic hardware configuration and all relative schematics are given in Appendix A.

IV.2 Signal Conditioning System

The major design criteria here was space, low power requirements, versatility in application and ease of maintenance and repair. Based on this, it was decided to build a single board that would work with all the different input transducers. The signal conditioning board that was designed for this application is made up of five distinct sections. The first is a bridge completion and breadboard area, an instrumentation amplifier section, a buffered high level signal conditioning section, a universal active analog filter section and a separate sample and hold circuit to minimize digital noise problems and to ensure simultaneous sampling of all 80 channels. All schematics and details are given in Appendix A.

Packaging. Size watertightness and corrosion resistance were the major restrictions on the packaging selection. The necessity to fit the units through a 24 inch-hatch eliminated any known off the shelf watertight enclosure. The obvious advantages

of separating the digital and analog sections (minimize electronic noise) also mandated two separate packages. For this reason, it was decided to use two separate PVC housing and water-tight plastic connectors. See Appendix IA for details.

IV.4 Software/Firmware

General Information. A 96 channel data acquisition system (referred to as DAS hereafter) capable of sampling each channel at a rate of up to 8 hz. was developed. The DAS consists of 6 analog to digital conversion microboards (A/D cards) with 16 channels on each card. Channels 1-16 have a resolution of 12 bits (0-4095 counts) and the remaining channels have a resolution of 8 bits (0-255 counts). Most of the parameters that determine the operating configuration are switch selectable. Once familiar with the software, further changes can be made by writing on to different pre-set random access memory (RAM) locations using either the RCA Microterminal or a standard video terminal.

Software. The DAS can be activated by turning the main power switch on on the front panel while pressing the [RST] on the Microterminal and then pressing [RP].

The driving software can be divided into 5 categories depending on the functions performed by each segment, as follows:

- a) Main (Calling) Program: This program initializes the tape-drive (rewinds to beginning of tape) , sets the real time clock to 00:00:00, defines scale factors and turns off the sampling interrupt signal and power if they are on. In addition, the number of timeseries samples are set to 2048 and the counters determining the number of STAT and TIMSER executions (see (b) and (c)) are set to zero. (The state of the digital line EF3, which can be controlled by a switch on the front panel, determines whether to skip the above initializing procedure or not. Such a situation arises only if the DAS is reset for trouble shooting and data acquisition has to continue on the same tape cartridge after that.) It will then wait for a pulse on the digital line EF2 to activate a sampling program called 1 Minute Statistical Program (hereafter referred to as the STAT program) which is described below.
- b) STAT Program: This program samples a selected group of channels (channel numbers are given elsewhere) for 1 minute and computes the basic statistics (Mean, Standard Deviation, Maximum and Minimum). Instantaneous samples are taken on another group of channels. After writing the data on tape it will then examine the statistics to check whether pre-set criteria are satisfied in order to trigger the program that collects a full timeseries of 2048 (4096 prior to 12/82) samples of Channels 1-80. This program called TIMSER is described below. A switch on the front panel generates a pulse on EF2 enabling

manual activation of STAT. Under normal circumstances this is generated by a real time clock at a front panel selectable interval (usually 1 or 2 hrs.)

c) **TIMESER Program:** This program is triggered by a pulse on the digital line EF4 usually generated by program STAT. A switch on the front panel can generate this signal enabling manual activation if necessary. A pre-set number of samples (number of samples are defaulted to 2048. This can be changed by writing on the relevant RAM location described at a later stage) are collected from Channels 1-80 and written on tape.

d) **MODEM Program:** The DAS can be interrogated by telephone using a 300 baud full duplex terminal and a modem. This function has the least priority in the working of the DAS so that channel sampling is not interrupted. If the modem does not answer after two rings, the DAS is probably in STAT or TIMESER. The caller should then hang up and try again at a later time. The program is menu driven and self explanatory. Utmost care should be exercised in inputting parameters since wrong parameters can result in a system crash. The following capabilities are available at present:

1. List last STAT record written on tape
2. Run new STAT record and list
3. Single channel scan
4. Examine/change TIMESER triggers
5. Alter number of TIMESER samples and Time
6. Comment
7. Leave message
8. Read message
9. Examine memory locations
10. Write on RAM
11. Change scale factors
12. Run TIMESER
13. Real time data
14. Quit

e) **Utility Program:** This program drives the RCA Microterminal and can be activated by pressing [RST] and then [RU] on the Microterminal. This is usually used to examine memory and CPU states in the field for trouble shooting. It can also be used to write on RAM to change no. of samples, time etc. See RCA Microterminal manual for further information.

The attached flow chart describes the utilization of program functions described in (a) through (d).

Tape Record Format. All the data on the tape cartridges have been written in 512 byte chunks. This 512 record size was determined as the optimum size compatible with both the tape

Channel No.	Quantity	No. of bytes	Description
3, 4, 16 (12 bit)	Mean	3	$B_1B_2.B_3$
	Std. dev.	3	$B_1B_2.B_3$ 10 bytes
	Max.	2	B_1B_2
	Min.	2	B_1B_2
61, 73, 75 (8 bit)	Mean	2	$B_1.B_2$
	Std. dev.	2	$B_1.B_2$ 6 bytes
	Max.	1	B_1
	Min.	1	B_1
Instantaneous samples			
77, 78, 81-89		1	B_1
3 channels @ 10 bytes/channels = 30 bytes			
3 channels @ 6 byte/channels = 18 bytes			
11 instantaneous sample bytes = 11 bytes			
Total		= 59 bytes	

For example the the first three bytes of the 59 data bytes give mean of Channel 3 as (in decimal)

$$\text{Mean} = (256*B_1 + B_2).B_3$$

Note that B_3 denotes the digits to the right of the decimal point. Similarly standard deviation & or Channel 3 can be computed using the second three bytes. 8 bit channel computations are straight forward. For example 31 st and 32 nd bytes of the 59 data bytes give the mean of Channel 63 as

$$\text{Mean} = B_1.B_2$$

The above number format was used to avoid the use of time consuming floating point number routines.

(2) TIMESER

These data records consist of 1 header record (512 bytes) followed by 384 data records (512 bytes per record). The header record has the following format:

#

	No. of bytes
Identification word - TIMESER (84/73/77/69/83/69/82)	7
Associated STAT record no. (0 if TIMESER was manually activated)	2
TIMESER record no.	2
Tape track	1
Time (Yr,mths/10,mths,Day of week, Days/10, Days, (hrs./10,hrs.,min./10,min)	10
Scale factors (96 channels, 2 bytes per channel-all set to unity)	192
Number of samples per channel taken	2
Zeros	rest
	<hr/> 512

Sampled data of Channels 1-80 follows the header. The channels with 12 bit resolution (Channels 1-16) require 2 bytes while the remaining channels require only 1 byte per sample. Data is written on the tape in the following channel order per 80 sample scans:

$$i, i+16, i+32, i+48, i+64 \quad i = 1, 2, \dots, 16$$

i.e. 1,17,33,49,65, 2,18,34,50,66,.....,16,32,48,64,80

The same order is repeated for each successive scan (2048 of them). This order minimized the effect of the time required for analog to digital conversion in each card.

TIMESER triggering criteria used by STAT. TIMESER triggering criteria is computed from the statistics computed by STAT.

Let x_i = Instantaneous sample value of Channel i

$\langle x_i \rangle$ = Mean of Channel i

s_i = Standard deviation of Channel i

Triggering occurs if:

a) $3500 < \langle x_3 \rangle < 4092$

b) $3500 < \langle x_4 \rangle < 4092$

c) $4 s_{16} > 164$

d) $\langle x_{73} \rangle > 25$

$$e) \quad x_{77} < 255 \quad x_{78} < 255$$

$$\quad \quad \quad \text{SQRT}(x_{77}^2 + x_{78}^2) > 102$$

All numbers are unscaled values.

Only triggers (c) and (d) were active during the West Point Breakwater Monitoring Program.

Memory Allocation. The following memory partitions have been used in the DAS (all locations are in hexadecimal).

0000 - 2FFF	BASIC Interpreter
3000 - 67FF	Modem and other miscellaneous software
6800 - 6DFF	Data acquisition software
7000 - 7FFF	DURACOM clock location
8000 - 87FF	Utility program (UT 5)
8800 - 9000	RAM used by utility program
9000 - 93FF	BASIC work page
9400 - EFFF	RAM
F000 - FFFF	Stack

Important RAM Locations

<u>Location</u>	<u>Quantity</u>
A006	Tape track
A00A, A00B	No. of TIMESER samples
A010, A011	No. of STAT executions
A012, A013	No. of TIMESER executions.

Clock locations:

700E	Start/ Stop
700D	Years
700C	Tens of months
700B	Units of months
700A	Days of week
7009	Tens of Days
7008	Units of days
7007	Tens of hrs.
7006	Units of hrs.
7005	Tens of mins.
7004	Units of mins.
7003	Tens of secs.

7002
7001

Units of secs.
Tenths of secs. (read only)

Updates and Changes. Effective 03/11/83:

In STAT,

Channel 3 was replaced by Channel 7

Channel 4 was replaced by Channel 8

TIMESER trigger (c) was changed to

4 s₁₆ < 164 <x₁₆> <4092

Effective 10/14/83:

Identification words in both STAT and TIMESER programs were changed as follows:

STATIST----->STATHDR

TIMESER----->TIMEHDR

Also, instantaneous sample values for Channels 83-96 were added to STAT. This increased the total number of data bytes from 59 to 73. These were written on the tape immediately after the previously assigned 59 byte locations.

CHAPTER V

ANALYSIS SYSTEM

V.1 System Description

The computer system used for reading and converting data tapes and for doing the analysis has the following major components.

1. CompuPro dual processor microcomputer system.
2. Qantex Model 100 cartridge tape reader.
3. Alloy Engineering, S-100, cartridge tape controller and software.
4. HP7470 digital plotter.
5. Cypher 9 track 1/2 magnetic tape drive.
6. Tape reading and sorting software.
7. Spectral analysis software.

The operation of these relative pieces of h/w is covered in the individual manufacturers' users manuals and literature which is summarized in the appendices and which has been provided with the equipment. The operation of the actual software for reading, converting and analyzing the data is covered in the following sections with complete program source listings and example runs given in the appendices.

V.3 Software

User's manual for Qantex Tape Reading Software. The following pertains to a group of computer programs designed to run with the Gifford Computer Systems Compupro System 8/16. The package consists of 4 programs: QIF, BWSORT, SRCON, and SSP whose purpose are to read and interpret data from a tape cartridge recorded by a remote data acquisition system. Program QIF actually controls and communicates with the tape drive. Program BWSORT converts and arranges timeseries data into a random access file with file records corresponding to channel numbers. Program SRCON writes a map and summary of one-minute record headers. Program SSP computes basic statistics, Fast Fourier transforms, filtering, etc. on timeseries data. The output of program QIF therefore provides the input for programs BWSORT and SRCON. The output of program BWSORT provides the input to program SSP. The operation instructions for program SSP are provided in a separate section. Since programs SRCON and BWSORT require no console input, the bulk of the present document is primarily concerned with the correct operation of program QIF.

Reference is made in this document to two files intended for use with the MP/M submit facility. These files are entitled 'TSER' and 'TMAP'. They are invoked by typing "submit tmap" or "submit tser". The use of submit files merely removes the need for the operator to issue a specific sequence of commands from system level, but there is no reason why the operator cannot 'manually' issue the same sequence contained in these files without the benefit of the submit facility.

Program QIF was written for the purpose of controlling a Qantex Model 150 Cartridge Drive. The Qantex Model 150 is manufactured by North Atlantic Industries, Inc. and is controlled via a DS-100 card manufactured by Alloy Engineering Inc. which is plugged into an S-100 bus slot on the Compupro System 8/16. The DS-100 card provides common RAM between the Qantex and the computer through which commands, status information, and data are exchanged. Although the DS-100 card comes equipped with 'canned' software enabling it to be used directly for file back-up on the Compupro, the nonstandard data format used by the data acquisition system which generates the tapes required that a set of special software drivers be written.

The Alloy Tape Utility Protocol was utilized for this purpose. This protocol is described in Section 5 of the Tape Interchange Package (TIP) Operators Guide by Alloy Engineering, Inc. (see attachment A). A three byte command sequence is issued to the drive, and the drive responds with a two byte status report. If the first two bytes of the command sequence remain unchanged, it is only necessary to transmit the third byte. The status bytes which are returned by the drive preceding command execution are referred to as the drive status and the interface status or DS and IS. These values will be displayed at various points during the operation of Program QIF and should be noted by the operator if any unusual or unacceptable events occur. The meaning of these bytes is detailed on pages 39-40 in the TIP manual.

The software for the DS-100 card is designed to operate in CP/M 2.2 and is therefore limited to an 8 bit processor. For this reason, program QIF is written in Microsoft Fortran 80, an 8-bit Fortran which can only utilize 64K of RAM. Program QIF processes little data, its functions being restricted to tape drive operation and dumping raw data to disk. Major processing is done subsequently by other programs which can utilize the capabilities of the 8088 16-bit processor. This method of processing provides for much faster overall execution times.

Once the Qantex drive is turned on, turning it off has been known to crash the Compupro, necessitating a reset. It is a good idea, therefore, to turn it off at the end of the day or else at a time when all users are not engaged in some critical process.

Definitions. The following terms will be employed in this document with the following meanings:

- record: A 512-byte block of data, whether a header or data.
- header: A record containing a certain initial sequence which marks it as containing summary and/or control information. This sequence is either 'statist' in the case of 1 minute records or 'timeser' for timeseries headers. For a full description of what is contained in headers byte by byte, see the field data acquisition system section.
- 1 minute: Record: a header containing data sampled from certain key channels. This sampling is used by the data acquisition system to determine if a timeseries is to be written. It is the data from 1 minute records which is used in the One-minute summary printouts.
- timeseries: A sequence of records which contains samples from all 80 breakwater data channels. A timeseries consists of a one-record header followed by a sequence of records containing actual sampled data. The header records the date, time, and number of samples as well as other information of interest. Most timeseries on the West Point Prototype breakwater project are either 384 records (2048 samples) or 768 records (4096 samples) not counting the initial header record. Additional information on the order in which timeseries bytes are written may be found in the writeup on the field data acquisition system.
- sample: A set of values for each of 80 channels taken at the same time. There are $5 \frac{1}{3}$ samples per record or 16 samples for every 3 records.
- mask: A sequence of bytes which is transmitted to the drive following a search command. The drive then searches records for this sequence, stopping when it is found.
- tape number: All data tapes are numbered, with numbers ranging from 1 to 121. This number is requested by the program because certain details such as the exact format of the header mask depend upon it.
- glitch: For unknown reasons, possibly relating to cold temperatures or malfunctions of the on-site recorder, certain records are unreadable by the Qantex drive. When such a record is encountered, the tape motion becomes spasmodic, with a back and forth motion, and status bytes indicating 'abort with attempt', or sometimes 'abort without

attempt' are received. Such occurrences are referred as glitches and are mapped as a "*" when option 11, one minute rec. search and dump, is selected.

Operation. Program QIF is menu-driven and provides the operator with commands allowing tape positioning, searching for headers, mapping of header locations, and dumping of data to disk. No facility is provided for writing of data to the tape by the computer as the program is intended as a means of retrieving data written by the data acquisition system rather than a means of back-up. QIF may be invoked directly by typing 'QIF' from a user area and drive which contains the file 'QIF.COM'. This is useful when experimenting with the program. If a header map of a tape is desired, a more convenient method is to type 'submit tmap'. This will also initiate the subsequent processing step, program SRCON which interprets the data written by program QIF and generates one-minute record summary printouts. If the process is not initiated as a submit file as described, then the user must himself type SRCON to initiate the second processing step. To dump a timeseries to disk, the user should first make certain that the 'N' drive has enough space (this will be described in detail below in the section on timeseries) and then type 'submit tser'. In this manner, not only will QIF be invoked, but programs BWSORT and SSP in turn as well.

When QIF first begins, it writes the message 'Program QIF vers. 1.4 for use with Qantex model 150'. After two status bytes are displayed (in response to an interface reset), the following main menu is then displayed:

```
1-Change Track or PA
2-Rewind Tape
3-Read next record & Display
4-Forward Space record
5-Reverse Space record
6-Search for record
7-Write Timeseries to disk
8-Reset Interface
9-1 min rec. Search & Dump
10-Stop
enter choice _
```

The tape drive is now ready to respond to commands. Every command consists of three parts: the MA, the PA, and the CA. The MA tells the Qantex which track is being used (there are 4 tracks which will be referred to as tracks 1-4) and which drive (assuming you have more than one hooked up to the DS-100). As currently configured, program QIF assumes only one drive. The track is set as track 1 and will stay that way unless changed by a) the user by means of menu option 1 or b) the program during a 1 minute record search and dump or c) by the program if a time-series which is being written to disk happens to span two adjacent tracks. The PA tells the drive 'how many data records'. The PA set to 1 record at the beginning of the program. It is

not necessary to change it unless you wish to forward or reverse space some other number of records than one. It does not matter what the PA has been set to except when forward or reverse spacing. Once the PA has been set, however, that value remains in effect for all record spacing commands until changed, even when there are other intervening commands. If you change the tape track, you will also have to change the PA. If you give an invalid PA (outside the range of 1 to 256), it will be assumed to be 1.

When program QIF is first initiated, it is always a good idea to check to make sure the interface is working correctly before attempting to initiate menu options 7 or 9. This may be easily done by, for example, setting the track to 1, the PA to 200 and executing a forward space record, followed by a rewind. The tape should visibly and audibly move forward during the forward spacing, and return to its original position after a rewind is issued. Furthermore, DS and IS values should be displayed after both motions, and they should be non-zero. If all of this occurs, then the interface is most likely functioning properly and the drive is ready for further commands. If there is a problem in getting this far, it is advisable to 1) abort the program 2) remove the tape 3) turn off tape drive and wait 5 seconds 4) turn drive back on 5) replace tape and 6) reinitiate program in that order. If after two or three times you have no success, check to make sure there is cabling running from the drive to the computer, that the drive is turned on and plugged in etc. Sometimes the interface can be stubborn getting started. No definite reason why this is the case has been found: possible reasons include head misalignment, tape mispositioning or something similar. Such problems are not common, however.

An important fact which is sometimes confusing to new users is that the tape format is 'serpentine'. This means that the beginning of track two is at the end of the tape. This is an advantage, as it removes the need for the tape to be rewound after each track is finished before more data can be read or written. Once a user is familiar with this convention, it may be used for rapid positioning within a tape. Often it is advantageous to switch between tracks one and two or three and four when positioning, switching back to the track desired when the proper location is approached. The ability to quickly and accurately locate specific timeseries headers on the tape with the use of the map and the search for record command can be developed with a little practice.

Whenever the tape is in motion, MP/M is 'suspended' and all users will find their terminals dead until the command is finished. This is generally not noticeable except when searches are being executed. The facility to quickly locate records, therefore, becomes a virtue which rapidly becomes appreciated by other users if there are any.

The commands will now be examined one by one.

1) Change Track or PA

In general this option will be used quite frequently. When QIF is first initiated, both the track and PA are set to one. To change either of these values, simply select menu item 1 and respond to the prompts. Valid track numbers range from 1 to 4 while valid PA's are from 1 to 256. Any time a spacing command is given in which the PA is greater than 8, a high-speed command is given. High-speed commands are somewhat approximate, so the user should bear in mind that if a PA of 200 is given followed by a forward space record, that about 200 records will be skipped.

2) Rewind Tape

Since the drive uses a serpentine format, a rewind issued when the MA is set for track 1 or 3 will result in the tape being rewound to the opposite spool as when the MA is set for track 2 or 4. The PA has no effect on a rewind command.

3) Read next record & Display

Selecting this option will result in a prompt requesting the number of records to be displayed. The program will then initiate a consecutive sequence of reads, reading and displaying to the console the number of 512 byte records specified by the user. The ASCII values are also displayed on the right.

4) Forward Space Record

The execution of this command results in the tape advancing forward PA records. If PA is larger than 8, a high-speed command is issued and the number of records spaced is then only approximately PA. If no records are present on the current track, the tape drive can get quite confused, due to the lack of any end-of-file marks. The PA may not be larger than 256.

5) Reverse Space Record

Exactly the same as forward space record except in the opposite direction.

6) Search for Record

The Qantex drive has the capability to search any data records for a specific sequence of bytes, skipping those records in which the sequence does not occur, and positioning itself on the first record exhibiting the proper mask or string which is being searched for. Program QIF is set up to search for either timeseries or 1 minute record headers. The user specifies whether a timeseries or 1 minute record header is to be searched for, and whether the search is to be for a specific 1 minute record number or merely for the next header of the appropriate type. The details of specifically which header mask is involved

is made by the program itself based upon the tape number input by the user. If the tape number is greater than 75 then headers have the form "stathdr" or "timehdr", but for tape numbers less than 75 "statist" and "timeser" is used. This is because the header masks were changed part way through the project.

If a glitch is encountered during a search, the user has the option of continuing the search or of terminating it.

Once the record is found, its contents are displayed at the console. Since the screen is not large enough to display the contents of an entire record, the user should be ready to type a ^s or control s to stop the display where desired. The display may then be reactivated with a ^q.

7) Write Timeseries to Disk

This option is most frequently used in the context of a 'submit tser' command. The following explanation will assume that program QIF was initiated in this manner.

The process of producing usable data summaries from field data cartridges has several steps. First the desired data must be located on the tape. Much of the preceding discussion has centered around this consideration. Once the data is located in program QIF, it is written directly to a file which is then read by the next program BWSORT for additional processing.

Program QIF merely dumps the numbers from the tape drive into a scratch file in exactly the same order that they are read off the tape. This order is dictated by the needs of the field sampling system, but a sort must subsequently be done by program BWSORT in order for the data to be grouped by channels. The files that Program QIF generates during a diskwrite are therefore temporary, being intended for use by program BWSORT only. Since the data must be read and written several times, the scratch files written by Program QIF are written to the 'N' drive which is a solid-state disk emulator. This 'drive' is not intended to be used as permanent storage since its contents are destroyed in the event that power is shut off or lost. The advantage of using the N drive in this manner is that Program BWSORT executes significantly faster than if the files were written to a floppy or even the hard disk. Program QIF is not used to do the sort directly because of memory limitations and because such a process would execute significantly slower on an 8-bit processor than it would by using the 16-bit capabilities of the Compupro.

To write a timeseries to disk, the following procedure is employed:

First, check the amount of memory available in the N drive. This is done by means of logging onto the N drive (by typing "N:") and then typing "stat". If 250 K bytes still remain on the N drive (about half the drive capacity) then there should be no problems for a timeseries of 385 records. For a timeseries of

769 records you will need just about all of the N drive. If less than these amounts are available, it is necessary to delete some files from the N drive. To find out which user areas contain files, type "stat usr:". With this information, go to the relevant user areas and type era "*.*" (but make sure you are still in the N drive when you do this or you may be erasing someone's permanent files). Alternatively, power the system down entirely and reboot. This will completely clear the N drive.

Next, go to user area 15, drive B. Type "submit tser" and Program QIF will be initiated. It is a good idea at this point to take the steps described above to make sure the interface is working properly. Next, locate the timeseries of interest by referring to the beginning of the one-minute record summary printouts, which contains a map of all headers found on the tape during a 1-minute record search and dump (if this has not already been done, you may want to do it first). Find out the tape track on which the record of interest is located. It is also a good idea to form some notion of where on the track it might be located. This takes some experience, but again, the maps are helpful. All tapes after March 15, 1983 contain 384 data records per timeseries, plus one header record. Those tapes from before this date probably contain 768 data records per timeseries. If you know, therefore that the record you want is two timeseries in from the beginning of track 3, and the tape is from June, 1983, then the header that you need is approximately 2×385 or about 730 records in. This will not be exact, because in general there will be some intervening 1-minute record headers.

Change the tape track to 3, set the PA and rewind to the beginning of track 3. At this point there are two options. Either simply search for the timeseries header desired by specifying menu option number 6. This is the slow way. A faster method would be to set the PA to about 230, then execute 3 consecutive forward space records. This will advance the tape about 700 records. Then issue the search. Spacing commands will execute much faster than searches.

When the header is displayed, check to be sure that it is the correct one. Each timeseries header has two associated numbers: the number of the associated 1-minute record header immediately preceding it, and the timeseries header number. The header number that is searched for is always the 1-minute record header number. The first 11 bytes of a timeseries will appear as follows:

54 49 4D 45 53 45 52 ## ## ## ## (tape number < 75)

or 54 49 4D 45 48 44 52 ## ## ## ## (tape number > 75)

The first seven bytes are the hexadecimal values for ASCII 'timeser'. The four spaces marked ## (bytes 8-11) are bytes containing the following information in hexadecimal:

1 minute record number = byte 8*256 + byte 9

timeseries number = byte 10*256 + byte 11

If you requested a search for a timeseries corresponding to a specific one-minute record number, then that number should match up with the information contained in bytes 8 and 9 as described above. You are now in a position to initiate a diskwrite. If you attempt to initiate a write with the tape in the wrong position, you will get a message from the program asking you if you really want to do this. Unless you are making modifications to program QIF and are debugging, it is suggested that you bail out as conveniently suggested by the program. Otherwise, meaningless data will be written into certain control files and program BWSORT, if invoked, will get very confused.

When a diskwrite is initiated, QIF prompts for certain necessary information such as tape number (if it has not already done so), tape dates and times etc. You might make a point of writing down the tape dates and times from the tape (if you don't have a log available) before you start the session. Once the tape is inserted in the Qantex, you won't be able to see what's written on the back of the cartridge and you will have to reposition the tape again if you remove it to find this information.

Program QIF will next issue the following prompt:

```
"enter number of records to be dumped
this timeseries contains ### records"
```

where ### is the number of data records contained in the timeseries. In general, you will be reading the entire time series and will just want to enter the number displayed by the program.

You are then asked for a drive letter for the output file. Usually this will be either 'D' or 'E' if you want results stored on a floppy. Make sure that a formatted floppy with adequate space is in the drive specified before the session starts and that you have typed "DSKRESET" from system level after you have inserted it. Unless you have 350K bytes remaining on the floppy for a timeseries of 385 records and about double this for a 769 record timeseries, it is suggested that you use another disk. The amount of space remaining on a drive may be found by logging onto that drive and typing "stat".

Next the program asks for any comments that you wish to be included in the final header file that is written to the output drive. Five lines are provided for this purpose. Either enter the comments desired with carriage returns, or simply enter five carriage returns if you have no comments.

The program then writes the data and stops itself. If you are running a 'submit tser', then Program BWSORT is automatically initiated and should run with no intervention (assuming a floppy

of adequate space is present in the drive specified or that the hard disk is not full if a hard drive was specified). When program BWSORT terminates execution, Program SSP is automatically invoked. The first question it asks is "what is the name of your data file".

Program BWSORT automatically generates filenames according to the following rules:

filename = <D:>BW<tape number>R<l min. record number>

where <D:> signifies the drive specified (if it is the current logged drive then it may be omitted). In addition, various extenders are added by program BWSORT such as (for example)

E:BW85R32.DAT
E:BW85R32.HDR

In this case, the timeseries read corresponds to tape 85, one minute record number 32. The '.DAT' signifies the main data file while the '.HDR' denotes the header. The format of the header file and further operation instructions are covered in the documentation for Program SSP.

8) Reset Interface

The interface is automatically reset by Program QIF at the time the program first begins. Sometimes it is useful for the operator to specify an interface reset if there appears to be a problem with the status bytes or if communication does not seem to be proceeding as expected. If this doesn't work, then often it is necessary to shut off the drive and start over as described above.

9) 1 min rec. search and dump

This option should normally be used in the context of a 'submit tmap' command. It causes files to be written to the logged drive which may be interpreted by the following program SRCON. Program SRCON tapes output from QIF and writes an ASCII file which may be spooled to the printer containing a map of all 1 minute record headers on the tape. If the tape number is greater than 75, all timeseries headers as well. The map shows which track a header was found on and the location of any glitches encountered, designated by a "*".

To use this option, first be prepared to answer the prompt regarding tape numbers, dates, times, etc. Before selecting option number 9, make sure the interface is working as described above, and then rewind to the beginning of track 1. Finally, select Option 9. The rest is automatic. Program QIF will stop itself and Program SRCON will be initiated. The output of program Program SRCON is a file called

BW<tape #>.LMR

This is an ASCII file which will appear on the B drive (or wherever Programs QIF and SRCON are run from). Two temporary files are left by Program QIF: "lMINREC.DAT" and "GMAP.DAT". These files are the input of Program SRCON. Once SRCON is finished, these files may be erased.

It is a good idea to use option 9 at times when system demand is low since other users will find their terminals almost totally dead during execution of this routine.

10) Stop

If Program QIF is being run as part of a submit file, either TMAP or TSER, then execution will be stopped automatically. If not, this option may be selected for an orderly exit from the program.

SSP User's Guide Overview. SSP (Simple Statistical Program) was developed in the fall of 1983 at the University of Washington to analyze data recorded on an instrumented floating breakwater built by the Army Corps of Engineers. SSP can compute summary statistics on the data as well as filtering, clipping and performing Fourier transformations on individual channels.

The data are originally recorded in a complicated format on 1/4" magnetic tape cartridges. Program QIF (documented elsewhere) reads these tapes and writes a scratch file of the data. This file is read by program BWSORT, which writes the data as a random access disk file of binary format integers with the file name BWxxxRyy.DAT, where xxx is the tape number and yy is the record number, and the .DAT suffix indicates that this is the data file. BWSORT also produces a header file (BWxxxRyy.HDR) of ASCII text that contains information such as when the tape was made and how many data points were recorded. The exact format of these files is described in the comments inside the source code for SSP.

In order for SSP to run, both the .DAT and .HDR files must be present on the same logical disk drive. SSP itself may reside on another drive.

SSP produces numerous output files, which are placed on the same drive that the .DAT and .HDR files are on. The names of most of these files is automatically determined from their contents. These files are:

BWxxxRyy.STS Contains statistical information in binary form. This is an internal workfile and cannot be edited or printed.

BWxxxRyy.OUT Is a printable two page report that lists summary statistics (min, max, mean and standard deviation) for each channel, as well as labels and scale

factors. It is produced automatically when summary statistics are computed.

BWxxxRyy.Czz Is an input file for program PLOTR that contains a plot of time series data from channel zz.

BWxxxRyy.Fzz Is a PLOTR input file of the Fourier transform of channel zz.

How to run SSP. After making sure that the .DAT and .HDR files are on the same drive, and that there is room on this drive for any output files you may create (100k is plenty), simply type SSP<cr>. (where <cr> stands for a carriage return) You should then see the message:

```
Simple Statistical Program v. 1.3
What is the name of your data file? _
```

where the underscore (_) indicates the final position of the cursor. You must now enter d:BWxxxRyy<cr> substituting the proper numbers for xxx and yy. The d: is the indicator for the drive where the input files exist and the output files are to be placed. If this is the currently logged drive then the d: does not need to be specified. The alphabetic part of the file name may be in either upper or lower case.

If the files are found then after a few seconds the main menu will come up on the screen, looking something like this:

- 1 Load a channel into an array
- 2 Summary statistics on one or all 80 channels
- 3 Scale an array
- 4 Extreme value smoothing
- 5 Detrend a time series
- 6 High/Low/Band pass filter on a time series
- 7 Write an array to the master data file
- 8 Perform a Fast Fourier Transform
- 9 Write raw FFT coefficients to a file
- 10 Compute and write cross-spectral phase and coherency
- 11 Execute a macro file
- 12 Plot an array
- 13 Quit

```
Array A: Unused
Array B: Unused
```

```
Enter option desired: _
```

This menu is the starting point for all functions in SSP. As well as listing the available functions, it shows the current status of the arrays which are used to hold the data of a channel while analysis is being done. When a channel is loaded into an array, the channel number and label will be displayed on the menu, as well as whether or not the data have been scaled and how

they have been transformed. This appears to the right of "Array A:" and "Array B:".

Description of Menu Options

1) Load a Channel into an Array

This reads an unscaled channel of data and places it into an array, overwriting whatever was there before. This must be done before any analysis can be done on a channel.

2) Summary Statistics on One or All 80 Channels

After selecting this you will be asked for either an array indicator (A or B) or an empty carriage return. If you enter an array designator, statistics for the data currently in that array will be printed on the screen. These statistics will reflect any transformations or changes made to the data since it has been loaded. If you just enter a carriage return then statistics will be computed for each channel of data from the master data file and will be printed on the BWxxxRyy.OUT file. This will overwrite anything stored in array A.

3) Scale an Array

This will multiply an entire array by the appropriate scale factor to turn it from a measure of raw counts into a measure in the correct units for the quantity measured. This should be done before a Fourier transform is performed.

4) Extreme Value Smoothing

This allows you to clip all data points in an array that fall outside of a certain range. Any values that are outside of the allowed range are set to a value calculated by interpolating a straight line between the first good data points on either side of the extreme-valued interval.

5) Detrend a Time Series

This function can be used to either remove the mean of the data in an array and hence center the data around zero, or to remove the least squares straight line to eliminate any drift in the data. The mean is automatically removed prior to an FFT, but removing the least squares straight line can often help to reduce low frequency noise in the computed spectrum.

6) High/Low/Band Pass Filter on a Time Series

This will filter out high and/or low frequency noise from a time series in an array. You may do high or low pass filters by specifying a filtering frequency outside the range of interest.

7) Write an Array to the Master Data File

This is the only function that makes changes to the .DAT file and as such it should be used with EXTREME caution. Always make a copy of your data file before altering it.

If you have invested much time in filtering, detrending and otherwise massaging an array of data, you may wish to save this so that it can be accessed easily in the future. This function will allow you to place your new data in the master data file, overwriting the original data. Since this alters the .DAT file, the corresponding .STS file is deleted so that the outdated statistics are not used in the future.

8) Perform a Fast Fourier Transform

Selecting this will allow you to perform a Fourier transformation of the data in either array, transforming the data from the time domain to the frequency domain. Options include performing the transformation on only a subrange of the time series, and doing FFTs using only every Nth point, which effectively decreases the sampling rate. Due to the algorithm used, N must be a power of 2, so you may use every 2nd, 4th or 8th, etc. point. With each doubling of N, the time required to produce the FFT (potentially as long as several minutes) falls by slightly more than half while the maximum frequency of the spectrum produced is halved. After the FFT is completed, FFT N will be printed after the array name on the main menu.

In order for the units of the Fourier coefficients to be correct, the time series data should be scaled before the transformation is done.

9) Write Raw FFT Coefficients to a File

It is sometimes necessary to do more processing on the Fourier coefficients than SSP can do and this procedure provides a way to record the raw coefficients so that another program can access them. The coefficients are written on a file of the user's choice in binary complex format. The file does not contain any information about the frequencies that the coefficients correspond to, so any program that uses this file must calculate this information itself.

10) Compute and Write Cross-spectral Phase and Coherency

This computes the cross-spectral phase and coherency between two channels. The two channels must have already been loaded into arrays A and B and have undergone identical Fourier transformations. The computed values will be written on a file of the user's choice in a format compatible with PLOTR so that a plot may be made. Details about how this is done are documented within the SSP source code.

11) Execute a Macro File

Sometimes it is necessary to perform the same series of functions on a number of channels. SSP v.1.3 provides a easy way to do this through the use of macro command files. Since SSP is menu-driven, these command files take the form of long lists of menu choices and are easily generated using programs such as MAKEMAC.

Once a macro file has been generated, it can be executed by choosing this menu option and entering its name. SSP will then take its commands from this file but will still write prompts and menus to the screen, so that you may gauge its progress, if desired.

The last command in the macro file should be either a Quit or an Execute macro with a file name of "ZZZZ". The quit will return you to the operating system and trying to execute a macro file named "ZZZZ" will return control to the terminal. If the macro file does not end of one of these two commands, it could cause the program to "hang" unrecoverably and ignore all input.

12) Plot an array

This routine will take the data currently in an array and write it to a file in X-Y coordinate pairs with the appropriate header information for program PLOTR. It also has the ability to plot only subranges of the data and you will be prompted for how many points (or for FFTs, what frequency range) you want plotted. To make a plot of the data, after leaving SSP execute PLOTR and when it asks for the file name, give it the file name (which will be of the form BWxxxRyy.Czz or BWxxxRyy.Fzz) which was printed on the screen by SSP.

13) Quit

Closes all files and returns the user to the operating system.

Technical details about how these functions work and interrelate may be found in the comments within the source code, with an overview of the program at the beginning of the file and details about the individual procedures in the procedures themselves.

PLOTR User's Guide

PLOTR is a relatively simple program designed to do plots of data written out by program SSP. The only reason that this ability was not included in the plotting routine in SSP is that the plotter driver software that we have is written in a different version of FORTRAN and indeed runs on a different processor than SSP.

PLOTR

Operation. When you invoke PLOTR you will see a menu asking

you to either open (and plot) a new file, finish a page, or quit. If you open a new file then you will be asked for the name of the file and the data from that file will be plotted. Once you have made enough plots to fill a page (one or two, depending on type) you should finish the page. On plotters with continuous forms capability, such as the C.Itoh CX-4800, this will cause the paper to advance to the beginning of the next sheet. Quitting closes all open files and terminates the program.

Input. PLOTR requires an SSP plotter file (either BWxxxRyy.Czz or BWxxxRyy.Fzz or a combined phase and coherency file) as input, and will prompt you for the name of this file. Unfortunately PLOTR does not have the capability to read files from other drives, so the input file must be on the currently logged drive, so when you type in the file name, do not include a drive designator.

After you have entered the file name, PLOTR will issue a message saying what is in the file and proceed to read the data. For complicated plots this can take several minutes. When it has finished reading the data it will ask whether the plot should be placed on the top or bottom of the page. (N.B. Row plots are done lengthwise on the page while FFTs are plotted one above the other. Combined phase and coherency plots take up an entire page so this question is not asked) On FFT plots you will also be asked to input a label for the vertical axis. This enables you to specify the units as you would like them to appear.

Output. PLOTR writes plotter commands to two places. One is the CP/M LST: device, enabling you to plot while the program is running. The other is a file named PLTiii.dev where iii is a number taken from a file named PLOTNUMB.ERS and dev is the plotting device selected (e.g. H-P for an HP 7470A or C-X for a C.Itoh CX-4800). The number in PLOTNUMB.ERS is incremented automatically every time a plot is made. The PLTiii.dev file can later be sent directly to the plotter if multiple copies are desired. Otherwise it can be purged to avoid cluttering up directories.

PLT2GTK

PLT2GTK (PLOTR to GrafTalk conversion program) was written to facilitate the unattended production of large numbers of plots of data from SSP. Very simply, PLT2GTK will scour a given drive area for all files of the form BWxxxRyy.tzz, where the file type t is either C or F, and combines them into a GrafTalk command file with the name BWxxxRyy.GTK. Then all the plots can be generated with the single command "GRAFTALK BWxxxRyy.GTK<CR>", requiring no operator intervention.

Operation. After PLT2GTK has been invoked, it will ask its only question, the name of the data set. At this point you should enter "d:BWxxxRyy." where d is the drive where the files are located and xxx and yy have been replaced with the appropriate tape and record numbers. PLT2GTK will then start searching for

files with this stem, updating the screen and transferring the information to the .GTK file when it finds one. When all the data files in the specified drive area have been transferred, PLT2GTK will quit.

CHAPTER VI

MONITORING SYSTEM RELIABILITY ASSESSMENT

This project had several challenging moments early on that were relatively common problems associated with this and other projects of this sort. These were brought on by oversights in the initial design and planning, inadequacies in construction, shortage of time and, to little man power. However, the major problems in the first winter season were associated with bad weather which resulted in too few full working days on the break-water. It should be kept in mind that most of the installation, maintenance and repair was done from a small 13 foot fiberglass boat in relatively open and hostile waters. The boat was very adequate and anything much over 15 to 16 feet would have proved to be too large and more difficult to work from and with. The boat was, however, a relatively inexpensive unit which resulted in some repair and maintenance problems.

The most important piece of equipment was the data acquisition system. The major problems here were associated with the power system, which used a bank of four lead acid truck batteries. Inadequate charging time and some early problems in the DAS which caused an excessive power drain, were the main source. Other failures were the loss of a DC to DC converter and several of the 8 bit RCA A to D boards, which were never completely understood.

The next piece of hardware was the signal conditioning and transducer input box. This unit used approximately forty dual channel analog input signal conditioning cards which were designed and manufactured for this application. They were low power units and were a universal design which could be altered to accept all of the different inputs. In general, these cards functioned well with no more than expected repairs, given the environment they were used in. The only feature that was left out, mainly because of space and power was a built in calibration system. This would have been well worth the effort had it been included. Also, an incremental gain adjustment, in place of an adjustable pot, would have helped.

One common and relatively important problem with both of these units, was that plastic moisture proof connectors were used for all signal inputs and system interconnects, this involved hundreds of connector pins. These connectors were chosen because of the extremely corrosive environment and because of lower cost. They turned out to be inadequate and were broken easily and did not make as positive a connection as was desired. Also, the connectors that were used, made it more difficult than necessary to install new transducer leads.

The next major installation and maintenance problem was associated with the lead wires. It was decided at the beginning to use a relatively inexpensive neoprene coated braided shielded four conductor wire. It was felt that due to the large number of cables, that once they were bundled together, they would act as a single unit with more than sufficient strength. Due to the extreme difficulty and expense of getting the required diving, it was impossible to ever get the cables installed as well as intended. However, they performed as well as could be expected, and it would be hard to say if any other approach would have worked any better.

The tide gage was exceptionally successful, with the only problem being that of maintaining the electrical lead, which lasted over a year the first time around. Two major problems were encountered with the spar buoys. The first was with the anchoring system. Until the center of drag was located and properly attached to, the buoys were pulled under water at extreme current conditions. This problem was eventually completely overcome. Also, the staff units proved to be inadequate and had to be reinforced at the threaded connection between it and the lower part of the buoy. The buoys required more than their share of maintenance, but once they were properly installed, they proved to be more than adequate for their intended use.

The anchor load cells functioned without failure except for the connectors and lead wires. The connectors were manufactured by Electro Corp. and were underwater make and break units. They proved to be inadequate for strain gage level signals without careful cleaning and installation procedures when connected underwater. A solution to this problem would be to use a better connector or place the amplifier circuitry in the load cell. This would reduce the reliability, however.

All other transducer problems were associated with the mechanical connections and installation procedures and could and were only improved on by improving these variables.

CHAPTER VII

CONCLUSIONS & RECOMMENDATIONS

In general, this project went well, given the time and field conditions. The main areas that were not completed or maintained as well as expected, due mainly to time, manpower constraints and equipment failures were:

1. The regular calibrations of both the transducers and electronics.
2. Data analysis.
3. Telephone link to the breakwater.
4. Early equipment shelter.
5. Electrical leads.
6. Miscellaneous electronic failures.
7. Mechanical integrity of transducer attachments.

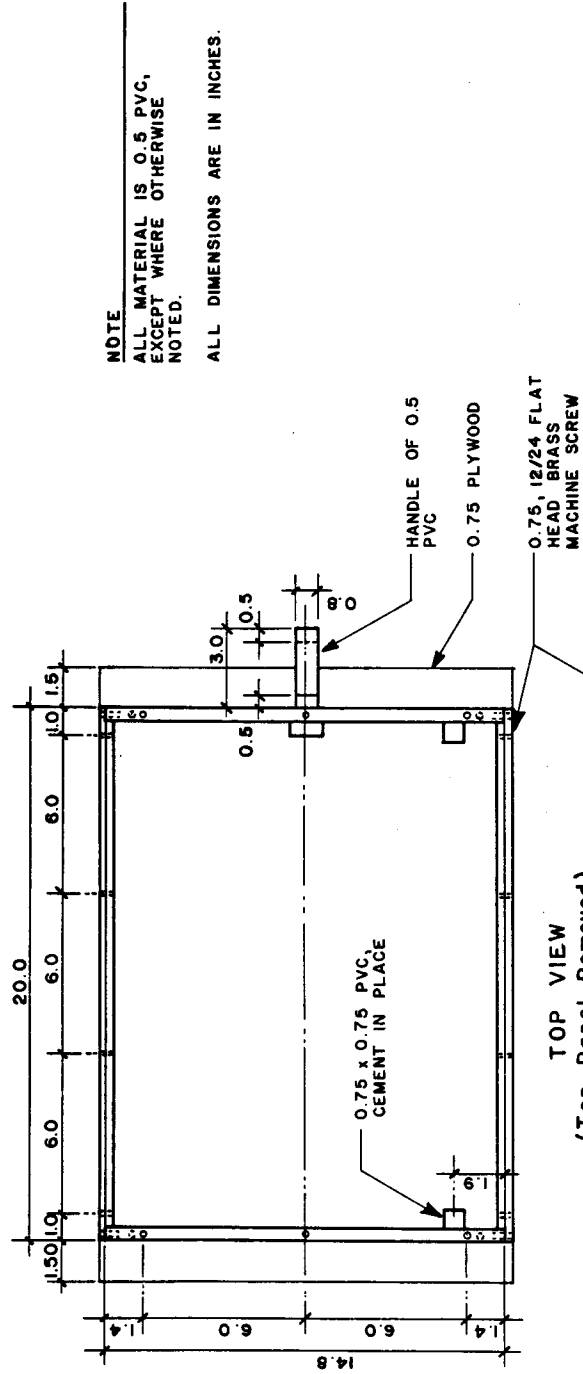
Almost all of the problems that were encountered, were solved and eventually repaired or retrofitted and made to work. The only problems left unsolved, could have been taken care of given more time and manpower. Some initial designs, both in mechanical and electrical components were poor and had to be redone. However, the main source of the problems that were encountered were attributable to extreme weather conditions the first season and to an under estimate in the original proposal as to the amount of manpower that was needed.

This is not to say that the initial design and planning stages were not comprehensive and complete, on the contrary, they were extensive and from the monitoring contractors point of view, there was little that could have been done to improve them, other than allow more time or relying more on off the shelf hardware. Current state-of-the-art equipment would also improve on the success of future projects of this type.

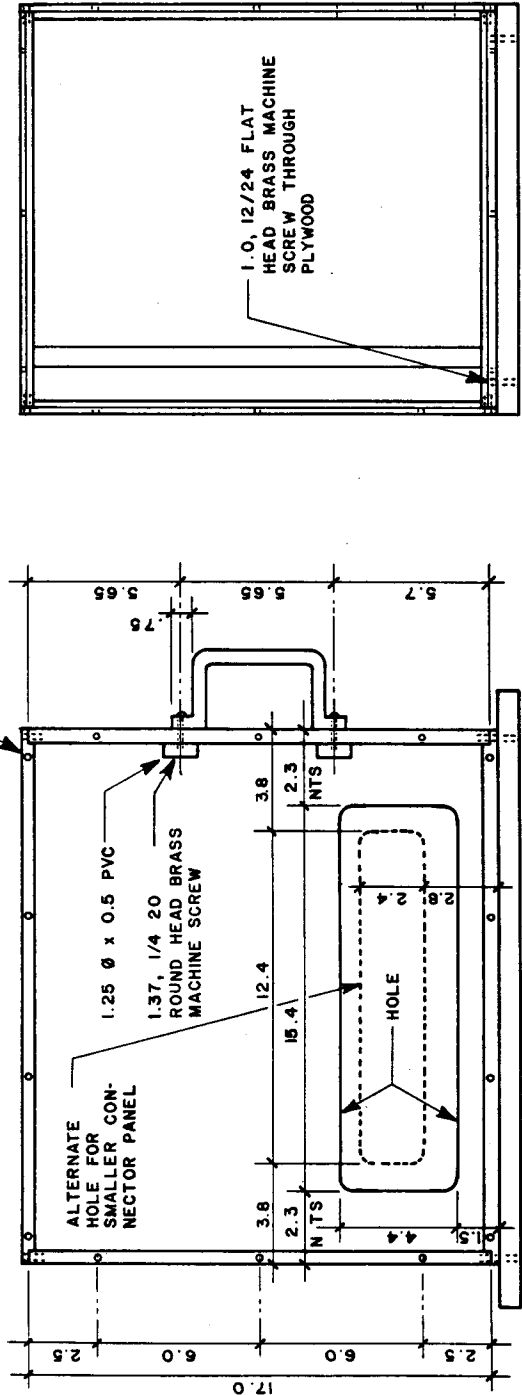
APPENDIX I

RECORDING AND ELECTRONICS SYSTEM DRAWINGS

APPENDIX IA MECHANICAL



TOP VIEW
(Top Panel Removed)

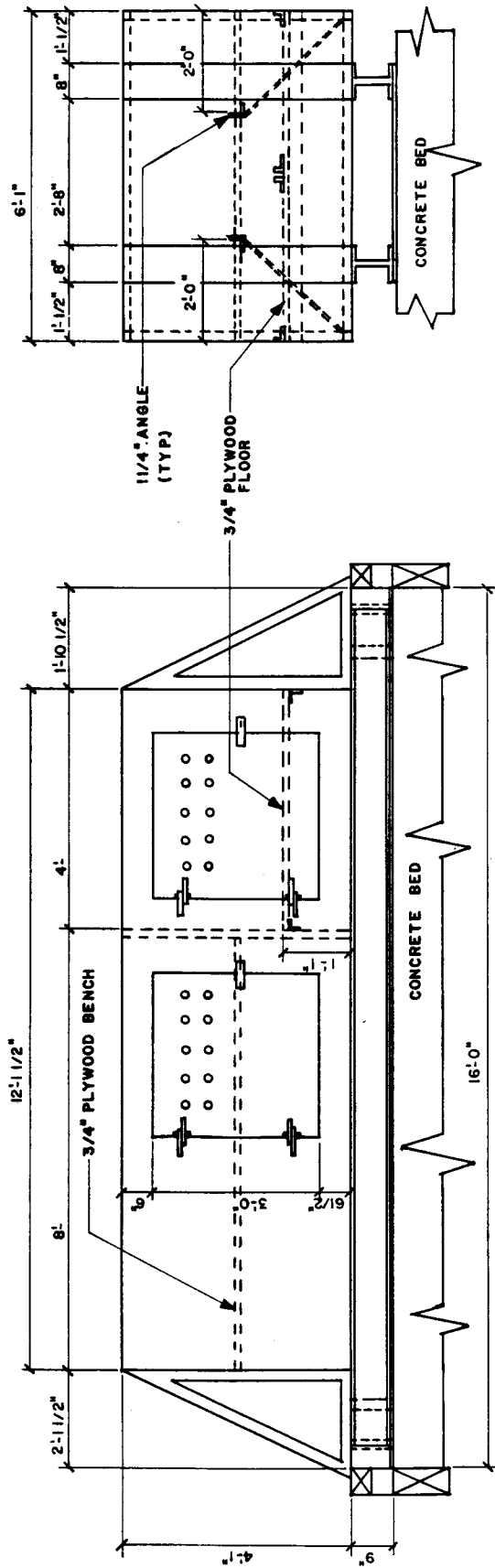


FRONT VIEW
(Front Panel Removed)

NOTE
ALL MATERIAL IS 0.5 PVC, EXCEPT WHERE OTHERWISE NOTED.
ALL DIMENSIONS ARE IN INCHES.

SIDE VIEW
(Side Panel Removed)

Figure I-1a. Field Instrument Housing Detail



NOTE: 5/16" SHEET METAL PLATE
SURROUNDS 1 1/2" METAL
TUBE FRAME STRUCTURE

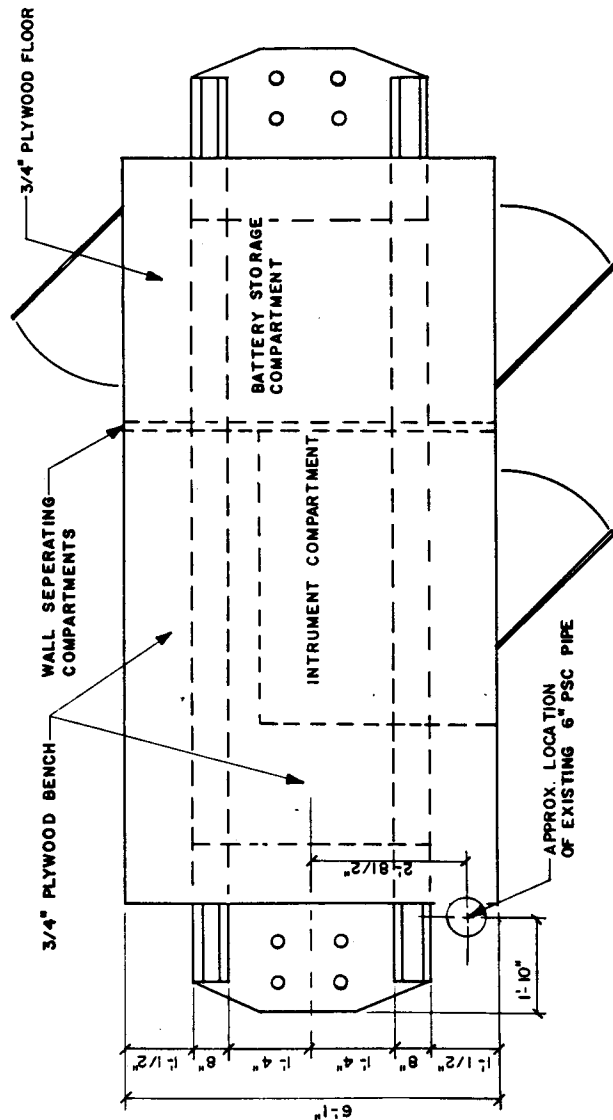
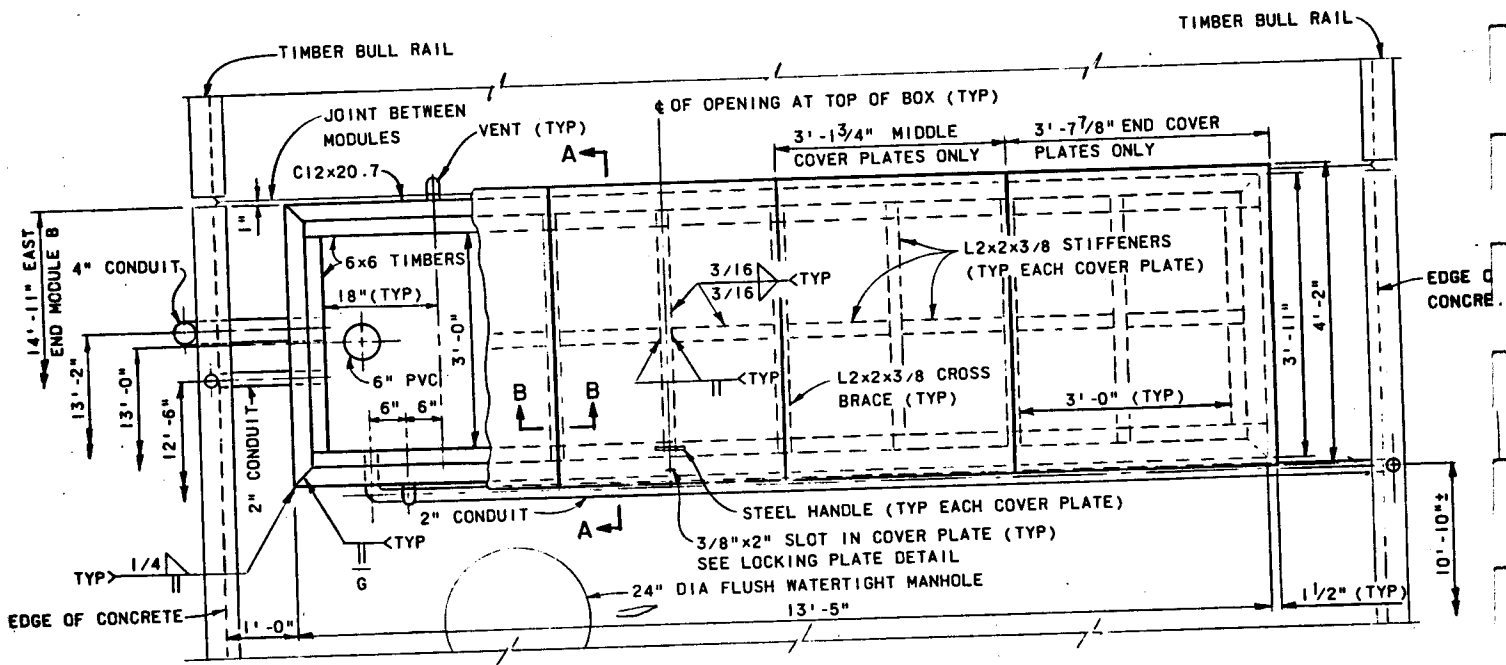
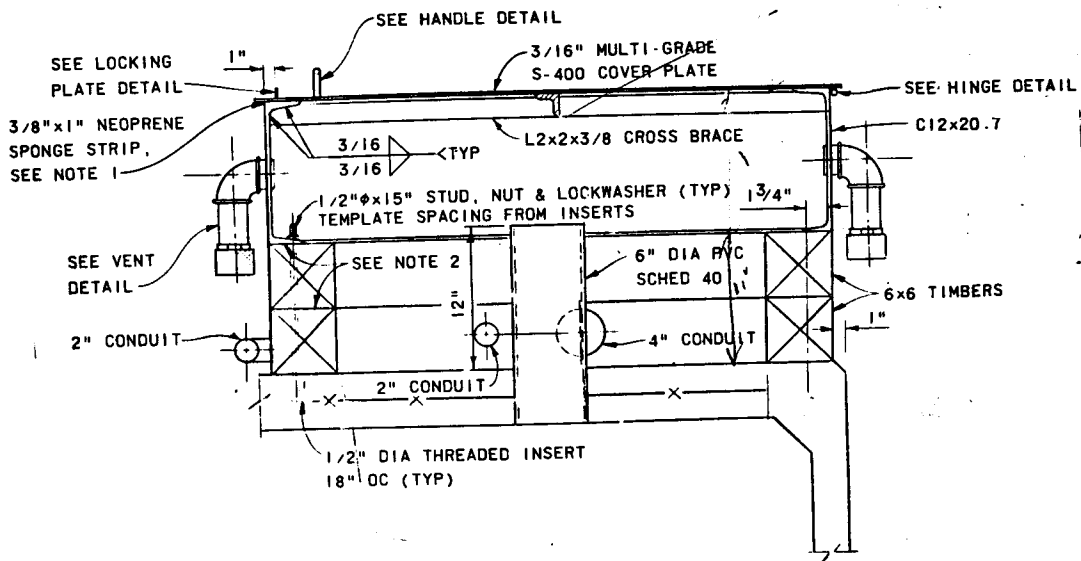


Figure I-2 Deck House Detail



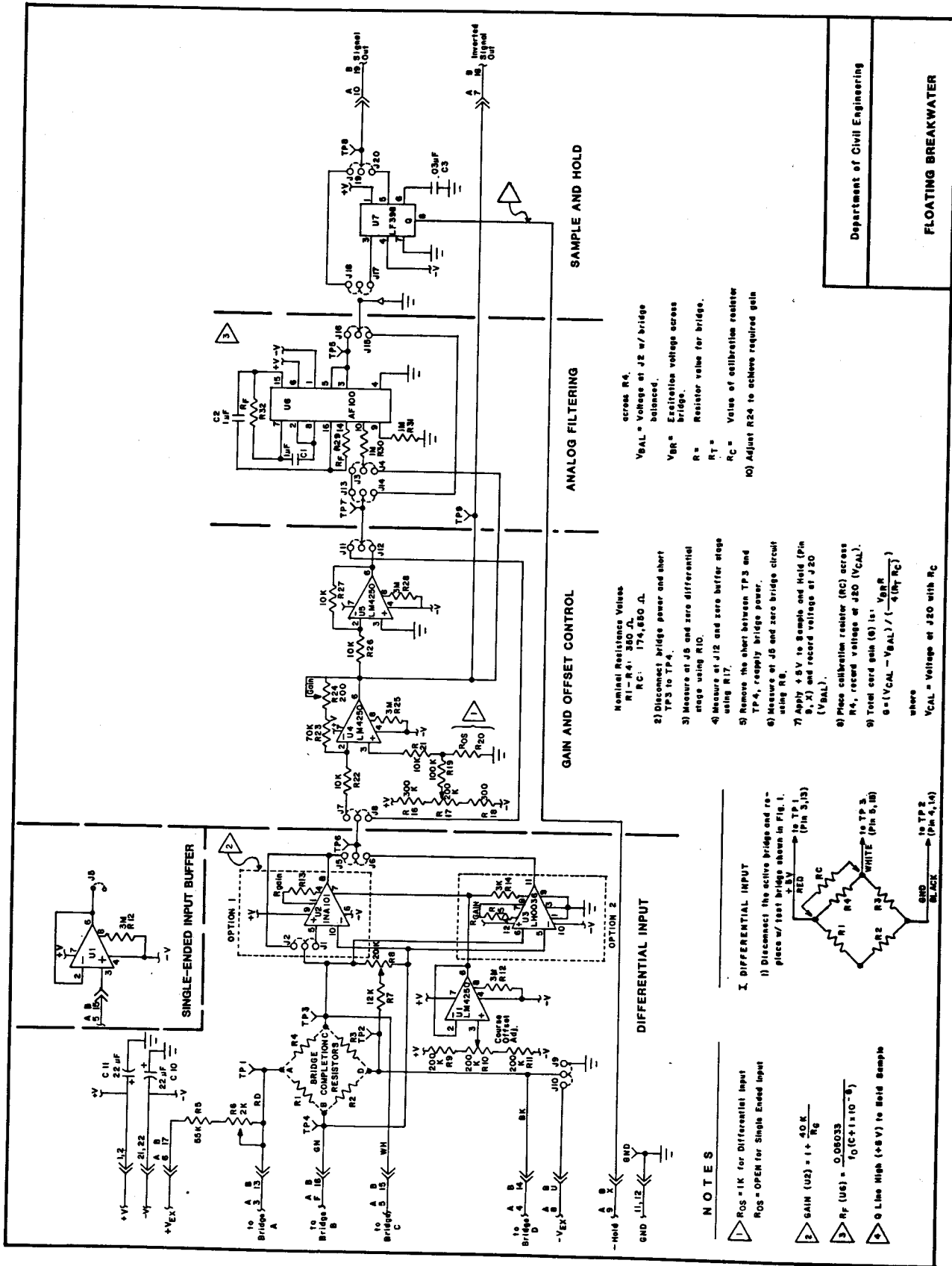
PLAN



SECTION A - A

Figure I-3 Original Deck Enclosure

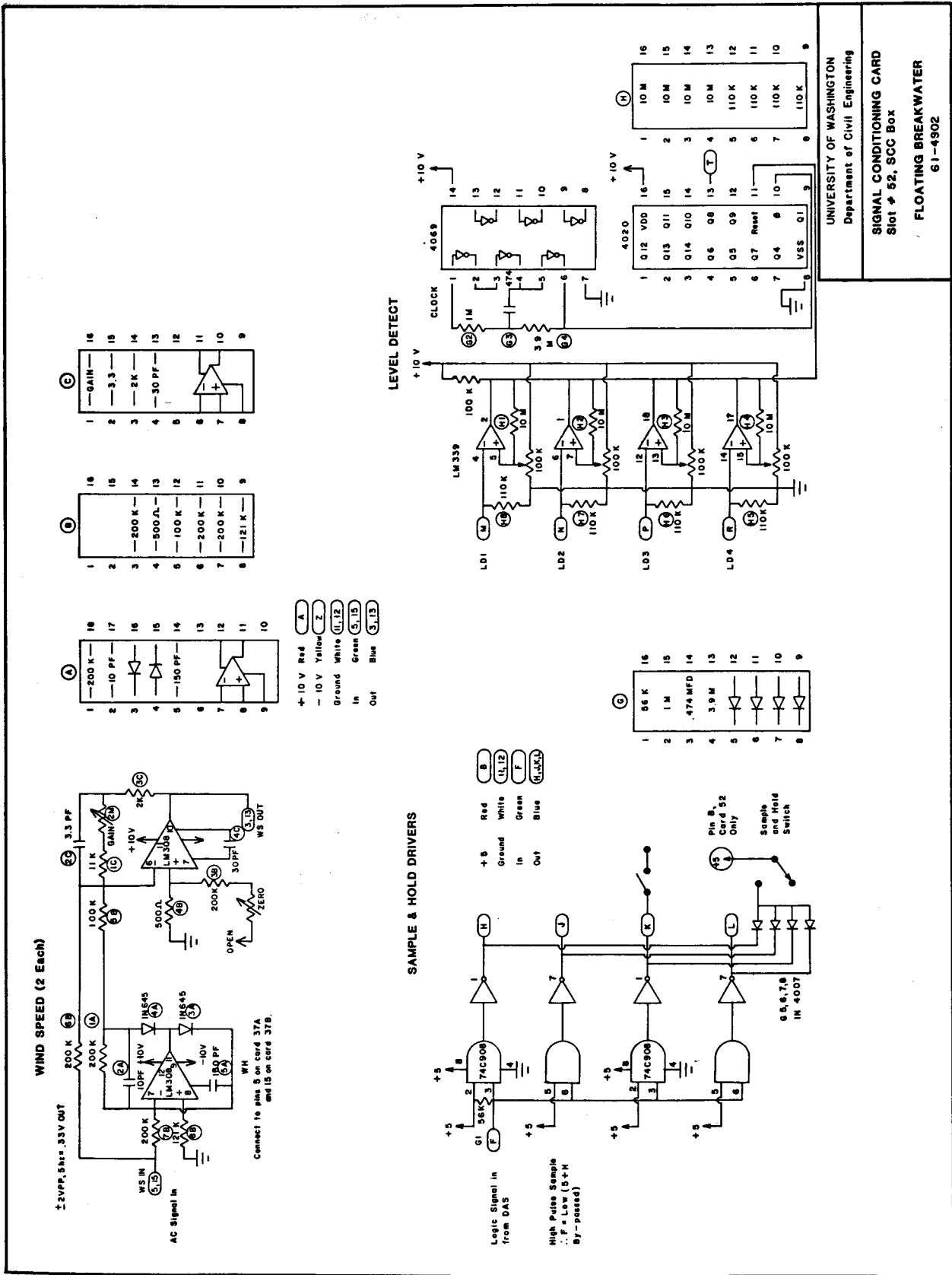
APPENDIX IB ELECTRONIC



Department of Civil Engineering

FLOATING BREAKWATER

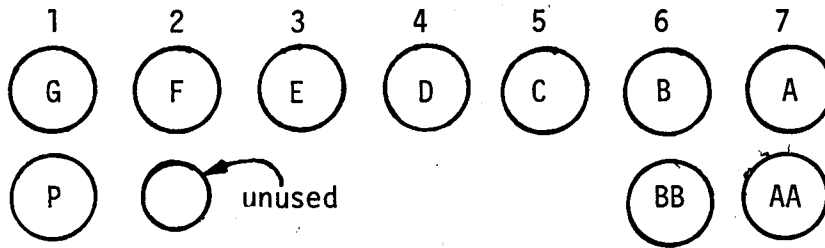
Figure I-4 Signal Conditioning Schematic



UNIVERSITY OF WASHINGTON
 Department of Civil Engineering
SIGNAL CONDITIONING CARD
 Slot # 52, SCC Box
FLOATING BREAKWATER
 61-4902

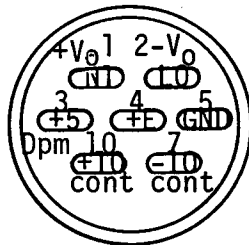
Figure I-5 Signal Conditioning Logic Schematic

On back: SCC box (viewing from back)



Note: To get continuous power to any card jump pins 22 & Z plus pins 1 & A.

Figure 8. Power Connector.



Signal Outputs

Connector AA

Channels 1-60
Channel # is the same as pin # on Connector AA

Connector BB

Channels 61-100
(Channel # 40) gives pin # on connector BB

Table 1. Channel Pin-outs
(n = channel #).

<u>Connector A</u>	<u>Connector B</u>	<u>Connector C</u>	<u>Connector D</u>
Channels 1-15 (Boards 1-8a) (n-1) x 4 + Br #	Channels 16-30 (Boards 8b-15) (n-16) x 4 + Br #	Channels 31-45 (Boards 16-23a) (n-31) x 4 + Br #	Channels 46-60 (Boards 23b-30) (n-46) x 4 + Br #
<u>Connector E</u>	<u>Connector F</u>	<u>Connector G</u>	
Channels 61-75 (Boards 31-37a) (n-61) x 4 + Br #	Channels 76-90 (Boards 37b-45) (n-76) x 4 + Br #	Channels 91-100 (Boards 46-50) (n-91) x 4 + Br #	

Figure 8a. Inconnection and signal input connector detail

Table 2.

To convert board # to Channel #: multiply board # by 2, subtract 1 for "a" suffix.

(e.g. Board # is 29a - channel # is $2 \times 29 - 1 = 57$)

To convert channel # to board #:

If even - divide channel # by two, add suffix "b"
If odd - add one to channel #, then divide by two, add suffix "a"

(e.g. Channel # is 77 - Board # is $(77+1)/2 = 39a$)

To find the location of a pin for channel "h", find connector and pin # (see Table 1)

For: Br (B) - add 1 GN
Br (C) - add 2 WH
Br (D) - add 3 BK
Br (A) - add 4 Rd

(e.g. Channel # is 69, Br (D) - using Table 1: Connector is E, pin is $(69-61) \times 4 + 3 = 35$)

Figure 8b. Inconnection and signal input connector detail

<u>SCC BOX</u>		<u>CONNECTOR</u>	A	B	C	D	E	F	G
CONNECTOR	CHANNELS	PIN NUMBER	CHANNEL NUMBERS						
A	1-15	1-4	1	16	31	46	61	76	91
B	16-30	5-8	2	17	32	47	62	77	92
C	31-45	9-12	3	18	33	48	63	78	93
D	46-60	13-16	4	19	34	49	64	79	94
E	61-75	17-20	5	20	35	50	65	80	95
F	76-90	21-24	6	21	36	51	66	81	96
G	91-105	25-28	7	22	37	52	67	82	
INPUT COLOR AND PIN ARRANGEMENT		29-32	8	23	38	53	68	83	
1	F,16	33-36	9	24	39	54	69	84	
2	5,15	37-40	10	25	40	55	70	85	
3	4,14	41-44	11	26	41	56	71	86	
4	3,13	45-48	12	27	42	57	72	87	
A TO D CARDS		49-52	13	28	43	58	73	88	
GREEN NUMBER	CHANNEL NUMBER	53-56	14	29	44	59	74	89	
30	1-16	57-60	15	30	45	60	75	90	
40	17-32								
50	33-48								
60	49-64								
70	65-80								
80	81-96								

Figure 8c. Connector Pin and Channel Arrangement Signal Conditioning Box

TRANSDUCER SPECIFICATION

Measurement	Design Range	Transducer Range	F.S. Accuracy	8-bit Resolution	12-bit Resolution	Excitation Voltage	Full Scale Output Volt
1. Wind speed direction	100 mph 360°	-	±3%	0.39 1.40°	-	0 5	0-2.5 V 2.5 ac
2. Temperature water air	40-60°F -10-100°F	-55-150°C	±0.3°C	0.08 0.43	-	5 5	1mV/°C 1mV/°C
3. Waves (1) tide (5) spar	30 ft. 8 ft.	-	±0.5% ±0.5%	1.41" 0.38"	0.09" 0.02"	±15 ±15	0-2.5 0-2.5
4. Anchor forces (4) tire (8) concrete	32.5 kip 175 kip	10 kip 50 kip	0.1% 0.1%	78 390	4.9 24	+5 +5	±10mV ±10mV
5. Dyn pressure (23)	0-5 psi	-	0.8%	.02	.001	+5	20mV
6. Concrete Str. (8) long. (4) trans.	500 μs 50 μs	-	.1% .1%	1.95 μs 0.2 μs	-	+5 +5	7mV 0.7mV
7. Accelerometers (4) linear (2) angular	±1 g ±.5 g	±1 g ±5 rad/sec ²	.06% .06%	.008 .04	.0005 .002	±10 ±10	±2.5 ±2.5
8. Connector force strain bolts 14" (4) 120" (4)	0-40 kip 0-185 kip	52 kip 50k-150k (range = 100 k)	1.0% 1.0%	157 lbs 392 lbs	9.8 lbs 2.4 lbs	+5 +5	26mV 31mV
9. Fixed reference (1) linear (2) rotation	0-20 ft. 0-20.83 ft.	-	0.1%	0.94"	0.06"	5.0	0-2.5V
10. Relative motion (1) linear (5) rotation	±4" 24k ±15° 25k	-	limited by pwr supply	0.03" 0.12°	-	5.0 5.0	±1V
11. Current Sp (1)	0-±10 fps	-	±2%	0.023 km	-	±10	±1V

APPENDIX IC SYSTEM OPERATION PROCEDURES

I. SYSTEM OPERATION PROCEDURE

Always remove old cartridge, mark date and time and return to U.W

With out monitor card

1. Insert new cartridge, mark date and time on back.
2. Place all six front panel toggle switches in the down position.
3. Turn power switch on, black rocker switch.
4. Tape should rewind immediately, followed by brief pause and second rewind and an advance to beginning of tape.
5. Turn the tape drive disable switch to the up position. This switch is mounted just below the tape drive.

With monitor card and hand terminal

1. Same procedure as above except:
 - (a) Hold top red reset button on terminal in while turning on the power.
 - (b) Press the RP (green) button to start the program. This will initiate the second tape rewind.

II. SYSTEM CALIBRATION AND CHECK OUT PROCEDURE (PRIORITIES)

1. Check through entire channel sequence and fill out summary sheet. This should be done with the system actually sampling and writing to a blank tape.
2. Repair any known or planned events.
3. Repair, adjust or reset any problems encountered in the initial check out and update check out sheet or start a new one.

III. MAINTENANCE - PRIORITY SCHEDULE

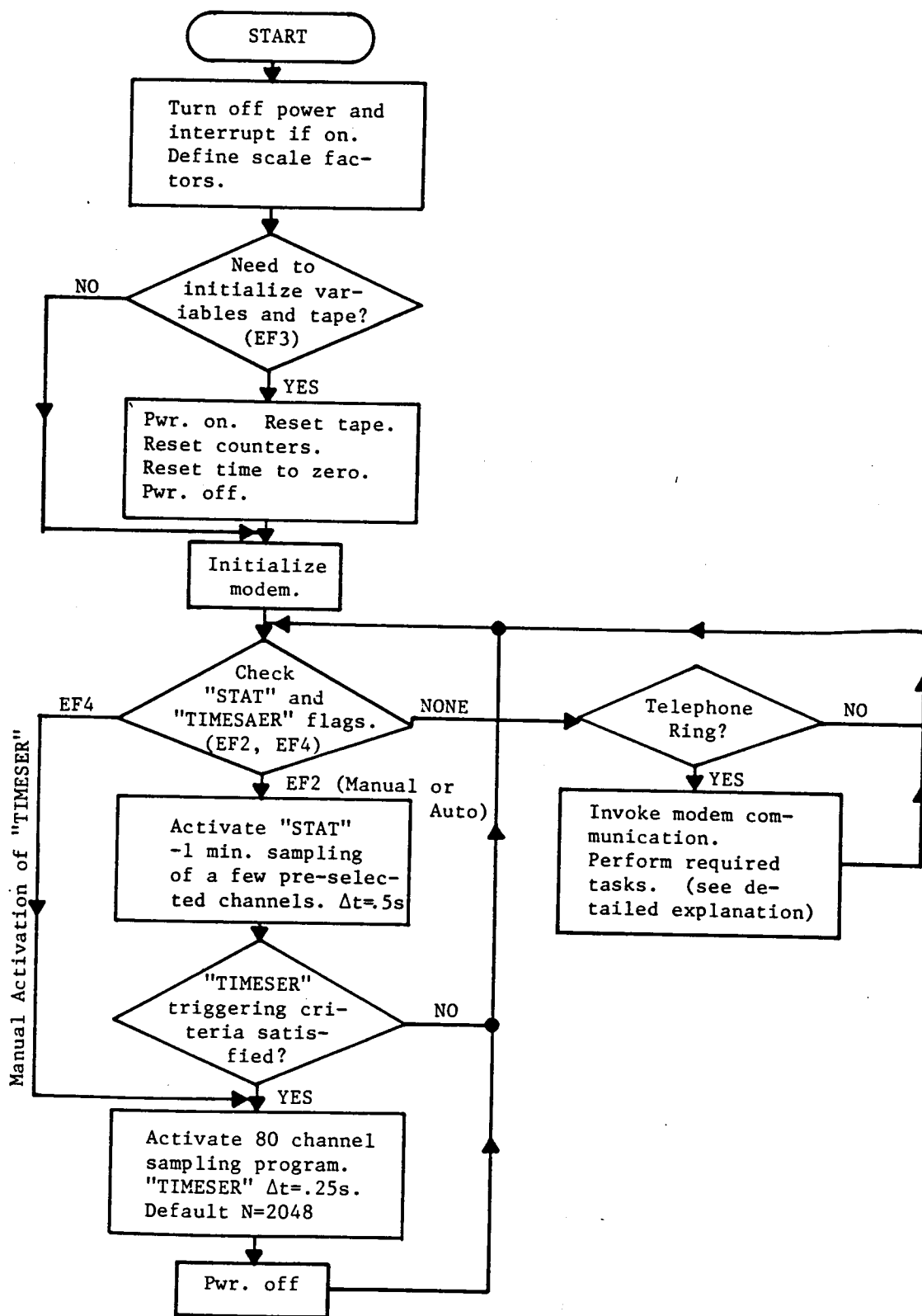
1. Tide gage Ch. 16
2. Wind speed #1 Ch. 73
3. Wave buoys Ch's 18,19,20,21 (in this order)
4. Load cells (South upper & lower same cable - tire)
(South upper & lower same cable - concrete)
(All others)
5. Other wind sensors Ch./ 74,75 & 76
6. Side pressures
7. Concrete strain
8. Accelerometers (at least one set, 61-63 or 64-66)
9. Current meter
10. Motions Ch's 13-15
11. Temperatures
12. Internal voltages

NEW TAPE INITIALIZATION

1. Switch tape to "Auto Rewind" mode and switch on power relay bypass if necessary.
2. Switch system power on. The tape will come to BOT. Now switch off power relay bypass.
3. Press RP to run program.
4. When tape stops at the "Load Point" switch the tape back to "No Auto Rewind" mode.

APPENDIX II
SOFTWARE LISTING

APPENDIX IIA RCA FIELD DATA ACQUISITION SYSTEM



Breakwater Data Acquisition Software
Effective 3/11/83

```

10 ENINT 1320
20 DIM S(96),F(5)
30 N1=0:N2=0
40 POKE(@908B,#02): POKE(@908F,#02): POKE(@90A9,#00)
50 FOR I=1 TO 96:S(I)=1: NEXT
60 G1=@6834:G2=@68EE:G3=@6A45:G4=@6800:G5=@6BB5:G6=@6BBA:G7=@6B82:G8=@6B98
70 G9=@6BBF:H1=@6BC2:H2=@6B00:H3=@6BCD:H4=@6BE2:R=@A000:RO=R+@0100:C=@7000
80 R2=@B000:H5=@6C00:H6=@6C59:AS=" ":BS=" ": FOR I=1 TO 5:F$(I)="OFF": NEXT
90 EB=0:F$(3)="ON":F$(4)="ON"
100 D$="Reg. not accessible. Input ignored":E$="CERCASS"
110 GOSUB 1130
120 X=EF(3): IF X=1 GOTO 210
130 POKE(R+6,0): POKE(R+10,#08): POKE(R+11,#00):K1=0:K2=0
140 FOR I=(R+#10) TO (R+#13): POKE(I,0): NEXT
150 GOSUB 1090: WAIT(750)
160 OUT (4,3,29):X=INP(4,2): CALL (G6): OUT (4,2,208)
170 CALL (G6): OUT (4,2,0): CALL (G6): OUT (4,2,#43): CALL (H1)
180 OUT (4,2,7): CALL (H1): GOSUB 1130
190 FOR I=0 TO 15: POKE(C+I,0): NEXT
200 POKE(C+13,1): POKE(C+14,1)
210 OUT (#F0,3,#19): OUT (#F0,4,#FF): OUT (#F0,6,0): OUT (#F0,7,#16)
220 POKE(R+7,1): POKE(R+8,1): POKE(R+9,1)
230 CALL (G4)
240 X2=PEEK(R+7):X3=PEEK(R+8):X4=PEEK(R+9)
250 IF X2=0 GOSUB 290
260 IF X3=0 GOSUB 1090: WAIT(750): GOSUB 1170:F=0: GOSUB 930
270 IF X4=0 GOTO 1350
280 GOTO 220
290 K1=K1+1:Q=0:A=R2:R1=RO
300 X=INT(K1/256): POKE(R+#10,X): POKE(R1,X):X=K1-X*256: POKE(R+#11,X): POKE(R1+
1,X):R1=R1+2
310 POKE(R1,0): POKE(R1+1,0):R1=R1+2
320 POKE(R1,PEEK(R+6)):R1=R1+1
330 CALL (G7,RO+5):R1=R1+10: GOSUB 1090: CALL (G1)
340 GOSUB 1130:N1=PEEK(R+1)*256+PEEK(R+2)
350 POKE(R1,PEEK(R+1)): POKE(R1+1,PEEK(R+2)):R1=R1+2
360 FOR I=1 TO 3
370 B=A:M=0:S1=0:M1=0:M2=4095
380 FOR J=1 TO N1:X=(PEEK(B)*256+PEEK(B+1))/16
390 M=M+X:S1=S1+X*X
400 IF M1<X THEN M1=X
410 IF M2>X THEN M2=X
420 B=B+9: NEXT J
430 A=A+2:M=M/N1:S1=(S1-N1*M*M)/(N1-1)
440 IF S1<0 THEN S1=0
450 S1=SQR(S1)
460 X=INT(M):B=INT(M/256):X=X-B*256
470 POKE(R1,B):R1=R1+1: POKE(R1,X):R1=R1+1
480 X=(M-INT(M))*100: POKE(R1,X):R1=R1+1
490 X=INT(S1):B=INT(S1/256):X=X-B*256
500 POKE(R1,B):R1=R1+1: POKE(R1,X):R1=R1+1
510 X=(S1-INT(S1))*100: POKE(R1,X):R1=R1+1
520 X=INT(M1/256):B=M1-X*256
530 POKE(R1,X):R1=R1+1: POKE(R1,B):R1=R1+1
540 X=INT(M2/256):B=M2-X*256
550 POKE(R1,X):R1=R1+1: POKE(R1,B):R1=R1+1
560 IF I=1 IF M<4092 IF M>3500 IF F$(1)="ON" THEN Q=1
570 IF I=2 IF M<4092 IF M>3500 IF F$(2)="ON" THEN Q=1

```



```

580 IF I=3 IF M<4092 IF 4*S1>164 IF F$(3)="ON" THEN Q=1
590 NEXT I
600 FOR I=1 TO 3
610 B=A:M=0:S1=0:M1=0:M2=255
620 FOR J=1 TO N1:X=PEEK(B)
630 M=M+X:S1=S1+X*X
640 IF M1<X THEN M1=X
650 IF M2>X THEN M2=X
660 B=B+9: NEXT J
670 A=A+1:M=M/N1:S1=(S1-N1*M*M)/(N1-1)
680 IF S1<0 THEN S1=0
690 S1=SQR(S1)
700 POKE(R1,M):R1=R1+1
710 X=(M-INT(M))*100: POKE(R1,X):R1=R1+1
720 POKE(R1,S1):R1=R1+1
730 X=(S1-INT(S1))*100: POKE(R1,X):R1=R1+1
740 POKE(R1,M1):R1=R1+1: POKE(R1,M2):R1=R1+1
750 IF I=2 IF M<254 IF M>25 IF F$(4)="ON" THEN Q=1
760 NEXT I
770 GOSUB 1090
780 I=#0C: GOSUB 1340:B=X:I=#0D: GOSUB 1340
790 IF B<255 IF X<255 IF SQR(B*B+X*X)>102 IF F$(5)="ON" THEN Q=1
800 I=#09: GOSUB 1340:I=#0B: GOSUB 1340
810 FOR X=2 TO 11: OUT (#80,5,X): CALL (G9)
820 B=INP(#80,3): POKE(R1,B):R1=R1+1: NEXT
830 WAIT(500): GOSUB 1170
840 WAIT(1): CALL (G6): OUT (4,2,2): FOR I=0 TO 6:X=PEEK(@682D+I)
850 CALL (G6): OUT (4,2,X):N2=N2+1: NEXT
860 FOR I=R0 TO R0+#0E: CALL (G6): OUT (4,2,PEEK(I)):N2=N2+1: NEXT : GOSUB 1280
870 FOR I=R0+#0F TO R1-1: CALL (G6): OUT (4,2,PEEK(I)):N2=N2+1: NEXT
880 X=512-N2: CALL (H3,X):N2=0: CALL (H2)
890 IF Q=1 GOTO 910
900 GOTO 1070
910 T=PEEK(R+6):F=1: CALL (G6): OUT (4,2,T+208)
920 CALL (G6): OUT (4,2,0)
930 K2=K2+1: WAIT(1): CALL (G6): OUT (4,2,2)
940 FOR I=0 TO 6:X=PEEK(@68E7+I)
950 CALL (G6): OUT (4,2,X):N2=N2+1: NEXT
960 IF F=1 THEN X=PEEK(R+#10):B=PEEK(R+#11)
970 IF F=0 THEN X=0:B=0
980 CALL (G6): OUT (4,2,X): CALL (G6): OUT (4,2,B):N2=N2+2
990 X=INT(K2/256): POKE(R+#12,X): CALL (G6): OUT (4,2,X):N2=N2+1
1000 X=K2-256*X: POKE(R+#13,X): CALL (G6): OUT (4,2,X):N2=N2+1
1010 CALL (G6): OUT (4,2,PEEK(R+6)):N2=N2+1
1020 CALL (G8):N2=N2+10: GOSUB 1280
1030 X=PEEK(R+10): CALL (G6): OUT (4,2,X)
1040 X=PEEK(R+11): CALL (G6): OUT (4,2,X):N2=N2+2
1050 X=512-N2: CALL (H3,X):N2=0: CALL (H2)
1060 POKE(R+3,#02): POKE(R+4,#00): CALL (G2):N2=0
1070 GOSUB 1130
1080 RETURN
1090 STQ 0: OUT (128,5,3): CALL (G9)
1100 Z=INP(128,3): IF Z>204 GOTO 1120
1110 OUT (16,6,255): OUT (16,6,254)
1120 RETURN
1130 STQ 0: OUT (128,5,3): CALL (G9)
1140 Z=INP(128,3): IF Z<75 GOTO 1160
1150 OUT (16,6,255): OUT (16,6,254)
1160 RETURN
1170 B=0

```

```

1180 T=PEEK(R+6): OUT (4,3,29):X=INP(4,2)
1190 CALL (G6): OUT (4,2,T+208)
1200 CALL (G6): OUT (4,2,0): CALL (G6)
1210 OUT (4,2,8): CALL (G5):X=INP(4,2)
1220 CALL (G5):X=INP(4,2)
1230 IF X<128 IF B<4 GOTO 1260
1240 IF X>131 IF B<4 GOTO 1260
1250 RETURN
1260 OUT (#10,6,#FF): OUT (#10,6,#FE): WAIT(500): OUT (#10,6,#FF): OUT (#10,6,#F
E)
1270 WAIT(1000):B=B+1: GOTO 1180
1280 FOR I=1 TO 96:X=INT(S(I))
1290 CALL (G6): OUT (4,2,X)
1300 X=(S(I)-X)*100: CALL (G6): OUT (4,2,X)
1310 N2=N2+2: NEXT : RETURN
1320 B=C+15: POKE(B,0):X=PEEK(B):X=PEEK(B):X=PEEK(B)
1330 OUT (#10,6,#FF): OUT (#10,6,#FD): RETURN
1340 OUT (#70,5,I): CALL (G9):X=INP(#70,3): POKE(R1,X):R1=R1+1: RETURN
1350 OUT (#F0,4,#7F): OUT (#F0,7,#95)
1360 FOR I=1 TO 250: IF INP(#F0,5) AND 2=2 EXIT 1380
1370 NEXT : OUT (#F0,7,0): GOTO 210
1380 WAIT(1000): PRINT "Floating Breakwater"
1390 PRINT "-----": PRINT AS: PRINT
1400 CALL (G7,R2):R1=R2: GOSUB 2880
1410 T=PEEK(R+6): PRINT "Current Tape Track = ";T+1
1420 X=256*PEEK(R+#10)+PEEK(R+#11): PRINT "Last 1 Min. Rec. # = ";X
1430 X=256*PEEK(R+#12)+PEEK(R+#13): PRINT "Latest Timeseries Rec. # = ";X
1440 N=256*PEEK(R+#0A)+PEEK(R+#0B): PRINT "# of Time Series Samples = ";N
1450 PRINT "Sampling interval: 1 Min rec. - .5 s"
1460 PRINT TAB(19);"Time Series - .25 s": PRINT
1470 OUT (0,1,#F0):R1=1500
1480 FOR I=1 TO 250: IF INP(#F0,5) AND 2=2 EXIT R1: NEXT
1490 OUT (#F0,7,0): GOTO 210
1500 PRINT : INPUT "Enter menu # (0 to display menu, 14 to QUIT)"CS
1505 IF CS<>"0" GOTO 1600
1510 PRINT : PRINT "Enter 1.List last 1 min data recorded "
1520 PRINT TAB(7);"2.Run new 1 min. record and list": PRINT TAB(7);"3.Single cha
nnel scan"
1530 PRINT TAB(7);"4.Examine/Change Time series triggers": PRINT TAB(7);"5.Alter
N, time"
1540 PRINT TAB(7);"6.Enter comment": PRINT TAB(7);"7.Leave message"
1550 PRINT TAB(7);"8.Read message": PRINT TAB(7);"9.Examine memory"
1560 PRINT TAB(6);"10.Write on RAM": PRINT TAB(6);"11.Change scale fac"
1570 PRINT TAB(6);"12.Run Timeseries": PRINT TAB(6);"13.Realtime data BYTES ";
1580 PRINT "(not usable without a computer) ": PRINT TAB(6);"14.Quit"
1600 IF CS="1" GOTO 1760
1610 IF CS="2" GOSUB 290:R1=1760: OUT (0,1,#F0): GOTO 1480
1620 IF CS="3" F1=0: GOTO 1890
1630 IF CS="4" GOTO 2290
1640 IF CS="5" GOTO 2360
1650 IF CS="6" INPUT "Enter Comment (128 char. max.) "AS: GOTO 1470
1660 IF CS="7" INPUT "Enter Message (128 char.max)"BS: GOTO 1470
1670 IF CS="8" PRINT : PRINT BS: GOTO 1470
1680 IF CS="9" GOTO 2530
1690 IF CS="10" GOTO 2630
1700 IF CS="11" GOTO 2670
1710 IF CS="12" GOSUB 1090: WAIT(750): GOSUB 1170:F=0: GOSUB 930: GOTO 1470
1720 IF CS="13" F1=1: GOTO 1890
1730 IF CS="14" OUT (#F0,7,0): GOTO 210
1740 IF CS="911" PRINT "Old password is ";ES: INPUT "Enter new pw "ES: GOTO 1470

```

```

1750 GOTO 1470
1760 PRINT : PRINT "Latest 1 Min Statistical Data Summary": PRINT
1770 X=10*PEEK(R0+6)+PEEK(R0+7):B=10*PEEK(R0+9)+PEEK(R0+0A)
1780 PRINT X;"/";B;"/83      ";X=10*PEEK(R0+0B)+PEEK(R0+0C):B=10*PEEK(R0+0D)+
+PEEK(R0+0E)
1790 PRINT X;" Hrs      ";B;" Min": PRINT "# of samples = ";(256*PEEK(R0+0F)+PEEK(
R0+10))
1800 PRINT "Tape Track = ";PEEK(R0+04)+1: PRINT "Rec. Number = ";256*PEEK(R0)+P
EEK(R0+1)
1810 PRINT : PRINT "Quantity (Ch.#)          Mean          Std.Dev.          Max
Min": PRINT
1820 CS="Anch.Force(kip) (7)":R1=R0+11:S=S(7): GOSUB 2730
1830 CS="Anch.Force(kip) (8)":R1=R1+10:S=S(8): GOSUB 2730
1840 CS="Tide Ht.(ft) (16)":R1=R1+10:S=S(16): GOSUB 2730
1850 CS="Vert.Acc(ft/s**2) (61)":R1=R1+10:S=S(61): GOSUB 2780
1860 CS="Wind Sp.(mph) (73)":R1=R1+6:S=S(73): GOSUB 2780
1870 CS="Wind Dir.(deg) (75)":R1=R1+6:S=S(75): GOSUB 2780
1880 GOSUB 2800: GOTO 1470
1890 IF F1=1 INPUT "Enter password or <CR> to exit "CS: IF CS<>ES GOTO 1470
1900 L=0: PRINT : INPUT "Enter Chan. # "CS
1910 FOR X=1 TO 96: IF CS=STR$(X) EXIT 1930
1920 NEXT : GOTO 1980
1930 INPUT "Enter # of samples (power of 2 and < 512 ) "CS
1940 FOR I=1 TO 9:B=2^I: IF B-INT(B)>0 THEN B=INT(B)+1: GOTO 1960
1950 B=INT(B)
1960 IF CS=STR$(B) EXIT 1990
1970 NEXT
1980 PRINT "INPUT ERROR, TRY AGAIN": GOTO 1470
1990 IF B>512 GOTO 1930
2000 PRINT : PRINT "Scale factor = ";S(X);" Dt=.5 s": PRINT
2010 G=#30+#10*INT((X-1)/16): POKE(R2,G)
2020 J=X-1-16*INT((X-1)/16): IF G=#30 THEN J=J+e0100
2030 IF F1=1 GOTO 2260
2040 M=0:S1=0:M1=0:M2=10000
2050 GOSUB 1090: CALL (H5,J,B): GOSUB 1130
2060 OUT (0,1,#FO):R1=2070: GOTO 1480
2070 J=0: PRINT "Sample Values": PRINT
2080 IF X<17 GOTO 2150
2090 FOR I=0 TO B-1:Y=S(X)*PEEK(R2+I):M=M+Y:S1=S1+Y*Y
2100 IF M1<Y THEN M1=Y
2110 IF M2>Y THEN M2=Y
2120 PRINT TAB(J);Y;J=J+20: IF J>70 THEN J=0:L=L+1: PRINT
2130 IF L=20 INPUT "*****"CS:L=0
2140 NEXT : GOTO 2210
2150 FOR I=0 TO 2*B-2 STEP 2:Y=S(X)*(16*PEEK(R2+I)+PEEK(R2+I+1))/16)
2160 M=M+Y:S1=S1+Y*Y: IF M1<Y THEN M1=Y
2170 IF M2>Y THEN M2=Y
2180 PRINT TAB(J);Y;J=J+20: IF J>70 THEN J=0:L=L+1: PRINT
2190 IF L=20 INPUT "*****"CS:L=0
2200 NEXT
2210 M=M/B:S1=(S1-B*M*M)/(B-1): IF S1<0 THEN S1=0
2220 S1=SQR(S1)
2230 PRINT : PRINT "Ch ";X;" # of samples (dt=.5 s) = ";B
2240 PRINT : PRINT "Mean = ";M: PRINT "Std.Dev = ";S1: PRINT "Max = ";M1: PRINT
"Min = ";M2
2250 GOTO 1470
2260 PRINT "Beginning to send ";B;" bytes"
2270 GOSUB 1090: CALL (H6,J,B-1): GOSUB 1130
2280 GOTO 1470
2290 PRINT : PRINT "Trigger 1.Anchor Force AF7(Ch.7) ";FS(1);TAB(40);"4.Wind Sp.
(Ch.73) ";FS(4)
2300 PRINT TAB(8);"2.Anchor Force AF8(Ch.8) ";FS(2);
2310 PRINT TAB(40);"5.Curr. Vel CN1,CE1 (Chs.77 & 78) ";FS(5)
2320 PRINT TAB(8);"3.Sig. Wave Ht. WV1 (Ch.16) ";FS(3)
2330 INPUT "Enter password to change trigger status or <CR>"CS: IF CS<>ES GOTO 1
470

```

```

2340 INPUT "Enter # of triggers to be changed" X
2350 FOR I=1 TO X: INPUT "Trigger # "B": INPUT "ON or OFF "Fs(B): NEXT : GOTO 147
0
2360 INPUT "Enter password to change N or <CR> "Cs: IF Cs<>Es GOTO 2400
2370 INPUT "Enter N "Cs:E9=2370: GOSUB 2910: IF E8=1 THEN E8=0: GOTO 1470
2380 X=FVAL(Cs)
2390 B=INT(X/256):X=X-B*256: POKE(R+#0A,B): POKE(R+#0B,X)
2400 CALL (G7,R2):R1=R2: PRINT "Present time": GOSUB 2880
2410 INPUT "Enter password to change time or <CR>"Cs: IF Cs<>Es GOTO 1470
2420 POKE(C+14,0): PRINT "Clock stopped": PRINT : INPUT "Enter # of registers to
be changed "Cs
2430 E9=2420: GOSUB 2910: IF E8=1 THEN E8=0: GOTO 2520
2440 X=FVAL(Cs)
2450 FOR I=1 TO X
2460 INPUT "Enter reg. (1 - 13)"Cs:E9=2460: GOSUB 2910
2470 IF E8=1 THEN E8=0: EXIT 2520
2480 B=FVAL(Cs)
2490 IF B<1 PRINT D$: GOTO 2515
2500 IF B>13 PRINT D$: GOTO 2515
2502 INPUT "Enter value "Cs:E9=2502: GOSUB 2910: IF E8=1 THEN E8=0: EXIT 2520
2510 M=FVAL(Cs): POKE(C+B,M)
2515 NEXT
2520 POKE(C+14,1): PRINT "Clock Started": GOTO 1470
2530 INPUT "Password "Cs: IF Cs<>Es GOTO 1470
2540 INPUT "Enter # of memory locations "Cs:E9=2540: GOSUB 2910
2550 IF E8=1 THEN E8=0: GOTO 1470
2560 X=FVAL(Cs)
2570 FOR I=1 TO X
2580 INPUT "Enter mem. location"Cs
2590 E9=2580: GOSUB 2910
2600 IF E8=1 THEN E8=0: EXIT 1470
2610 B=FVAL(Cs)
2620 PRINT "Byte (decimal) = ";PEEK(B): PRINT : NEXT : GOTO 1470
2630 INPUT "Enter password"Cs: IF Cs<>Es GOTO 1470
2640 INPUT "Enter # of memory locations"X
2650 FOR I=1 TO X: INPUT "Enter mem. location AND byte"E,B
2660 POKE(E,B): NEXT : GOTO 1470
2670 INPUT "Enter password "Cs: IF Cs<>Es GOTO 1470
2680 INPUT "Enter # of channnels "Cs:E9=2680: GOSUB 2910: IF E8=1 THEN E8=0: GOT
0 1470
2685 X=FVAL(Cs)
2688 FOR I=1 TO X
2690 INPUT "Enter Chan. # "Cs:E9=2690: GOSUB 2910: IF E8=1 THEN E8=0: EXIT 1470
2695 B=FVAL(Cs)
2700 PRINT "Present s.f = ";S(B): INPUT "Need to change(Y/N)"Cs
2705 IF Cs<>"Y" PRINT "Unchanged": GOTO 2720
2710 INPUT "Enter new s.f "Cs:E9=2700: GOSUB 2910: IF E8=1 THEN E8=0: EXIT 1470
2715 Y=FVAL(Cs)
2719 S(B)=Y
2720 NEXT : GOTO 1470
2730 M1=S*(256*PEEK(R1)+PEEK(R1+1)+PEEK(R1+2)/100)
2740 M2=S*(256*PEEK(R1+3)+PEEK(R1+4)+PEEK(R1+5)/100)
2750 M3=S*(256*PEEK(R1+6)+PEEK(R1+7)):M4=S*(256*PEEK(R1+8)+PEEK(R1+9))
2760 PRINT Cs:; PRINT TAB(24);M1:;PRINT TAB(37);M2:; PRINT TAB(54);M3:; PRINT T
AB(66);M4
2770 RETURN
2780 M1=S*(PEEK(R1)+PEEK(R1+1)/100):M2=S*(PEEK(R1+2)+PEEK(R1+3)/100)
2790 M3=PEEK(R1+4):M4=PEEK(R1+5): GOTO 2760
2800 ES(1)="Curr. Vel X(ft/s) (77)":ES(2)="Curr.Vel Y(ft/s) (78)"
2810 ES(3)="Wind Sp.(mph) (74)":ES(4)="Wind dir.(deg) (76)"
2820 ES(5)="-10V (83)":ES(6)="+10V (84)":ES(7)="-24V (85)"
2830 ES(8)="+24V (86)":ES(9)="+5V (87)":ES(10)="-5V (88)":ES(11)="Water Temp (F)
(89)"
2835 ES(12)="Air Temp (F) (90)":ES(13)="SCC Temp (F) (90)"
2837 ES(14)="DAS Temp (F) (91)"
2840 PRINT : PRINT "Instantaneous Samples": PRINT

```

```
2850 J=76: FOR I=1 TO 14: PRINT ES(I): PRINT TAB(24);S(I+J)*PEEK(RO+#40+I): IF  
I=2J=71  
2860 IF I=4J=78  
2865 IF I=12 INPUT "*****"CS  
2870 NEXT : RETURN  
2880 B=10*PEEK(R1+1)+PEEK(R1+2):X=10*PEEK(R1+4)+PEEK(R1+5)  
2890 PRINT B;"/";X;"/83";:B=10*PEEK(R1+6)+PEEK(R1+7):X=10*PEEK(R1+8)+PEEK(R1+9)  
2900 PRINT TAB(15);B;" Hrs. ";X;" Min ": RETURN  
2910 PRINT : PRINT "*****";CS;" Entered (Y/N)": INPUT FS: IF FS="Y" RETURN  
2920 IF FS="N" EXIT E9
```

```

.....
..* FLOATING BREAKWATER DATA ACQUISITION SOFTWARE
..* Revised Tape Headers included 10/4/83
..* Senaka Ratnayake, University of Washington
.....

```

```

..MACHINE LANGUAGE SUBROUTINES

```

```

..
FLAG EQU 0A000H
SP EQU 02H
PROG EQU 03H
FLAG2 EQU FLAG+7
ORG 6800H
SEX PROG;OUT 1;DC OFOH;SEX SP ..MODEM GP.*
LOOK
  B2 DATA          ..SET FLAGS
  B4 TS             .. THRU EF LINES
  INP 5; .AND.10H  ..  AND MODEM
  BZ MODEM          ..  RING
  BR LOOK
DATA
  A.1(FLAG2)->R8.1;A.0(FLAG2)->R8.0
  OOH;STR R8        ..HRLY. ACTIVATED PROG. FLAG
  EXIT
TS
  A.1(FLAG2+1)->R8.1;A.0(FLAG2+1)->R8.0
  OOH;STR R8        ..TIME SERIES FLAG
  EXIT
MODEM
  A.1(FLAG2+2)->R8.1;A.0(FLAG2+2)->R8.0
  OOH;STR R8        ..SET TEL. RING FLAG
  EXIT

```

```

.....
.. THE FOLLOWING SEGMENT OF THE PROGRAM SCANS
.. SOME INPUT CHANNELS AND WRITES ON RAM

```

```

..
MA EQU 0EH
WRITE1 EQU 0AH
NSAMP EQU 0CH          ..* OF SAMPLES
WRITE2 EQU 0DH
COUNTER EQU 08H       ..SAMPLE COUNTER
RAM EQU 0B000H        ..RAM ADDRESS RAW DATA STORAGE
NS EQU FLAG+1         ..RAM LOC.STORES # OF SAMPLES
NBYTE EQU FLAG+3      ..RAM LOC. FOR # OF TAPE BYTES
CLOCK EQU 70H         ..HIGHEST SIG.BYTE OF CLOCK AD
..
DC 'STATHDR'          ..IDENTIFICATION WORD
..
..FIRST SAVE RBASIC REGISTERS IN STACK
..
R1.1;STXD;R1.0;STXD
R9.1;STXD;R9.0;STXD
..
A.1(WRAM1)->WRITE1.1;A.0(WRAM1)->WRITE1.0
A.1(WRAM2)->WRITE2.1;A.0(WRAM2)->WRITE2.0
OOH->COUNTER.1;OOH->COUNTER.0 ..ZERO INTO SAMPLE COUNT
A.1(RAM)->MA.1;A.0(RAM)->MA.0 ..STARTING ADDRESS
..FOR DUMPING RAW DATA INTO RAM
OOH->NSAMP.1;77H->NSAMP.0 ..120-1 IN COUNTER
CLOCK->R9.1;OFH->R9.0
OOH;STR R9           ..INITIALIZE

```

```

LDN R9;LDN R9;LDN R9 .. INTERRUPT
REQ
A.1(INT)->R1.1
A.0(INT)->R1.0
09H;STR R9 ..0.5 S INTERRUPT
C5
NSAMP.1;BNZ C5
NSAMP.0;BNZ C5
BR EXITA
.. INTERRUPT SERVICE
RET
INT
DEC SP
SAVE;DEC SP ..SAVE OLD X,P
STXD ..SAVE D
PROG.1;STXD ..SAVE R3
PROG.0;STXD
LDN R9;LDN R9 .. INTERRUPT FLIP FLOP
A.1(BEGIN)->PROG.1 ..CHANGE PC
A.0(BEGIN)->PROG.0 .. TO
SEP PROG .. R3
BEGIN
SEQ ..HOLD SAMPLE UNTIL 'REQ'
INC COUNTER ..INCREASE COUNTER BY 1
DEC NSAMP ..#OF SAMPLES
SEX PROG
OUT 1;DC 30H
OUT 5;DC 06H ..CHAN 7
SEP WRITE1
OUT 5;DC 07H ..CHAN 8
SEP WRITE1
OUT 5;DC 0FH ..CHAN 16
SEP WRITE1
OUT 1;DC 60H
OUT 5;DC 0CH ..CHAN 61
SEP WRITE2
OUT 1;DC 70H
OUT 5;DC 08H ..CHAN 73
SEP WRITE2
OUT 5;DC 0AH ..CHAN 75
SEP WRITE2
REQ
SEX SP
A.1(ENDINT)->R1.1;A.0(ENDINT)->R1.0
SEP R1
ENDINT
INC SP ..RESTORE
LDXA;->PROG.0 .. ALL
LDXA;->PROG.1 .. QUANTITIES
LDXA
BR INT-1 ..EXIT INT. PROG.
EXITA
00H;STR R9 ..STOP INTERRUPT
LDN R9;LDN R9;LDN R9
..
A.1(NS)->RA.1;A.0(NS)->RA.0
COUNTER.1;STR RA;INC RA ..# OF SAMPLES
COUNTER.0;STR RA
..
..RESTORE REGISTERS BEFORE RETURNING
..
SEX SP;INC SP
LDXA;->R9.0;LDXA;->R9.1
LDXA;->R1.0;LDX;->R1.1
EXIT ..RETURN TO MAIN PROG
.....

```

```

.. SUBROUTINE WRAM1 - WRITES 12 BIT DATA ON RAM
..
SEP PROG
WRAM1
BN1 S;SEX MA
INP 3;IRX
INP 2;IRX
SEX PROG
BR WRAM1-1
.....
.. SUBROUTINE WRAM2 -WRITES 8 BIT DATA ON RAM
..
SEP PROG
WRAM2
BN1 S          ..WAIT TILL CONV. IS DONE
SEX MA
INP 3          ..INPUT DATA BYTE
IRX            ..INCREMENT MEMORY ADDRESS
SEX PROG
BR WRAM2-1     ..RETURN
.....
..2048 SAMPLE SCAN PROGRAM
..
PROG EQU 03H
CHANNEL EQU 0DH
COUNT1 EQU 0AH
COUNT2 EQU 0FH
ANADIG EQU 0BH
TAPE EQU 0CH
UART EQU 0EH
WRITE EQU 09H
NUMSAMP EQU FLAG+10
..
DC 'TIMEHDR'          ..IDENTIFICATION WORD
..
RO.1;STXD;RO.0;STXD
R1.1;STXD;R1.0;STXD  .. SAVE
R9.1;STXD;R9.0;STXD  .. RBASIC
RB.1;STXD;RB.0;STXD  .. REGISTERS
RF.1;STXD;RF.0;STXD
..
A.1(NBYTE)->RA.1
A.0(NBYTE)->RA.0     ..LOAD 512-N2 INTO RO
LDN RA;->RO.1
INC RA;LDN RA;->RO.0
REQ
A.1(FLAG+9)->WRITE.1 ..RAM LOCATION FOR DATA BYTE
A.0(FLAG+9)->WRITE.0 .. INPUT THROUGH INP STATEMENT
A.1(ANALOG)->ANADIG.1 ..LOAD ADDRESS OF A/D CONV.
A.0(ANALOG)->ANADIG.0 .. ROUTINE INTO RB
A.1(SUB2)->TAPE.1    ..LOAD ADDRESS OF TAPE
A.0(SUB2)->TAPE.0    ..OUTPUT ROUTINE
A.1(THRE)->UART.1    ..UART
A.0(THRE)->UART.0    .. ROUTINE
..
A.1(NUMSAMP)->R8.1    ..READ
A.0(NUMSAMP)->R8.0    ..# OF SAMPLES
LDN R8;->COUNT1.1    .. FROM RAM
INC R8;LDN R8;->COUNT1.0
..
A.1(LOOP1)->R1.1
A.0(LOOP1)->R1.0     ..INT. PROG. ADDRESS
..
SEX PROG          ..BEGIN .25 S
OUT 1;DC 04H
OUT 3;DC 1DH

```



```

SEP UART;OUT 2;DC 02H ..NEW WRITE COMMAND
OUT 1;DC 10H
OUT 6;DC 0FFH
OUT 6;DC 0FDH      .. INT.
SEX SP

..
CHECK
COUNT1.1;BNZ CHECK ..KEEP CHECKING
COUNT1.0;BNZ CHECK .. FOR THE CORRECT *
BR TERMINATE      .. OF SAMPLES

..
RET              ..RETURN FROM INT. TASK
LOOP1
.. INTERRUPT SERVICE FIRST
DEC SP
SAVE;DEC SP      ..SAVE OLD X,P
STXD             ..SAVE D
PROG.1;STXD      ..SAVE OLD
PROG.0;STXD      .. R3
A.1(SAMPLE)->PROG.1
A.0(SAMPLE)->PROG.0
SEX PROG
SAMPLE
SEQ             ..SAMPLE AND HOLD SIGNAL
DEC COUNT1      ..DEC COUNTER
00H->COUNT2.1  ..CHANNEL COUNTER (16)
10H->COUNT2.0
A.1(CHAN)->CHANNEL.1
A.0(CHAN)->CHANNEL.0
LOOP2
SEX PROG
OUT 1;DC 30H    ..SELECT CARD 1
OUT 6;DC 00H    ..12 BIT RESOLUTION
SEP ANADIG      ..INITIATE A/D COVERSION
OUT 1;DC 40H    ..CARD 2
SEP ANADIG
OUT 1;DC 50H    ..CARD 3
SEP ANADIG
OUT 1;DC 60H    ..CARD 4
SEP ANADIG
OUT 1;DC 70H    ..CARD 5
SEX CHANNEL
OUT 5           ..INITIATE CONV. CHANNEL * IS
                .. INCREMENTED

SEX PROG
OUT 1;DC 30H    ..ACCESS CARD 1 AGAIN FOR
                .. INPUT/OUTPUT
SEP TAPE        ..CALL SUBROUTINE SUB2 TO INPUT
                .. MOST SIG. 8 BITS,AND WRITE IT

OUT 1;DC 30H
SEX WRITE
INP 2          ..LEAST SIG 4 BITS (+TRAILING 4 ZEROS)
SEX PROG
OUT 1;DC 04H
OUT 3;DC 1DH
SEP UART
SEX WRITE;OUT 2
DEC WRITE
DEC RO
RO.1;BNZ J1    ..CHECK WHETHER 512 BYTES
RO.0;BNZ J1    .. IF SO CHECK TAPE STATUS
CALL STATBYT   .. AND RESET TAPE
02H->RO.1;00H->RO.0
J1
SEX PROG
OUT 1;DC 40H    .. CARD 2

```

```

SEP TAPE
OUT 1;DC 50H          .. CARD 3
SEP TAPE
OUT 1;DC 60H          .. CARD 4
SEP TAPE
OUT 1;DC 70H          .. CARD 5
SEP TAPE
DEC COUNT2
COUNT2.0;BNZ LOOP2   ..CHECK CHANNEL COUNTER. IF
                        .. NOT ZERO REPEAT.
REQ;SEX SP             ..ELSE TOGGLE SAMPLING FLAG
A.1(ENDO)->R1.1;A.0(ENDO)->R1.0
SEP R1                 .. AND EXIT INT. PROG.
ENDO
INC SP                 .. AFTER RESTORING
LDXA;->PROG.0
LDXA;->PROG.1          .. ALL QUANTITIES
LDXA;BR LOOP1-1
TERMINATE
SEX PROG
OUT 1;DC 10H          ..STOP INTERRUPT
OUT 6;DC OFFH
OUT 6;DC OFDH
OUT 1;DC 04H
OUT 3;DC 1DH
J6
SEP UART
OUT 2;DC 00H          ..WRITE 403 0'S TO
DEC RO                 .. COMPLETE 512
RO.1;BNZ J6           .. BYTE SEGMENT
RO.0;BNZ J6
CALL STATUS           ..CHECK TAPE STATUS
..
INC SP                 ..RESTORE
LDXA;->RF.0;LDXA;->RF.1
LDXA;->RB.0;LDXA;->RB.1 .. RBASIC
LDXA;->R9.0;LDXA;->R9.1 .. REGISTERS
LDXA;->R1.0;LDXA;->R1.1
LDXA;->RO.0;LDX;->RO.1 .. AND
EXIT                  .. RURN
.....
CHAN
DC 00H                ..C
DC 01H                ..H NOTE: FULL SCALE
DC 02H                ..A
DC 03H                ..N IS REACHED AT 5.0V.
DC 04H                ..N FULL SCALE VOLTAGE
DC 05H                ..E CAN BE LOWERED IN
DC 06H                ..L SOFTWARE BY ADJUSTING
DC 07H                .. THE GAIN.
DC 08H                ..N
DC 09H                ..U
DC 0AH                ..M
DC 0BH                ..B
DC 0CH                ..E
DC 0DH                ..R
DC 0EH                ..S
DC 0FH
.....
.. SUBROUTINE WRITES THE 8 BIT DATA BYTE ON TAPE
..
SEP R3
SUB2                   ..WAIT FOR CONVERSION
BN1 $
SEX WRITE
INP 3                 ..INPUT DATA BYTE

```

```

SEX TAPE
OUT 1;DC 04H      ..UART
OUT 3;DC 1DH
SEX SP
J11
INP 3;SHL;BNF J11
SEX WRITE
OUT 2;DEC WRITE
SEX PROG
DEC RO
RO.1;BNZ SUB2-1   ..512 BYTES ?
RO.0;BNZ SUB2-1   ..IF NOT RETURN
SEX SP
PROG.1;STXD
PROG.0;STXD
A.1(J2)->PROG.1
A.0(J2)->PROG.0
SEP PROG
J2
CALL STATBYT      .. CHECK TAPE
A.1(J3)->TAPE.1   .. STATUS
A.0(J3)->TAPE.0
SEP TAPE          .. AND
J3
INC SP
LDXA;->PROG.0
LDX;->PROG.1
O2H->RO.1;00H->RO.0 .. RESET BYTE COUNTER
SEX PROG
BR SUB2-1
.....
.. A/D CONVERSION INITIATING SUBROUTINE
..
SEP PROG
ANALOG
SEX CHANNEL
OUT 5;DEC CHANNEL ..INITIATE CONV. REPAIR CHAN.*
SEX PROG
BR ANALOG-1      ..RETURN
.....
..THIS SUBROUTINE CHECKS TAPE STATUS BYTES
..
STATBYT
TRAK EQU FLAG+6
SEX PROG
OUT 3;DC 1DH     ..UART CONTROL REGISTER
SEX SP
A4
INP 3;SHR;BNF A4
INP 2;STXD      ..DRIVE STATUS (DS)
A5
INP 3;SHR;BNF A5
INP 2;STXD      ..INTERFACE STATUS (IS)
A.1(TRAK)->R8.1   ..TRAK #
A.0(TRAK)->R8.0   ..RAM LOC.
A.1(THRE)->UART.1 ..UART ROUTINE
A.0(THRE)->UART.0
INC SP;INC SP   ..CHECK 'DS' FIRST
LDX
SHR;SHR;SHR    ..AT EOT ?
BNF A3
SEX PROG
OUT 1;DC 10H
OUT 6;DC OFFH
OUT 6;DC OFDH
DEC SP;DEC SP

```

```

CALL SLEEP
SEX PROG
OUT 1;DC 04H
OUT 3;DC 1DH
SEP UART
OUT 2;DC 03H .. IF
CALL BYTES
CALL SLEEP .. EOT
SEP UART
OUT 2;DC 03H ..WRITE 2 FMKS
CALL BYTES
INC SP;LDX ..LOAD 'IS'
.AND.03H ..ZERO ALL BITS EXCEPT TK
.XOR.03H ..TK 4 ?
BNZ A1
STR R8
ODOH;STXD ..IF TK4, CHANGE TO TK1 AGAIN
BR A2
A1
LDX;.AND.03H ..IF NOT TK4
+01H;STR R8 ..INC. TRAK AND STORE
+ODOH;STXD
A2
CALL SLEEP
SEP UART;
INC SP;SEX SP
OUT 2 ..OUTPUT NEW MODE ARGUMENT
SEP UART
OUT 2;DC 00H ..POS. ARGUMENT
SEP UART
OUT 2;DC 43H ..WRITE FMK
CALL BYTES
CALL SLEEP .. AND
SEP UART
OUT 2;DC 07H .. REV. SPACE ONE FILE
CALL BYTES
CALL SLEEP
SEP UART
OUT 2;DC 02H ..NEW WRITE COMMAND
OUT 1;DC 10H
OUT 6;DC 0FFH ..BEGIN INT.
OUT 6;DC 0FDH
EXIT
A3
CALL SLEEP
SEP UART
OUT 2;DC 02H ..NEW WRITE COMMAND (WRT CURR. POS)
EXIT
..
PAGE
.....
..AUXILIARY TAPE ROUTINE
..
STATUS
A.1(TRAK)->R8.1
A.0(TRAK)->R8.0
A.1(THRE)->UART.1
A.0(THRE)->UART.0
A15
INP 3;SHR;BNF A15
INP 2;STXD .. 'DS'
A16
INP 3;SHR;BNF A16
INP 2;STXD .. 'IS'
INC SP;INC SP;LDX ..'DS'
SHR;SHR;SHR ..AT EOT ?

```

```

BNF A20
DEC SP;DEC SP
CALL SLEEP
SEP UART;OUT 2
DC 03H
CALL BYTES
CALL SLEEP
SEP UART;OUT 2
DC 03H
CALL BYTES
INC SP;LDX          ..'IS'
.AND.03H           ..OBTAIN TK*
.XOR.03H           ..TK 4 ?
BNZ A17
STR R8             ..ZERO TK*
ODOH;STXD
BR A18

A17
LDX;.AND.03H
+01H;STR R8
+ODOH;STXD

A18
CALL SLEEP
SEP UART
INC SP;SEX SP
OUT 2              ..'MA'
SEP UART
OUT 2;DC 00H      ..'PA'
SEP UART
OUT 2;DC 43H      ..WRITE FMK AND
CALL BYTES;CALL SLEEP .. REV. SPACE
SEP UART;OUT 2;DC 07H .. OVER IT
CALL BYTES

A20
EXIT
.....
..THIS SUBROUTINE IMPOSES A 2 MS DELAY
SLEEP
R7.1;STXD         ..SAVE R7 FIRST
R7.0;STXD
OOH->R7.1;66H->R7.0

A9
DEC R7;R7.0
BNZ A9
INC SP
LDXA;->R7.0
LDX;->R7.1        ..RESTORE R7
EXIT
.....
..THIS SUBROUTINE CHECKS UART STATUS BYTE 'THRE'
SEP PROG
THRE
SEX SP

A6
INP 3;SHL;BNF A6
SEX PROG
BR THRE-1
.....
..THIS SUBROUTINE WRITES 10 TIME BYTES ON RAM
CLOCK->RA.1;ODH->RA.0 ..POINT TO YRS.
TIME
LDN RA;.AND.OFH
STR R8;.XOR.OFH
BZ TIME
DEC RA;INC R8
RA.0;.XOR.03H

```

```

BNZ TIME
EXIT
.....
..THIS SUBROUTINE WRITES CURRENT TIME (10 BYTES)
..
CLOCK->RA.1;ODH->RA.0 ..POINT TO 'YEARS'
L2
LDN RA ..READ TIME REG.
.AND.OFH ..REMOVE MOST SIG 4 BITS
STR SP
.XOR.OFH ..CHECK WHETHER A OF
BZ L2 ..IF SO RE-READ TIME
DEC SP;DEC RA ..POINT TO NEXT TIME REG.
C2
INP 3;SHL;BNF C2
INC SP;OUT 2 ..WRITE TIME ON TO TAPE
DEC SP
NO
RA.0;.XOR.03H ..POINTER AT 'TENS OF SEC' ?
BNZ L2 ..IF NOT REPEAT
EXIT ..RETURN
.....
..CHECK WHETHER UART IS READY TO RECEIVE
RECV
INP 3;SHR;BNF RECV
EXIT
.....
..CHECK WHETHER UART IS READY TO TRANSMIT
TRANS
INP 3;SHL;BNF TRANS
EXIT
.....
..CHECK WHETHER A/D CONVERSION IS COMPLETE
..
BN1 $
EXIT
.....
BYTES
INP 3;SHR;BNF BYTES;INP 2
B1
INP 3;SHR;BNF B1;INP 2
EXIT
.....
..THIS ROUTINE WRITE ZEROS ON TAPE
.. TO COMPLETE 512 BYTES
..
SEX PROG
OUT 1;DC 04H
OUT 3;DC 1DH
Q2
SEX SP
INP 3;SHL;BNF Q2
SEX PROG
OUT 2;DC 00H
DEC R8;R8.1
BNZ Q2;R8.0
BNZ Q2
EXIT
.....
..ROUTINE TO FORCE MICROTERMINAL STATE
SEX PROG
OUT 1;DC 00H
EXIT
.....
PAGE
..SHORT SCAN OF A SINGLE CHANNEL

```

```

COUNT EQU OAH
RAM EQU OBOOOH
REQ
R1.1;STXD;R1.0;STXD ..SAVE R1
A.1(RAM)->MA.1;A.0(RAM)->MA.0
CLOCK->RD.1;OFH->RD.0
A.1(INTERRUPT)->R1.1
A.0(INTERRUPT)->R1.0
OOH;STR RD
LDN RD;LDN RD;LDN RD
SEX MA;OUT 1;DEC MA ..GP #
SEX SP
O9H;STR RD
CHEK
COUNT.1;BNZ CHEK
COUNT.0;BNZ CHEK
BR FINISH
..
..SAMPLING
..
RET
INTERRUPT
DEC SP
SAVE;DEC SP;STXD
LDN RD;LDN RD
SEQ ..SAMPLE & HOLD
DEC COUNT
R8.0;STR SP
OUT 5;DEC SP ..CHAN #
EN1 $;SEX MA
INP 3;IRX ..READ BYTE(S)
R8.1;BZ JUMP1 ..IF 8 BITS JUMP
INP 2;IRX
JUMP1
REQ;SEX SP
INC SP;LDXA ..RESTORE D
BR INTERRUPT-1 ..END SAMPLE
FINISH
OOH;STR RD
LDN RD;LDN RD;LDN RD ..STOP INT.
SEX PROG;OUT 1;DC OFOH
SEX SP;INC SP
LDXA;->R1.0;LDX;->R1.1..RESTORE R1
EXIT

```

.....
..REAL TIME DATA SCAN

```

COUNT EQU OAH
RAM EQU OBOOOH
A.1(RAM)->RE.1;A.0(RAM)->RE.0
REQ
R1.1;STXD;R1.0;STXD ..SAVE R1
CLOCK->RD.1;OFH->RD.0
A.1(INTERR)->R1.1
A.0(INTERR)->R1.0
OOH;STR RD
LDN RD;LDN RD;LDN RD
O9H;STR RD
CHEK1
COUNT.1;BNZ CHEK1
COUNT.0;BNZ CHEK1
BR FINIS

```

..SAMPLING

```

RET
INTERR
  DEC SP
  SAVE;DEC SP;STXD
  LDN RD;LDN RD
  PROG.1;STXD;PROG.0;STXD
  A.1(PC)->PROG.1;A.0(PC)->PROG.0
  SEP PROG
PC
  SEQ                      ..SAMPLE & HOLD
  DEC COUNT
  SEX RE;OUT 1;DEC RE     ..GP *
  SEX SP;R8.0;STR SP
  OUT 5;DEC SP           ..CHAN *
  BN1 $
  INP 3;->RC.1
  R8.1;BZ JUMP2         ..IF 8 BITS JUMP
  INP 2;->RC.0
JUMP2
  SEX PROG;OUT 1;DC OFOH
  CALL TRANS
  RC.1;STR SP;OUT 2;DEC SP
  R8.1;BZ JUMP3
  CALL TRANS
  RC.0;STR SP;OUT 2;DEC SP
JUMP3
  A.1(PC1)->R1.1;A.0(PC1)->R1.0
  SEP R1
PC1
  INC SP;LDXA;->PROG.0
  LDX;->PROG.1
  REQ
  INC SP;LDXA           ..RESTORE D
  BR INTERR-1         ..END SAMPLE
FINIS
  OOH;STR RD
  LDN RD;LDN RD;LDN RD  ..STOP INT.
  SEX SP;INC SP
  LDXA;->R1.0;LDX;->R1.1..RESTORE R1
  EXIT

```

```

..
PAGE
.. TEST. CDP18S643 MICROBOARD
PROG EQU 03H
SP EQU 02H
TYPE1 EQU 81A4H
TYPE2 EQU 81AEH
AUX EQU 0CH
CHAR EQU OFH
CONT R8.1;STR SP;OUT 1;DEC SP ..GP *
R8.0;STR SP;OUT 5;DEC SP ..CHAN *
  BN1 $
  INP 3
  ->AUX.1
  INP 2;->AUX.0
  SEX PROG
  OUT 1;DC 01H
  AUX.1->CHAR.1
  CALL TYPE2
  AUX.0->CHAR.1
  CALL TYPE
  ODH->CHAR.1;CALL TYPE1
  OAH->CHAR.1;CALL TYPE1
  BR CONT

```

```

..
TYPE

```



```

GHI CHAR
SHR;SHR;SHR;SHR
ADI OF6H
BNF $+04
ADI 7
SMI OC6H;PLO RE
LDI 10H
PLO CHAR
SEX R2
BEG INP 3
    SHL
    BNF BEG
    GLO RE
    STR R2
    OUT 2
    DEC R2
    BR NEXCHAR
    ORG $+15
NEXCHAR GLO CHAR;ADI OFOH
        PLO CHAR
        BNF TEXTIT
        SMI 10H
        BZ TEXTIT
        BNF TEXTIT
        LDI 0
        PLO RE
        BR BEG
TEXTIT EXIT
..
.. CDP18S 644/648/654
..
ADVAL EQU OCH
START R8.1;STR SP;OUT 1;DEC SP      ..GP *
      R8.0;STR SP;OUT 5;DEC SP      ..CHAN *
      BN1 $
      INP 3;->ADVAL.0
      SEX PROG;OUT 1;DC 01H
      OSH->RD.0;OOH->RD.1
CONTINUE ADVAL.0*2->ADVAL.0
        OOH+"3OH->CHAR.1
        CALL TYPE1
        DEC RD;RD.0
        BNZ CONTINUE
        ODH->CHAR.1;CALL TYPE1
        BR START

```

APPENDIX IIB COMPUPRO TAPE READING SYSTEM

PROGRAM SCRON

BASIC STATISTICAL CALCULATIONS

```

C PROGRAM SRCON: STATIST RECORD CONVERSION
C
C REVISION 1.2 REVISED OCT 5 1983 TO INCLUDE CHS. 90,91, AND 92
C AIR TEMP, SCC TEMP (WHATEVER THAT MEANS), AND SYSTEM TEMP RESPCT.
C
C PROGRAMED BY ROBERT W. MILLER, U.W. SUMMER 1983
C DEPARTMENT OF CIVIL ENGINEERING
C FOR US ARMY CORPS OF ENGINEERS WEST POINT PROTOTYPE BREAKWATER PROGRAM
C FOR USE WITH SUPER-SOFT 16-BIT FORTRAN
C
C THIS PROGRAM TAKES TAKES 1 MIN RECORD FILES OUTPUT FROM PROGRAM
C QIF AND PRODUCES A READABLE SUMMARY OF STATISTICS. IN ADDITION,
C A MAP IS PRODUCED SHOWING LOCATION OF ALL HEADERS AS WELL AS ANY
C GLITCHES ON THE TAPE THAT MIGHT BE PRESENT.
C
C UNIT # FILES USED FOR I/O:
C
C 6 - "GMAP.DAT" TEMPORARY FILE WRITTEN BY PROGRAM QIF WHICH PROVIDES
C INFORMATION FOR TAPE MAPS LISTED AT BEGINNING OF
C TAPE SUMMARY. THIS FILE IS USED ONLY IN THE FIRST
C PORTION OF THE PROGRAM.
C
C 6 - "1MINREC.DAT" TEMPORARY FILE WRITTEN BY PROGRAM QIF WHICH CONTAINS
C AN ACTUAL COPY OF ALL "STATIST" HEADERS ENCOUNTERED
C ON A TAPE DURING A TAPE SEARCH
C
C 8 - "BW[TAPE #].1MR" OUTPUT FILE WRITTEN BY THIS PROGRAM WHICH IS
C ASCII TEXT OF TAPE SUMMARIES. TO PRODUCE HARD
C COPY IT IS ONLY NECESSARY TO LIST THIS FILE OUT
C TO PRINTER
C
C IMPORTANT VARIABLES:
C IBUF1(2,512) DATA ARRAY
C IBUF2(512) EQUIVALENT TO IBUF1(2,512) (SEE BELOW)
C IDATI DATE AND TIME TAPE PUT INTO BREAKWATER ACQUISITION SYST.
C IDATO DATE AND TIME TAPE TAKEN OUT
C IDATC DATE AND TIME OF CURRENT RECORD
C FNAME, SUBSTG, TRACK, SUEX,
C SR, TS, BW, TAPNUM,
C STRCHK CHARACTER FILENAMES USED FOR FILENAME GENERATION & OPENING
C KOUNT COUNT OF NUMBER OF ENTRIES ON CURRENT LINE
C MAPBUF ARRAY CONTAINING INTEGER CODES TO BE INTERPRETED IN MAKING MAP
C THESE VARIABLES ARE INPUT FROM TEMPORARY FILE "GMAP.DAT"
C WRITTEN BY PROGRAM QIF
C
C IGLTCH COUNT OF NUMBER OF GLITCHES FOUND SINCE LAST TIME LINE
C WAS WRITTEN
C
C NTS TIMESERIES NUMBER READ FROM MAPBUF - 4
C TRACK NUMBERS ARE INDICATED BY -1,-2,-3,-4 IN MAPBUF
C TIMESERIES #'S MUST THEREFORE BE -5,-6....
C WHILE 1 MIN REC #'S ARE 1,2,3,...
C
C ITAPE TAPE NUMBER
C ITRK TAPE TRACK RECORD TAKEN FROM
C

```

INTEGER IBUF2(512),MAPBUF(100)

```

INTEGER IDATI (5), IDATO (5), IDATC (5)
INTEGER*1 IBUF1 (2, 512)
CHARACTER*15 FNAME, SUBSTG
CHARACTER*2 ASI, ASO, ASC
CHARACTER*2 SR, TS, BW
CHARACTER*6 TAPNUM, STRCHK
CHARACTER*5 TRACK
CHARACTER*4 SUFX
CHARACTER*90 LABLIS
CHARACTER*210 LABLS2
EQUIVALENCE (IBUF2, IBUF1)

```

```

C
C HEADER BYTES AS DUMPED OFF QANTEX ARE LOADED DIRECTLY INTO IBUF1
C CONVERSION OF TWO-BYTE INTEGERS IS DONE BY EQUIVALENCING 2-BYTE ARRAY
C IBUF2 TO IBUF1. FOR FURTHER DETAILS, SEE DOCUMENTATION FOR
C PROGRAM BWSORT
C

```

```

C THIS PROGRAM FILLS THESE ARRAYS ONLY WITH 1 MINUTE RECORD HEADERS
C

```

```

COMMON /ONE/ SR, TS, KOUNT, MAPBUF, IGLTCH, NTS
COMMON /TWO/ IBUF2
DATA TRACK /"TRACK"/
DATA BW /"BW"/
DATA SUFX /".LMR"/
DATA SR /"SR"/
DATA TS /"TS"/
DATA MAPBUF /100*0/
DATA IBUF2 /512*0/
DATA LABLIS /"ANCH.FORCE (7) ANCH.FORCE (8) TIDE HT (16) VERT.A
2CC (61) WIND SP. (73) WIND DIR. (75) "/
DATA LABLS2 /"CURR.VEL.X (77) CURR.VEL.Y (
278) WIND SP. (74) WIND DIR. (75) -10V (83) +10V (84) -
324V (85) +24V (86) -5V (87) +5V (88) WATER T
4EMP (89) AIR TEMP (90) SCC TEMP (91) SYS TEMP (92) "/

```

```

C*****

```

```

502 FORMAT (A0)

```

```

C OPEN MAP FILE AND READ TAPE NUMBER, ITAPE
IF (IOREAD (6, 2, 0, "GMAP.DAT")) GOTO 900
READ (6, 500) ITAPE

```

```

C GENERATE OUTPUT FILENAME BY WRITING TAPE NUMBER INTO ALPHA VARIABLE AND
C CONCATENATING PROPER PREFIXES AND SUFFIXES TO IT
WRITE (TAPNUM, 525) ITAPE

```

```

525 FORMAT (" ", I6)
TAPNUM=STRCHK (TAPNUM)
CALL CONCAT (FNAME, BW, TAPNUM, SUFX)

```

```

C OPEN OUTPUT FILE
IF (IOWRIT (8, 2, 0, FNAME)) GOTO 900

```

```

C WRITE MAP HEADING
WRITE (8, 403)
WRITE (8, 400)

```

```

400 FORMAT ("1 STATISTICAL RECORD AND GLITCH MAP"/
2" -----")
WRITE (8, 406)

```

```

C READ AND WRITE TAPE DATES & TIMES
READ (6, 503) (IDATI (I), I=1, 5)

```

```

503 FORMAT(5I6)
  READ(6,503) (IDATO(I),I=1,5)
C  CALL SUBROUTINE THAT ADJUSTS FROM 2400 HOUR FORMAT TO AM/PM
  CALL DADJS (ASI,IDATI(4))
C  IDOI INSURES THAT 11:00, FOR EXAMPLE, IS WRITTEN WITH TWO ZEROS
  IDOI=1
  IF (IDATI(5) .LT. 10) IDOI=0
  IF (IDOI .EQ. 1) WRITE(8,401) (IDATI(I),I=1,5),ASI
401 FORMAT("    TAPE IN : ",I2,"/",I2,"/",I2,2X,I2,":",I2,LX,A2)
  IF (IDOI .EQ. 0) WRITE(8,422) (IDATI(I),I=1,4),IDOI,IDATI(5),ASI
422 FORMAT("    TAPE IN : ",I2,"/",I2,"/",I2,2X,I2,":",I1,I1,LX,A2)
  CALL DADJS(ASO,IDATO(4))
  IDOO=1
  IF (IDATO(5) .LT. 10) IDOO=0
  IF (IDOO .EQ. 1) WRITE(8,402) (IDATO(I),I=1,5),ASO
402 FORMAT("    TAPE OUT :",I2,"/",I2,"/",I2,2X,I2,":",I2,LX,A2)
  IF (IDOO .EQ. 0) WRITE(8,423) (IDATO(I),I=1,4),IDOO,IDATO(5),ASO
423 FORMAT("    TAPE OUT :",I2,"/",I2,"/",I2,2X,I2,":",I1,I1,LX,A2)
  WRITE(8,410) ITAPE
  WRITE(8,403)
403 FORMAT(/)
C  BEGIN PRINTING OF MAP: START WITH TRACK ONE
  ITRK=1
  WRITE(8,404) TRACK,ITRK
404 FORMAT("    ",A5,LX,I2)
  WRITE(8,405)
405 FORMAT("    -----")
  WRITE(8,403)
406 FORMAT(//)
  IGLTCH=0
  NTS=0
  20 CONTINUE
C  READ AND ENTRY AND TEST IT TO SEE WHAT IT IS
  READ(6,500,ENDFILE=910) I
500 FORMAT(I6)
  IF (I .LT. 0) GO TO 50
  KOUNT=KOUNT+1
C  MAPBUF= ARRAY OF 1 MIN RECORDS THAT HAVEN'T YET BEEN PRINTED.
C  KOUNT = ELEMENTS OF MAPBUF HAVE VALID ENTRIES
  MAPBUF(KOUNT)=I
C  ONLY FOURTEEN ENTRIES CAN FIT ON ONE LINE: START NEW LINE IF NECESSARY
  IF (KOUNT .GE. 14) GO TO 70
C  IF NO TIMESERIES HEADER HAS BEEN MAPPED, GO READ ANOTHER ENTRY
  IF (NTS .EQ. 0) GO TO 20
C  IF A TIMESERIES ENTRY HAS BEEN READ, GO PRINT IT AS WELL AS ANY 1 MIN
C  RECORDS IN MAPBUF
C  THE FOLLOWING FIDDLING WITH KOUNT IS NECESSARY BECAUSE SRPNT IS NOT CALLED
C  WHEN A TIMESERIES IS DETECTED BUT ONLY AFTER THE NEXT ONE MINUTE RECORD
C  HAS BEEN READ (SO THAT GLITCHES CAN BE RECORDED WITH THE TIMESERIES)
  KOUNT=KOUNT-1
  CALL SRPNT
  MAPBUF(1)=MAPBUF(KOUNT+1)
  KOUNT=1
  GO TO 20
50 CONTINUE

```

```

C WE COME HERE IF A NEGATIVE 'I' WAS READ EITHER SIGNIFYING END OF A
C TRACK (-1 TO -4) OR TIMESERIES HEADER (-5 AND BELOW) OR A GLITCH
C (-999)
    I=-I
C IF I LT 5, END OF TRACK IS INDICATED
    IF (I .LT. 5) GO TO 60
    IF (I .EQ. 999) GO TO 95
C I=888 SIGNIFIES END OF RECORD
    IF (I .EQ. 888) GO TO 100
C ELSE, A TIMESERIES HEADER WAS ENCOUNTERED
C TIMESERIES NUMBERS ARE OFFSET BY -4 TO AVOID CONFUSION WITH TRACK NOS.
    NTS=I-4
    GO TO 20
60 CONTINUE
C WE COME HERE IF END OF TRACK HAS BEEN DETECTED
C FIRST PRINT OUT ANY PENDING 1 MIN REC #'S
    IF (KOUNT .NE. 0) CALL SRPNT
C NOW WRITE TRACK HEADING
    WRITE(8,403)
    I=I+1
    WRITE(8,407) TRACK, I
407 FORMAT(4X,A5,I3)
    WRITE(8,405)
    WRITE(8,403)
    KOUNT=0
    GO TO 20
70 CONTINUE
    CALL SRPNT
C RESET KOUNT AND RETURN
    KOUNT=0
    GO TO 20
95 CONTINUE
    IGLTCH=IGLTCH+1
    GO TO 20
100 CONTINUE
C WE COME HERE IF END OF MAP FLAG WAS ENCOUNTERED:
C FIRST CLEAR MAPBUF OF ANY PENDING ENTRIES
    CALL SRPNT
C READ AND WRITE # OF 1 MINUTE RECORDS TO SCREEN
    READ(6,503) NREC
    WRITE(1,510) NREC
510 FORMAT(" NUMBER OF RECORDS = ",I5)
    IF (IOCL0S(6)) GO TO 900
C*****
C OPEN FILE CONTAINING 1 MIN REC HEADER BYTES AND PROCEED TO WRITE NREC DATA
C SUMMARIES
    IF (IORAND(512,1,6,0,"1MINREC.DAT")) GO TO 900
C LOOP OVER NUM OF RECORDS
    DO 200 I=1,NREC
    WRITE(8,408)
C WRITE HEADING AT TOP OF PAGE
408 FORMAT("1      1 MIN. STATISTICAL DATA SUMMARY")
    WRITE(8,409)
409 FORMAT("      -----")
    WRITE(8,406)

```

```

WRITE(8,410) ITAPE
410 FORMAT ("    TAPE # ",I3)
IF (IDOI .EQ. 1) WRITE(8,401) (IDATI(J),J=1,5),ASI
IF (IDOI .EQ. 0) WRITE(8,422) (IDATI(J),J=1,5),IDOI,ASI
IF (IDOO .EQ. 1) WRITE(8,402) (IDATO(J),J=1,5),ASO
IF (IDOO .EQ. 0) WRITE(8,423) (IDATO(J),J=1,5),IDOO,ASO
C READ IN Ith BLOCK
READ(6/I) (IBUF1(1,K),K=1,512)
C WRITE # OF SAMPLES
IPVAL1=IBUF2(215)*256+IBUF2(216)
WRITE(8,411) IPVAL1
411 FORMAT ("    # OF SAMPLES = ",I4)
C WRITE TAPE TRACK
WRITE(8,412) IBUF2(12)+1
412 FORMAT ("    TAPE TRACK =",I2)
IDA=IBUF2(17)*10+IBUF2(18)
IHR=IBUF2(19)*10+IBUF2(20)
IMIN=IBUF2(21)*10+IBUF2(22)
C CALL ROUTINE TO CALCULATE DATE AND TIME OF CURRENT RECORD BASED UPON
C DATE TAPE PUT IN AND TIME ELAPSED SINCE THEN
CALL CURDAT(IDATC,ASC,IDATI,IDA,IHR,IMIN,ASI)
IDOC=1
C NOW WRITE DATE AND TIME OF RECORD
IF (IDATC(5) .LT. 10) IDOC=0
IF (IDOC .EQ. 1) WRITE(8,426) IBUF2(8)*256+IBUF2(9), (IDATC(J)
2,J=1,5),ASC
426 FORMAT ("    1 MIN REC. No. =",I3,6X,I2,"/",I2,"/",I2,2X,I2,":",
2I2,1X,A2)
IF (IDOC .EQ. 0) WRITE(8,413) IBUF2(8)*256+IBUF2(9), (IDATC(J)
2,J=1,4),IDOC,IDATC(5),ASC
413 FORMAT ("    1 MIN REC. No. =",I3,6X,I2,"/",I2,"/",I2,2X,I2,":",
2I1,I1,1X,A2)
WRITE(8,406)
WRITE(8,414)

C
C PROCESS MULTI-BYTE SAMPLING VALUES
C
414 FORMAT ("    QUANTITY/CH.#          MEAN          STD.DEV
2    MAX          MIN")
WRITE(8,406)
IARG=217
ISP=-14
C K=1 : PROCESS ANCH. FORCES AND TIDE HEIGHT
C K=2 : PROCESS VERT ACCEL, WIND SP, WIND DIR
DO 125 K=1,2
C J=1,2,3 = WHICH OF THREE CHANNELS CURRENTLY PROCESSING FOR GIVEN K
C ISP=STARTING POSITION OF LABEL STRING
C IEP=ENDING POSITION OF LABEL STRING
DO 120 J=1,3
ISP=ISP+15
IEP=ISP+14
C VARIABLE FNAME BEING REUSED AS SCRATCH
FNAME=SUBSTG(LABLIS,ISP,IEP)
IFLAG=(K-1)*2
C FUNCTION CONVER DOES CONVERSION AND POINTER INCREMENTING IN THIS LOOP

```

```

PVAL1=CONVER (IFLAG, IARG)
PVAL2=CONVER (IFLAG, IARG)
IFLAG=IFLAG+1
PVAL3=CONVER (IFLAG, IARG)
PVAL4=CONVER (IFLAG, IARG)
WRITE (8, 415) FNAME, PVAL1, PVAL2, INT (PVAL3), INT (PVAL4)
120 CONTINUE
125 CONTINUE
C END OF MULTI-BYTE CHANNELS LOOP
415 FORMAT (5X, A15, 8X, F8.2, 5X, F8.2, 2 (8X, I6))
WRITE (8, 406)
WRITE (8, 416)
416 FORMAT (" SINGLE SAMPLE VALUES")
WRITE (8, 417)
417 FORMAT (" -----")
WRITE (8, 418)
418 FORMAT (/)
ISP=-14
C LOOP THROUGH SINGLE SAMPLE VALUES
DO 150 J=0, 13
C CHECK TAPE # TO SEE IF IT'S RECENT ENOUGH TO HAVE CHS. 90-92
IF ((ITAPE .LE. 75) .AND. (J .GT. 10)) GO TO 150
ISP=ISP+15; IEP=ISP+14
FNAME=SUBSTG (LABLS2, ISP, IEP)
WRITE (8, 419) FNAME, IBUF2 (IARG+J)
419 FORMAT (5X, A15, 8X, I6)
150 CONTINUE
C
200 CONTINUE
C DONE
GO TO 920
C ERROR TRAPS
900 CONTINUE
WRITE (1, 511)
511 FORMAT (" FILE IO ERROR")
STOP
910 CONTINUE
WRITE (1, 512) I
512 FORMAT (" ENDFILE ERROR: LAST # READ =", I5)
920 IF (IOCLOS (8)) GO TO 900
WRITE (1, 513)
513 FORMAT (" ORDERLY HALT")
STOP
END
C
C SUBROUTINE DADJS ADJUSTS THE DAT FROM 2400 HR FORMAT TO AM/PM
C
SUBROUTINE DADJS (AS, IDAT)
CHARACTER*2 AS
AS="AM"
IF (IDAT .GE. 12) AS="PM"
IF (IDAT .GT. 12) IDAT=IDAT-12
IF ((IDAT .EQ. 0) .AND. (AS .EQ. "AM")) IDAT=12
RETURN
END

```


C
C SUBROUTINE SRPNT IS THE PRINTING SUBROUTINE FOR THE GLITCH
C MAPPING SEGMENT OF THE PROGRAM
C

```
      SUBROUTINE SRPNT
      INTEGER MAPBUF(100)
      CHARACTER*1 AST
      CHARACTER*2 SR,TS,BLK
      CHARACTER*80 LINE
      CHARACTER*6 STRCHK,CBUF
      COMMON /ONE/ SR,TS,KOUNT,MAPBUF,IGLTCH,NTS
      DATA AST /"*/
      BLK="  "
      SR="SR";TS="TS"
      LINE="
2          "
      IF ((IGLTCH .NE. 0) .AND. (NTS .EQ. 0)) WRITE(1,100)
100  FORMAT(" POSSIBLE ERROR: GLITCH OUTSIDE CONTEXT OF TIMESERIES")
      IF (KOUNT .EQ. 0) GO TO 200
      KCOMP=1
      IPOS=3
10  CONTINUE
      WRITE(CBUF,103) MAPBUF(KCOMP)
103  FORMAT(LX,I6)
      CBUF=STRCHK(CBUF)
      I=KLEN(CBUF)
      CALL SETLEN(CBUF,I)
      CALL INSERT(SR,LINE,IPOS)
      IPOS=IPOS+2
      CALL INSERT(CBUF,LINE,IPOS)
      IPOS=IPOS+2
      IF (MAPBUF(KCOMP) .GT. 9) IPOS=IPOS+1
      IF (KCOMP .EQ. KOUNT) GO TO 200
      KCOMP=KCOMP+1
      GO TO 10
20  CONTINUE
      IF (NTS .EQ. 0) GO TO 40
      WRITE(CBUF,103) NTS
      CBUF=STRCHK(CBUF)
      I=KLEN(CBUF)
      IPOS=IPOS+2
      CALL SETLEN(CBUF,I)
      CALL INSERT(TS,LINE,IPOS)
      IPOS=IPOS+2
      CALL INSERT(CBUF,LINE,IPOS)
C  IGCOC COUNTS '*'S PUT ON MAP AND IS COMPARED TO IGLTCH
      IGCOC=1
      IPOS=IPOS+1
      IF (NTS .GT. 9) IPOS=IPOS+1
      IF (IGLTCH .EQ. 0) GO TO 40
25  CONTINUE
      IPOS=IPOS+1
      CALL INSERT(AST,LINE,IPOS)
      IF (IGCOC .GE. IGLTCH) GO TO 40
      IGCOC=IGCOC+1
```

```

GO TO 25
40 CONTINUE
WRITE (8,105) LINE
105 FORMAT(" ",A80)
GO TO 220
200 CONTINUE
WRITE (1,101)
101 FORMAT(" IN SRPNT SR: 'KOUNT' SHOULD NOT EQUAL ZERO")
220 CONTINUE
NTS=0
IGLTCH=0
RETURN
END

```

C-

C- FUNCTION TO RETURN VALUE OF ALPHA STRINGS LEFT JUSTIFIED IN FIELD

C-

```

CHARACTER*6 FUNCTION STRCHK(ALPARG)
CHARACTER*6 ALPARG,SUBSTG
CHARACTER*1 ALPTST
IPTR1=0
20 CONTINUE
IPTR1=IPTR1+1
ALPTST=SUBSTG(ALPARG,IPTR1,IPTR1)
IF (ALPTST.EQ." ") GOTO 20
IF (ALPTST.EQ."0") GOTO 20
IF (IPTR1 .GT.6) GOTO 50
STRCHK=SUBSTG(ALPARG,IPTR1,6)
RETURN
50 WRITE (1,100)
100 FORMAT(" ERROR IN STRING OR BLANK STRING: SR STRCHK")
IF (IOCLOS(8)) STOP
STOP
END

```

C

C FUNCTION TO DO INTEGER TO REAL CONVERSIONS WITH PROPER SCALING

C ISAD=STARTING ADDRESS IN IBUF ARRAY

C IFLAG=0 ==> XX.X ; IFLAG=1 ==> XX. ; IFLAG=2 ==> X.X ; IFLAG=3 ==> X.

C

```

REAL FUNCTION CONVER(IFLAG,ISAD)
COMMON /TWO/ IBUF2(512)
SCRATC=0.
IF (IFLAG .AND. 2) GO TO 20
SCRATC=FLOAT(IBUF2(ISAD))*256.
ISAD=ISAD+1
20 CONTINUE
SCRATC=SCRATC+FLOAT(IBUF2(ISAD))
ISAD=ISAD+1
IF (IFLAG .AND. 1) GO TO 30
SCRATC=SCRATC+FLOAT(IBUF2(ISAD))/100.
ISAD=ISAD+1
30 CONTINUE
CONVER=SCRATC
RETURN
END

```

C

```

C  SUBROUTINE TO PRODUCE DATE AND TIME OF INDIVIDUAL 1 MIN. REC.
  SUBROUTINE CURDAT (IDATC,ASC, IDATI, IDA, IHR, IMIN, ASI)
  INTEGER IDATC (5), IDATI (5), IDLIM (12)
  CHARACTER*2 ASC, ASI
  DATA IDLIM /31,28,31,30,31,30,31,31,30,31,30,31/
  IF (MOD (IDATI (3),4) .EQ. 0) IDLIM (2)=IDLIM (2)+1
  DO 10 J=1,5
10  IDATC (J)=0
  IDUM=IDATI (4)
  IF (ASI .EQ. "PM") IDUM=IDUM+12
  IDATC (5)=IDATI (5)+IMIN
  IF (IDATC (5) .GE. 60) IDATC (4)=IDATC (4)+1
  IF (IDATC (5) .GE. 60) IDATC (5)=IDATC (5)-60
  IDATC (4)=IDATC (4)+IDUM+IHR
  IF (IDATC (4) .GE. 24) IDATC (2)=IDATC (2)+1
  IF (IDATC (4) .GE. 24) IDATC (4)=IDATC (4)-24
  IDATC (1)=IDATC (1)+IDATI (1)
  IDATC (2) = IDATI (2)+IDA+IDATC (2)
  IDATC (3)=IDATI (3)+IDATC (3)
  ILIM=IDLIM (IDATI (1))
  IF (IDATC (2) .GT. ILIM) IDATC (1) = IDATC (1) + 1
  IF (IDATC (2) .GT. ILIM) IDATC (2)=IDATC (2)-ILIM
  IF (IDATC (1) .GT. 12) IDATC (3)=IDATC (3)+1
  IF (IDATC (1) .GT. 12) IDATC (2)=IDATC (2)-31
  IF (IDATC (1) .GT. 12) IDATC (1) = IDATC (1)-12
  CALL DADJS (ASC, IDATC (4))
  RETURN
  END

```

APPENDIX IIC COMPUPRO ANALYSIS SYSTEM
SPECTRAL ANALYSIS PROGRAM

C SIMPLE STATISTICAL PROGRAM v. 1.3 4-19-84

C

C PROGRAM WRITTEN FOR ANALYSIS OF BREAKWATER DATA

C

C Written in SuperSoft FORTRAN v.1.04cpm

C

C*****

C

Guiding Philosophy

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

Explanation of control variables

COMMON BLOCK /CONTRL/

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

NSATT = NUMBER OF SAMPLES ATTEMPTED
NSGOOD = NUMBER OF SAMPLES SUCCESSFULLY TAKEN
TAPENO = TAPE NUMBER
RECNO = RECORD NUMBER
TSNO = TIME SERIES NUMBER
TPTRAK = TAPE TRACK (1 - 4)
TIME(I,J) = TIME TAPE PUT IN (J=1) AND TAKEN OUT (J=2)
 (I=1,5) = MONTH, DAY, YEAR, HOUR, MINUTE
ELTIME(I) = ELAPSED TIME SINCE TAPE PUT IN (DAYS, HOURS, MINUTES)
SMSTAT(I,J) = UNSCALED SUMMARY STATISTICS FOR DATA CHANNELS
 (I=1,5) = MIN, MAX, MEAN, STANDARD DEVIATION, SCALE FACTOR
 J = CHANNEL NUMBER (1-80)
STCOMP = LOGICAL VARIABLE, TRUE IF SMSTAT HAS BEEN COMPUTED
CHINF(I,J) = CHANNEL INFORMATION FOR DATA IN ARRAY A
 I = ARRAY NUMBER (1 OR 2)
 (J=1,3) = CHANNEL NUMBER, SCALED (1=YES), FFT'ED (1=YES, 2=HALF-FREQ)
TITLE(I) = 8 CHARACTER LABEL FOR CHANNEL I
FILNAM = NAME OF DATA FILE WITHOUT EXTENDER
IL(I) = ARRAY OF STRING LENGTHS IN COMMON
 I=1-80 --> TITLE
 I=81 --> FILNAM
FILE2 = SCRATCH CHARACTER VARIABLE, put here during memory shortage
HERTZ = SAMPLING FREQUENCY IN HERTZ
ARRSTS(I,J) = Scaled and transformed stats for data in arrays
 I = Min, Max, Mean, Standard Deviation, Scale Factor
 J = Array number (1 or 2)
INDEV = Input device (1 for console input, 9 for macros)

C COMMON BLOCK/NO PLOT/
 C PLOTP = True means suppress plotting of arrays with a zero
 C standard deviation during macro files. This block
 C is common to only MACRO and PLOTR.

C*****

C Blank Common (data)

C A(I,J) = IN CORE REAL STORAGE OF TS DATA
 C I = SAMPLE POINT (1 - 4096)
 C J = ARRAY NUMBER (1 OR 2)

C*****

C Files

UNIT	NAME	CONTENTS
1	<console I/O>	
5	file.DAT	Raw binary integer data (changed only by UPDATE)
6	file.HDR	Header information
7	file.STS	Summary statistics (binary)
8	file.OUT	Printable 80 channel statistical summary
9	<read in>	SSP command macro (convention is type of .SSP)
10	file.Cxx	PLOTR output for channel xx (.Fxx for FFTs)
11	<read in>	Raw FFT coefficients in binary form
12	<read in>	Phase and coherency plotter file

C*****

C Required Input

C SSP requires two files for input: a file of two-byte binary integers
 C that make up the data and a header file of ASCII text that contains
 C information about the data. The format for the header file is:

Line No.	Format	Variables
1	4(5X,I5)	TAPENO,RECNO,TSNO,TPTRAK
2	5I10	TIME(i,1) (mo.,day,yr.,hr.,min. tape put in)
3	5I10	TIME(i,2) (mo.,day,yr.,hr.,min. tape taken out)
4	3I10	ELTIME(i) (days,hrs.,mins. since tape put in)
5	2I10	NSATT,NSGOOD
6-15	8F10.5	SMSTAT(i,5) (scale factors, -9999.=> use default)
16+	A80	Comment (ignored)

C The binary file must be a rectangular file with 80 blocks of NSATT
 C two byte integers each. Any data past NSGOOD in each block is ignored,
 C but must be present in order to preserve rectangularity.

C
 C COMMON/CONTRL/NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME(5,2),
 C & ELTIME(3),SMSTAT(5,80),STCOMP,CHINF(2,3),TITLE(80),FILNAM,
 C & IL(81),FILE2,HERTZ,ARRSTS(5,2),INDEV
 C INTEGER NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME,ELTIME,CHINF
 C REAL SMSTAT
 C LOGICAL STCOMP

```

CHARACTER*14 FILNAM,FILE2
CHARACTER*8 TITLE
COMMON A(4098,2)
COMMON/NO PLOT/PLOTP
LOGICAL PLOTP
DATA PLOTP/.FALSE./
DATA INDEV/1/
DATA SMSTAT/4*0.,.027,4*0.,.027,4*0.,.027,4*0.,.027,4*0.,.027,4*0.,.027,
& 4*0.,.027,4*0.,.027,4*0.,
& .027,4*0.,.019,4*0.,.019,4*0.,.019,4*0.,.019,4*0.,1.0,4*0.,
& 1.0,4*0.,1.0,4*0.,.012,4*0.,1.0,4*0.,.031,4*0.,.031,4*0.,.031,4*0.,
& .031,4*0.,.004,4*0.,1.0,4*0.,.004,4*0.,.004,4*0.,.004,4*0.,.004,
& 4*0.,1.0,4*0.,.004,4*0.,.004,4*0.,1.0,4*0.,1.0,4*0.,1.0,4*0.,
& 1.0,4*0.,1.0,4*0.,1.0,4*0.,1.0,4*0.,1.0,4*0.,1.0,4*0.,1.0,4*0.,
& 1.0,4*0.,1.0,4*0.,1.0,4*0.,1.0,4*0.,1.47,4*0.,1.47,4*0.,1.47,4*0.,
& 1.47,4*0.,1.47,4*0.,1.47,4*0.,1.47,4*0.,1.47,4*0.,1.47,4*0.,
& 1.47,4*0.,1.47,4*0.,1.47,4*0.,1.47,4*0.,1.47,4*0.,
& 1.47,4*0.,1.47,4*0.,1.0,4*0.,1.0,4*0.,1.0,4*0.,1.0,4*0.,1.0,
& 4*0.,1.0,4*0.,1.0,4*0.,1.0,4*0.,1.0,4*0.,1.0,4*0.,1.0,4*0.,
& 1.0,4*0.,.4,4*0.,
& .4,4*0.,1.412,4*0.,1.412,4*0.,1.0,4*0.,1.0,4*0.,1.0,4*0.,1.0/
DATA TITLE /"LNWC","LNEC","UNWC","USWC","UNEC","USEC","LSWC",
& "LSEC","LNT","UNT","UST","LST","FRS","FRC","FRN","TIDE",
& "INC BUOY","WV-NW","WV-NE","WV-SW","WV-SE","P1-NU","P2-NL",
& "P3-BNC","P4-BEO","P5-BEI","P6-BCC","P7-BWI","P8-BWO","P9-BSC",
& "P10-SCU","P11-SW1","P12-SW2","P13-SW3","P14-SCL","P15-SE3",
& "P16-SE2","P17-SE1","P18-EP1","P19-EP2","P20-EP3","P21-EP4",
& "P22-EP5","P23-EP6","NULE","NBLE","SBLE","SULE","NT1","NT2",
& "BT1","BT2","UT1","UT2","ST2","ST1","NULC","NBLC","SULC","SBLC",
& "W VERT","W HORZ","W ROT","E VERT","E HORZ","E ROT","WVR","WHR",
& "EVR","EHR","CLM","CRM","WIND SP","WIND SP","WIND DIR","WIND DIR",
& "N-S","E-W","CON1","CON2"/
DO 99 I=1,80
  IL(I)=KLEN(TITLE(I))
99 CONTINUE
C
  CALL INIT(IERR)
  IF(IERR.EQ.1)GOTO 190
C
101 CALL MENU(ISUB)
C
C ISUB = NUMBER OF DESIRED SUBROUTINE
C
  GOTO (111,112,113,114,115,116,117,118,119,120,121,122,190),ISUB
C
C LOAD A DATA ARRAY
111 CALL READA
  GO TO 150
C
C SUMMARY STATISTICS
112 CALL STATS
  GO TO 150
C
C SCALE A ARRAY
113 CALL SCALE

```

```

      GO TO 150
C
C CLIP AN ARRAY
114 CALL CLIP
      GO TO 150
C
C Detrend an array containing time series data
115 CALL DETRND
      GO TO 150
C
C Perform band-pass filtering on an array
116 CALL FILTER
      GO TO 150
C
C Write an array of data to the master data file (.DAT)
117 CALL UPDATE
      GO TO 150
C
C DO FAST FOURIER TRANSFORM
118 CALL DOFFT
      GO TO 150
C
C WRITE OUT RAW FFT COEFFICIENTS
119 CALL FFTOUT
      GO TO 150
C
C COMPUTE CROSS-SPECTRAL PHASE AND COHERENCY
120 CALL CROSPC
      GO TO 150
C
C Execute a macro command file
121 CALL MACRO
      GO TO 150
C
C PLOT DATA
122 CALL PLOTR
      GO TO 150
C
C LOOP BACK TO MENU CALL
150 GO TO 101
C
C END OF JOB PROCESSING
190 WRITE(1,200)
200 FORMAT(" Goodbye...")
      STOP
195 WRITE(1,210)
210 FORMAT(" Error exit")
      STOP
      END
C
C-----
C
C SUBROUTINE INIT(IERR)
C
C THIS ROUTINE IS CALLED ONCE, AT THE BEGINING OF THE PROGRAM AND

```


C FIRST PROMPTS THE USER FOR THE NAME OF THE DATA FILE. AFTER THE
 C NAME IS "NORMALIZED" FOR EASE OF USE, THE DATA FILE (.DAT) IS OPENED
 C FOR READING AND THE ACCOMPANYING HEADER (.HDR) AND SUMMARY STATISTICS
 C FILES (.STS) ARE READ INTO THE CONTRL COMMON BLOCK

C
 COMMON/CONTRL/NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME(5,2),
 & ELTIME(3),SMSTAT(5,80),STCOMP,CHINF(2,3),TITLE(80),FILNAM,
 & IL(81),FILE2,HERTZ,ARRSTS(5,2),INDEV
 INTEGER NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME,ELTIME,CHINF
 REAL SMSTAT
 LOGICAL STCOMP
 CHARACTER*14 FILNAM,UPCASE,FILE2
 CHARACTER*8 TITLE
 REAL TEMP(80)
 CHARACTER*1 ANS
 IERR = 0
 WRITE(1)12
 101 WRITE(1,200)
 200 FORMAT(" Simple Statistical Program v. 1.3",//,
 & " What is the name of your data file? "\$
 READ(INDEV,210,ENDFILE=180,ERREXIT=180)FILNAM
 210 FORMAT(A0)
 FILNAM=UPCASE(FILNAM)

C
 C Modify FILNAM to <filename>"." so that extenders can be added easily
 C

103 K=INDEX(".",FILNAM,(1))
 IF(K.NE.0)GO TO 105
 K=KLEN(FILNAM)
 CALL CONCAT(FILE2,FILNAM,".")
 FILNAM=FILE2
 GO TO 103
 105 CALL SETLEN(FILE2,(0))
 CALL SETLEN(FILNAM,K)
 IL(81)=K

C
 C OPEN HEADER FILE
 C

CALL CONCAT(FILE2,FILNAM,"HDR")
 IF(.NOT.IOLOOK((0),FILE2))GO TO 170
 IF(IOREAD((6),(2),(0),FILE2))GO TO 170

C
 C FILE OPENED, SO READ IT AND ISSUE MESSAGE
 C

WRITE(1,215)FILE2
 215 FORMAT(" Reading ",A0)
 READ(6,217)TAPENO,RECNO,TSNO,TPTRAK,HERTZ
 217 FORMAT(4I10,F10.0)
 IF(HERTZ.EQ.0)HERTZ=4.0
 220 FORMAT(5I10)
 READ(6,220)(TIME(I,1),I=1,5)
 READ(6,220)(TIME(I,2),I=1,5)
 READ(6,220)(ELTIME(I),I=1,3)
 READ(6,220)NSATT,NSGOOD

C

C Everything but the scale factors has now been read, so attempt to
C open and read .DAT (data) and .STS (statistics) files
C

```
CALL SETLEN(FILE2,(0))
CALL CONCAT(FILE2,FILNAM,"DAT")
IF (IORAND(2*NSATT,(2),(5),(0),FILE2))GO TO 170
CALL SETLEN(FILE2,(0))
CALL CONCAT(FILE2,FILNAM,"STS")
IF (.NOT.IOLOOK((0),FILE2))GO TO 113
```

C
C JUMP TO 113 IF FILE.STS DOES NOT EXIST, OTHERWISE READ IT AND THE
C NEW SCALE FACTORS FROM THE HEADER FILE
C

```
IF (IOREAD((7),(0),(0),FILE2))GO TO 170
READ(7)((SMSTAT(I,J),I=1,5),J=1,80)
IF (IOCLOS(7))GO TO 170
STCOMP=.TRUE.
113 READ(6,230)(TEMP(I),I=1,80)
230 FORMAT(9(8F10.5,/),8F10.5)
DO 107 I=1,80
    IF ((TEMP(I).NE.-9999.).AND.(TEMP(I).NE.0.0)) SMSTAT(5,I)=TEMP(I)
107 CONTINUE
RETURN
```

C
C ERROR HANDLING CODE FOR FILE I/O ERRORS
C

```
170 WRITE(1,250)FILE2
250 FORMAT(" ***Unable to open file ",A0)
    IERR=1
    RETURN
180 WRITE(1,260)
260 FORMAT(" ***Huh? Please try again")
    GO TO 101
END
```

C
C
C

```
CHARACTER*14 FUNCTION UPCASE(X)
CHARACTER*14 X
INTEGER*1 KHAR,K
```

C Function to convert filenames to uppercase

```
UPCASE=X
DO 110 I=1,KLEN(X)
    K=KHAR(X,I)
    IF ((K.GE.97).AND.(K.LE.122))CALL PUTCHR(UPCASE,I,K-32)
110 CONTINUE
CALL SETLEN(UPCASE,KLEN(X))
RETURN
END
```

C
C
C

```
SUBROUTINE MENU(ISUB)
COMMON/CONTRL/NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME(5,2),
& ELTIME(3),SMSTAT(5,80),STCOMP,CHINF(2,3),TITLE(80),FILNAM,
```

```

& IL(81),FILE2,HERTZ,ARRSTS(5,2),INDEV
INTEGER NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME,ELTIME,CHINF
REAL SMSTAT
LOGICAL STCOMP
CHARACTER*8 TITLE
CHARACTER*14 FILNAM,FILE2
COMMON A(4098,2)
CHARACTER*6 SCALED
CHARACTER*3 FFTED
DATA NSUB/13/

```

```

C
C THIS SUBROUTINE DISPLAYS THE OPTIONS OPEN TO THE USER AND READS HIS CHOICE
C

```

```

C NSUB == NUMBER OF ACTIVE SUBROUTINES PLUS 1 (FOR STOP)
C

```

```

C START BY LISTING OUT AVAILABLE SUBROUTINES IN A NUMBERED MENU FORM
C

```

```

101 WRITE(1,200)
200 FORMAT(//," 1 Load a channel into an array",/,
& " 2 Summary statistics on one or all 80 channels",/,
& " 3 Scale an array",/,
& " 4 Extreme value smoothing (clipping)",/,
& " 5 Detrend a time series",/,
& " 6 High/Low/Band pass filter on a time series",/,
& " 7 Write an array to the master data file",/,
& " 8 Perform a Fast Fourier Transform",/,
& " 9 Write raw FFT coefficients to a file",/,
& " 10 Compute and write cross-spectral phase and coherency",/,
& " 11 Execute a macro file",/,
& " 12 Plot an array",/,
& " 13 Quit",//)

```

```

C
C WRITE CONTENTS OF ARRAYS A & B (THE TWO HALVES OF TRUE ARRAY A),
C INDICATING THE STATUS OF EACH
C

```

```

DO 105 I=1,2
  SCALED=" "
  FFTED=" "
  IF(CHINF(I,1).EQ.0)GO TO 103
  IF(CHINF(I,2).EQ.1) SCALED="Scaled"
  IF(CHINF(I,3).NE.0) FFTED="FFT"
  CALL SETLEN(TITLE(CHINF(I,1)),IL(CHINF(I,1)))
  WRITE(1,210) I+64,CHINF(I,1),TITLE(CHINF(I,1)),SCALED,FFTED
210  FORMAT(" Array ",A1,":",I3,2X,A0,4X,A0,4X,A0$
  IF(CHINF(I,3).NE.0)WRITE(1,211)CHINF(I,3)
211  FORMAT(2X,I3$
  WRITE(1,212)
212  FORMAT(1X)
  GO TO 105
103  WRITE(1,220) I+64
220  FORMAT(" Array ",A1,": Unused")
105 CONTINUE

```

```

C
C NOW PROMPT FOR AND READ THE USER'S CHOICE, CHECKING TO SEE THAT IT
C IS IN THE VALID RANGE OF 1 TO NSUB

```

C

```
WRITE(1,230)
230 FORMAT(//," Enter option desired: "$
READ(INDEV,240,ENDFILE=180,ERREXIT=180) ISUB
IF((ISUB.LE.0).OR.(ISUB.GT.NSUB))GO TO 180
240 FORMAT(I0)
WRITE(1) 12
RETURN
180 WRITE(1) 12
WRITE(1,250) NSUB
250 FORMAT(" ***Response must be an integer between 1 and ",I2,/,
& " ***Please try again")
GO TO 101
END
```

C

C

C

```
SUBROUTINE STATS
COMMON/CONTRL/NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME(5,2),
& ELTIME(3),SMSTAT(5,80),STCOMP,CHINF(2,3),TITLE(80),FILNAM,
& IL(81),FILE2,HERTZ,ARRSTS(5,2),INDEV
INTEGER NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME,ELTIME,CHINF
REAL SMSTAT
LOGICAL STCOMP
CHARACTER*8 TITLE
CHARACTER*14 FILNAM,FILE2
COMMON A(4098,2)
CHARACTER*1 ANS
```

C

C THIS SUBROUTINE CONTROLS THE COMPUTATION AND WRITING OF SUMMARY STATISTICS,
C CHECKING TO AVOID UNNECESSARY RECOMPUTATION OF THE SAME.

C

C Ask whether statistics are wanted for all 80 channels or just for a
C single array

C

```
WRITE(1,190)
190 FORMAT(" Enter A or B for statistics on a single array or <CR> ",
& " for all 80 channels "$
READ(INDEV,200) ANS
200 FORMAT(A0)
IF(ANS.EQ."")GO TO 101
```

C

C Statistics are wanted for only a single array, so get them from the
C ARRSTS array and display them to the screen, then return

C

```
IARR=ICARD(ANS)
IF((IARR.NE.1).AND.(IARR.NE.2))GO TO 180
IF(CHINF(IARR,1).EQ.0)GO TO 182
WRITE(1,205) ANS,CHINF(IARR,1),(ARRSTS(I,IARR),I=1,4)
205 FORMAT(/," Array ",A0," Channel",I3,/, " Min. = ",G15.8,/,
& " Max. = ",G15.8,/, " Mean = ",G15.8,/, " StDev. = ",G15.8,
& //," Hit <CR> to continue"$
READ(INDEV,200) ANS
RETURN
```

C

C Statistics have been requested for the entire data file, so open output
C file for tabular data on all 80 channels. If statistics have already
C been computed then do not recompute but merely write them out without
C affecting the .STS file (to do this, call WRTSTS with an argument of 1)
C

```
101 CALL SETLEN(FILNAM,IL(81))
    CALL SETLEN(FILE2,(0))
    CALL CONCAT(FILE2,FILNAM,"OUT")
    IF(IOWRIT((8),(2),(0),FILE2))GO TO 184
    IF(STCOMP)GO TO 130
```

C
C COMPUTE THE STATISTICS, UPDATING THE SCREEN EACH ITERATION
C

C IN THE WRITE BEFORE THE LOOP (WITH FORMAT 210) THE FINAL CARRIAGE
C RETURN IS SUPPRESSED AND THEN IN LOOP, J10 AND J ARE THE TENS AND
C ONES DIGITS OF THE CURRENT CHANNEL. THEN THE UNFORMATTED WRITE SENDS
C TWO ASCII BACKSPACES (THE 8'S) FOLLOWED BY J10 AND J CONVERTED TO
C CHARACTER BY THE ADDITION OF AN ASCII '0' (48)
C

```
103 WRITE(1,210)
210 FORMAT(" Computing statistics for channel 01"$
    DO 105 I=1,80
        J10=I/10
        J=I- (J10*10)
        WRITE(1)8,8,J10+48,J+48
        CALL READCH(I,1)
        CALL COMSTS(I,1)
        DO 104 K=1,4
            SMSTAT(K,I)=ARRSTS(K,1)
104 CONTINUE
105 CONTINUE
    CHINF(1,1)=0
```

C
C Stats are now computed, so write them to output file
C

```
    WRITE(1,220)
220 FORMAT(/," Writing output file...")
    CALL WRTSTS((0))
    STCOMP=.TRUE.
    IF(IOCLOS(8))GO TO 186
    RETURN
130 WRITE(1,220)
```

C
C Write a new page of statistics without rewriting the .STS file
C (program branches here only if stats have already been calculated)
C

```
    CALL WRTSTS((1))
    IF(IOCLOS(8))GO TO 186
    RETURN
```

C
C Error messages
C

```
180 WRITE(1,230)
230 FORMAT(" ***Array designator must be either A or B")
    RETURN
```

```

182 WRITE(1,240)ANS
240 FORMAT(" ***Array ",A0," is empty")
RETURN
184 WRITE(1,250)FILE2
250 FORMAT(" ***Unable to open ",A0)
RETURN
186 WRITE(1,260)FILE2
260 FORMAT(" ***Unable to close ",A0)
RETURN
END

```

C
C
C

```

SUBROUTINE COMSTS (ICHAN,IARR)
COMMON/CONTRL/NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME(5,2),
& ELTIME(3),SMSTAT(5,80),STCOMP,CHINF(2,3),TITLE(80),FILNAM,
& IL(81),FILE2,HERTZ,ARRSTS(5,2),INDEV
INTEGER NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME,ELTIME,CHINF
REAL SMSTAT
LOGICAL STCOMP
CHARACTER*8 TITLE
CHARACTER*14 FILNAM,FILE2
COMMON A(4098,2)
DOUBLE PRECISION SUM,SUMSQ,DBLE,DSQRT
REAL MAX,MIN

```

C
C
C
C
C
C
C

This sub computes the min, max, mean and standard deviation of channel ICHAN in array IARR
Double precision variables are used to prevent real overflow in the sum and sum of squares variables

```

MAX=-10.
MIN=300000.
SUM=0.0D0
SUMSQ=0.0D0
DO 103 J=1,NSGOOD
IF (A(J,IARR).LT.MIN)MIN=A(J,IARR)
IF (A(J,IARR).GT.MAX)MAX=A(J,IARR)
SUM=SUM + DBLE(A(J,IARR))
SUMSQ=SUMSQ + DBLE(A(J,IARR)*A(J,IARR))

```

```

103 CONTINUE
ARRSTS(1,IARR)=MIN
ARRSTS(2,IARR)=MAX
ARRSTS(3,IARR)=SNGL(SUM/DBLE(FLOAT(NSGOOD)))
ARRSTS(4,IARR)=SNGL(DSQRT((SUMSQ- SUM*SUM/DBLE(FLOAT(NSGOOD)))) /
& DBLE(FLOAT(NSGOOD-1)))
ARRSTS(5,IARR)=SMSTAT(5,ICHAN)
RETURN
END

```

C
C
C

```

SUBROUTINE WRTSTS (IBOTH)
COMMON/CONTRL/NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME(5,2),
& ELTIME(3),SMSTAT(5,80),STCOMP,CHINF(2,3),TITLE(80),FILNAM,

```

```

& IL(81),FILE2,HERTZ,ARRSTS(5,2),INDEV
INTEGER NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME,ELTIME,CHINF
REAL SMSTAT
LOGICAL STCOMP
CHARACTER*14 FILNAM,FILE2
CHARACTER*8 TITLE
COMMON A(4098,2)
DO 101 I=1,80
    CALL SETLEN(TITLE(I),IL(I))
101 CONTINUE
C
C WRSTTS writes out the summary statistics in SMSTAT to the output file.
C If IBOTH is passed in as 1 then the .STS file will not be written.
C
C Start by writing out the title
C
    WRITE(8,200)((TIME(I,J),I=1,3),J=1,2),NSATT,HERTZ
200 FORMAT("1Summary of Statistical Data for West Point Floating",
& " Breakwater Project ",I2,"/",I2,"/",I2," to ",I2,"/",I2,"/",
& I2,/, " Number of samples per event = ",I4,5X,
& " Sampling Rate = ",F4.2, " hertz")
    WRITE(8,210)(ELTIME(I),I=1,3),TAPENO,RECNO,TSNO
210 FORMAT(1X,I2," Days ",I2," Hrs. ",I2," Min. from beginning of tape",
& " (All Min, Max values Measured from Zero Mean)",/,
& " Tape Number ",I3,/,
& " 1 Minute Record Number ",I3,/, " Time Series Number ",I3,
& ///,36X,"SUMMARY OF SCALE FACTORS",/)
C
C WRITE PAGE OF SCALE FACTORS AND TITLES
C
    DO 103 I=1,80,10
        WRITE(8,220)((J),J=I,I+9)
220    FORMAT(/," CH.NO.",5X,10(I2,7X))
        WRITE(8,230)(TITLE(J),J=I,I+9)
230    FORMAT(" L CODE",4X,10(A8,1X))
        WRITE(8,240)(SMSTAT(5,J),J=I,I+9)
240    FORMAT(" SFACT.",4X,10(F5.3,4X))
103 CONTINUE
C
C WRITE PAGE OF STATISTICS
C
    WRITE(8,250)
250    FORMAT("1")
    DO 107 I=1,80,10
        WRITE(8,220)((J),J=I,I+9)
        WRITE(8,270)((SMSTAT(2,J)-SMSTAT(3,J))*SMSTAT(5,J),J=I,I+9)
270    FORMAT(" MAX. ",1X,10(F8.3,1X))
        WRITE(8,260)((SMSTAT(1,J)-SMSTAT(3,J))*SMSTAT(5,J),J=I,I+9)
260    FORMAT(" MIN. ",10F9.3)
        WRITE(8,280)(SMSTAT(3,J)*SMSTAT(5,J),J=I,I+9)
280    FORMAT(" MEAN ",1X,10(F8.3,1X))
        WRITE(8,290)(SMSTAT(4,J)*SMSTAT(5,J),J=I,I+9)
290    FORMAT(" STDEV.",1X,10(F8.3,1X))
107 CONTINUE
C

```

C Write <file>.STS of unscaled summary statistics in binary form, if
C IBOTH != 1
C

```
      IF (IBOTH.EQ.1) RETURN
      CALL SETLEN (FILNAM, IL(81))
      CALL CONCAT (FILE2, FILNAM, "STS")
      IF (IOWRIT((7), (0), (0), FILE2)) GO TO 180
      WRITE (7) ((SMSTAT(I, J), I=1, 5), J=1, 80)
      IF (IOCLOS(7)) GO TO 180
      RETURN
180 WRITE (1, 300) FILE2
300 FORMAT (" ***Unable to write file ", A0)
      RETURN
      END
```

C
C
C

```
      SUBROUTINE READA
      COMMON/CONTRL/NSATT, NSGOOD, TAPENO, RECNO, TSNO, TPTRAK, TIME(5, 2),
      & ELTIME(3), SMSTAT(5, 80), STCOMP, CHINF(2, 3), TITLE(80), FILNAM,
      & IL(81), FILE2, HERTZ, ARRSTS(5, 2), INDEV
      INTEGER NSATT, NSGOOD, TAPENO, RECNO, TSNO, TPTRAK, TIME, ELTIME, CHINF
      REAL SMSTAT
      LOGICAL STCOMP
      CHARACTER*8 TITLE
      CHARACTER*14 FILNAM, FILE2
      COMMON A(4098, 2)
      CHARACTER*1 ANS
```

C
C READA reads a channel, computes the statistics and pads the array out
C to NSATT samples, if necessary.
C

```
101 WRITE (1, 200)
200 FORMAT (" What data channel do you want to load?(1-80) "$
      READ (INDEV, 210, ENDFILE=180, ERREXIT=180) ICHAN
210 FORMAT (I0)
      IF ((ICHAN.LE.0).OR.(ICHAN.GT.80)) GO TO 180
103 WRITE (1, 220)
220 FORMAT (" And into what array?(A/B) "$
      READ (INDEV, 230, ENDFILE=183, ERREXIT=183) ANS
230 FORMAT (A0)
      IARR=ICARD (ANS)
      IF ((IARR.NE.1).AND.(IARR.NE.2)) GO TO 183
      WRITE (1, 240) ICHAN, ANS
240 FORMAT (/," Reading channel ", I2, " into array ", A1, "...")
      CALL READCH (ICHAN, IARR)
```

C
C Compute summary stats for this channel to eliminate effect of earlier
C transformations. Hence whenever a channel is in memory, summary
C stats for it are available, whether or not STCOMP is set.
C

```
      IF (.NOT.STCOMP) CALL COMSTS (ICHAN, IARR)
      IF (.NOT.STCOMP) GO TO 105
      DO 104 I=1, 5
          ARRSTS (I, IARR) = SMSTAT (I, ICHAN)
```



```

104 CONTINUE
C
C PAD ARRAY WITH MEAN VALUE (NULL LOOP IF # OF GOOD SAMPLES IS EQUAL
C TO # OF SAMPLES ATTEMPTED)
C
105 DO 109 I=NSGOOD+1,NSATT
      A(I,IARR)=SMSTAT(3,ICHAN)
109 CONTINUE
C
C REINITIALIZE CHANNEL INFORMATION ARRAY
C
      CHINF(IARR,1)=ICHAN
      CHINF(IARR,2)=0
      CHINF(IARR,3)=0
      RETURN
C
C Error messages
C
180 WRITE(1,250)
250 FORMAT(/," ***Channel number must be an integer from 1 to 80,"
      & " please try again",/)
      GO TO 101
183 WRITE(1,260)
260 FORMAT(/," ***Array descriptor must be either A or B, please",
      & " try again",/)
      GO TO 103
      END
C
C-----
C
SUBROUTINE PLOTR
COMMON/CONTRL/NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME(5,2),
& ELTIME(3),SMSTAT(5,80),STCOMP,CHINF(2,3),TITLE(80),FILNAM,
& IL(81),FILE2,HERTZ,ARRSTS(5,2),INDEV
COMMON/NOPLLOT/PLOTP
INTEGER NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME,ELTIME,CHINF
REAL SMSTAT
LOGICAL STCOMP
CHARACTER*8 TITLE
CHARACTER*14 FILNAM,FILE2
COMMON A(4098,2)
COMPLEX B(2049,2)
EQUIVALENCE (A,B)
CHARACTER*1 ANS
CHARACTER*3 TAIL
LOGICAL PLOTP
C
C THIS ROUTINE IS MAINLY A PLACE KEEPER, REFLECTING THE CURRENT LACK
C OF PLOTTING SOFTWARE COMPATIBLE WITH SS FORTRAN. INSTEAD OF ACTUALLY
C DOING ANY PLOTTING, THE SELECTED DATA ARE WRITTEN OUT IN ASCII WITH A
C CONTROL RECORD ON TOP, IN A FORMAT THAT CAN BE READ BY THE F80 PROGRAM
C "PLOTR" WHICH IS ABLE TO ACCESS THE PLOTTER DIRECTLY.
C
      WRITE(1,200)
200 FORMAT(" Plotting routine is not active, data will be written on",

```

```

& /," a file compatible with PLOTR",/)
101 WRITE(1,210)
210 FORMAT(" Which array do you want to plot?(A/B) "$
      READ(INDEV,220) ANS
220 FORMAT(A0)
      IARR=ICARD(ANS)
      IF((IARR.NE.1).AND.(IARR.NE.2))GO TO 170
C
C PUT "CXX" INTO TAIL, WHERE XX IS THE CHANNEL NUMBER BEING PLOTTED
C AND THEN CONCATENATE THIS TO FILE2 TO FORM THE NAME OF THE FILE THAT
C THE DATA WILL BE WRITTEN ON
C FILE EXTENDER IS FXX FOR PLOTS OF FFT COEFFICIENTS
C
103 CALL SETLEN(FILNAM,IL(81))
      CALL PUTCHR(TAIL,(1),(67#))
      IF(CHINF(IARR,3).NE.0)CALL PUTCHR(TAIL,(1),(70#))
      CALL PUTCHR(TAIL,(2),INT1(CHINF(IARR,1)/10 + 48))
      CALL PUTCHR(TAIL,(3),INT1(MOD(CHINF(IARR,1),10) + 48))
      CALL SETLEN(TAIL,(3))
      CALL CONCAT(FILE2,FILNAM,TAIL)
      IF(IOWRIT((10),(2),(0),FILE2))GO TO 172
      CALL SETLEN(TITLE(CHINF(IARR,1)),IL(CHINF(IARR,1)))
      IF(CHINF(IARR,3).NE.0)GO TO 120
C
C FFT COEFFICIENTS MUST BE HANDLED DIFFERENTLY THAN RAW DATA, SO IF
C THE ARRAY HAS BEEN TRANSFORMED PROCESSING JUMPS TO LINE 120.
C NOW THAT ONLY RAW DATA REMAINS, ASK ABOUT PLOTTING PARAMETERS.
C
      NPLOT=NSGOOD
104 WRITE(1,232)NSGOOD
232 FORMAT(" How many points do you want plotted? (max of ",I4,)"$
      READ(INDEV,233)NPLOT
233 FORMAT(I0)
      IF(NPLOT.GT.0)GO TO 1047
          NPLOT=NSGOOD
          WRITE(1,234)NPLOT
234   FORMAT(" Assuming ",I4," points")
1047 IF(NPLOT.GT.NSGOOD)GO TO 104
      WRITE(1,235)NSGOOD-NPLOT+1
235 FORMAT(" Starting at what point? (max of ",I4,)"$
      READ(INDEV,233)NSTART
      IF(NSTART.EQ.0)NSTART=1
      IF((NSTART.LT.0).OR.(NSTART+NPLOT-1.GT.NSGOOD))GO TO 104
C
C NOW THAT PLOTTING RANGES HAVE BEEN ESTABLISHED, WRITE THE DATA
C But if PLOTP is true and standard deviation is near zero,
C skip the plotting and return.
C
      IF(PLOTP.AND.(ARRSTS(4,IARR).LT.0.001))GO TO 190
      WRITE(1,240)FILE2
240 FORMAT(" Writing data to ",A0,"...")
      WRITE(10,250)0,CHINF(IARR,1),TITLE(CHINF(IARR,1)),NPLOT
250 FORMAT(LX,I1,I4,A8,2X,I5)
      DO 105 I=NSTART,NSTART+NPLOT
          WRITE(10,260)FLOAT(I)/HERTZ, A(I,IARR)

```

```

260     FORMAT(1X,F9.4,1X,F15.8)
105 CONTINUE
     IF (IOCLOS(10)) GO TO 175
     WRITE(1,270)
270 FORMAT(" Writing finished")
     RETURN

C
C  ERROR HANDLING CODE
C
170 WRITE(1,280)
280 FORMAT(" ***Array designator must be either A or B, please try again")
     GO TO 101
172 WRITE(1,290)FILE2
290 FORMAT(" ***Unable to open ",A0)
     RETURN
175 WRITE(1,300)FILE2
300 FORMAT(" ***Unable to close ",A0)
     RETURN
190 WRITE(1,305)
305 FORMAT(///," ***Plotting suppressed because of zero standard deviation")
     I=IOCLOS(10)
     I=IODEL(0,FILE2)
     RETURN

C
C  CODE FOR WRITING FFT COEFFICIENTS IN ARRAY B VERSUS FREQUENCY
C
C  FIRST DETERMINE THE UPPER AND LOWER PLOTTING LIMITS
C
120 DELTAF=HERTZ/FLOAT(NSATT)
     WRITE(1,310)
310 FORMAT(" Enter the lower frequency bound for plotting",
     & " (hit <CR> for 0 Hz) "$
     READ(INDEV,320)FRQLOW
320 FORMAT(F0.0)
     IF (FRQLOW.LT.0.0)FRQLOW=0.0
     WRITE(1,330)HERTZ/FLOAT(2*CHINF(IARR,3))
330 FORMAT(" Enter the upper bound (hit <CR> for ",F4.2," Hz) "$
     READ(INDEV,320)FRQHI
     IF ((FRQHI.EQ.0.0).OR.(FRQHI.GT.HERTZ/FLOAT(2*CHINF(IARR,3))).OR.
     & (FRQHI.LE.FRQLOW))FRQHI=HERTZ/FLOAT(2*CHINF(IARR,3))
     IF (FRQLOW.GE.FRQHI)FRQLOW=0.
     NCOEFF=MIN0(NSATT,2048)/2*CHINF(IARR,3)
     NLOW=MAX0(INT(FRQLOW/DELTAF),1)
     NHIGH=MIN0(INT(FRQHI/DELTAF),NCOEFF)

C
C  DELTAF == DIFFERENCE IN FREQUENCY BETWEEN TWO ELEMENTS IN ARRAY B
C  NCOEFF == TOTAL NUMBER OF RAW FFT COEFFICIENTS IN B
C  NLOW == ARRAY SUBSCRIPT FOR FIRST POINT TO BE PLOTTED
C  NHIGH == ARRAY SUBSCRIPT FOR LAST POINT TO BE PLOTTED
C
     WRITE(1,331)
331 FORMAT(" Do you want the spectrum smoothed for plotting? (Y/N) "$

C
C  If smoothing is selected, data will be "block smoothed" with a window
C  of 16. With no smoothing, all coefficients will be written out.

```

C

```
READ (INDEV,220) ANS
IF (PLOT.P.AND. (ARRSTS (4, IARR) .LT. 0.001)) GO TO 190
WRITE (1,340) FROLOW, FROHI
340 FORMAT (" Plotting from ", F6.3, "Hz to ", F6.3, "Hz")
WRITE (1,240) FILE2
IF ((ANS.NE. "Y") .AND. (ANS.NE. "y")) GO TO 123
NHIGH=MIN0 (NHIGH, NCOEFF-15)
WRITE (10,250) 2, CHINF (IARR,1), TITLE (CHINF (IARR,1)), (NHIGH+15-NLOW)/16
AREA=0.
DO 122 I=NLOW, NHIGH, 16
    SUM=0.0
    DO 121 J=I, I+15
        SUM=SUM+CABS (B (J, IARR)) *CABS (B (J, IARR))
121 CONTINUE
    SUM=SUM*2.0*FLOAT (NSATT)/HERTZ
    WRITE (10,260) FLOAT (I+7)*DELTA, SUM/16.
    AREA=AREA+SUM
122 CONTINUE
    AREA=AREA*DELTA
    WRITE (1,345) AREA
345 FORMAT (" Approximate area under plotted curve is ", E15.6, " square units")
GO TO 105
```

C

C Code for writing unsmoothed coefficients

C

```
123 WRITE (10,250) 2, CHINF (IARR,1), TITLE (CHINF (IARR,1)), NHIGH-NLOW+1
DO 124 I=NLOW, NHIGH
    WRITE (10,260) FLOAT (I)*DELTA,
    & CABS (B (I, IARR)) *CABS (B (I, IARR)) *2.0*FLOAT (NSATT)/HERTZ
124 CONTINUE
GO TO 105
RETURN
END
```

C

C

C

```
SUBROUTINE DOFFT
COMMON/CONTRL/NSATT, NSGOOD, TAPENO, RECNO, TSNO, TPTRAK, TIME (5,2),
& ELTIME (3), SMSTAT (5,80), STCOMP, CHINF (2,3), TITLE (80), FILNAM,
& IL (81), FILE2, HERTZ, ARRSTS (5,2), INDEV
INTEGER NSATT, NSGOOD, TAPENO, RECNO, TSNO, TPTRAK, TIME, ELTIME, CHINF
REAL SMSTAT
LOGICAL STCOMP
CHARACTER*14 FILNAM, FILE2
CHARACTER*8 TITLE
COMMON A (4098,2)
COMPLEX B (2049,2)
EQUIVALENCE (A,B)
CHARACTER*1 ANS, ANS2
REAL MEAN
COMPLEX CMLX
LOGICAL POWOF2
```

C

C THIS IS THE DRIVER ROUTINE THAT PROMPTS FOR INPUT AND CALLS THE ACTUAL

```

C FFT SUBROUTINE
C
C THE ARRAY B IS EQUIVALENCED TO A AND CONTAINS THE SAME DATA, BUT IN
C COMPLEX FORM. THE DATA ARE LEFT IN THIS FORM AFTER THE END OF THE
C ROUTINE AND MUST BE ACCESSED AS COMPLEX SUBSEQUENTLY.
C
  95 WRITE(1,200)
  200 FORMAT(" Which array do you wish to transform? (A/B) "$
    READ(INDEV,210)ANS
  210 FORMAT(A0)
    IARR=ICARD(ANS)
    IF((IARR.NE.1).AND.(IARR.NE.2))GOTO 170
    IF(CHINF(IARR,1).EQ.0)GO TO 172
    IF(CHINF(IARR,3).NE.0)GO TO 175
    WRITE(1,211)
  211 FORMAT(" If you want only a sub-range transformed, enter the beginning",/
    & , " and ending points, otherwise hit return "$
    READ(INDEV,212) IBEG,IEND
  212 FORMAT(2I0)
    IF((IEND.LE.0).OR.(IBEG.LE.0))GO TO 96
C
C Move sub-range up to front of array, set rest of the array to the mean
C value of the subrange, convert to complex and go
C
    DO 951 I=1,IEND-IBEG+1
      A(I,IARR)=A(I+IBEG-1,IARR)
  951 CONTINUE
    I=NSGOOD
    NSGOOD=IEND-IBEG+1
    CALL COMSTS(CHINF(IARR,1),IARR)
    MEAN=ARRSTS(3,IARR)
    NSGOOD=I
    DO 952 I= IEND-IBEG+1,1,-1
      B(I,IARR)=CMLPX(A(I,IARR)-MEAN,0.0)
  952 CONTINUE
    DO 953 I=IEND-IBEG+2,NSATT
      B(I,IARR)=(0.0,0.0)
  953 CONTINUE
    CHINF(IARR,3)=1
    NB=MIN0(NSATT,2048)
    GO TO 100
  96 WRITE(1,219)
  219 FORMAT(" Hit <CR> for a full FFT, or n (any power of 2) for every ",
    & "nth point "$
    READ(INDEV,218)CHINF(IARR,3)
  218 FORMAT(I0)
    IF(CHINF(IARR,3).LE.0)CHINF(IARR,3)=1
    IF(POWOF2(CHINF(IARR,3)))GO TO 965
    WRITE(1,217)CHINF(IARR,3)
  217 FORMAT(1X,I5," is not a power of 2, please try again")
    GO TO 96
C
C Remove mean from array and change to complex format before doing transform
C
  965 WRITE(1,220)ANS

```

```

220 FORMAT(" Removing mean from array ",A0)
    MEAN=ARRSTS(3,IARR)
    NB=MIN0(NSATT,2048)/CHINF(IARR,3)
    IF(CHINF(IARR,3).EQ.1)GO TO 98
    DO 97 I=1,NB
        B(I,IARR)=CMLPX( A(CHINF(IARR,3)*I,IARR)-MEAN , 0.0)
97 CONTINUE
    GO TO 100
98 DO 99 I=NB,1,-1
    B(I,IARR)=CMLPX(A(I,IARR)-MEAN , 0.0)
99 CONTINUE
100 WRITE(1,230)
230 FORMAT(" Computing transform...")
    CALL FFT(IARR,NB,-1)
    RETURN

```

C
C
C

```

Error messages
170 WRITE(1,240)ANS
240 FORMAT(" ***Array designator must be either A or B, please",
    & " try again")
    GO TO 95
172 WRITE(1,245)ANS
245 FORMAT(" ***Array ",A1," is empty")
    RETURN
175 WRITE(1,250)ANS
250 FORMAT(" ***Array ",A1," has already been transformed")
    RETURN
    END

```

C
C
C

LOGICAL FUNCTION POWOF2(I)

C
C
C

```

This function determines returns true if I is a power of 2
N = I
POWOF2= .TRUE.
IF(N.EQ.0) RETURN
101 IF(N.LE.1) RETURN
    IF(MOD(N,2).NE.0)GO TO 199
    N=N/2
    GO TO 101
199 POWOF2= .FALSE.
    RETURN
    END

```

C
C
C

```

SUBROUTINE SCALE
COMMON/CONTRL/NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME(5,2),
& ELTIME(3),SMSTAT(5,80),STCOMP,CHINF(2,3),TITLE(80),FILNAM,
& IL(81),FILE2,HERTZ,ARRSTS(5,2),INDEV
INTEGER NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME,ELTIME,CHINF
REAL SMSTAT

```

```
LOGICAL STCOMP
CHARACTER*14 FILNAM,FILE2
CHARACTER*8 TITLE
COMMON A(4098,2)
CHARACTER*1 ANS
DOUBLE PRECISION DBLE,SCFACT
```

```
C
C THIS ROUTINE SCALES A CHANNEL IN ARRAY A BY ITS SCALE FACTOR.
C THE SUMMARY STATS IN ARRSTS ARE CHANGED AND SO THE MIN, MAX
C MEAN AND STANDARD DEVIATION NEED NOT BE SCALED FOR ANY FUTURE USE.
C
```

```
101 WRITE(1,200)
200 FORMAT(" Which array do you want to scale (A/B)? "$
      READ(INDEV,210) ANS
210 FORMAT(A0)
      IARR=0
      IF((ANS.EQ."A").OR.(ANS.EQ."a")) IARR=1
      IF((ANS.EQ."B").OR.(ANS.EQ."b")) IARR=2
      IF(IARR.EQ.0) GO TO 170
```

```
C
C Check to see if the array has already been scaled or is empty
C
```

```
      IF(CHINF(IARR,2).EQ.1) GO TO 175
      IF(CHINF(IARR,1).EQ.0) GO TO 180
      WRITE(1,215) ANS
215 FORMAT(" Scaling array ",A0,"...")
      SCFACT=DBLE(ARRSTS(5,IARR))
      DO 105 I=1,NSATT
          A(I,IARR)=SINGL(SCFACT*DBLE(A(I,IARR)))
105 CONTINUE
      CHINF(IARR,2)=1
```

```
C
C Scale the summary statistics
C
```

```
      DO 106 I=1,4
          ARRSTS(I,IARR)=ARRSTS(I,IARR)*ARRSTS(5,IARR)
106 CONTINUE
      RETURN
```

```
C
C Error messages
C
```

```
170 WRITE(1,220)
220 FORMAT(" ***Array designator must be either A or B, please try again")
      GO TO 101
175 WRITE(1,230) ANS
230 FORMAT(" ***Array ",A0," has already been scaled")
      RETURN
180 WRITE(1,240) ANS
240 FORMAT(" ***Array ",A0," is empty")
      RETURN
      END
```

```
C
C-----
C
```

```
SUBROUTINE FFTOUT
```

```

COMMON/CONTRL/NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME(5,2),
& ELTIME(3),SMSTAT(5,80),STCOMP,CHINF(2,3),TITLE(80),FILNAM,
& IL(81),FILE2,HERTZ,ARRSTS(5,2),INDEV
INTEGER NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME,ELTIME,CHINF
REAL SMSTAT
LOGICAL STCOMP
CHARACTER*14 FILNAM,FILE2,UPCASE
CHARACTER*8 TITLE
COMMON A(4098,2)
COMPLEX B(2049,2)
EQUIVALENCE (A,B)
CHARACTER*1 ANS

```

```

C
C THIS ROUTINE ALLOWS THE USER TO CREATE A DISK FILE CONTAINING FFT
C COEFFICIENTS IN RAW BINARY FORM
C

```

```

101 WRITE(1,200)
200 FORMAT(" Which array of coefficients do you want to output? (A/B) "$
READ(INDEV,210) ANS
210 FORMAT(A0)
IARR=ICARD(ANS)
IF((IARR.NE.1).AND.(IARR.NE.2))GO TO 180
IF(CHINF(IARR,3).EQ.0)GO TO 182
WRITE(1,220)
220 FORMAT(" On what file? "$
READ(INDEV,210) FILE2
FILE2=UPCASE(FILE2)
IF(IOLOOK(0,FILE2))GO TO 184
103 IF(IOWRIT(11,0,0,FILE2))GO TO 186
WRITE(1,230) FILE2
230 FORMAT(" Writing data to ",A0,"...")
WRITE(11) (B(I,IARR),I=1,NSATT/(2*CHINF(IARR,3)))
IF(IOCLOS(11))GO TO 188
RETURN

```

```

C
C ERROR HANDLING CODE
C

```

```

180 WRITE(1,240)
240 FORMAT(" ***Array designator must be either A or B, please try again")
GO TO 101
182 WRITE(1,245) ANS
245 FORMAT(" ***Array ",A0," has not been transformed")
RETURN
184 WRITE(1,250) FILE2
250 FORMAT(" *** ",A0," already exists, do you want it deleted? (Y/N) "$
READ(INDEV,210) ANS
IF((ANS.EQ."Y").OR.(ANS.EQ."y"))GO TO 103
RETURN
186 WRITE(1,260) FILE2
260 FORMAT(" ***Unable to open ",A0)
RETURN
188 WRITE(1,270) FILE2
270 FORMAT(" ***Unable to close "A0)
RETURN
END

```


C
C
C

```
      SUBROUTINE CROSPC
      COMMON/CONTRL/NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME(5,2),
&     ELTIME(3),SMSTAT(5,80),STCOMP,CHINF(2,3),TITLE(80),FILNAM,
&     IL(81),FILE2,HERTZ,ARRSTS(5,2),INDEV
      INTEGER NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME,ELTIME,CHINF
      REAL SMSTAT
      LOGICAL STCOMP
      CHARACTER*14 FILNAM,FILE2,UPCASE,FILE3
      CHARACTER*8 TITLE
      COMMON A(4098,2)
      CHARACTER*1 ANS
      COMPLEX B(2049,2)
      EQUIVALENCE (A,B)
      COMPLEX CRSPEC,CONJG,CMLPX,CTEMP(32),AVG
      REAL PHASE,COHERE
      DOUBLE PRECISION DBLE,DSQRT
```

C
C
C
C

COMPUTES CROSS SPECTRAL JUNK WITH THE TWO ARRAYS OF FFT COEFFICIENTS IN B

```
      IF((CHINF(1,3).EQ.0).OR.(CHINF(2,3).EQ.0).OR.
&     (CHINF(1,3).NE.CHINF(2,3)))GO TO 170
101 WRITE(1,200)
200 FORMAT(" Enter an averaging window size (hit <CR> for 16) "$
      READ(INDEV,210)NWIN
210 FORMAT(I0)
      IF(NWIN.EQ.0)NWIN=16
      IF(NWIN.LT.8)GO TO 172
      IF(NWIN.GT.32)GO TO 101
C NFREQ == # OF FREQUENCIES AT WHICH COMPUTATION WILL TAKE PLACE
      NFREQ=NSATT/(2*NWIN)
102 WRITE(1,225)
225 FORMAT(" What file do you want the data written to? "$
104 READ(INDEV,230)FILE2
230 FORMAT(A0)
      FILE2=UPCASE(FILE2)
      IF(IOLOOK(0,FILE2))GO TO 174
105 IF(IOWRIT(12,2,0,FILE2))GO TO 176
115 DELTAF=(HERTZ/FLOAT(NSATT))*FLOAT(NWIN)
      WRITE(1,220)
220 FORMAT(" Computing cross-spectral phase and coherency...")
      WRITE(12,240)1,CHINF(1,1)," X spec ",NFREQ
240 FORMAT(1X,I1,I4,A7,3X,I5)
      WRITE(12,250)CHINF(2,1)
250 FORMAT(1X,I5)
      DO 109 I=1,NFREQ
          J=(I-1)*NWIN + 1
          FREQ=DELTAF*I
          S1=AVGMAG(B(J,1),NWIN)
          S2=AVGMAG(B(J,2),NWIN)
          DO 107 K=1,NWIN
              CTEMP(K)=B(J+K-1,1)*CONJG(B(J+K-1,2))
```

```

107 CONTINUE
   CRSPEC=AVG (CTEMP,NWIN)
   PHASE=CATAN (AIMAG (CRSPEC) ,REAL (CRSPEC) )
   COHERE=SNGL (DBLE (REAL (CRSPEC) ) *DBLE (REAL (CRSPEC) ) +
&     DBLE (AIMAG (CRSPEC) ) *DBLE (AIMAG (CRSPEC) )) / (S1*S2)
   WRITE (12,260) FREQ,PHASE,COHERE
260  FORMAT (1X,F9.4,1X,2G15.8)
109 CONTINUE
   IF (IOCLOS (12)) GO TO 178
   RETURN

```

```

C
C ERROR HANDLING CODE
C

```

```

170 WRITE (1,280)
280 FORMAT (" *** Both arrays must be identically transformed first")
   RETURN
172 WRITE (1,290)
290 FORMAT (" *** Window sizes less than 8 not supported, please try again")
   GO TO 101
174 WRITE (1,300) FILE2
300 FORMAT (" *** ",A0," already exists, do you want it deleted? (Y/N)"$
   READ (INDEV,230) ANS
   IF ((ANS.EQ."Y").OR.(ANS.EQ."y")) GO TO 105
   GO TO 109
176 WRITE (1,310) FILE2
310 FORMAT (" ***Unable to open ",A0)
   GO TO 102
   RETURN
178 WRITE (1,320) FILE2
320 FORMAT (" ***Unable to close ",A0)
   RETURN
   END

```

```

C
C -----

```

```

C
C FUNCTION CATAN(Y,X)
C
C ATAN2 FUNCTION THAT RETURNS AN ANGLE IN DEGREES AND DOES NOT DIE
C ON AN INFINITE TANGENT
C

```

```

   CATAN=0.0
   IF ((X.EQ.0.0) .AND. (Y.GE.0.0)) CATAN=90.
   IF ((X.EQ.0.0) .AND. (Y.LT.0.0)) CATAN=-90.
   IF (CATAN.EQ.0.0) CATAN=ATAN2 (-Y,X) * 57.29577951
   RETURN
   END

```

```

C
C -----

```

```

C
C COMPLEX FUNCTION AVG (ARR,N)
C
C AVG RETURNS THE AVERAGE OF ELEMENTS 1 THROUGH N OF THE COMPLEX ARRAY ARR
C
C
C COMPLEX ARR (N) ,CMLPX
C AVG=CMLPX (0.0,0.0)

```

```

DO 101 I=1,N
  AVG=AVG+ARR(I)
101 CONTINUE
  AVG=CMPLX (REAL (AVG) /FLOAT (N) ,AIMAG (AVG) /FLOAT (N) )
  RETURN
END

```

C

C-----

C

```

FUNCTION AVGMAG (ARR,N)

```

C

```

C AVGMAG RETURNS THE AVERAGE OF THE SQUARES OF THE MAGNITUDES
C OF ELEMENTS 1 THROUGH N OF THE COMPLEX ARRAY ARR
C

```

C

```

  COMPLEX ARR(N)
  DOUBLE PRECISION DBLE,DSUM,DR,DI,DSQRT
  DSUM=0.D0
  DO 101 I=1,N
    DR=DBLE (REAL (ARR (I) ))
    DI=DBLE (AIMAG (ARR (I) ))
    DSUM=DSUM + DR*DR + DI*DI
101 CONTINUE
  AVGMAG=SNGL (DSSUM/DBLE (FLOAT (N) ))
  RETURN
END

```

C

C-----

C

```

SUBROUTINE CLIP

```

C

```

C This routine "clips" a data channel, setting all data values that are
C outside of a given range to the appropriate value on the straight line
C between the two good values on either side of the clipped range.
C

```

C

```

  COMMON/CONTRL/NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME(5,2),
&  ELTIME(3),SMSTAT(5,80),STCOMP,CHINF(2,3),TITLE(80),FILNAM,
&  IL(81),FILE2,HERTZ,ARRSTS(5,2),INDEV
  INTEGER NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME,ELTIME,CHINF
  REAL SMSTAT
  LOGICAL STCOMP
  CHARACTER*14 FILNAM,FILE2
  CHARACTER*8 TITLE
  COMMON A(4098,2)
  CHARACTER*1 ANS
  REAL MAXVAL,MINVAL
101 WRITE(1,200)
200 FORMAT(" which array do you want to clip? (A/B) "$
  READ(INDEV,210) ANS
210 FORMAT(A0)
  IARR=ICARD(ANS)
  IF((IARR.NE.1).AND.(IARR.NE.2))GO TO 180
  IF(CHINF(IARR,1).EQ.0)GO TO 181
  IF(CHINF(IARR,2).EQ.0)GO TO 182
  IF(CHINF(IARR,3).NE.0)GO TO 184
  WRITE(1,220)

```

```

220 FORMAT(" Enter the maximum allowable value (<CR> for no clipping) "$
      READ(INDEV,230)MAXVAL
230 FORMAT(F0.0)
      IF(MAXVAL.EQ.0.0)MAXVAL=1.0E30
      WRITE(1,240)
240 FORMAT(" Enter the minimum allowable value (<CR> for no clipping) "$
      READ(INDEV,230)MINVAL
      IF(MINVAL .EQ. 0.) MINVAL = -1.0E30
      WRITE(1,250)CHINF(IARR,1),MINVAL,MAXVAL
250 FORMAT(" Clipping values in channel ",I2," outside ("F7.4,""
      & F7.4,") OK? (Y/N)"$
      READ(INDEV,210)ANS
      IF((ANS.NE."Y").AND.(ANS.NE."y"))RETURN
C
C Clip the first point as a special case
C
      IF(A(1,IARR).GT.MAXVAL)A(1,IARR)=AMIN1(MAXVAL,A(2,IARR))
      IF(A(1,IARR).LT.MINVAL)A(1,IARR)=AMAX1(MINVAL,A(2,IARR))
C
C Now clip the rest of the time series
C
      WRITE(1,131) NSATT
131  FORMAT(" Series contains ",I4," points")
      DO 105 I = 2, NSATT
          A1      = A(I,IARR)
          A2      = A(I+1, IARR)
C
C Test to see if only a single, isolated value is wild
C
          IFLAG   = 0
          IF((A1 .GT. MAXVAL) .AND. (A2 .GT. MAXVAL)) IFLAG = 1
          IF((A1 .LT. MINVAL) .AND. (A2 .LT. MINVAL)) IFLAG = 2
          IF(IFLAG .NE. 0) GOTO 108
C
C If only single value is wild then set it to the average of its neighbors
C and loop back to the next point
C
          IF(A(I,IARR).GT.MAXVAL)A(I,IARR)=AMIN1(MAXVAL,
      & (A(I-1,IARR)+A(I+1,IARR))/2.)
          IF(A(I,IARR).LT.MINVAL)A(I,IARR)=AMAX1(MINVAL,
      & (A(I-1,IARR)+A(I+1,IARR))/2.)
          GOTO 105
C
C We now know that multiple sequential points are bad, so first search
C for the next good point and put its subscript in K.
C
108      K = I
109      K = K+1
          IF(K .GT. NSATT) GOTO 105
          IF(IFLAG .EQ. 2) GOTO 110
          IF(A(K,IARR) .GT. MAXVAL) GOTO 109
          GOTO 111
110      IF(A(K,IARR) .LT. MINVAL) GOTO 109
C
C Now determine the line between the two good points. Actually determine

```

C the increment that needs to be added to each successive point in order
C to form a straight line and store this in DEL.

C
111 ALOW = A(I-1,IARR)
 AUP = A(K,IARR)
 DEL = (AUP - ALOW)/FLOAT(K-I+1)
 WRITE(1,132)I,K
132 FORMAT(' Clipping pts. ',I4,' to ',I4)

C
C Now do the actual clipping, assigning the spurious points the
C calculated straight line interpolation.

C
 DO 116 J = I,K
 A(J,IARR) = ALOW + DEL*FLOAT(J-I+1)
116 CONTINUE
 I = K

105 CONTINUE

C
C Clipping all done now, so compute statistics and go home

C
 CALL COMSTS(CHINF(IARR,1),IARR)
 RETURN

C
C Error messages

C
180 WRITE(1,280)
280 FORMAT(" ***Array designator must be either A or B, please try again")
 GO TO 101
181 WRITE(1,285)ANS
285 FORMAT(" *** Array ",A0," is empty")
 RETURN
182 WRITE(1,290)ANS
290 FORMAT(" ***Array ",A0," has not been scaled")
 RETURN
184 WRITE(1,300)ANS
300 FORMAT(" ***Array ",A0," contains raw Fourier coefficients")
 RETURN
 END

C
C-----
C

 SUBROUTINE READCH(ICHAN,IARR)
 COMMON/CONTRL/NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME(5,2),
& ELTIME(3),SMSTAT(5,80),STCOMP,CHINF(2,3),TITLE(80),FILNAM,
& IL(81),FILE2,HERTZ,ARRSTS(5,2),INDEV
 INTEGER NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME,ELTIME,CHINF
 REAL SMSTAT
 LOGICAL STCOMP
 CHARACTER*14 FILNAM,FILE2
 CHARACTER*8 TITLE
 COMMON A(4098,2)

C
C READCH reads channel ICHAN into array IARR

C
 READ(5/ICHAN) ITEMP

```

A(1,IARR)=FLOAT(ITEMP)
DO 101 I=2,NSGOOD
  READ(5)ITEMP
  A(I,IARR)=FLOAT(ITEMP)
101 CONTINUE
RETURN
END

```

C
C
C

```

FUNCTION ICARD(CHAR)

```

C
C
C

Computes numeric value of characters, e.g. 'A'='a'=1, 'B'='b'=2, etc.

```

INTEGER*1 TINT,KHAR
CHARACTER*1 CHAR
TINT=KHAR(CHAR,1) - KHAR("A",1) + 1
IF(TINT.GT.26) TINT=KHAR(CHAR,1) - KHAR("a",1) + 1
ICARD=TINT
RETURN
END

```

C
C
C

```

SUBROUTINE DETRND
COMMON/CONTRL/NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME(5,2),
& ELTIME(3),SMSTAT(5,80),STCOMP,CHINF(2,3),TITLE(80),FILNAM,
& IL(81),FILE2,HERTZ,ARRSTS(5,2),INDEV
INTEGER NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME,ELTIME,CHINF
REAL SMSTAT
LOGICAL STCOMP
CHARACTER*14 FILNAM,FILE2
CHARACTER*8 TITLE
COMMON A(4098,2)
CHARACTER*1 ANS
REAL MEAN

```

C
C
C
C

This subroutine computes and subtracts from the series in A either the series mean or the least squares straight line

```

101 WRITE(1,200)
200 FORMAT(" Which array do you want to detrend? (A/B) "$
  READ(INDEV,210)ANS
210 FORMAT(A0)
  IARR=ICARD(ANS)
  IF((IARR.NE.1).AND.(IARR.NE.2))GO TO 180
  IF((CHINF(IARR,1).EQ.0).OR.(CHINF(IARR,3).NE.0))GO TO 182
  WRITE(1,220)
220 FORMAT(" Enter 0 to remove the mean or 1 to remove the least ",
& "squares straight line "$
  READ(INDEV,230)K
230 FORMAT(I0)
  IF(K.EQ.1)GO TO 105
  IF(K.NE.0)GO TO 184

```

C

```

C MEAN REMOVAL CODE
C
  DO 103 I=1,NSATT
    A(I,IARR)=A(I,IARR)-ARRSTS(3,IARR)
103 CONTINUE
  CALL COMSTS(CHINF(IARR,1),IARR)
  RETURN
C
C Code for least squares line removal
C
105 SUMX=0.0
  SUMY=0.0
  SUMXY=0.0
  SUMX2=0.0
  SUMY2 = 0.0
  DO 107 I = 1, NSATT
    XI = .25 * FLOAT(I-1)
    YI = A(I,IARR)
    SUMX = SUMX + XI
    SUMY = SUMY + YI
    SUMXY=SUMXY + XI*YI
    SUMX2=SUMX2 + XI*XI
    SUMY2=SUMY2 + YI*YI
107 CONTINUE
C
  SXX = SUMX2 - SUMX * SUMX / FLOAT(NSATT)
  SXY = SUMXY - SUMX * SUMY / FLOAT(NSATT)
  SYX = SUMY2 - SUMY * SUMY / FLOAT(NSATT)
  BB = SXY / SXX
  AA = (SUMY - BB* SUMX) /FLOAT(NSATT)
C
  DO 109 I = 1, NSATT
    XI = .25*FLOAT(I-1)
    A(I,IARR) = A(I,IARR) - (AA + BB*XI)
109 CONTINUE
  CALL COMSTS(CHINF(IARR,1),IARR)
  RETURN
C
C Error processing
C
180 WRITE(1,240)
240 FORMAT(" ***Array designator must be either A or B, please try "
& "again")
  GO TO 101
182 WRITE(1,250)ANS
250 FORMAT(" ***Array ",A0," does not contain time series data")
  RETURN
184 WRITE(1,260)
260 FORMAT(" ***Detrending polynomial order must be either 0 or 1")
  RETURN
  END
C
C-----
C
SUBROUTINE FFT(IARR,NB,IFL)

```

```

COMMON B(2049,2)
COMPLEX B
COMPLEX U,W,T,CMLPX
C
C THIS ROUTINE IS TAKEN FROM "AN ITRODUCTION TO RANDOM VIBRATIONS AND
C SPECTRAL ANALYSIS" BY D.E. NEWLAND, 1975.
C IFL = -1 MEANS FORWARD TRANSFORM, IFL=+1 MEANS REVERSE TRANSFORM
C
      IF(IFL.GT.0)GO TO 107
      T=CMLPX(FLOAT(NB),0.)
      DO 101 J=1,NB
        B(J,IARR)=B(J,IARR)/T
101  CONTINUE
107  N=INT(ALOG(FLOAT(NB))/ALOG(2.) + .05)
      NBD2=NB/2
      NBM1=NB-1
      J=1
      DO 104 L=1,NBM1
        IF(L.GE.J)GO TO 102
        T=B(J,IARR)
        B(J,IARR)=B(L,IARR)
        B(L,IARR)=T
102  K=NBD2
103  IF(K.GE.J)GO TO 104
        J=J-K
        K=K/2
        GO TO 103
104  J=J+K
      PI=-SIGN(2.,FLOAT(IFL))*1.570796327
      DO 106 M=1,N
        U=(1.0,0.0)
        ME=2**M
        K=ME/2
        W=CMLPX(COS(PI/FLOAT(K)), -SIN(PI/FLOAT(K)))
        DO 105 J=1,K
          DO 105 L=J,NB,ME
            LPK=L+K
            T=B(LPK,IARR)*U
            B(LPK,IARR)=B(L,IARR)-T
            B(L,IARR)=B(L,IARR)+T
105  CONTINUE
106  U=U*W
      RETURN
      END
C
C -----
C
SUBROUTINE FILTER
COMMON/CONTRL/NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME(5,2),
&  ELTIME(3),SMSTAT(5,80),STCOMP,CHINF(2,3),TITLE(80),FILNAM,
&  IL(81),FILE2,HERTZ,ARRSTS(5,2),INDEV
INTEGER NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME,ELTIME,CHINF
REAL SMSTAT
LOGICAL STCOMP
CHARACTER*14 FILNAM,FILE2

```



```

CHARACTER*8 TITLE
COMMON A(4098,2)
COMPLEX B(2049,2),CMLX
EQUIVALENCE (A,B)
CHARACTER*1 ANS
REAL MEAN

C
C This subroutine performs an FFT band pass filter on an array of time
C series data
C
C Read in the array designator
C
WRITE(1,200)
200 FORMAT(" Which array do you want filtered? (A/B)"$)
READ(INDEV,210) ANS
210 FORMAT(A0)
IARR=ICARD(ANS)
IF((IARR.NE.1).AND.(IARR.NE.2))GOTO 180
IF(CHINF(IARR,1).EQ.0)GO TO 184

C
C Input cutoff frequencies
C
WRITE(1,220)
220 FORMAT(" Enter lower and upper cutoff frequencies "$)
READ(INDEV,230) FLO,FHI
230 FORMAT(2F0.0)
IF(FHI.LE.FLO)GO TO 182
IF(FLO.LT.0.0)GO TO 182
IF(FHI.GT.(HERTZ/2.))FHI=HERTZ/2.

C
C Convert data to complex and do FFT (Return with error if
C the array has already been transformed)
C
IF(CHINF(IARR,3).NE.0)GO TO 184
CHINF(IARR,3)=1
NB=MIN0(NSATT,2048)
WRITE(1,232)
232 FORMAT(" Transforming data...")
MEAN=ARRSTS(3,IARR)
105 DO 106 I=NB,1,-1
B(I,IARR)=CMLX(A(I,IARR)-MEAN,0.0)
106 CONTINUE
CALL FFT(IARR,NB,-1)

C
C Filter transformed data
C
109 DELTAF=HERTZ/FLOAT(NB)
WRITE(1,234)
234 FORMAT(" Filtering data...")
N1= 1 + IFIX(FLO/DELTAF)
N2= 1 + IFIX(FHI/DELTAF)
IF(N1.EQ.1)GO TO 113
DO 111 I=1,N1
B(I,IARR)=(0.0,0.0)
K=NB + 2 - I

```

```

        B(K,IARR)=(0.0,0.0)
111 CONTINUE
113 IF(N2.EQ.1)GO TO 117
    DO 115 I=N2,NB/2
        B(I,IARR)=(0.0,0.0)
        K=NB + 2 - I
        B(K,IARR)=(0.0,0.0)
115 CONTINUE
C
C Transform data back to time domain
C
117 WRITE(1,236)
236 FORMAT(" Retransforming data...")
    CALL FFT(IARR,NB,1)
    CHINF(IARR,3)=0
C
C Convert filtered time series data back to real form
C
    DO 119 I=1,NB
        A(I,IARR)=REAL(B(I,IARR))
119 CONTINUE
C
C Compute summary statistics on filtered data
C
    CALL COMSTS(CHINF(IARR,1),IARR)
    RETURN
C
C Error messages
C
180 WRITE(1,240)
240 FORMAT(" ***Array designator must be either A or B")
    RETURN
182 WRITE(1,250)
250 FORMAT(" ***Bad cutoff frequency")
    RETURN
184 WRITE(1,260)ANS
260 FORMAT(" ***Array ",A1," does not contain time series data")
    RETURN
    END
C
C -----
C
SUBROUTINE UPDATE
COMMON/CONTRL/NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME(5,2),
& ELTIME(3),SMSTAT(5,80),STCOMP,CHINF(2,3),TITLE(80),FILNAM,
& IL(81),FILE2,HERTZ,ARRSTS(5,2),INDEV
INTEGER NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME,ELTIME,CHINF
REAL SMSTAT
LOGICAL STCOMP
CHARACTER*14 FILNAM,FILE2
CHARACTER*8 TITLE
COMMON A(4098,2)
CHARACTER*1 ANS
C
C This subroutine allows the user to take a channel that he has filtered,

```

C clipped, etc. and write it to the master data file. This is the only
C place in the program that the .DAT file is written to and caution is
C highly recommened

C

```
101 WRITE(1,200)
200 FORMAT(" Which array do you want to write? (A/B) "$
      READ(INDEV,210)ANS
210 FORMAT(A0)
      IARR=ICARD(ANS)
      IF((IARR.NE.1).AND.(IARR.NE.2))GOTO 180
```

C

C Test for empty or transformed array

C

```
      IF((CHINF(IARR,1).EQ.0).OR.(CHINF(IARR,3).NE.0))GO TO 184
      SCFAC=1.0
      IF(CHINF(IARR,2).NE.0)SCFAC=ARRSTS(5,IARR)
```

C

C Write UNSCALED data on to master file

C

```
      WRITE(1,220)
220  FORMAT('/ Writing data to master file ')
      WRITE(5/CHINF(IARR,1))(IFIX(A(I,IARR)/SCFAC),I=1,NSATT)
      STCOMP=.FALSE.
      CALL SETLEN(FILNAM,IL(81))
      CALL CONCAT(FILE2,FILNAM,"STS")
      IF(IODEL(0,FILE2))GO TO 186
      WRITE(1,230)
230  FORMAT(" Data file rewritten and statistics file deleted")
      RETURN
```

C

C Error messages

C

```
180 WRITE(1,240)
240 FORMAT(" ***Array designator must be either A or B")
      RETURN
184 WRITE(1,250)ANS
      RETURN
186 WRITE(1,260)FILE2
260 FORMAT(" ***Please delete ",A0,", it is now outdated")
250 FORMAT(" ***Array ",A1," does not contain time series data")
      RETURN
      END
```

C

C

C

```
      SUBROUTINE MACRO
      COMMON/CONTRL/NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME(5,2),
&  ELTIME(3),SMSTAT(5,80),STCOMP,CHINF(2,3),TITLE(80),FILNAM,
&  IL(81),FILE2,HERTZ,ARRSTS(5,2),INDEV
      COMMON/NOLOT/PLOTP
      INTEGER NSATT,NSGOOD,TAPENO,RECNO,TSNO,TPTRAK,TIME,ELTIME,CHINF
      REAL SMSTAT
      LOGICAL STCOMP
      CHARACTER*14 FILNAM,FILE2,UPCASE
      CHARACTER*8 TITLE
```

LOGICAL IOLOOK,IOREAD,IOCLOS,PLOTP
CHARACTER*1 ANS

C
C This routine changes where SSP gets its commands from from the terminal
C to device 9 and back again. This allows the construction of "macro
C files" that contain SSP menu choices. Note that the macro file must
C end with either l3 (to end SSP) or ll<cr>ZZZZ (to return control to
C the terminal. If the macro file does not end in one of these two ways,
C then the program will hang and ignore keyboard input.
C

PLOTP=.FALSE.
WRITE(1,200)
200 FORMAT(" Enter the name of the macro command file ('ZZZZ' to quit)"\$
READ(INDEV,210)FILE2
210 FORMAT(A0)
WRITE(1,211)
211 FORMAT(" ")

C
C Check (in order) for command to return control to terminal, for
C non-existent file and for impossible to open file.
C

FILE2=UPCASE(FILE2)
IF(FILE2.EQ."ZZZZ")GO TO 185
IF(.NOT.IOLOOK(0,FILE2))GO TO 180
IF(IOREAD(9,2,0,FILE2))GO TO 190

C
C If file was successfully opened, then redirect input to device 9
C

INDEV=9
WRITE(1,212)
212 FORMAT(" Suppress plotting of single-valued channels? (Y/N) "\$
READ(1,210)ANS
IF((ANS.EQ."Y").OR.(ANS.EQ."y"))PLOTP=.TRUE.
RETURN

C
C Error messages
C

180 WRITE(1,220)FILE2
220 FORMAT(" *** ",A0," not found")

C
C Ring bell to wake up the user and redesignate terminal as input.
C

185 WRITE(1)7
WRITE(1,230)
230 FORMAT(" Control returned to terminal")
INDEV=1
I=IOCLOS(9)
RETURN
190 WRITE(1,240)FILE2
240 FORMAT(" ***Unable to open ",A0," for input")
INDEV=1
I=IOCLOS(9)
RETURN
END

PLOTR 2.1

```

C
C PROGRAM PLOTR TO DO PLOTS OF DATA FROM PROGRAM SSP
C Version 2.1      3-28-84
C Written in Microsoft F80 FORTRAN and linked with Pacific Basin
C Graphics' PBG 100 supposedly CalComp compatible plotter subroutines

```

```

C
C This program is set up to read plotter files from SSP. The
C information expected in the header record is:

```

```

C   Col.      What
C   1         Code for row, FFT or phase plot
C   2-5       Channel number (I4)
C   6-13      Channel label (A8)
C   16-20     Number of data points (I5)

```

```

C N.B. For phase and coherency plots, the second record does not
C contain data but instead contains the second channel number. (I5)
C Also, the label field is ignored.

```

Installation for different plotters

```

C Right now this program is set up for a C.Itoh CX-4800 plotter.
C To run it with a HP 7470A plotter, change CALL PLOTS('CX4800') to
C CALL PLOTS('HP7470') and remove the CALL FACTOR(.7) that appear
C in each section. (Argument is probably not exactly .7, you can
C adjust it to suit you taste and paper size.

```

```

C
C REAL XARR(2050),YARR(2050)
C INTEGER*1 FILNAM(16),TITLE(16),ANS,TEMP(3)
C COMMON FILNAM,TITLE,XARR,YARR,ICHAN,NPNT
C DATA TITLE /16*1H /
C WRITE(3,199)
C 199 FORMAT(' PLOTR v. 2.1')
C CALL PLOTS('CX4800')

```

```

C Prompt for filename and convert it to F80 format

```

```

C
C 305 WRITE(3,300)
C 300 FORMAT('/ Enter:/' <1> Open a new file/' <2> Finish this page
C % (a plot should already have been made) '/' <cr> to quit '/')
C READ(1,301) IANS
C 301 FORMAT(I1)
C IF(IANS .EQ. 0) GOTO 310
C IF(IANS .EQ. 1) GOTO 303
C IF(IANS .EQ. 2) CALL PLOT(0.,0.,-999)
C GOTO 305
C 303 WRITE(3,200)
C 200 FORMAT(' What is the data file name?')
C READ(1,210) (FILNAM(I),I=1,16)
C 210 FORMAT(16A1)

```

```

C Here starts the long and laborious process of converting a filename
C with a dot extender into a file name padded with spaces as is required

```

```

C by F80 FORTRAN.
C
C Find the dot
  K=0
  DO 101 I=1,13
    IF (FILNAM(I).EQ.46)K=I
101 CONTINUE
C If dot was not found, jump to 170, otherwise store the extender in TEMP
  IF (K.EQ.0)GO TO 170
  DO 103 I=1,3
    J=I+K
    TEMP(I)=FILNAM(J)
103 CONTINUE
C Blank FILNAM from the dot to the end
  DO 105 I=K,13
    FILNAM(I)=' '
105 CONTINUE
C Insert extender into proper spot in FILNAM
  DO 106 I=9,11
    FILNAM(I)=TEMP(I-8)
106 CONTINUE
C Make sure that everything is upper case
  DO 107 I=1,11
    IF (FILNAM(I).GT.91)FILNAM(I)=FILNAM(I) - 32
107 CONTINUE
C
C Open file and read first record to determine proper kind of plot
C   IFLAG = 0 -> row plot
C   IFLAG = 1 -> combined phase and coherency plot
C   IFLAG = 2 -> FFT plot
C
  CALL OPEN(5,FILNAM,0)
  WRITE(3,215) (FILNAM(I),I=1,13)
215 FORMAT(' Opening ',13A1)
  WRITE(3,220)
220 FORMAT(' Reading data...')
  READ(5,230) IFLAG, ICHAN, (TITLE(I),I=9,16), NPNT
230 FORMAT(I1,I4,8A1,2X,I5)
  IF (IFLAG.EQ.0)CALL ROWPLT
  IF (IFLAG.EQ.1)CALL PHSPLT
  IF (IFLAG.EQ.2)CALL FFTPLT
  ENDFILE 5
  GOTO 305
170 WRITE(3,290) (FILNAM(I),I=1,13)
290 FORMAT(' No "." in ',13A1)
310 STOP
  END
C
C-----
C
C SUBROUTINE ROWPLT
  REAL XARR(2050),YARR(2050)
  INTEGER*1 FILNAM(16),TITLE(16),ANS
  COMMON FILNAM,TITLE,XARR,YARR,ICHAN,NPNT
C

```

```

C Subroutine to do row plots lengthwise on paper
C
WRITE(3,200) ICHAN, (TITLE(I), I=9,16), NPNT
200 FORMAT(' Channel ', I2, 2X, 8A1, 3X, I5, ' points')
SUM = 0.
DO 101 I=1, NPNT
    READ(5,210) XARR(I), YARR(I)
210    FORMAT(F10.4, F15.5)
    SUM = SUM + YARR(I)
101 CONTINUE
SUM = SUM/FLOAT(NPNT)
WRITE(3,511)
511 FORMAT(' Type <cr> to remove mean, else type <1> ')
READ(1,112) IANS
112    FORMAT(I1)
    IF(IANS .EQ. 1) GOTO 513
    DO 502 I = 1, NPNT
502    YARR(I) = YARR(I) - SUM
513    WRITE(3,220)
C
C Scale data
C
220 FORMAT(' Scaling data...')
    CALL SCALE(XARR, 9.0, NPNT, 1)
    CALL SCALE(YARR, 3.0, NPNT, 1)
C
C Prompt for plotting parameters
C
WRITE(3,230)
230 FORMAT(' Do you want a mean value line plotted? (Y/N)')
READ(1,240) ANS
240 FORMAT(A1)
    IMEAN=0
    IF((ANS.EQ.89).OR.(ANS.EQ.121)) IMEAN=1
109 WRITE(3,250)
250 FORMAT(' Should this be plotted on the <T>op or <B>ottom of',
    & ' the page?')
    READ(1,240) ANS
C
C Set offsets (in inches) with information from above
C
C 84='T' 116='t'
C 66='B' 98='b'
C
OFFSET=0.0
IF((ANS.EQ.84).OR.(ANS.EQ.116)) OFFSET=5.5
IF((ANS.EQ.66).OR.(ANS.EQ.98)) OFFSET=1.75
IF(OFFSET.EQ.0.0) GO TO 109
WRITE(3,260)
260 FORMAT(' Hit <return> when the plotter is ready')
READ(1,240) ANS
C
CALL FACTOR(.775)
C
CALL SYMBOL(.25, OFFSET+.375, 2, FILNAM, 90.0, 8)

```



```

TITLE(1)=1HC
TITLE(2)=1Hh
TITLE(3)=1H.
TITLE(5)=ICHAN/10 + 48
TITLE(6)=MOD(ICHAN,10) + 48
CALL AXIS(1.0,OFFSET,TITLE,16,3.0,90.0,YARR(NPNT+1),YARR(NPNT+2))
CALL AXIS(1.0,OFFSET,' ',1,9.0.,XARR(NPNT+1),XARR(NPNT+2))
CALL LINE(XARR,YARR,NPNT,1,0,0)
IF(IMEAN.EQ.0)GO TO 113

```

```

C
C Plot mean value line, if requested
C

```

```

YMEAN= SUM
YARR(1)=YMEAN
YARR(2)=YMEAN
YARR(3)=YARR(NPNT+1)
YARR(4)=YARR(NPNT+2)
XARR(2)=XARR(NPNT)
XARR(3)=XARR(NPNT+1)
XARR(4)=XARR(NPNT+2)
CALL LINE(XARR,YARR,2,1,0,0)
113 WRITE(3,280)
280 FORMAT(' Plotting finished')
CALL FACTOR(1.0)
RETURN
END

```

```

C
C -----
C

```

```

SUBROUTINE FFTPLT
REAL XARR(2050),YARR(2050)
INTEGER*1 FILNAM(16),TITLE(16),ANS,TEMP(3),HUNIT(6),VUNIT(20)
INTEGER*1 TEXT1(16),TEXT2(16),TEXT3(16)
COMMON FILNAM,TITLE,XARR,YARR,ICHAN,NPNT
DATA HUNIT/ 'H','e','r','t','z',' '/
WRITE(3,200) ICHAN,(TITLE(I),I=9,16),NPNT
200 FORMAT(' FFT of Channel ',I2,2X,8A1,3X,I5,' points')

```

```

C
C Input of data and calculation of variance (area under curve) and
C peak frequency
C

```

```

FMAX=0.
YSUM=0.
YMAX=0.
DO 101 I=1,NPNT
  READ(5,201) XARR(I),YARR(I)
201  FORMAT(F10.4,F15.5)
  YSUM=YSUM+YARR(I)
  IF(YARR(I).LE.YMAX)GO TO 101
  YMAX=YARR(I)
  FMAX=XARR(I)
101 CONTINUE
YMEAN=YSUM/FLOAT(NPNT)
VAR=YSUM*(XARR(2)-XARR(1))
Y1=5.

```

```

WRITE(3,203)VAR,FMAX,YMAX
203 FORMAT(' var.=',G15.8,', Fmax=',G15.8,', S(Fm)=',G15.8)
C
C Convert variance and max frequencies into character for plotting
C
DO 103 I=1,16
  TEXT1(I)=1H
  TEXT2(I)=1H
  TEXT3(I)=1H
103 CONTINUE
ENCODE(TEXT1,202)VAR
202 FORMAT(7X,G9.3)
ENCODE(TEXT2,202)YMAX
ENCODE(TEXT3,204)FMAX
204 FORMAT(7X,F7.4)
C
C Assign labels, one character at a time
C
TEXT1(1)=1Hv
TEXT1(2)=1Ha
TEXT1(3)=1Hr
TEXT1(4)=1H.
TEXT1(6)=1H=
TEXT2(1)=1HS
TEXT2(2)=1H(
TEXT2(3)=1HF
TEXT2(4)=1Hm
TEXT2(5)=1H)
TEXT2(6)=1H=
TEXT3(1)=1HF
TEXT3(2)=1Hm
TEXT3(3)=1Ha
TEXT3(4)=1Hx
TEXT3(6)=1H=
TEXT3(15)=1HH
TEXT3(16)=1Hz
FILNAM(9)=1H
FILNAM(10)=1HC
FILNAM(11)=1Hh
FILNAM(12)=1H.
FILNAM(13)=ICHAN/10 + 48
FILNAM(14)=MOD(ICHAN,10) + 48
C
C Prompt for plotting parameters
C
WRITE(3,210)
210 FORMAT(' Should this be plotted on the <T>op or <B>ottom of',
& ' the page?')
READ(1,220) ANS
220 FORMAT(30A1)
IF ((ANS .EQ. 66) .OR. (ANS .EQ. 98)) Y1=Y1+5.
DO 105 I=1,20
  VUNIT(I)=1H
105 CONTINUE
WRITE(3,230)

```

```

230 FORMAT(' Input vertical label ')
    READ(1,220) (VUNIT(I),I=1,20)
C
C Plot
C
    WRITE(3,240)
240 FORMAT(' Hit <return> when the plotter is ready')
    READ(1,250)ANS
250 FORMAT(A1)
    COORD=Y1-4.5
    CALL SYMBOL(COORD,5.,2,FILNAM,90.,16)
    CALL SYMBOL(COORD,6.75,2,TITLE(9),90.,8)
    COORD=COORD+.25
    CALL SYMBOL(COORD,5.,2,TEXT1,90.,16)
    COORD=COORD+.25
    CALL SYMBOL(COORD,5.,2,TEXT2,90.,16)
    COORD=COORD+.25
    CALL SYMBOL(COORD,5.,2,TEXT3,90.,16)
C Call to FACTOR forces fudging of Y1 in order to place plots properly
    CALL FACTOR(.7)
C
    IF(Y1.GT.5.5)Y1=Y1+.7
    CALL SCALE(XARR,5.,NPNT,1)
    CALL SCALE(YARR,4.,NPNT,-1)
    CALL AXIS(Y1,2.5,HUNIT,-6,5.,90.,XARR(NPNT+1),XARR(NPNT+2))
    CALL AXIS(Y1,2.5,VUNIT,20,4.0,180.0,YARR(NPNT+1),YARR(NPNT+2))
    CALL LINE(YARR,XARR,NPNT,1,0,0)
    CALL FACTOR(1.0)
    RETURN
    END
C
C -----
C
C SUBROUTINE PHSPLT
    REAL XARR(2050),YARR(1025,2)
    INTEGER*1 FILNAM(16),TITLE(16),ANS,TEMP(3)
    COMMON FILNAM,TITLE,XARR,YARR,ICHAN,NPNT
C
C Subroutine to plot phase and coherency
C
    READ(5,200) ICHAN2
200 FORMAT(I5)
    WRITE(3,210) ICHAN, ICHAN2, NPNT
210 FORMAT(' Phase and coherency of Chs.',I2,' and ',I2,I8,' points')
    NPNT=NPNT+2
    DO 101 I=3,NPNT
        READ(5,215)XARR(I),YARR(I,1),YARR(I,2)
215 FORMAT(F10.4,2F15.5)
101 CONTINUE
C
C Dummy values are put into the first two spots in all three arrays
C to force the plotting routines to plot between -100 and 300 degrees
C and between 0 and 1 on the two plots.
C
    XARR(1)=0.

```

```

XARR(2)=0.
YARR(1,1)=290.
YARR(2,1)=-90.
YARR(1,2)=.9999
YARR(2,2)=.0001
BOUND=-70.
DO 107 I=3, NPNT
    IF (YARR(I,1) .LT. BOUND) YARR(I,1)=YARR(I,1)+360.
107 CONTINUE
108 Y1=5.
    Y2=10.
C
C Create title, one character at a time
C
    TITLE(1)=lHC
    TITLE(2)=lHh
    TITLE(3)=lH.
    TITLE(4)=lH
    TITLE(5)=ICHAN/10 + 48
    TITLE(6)=MOD(ICHAN,10) + 48
    TITLE(7)=lH
    TITLE(8)=lH&
    TITLE(9)=lH
    TITLE(10)=ICHAN2/10 + 48
    TITLE(11)=MOD(ICHAN2,10) + 48
C
C Do actual plotting
C
    WRITE(3,240)
240 FORMAT(' Hit <return> when the plotter is ready')
    READ(1,250)ANS
250 FORMAT(A1)
    YTEMP=Y1-4.5
    CALL SYMBOL(YTEMP,3.,2,TITLE,90.,11)
C
    CALL FACTOR(.7)
C
    CALL SCALE(XARR,5.,NPNT,1)
    CALL SCALE(YARR(1,1),4.,NPNT,-1)
    CALL SCALE(YARR(1,2),4.,NPNT,-1)
    CALL AXIS(Y1,2.5,'Hertz',-5,5.,90.,XARR(NPNT+1),XARR(NPNT+2))
    CALL AXIS(Y1,2.5,'Phase in degrees',16,4.0,180.0,YARR(NPNT+1,1),
& YARR(NPNT+2,1))
    CALL LINE(YARR(1,1),XARR,NPNT,1,0,0)
C
    CALL FACTOR(.7)
C
    CALL AXIS(Y2,2.5,'Hertz',-5,5.,90.,XARR(NPNT+1),XARR(NPNT+2))
    CALL AXIS(Y2,2.5,'Coherency',9,4.0,180.0,YARR(NPNT+1,2),
& YARR(NPNT+2,2))
    CALL LINE(YARR(1,2),XARR,NPNT,1,0,0)
    CALL FACTOR(1.0)
    RETURN
    END

```

PLOTR 2.0

```

C
C PROGRAM PLOTR TO DO PLOTS OF DATA FROM PROGRAM SSP
C Version 2.0
C
      REAL XARR(2050),YARR(2050)
      INTEGER*1 FILNAM(16),TITLE(16),ANS,TEMP(3)
      COMMON FILNAM,TITLE,XARR,YARR,ICHAN,NPNT
      DATA TITLE /16*1H /
      WRITE(3,199)
199 FORMAT(' PLOTR ver.2.0')
C
C Prompt for filename and convert it to F80 format
C
      WRITE(3,200)
200 FORMAT(' What is the data file name?')
      READ(1,210) (FILNAM(I),I=1,16)
210 FORMAT(16A1)
      K=0
      DO 101 I=1,13
          IF (FILNAM(I).EQ.46)K=I
101 CONTINUE
      IF (K.EQ.0)GO TO 170
      DO 103 I=1,3
          J=I+K
          TEMP(I)=FILNAM(J)
103 CONTINUE
      DO 105 I=K,13
          FILNAM(I)=' '
105 CONTINUE
      DO 106 I=9,11
          FILNAM(I)=TEMP(I-8)
106 CONTINUE
C
C Open file and read first record to determine proper kind of plot
C IFLAG = 0 --> row plot
C IFLAG = 1 --> combined phase and coherency plot
C IFLAG = 2 --> FFT plot
C
      CALL OPEN(5,FILNAM,0)
      WRITE(3,215) (FILNAM(I),I=1,13)
215 FORMAT(' Opening ',13A1)
      WRITE(3,220)
220 FORMAT(' Reading data...')
      READ(5,230) IFLAG,ICHAN,(TITLE(I),I=9,16),NPNT
230 FORMAT(I1,I4,8A1,2X,I5)
      IF (IFLAG.EQ.0)CALL ROWPLT
      IF (IFLAG.EQ.1)CALL PHSPLT
      IF (IFLAG.EQ.2)CALL FFTPLT
      STOP
170 WRITE(3,290) (FILNAM(I),I=1,13)
290 FORMAT(' No "." in ',13A1)
      STOP
      END
      SUBROUTINE ROWPLT
      REAL XARR(2050),YARR(2050)

```

```

      INTEGER*1 FILNAM(16),TITLE(16),ANS
      COMMON FILNAM,TITLE,XARR,YARR,ICHAN,NPNT
C
C Subroutine to do row plots lengthwise on paper
C
      WRITE(3,200) ICHAN,(TITLE(I),I=9,16),NPNT
200  FORMAT(' Channel ',I2,2X,8A1,3X,I5,' points')
      DO 101 I=1,NPNT
          READ(5,210) XARR(I),YARR(I)
210  FORMAT(F10.4,F15.5)
101  CONTINUE
      WRITE(3,220)
C
C Scale data
C
220  FORMAT(' Scaling data...')
      CALL PLOTS('HP7470')
      CALL SCALE(XARR,9.0,NPNT,1)
      CALL SCALE(YARR,3.0,NPNT,1)
C
C Prompt for plotting parameters
C
      WRITE(3,230)
230  FORMAT(' Do you want a mean value line plotted? (Y/N)')
      READ(1,240) ANS
240  FORMAT(A1)
      IMEAN=0
      IF((ANS.EQ.89).OR.(ANS.EQ.121)) IMEAN=1
109  WRITE(3,250)
250  FORMAT(' Should this be plotted on the <T>op or <B>ottom of',
          & ' the page?')
      READ(1,240) ANS
C
C Set offsets (in inches) with information from above
C
      OFFSET=0.0
      IF((ANS.EQ.84).OR.(ANS.EQ.116)) OFFSET=4.375
      IF((ANS.EQ.66).OR.(ANS.EQ.98)) OFFSET=0.75
      IF(OFFSET.EQ.0.0) GO TO 109
      WRITE(3,260)
260  FORMAT(' Hit <return> when the plotter is ready')
      READ(1,240) ANS
      CALL SYMBOL(.25,OFFSET+.375,1,FILNAM,90.0,8)
      TITLE(1)=LHC
      TITLE(2)=LHh
      TITLE(3)=LH.
      TITLE(5)=ICHAN/10 + 48
      TITLE(6)=MOD(ICHAN,10) + 48
      CALL AXIS(1.0,OFFSET,TITLE,16,3.0,90.0,YARR(NPNT+1),YARR(NPNT+2))
      CALL AXIS(1.0,OFFSET,' ',1,9.,0.,XARR(NPNT+1),XARR(NPNT+2))
      CALL LINE(XARR,YARR,NPNT,1,0,0)
      IF(IMEAN.EQ.0) GO TO 113
C
C Plot mean value line, if requested
C

```

```

SUM=0.
DO 111 I=1, NPNT
    SUM=SUM+YARR(I)
111 CONTINUE
YMEAN=SUM/FLOAT(NPNT)
YARR(1)=YMEAN
YARR(2)=YMEAN
YARR(3)=YARR(NPNT+1)
YARR(4)=YARR(NPNT+2)
XARR(2)=XARR(NPNT)
XARR(3)=XARR(NPNT+1)
XARR(4)=XARR(NPNT+2)
CALL LINE(XARR, YARR, 2, 1, 0, 0)
113 CALL PLOT(0.0, 0.0, 999)
WRITE(3, 280)
280 FORMAT(' Plotting finished')
RETURN
END

```

C
C
C
C

Subroutine to plot FFTs vertically on page

```

SUBROUTINE FFTPLT
REAL XARR(2050), YARR(2050)
INTEGER*1 FILNAM(16), TITLE(16), ANS, TEMP(3), HUNIT(6), VUNIT(20)
INTEGER*1 TEXT1(16), TEXT2(16), TEXT3(16)
COMMON FILNAM, TITLE, XARR, YARR, ICHAN, NPNT
DATA HUNIT/ 'H', 'e', 'r', 't', 'z', ' '/
WRITE(3, 200) ICHAN, (TITLE(I), I=9, 16), NPNT
200 FORMAT(' FFT of Channel ', I2, 2X, 8A1, 3X, I5, ' points')

```

C
C
C

Input of data and calculation of standard deviation and peak frequency

```

FMAX=0.
YSUM=0.
Y2SUM=0.
YMAX=0.
DO 101 I=1, NPNT
    READ(5, 201) XARR(I), YARR(I)
201    FORMAT(F10.4, F15.5)
    YSUM=YSUM+YARR(I)
    Y2SUM=Y2SUM + YARR(I)*YARR(I)
    IF(YARR(I).LE.YMAX)GO TO 101
    YMAX=YARR(I)
    FMAX=XARR(I)
101 CONTINUE
YMEAN=YSUM/FLOAT(NPNT)
VAR=ABS((Y2SUM - YSUM*YSUM/FLOAT(NPNT))/FLOAT(NPNT-1))
Y1=5.
WRITE(3, 203) VAR, FMAX, YMAX
203 FORMAT(' var.=' , G15.8, ', Fmax=' , G15.8, ', S(Fm)=' , G15.8)

```

C
C
C

Convert variance and max frequencies into character for plotting

```

DO 103 I=1, 16

```



```

        TEXT1(I)=1H
        TEXT2(I)=1H
        TEXT3(I)=1H
103 CONTINUE
    ENCODE(TEXT1,202)VAR
202 FORMAT(7X,G9.3)
    ENCODE(TEXT2,202)YMAX
    ENCODE(TEXT3,204)FMAX
204 FORMAT(7X,F7.4)
    TEXT1(1)=1Hv
    TEXT1(2)=1Ha
    TEXT1(3)=1Hr
    TEXT1(4)=1H.
    TEXT1(6)=1H=
    TEXT2(1)=1HS
    TEXT2(2)=1H(
    TEXT2(3)=1HF
    TEXT2(4)=1Hm
    TEXT2(5)=1H)
    TEXT2(6)=1H=
    TEXT3(1)=1HF
    TEXT3(2)=1Hm
    TEXT3(3)=1Ha
    TEXT3(4)=1Hx
    TEXT3(6)=1H=
    TEXT3(15)=1HH
    TEXT3(16)=1Hz
    FILNAM(9)=1H
    FILNAM(10)=1HC
    FILNAM(11)=1Hh
    FILNAM(12)=1H.
    FILNAM(13)=ICHAN/10 + 48
    FILNAM(14)=MOD(ICHAN,10) + 48

```

```

C
C Prompt for plotting parameters
C

```

```

    WRITE(3,210)
210 FORMAT(' Should this be plotted on the <T>op or <B>ottom of',
    & ' the page?')
    READ(1,220) ANS
220 FORMAT(30A1)
    IF ((ANS .EQ. 66) .OR. (ANS .EQ. 98)) Y1=Y1+5.
    DO 105 I=1,20
        VUNIT(I)=1H
105 CONTINUE
    WRITE(3,230)
230 FORMAT(' Input vertical label ')
    READ(1,220) (VUNIT(I),I=1,20)

```

```

C
C Plot
C

```

```

    WRITE(3,240)
240 FORMAT(' Hit <return> when the plotter is ready')
    READ(1,250) ANS
250 FORMAT(A1)

```

```

CALL PLOTS ('HP7470')
COORD=Y1-4.5
CALL SYMBOL(COORD,5.,1,FILNAM,90.,16)
COORD=COORD+.25
CALL SYMBOL(COORD,5.,1,TEXT1,90.,16)
COORD=COORD+.25
CALL SYMBOL(COORD,5.,1,TEXT2,90.,16)
COORD=COORD+.25
CALL SYMBOL(COORD,5.,1,TEXT3,90.,16)
FAC=0.9
CALL FACTOR(FAC)
CALL SCALE(XARR,5.,NPNT,1)
CALL SCALE(YARR,4.,NPNT,-1)
CALL AXIS(Y1,2.5,HUNIT,-6,5.,90.,XARR(NPNT+1),XARR(NPNT+2))
CALL AXIS(Y1,2.5,VUNIT,20,4.0,180.0,YARR(NPNT+1),YARR(NPNT+2))
CALL LINE(YARR,XARR,NPNT,1,0,0)
CALL PLOT(0.,0.,999)
STOP
END

```

C
C
C

```

SUBROUTINE PHSPLT
REAL XARR(2050),YARR(1025,2)
INTEGER*1 FILNAM(13),TITLE(16),ANS,TEMP(3)
COMMON FILNAM,TITLE,XARR,YARR,ICHAN,NPNT

```

C
C Subroutine to plot phase and coherency

```

C
READ(5,200) ICHAN2
200 FORMAT(I5)
WRITE(3,210) ICHAN,ICHAN2,NPNT
210 FORMAT(' Phase and coherency of Ch.',I2,' and ',I2,I8,' points')
NPNT=NPNT+2
DO 101 I=3,NPNT
READ(5,215) XARR(I),YARR(I,1),YARR(I,2)
215 FORMAT(F10.4,2F15.5)
101 CONTINUE

```

C
C Dummy values are put into the first two spots in all three arrays
C to force the plotting routines to plot between -100 and 300 degrees
C and between 0 and 1 on the two plots.

```

C
XARR(1)=0.
XARR(2)=0.
YARR(1,1)=290.
YARR(2,1)=-90.
YARR(1,2)=.9999
YARR(2,2)=.0001
BOUND=-70.
DO 107 I=3,NPNT
IF(YARR(I,1).LT.BOUND) YARR(I,1)=YARR(I,1)+360.
107 CONTINUE
108 Y1=5.
Y2=10.

```

```
C
C Create title, one character at a time
C
```

```
TITLE(1)=LHC
TITLE(2)=LHh
TITLE(3)=LH.
TITLE(4)=LH
TITLE(5)=ICHAN/10 + 48
TITLE(6)=MOD(ICHAN,10) + 48
TITLE(7)=LH
TITLE(8)=LH&
TITLE(9)=LH
TITLE(10)=ICHAN2/10 + 48
TITLE(11)=MOD(ICHAN2,10) + 48
```

```
C
C Do actual plotting
C
```

```
CALL PLOTS('HP7470')
WRITE(3,240)
240 FORMAT(' Hit <return> when the plotter is ready')
READ(1,250)ANS
250 FORMAT(A1)
YTEMP=Y1-4.5
CALL SYMBOL(YTEMP,3.,2,TITLE,90.,11)
CALL FACTOR(.9)
CALL SCALE(XARR,5.,NPNT,1)
CALL SCALE(YARR(1,1),4.,NPNT,-1)
CALL SCALE(YARR(1,2),4.,NPNT,-1)
CALL AXIS(Y1,2.5,'Hertz',-5,5.,90.,XARR(NPNT+1),XARR(NPNT+2))
CALL AXIS(Y1,2.5,'Phase in degrees',16,4.0,180.0,YARR(NPNT+1,1),
& YARR(NPNT+2,1))
CALL LINE(YARR(1,1),XARR,NPNT,1,0,0)
CALL PLOT(0.,0.,-999)
CALL FACTOR(.9)
CALL AXIS(Y2,2.5,'Hertz',-5,5.,90.,XARR(NPNT+1),XARR(NPNT+2))
CALL AXIS(Y2,2.5,'Coherency',9,4.0,180.0,YARR(NPNT+1,2),
& YARR(NPNT+2,2))
CALL LINE(YARR(1,2),XARR,NPNT,1,0,0)
CALL PLOT(0.,0.,999)
RETURN
END
```

PLOTR TO GRAFTALK CONVERSION PROGRAM v.1.0

C PLT2GTK PLOTR to GrafTalk conversion program v.1.0 4-24-84

C

C Written by: Don Hacherl
C Floating Breakwater Project
C Univ. of Wash. Dept. of Civil Engineering
C Written for: Coastal Engineering Research Center
C U.S. Army Corps of Engineers
C

C This program gathers up all the .Cxx and .Fxx PLOTR files with a
C given stem and combines them into a single file with the same stem
C and an extender of .GTK which can be used as a GRAFTALK command file.
C

LOGICAL IOLOOK,IOREAD,IOCLOS
CHARACTER*11 FILSTM
CHARACTER*14 INFILE
CHARACTER*1 PLTYPE
CHARACTER*2 ITOC
CHARACTER*8 TITLE
CHARACTER*30 LINE
INTEGER*1 KHAR

C

C Start off by getting the file name and opening the output file as unit 6
C

C

CALL INIT(FILSTM)

C

C Now write "processing" message and begin looping. Outside loop is on
C plot type (FFT or row), inside loop is on channel number.
C

C

INFILE=""
CALL CONCAT(INFILE,FILSTM," ")
WRITE(1) 13,10,"Processing ",INFILE
DO 117 K=1,2
IF(K.EQ.1) PLTYPE="C"
IF(K.EQ.2) PLTYPE="F"
DO 115 J=1,80

C

C Prepare next filename and test for its existence
C

C

INFILE=""
CALL CONCAT(INFILE,FILSTM,PLTYPE,ITOC(J))
IF(.NOT.IOLOOK(0,INFILE))GO TO 115

C

C File INFILE has been found, so write its name to the terminal and
C begin processing by reading first line of important information.
C

C

WRITE(1) 8,8,8,PLTYPE,ITOC(J)
IF(IOREAD(5,2,0,INFILE))GO TO 180
READ(5,200) ITYPE,ICHAN,TITLE,NUM
200 FORMAT(11,I4,A8,2X,I5)

C

C Check to see if file contains supported plot type

C ITYPE == 0 --> row plot

C ITYPE == 2 --> FFT plot

C

IF((ITYPE.NE.0).AND.(ITYPE.NE.2))GO TO 182

```

        WRITE(6,210)
210      FORMAT(" DEVICE SCREEN",/," @DATA")
C
C Transfer X-Y coordinate pairs from source to destination as character
C without doing any reformatting.
C
        DO 101 I=1,NUM
            READ(5,220)LINE
220          FORMAT(A0)
            WRITE(6,220)LINE
101      CONTINUE
            WRITE(6,230)
230      FORMAT(" @END")
C
C Reformat TITLE by replacing embedded blanks with underscores ( )
C
        DO 103 I=1,KLEN(TITLE)
            IF(KHAR(TITLE,I).EQ.32)CALL PUTCHR(TITLE,I,!5f!)
103      CONTINUE
C
C Write titles and labelling information to GRAFTALK file
C
        IF(ICHAN.GE.10)WRITE(6,240)FILSTM,ICHAN,TITLE
240      FORMAT(" TITLE ",A0," Channel ",I2," ",A0)
        IF(ICHAN.LE.9)WRITE(6,250)FILSTM,ICHAN,TITLE
250      FORMAT(" TITLE ",A0," Channel __",I1," __",A0)
        IF(ITYPE.EQ.0)WRITE(6,260)
260      FORMAT(" X NAME Time in seconds")
        IF(ITYPE.EQ.2)WRITE(6,270)
270      FORMAT(" X NAME FREQUENCY_IN_Hz")
C
C The next write statement produces code that should eliminate those annoying
C "Unable to determine scaling" messages that crop up on single valued
C plots. Unfortunately, although the functions used are described in all
C their glory in the GrafTalk manual, they do not work on the version of
C GrafTalk that is currently running on our system. (the CompuPro) When
C this code is moved to a new system or a new version of GrafTalk arrives
C that agrees with the manual, the comment markers should be removed and
C the program recompiled.
C
C
C        WRITE(6,271)
C 271      FORMAT(" COMPARE C2[1] C2[10]",/,"
C      &      " IF TRUE C2[1] = C2[10] - 1.0")
C
C        WRITE(6,280)
280      FORMAT(" PLOT C2 VS C1",/," DUMP UNIFORM PAGE",/," INITIALIZE")
C
C Close input file and loop back to next filename.
C
        IF(IOCLOS(5))GO TO 180
115      CONTINUE
117      CONTINUE
C
C Individual files have now been completely processed, so write an exit
C command to GrafTalk, close the output file and quit.

```

```

C
  WRITE(6,290)
290 FORMAT(" EXIT")
  IF(IOCLOSE(6))GO TO 180
  STOP
C
C Error messages
C
180 WRITE(1,300)
300 FORMAT(/," ***File open/close error")
  STOP
182 WRITE(1,310) INFILE
310 FORMAT(/," ***",A0," does not contain recognizable data")
  STOP
  END
C
-----
C
  SUBROUTINE INIT(FILSTM)
  CHARACTER*11 FILSTM
  CHARACTER*14 OUTFIL,UPCASE
  LOGICAL IOLOOK,IOWRIT
C
  WRITE(1,200)
200 FORMAT(//," PLT2GTK PLOTR to GRAFTALK conversion program v.1.0",//,
  & " Enter the data file stem name "$
  READ(1,210)FILSTM
210 FORMAT(A0)
C
C Convert FILSTM to upper case and make sure it ends with a dot.
C
  FILSTM=UPCASE(FILSTM)
  IF(INDEX(".",FILSTM,1).EQ.0)GO TO 103
  CALL SETLEN(FILSTM,INDEX(".",FILSTM,1))
  GO TO 105
103 CALL ADDSTG(FILSTM, ".")
C
C Now generate output file name in OUTFIL and open the file.
C
105 OUTFIL=""
  CALL CONCAT(OUTFIL,FILSTM,"GTK")
  IF(IOLOOK(0,OUTFIL))GO TO 180
  IF(IOWRIT(6,2,0,OUTFIL))GO TO 182
C
C Write the very first GRAFTALK command, a plea for patience.
C
  WRITE(6,215)
215 FORMAT(" MESSAGE SSP_plotting_file_being_processed_-_please_wait...")
  RETURN
C
C Error messages
C
180 WRITE(1,220)OUTFIL
220 FORMAT(" ***",A0," already exists")
  STOP

```

```
182 WRITE(1,230)OUTFIL
230 FORMAT(" ***Unable to open ",A0," for output")
STOP
END
```

C

C-----

C

```
CHARACTER*2 FUNCTION ITOC(I)
CHARACTER*2 TEMP
INTEGER*1 C
```

C

C Function to convert integer (00-99) into ASCII form with leading zero.

C

```
TEMP="00"
C = I/10 + 48
CALL PUTCHR(TEMP,1,C)
C = I - (I/10)*10 + 48
CALL PUTCHR(TEMP,2,C)
ITOC = TEMP
RETURN
END
```

C

C-----

C

```
CHARACTER*14 FUNCTION UPCASE(X)
CHARACTER*14 X
INTEGER*1 KHAR,K
```

C Function to convert filenames to uppercase

```
UPCASE=X
DO 110 I=1,KLEN(X)
K=KHAR(X,I)
IF((K.GE.97).AND.(K.LE.122))CALL PUTCHR(UPCASE,I,K-32)
```

110 CONTINUE

```
CALL SETLEN(UPCASE,KLEN(X))
RETURN
END
```


MAKEMAC

C
C Program MakeMac v.1.0 Creates simple macro files for SSP v.1.3

C
C 4-12-84

C The macros created by this program take the form of columns of menu
C choices and answers to questions posed by SSP. Because of this, if
C any changes are made to SSP, such as renumbering the main menu or
C the addition of new options, that would change the commands needed,
C great care must be taken to make the corresponding change to the
C appropriate subroutine of this program.

C
C INTEGER ICHAN(80),PSTART,PNUM
C LOGICAL SCALEP,PLOTP,FFTP,FPLOTP,QUITP,IOLOOK,IOWRIT,IOCLOS
C CHARACTER*1 ANS
C CHARACTER*14 FILE,UPCASE
C DATA SCALEP,PLOTP,FFTP,FPLOTP,QUITP /5*.FALSE./
C WRITE(1,200)
C 200 FORMAT(/," MakeMac v. 1.0 SSP Macro Generator",/,
C & " Enter a name for the macro file "\$
C READ(1,210)FILE
C 210 FORMAT(A0)
C FILE=UPCASE(FILE)
C IF(IOLOOK(0,FILE))GO TO 180
C IF(IOWRIT(6,2,0,FILE))GO TO 185

C
C Output file is now open so give brief explanation and proceed to
C boring questions.

C
C WRITE(1,220)
C 220 FORMAT(/," MakeMac creates simple macro command files for SSP v.1.3 ",/
C & " The idea behind this is to perform an identical series",
C & " of actions on",/,
C & " a large number of channels. To do this you must first ",
C & "specify what",/,
C & " actions you want to take (scaling, plotting and/or ",
C & "transforming) and",/,
C & " then giving a list of which channels you want to do this to.",/)

C
C Begin asking point by point whether or not each action is to be performed.
C If the action is selected, then its predicate (e.g. SCALEP) is set to TRUE.
C This value is then tested in the next section to determine what code
C should be written.

C
C Ask about scaling

C
C WRITE(1,230)
C 230 FORMAT(" Do you want the data scaled? (Y/N) "\$
C READ(1,210)ANS
C IF((ANS.EQ."Y").OR.(ANS.EQ."y"))SCALEP=.TRUE.

C
C Ask about plotting raw data, getting more details if necessary

C
C WRITE(1,240)
C 240 FORMAT(" Do you want the untransformed data plotted? (Y/N) "\$

```

READ(1,210)ANS
IF((ANS.EQ."N").OR.(ANS.EQ."n"))GO TO 105
WRITE(1,250)
250  FORMAT(" Starting at what point? "$
READ(1,260)PSTART
260  FORMAT(I0)
WRITE(1,270)
270  FORMAT(" And for how many points? "$
READ(1,260)PNUM
PLOTP=.TRUE.

C
C Ask about doing Fourier transforms
C
105 WRITE(1,280)
280  FORMAT(" Do you want the channel FFTed? (Y/N) "$
READ(1,210)ANS
IF((ANS.EQ."N").OR.(ANS.EQ."n"))GO TO 107
WRITE(1,290)
290  FORMAT(" Enter 1 for full frequency or 2 for every other point FFT "$
READ(1,260)N
FFTP=.TRUE.

C
C If the channel has been transformed, ask about plotting the spectrum
C
WRITE(1,300)
300  FORMAT(" Do you want to plot the spectrum? (Y/N) "$
READ(1,210)ANS
IF((ANS.EQ."Y").OR.(ANS.EQ."y"))FPLOTP=.TRUE.

C
C All done asking about what should be done, now ask about what it should
C be done to, specifically which channels. If the first number entered is
C negative then ignore the rest of the line and generate the channel
C numbers 1 to 80 instead.
C
107 WRITE(1,310)
310  FORMAT(" Enter the channel numbers, separated with commas. ",
& "Enter a negative",/,
& " number for all 80 channels. Use ^E if input exceeds one line")
READ(1,320)(ICHAN(I),I=1,80)
320  FORMAT(80I0)
IF(ICHAN(1).GT.0)GO TO 111
DO 109 I=1,80
ICHAN(I)=I
109  CONTINUE

C
C Now ask whether the macro should end with a return or a quit
C
111 WRITE(1,330)
330  FORMAT(" When done processing this macro, should SSP <Q>uit or <R>eturn",
& /," control to the terminal? (Q/R) "$
READ(1,210)ANS
IF((ANS.EQ."Q").OR.(ANS.EQ."q"))QUITP=.TRUE.
WRITE(1,340)
340  FORMAT(" Writing macro file...")

C

```

C
C
C Now write the actual macro. Start index I at 1 and increment stopping
C when it reaches 80 or finds an invalid channel number.
C

```
      I=0  
113 I=I+1  
      IF((I.GT.80).OR.(ICHAN(I).LE.0))GO TO 117  
          CALL LOAD(ICHAN(I))  
          IF(SCALEP)CALL SCALE  
          IF(PLOTP)CALL PLOT(PNUM,PSTART)  
          IF(FFTP)CALL FFT(N)  
          IF(FPLOTP)CALL FPLOT  
          GO TO 113
```

C
C Now the entire macro has been written except for the quit or return
C command, so write that, close the file and stop.
C

```
117 CALL QUIT(QUITP)  
      I=IOCLOS(6)  
      STOP
```

C
C Error messages
C

```
180 WRITE(1,350)FILE  
350 FORMAT(1X,A0," already exists")  
      STOP  
185 WRITE(1,360)FILE  
360 FORMAT(" ***Unable to open ",A0," for output")  
      STOP  
      END
```

C
C-----
C

```
      CHARACTER*14 FUNCTION UPCASE(X)  
      CHARACTER*14 X  
      INTEGER*1 KHAR,K  
C Function to convert filenames to uppercase  
      UPCASE=X  
      DO 110 I=1,KLEN(X)  
          K=KHAR(X,I)  
          IF((K.GE.97).AND.(K.LE.122))CALL PUTCHR(UPCASE,I,K-32)  
110 CONTINUE  
      CALL SETLEN(UPCASE,KLEN(X))  
      RETURN  
      END
```

C
C-----
C*****

C
C Each of the following subroutines writes out enough to go from SSP's
C main menu into an option and then back out to the main menu again.
C In the interest of brevity, they will only be described as to what
C option they drive and what effects (if any) their argument has.
C Unless otherwise noted, all commands will use array A in SSP.

```

C
C*****
C
      SUBROUTINE LOAD(ICHAN)
C Loads channel ICHAN into array A
      WRITE(6,200) ICHAN
      200 FORMAT(" 1",/,I3,/, " A")
      RETURN
      END

```

```

C
C-----
C
      SUBROUTINE SCALE
C Scales array A
      WRITE(6,200)
      200 FORMAT(" 3",/, " A")
      RETURN
      END

```

```

C
C-----
C
      SUBROUTINE PLOT(I,J)
C Plots I points of raw data starting at point J
      WRITE(6,200) I,J
      200 FORMAT(" 12",/, " A",/,I5,/,I5)
      RETURN
      END

```

```

C
C-----
C
      SUBROUTINE FFT(N)
C Performs every Nth point FFT
      WRITE(6,200) N
      200 FORMAT(" 8",/, " A",/,/,I3)
      RETURN
      END

```

```

C
C-----
C
      SUBROUTINE FPLOT
C Plots smoothed spectrum (full range)
      WRITE(6,200)
      200 FORMAT(" 12",/, " A",///, " Y")
      RETURN
      END

```

```

C
C-----
C
      SUBROUTINE QUIT(QUITP)
C
C If QUITP is true, then writes a quit command. If QUITP is false then
C writes a return command.
C
      LOGICAL QUITP
      IF(QUITP) GO TO 101

```

```
      WRITE(6,200)
200   FORMAT(" 11",/," ZZZZ")
      RETURN
101  WRITE(6,210)
210  FORMAT(" 13")
      RETURN
      END
```

PROGRAM QIF
QANTEX INTERFACE

```

C PROGRAM QIF: QANTEX INTERFACE
C VERSION 1.4 LAST REVISED 1/23/84
C
C THIS PROGRAM IS WRITTEN TO OPERATE ON MICRO-SOFT 8-BIT FORTRAN
C UNDER CP/M 2.2 or MP/M 8/16.
C THE BULK OF THE SOURCE CODE WAS WRITTEN BY ROBERT W. MILLER OF
C OF U.W. ALTHOUGH THE MACHINE-LANGUAGE PATCH SEGMENTS WERE PROVIDED
C BY ALAN LINDSAY.
C
C IF ANY CHANGES ARE MADE TO THIS CODE, IT MUST BE COMPILED USING
C THE COMMAND " SUBMIT QTCOM QIF"
C
C FILES WHICH MUST BE PRESENT FOR THIS PROCESS: QTMAIN.MAC, PATCH.COM
C L80.COM, M80.COM, ACLLIB.IRL, FORLIB.IRL, QTU5.HEX, SUBMIT.COM, AND
C QTCOM.SUB
C
C THE FOLLOWING LOGICAL UNIT NUMERS ARE USED FOR FILE I/O:
C
C 1 READING FROM SCREEN
C 3 WRITING TO SCREEN
C 6 "GMAP.DAT" TEMPORARY FILE CONTAINING NUMBER AND TYPE
C AND ORDER OF ALL HEADERS READ DURING 1-MIN REC
C SEARCH AND DUMP
C 7 "SCRATCH.DAT" TEMPORARY FILE CONTAINING ALL UNSORTED RAW DATA
C 8 "HEADR.DAT" TEMPORARY COPY OF HEADER RECORD IN 'N' DRIVE
C WHEN TIMESERIES BEING READ
C 8 "1MINREC.DAT" TEMPORARY COPY OF HEADER RECORDS IN DEFAULT DRIVE
C WHEN 1-MIN RECORD SEARCH IS PROCEEDING
C 10 "TEMP.DAT" TEMPORARY STORAGE FOR ALPHA COMMENTS
C*MAIN
SUBROUTINE MAIN(ATUS ATUMEM, MEM)
EXTERNAL ATUS
BYTE ATUMEM(2), MEM(2)
INTEGER*1 IMSK(16), IFLAG, IHD, IANS
INTEGER RECNUM(2)
C-
C- MAIN PROGRAM IS SUPPLIED BY QTMAIN.MAC
C-
C- ATUS = ENTRY POINT FOR QTU5 CODE
C- ATUMEM = BASE OF "COMMON" DATA AREA IN QTU5
C- MEM(2) = AN ARRAY THAT ACCESSES ABSOLUTE MEMORY
C- I.E. MEM(45) IS THE CONTENTS OF ADDRESS 45 DECIMAL
C-
C- THINGS HAVE BEEN ARRANGED TO CONFORM WITH THE INFORMATION
C- PROVIDED IN THE "TIP" MANUAL AS MUCH AS POSSIBLE. YOU CAN
C- ACCESS THE DATA AREA USING ATUMEM(...), AND INVOKE THE
C- CONTROLLER SOFTWARE BY DOING A "CALL ATUS":
C-
C-----
C-
C- BYTE JTST
C-
C- OFFSETS INTO DATA AREA (SEE PAGE 36 OF TIP MANUAL)
C-
C- INTEGER*1 MENU, ALPHA(80), DDRV

```


INTEGER RAREA,WRDCNT,MA,PA,CA,DS,IS,ECODE
INTEGER IDATI(5),IDATO(5)

C-
C-

DATA RAREA /Z'76C'/
DATA WRDCNT /Z'76E'/
DATA MA /Z'770'/
DATA PA /Z'771'/
DATA CA /Z'772'/
DATA DS /Z'774'/
DATA IS /Z'775'/
DATA ECODE /Z'77C'/

C- 'MASK' IS RELATIVE TO MEM, I.E. MEM(MASK)
DATA MASK /Z'980'/

C-
C-
C-
C-
C-
C-
C-

THE FOLLOWING THREE LINES ARE ASCII CODES IDENTIFYING INITIAL HEADER SEQUENCES. A REVISION WAS MADE WHICH CHANGED THE HEADER STRUCTURE FROM 'STATIST' AND 'TIMESER' TO 'STATHDR' AND 'TIMEHDR' THEREFORE, IF THE TAPE NUMBER IS GREATER THAN 74 THE SEARCH IS FOR '????HDR' RATHER THAN 'STATIST' OR 'TIMESER' WHERE '?' IS A WILDCARD

C
C
C
C

DATA IMSK /63,63,63,63,72,68,82,63,63,63,63,63,63,63,63/
DATA IMSK1 /84,73,77,69,83,69,82/
DATA IMSK2 /83,84,65,84,73,83,84/
DATA ALPHA /80 * ' ' /

C-
C-

C-
C-
C-
C-
C

.....
THE FOLLOWING LINE DETECTS WHEN CONTROL DROPS IN FROM ABOVE FROM SUBROUTINE ATU5. THIS HAPPENS WHEN THERE IS NO MORE DATA ON THE TAPE DUE TO THE LACK OF FILE MARKS

C
C
C
C
C
C
C
C
C
C
C

IF (MENU .NE. 0) GO TO 960
IFLAG=0
IPASS=0

IPASS = VARIABLE TO DETERMINE IF MENU HAS BEEN USED

IFLAG = 0 NOTHING MUCH HAS HAPPENED YET
IFLAG = 1 SEARCHING FROM CONSOLE
IFLAG = 2 ONE MINUTE SEARCH AND DUMP
IFLAG = 3 WRITING T.S. TO DISK

WRITE(3,300)

300 FORMAT(' Program QIF vers. 1.4 for use with Qantex model 150')

C-
C-
C-
C-
C-
C-
C-
C

IDRV=DRIVE NUMBER (ALWAYS 1 UNLESS 2 OR MORE TAPE MACHINES HOOKED UP
ITRK=TRACK NUMBER RANGING FROM 1 TO 4
MA=MODE ARGUMENT (SEE TIP MANUAL)
PA=POSITIONAL ARGUMENT RANGING FROM 0 TO 255
CA=COMMAND ARGUMENT

IDRV = 1
ITRK = 1

C-
C-

PRESET MODE AND POSITION

```

C-      ATUMEM(MA) = ((IDRV-1)*4)+(ITRK-1)
      ATUMEM(PA) = 0
C-
C- JUMP TO RESET INTERFACE COMMAND AND THEN COME BACK
C INTERFACE MUST BE SET ONCE AT BEGINNING OF USE
C-
      GOTO 900
988 FORMAT(' this tape is not write protected',/,,'you may want to
      2remove it and turn write-protect screw on cartridge to safe')
C-
C-      50 CONTINUE
C-
C- DISPLAY STATUS BYTES AND ERROR CODE IF ANY
C-
      CALL DSPVAL('DS$',ATUMEM(DS),1)
      CALL DSPVAL('IS$',ATUMEM(IS),1)
      IF (ATUMEM(ECODE).NE.0) CALL DSPVAL('ECODE$',ATUMEM(ECODE),1)
C-
C- ENTRY POINT IF ERROR MESSAGES ARE HANDLED BY SUBROUTINE ALREADY
C-
      60 CONTINUE
C
C IF 'MENU'=0 I.E. IF THIS IS FIRST TIME THROUGH, LET OPERATOR KNOW
C IF TAPE IS NOT WRITE PROTECTED
C
      IF ((IPASS .EQ. 0) .AND. (DS .AND. 1)) WRITE(3,988)
      IF ((IPASS .EQ. 0) .and. (DS .AND. 1)) ipass=1
C-
C- RESET ERROR INDICATOR AND DISPLAY MENU
C-
      ATUMEM(ECODE)=0
      WRITE(3,303)
303 FORMAT(' 1-Change Track or PA')
      WRITE(3,313)
313 FORMAT(' 2-Rewind Tape')
      WRITE(3,314)
314 FORMAT(' 3-Read next record & Display')
      WRITE(3,315)
315 FORMAT(' 4-Forward Space record')
      WRITE(3,316)
316 FORMAT(' 5-Reverse Space record')
      WRITE(3,317)
317 FORMAT(' 6-Search for record')
      WRITE(3,318)
318 FORMAT(' 7-Write Timeseries to disk')
      WRITE(3,319)
319 FORMAT(' 8-Reset Interface')
      WRITE(3,338)
338 FORMAT(' 9-1 min rec. Search & Dump')
      WRITE(3,320)
320 FORMAT(' 10-Stop')
      WRITE(3,304)
304 FORMAT(' enter choice ')

```

```

        READ(1,201) MENU
201  FORMAT(I2)
        GOTO (550,600,650,700,750,800,850,900,1200,950),MENU
C-
C-  ERROR TRAPPING TO GUARD AGAINST DUFUSES
C-
        WRITE(3,331)
331  FORMAT(' invalid menu entry: try the chicken salad')
        GO TO 60
C
C  ROUTINE TO CHANGE PA AND MA
550  CONTINUE
        WRITE (3,305)
305  FORMAT (' enter track  ')
        READ (1,201) ITRK
        IF ((ITRK .LT. 1) .OR. (ITRK .GT. 4)) GO TO 560
        ATUMEM(MA)=((IDRV-1)*4)+(ITRK-1)
C
C  PROMPT FOR PA
C
        WRITE(3,306)
306  FORMAT(' enter positional argument  (1-256)')
        READ(1,202) IPAV
202  FORMAT(I3)
C
C  SET DEFAULT VALUE OF PA TO 0 IN CASE OF ABSURD INPUTS
C  NOTE: VALID PA'S ACCEPTABLE TO QANTEXT RANGE FROM 0 TO 255
C  THIS PROGRAM HAS BEEN ARRANGED SO USER INPUTS VALUES FROM 1 TO 256
C
        IF ((IPAV .GT. 256) .OR. (IPAV .LT. 1)) IPAV = 1
        ATUMEM(PA)=IPAV-1
C  NOW RETURN TO MENU
        GOTO 60
560  CONTINUE
        WRITE(3,358)
358  FORMAT(' not a reasonable track number; try again')
        GO TO 550
C-
C-  SUBROUTINE TO REWIND TAPE
C-
600  CONTINUE
        ATUMEM(CA) = Z'40'
        WRITE(3,982)
982  FORMAT(' rewinding . . .')
        CALL ATUS
        GOTO 50
C-
C-  READ AND DISPLAY RECORD
C-
650  CONTINUE
C  SET WRDCNT TO 512 (SIZE OF DATA RECORD)
        CALL IPUT(ATUMEM(WRDCNT),512)
C-
C-  READ AND DISPLAY
C-

```

```

WRITE(3,302)
302 FORMAT(' enter # of records to be read & displayed')
READ(1,202) KE
K = 1
100 CONTINUE
IF (K.GT.KE) GO TO 60
ATUMEM(ECODE)=0
C SET COMMAND ARGUMENT TO READ AND CALL ATU5
ATUMEM(CA) = Z'01'
CALL ATUS
C-
C DISPLAY STATUS BYTES
C
CALL DSPVAL('DS$',ATUMEM(DS),1)
CALL DSPVAL('IS$',ATUMEM(IS),1)
IF (ATUMEM(ECODE).NE.0) CALL DSPVAL('ECODE$',ATUMEM(ECODE),1)
C-
C NOW DISPLAY DATA
IBUF = IGET(ATUMEM(RAREA))
CALL DSPVAL('DATA$',MEM(IBUF),512)
C-
K = K+1
GO TO 100
C-
C-
C- FORWARD SPACE RECORD
C-
700 CONTINUE
ATUMEM(CA)=12
CALL ATUS
GOTO 50
C
C REVERSE SPACE RECORD
C
750 CONTINUE
ATUMEM(CA)=14
CALL ATUS
GOTO 50
C-
C- SEARCH UNDER MASK
C-
800 CONTINUE
C SET FLAG AND CALL ROUTINE TO GENERATE PROPER MASK (GOTO 1100)
IFLAG=1
GOTO 1100
C-
810 CONTINUE
C PROMPT FOR 1-MIN. REC. NUMBER OF HEADER TO BE SEARCHED FOR
WRITE(3,321)
321 FORMAT(' input # for 1-min. rec. #, 0 for next header <FIELD IS
2 I2>')
READ(1,203) I
203 FORMAT(I2)
C IF NUMBER OF 1-MIN. REC. IS ZERO, DISREGARD

```

```

        IF (I.LE.0) GO TO 828
        MEM(MASK+7) = I/256
        MEM(MASK+8) = I.AND.255
C-
828 CONTINUE
C  NOW DISPLAY MASK TO BE SEARCHED FOR
    CALL DSPVAL('MASK$',MEM(MASK),16)
C  SUBMIT SEARCH COMMAND
    ATUMEM(CA)=11
    CALL ATUS
C  CHECK FOR ERRORS
    IF (ATUMEM(ECODE) .EQ. 7) GO TO 960
    IF (ATUMEM(ECODE) .NE. 0) GO TO 840
C  DISPLAY RESULTS
    IBUF = IGET(ATUMEM(RAREA))
    CALL DSPVAL('DATA$',MEM(IBUF),512)
    GOTO 50
C
C  ROUTINE TO RESUBMIT COMMAND IN CASE OF GLITCH
C
840 CONTINUE
    WRITE(3,351)
351 FORMAT(' suspected glitch: attempt to resubmit search',/,
2' <CR> if you agree, any non-blank integer if not')
    READ(1,201) IANS
    ATUMEM(ECODE) = 0
    IF (IANS .NE. 0) go to 60
    GO TO 828
C-
C- WRITE FILE TO DISK
C
850 CONTINUE
C
C CHECK TO SEE IF QANTEX IS POSITIONED AT T.S. HEADER
C IF NOT, INQUIRE AS TO THE SANITY OF THE OPERATOR
C
    IF (.NOT.((MEM(IBUF).EQ.84) .AND. (MEM(IBUF+1) .EQ. 73).AND.
2(MEM(IBUF+2) .EQ. 77).AND.(MEM(IBUF+3) .EQ. 69))) GO TO 893
C QANTEX POSITION OK
851 CONTINUE
    IFLAG = 3
    CALL IPUT(ATUMEM(WRDCNT),512)
C CALCULATE NUMBER OF RECORDS CONTAINED IN TIMESERIES AND USE WITH PROMPT
    INUM1=MEM(IBUF+214)
    INUM2=MEM(IBUF+215)
    IF (INUM1 .LT. 0) INUM1=INUM1+256
    IF (INUM2 .LT. 0) INUM2=INUM2+256
    INUM1=((INUM1*256+INUM2)*3)/16
    WRITE(3,307)
307 FORMAT(' enter number of records to be dumped')
    WRITE(3,987) INUM1
987 FORMAT(' this timeseries contains ',I5,' records')
    READ(1,202) KE
C FIND OUT WHAT DRIVE SORTED OUTPUT IS TO BE WRITTEN TO AND WRITE TO FILE
C FILE "CONTOL.TMP"

```

```

WRITE(3,323)
323 FORMAT(' enter drive letter for output of sort <A1 field>')
READ(1,204)DDRIV
204 FORMAT(A1)
C FIND OUT DATE AND TIME TAPE PUT IN AND TAKEN OUT
CALL GDATE(IDATI,IDATO)
C IF TAPE NUMBER NOT YET KNOWN, FIND OUT
IF (ITAPE .LE. 0) CALL GTAP(ITAPE,IHD)
C OPEN FILE 'TEMP.DAT' FOR N DRIVE
3 CALL OPEN(10,'TEMP DAT',14)
C OBTAIN AND WRITE ANY COMMENTS (5 LINES PROVIDED IN HEADER)
DO 853 J=1,5
WRITE(3,327)
327 FORMAT(' input alpha comments , (5 lines tot.)')
READ(1,206,ERR=1000,END=853) (ALPHA(I),I=1,80)
206 FORMAT(80(A1))
WRITE(10,328) (ALPHA(I),I=1,80)
328 FORMAT(1X,80(A1))
853 CONTINUE
. ENDFILE 10

C-
C- TRANSFER HEADER FILE TO DISK
C-
C- DO IT IN 128 BYTE BLOCKS FOR EFFICIENY
C
CALL OPEN (8,'HEADR DAT',14)
IBUF=IGET(ATUMEM(RAREA))
DO 855 LOOP=0,511,128
L1=IBUF+LOOP
L2=L1+127
WRITE(8) (MEM(J),J=L1,L2)
855 CONTINUE
ENDFILE 8

C-
C- MAIN LOOP TO READ DATA AND WRITE TO FILE SCRATCH.DAT
C-
CALL OPEN (7,'SCRATCH DAT',14)
C IGCT = COUNT OF GLITCHES ENCOUNTERED
IGCT=0
C K = COUNT OF RECORDS READ
K=1
860 CONTINUE
C WHEN RECORDS READ EQUAL TO RECORDS DESIRED TRANSFER CONTROL TO
C 'FINISHING UP' ROUTINE
IF (K.GT.KE) GOTO 870
C READ NEXT DATA RECORD
ATUMEM(CA)=Z'01'
CALL ATUS

C-
C- CHECK FOR GLITCHES & IF FOUND TRANSFER CONTROL TO SR GLITCH
C-
C FIRST DISPLAY ANY NON-COPISCETIC ERROR CODES
IF (ATUMEM(ECODE).NE.0) CALL DSPVAL('ECODE$',ATUMEM(ECODE),1)
IF (ATUMEM(ECODE) .EQ. 7) GO TO 960
IGLTCH=(48 .AND. ATUMEM(IS))

```

```

C  GLITCH DEFINED AS ABORT WITH ATTEMPT
    IF (IGLTC .EQ. 32) CALL GLITCH(ATUMEM,CA,IGCT,K,IS)
C  CHECK FOR END OF TRACK
    IF (4 .AND. ATUMEM(DS)) GO TO 880
C-
C  NOW WRITE RECORD TO "SCRATCH.DAT"
    IBUF = IGET(ATUMEM(RAREA))
    DO 865 LOOP=0,511,128
    L1=IBUF+LOOP
    L2=L1+127
    WRITE(7)(MEM(J),J=L1,L2)
865 CONTINUE
C  INCREMENT RECORD COUNT
    K=K+1
    ATUMEM(ECODE)=0
C  NOW LOOP BACK
    GOTO 860
C  WE COME HERE WHEN WE HAVE READ ALL RECORDS
870 CONTINUE
C  IF THERE HAVE BEEN ANY GLITCHES, CALL GLPAD TO PAD LOST DATA SPACE WITH
C  WITH ZEROS
    IF (IGCT .NE. 0) CALL GLPAD(IGCT)
C  CLOSE "SCRATCH.DAT"
    ENDFILE 7
C  CALL ROUTINE TO WRITE CONTROL FILE FOR PROGRAM BWSORT USING CONTENTS
C  OF VARIOUS TEMPORARY FILES WRITTEN. THIS PROCEDURE WAS USED SO ALL
C  USER PROMPTS WOULD OCCUR AT BEGINNING OF PROCEDURE
    CALL CONFIL(KE,IGCT,ITAPE,IDATI,IDATO,ALPHA,DDRV)
C  PRINT MESSAGE AND STOP
    WRITE(3,350)
350 FORMAT(' write successfully completed')
    GOTO 950
880 CONTINUE

C
C  ROUTINE TO CHANGE TAPE TRACKS DURING A WRITE IF NECESSARY
C
    ITRK=ITRK+1
C  ONLY 4 TRACKS EXIST
    IF (ITRK .EQ. 5) ITRK=1
    ATUMEM(ECODE)=0
C  RESET MA AND CA
    ATUMEM(MA)=((IDRV-1)*4)+(ITRK-1)
    ATUMEM(CA) = Z'40'
    WRITE(3,989)
989 FORMAT(' end of track...starting next one')
    CALL ATUS
    IF (ATUMEM(ECODE) .NE. 0) CALL DSPVAL('ECODE$',ATUMEM(ECODE),1)
    ATUMEM(ECODE)=0
    GO TO 860
893 CONTINUE

C
C  THIS IS THE NORM S. MEMORIAL ERROR-TRAPPING SEGMENT FOR THOSE WHO
C  AREN'T CONVINCED OF THE IMPORTANCE OF STARTING A TAPE WRITE ON A
C  TIMESERIES HEADER AND WONDER WHY THERE'S GARBAGE IN THEIR FILES
C

```

```

WRITE(3,355)
355 FORMAT(' ahem! did you realize that what you are doing is highly
2 questionable? ',/, ' timeseries writes are best initiated when
3 tape is positioned at a header.',/, ' enter a 1 if you think you
4 know what you are doing,',/, ' a <CR> if you want to try again')
READ(1,201) IANS
IF (IANS .EQ. 1) GO TO 851
GO TO 60

C-
C- RESET INTERFACE, THEN WASTE SOME TIME
C-
900 CONTINUE
ATUMEM(CA) = Z'10'
CALL ATUS
K = 0
C FOLLOWING LOOP EXITED BY OVERFLOWING
910 CONTINUE
K = K+1
IF (K.NE.0) GO TO 910

C-
WRITE(3,301)
301 FORMAT(' reset complete')
GOTO 60
950 STOP
C WE COME HERE IF 'MENU' IS NOT EQUAL TO ZERO BEFORE FIRST EXECUTABLE
C STATEMENT OF PROGRAM (CONTROL HAS DROPPED IN FROM ABOVE I.E. FROM ATU5
960 CONTINUE
WRITE(3,346)
346 FORMAT(' tape machine has gone crazy: no more data?')
C DEPENDING UPON WHAT IS BEING DONE, TAKE VARIOUS ACTIONS
C IF 1-MINUTE SEARCH ONGOING, END IT
IF (IFLAG .EQ. 2) GO TO 1270
C IF DATA BEING WRITTEN TO DISK, WARN OPERATOR OF PROBLEMS AND CLOSE
C FILES
IF (IFLAG .EQ. 3) WRITE(3,347)
347 FORMAT(' Closing files: no guarantees as to what`s in them')
IF (IFLAG .EQ. 3) GO TO 870
C OTHERWISE, REDISPLAY MENU
GO TO 60
C ERROR TRAP
1000 CONTINUE
WRITE(3,330)
330 FORMAT(' problems with alpha write')
GOTO 950

C
C ROUTINE TO GENERATE MASK FOR SEARCHES
C
1100 CONTINUE
C
C CONTROL COMES HERE FROM 800 WITH IFLAG = 1 (SEARCH FROM CONSOLE)
C FROM 1200 WITH IFLAG = 2 (1-MIN. REC. SEARCH $ DUMP)
C
C IHD=0 TAPE # <=75
C IHD=1 TAPE # >75
C

```



```

CALL IPUT(ATUMEM(WRDCNT),512)
K1=MASK
K2 = 1
1105 CONTINUE
C ONLY 16 CHARACTER MASK IS CODED FOR IN THIS PROGRAM I.E.
C SET MASK CHARACTERS TO ???HDR????????? WHERE '?' IS WILD
  IF (K2.GT.16) GO TO 1110
  MEM(K1) = IMSK(K2)
  K1 = K1+1
  K2 = K2+1
  GO TO 1105
1110 CONTINUE
C GET TAPE NUMBER IF NOT KNOWN
  IF (ITAPE .LE. 0) CALL GTAP(ITAPE,IHD)
C BRANCH DEPENDING UPON WHICH HEADER FORMAT IS TO BE USED
C IF TAPE # < 75 CHANGE MASK TO ?????????????? AND COME BACK TO 1115
  IF (IHD .NE. 1) GO TO 1135
1115 CONTINUE
C IF IHD = 0 THEN FIRST 4 MASK CHARACTERS ARE ALWAYS 'STAT' I.E. 83,84,6,
C SINCE NO PRACTICAL WAY EXISTS ON THESE EARLIER TAPES TO SEARCH FOR BOTH
C 'TIMESER' AND 'STATIST'
  IF ((IHD .EQ. 0) .AND. (IFLAG .EQ. 2)) GO TO 1120
C ELSE IF IHD=1 AND IFLAG=2 THEN MASK IS LEFT AS '???HDR?????????' FOR
C PURPOSES OF MAPPING TAPE
  IF (IFLAG .EQ. 2) GOTO 1203
C WE COME HERE ONLY IF CONSOLE SEARCH IS BEING ATTEMPTED AND USER MUST
C SPECIFY WHICH HEADER IS BEING SEARCHED FOR
  WRITE(3,310)
310 FORMAT(' input "T" for timeseries, "S" for one min. record')
  I=1
  READ(1,983) IANS
983 FORMAT(A1)
  IF ((IANS .EQ. 83) .OR. (IANS .EQ. 115)) I=1
  IF ((IANS .EQ. 84) .OR. (IANS .EQ. 116)) I=0
C DEPENDING ON WHETHER 'STATIST' OR 'TIMESER' HEADER IS DESIRED, PUT IN
C CORRECT FIRST FOUR BYTES
  IF (I.EQ.0) GO TO 1125
C 1-MIN REC HEADER BEING SEARCHED FOR
1120 MEM(MASK)=83
  MEM(MASK+1)=84
  MEM(MASK+2)=65
  MEM(MASK+3)=84
  IF (IFLAG .EQ. 2) GO TO 1203
C RETURN CONTROL TO CONSOLE SEARCH ROUTINE
  GO TO 810
C TIMESERIES HEADER BEING SEARCHED FOR
1125 MEM(MASK)=84
  MEM(MASK+1)=73
  MEM(MASK+2)=77
  MEM(MASK+3)=69
C RETURN CONTROL TO CONSOLE SEARCH ROUTINE
  GO TO 810
C INSERT WILD CARD '?' IN MASK POSITION 5-7
1135 CONTINUE
  MEM(MASK+4)=63

```

```
MEM(MASK+5)=63
MEM(MASK+6)=63
GO TO 1115
```

```
C-
C- THIS PORTION OF THE PROGRAM CONTROLS AN EXTENDED SEARCH FOR ONE
C- ONE MINUTE RECORDS. THE 1-MIN. REC. DATA IS DUMPED TO DISK ALONG
C- WITH A MAP OF WHAT IS ON THE TAPE AND WHERE.
C DUE TO A LACK OF END-OF-FILE MARKS ON TAPE, IT IS DIFFICULT FOR
C- PROGRAM TO ASCERTAIN WHEN DATA IS AT AN END AND INITIATE AN ORDERLY
C EXIT FROM THE ROUTINE, CLOSING FILES ETC.
```

```
C-
C IF ORDERLY EXIT FROM SEARCH ROUTINE DOES NOT OCCUR, GENERATED
C FILES WILL BE LOST.
```

```
C-
C CURRENTLY PROGRAM ASSUMES THAT AN ECODE OF 7 IMPLIES NO MORE DATA
C AND SEARCH IS DISCONTINUED AT THIS POINT
```

```
C-
C OTHER CAUSES FOR A SEARCH DISCONTINUATION ARE IF CONTROL DROPS IN
C FROM ABOVE AT ANY OTHER TIME THAN AT PROGRAM INITIATION (BY USER)
C THIS SEEMS TO OCCUR OCCASIONALLY WHEN TAPE IS OUT OF DATA AND
C SOMETHING GOES WRONG IN ATUS. THIS CONDITION IS TRAPPED BY TESTING
C THE VARIABLE 'MENU' AT THE FIRST EXECUTABLE STEP, WHICH WILL ONLY
C BE ZERO IF THE PROGRAM HAS NOT YET RUN
```

```
C-
C IF OTHER UNANTICIPATED ERROR CODES (OR DS OR IS CODES) APPEAR TO
C SIGNIFY THE END OF DATA OR OTHERWISE INDICATE THAT A SEARCH SHOULD
C BE DISCONTINUED, THEY SHOULD BE ADDED TO THE CONDITIONAL 3 LINES
C AFTER STATEMENT 1250. FOR THIS REASON, IT IS IMPORTANT FOR OPERATORS
C TO NOTE ANY ECODES, DS, OR IS CODES IN THE EVENT OF UNUSUAL BEHAVIOR
```

```
C-
1200 CONTINUE
```

```
C-
C NOTE: IF BY SOME FANTASTIC COINCIDENCE, THE TAPE # IS GREATER THAN
C 75 AND A DATA RECORD (NOT A HEADER) HAPPENS TO HAVE ASCII 'HDR'
C IN POSITIONS 5 6 & 7, THEN THIS PROGRAM WILL THINK THAT IT IS A
C HEADER AND POSSIBLY GET VERY CONFUSED. THE CHANCES AGAINST THIS ARE
C 1 IN 255**3 BUT SOME PEOPLE ARE VERY LUCKY
```

```
C GET TAPE DATES
```

```
CALL GDATE(IDATI, IDATO)
```

```
C GET TAPE NUMBER IF NOT KNOWN
```

```
IF (ITAPE .LE. 0) CALL GTAP(ITAPE, IHD)
```

```
C OPEN TEMPORARY FILE TO STORE 1-MIN REC HEADERS
```

```
CALL OPEN(8, '1MINREC DAT', 0)
```

```
C OPEN TEMPORARY FILE TO STORE MAP OF HEADER POSITIONS
```

```
CALL OPEN(6, 'GMAP DAT', 0)
```

```
WRITE(6, 335) ITAPE
```

```
WRITE(6, 344) (IDATI(I), I=1, 5)
```

```
344 FORMAT(1X, 5I6)
```

```
WRITE(6, 344) (IDATO(I), I=1, 5)
```

```
C SET FLAG INDICATING AUTOMATED 1-MIN REC SEARCH AND DUMP
```

```
IFLAG=2
```

```
C INITIALIZE COUNTER OF 1-MIN REC HEADERS READ
```

```
NKNT=0
```

```
1202 CONTINUE
```

```

C TRANSFER CONTROL TO ROUTINE TO GENERATE CORRECT MASK FOR SEARCH
  GOTO 1100
1203 CONTINUE
C SET CA TO SEARCH AND CALL ATU5
  ATUMEM(CA)=11
  CALL ATUS
C CHECK FOR ANY ERROR CODES
  IF (ATUMEM(ECODE) .NE. 0) GOTO 1250
  IBUF=IGET(ATUMEM(RAREA))
C CHECK TO SEE IF HEADER READ BEGINS WITH 'TSER' AND IF SO GO TO 1210
  IF ((MEM(IBUF).EQ.84).AND.(MEM(IBUF+1).EQ.73).AND.(MEM(IBUF+2)
    2.EQ.77).AND.(MEM(IBUF+3).EQ.69)) GO TO 1210
C ELSE DUMP HEADER INTO FILE "LMINREC.DAT" (UNIT 8) SINCE IT MUST BE 1-MIN
1204 CONTINUE
  DO 1205 LOOP=0,511,128
  L1=IBUF+LOOP
  L2=L1+127
  WRITE(8)(MEM(J),J=L1,L2)
1205 CONTINUE
C CALCULATE RECORD NUMBER ENCODED IN HEADER
  RECNUM(1)=MEM(IBUF+7)
  RECNUM(2)=MEM(IBUF+8)
C COMPENSATE FOR FACT THAT THIS FORTRAN INTERPRETS INTEGER*1 FROM -128 TO
C INSTEAD OF 0 TO 255 AS INTENDED
  IF (RECNUM(1) .LT. 0) RECNUM(1)=RECNUM(1)+256
  IF (RECNUM(2) .LT. 0) RECNUM(2)=RECNUM(2)+256
C NOW COMBINE BOTH BYTES TOGETHER
  RECNUM(1)=RECNUM(1)*256+RECNUM(2)
C WRITE REC # TO SCREEN
  WRITE(3,334) RECNUM(1)
334 FORMAT(' reading rec # ',I6)
  WRITE(6,335) RECNUM(1)
335 FORMAT(' ',I6)
C INCREMENT COUNTER
  NKNT=NKNT+1
C SINCE LAST HEADER WAS 'STAT', READ NEXT RECORD AND SEE WHAT IT IS
  ATUMEM(CA)=1
  CALL ATUS
  IF (ATUMEM(ECODE) .NE. 0) GOTO 1250
  IBUF=IGET(ATUMEM(RAREA))
C CHECK TO SEE IF IT'S 'STAT' AND IF SO GO BACK UP ABOVE AND MAP IT
  IF ((MEM(IBUF) .EQ. 83) .AND. (MEM(IBUF+1) .EQ. 84) .AND.
    1(MEM(IBUF+2) .EQ. 65) .AND. (MEM(IBUF+3).EQ.84)) GO TO 1204
C ELSE SEE IF IT IS 'TIME' AND IF SO GO BELOW AND MAP IT
  IF ((MEM(IBUF).EQ.84).AND.(MEM(IBUF+1).EQ.73).AND.(MEM(IBUF+2)
    2.EQ.77).AND.(MEM(IBUF+3).EQ.69)) GO TO 1210
C ELSE SOMETHING IS WRONG SINCE 'STAT' HEADER SHOULD ALWAYS BE FOLLOWED
C BY ANOTHER HEADER (EXCEPT ON MANUALLY ACTIVATED TEST TAPES)
  GO TO 1220
1210 CONTINUE
C
C- WE GET HERE IF TS HEADER HAS BEEN READ
C-
  IF (MEM(IBUF+9) .LT. 0) MEM(IBUF+9)=MEM(IBUF+9)+256
  IF (MEM(IBUF+10) .LT. 0) MEM(IBUF+10)=MEM(IBUF+10)+256

```

```

MV=-4-256*MEM(IBUF+9)-MEM(IBUF+10)
WRITE(6,335) MV
GOTO 1202
1220 CONTINUE
C FOLLOWING STATEMENTS HAVE CAUSED PROBLEMS ON TEST TAPES
WRITE(3,337)
337 FORMAT (' record just read should have been a header.'/
1'either end of data or error: search discontinued.')
GOTO 1270
1250 CONTINUE
C WE COME HERE IF ERROR HAS BEEN DETECTED
C FIRST DISPLAY STATUS BYTES AND ERROR CODE
CALL DSPVAL('DS$',ATUMEM(DS),1)
CALL DSPVAL('IS$',ATUMEM(IS),1)
CALL DSPVAL('ECODE$',ATUMEM(ECODE),1)
IF (ATUMEM(ECODE) .EQ. 7) GO TO 1270
ATUMEM(ECODE)=0
C- IF END OF TRACK GO TO 1280, INCREMENT MA AND RESUBMIT SEARCH
IF (4 .AND. ATUMEM(DS)) GO TO 1280
C- ELSE ASSUME IT'S A GLITCH, MAP WITH -999 AND RESUBMIT SEARCH
MV=-999
WRITE(6,335) MV
GOTO 1202
C- CLOSE FILES AND RETURN TO MENU: -888 SIGNIFIES END OF MAP
1270 CONTINUE
MV=-888
WRITE(6,335) MV
C FINALLY, WRITE NUMBER OF RECORDS READ
WRITE(6,335) NKNT
C CLOSE BOTH FILES
ENDFILE 6
ENDFILE 8
WRITE(3,336)
336 FORMAT(' orderly exit from search routine')
C NOW STOP
GO TO 950
C- WE GET HERE IF BIT 2 OF DS IS ON-END OF TRACK
1280 CONTINUE
ITRK=ITRK+1
C SINCE ONLY 4 TRACKS ON TAPE, ASSUME END OF SEARCH AT END OF TRACK 4
IF (ITRK .GT. 4) GO TO 1285
ITM1=ITRK-1
WRITE(3,341) ITM1
341 FORMAT(' END OF TRACK ',I5)
C MAP END OF TRACK ON GLITCH MAP WITH A NEGATIVE TRACK NUMBER
ITM1=-ITM1
WRITE(6,335) ITM1
C RESET MA AND PA BEFORE REISSUING SEARCH
ATUMEM(MA)=(IDRV-1)*4+(ITRK-1)
ATUMEM(PA)=0
GO TO 1202
1285 CONTINUE
WRITE(3,340)
340 FORMAT(' end of track four: end of dump')
GO TO 1270

```

```

      END
C*IGET
      INTEGER FUNCTION IGET(IMEM)
      INTEGER IMEM
C-
C-   IGET - FUNCTION TO FETCH WORD DATA FROM BYTE STORAGE
C-
      IGET = IMEM
      RETURN
C-
      END
C*IPUT
      SUBROUTINE IPUT(IMEM,IVAL)
      INTEGER IMEM,IVAL
C-
C-   IPUT - ROUTINE TO STORE WORD DATA IN BYTE STORAGE
C-
      IMEM = IVAL
      RETURN
C-
      END
C-
C-   SUBROUTINE TO CONTROL READ IN THE EVENT OF A GLITCH
C-   TWO READS ARE PERFORMED AND THE RESULTS NOT STORED
C-   WHEN DISKWRITE IS FINISHED, SUBROUTINE GLTPAD PADS REMAINING
C-   SPACE WITH ZEROS
C
C   THE TWO 'DUMMY' READS ARE NECESSARY TO KEEP CHANNEL SEQUENCING
C   CORRECT
C-
      SUBROUTINE GLITCH(ATUMEM,CA,IGCT,K)
      INTEGER*1 ATUMEM(2)
      INTEGER CA,IGCT,K
180 CONTINUE
C   ISKP IS THE NUMBER OF DATA RECORDS SKIPPED WHEN GLITCH IS ENCOUNTERED
C
      ISKP=1
200 CONTINUE
      IF (ISKP .EQ. 4) GOTO 220
      ATUMEM(CA)=1
      CALL ATUS
      ISKP=ISKP+1
      GOTO 200
220 CONTINUE
C   WE COME HERE AFTER A GLITCH AND TWO ADDITIONAL READS
      K=K+3
C   LOG GLITCH AS HAVING OCCURRED
      IGCT=IGCT+1
      IGTST=(ATUMEM(IS) .AND. 48)
C   NOW CHECK TO SEE IF RECORD JUST READ IS ALSO A GLITCH:  IF SO, START
C   SUBROUTINE OVER
      IF (IGTST .EQ. 32) GOTO 180
      RETURN
      END
C-

```

```

C-   SUBROUTINE TO PAD FILE WITH APPROPRIATE # OF ZEROS TO REPLACE VALUES
C-   LOST BY GLITCHES.  THIS IS DONE AT END OF DISKWRITE
C-
      SUBROUTINE GLPAD(IGCT)
      INTEGER*1 DUM(128)
      DATA DUM /128 * 0/
C   EACH GLITCH REQUIRES 1536 VALUES OR 16 SAMPLES/CHANNEL BE SKIPPED
C   I.E. 12 X 128
      LOOP=12*IGCT
      DO 100 I=1,LOOP
      WRITE(7) (DUM(J),J=1,128)
100  CONTINUE
      RETURN
      END

C-
C-   SUBROUTINE TO WRITE CONTROL FILE FOR BWSORT
C-
      SUBROUTINE CONFIL(KE,IGCT,ITAPE,IDATI,IDATO,ALPHA,DDRV)
      INTEGER IDATI(5),IDATO(5)
      INTEGER*1 ALPHA(80),DDRV
      CALL OPEN(8,'CONTOL TMP',14)
C   FIRST WRITE TAPE NUMBER
      WRITE(8,101) ITAPE
101  FORMAT(1X,I6,I6)
C   WRITE NUMBER OF SAMPLES SUCCESSFULLY READ = KES
C   AND NUMBER OF SAMPLES ATTEMPTED = KEA
      KES=16*(KE-3*IGCT)/3
      KEA=16*KE/3
      WRITE(8,101) KES,KEA
C   WRITE DATE TAPE IN AND OUT
      WRITE(8,102) (IDATI(I),I=1,5)
102  FORMAT(1X,5(I6))
      WRITE(8,102) (IDATO(I),I=1,5)
C   WRITE DISK DRIVE WHICH FINAL DATA DISK WILL BE IN WHEN BWSORT IS RUN
      WRITE(8,103) DDRV
103  FORMAT(1X,A1)
      CALL OPEN(10,'TEMP DAT',14)
      40 CONTINUE
C   NOW TRANSFER ALPHA COMMENTS PREVIOUSLY WRITTEN IN 'TEMP.DAT' TO 'CONFIL.
      READ (10,104,ERR=200,END=50) (ALPHA(I),I=1,80)
104  FORMAT(80A1)
      WRITE(8,105) (ALPHA(I),I=1,80)
105  FORMAT(1X,80A1)
      GOTO 40
      50 CONTINUE
C   CLOSE FILES AND RETURN
      ENDFILE 8
      ENDFILE 10
      RETURN
200  CONTINUE
C   ERROR TRAP
      WRITE(3,106)
106  FORMAT(' problems with alpha reading in sr CONFIL')
      STOP
C

```

```

END
SUBROUTINE GDATE(IDATI, IDATO)
C SUBROUTINE TO PROMPT USER FOR DATES TAPE PUT IN AND TAKEN OUT OF BREAKWA
DIMENSION IDATI(5), IDATO(5)
WRITE(3, 325)
325 FORMAT(' input mo da yr hr min tape in '/' XX XX XX XX XX'/)
READ(1, 205) (IDATI(I), I=1, 5)
205 FORMAT(5(I2, 1X))
WRITE(3, 326)
326 FORMAT(' input mo da yr hr min tape out '/' XX XX XX XX XX'/)
READ(1, 205) (IDATO(I), I=1, 5)
RETURN
END

C
C SUBROUTINE TO GET TAPE NUMBER AND DETERMINE WHICH OF 2 HEADER FORMATS
C ARE APPLICABLE DEPENDING ON WHETHER TAPE # IS > OR < 74
SUBROUTINE GTAP(ITAPE, IHD)
100 WRITE(3, 324)
324 FORMAT(' input tape number <I3 FIELD>')
READ(1, 202) ITAPE
202 FORMAT(I3)
IF (ITAPE .LE. 0) GO TO 120
IF (ITAPE .GT. 75) IHD=1
RETURN
120 CONTINUE
WRITE(3, 327)
327 FORMAT(' PLEASE! Zero or negative tape numbers have been known
2 to blow up the qantex and advance world communism. ')
GO TO 100
END

```

PROGRAM BWSORT
TO REFORMAT OUTPUT OF QIF FOR SSP

C- PROGRAM BWSORT TO SORT DATA INTO RANDOM ACCESS FILE FOR INPUT
C- INTO PROGRAM SSO
C
C
C

C THIS PROGRAM IS NECESSARY DUE TO THE FACT THAT PROGRAM QIF MERELY
C TRANSFERS DATA DIRECTLY OFF QANTEX DRIVE
C BEFORE DATA IS USEFUL, IT MUST BE SORTED SO THAT THE DATA FOR EACH
C CHANNEL IS TOGETHER
C
C

C SOME CHANNELS (1-16) ARE TWO BYTE CHANNELS WHILE SOME (17-80) ARE
C ONE BYTE CHANNELS. THIS PROGRAM ALSO HANDLES THESE CONVERSIONS
C
C

C THESE FUNCTIONS ARE NOT HANDLED DIRECTLY BY PROGRAM QIF BECAUSE THIS
C IS THE MOST TIME-CONSUMING PORTION OF THE PROCESSING, AND IT THEREFORE
C HAPPENS MUCH FASTER WHEN A 16-BIT FORTRAN IS USED
C MORE RAM IS ALSO AVAILABLE FOR THIS PURPOSE
C
C

C PROGRAMMED BY ROBERT MILLER, UNIVERSITY OF WASHINGTON IN SUMMER OF 1983
C FOR WEST POINT PROTOTYPE FLOATING BREAKWATER TEST PROJECT
C US ARMY CORPS OF ENGINEERS
C
C

C FINAL RANDOM ACCESS DATA FILE IS ARRANGED SO THAT EACH CHANNEL
C MAY BE OPENED AS A FILE RECORD BY PROGRAM SSP
C-
C

UNIT NUMBER	FILE	DESCRIPTION
1		CONSOLE
5	"N:SCRATCH.DAT"	INPUT FILE CONTAINING UNSORTED DATA
6	"<d:>BW<tape #>R<rec #>.DAT"	FINAL OUTPUT FILE FOR DATA
7	"N:CONTOL.TMP"	TEMPORARY FILE LEFT BY PROGRAM QIF CONTAINING CONTROL INFO
9	"N:HEADR.DAT"	TEMPORARY HEADER FILE LEFT BY PROGRAM QIF
11	"<d:>BW<tape #>R<rec #>.HDR"	HEADER CONTAINING CONTROL INFO

C- DFN=DUMMY FILENAME; IFN=INPUT FILENAME; FNAME=OUTPUT FILENAME
C- TAPE=TAPE #; RECD=1 MIN-REC. #; FTYP=FILETYPE; FPFX=FILE PREFIX
C-

C CHARACTER*11 DFN
C CHARACTER*10 FTRAN
C CHARACTER*15 FNAME
C CHARACTER*4 FTYP
C

C- INTEGER*1 CHAN1B(2,256)
C- INTEGER CHAN2B(256)
C- INTEGER*1 IBUF(24576)
C

C- CONVERSION FROM ONE TO TWO BYTE INTEGERS IS DONE BY MEANS OF EQUIVALENCE
C-

```

C- OF ONE AND TWO BYTE ARRAYS. THIS AUTOMATICALLY WRITES TWO BYTE CHANNELS
C- AS ONE INTEGER AND RIDS NEGATIVE INTEGERS FROM ONE BYTE CHANNELS
C-
C- EQUIVALENCE (CHAN1B,CHAN2B)
C-
C- DATA FTYP /".DAT"/
C- DATA DFN /"SCRATCH.DAT"/
C-
C- INPUT SIZE OF FILE AND CHECK IF NUMBER OF SAMPLES IS A MULTIPLE
C- OF 256 (SAMPLES = RECORDS * 16/3)
C-
C- WRITE FINAL FORM OF HEADER ON DRIVE SPECIFIED BY USER FOR OUTPUT
C (THIS WAS DONE IN PROGRAM QIF)
C CALL HDRSR (FTRAN, ISAMP)
C IREC=(ISAMP*3)/16
C ICHK=IREC/48
C ICHK=ICLK*48
C IF (ICLK.NE.IREC) GOTO 900
C-
C- ISAMP=# OF SAMPLES IN TIMESERIES
C-
C- ATTEMPT TO OPEN AND/OR CREATE RELEVANT FILES
C-
C- IF (IORAND(24576,1,5,0,DFN)) GOTO 910
C-
C- ROUTINE TO CREATE OUTPUT FILE OF PROPER SIZE
C- OTHERWISE SUPERSOFT FORTRAN SEEMS UNABLE TO PROPERLY WRITE
C- RANDOM ACCESS FILES. THIS STEP IS A TIMEWASTER BUT WE HAVEN'T
C- FIGURED OUT A WAY AROUND IT YET
C
C CALL CONCAT (FNAME,FTRAN,FTYP)
C FOR REASONS OF EFFICIENCY IT IS CONVENIENT TO USE IMPLIED WRITES IIN
C IN BLOCKS OF 256 BYTES
C LC=ISAMP/256
C IF (LC*256 .LT. ISAMP) LC=LC+1
C DO 400 I=1,256
400 CHAN2B(I)=0
C FILE IS FIRST OPENED AS SEQUENTIAL
C IF (IOWRIT(6,0,0,FNAME)) GOTO 915
C DO 407 I=1,80
C DO 405 K=1,LC
C WRITE (6) (CHAN2B(J),J=1,256)
405 CONTINUE
407 CONTINUE
C NOW CLOSE FILE SO IT MAY BE REOPENED AS RANDOM ACCESS FILE
C IF (IOCLOS(6)) GOTO 920
C
C- RE-OPEN FILE AS RANDOM ACCESS WITH BLOCK SIZES OF 256 INTEGER*2
C- RECORDS
C-
C- 410 IF (IORAND(512,2,6,0,FNAME)) GOTO 905
C-
C- ICHN = CHANNEL #; ITRN = SCRATCH INTEGER
C IOUTL = OUTER LOOP CONTROLLER
C- IOFFS = FUNCTION RETURNING STARTING ADDRESS WITHIN 24576 BYTE

```

```

C-      RECORD TO ACCESS FIRST BYTE OF ICHN;
C-
C-      ICHAMP = NUMBER OF ITERATIONS THAT MUST BE PERFORMED @ 256 SAMPLES
C-      PER ITERATION TO COMPLETE SORT
C-      IDUM   = DUMMY VARIABLE TO STORE IOFFS
C
C      RINP IS A SUBROUTINE THAT READS IN THE DATA
C
C      WOUT IS A SUBROUTINE THAT WRITES OUT THE DATA
C
C      NOTE THE DIFFERENT MANNER IN WHICH THE TWO-BYTE (17 TO 80) VERSUS ONE
C      BYTE (1-16) CHANNELS MUST BE HANDLED.
C
C      INPUT FILE IS BROKEN INTO CHUNKS OF 24,576 BYTES = 256 SAMPLES/CHANNEL
C      ICHAMP IS TOTAL NUMBER OF ITERATIONS REQUIRED TO COMPLETE SPECIFIED
C      NUMBER OF RECORDS
C      ICHAMP=ISAMP/256
C      DO 450 IOUTL=1, ICHAMP
C          WRITE(1,360) IOUTL, ICHAMP
360      FORMAT(" beginning of iteration # ", I2, " of ", I2)
C          CALL RINP(IOUTL, IBUF)
C
C      FIRST PROCESS TWO-BYTE CHANNELS
C
C          DO 440 ICHN=1, 16
C              IDUM = IOFFS(ICHN)
C              DO 435 K=1, 256
C                  INLOOP = IDUM + (K-1)*96
C                  CHAN1B(2, K)=IBUF(INLOOP-1)
C                  CHAN1B(1, K)=IBUF(INLOOP)
435          CONTINUE
C              CALL WOUT(IOUTL, ICHN, CHAN2B, ICHAMP)
440          CONTINUE
C
C      THE FOLLOWING LOOP NECESSARY TO INSURE THAT CONVERSION FROM ONE BYTE
C      TO TWO BYTE INTEGER OCCURS WITHOUT NON-ZERO VALUES IN THE FOUR MOST
C      SIGNIFICANT BITS
C
C          DO 442 K=1, 256
C              CHAN1B(2, K)=0#
442          CONTINUE
C
C      NOW PROCESS ONE-BYTE CHANNELS
C
C          DO 447 ICHN = 17, 80
C              IDUM = IOFFS(ICHN)
C              DO 445 K=1, 256
C                  INLOOP = IDUM + (K-1)*96
C                  CHAN1B(1, K)=IBUF(INLOOP)
445          CONTINUE
C              CALL WOUT(IOUTL, ICHN, CHAN2B)
447          CONTINUE
450          CONTINUE
C-
C-      PROGRAM FINISHED: CLOSE OUTPUT FILE AND WRITE MESSAGE

```

```

C-      IF (IOCLOS(6)) GOTO 920
        WRITE(1,305) FNAME
305  FORMAT(" sort compleged: file ",A15)
        GOTO 1000

C-      900 CONTINUE
        WRITE(1,350)
350  FORMAT(" # of records must be multiple of 48: reread record")
        GOTO 1000

C
C  ERROR TRAPS
C-
905  CONTINUE
        WRITE(1,351)
351  FORMAT(" file opening error")
        GOTO 1000

C-
910  CONTINUE
        WRITE(1,352)
352  FORMAT(" error opening input file 'SCRATCH.DAT'")
        GOTO 1000

C-
915  CONTINUE
        WRITE(1,353)
353  FORMAT(" iowrit error: output file")
        GOTO 1000

C-
920  CONTINUE
        WRITE(1,354)
354  FORMAT(" ioclos error: output file")
        GOTO 1000
922  CONTINUE
        WRITE(1,357)
357  FORMAT(" error in ioread: file contol.tmp")
        GOTO 1000

C-
C-
1000 CONTINUE
        WRITE(1,306)
306  FORMAT(" orderly exit...")
        STOP
        END

C-
C-  SUBROUTINE TO DETERMINE OFFSET FOR LOOPING IN IBUF ARRAY
C-
C  IOFFS(CHANNEL NUMBER) IS THE NUMBER OF BYTES FROM THE BEGINNING OF THE
C  ARRAY THAT THE FIRST OCCURANCE OF A BYTE OF CHANNEL NUMBER ICHN OCCURS
C  THE FORMULA FOR CALCULATING THIS IS
C
C  OFFSET= 6*(CHANNEL NUMBER) - 95*INT(((CH #)-1)/16) - 4
C  THIS IS A CONSEQUENCE OF THE ORDER IN WHICH BYTES WERE ORIGINALLY
C  WRITTEN TO TAPE (SEE INFO ON FIELD DATA AQUISITION SYSTEM)
C
C  INTEGER FUNCTION IOFFS(ICHN)

```

```
ITRN=(ICHN-1)/16
IOFFS = 6*ICHN-95*ITRN-4
RETURN
END
```

```
C-
C- SUBROUTINE TO READ BLOCK OF 24576 BYTES FROM INPUT FILE
C-
```

```
SUBROUTINE RINP (IBLK,IBUF)
INTEGER*1 IBUF (24576)
READ (5/IBLK) (IBUF (I),I=1,24576)
RETURN
END
```

```
C-
C- SUBROUTINE TO WRITE OUT DATA
C-
```

```
SUBROUTINE WOUT (IOUTL,ICHN,CHAN2B,ICHAMP)
INTEGER CHAN2B (256)
```

```
C ICHN=(IOUTL-1)*8+J
IF (ICHN .GT. 16) GOTO 40
WRITE (6/((ICHN-1)*ICHAMP+IOUTL)) ((ISHIFT (CHAN2B (I),-4) .AND. 4095
2),I=1,256)
GOTO 50
40 CONTINUE
WRITE (6/((ICHN-1)*ICHAMP+IOUTL)) (CHAN2B (I),I=1,256)
50 CONTINUE
C WRITE (1,100) ICHN
C100 FORMAT (' UP TO CHANNEL # ',I6)
RETURN
END
```

```
C-
C- SUBROUTINE TO WRITE HEADR IN ASCII FORMAT FOR USE WITH PROGRAM
C- SSP
C FORMAT FOR THIS FILE MAY BE SEEN IN DOCUMENTATION OF PROGRAM SSP
SUBROUTINE HDRSR (FTRAN,ISAMP)
```

```
C-
INTEGER*1 HB1B (2,512)
INTEGER*2 HB2B (512)
INTEGER*2 IDATO (5),IDATI (5)
CHARACTER*1 DRV,COL,PART
CHARACTER*15 FNAME
CHARACTER*10 FTRAN
CHARACTER*2 FPFX
CHARACTER*80 COMMNT
CHARACTER*6 TAPE,RECD,STRCHK
CHARACTER*4 FTYP
```

```
C-
DATA FTYP /".HDR"/
DATA FPFX /"BW"/
DATA COL /":"/
DATA HB2B /512 * 0/
DATA PART /"R"/
```

```
C-
EQUIVALENCE (HB1B,HB2B)
C-
C- OPEN AND READ HEADR INPUT FILE
```

```

C-      IF (IORAND(512,1,9,0,"HEADR.DAT")) GOTO 900
      READ(9/1) (HB1B(1,I),I=1,512)
      IF (IOCLOS(9)) GOTO 920

C-
C-      READ IN CONTROL FILE TELLING HOW MANY SUCCESSFUL SAMPLES,DATES ETC.
C-      (SEE DOCUMENTATION FOR PROGRAM QIF FOR MORE DETAILED EXPLANATION OF TERMS
C
      IF (IOREAD(7,2,0,"CONTOL.TMP")) GOTO 922
      READ(7,700) ITAPE
700  FORMAT(2(I6))
      READ(7,700) ICOMP,ISAMP
      READ(7,701) (IDATI(I),I=1,5)
701  FORMAT(5(I6))
      READ(7,701) (IDATO(I),I=1,5)
      READ(7,702) DRV
702  FORMAT(A1)
      WRITE(TAPE,715) ITAPE
715  FORMAT(1X,I6)
      IREC=256*HB2B(8)+HB2B(9)
      ITSREC=256*HB2B(10)+HB2B(11)
C  RECD IS CHARACTER FORM OF 1-MIN RECORD NUMBER
      WRITE(RECD,715) IREC

C-
C-      GENERATE OUTPUT FILENAME
C-
      TAPE=STRCHK(TAPE)
      RECD=STRCHK(RECD)
      I=KLEN(TAPE)
      J=KLEN(RECD)
      CALL SETLEN(TAPE,I)
      CALL SETLEN(RECD,J)
      CALL CONCAT(FTRAN,DRV,COL,FPEX,TAPE,PART,RECD)
      CALL CONCAT(FNAME,FTRAN,FTYP)
      WRITE(1,309) FNAME

C-
C  NOW OPEN FILE
C
      IF (IOWRIT(11,2,0,FNAME)) GOTO 910

C
C  TRANSFER CONTENTS INTO FILE CONSISTENT WITH USERS MANUAL
C
      WRITE(11,204) ITAPE,IREC,ITSREC,HB2B(12)+1
      WRITE(11,205) (IDATI(I),I=1,5)
      WRITE(11,205) (IDATO(I),I=1,5)
      IDA=HB2B(17)*10 + HB2B(18)
      IHR= HB2B(19)*10 + HB2B(20)
      IMIN=HB2B(21)*10 + HB2B(22)
      WRITE(11,203) IDA,IHR,IMIN
      WRITE(11,202) ISAMP,ICOMP

C
C  THE FOLLOWING DUMMY VALUES ARE WRITTEN IN A SPACE THAT WAS ORIGINALLY
C  INTENDED FOR SCALE FACTORS
C
      DUM=-9999.

```

```

DO 50 I=1,10
WRITE (11,201) DUM,DUM,DUM,DUM,DUM,DUM,DUM,DUM
50 CONTINUE
C-
C- WRITE COMMENTS TO HEADR FILE
C-
60 CONTINUE
READ (7,704,ENDFILE=70) COMMNT
704 FORMAT(A80)
WRITE (11,200) COMMNT
GOTO 60
70 CONTINUE
C NOW CLOSE ALL FILES AND RETURN
IF (IOCLOS(7)) GOTO 940
IF (IOCLOS(11)) GOTO 930
RETURN
80 STOP
C ERROR TRAPS
900 CONTINUE
WRITE (1,304)
GOTO 80
910 CONTINUE
WRITE (1,305)
GOTO 80
920 CONTINUE
WRITE (1,306)
GOTO 80
922 CONTINUE
WRITE (1,310)
GOTO 80
930 CONTINUE
WRITE (1,307)
GOTO 80
940 CONTINUE
WRITE (1,308)
C-
C- *****
C-
101 FORMAT(5I0)
102 FORMAT(A0)
200 FORMAT(1X,A80)
205 FORMAT(1X,5(I10))
201 FORMAT(1X,8(F10.0))
203 FORMAT(1X,3(I10))
204 FORMAT(1X,4(I10))
202 FORMAT(1X,2(I10))
304 FORMAT (" error opening input header file")
305 FORMAT (" error opening output header file")
306 FORMAT (" error closing input header file")
307 FORMAT (" error closing output header file")
308 FORMAT (" error closing contol.tmp file")
309 FORMAT (" ",A15)
310 FORMAT(" error opening control file")
END
C-

```

C- FUNCTION TO RETURN VALUE OF ALPHA STRINGS LEFT JUSTIFIED IN FIELD

C-

CHARACTER*6 FUNCTION STRCHK(ALPARG)

CHARACTER*6 ALPARG,SUBSTG

CHARACTER*1 ALPTST

IPTRL=0

20 CONTINUE

IPTRL=IPTRL+1

ALPTST=SUBSTG(ALPARG,IPTRL,IPTRL)

IF (ALPTST.EQ." ") GOTO 20

IF (ALPTST.EQ."0") GOTO 20

IF (IPTRL .GT.6) GOTO 50

STRCHK=SUBSTG(ALPARG,IPTRL,6)

RETURN

50 WRITE(1,100)

100 FORMAT(" error in string or blank string: sr STRCHK",/, "

2 record number will be \$\$")

STRCHK="\$\$"

RETURN

END

MODIFIED ALLOY DS100
TAPE CONTROL SOURCE CODE

```

.TITLE "{ALLOY QANTEX 401 TAPE UTILITY}- REV-{Q5.x1}"
.SBTTL "ASSEMBLY CONTROL"
.I8080 ; ALLOW ONLY 8080 OPERATORS
.PABS ; ABSOLUTE ASSEMBLY FORMAT
.XLINK ; SUPPRESS LINKER DATA
.PHEX ; HEXIDECIMAL OBJECT GENERATION
.XSYM ; DELETE SYMBOL TABLE
.SALL ; INHIBIT MACRO EXPANSIONS
;
;
;
; COPYRIGHT 1981 BY:
;
; ALLOY ENGINEERING COMPANY, INC.
; COMPUTER PRODUCTS DIVISION
; 12 Mercer Road
; Natick, Ma. 01760
; (617) 655-3900
;
;
; This DOCUMENT contains INFORMATION
; which is PROPRIETARY to ALLOY ENGINEERING
; COMPANY, INC. REPRODUCTION or USE WITHOUT
; an EXPRESS WRITTEN CONSENT from the ALLOY
; ENGINEERING COMPANY IS PROHIBITED.....
;
; DOCUMENT # {FW-100065:MM}
;
.PAGE
.SBTTL "REVISION HISTORY"
; {REVISIONS:}
;
; REV----DATE----DETAILS
;
; Q5.x0 06/30/81 BASE VERSION FROM ATU4
;                REV 4.x3 WITH MODIFICATIONS
;                FOR BI-DIRECTIONAL OPERATION
; Q5.x1 04/25/83 CORRECT REWIND CIRCUIT TO
;                PERFORM REW IF AT LOGICAL BOT
;                IF FLG IS SET

.PAGE
.SBTTL "MISC EQUATES AND MAPPING"
;---- UNIQUE INSTRUCTIONS ----
;
.DEFINE CLA=[.BYTE 0AFH]
; CLEAR ACCUMULATOR (XRA A)
;
;
;---- ALLOY SPECIFIC EQUATES ----
;
ATU    =1800H ; ALLOY TAPE UTILITY
;          ; PROGRAM START ADDR.
;
P      =\ "I/O PORT GROUP?"

```

```

; (16 LONG)
;
R.MBA =2900H ; MASK BUFFER AREA
R.MBL =16 ; MASK BUFFER LENGTH
;
R.DBA =3000H ; DATA BUFFER AREA
R.DBL =8208 ; DATA BUFFER LENGTH
;
;
;
;----- PHYSICAL RECORD I.D.'S -----
;
FMKID =55H ; FILE MARK I.D.
RID =22H ; NORMAL RECORD I.D.
EORID =01H ; NORMAL END OR RECORD I.D.
DSID =08H ; DEI SYNC I.D.
PSID =0FFH ; PCI SYNC I.D.
MSYN1 =0BCH ; SINGLE SYNC MODE FOR PCI
MSYN2 =03CH ; DOUBLE SYNC MODE FOR PCI
RCNT =25 ; RETRY COUNT + 1
RRCNT =50 ; READ RETRY COUNT
.PAGE
.SBTTL "PROGRAM OPERATION"
.REMARK "

```

THIS PROGRAM CONFORMS TO THE SPECIFICATIONS OF THE 'Alloy Tape Utilities'. PLEASE REFER TO THIS DOCUMENT FOR FURTHER INFORMATION"

```

.PAGE
.SBTTL "{ERROR CODES}"
.REMARK " WHEN THE SUBROUTINES RETURN TO THE CALLER,
THE (B) REGISTER CONTAINS INFORMATION RELATED TO
THE SOURCE OF THE ERROR. THIS ERROR CODE IS VALID
ANY TIME THE SUBROUTINE RETURNS WITH THE (CY) SET."

```

```

;----- ERROR CODES FOLLOW: (B) -----
;
;**** CODES 0-3 WILL "ABORT WITHOUT MOTION" ****
;
E0 =0 ; WARNING— SELECTED DRIVE HAS
; EXECUTED AUTO-REWIND SEQUENCE
; SINCE PREVIOUS INIT OR REWIND
; CMD. ISSUE REWIND TO CLEAR.
;
E1 =1 ; WRITE OPERATION REQUEST TO A
; WRITE-PROTECTED DRIVE.
;
E2 =2 ; CMD. TO NON-PRESENT DRIVE OR
; DRIVE WITH CARTRIDGE REMOVED.
;
E3 =3 ; DRIVE FAILED TO RESPOND TO THE
; REQUESTED COMMAND. I.E. BACK-
; SPACE AT BOT ETC.

```

```

;
;**** CODES 6-13 WILL "ABORT WITH MOTION" ****
;
E6      =6      ; FILE-MARK VERIFICATION FAILURE
          ; AFTER WRITING IT.
;
E7      =7      ; TRANSPORT ABORT PRIOR TO COMMAND
          ; COMPLETION. I.E. SKIP RECORD ON
          ; BLANK TAPE
;
E8      =8      ; READ FAIL- MISSING DATA OR FMK.I.D.
;
E9      =9      ; READ FAIL- BAD LRCC
;
E10     =10     ; READ FAIL- SHORT RECORD ERROR
;
E11     =11     ; READ FAIL- BAD VERTICAL PARITY
;
E12     =12     ; WRITE FAIL- R-A-W VERIFY ERROR
;
E13     =13     ; WRITE FAIL- READ DATA NOT
          ; DETECTED PRIOR TO RECORD
          ; WRITE OPERATION COMPLETE.

```

```

;
;**** CODE 14 WILL REPORT A FILE MARK ****
;

```

```

E14     =14     ; READ FAIL- FILE MARK DETECTED

```

```

.PAGE

```

```

.SBTTL "{I/O ASSIGNMENTS}"

```

```

;NOTE: SEE DEI MANUAL FOR BIT EXPLANATIONS.
;

```

```

IDEIS1  =P+0    ; INPUT DEI STATUS-1
;
;

```

```

;      B0-      SLD      ; DRIVE IS SELECTED
;      B1-      BSY      ; DRIVE BUSY
;      B2-      WND      ; DRIVE WRITE LOGIC ON
;      B3-      FUP      ; FILE IS PROTECTED (SAFE)
;
;      B4-      FLG      ; REWIND HAS OCCURRED
;      B5-      EWS      ; TAPE EARLY END WARNING
;      B6-      LPS      ; TAPE IN LOAD-POINT AREA
;      B7-      RDY      ; DRIVE 'ON WITH CART.'
;

```

```

ODEIL1  =P+5    ; OUTPUT DEI LATCH-1
;

```

```

;      NOTE: THE SENSE OF THESE BITS IS LOW=TRUE
;
;

```

```

;      B0-      REV*     ; MOVE REVERSE
;      B1-      FWD*     ; MOVE FORWARD
;      B2-      HSP*     ; MOVE AT HIGH-SPEED
;      B3-      WEN*     ; SET WRITE-ENABLE
;
;      B4-      PASS-THROUGH BIT* ; ALLOW FLG OP'S
;

```

```

;      B5-      RWD*      ; REWIND THE DRIVE
;      B6-      NOT USED
;      B7-      SLG (HIGH TO SELECT DRIVE)
;
;
IMSR   =P+1      ; INPUT MISC. STATUS REG.
;
;      B0-B3    NOT ASSIGNED
;
;      B4-      PASS-THROUGH BIT
;      B5-      DATA DETECTED
;      B6-      PCIL-TXRDY
;      B7-      PCIL-RXRDY
;
;
ODEIL2 =P+4      ; OUTPUT DEI LATCH-2
;
;      NOTE: THE SENSE OF THESE BITS IS LOW=TRUE
;
;      B0-      LED1* (ON LINE)
;      B1-      LED2* (FAULT)
;      B2-      RESERVED
;      B3-      SL1*
;
;      B4-      SL2*
;      B5-      TR1*
;      B6-      TR2*
;      B7-      SL4*
.PAGE
;NOTE: SEE SIGNETICS SPECIFICATION (2651) FOR
;      BIT ASSIGNMENTS ETC.
;
;
OPCIDR =P+12     ; OUTPUT PCI DATA REGISTER
;
OPCISR  =P+13     ; OUTPUT PCI SYNC REGISTER
;
OPCIMR  =P+14     ; OUTPUT PCI MODE REGISTERS
;
OPCICR  =P+15     ; OUTPUT PCI COMMAND REGISTER
;
;
IPCIDR  =P+8      ; READ PCI DATA REGISTER
;
IPCISR  =P+9      ; READ PCI STATUS REGISTER
;
IPCIMR  =P+10     ; READ PCI MODE REGISTERS
;
IPCICR  =P+11     ; READ PCI COMMAND REGISTER
.PAGE
.SBTTL "{ATU PROGRAM ENTRY}"
;
;*****
;      .LOC      ATU
;

```

```

; NOTE: SET-UP MA/PA/CA PRIOR TO CALL
; ALL PRIMARY REGISTERS SAVED U USED
; BUT RESTORED ON RETURN
;

```

```

ATUS:  PUSH   PSW      ; SAVE CALLING REG'S
        PUSH   B
        PUSH   D
        PUSH   H
        LXI   H,0      ; SAVE CURRENT STACK LOC
        DAD   SP      ; IN CASE OF TROUBLE
        SHLD  SSAVE
;

```

```

        LDA   CA      ; GET THE REQUESTED CMD.
        STA   CSR      ; SAVE FOR COORDINATION
        ANI   17H     ; STRIP TO ACTUAL CMD
        CPI   2        ; WRITE COMMAND?
        JZ    WDLRC    ; IF SO, GO CALCULATE LRC
        ANI   10H     ; CHECK IF "RESET COMMAND"
        JZ    SAP      ; NO INIT IF ZERO
        CALL  INIT     ; GO DO TAPE INIT THINGS
        JMP   RTRAN    ; EXIT TO CALLER
;

```

```

.PAGE

```

```

;----- HERE WE CHECK IF RE-TRANSMIT COMMAND -----
;

```

```

SAP:   LDA   CA      ; GET THE COMMAND
        ANI   7FH     ; CHECK IF RE-TRANSMIT
        JZ    RTRAN    ; IF SO, JUST DO IT
;

```

```

;----- HERE WE PRE-PROCESS MA -----
;

```

```

        LDA   MA      ; GET CALLING MODE ARGUMENT
        ANI   8FH     ; REMOVE TRANSIENT BITS
        ORI   80H     ; SET DRS-232 COMPATIBLE BIT
        STA   IS      ; SAVE INITIALIZED ISW
        MVI   A,80H   ; GET NAKED DRIVE STATUS
        STA   DS      ; SAVE INITIALIZED DSW
;

```

```

;----- HERE WE SELECT THE DRIVE & TRACK -----
;

```

```

        LDA   MA      ; SET DRIVE & TRACK
        MOV   B,A
        CALL  SDAT
;

```

```

;----- HERE WE PRE-PROCESS CA -----
;

```

```

        LDA   CA      ; GET THE COMMAND
        MOV   B,A      ; SAVE FOR LATER
;

```

```

;----- HERE WE INIT THE RETRY COUNT -----
;

```

```

        ANI   20H     ; CHECK AUTO-REWRITE BIT
        MVI   A,RCNT ; IF FALSE, DONT'T DISABLE
        JZ    ..B
        SUI   RCNT-1 ; SET RETRY TO 1
..B:   STA   RETRY

```

.PAGE

;----- HERE WE DECODE THE SPECIFIED COMMAND -----
;

```
MOV    A,B    ; GET COMMAND
ANI    40H    ; RELATIVE TO BOT?
JZ     ..C    ; IF NOT, NO REWIND
CALL   REWIND ; IF SO, REWIND
JC     ABORT  ; CHECK ABORT CAUSE & REPORT
..C:   LDA    CA    ; GET COMMAND BACK
MOV    B,A    ; STORE FOR LATER
ANI    80H    ; CHECK RESERVED COMMAND
JNZ    RDIAG  ; GO DO RAM DIAGNOSTIC READ
MOV    A,B    ; GET CA BACK
ANI    0FH    ; MASK DOWN TO COMMAND
JZ     NOC    ; IF ZERO, GO SET THAT FLAG
CPI    8      ; CHECK EXTENDED COMMAND BIT
JNC    EXT    ; IF SET, GO EXTEND
DCR    A      ; READ?
JZ     RR     ; IF SO, GO READ
DCR    A      ; WRITE?
JZ     WR     ; IF SO, GO DO THAT
DCR    A      ; WRITE FILE MARK?
JZ     WFMR   ; IF SO, GO DO THAT
```

;
;--- HERE IS LOW SPEED COMMAND 4-7 DISPATCH ---

;
JTRLS: MVI D,0FFH ; SET FOR LOW SPEED

;
;----- HERE WE DISPATCH ON CMD 4-7 -----

```
JTR:   LXI    H,SRET ; GET RETURN ADDR FOR SUBS
        PUSH  H      ; SAVE AS RETURN PC
        LDA   PA     ; GET POSITIONAL ARGUMENT
        MOV  C,A    ; PUT IN C IN CASE IT'S USED
        MOV  A,B    ; GET THE COMMAND BYTE
        ANI  7      ; MASK TO COMMAND PROPER
        CPI  6      ; CHECK IF FORWARD OR REVERSE
        MOV  A,D    ; GET SPEED CALLING ARGUMENT
        JC   SRF    ; DO FORWARD ON 4 OR 5
        JMP  SRR    ; DO REVERSE ON 6 OR 7
```

;
;----- HERE WE RETURN FROM CMD 3-7 -----

;
SRET: JC ABORT ; GO SORT OUT REASON ABORTED

;
;----- FALL INTO TRANSMIT ROUTINE -----

.PAGE

.SBTTL "{TRANSMIT} DATA CIRCUIT"

;----- HERE WE DEVELOP THE DS WORD -----

```
Cpra:  LDA    DS    ; GET THE DS WORD
        MOV  B,A    ; SAVE FOR LATER USE
        IN   IDEIS1
        RRC
        RRC
```

```

RRC
ANI 1FH ; ISOLATE DEI USABLE BITS
ORA B
MOV B,A ; SAVE UPDATED STATUS
ANI 0CH ; CHECK FOR EOT OR BOT
JZ ..A ; NO MOD IF NEITHER
LDA MA ; CHECK THE TRACK
ANI 1
JZ ..A ; NO MOD FOR TK 0 OR 2
MVI A,0CH ; REVERSE ROLES OF EOT
XRA B ; AND BOT ON TK 1 OR 3
MOV B,A
..A: MOV A,B ; GET THE STATUS
STA DS ; SAVE DRIVE STATUS
STA ODS ; SAVE AS OLD DRIVE STATUS

```

```

;
;----- WE ENTER HERE TO EXIT TO MAIN CALLER -----
;

```

```

RTRAN: LDA CSR ; GET SAVE CA VALUE
STA CA ; RESTORE CALLERS'
LHLD SSAVE ; RESTORE STACK -IN CASE
SPHL
POP H ; AND ALL REG'S
POP D
POP B
POP PSW
RET ; RETURN TO MAIN PROGRAM

```

```

.PAGE
.SBTTL "{COPY} SUBROUTINE"

```

```

;----- CALL
;
RET CY=1 IF ABORTED
;

```

```

;----- DYNAMICALLY D= DRIVE 1- SELECTOR
;
E= DRIVE 2 SELECTOR
;

```

```

COPYS: LXI D,0004H ; DRL/2 TK 0 INITIAL VALUE
CLA ; CLEAR FMK/ DONE FLAG
STA FMFLG

```

```

;
;----- HERE WE REWIND BOTH DRIVES -----
;

```

```

..A: MOV B,D ; SELECT DRIVE 1
CALL SDAT
CALL REWIND
LDA ODS ; GET OLD DRIVE STATUS
RLC ; CHECK RDY
JNC ..A1 ; IF NOT, VERIFY MODE
MOV B,E ; SELECT DRIVE 2
CALL SDAT
CALL REWIND

```

```

;
;----- HERE WE CLEAR FLG CONDITION ON DRIVE 2 -----
;

```

```

CALL WRDY ; WAIT FOR DRIVE READY
MVI A,0FDH ; GET DRIVE FOWARD

```



```

OUT      ODEILL
..A1: MVI  A,0EFH ; IDLE & PTB FOR DRIVE 1
      OUT  ODEILL
;
;----- CHECK IF ALL DONE -----
;
      LDA  FMFLG ; GET FILE MARK FLAG
      ANI  4     ; CHECK ALL DONE FLAG
      RNZ          ; RETURN, JOB IS COMPLETE
      STA  FMFLG ; IF NOT, CLEAR FMK FLAG
;
;----- HERE TO READ SOURCE RECORD -----
;
..B:  PUSH  D      ; SAVE DE PAIR
      MOV  B,D     ; SELECT SOURCE DRIVE
      CALL SDAT
      MVI  A,RCNT ; GET RETRY CONSTANT
      CALL RRS    ; DO READ/RETRY
      POP  D      ; GET SELECTOR BACK
      PUSH D      ; BUT KEEP IT SAFE
      JNC  ..E    ; GO CONTINUE ON NO ERROR
      MOV  A,B     ; GET ERROR CODE
      CPI  14     ; WAS IT A FILE MARK
      JZ   ..C    ; IF FILE MARK, WRITE SAME
      POP  D      ; GET SELECTOR BACK
      STC          ; SET CARRY FOR ERROR RET
      RET          ; ABORT IF NOT FMK
;
.PAGE
;----- HERE TO WRITE A RECORD ON DESTINATION -----
;
..E:  CLA          ; CLEAR FILE MARK FLAG
      STA  FMFLG
      LDA  ODS    ; IF VERIFY MODE TO ..E1
      RLC
      JNC  ..E1
      MOV  B,E     ; SELECT DESTINATION DRIVE
      CALL SDAT   ; GO SELECT
      MVI  A,RCNT ; GET RETRY CONSTANT
      CALL WRS    ; GO WRITE/RETRY
..E1: POP  D      ; GET SELECTOR BACK
      RC          ; ABORT IF CY=1
      JMP  ..B    ; LOOP TO GET NEXT RECORD
;
;----- HERE TO WRITE A FILE MARK -----
;
..C:  LDA  ODS    ; IF VERIFY MODE TO ..C1
      RLC
      JNC  ..C1
      MOV  B,E     ; GET DESTINATION DRIVE
      CALL SDAT   ; SELECT IT
      MVI  A,RCNT ; GET RETRY COUNT
      CALL WEMRS  ; GO WRITE FMK/RETRY
..C1: POP  D      ; GET SELECTORS BACK
      RC          ; ABORT IF CY=1
;

```

```

;---- CHECK IF 1ST OR 2ND FILE MARK ----
;
LDA    FMFLG    ; FMK ALREADY SET
ORA    A        ; SET PSW FOR LDA
JNZ    ..D      ; IF SO, BUMP TRACK
ORI    1        ; IF NOT, SET IT
STA    FMFLG
JMP    ..B      ; GO GET ANOTHER RECORD
;
;---- HERE SECOND FMK IN SUCCESSION ----
;
..D:   INR      D        ; BUMP SOURCE TRK.
      INR      E        ; BUMP DEST. TRK.
      MOV     A,D      ; GET DEST. TK
      CPI    4        ; CHECK IF LAST + 1
      JNZ    ..A      ; IF NOT, GO DO THIS TRACK
      STA    FMFLG    ; IF SO, SAVE IT
      DCR    D        ; DEC. TO PREV. TRK
      DCR    E
      JMP    ..A      ; GO REWIND & EXIT
;
.PAGE
.SBTTL "{EXTENDED} COMMAND ROUTINES"
;---- EXTENDED COMMAND HANDLER ----
;
EXT:   ANI     7        ; MASK TO COMMAND PROPER
      JZ      WRRS     ; WAIT READY & SEND DS/IS
                        ; ON ZERO COMMAND
      CPI    4        ; SEE IF HSPD SEARCH
      JC     ..A      ; IF NOT, GO SEE WHAT
      LDA    PA        ; GET THE POSITIONAL ARG.
      ORA    A        ; SET THE PSW BITS
      JZ     JTRLS    ; GO SET LOW SPEED ON
                        ; SINGLE SEARCH
..B:   MVI     D,0FBH   ; SET HSP* TRUE
      JMP    JTR      ; AND GO CALL THE ROUTINES
;
;---- CHECK IF COPY/MASK OR WRITE-RANGE ----
;
..A:   DCR     A        ; CHECK IF COPY COMMAND
      JNZ    ..C      ; IF NOT, TRY NEXT
;
      CALL   COPYS    ; CALL THE COPY SUBROUTINE
      JC     AWA      ; IF BAD, SIGNAL ABORT/ATMTT
      MVI    A,80H    ; INITIALIZE ISW & DSW
      STA    DS
      STA    IS
      JMP    CPRA     ; AND GO RETURN
;
..C:   DCR     A        ; CHECK IF WRITE-RANGE
      JZ     SRE      ; IF SO, IT IS IN ERROR
; FALL INTO MASK SEARCH ROUTINE
.PAGE
.SBTTL "{MASK} ROUTINES"
;---- MASK DATA SEARCH ROUTINE ----
;

```

```

;---- CHECK IF 1ST OR 2ND FILE MARK ----
;
LDA    FMFLG    ; FMK ALREADY SET
ORA    A        ; SET PSW FOR LDA
JNZ    ..D      ; IF SO, BUMP TRACK
ORI    1        ; IF NOT, SET IT
STA    FMFLG
JMP    ..B      ; GO GET ANOTHER RECORD
;

```

```

;---- HERE SECOND FMK IN SUCCESSION ----
;
..D:   INR      D        ; BUMP SOURCE TRK.
        INR      E        ; BUMP DEST. TRK.
        MOV     A,D      ; GET DEST. TK
        CPI     4        ; CHECK IF LAST + 1
        JNZ    ..A      ; IF NOT, GO DO THIS TRACK
        STA    FMFLG    ; IF SO, SAVE IT
        DCR    D        ; DEC. TO PREV. TRK
        DCR    E
        JMP    ..A      ; GO REWIND & EXIT
;

```

```

.PAGE
.SBTTL "{EXTENDED} COMMAND ROUTINES"
;---- EXTENDED COMMAND HANDLER ----
;

```

```

EXT:   ANI      7        ; MASK TO COMMAND PROPER
        JZ      WRRS     ; WAIT READY & SEND DS/IS
        ; ON ZERO COMMAND
        CPI     4        ; SEE IF HSPD SEARCH
        JC      ..A      ; IF NOT, GO SEE WHAT
        LDA    PA        ; GET THE POSITIONAL ARG.
        ORA    A        ; SET THE PSW BITS
        JZ      JTRLS    ; GO SET LOW SPEED ON
        ; SINGLE SEARCH
..B:   MVI      D,0FBH   ; SET HSP* TRUE
        JMP     JTR      ; AND GO CALL THE ROUTINES
;

```

```

;---- CHECK IF COPY/MASK OR WRITE-RANGE ----
;
..A:   DCR      A        ; CHECK IF COPY COMMAND
        JNZ    ..C      ; IF NOT, TRY NEXT
;
        CALL   COPYS    ; CALL THE COPY SUBROUTINE
        JC    AWA       ; IF BAD, SIGNAL ABORT/ATTMT
        MVI   A,80H     ; INITIALIZE ISW & DSW
        STA  DS
        STA  IS
        JMP  CPRA      ; AND GO RETURN
;

```

```

..C:   DCR      A        ; CHECK IF WRITE-RANGE
        JZ      SRE      ; IF SO, IT IS IN ERROR
; FALL INTO MASK SEARCH ROUTINE

```

```

.PAGE
.SBTTL "{MASK} ROUTINES"
;---- MASK DATA SEARCH ROUTINE ----
;

```

```

MDS:  MVI    A,RCNT  ; GET RETRY CONSTANT
      CALL   RRS     ; GO READ A RECORD
      JC    ABORT   ; IF ERROR, ABORT/ATTMT
      LXI   H,R.DBA ; SET PTR TO RECORD
      LXI   D,R.MBA ; AND ONE TO MASK
..A:  LDAX   D       ; GET MASK CHAR
      MOV   B,A     ; SAVE THIS
      CPI   "?"     ; IS IT A "?"?
      JZ    ..B    ; IF SO, IT MATCHES
      MOV   A,M     ; GET CHARACTER FROM REC
      ANI   07FH   ; CONSIDER ON LS 7 BITS
      CMP   B       ; COMPARE WITH MASK CHAR
      JNZ   MDS    ; GET NEXT REC. ON NO MATCH
..B:  INX   D       ; INC TO NEXT BYTE
      INX   H
      LXI   B,R.MBA+R.MBL ; FETCH END MARKER
      MOV   A,C     ; GET LS BYTE
      CMP   E       ; COMPARE TO DYNAMIC
      JNZ   ..A    ; LOOP IF NOT EQUAL
      MOV   A,B     ; GET MS BYTE
      CMP   D       ; COMPARE TO DYNAMIC
      JNZ   ..A    ; LOOP TILL EQUAL
      JMP   CPRA   ; GO SEND ON MATCHING REC.

```

.PAGE

.SBTTTL "MISC. {STANDARD} COMMAND ROUTINES"

;----- HERE WE DETERMINE LRC FOR RECORD -----

;

WDLRC: LHL D WRDCNT ; GET DATA BUFFER LENGTH

XCHG

LHL D R.AREA ; AND BUFFER LOCATION

MVI B,0 ; INITIALIZE LRC VALUE

..A: MOV A,M ; GET THE DATA

XRA B ; UPDATE LRC

MOV B,A ; RESTORE IT

INX H ; INC THE POINTER

DCX D ; DECREMENT COUNTER

MOV A,D ; SEE IF DONE READING

ORA E

JNZ ..A ; IF NOT, LOOP A WHILE

MOV A,B ; GET ACCUMULATED LRC

STA LRC ; SAVE FOR WRITE USE

JMP SAP ; GO SEND & PROCESS

;

;----- NOP ROUTINE -----

;

NOC: LDA DS ; GET DRIVE STATUS

ORI 20H ; SET REWIND FLAG

STA DS

JMP CPRA ; AND EXIT

;

;----- HERE IS RECORD/WRITE CALLER -----

;

WR: MVI B,20H ; SET EOT AS DEFAULT CHECK

LDA MA ; CHECK THE TRACK

ANI 1

```

JZ      ..A      ; USE DEFAULT FOR TK 0 OR 2
MVI     B,40H   ; CHECK LPS FOR TK 1 AND 3
..A:   IN       IDEIS1 ; GET DRIVE STATUS
ANA     B       ; CHECK LOGICAL EOT
JNZ     AWOA    ; ABORT W/O ATTEMPT IF TRUE
LDA     RETRY   ; GET SPECIFIED RETRY COUNT
CALL    WRS     ; CALL WRITE/RETRY
JC      ABORT   ; ABORT IF ERROR
JMP     CPRA    ; SEND & PROCESS
;
;----- HERE IS RECORD/READ CALLER -----
;
RR:     MVI     A,RRCNT ; GET RETRY CONSTANT
CALL    RRS     ; CALL READ/RETRY
JC      ABORT   ; ABORT IF ERROR
JMP     CPRA    ; SEND IF DATA
;
;----- HERE IS THE WRITE FILE MARK CALLER -----
;
WFMR:   MVI     A,1    ; ONLY 1 RETRY ON FMK'S
CALL    WFMRS   ; CALL WFMK/RETRY SUB.
JC      ABORT   ; LET ABORT HANDLE ERRORS
JMP     CPRA    ; GO SEND & PROCESS
;
.PAGE
.SBTTL  "{WRITE & WFM/RETRY} SUBROUTINES"
;----- HERE IS THE WRITE/RETRY SUBROUTINE -----
;
;----- CALL A= RETRY COUNT
; RET CY=1 IF ABORT B= ABORT CODE
;
;
WRS:    STA     DRETRY ; MOVE TO DYNAMIC COUNTER
..A:    CALL    WRITE  ; TRY TO WRITE PROPER
RNC     ; RETURN IF NO ERROR
LDA     DRETRY ; GET DYNAMIC RETRY COUNTER
DCR     A      ; DEC RETRY
RZ      ; ABORT. CY=1
STA     DRETRY ; RESTORE UPDATED RETRY COUNT
CALL    CLRTP   ; ELSE CLEAR TAPE
MVI     B,20H   ; SET EOT AS DEFAULT CHECK
LDA     MA      ; CHECK THE TRACK
ANI     1
JZ      ..B      ; USE DEFAULT FOR TK 0 OR 2
MVI     B,40H   ; CHECK LPS FOR TK 1 OR 3
..B:   IN       IDEIS1 ; GET DEI STATUS
ANA     B       ; CHECK LOGICAL EOT
STC     ; SET ABORT JUST IN CASE
MVI     B,1     ; WITHOUT ATTEMPT SIGNAL
RNZ     ; ABORT IF EWS PRESENT
JMP     ..A     ; IF NOT, TRY AGAIN
;
;
;----- HERE IS THE WFM/RETRY SUBROUTINE -----
;
;----- CALL A= RETRY COUNT

```

```

;      RET      CY=1 IF ABORT B= ABORT CODE
;
;
;
WFMRS: STA      DRETRY ; MOVE TO DYNAMIC COUNTER
..A:  CALL      WFM      ; TRY TO WRITE FMK PROPER
      RNC      ; RETURN ON NO ERROR
      LDA      DRETRY ; GET DYNAMIC RETRY COUNTER
      DCR      A        ; DEC RETRY
      RZ      ; ABORT. CY=1
      STA      DRETRY ; RESTORE UPDATED RETRY COUNT
      CALL     CLRTP    ; ELSE CLEAR TAPE
      JMP      ..A     ; AND TRY AGAIN

.PAGE
.SBTTL "{READ/RETRY} SUBROUTINE"
;----- CALL      A= RETRY COUNT
;      RET      CY=1 IF ABORT EXCEPT FMK. B=CODE
;      NON-ZERO IF FMK
;      DSW/ISW SET FOR DATA BLOCK/FMK ETC.
;
RRS:  STA      DRETRY ; SET IN DYNAMIC COUNTER
R.A:  CALL      READ   ; DO ACTUAL READ
      JC       ..B    ; IF ERROR, HANDLE IT
      LDA      IS      ; GET INTERFACE STATUS
      ORI      40H    ; SET DATA BLOCK FOLLOWS
      STA      IS
      CLA      ; NO FMK, SO CLEAR ZERO
      RET      ; RETURN

;
..B:  MOV      A,B    ; GET THE ERROR CODE
      CPI      14     ; WAS IT FMK DETECTED?
      STC      ; SET CARRY FOR RETURN
      RZ      ; IF SO, RETURN WITH IT

;
;----- HERE ERROR, SO CLEAR TAPE AND TRY AGAIN -----
;
R.C:  STC      ; SET CY FOR ABORT
      LDA      DRETRY ; GET DYNAMIC RETRY COUNTER
      DCR      A        ; DEC RETRY COUNTER
      RZ      ; ABORT. CY=1
      STA      DRETRY ; RE-STORE COUNTER
      MVI      A,6     ; AND BACKSPACE
      STA      CA
      MVI      C,0     ; ONCE
      MVI      A,0FFH ; AT LOW SPEED
      CALL     SRR
      JMP      R.A     ; AND TRY AGAIN

;
;----- HERE FOR RAM DIAGNOSTIC READ -----
;
RDIAG: LDA      IS      ; SET DATA FOLLOWS IS ISW
      ORI      40H
      STA      IS
      JMP      CPRA    ; GO SEND THE DATA W/O READ

.PAGE

```

.SBTTL "{ABORT} ROUTINES"

;--- HERE ON SYNTAX/PARITY ERRORS ---

;
SRE: LDA IS ; GET THE ISW
ORI 30H ; SET SYNTAX/PE BITS
STA IS ; SAVE FOR SEND ROUTINE
CALL S.L ; SET ERROR LED
TSTAT: CLA ; CLEAR THE CA TO FAKE
STA CA ; A RE-TRANSMIT
JMP SAP ; GO SEND & PROCESS

;
;--- HERE ABORT WITHOUT ANY ATTEMPT ---

;
AWQA: MVI B,10H ; SET ABORT WITHOUT ATTEMPT
JMP SFL ; GO SET FAULT LED

;
;--- HERE TO DETERMINE IF ATTEMPT WAS MADE ---
;--- B REG. = TAPE MODULE ABORT CODE

;
ABORT: MOV A,B ; GET THE ABORT CODE
STA ECODE ; SAVE IT
CPI 4 ; CHECK IF MOTION
JC AWQA ; IF <4 THEN NO MOTION
; IF >4 FALL TO AWA

;
;--- HERE ABORT WITH CONCEIVABLE ATTEMPT ---

;
AWA: MVI A,0FFH ; STOP TAPE MOTION
OUT ODEIL1
MVI B,20H ; SET ABORT WITH ATTEMPT

;
SFL: LDA IS ; GET INTERFACE STATUS
ANI 0BFH ; CLEAR POSSIBLE BLOCK/FOL.
ORA B ; SET ERROR CODE
STA IS
LDA DS ; GET THE DRIVE STATUS
ANI 40H ; CHECK IF FMK DET.
JNZ CPRA ; IF SO, NO FAULT LED
CALL S.L
JMP CPRA ; GO EXIT

;
;--- SET FAULT LED ---

;
S.L: LDA DTLS ; GET DRIVE SELECT WORD
ANI 0FDH ; SET ERROR LED
OUT ODEIL2 ; AND SEND THIS
RET ; BACK TO CALLER

.PAGE

.SBTTL "{INIT & REWIND}-PROGRAMS"

;--- HERE WE INIT THE PCI ---

;
INIT: IN IPCICR ; READ IT TO CLEAR TO MR(1)
MVI A,MSYN1 ; SET MODE TO SINGLE SYNC
OUT OPCIMR ; SET IT
XRA A ; GET A CLEAR

```

OUT      OPCIMR ; CLEAR MR2
IN       IPCICR ; READ CR TO CLEAR TO SR(1)
MVI     A,PSID ; DITTO ABOVE
OUT     OPCISR  ; SET SYN1 TO A 01
MVI     A,RID  ; SET SYN2 TO REC. ID FOR RAW
OUT     OPCISR  ; SET IT
MVI     A,10H  ; RESET ERRORS
OUT     OPCICR ; SET IN CMD. REG.
;
;----- HERE WE INIT THE D.E.I. -----
;
MVI     A,0FEH ; REV. TO CLEAR ERA F/F
OUT     ODEIL1
MVI     A,0EFH ; IDLE WITH PTB*
OUT     ODEIL1
CALL    DLY.5  ; ALLOW ERASE HEAD TO SETTLE
MVI     A,11010111B ; TK1, DRIVE 1, NO LEDS
STA     DTLS   ; SAVE CODE
OUT     ODEIL2
;
;----- SET DEFAULT RECORD BUFFER LENGTH AND START -----
;
LXI     H,R.DBL ; GET DEFAULT LENGTH
SHLD   WRDCNT  ; SAVE AS ACTIVE
LXI     H,R.DBA ; GET DEFAULT START
SHLD   R.AREA ; SAVE AS ACTIVE
;
;----- NORMAL -OK RETURN FOLLOWS -----
;
EOK:    CLA           ; CLEAR (A) & CY
        RET          ; RETURN TO CALLER -OK
;
;----- HERE TASK IS REWIND CMD. -----
;
REWIND: MVI     B,40H ; LPS CHECK IS DEFAULT
        LDA     MA    ; CHECK THE TRACK
        ANI     1
        JZ      ..C   ; USE DEFAULT FOR TK 0 OR 2
        MVI     B,20H ; CHECK EOT ON TK 1 OR 3
..C:    IN      IDEIS1 ; CHECK THE STATUS
;[.x1] ANA     B      ; AT LOGICAL BOT?
;[.x1] JNZ     ..A    ; IF SO, GO RETURN
        ANI     70H   ;[.x1] MASK TO EWS,LPT,FLG
        XRA     B     ;[.x1] CHECK LOGICAL BOT SET
        ;         ; WITH NONE OF THE OTHERS
        JZ      ..A   ;[.x1] IF SO, REW SUPERFLOUS
        MVI     A,0EFH ; IDLE + PTB*
        OUT     ODEIL1
        CALL    DLY2  ; ALLOW SETTLEING
        CALL    WRDY  ; CHECK READY STATUS
        MVI     A,0EAH ; PTB*, REV*, AND HSP*
        LXI     H,..B ; LOAD RETURN ADDRESS
        ;         ; ONTO STACK FOR ERROR RETURN
        PUSH    H

```



```

CALL    DCAC
..B:    JC      ..D      ; IF ERROR CONDITION,
                ; STACK IS ALIGNED
                ; ELSE REALIGN IT
POP     H
..D:    IN      IDEIS1  ; WAIT FOR DRIVE NOT BUSY
        ANI    2
        JNZ    ..D
        MVI    A,0EFH  ; DRIVE IDLE + PTB*
        OUT    ODEIL1 ; SEND TO DRIVE
..A:    CLA
        RET      ; EXIT OK

```

```

.PAGE
.SBTTL "{SDAT & CLEAR TAPE}"
;---- SELECT DRIVE AND TRACK ----
;

```

```

SDAT:   IN      IDEIS1  ; GET DRIVE STATUS
        ANI    81H      ; CHECK RDY & SLD
        CPI    81H      ; VERIFY BOTH TRUE
        JNZ    ..A      ; IF NOT, ALLOW IMMEDIATE
                ; RE-SELECT TO OCCUR
        IN      IMSR    ; READ MISC. STATUS
        ANI    10H      ; CHECK PTB
        JNZ    ..A      ; IF SO, IMMED. RE-SELECT
        CALL   WRDY    ; GO WAIT FOR READY
..A:    MOV     A,B      ; GET THE BYTE
        INR    A        ; ADJUST TO DEI TRACK CODE
        ANI    3
        RRC
        RRC
        RRC
        MOV    C,A      ; STORE FOR LATER
        MOV    A,B      ; GET MA BACK
        ADI    4H      ; ADJUST DRIVE FOR DEI
        RLC
        ANI    18H
        JNZ    ..C
        ORI    80H
..C:    ORA     C        ; COMBINE WITH TRACK CODE
        XRI    0FFH     ; COMPLEMENT
        ORI    7        ; PRESERVE LEDS
        MOV    C,A      ; STORE IT AGAIN
        LDA    DTLS     ; GET OLD DRIVE/TRACK/LED
        ORI    0F8H     ; SAVE ONLY LED CODES
        ANA    C        ; SET TO NEW DRIVE/TRACK
        STA    DTLS     ; STORE AGAIN
        OUT    ODEIL2  ; AND TELL IT TO THE TAPE
        XRA    A        ; EXIT OK
        RET

```

```

;
;---- CLEAR TAPE GAP PROGRAM ----
;

```

```

CLRTP: MVI    C,0      ; SET COUNTER FOR 1 REC
        MOV    A,C      ; AND CA FOR RECS.
        STA    CA
        MVI    A,0EFH  ; LOW SPEED PLEASE

```

```

CALL    SRR      ; DO NORMAL REVERSE SPACE
JMP     ERASE    ; GO DO ERASE AT CMD. LEVEL

.PAGE
.SBTTL  "{WRITE}-RECORD ROUTINE"
WRITE:  CALL     WRDY    ; CHECK READY STATUS
        CALL     CWEN    ; CHECK WRITE ENABLED
        LDA      LRC     ; GET PRE-DETERMINED LRC
        MOV     C,A     ; SAVE IN C FOR LATER
;
;----- HERE WE SET TO 2-SYN MODE TO RAW TEST -----
;
        IN      IPCICR  ; READ CR TO CLR TO MRL
        MVI     A,MSYN2 ; DOUBLE SYNC MODE
        OUT     OPCIMR  ; SET MODE REG.
;
;----- HERE PREAMBLE IS WRITTEN -----
;
..A:    CALL     WS1     ; WRITE SEQUENCE 1
        MVI     A,0F5H  ; WEN* + FWD*
        CALL     DCAC   ; ISSUE CMD. AND CHECK
        CALL     WS2     ; WRITE SEQUENCE 2
        CALL     WPRE   ; WRITE PREAMBLE
        MVI     A,RID   ; GET NORMAL RECORD I.D.
        OUT     OPCIDR  ; WRITE IT
        LHL    WRDCNT  ; GET THE DATA BUFFER LENGTH
        XCHG
        LHL    R.AREA  ; DITTO STARTING ADDR.
;
;----- WRITE RECORD BODY CIRCUIT W/O DAD -----
;
WCK:    IN      IMSR    ; GET TXRDY
        ADD     A
        ADD     A      ; TXRDY=CY, DAD=MINUS
        JM      WCKE   ; IF DAD, GO TO NEXT LOOP
        JNC     WCK    ; LOOP FOR XMT ALLOW
;
        MOV     A,M     ; GET BYTE FROM MEMORY
        OUT     OPCIDR  ; SEND IT
        DCX    D      ; DEC. THE BYTE COUNTER
        INR    L      ; INC LS POS. INDEX
        JNZ    WCK    ; LOOP TILL COUNTER IS ZERO
        JMP    W.D     ; IF NOT, ERROR OUT
;
;----- WRITE RECORD BODY CIRCUIT WITH DAD -----
;
WCKE:   IN      IPCISR  ; READ TO CLEAR TXDSG
WCK1:   IN      IMSR    ; GET TXRDY
        ADD     A      ; MOVE IT TO SIGN BIT
        JP      WCK1   ; LOOP FOR XMT ALLOW
;
        MOV     A,M     ; GET BYTE FROM MEMORY
        OUT     OPCIDR  ; SEND IT
        INX    H      ; INDEX NEXT BYTE OF DATA
        DCX    D      ; DEC. THE BYTE COUNTER
        MOV     A,D     ; CHECK FOR ZERO COUNTER

```

```

ORA      E
JNZ      WCKL ; LOOP TILL COUNTER IS ZERO
.PAGE
;----- WRITE THE LRC / EOR / POSTAMBLE -----
;
..A:     IN      IMSR ; GET TXRDY
        ADD     A
        JP      ..A ; LOOP TILL READY FOR LRCC
        MOV     A,C ; GET LRCC
        CALL    WBYT ; GO SEND IT
        LXI    H,AMBLE ; GO SEND POSTAMBLE
        MVI    B,8
        CALL    WS
;
;----- HERE WE VALIDATE LRC & EOR -----
;
.REMARK  "WE NOW REGAIN SYNC WITH THE WRITE OP.
        IN PROGRESS. THE RECEIVER HAS BEEN IN
        OPERATION DURING THE ENTIRE WRITE AND
        HAS RETAINED ANY PARITY ERROR HISTORY."
;
W.CS:    MVI     A,4 ; RXEN ONLY
        OUT     OPCICR ; STOP TRANSMITTING
;
W.C:     CALL    RBYT ; GO READ A CHAR.
..E:     XRA     C ; IS THIS THE LRC?
        JNZ     W.C ; IF NOT, LOOP
        CALL    RBYT ; READ ANOTHER CHAR.
..F:     CPI     RID ; IS IT THE RECORD I.D.
        JNZ     ..E ; IF NOT, FORGET LRC
        CALL    RBYT ; READ ANOTHER
        CPI     EORID ; IS IT THE EOR I.D.
        JNZ     ..F ; IF NOT, TRY REC. ID.
;
;----- HERE WE VALIDATE VPE -----
;
        IN      IPCISR ; GET THE PCI STATUS
        ANI     0CH ; CHECK PARITY ERROR/TXDSCG
        JNZ     W.D ; IF ERROR, EXIT
        CALL    DLY2
        MVI     A,0F7H ; WEN* ONLY
        OUT     ODEILL ; STOP THE DRIVE
        RET
;
;----- HERE IS THE WRITE ERROR EXIT -----
;
W.D:     MVI     A,10H ; RESET PCI
        OUT     OPCICR
        MVI     B,E13 ; SIGNAL RCVR. FAILURE
;
WEE:     MVI     A,0F7H ; WEN* ONLY
        OUT     ODEILL ; STOP THE DRIVE
        STC
        RET
.PAGE

```

```

ORA      E
JNZ      WCK1 ; LOOP TILL COUNTER IS ZERO
.PAGE
;----- WRITE THE LRC / EOR / POSTAMBLE -----
;
..A:     IN      IMSR ; GET TXRDY
        ADD     A
        JP      ..A ; LOOP TILL READY FOR LRCC
        MOV     A,C ; GET LRCC
        CALL    WBYT ; GO SEND IT
        LXI    H,AMBLE ; GO SEND POSTAMBLE
        MVI    B,8
        CALL    WS
;
;----- HERE WE VALIDATE LRC & EOR -----
;
.REMARK "WE NOW REGAIN SYNC WITH THE WRITE OP.
        IN PROGRESS. THE RECEIVER HAS BEEN IN
        OPERATION DURING THE ENTIRE WRITE AND
        HAS RETAINED ANY PARITY ERROR HISTORY."
;
W.CS:    MVI    A,4 ; RXEN ONLY
        OUT    OPCICR ; STOP TRANSMITTING
;
W.C:     CALL    RBYT ; GO READ A CHAR.
..E:     XRA    C ; IS THIS THE LRC?
        JNZ    W.C ; IF NOT, LOOP
        CALL    RBYT ; READ ANOTHER CHAR.
..F:     CPI    RID ; IS IT THE RECORD I.D.
        JNZ    ..E ; IF NOT, FORGET LRC
        CALL    RBYT ; READ ANOTHER
        CPI    EORID ; IS IT THE EOR I.D.
        JNZ    ..F ; IF NOT, TRY REC. ID.
;
;----- HERE WE VALIDATE VPE -----
;
        IN      IPCISR ; GET THE PCI STATUS
        ANI    0CH ; CHECK PARITY ERROR/TXD SG
        JNZ    W.D ; IF ERROR, EXIT
        CALL    DLY2
        MVI    A,0F7H ; WEN* ONLY
        OUT    ODEILL ; STOP THE DRIVE
        RET ; RETURN SUCCESSFUL
;
;----- HERE IS THE WRITE ERROR EXIT -----
;
W.D:     MVI    A,10H ; RESET PCI
        OUT    OPCICR
        MVI    B,E13 ; SIGNAL RCVR. FAILURE
;
WEE:     MVI    A,0F7H ; WEN* ONLY
        OUT    ODEILL ; STOP THE DRIVE
        STC
        RET
.PAGE

```

.SBTTL "WRITE SUBROUTINES"

;--- WRITE SEQUENCE -1 ---

;
WS1: MVI A,0F7H ; WEN* - ERASE POWER
OUT ODEIL1
MVI A,10H ; RESET PCI ERRORS
OUT OPCICR
CALL DLY2
RET ; RETURN TO CALLER

;
;--- WRITE SEQUENCE 2 ---

;
WS2: MVI B,40H ; SET LPS AS DEFAULT CHECK
LDA MA ; CHECK THE TRACK
ANI 1
JZ ..B ; USE DEFAULT FOR TK 0 OR 2
MVI B,20H ; EOT CHECK FOR TK 1 OR 3
..B: IN IDEIS1 ; GET DEI STATUS
ANA B ; CHECK LOGICAL BOT
JNZ ..B ; SKIP OVER LOAD POINT
MVI B,66 ; INIT 33 MS. CONSTANT
..A: CALL DLY.5 ; 33 MILLISECOND DELAY
DCR B
JNZ ..A
RET

;
;--- WRITE PREAMBLE ---

;
WPRE: CLA ; LOAD FIRST PCI DATA
OUT OPCIDR
MVI A,37H ; TXEN+RXEN+RE+DTR
OUT OPCICR
MVI B,7 ; SET BYTE COUNTER
LXI H,AMBLE+3 ; AND POINTER

;
;--- SUB. TO SEND PRE/POST AMBLES ---

;
WS: MOV A,M ; GET A BYTE
CALL WBYT ; GO SEND IT
INX H ; INC BYTE POINTER
DCR B ; DEC BYTE COUNTER
RZ ; RETURN IF ALL SENT
JMP WS ; ELSE LOOP SOME MORE

;
AMBLE: .BYTE RID,EORID,0,0,0,0,0,0,DSID,PSID

;
;--- WRITE A BYTE OF DATA ---

;
WBYT: OUT OPCIDR ; SEND BYTE TO DRIVE
..A: IN IMSR ; GET MISC. STATUS
ADD A
RM ; RETURN WHEN READY FOR NEXT
JMP ..A

.PAGE

.SBTTL "{READ}-RECORD ROUTINES"

```

READ:  LHL  WRDCNT ; GET THE DBUFFER LENGTH
        XCHG
        LHL  R,AREA ; DITTO STARTING ADDR.
        CALL WRDY   ; WAIT FOR DRIVE READY
        MVI  A,0FDH ; FWD*
        CALL DCAC   ; ISSUE CMD. & CHECK TAKEN
        MVI  C,0    ; INIT LRC

```

```

;
;----- WAIT 24 MS. BEFORE ALLOWING READ -----
;

```

```

        MVI  B,12   ; CONSTANT FOR 24 MS.
..G:   CALL  DLY2
        DCR   B
        JNZ  ..G    ; LOOP FOR 12 COUNT

```

```

;
;----- HERE WE INIT. THE PCI TO READ -----
;

```

```

        MVI  A,10H  ; RESET ERRORS
        OUT  OPCICR
        MVI  A,4    ; RXEN
        OUT  OPCICR
        MVI  B,0    ; RETURN ON DAD
        CALL WDAD   ; WAIT FOR DATA DETECT
        CALL RBYT   ; GO GET FIRST BYTE
        CPI  RID    ; CHECK IF DATA RECORD
        JZ   S.C    ; IF -OK, CONTINUE
        CPI  FMKID  ; CHECK IF FILE-MK ID.
        CNZ  RBE    ; IF NOT, USE E8 EXIT
        CALL RBYT   ; GET NEXT BYTE
        CPI  EORID  ; IS IT EXPECTED
        CNZ  RBE    ; IF NOT SIGNAL ERROR
        CALL RSTP   ; STOP THE DRIVE
FME:   MVI  B,E14   ; SIGNAL FILE MARK
        LDA  DS     ; GET DRIVE STATUS
        ORI  40H   ; SIGNAL FILE MARK
        STA  DS
        STC
        RET

```

```

;
S.C:   IN   IMSR   ; CHECK RXRDY
        ADD  A
        JNC ..D   ; LOOP TILL READY
        IN  IPCIDR ; READ THE PCI DATA
        MOV  M,A   ; PUT IN RAM
        XRA  C    ; UPDATE LRC
        MOV  C,A   ; RESTORE LRC
        INX  H    ; INC. MEMORY POINTER
        DCX  D    ; DECR. THE WORD COUNT
        MOV  A,D   ; CHECK FOR ZERO CONDITION
        ORA  E
        JNZ  S.C   ; LOOP TILL IT HAPPENS

```

```

.PAGE
;----- HERE BODY OF RECORD IS READ -----
;

```

```

        CALL RBYT ; GO READ LRCC

```

```

CMP      C      ; CHECK AGAINST EXPECTED VALUE
JZ       ..A    ; IF OK, CONTINUE
CALL    RSTP   ; STOP THE DRIVE
MVI     B,E9   ; SIGNAL LRC ERROR
STC
RET

;
;----- HERE RECORD OK ON SIZE & LRC -----
;
..A:    MOV     A,C      ; GET CALCULATED LRC
        STA     LRC     ; SAVE IN CASE OF COPY
        IN     IPCISR  ; READ THE PCI STATUS REG.
        ANI    18H    ; CHECK VPE & OVER-RUN
        JZ     ..B    ; IF OK, CONTINUE
        CALL   RSTP   ; STOP THE DRIVE
        MVI    B,E11  ; SIGNAL VPE ERROR
        STC
        RET

.PAGE
;----- HERE WE HAVE A "WINNER" -----
;
..B:    CALL   RSTP   ; STOP THE DRIVE
        IN     IDEIS1 ; GET DRIVE STATUS
        ANI    60H    ; CHECK LOGICAL EOT
        RZ
        ORI    80H    ; SET MINUS FLAG ON EWS
        RET        ; RETURN TO CALLER

;
..D:    ADD     A      ; CHECK DAD STILL TRUE
        JM     S.C    ; IF SO, LOOP
        CALL   RSTP   ; STOP THE DRIVE
        MVI    B,E10  ; SIGNAL SHORT RECORD ERROR
        STC
        RET

;
;----- HERE TO READ A BYTE -----
;
RBYT:   IN     IMSR   ; GET THE MISC. STATUS
        RLC
        JC     RBA    ; GO INPUT ON RXRDY
        ADD    A      ; CHECK DAD
        JM     RBYT   ; LOOP IF DAD STILL TRUE

;
RBE:    MVI    B,40H  ; SET RETURN DAD*
        CALL   WDAD   ; GO WAIT
        MVI    B,E8   ; GET MISSING DATA/FMK ERROR
        IN     IDEIS1 ; GET DEI STATUS
        ANI    4      ; WERE WE WRITING?
        JZ     SEE    ; IF NOT, EXIT W/O WEN
        MVI    A,0F7H ; IF SO, SET WEN/IDLE
        JMP    SEEL

;
RBA:    IN     IPCIDR ; GET THE DATA
        RET        ; AND BACK TO CALLER

```

```

;
;----- STOP DRIVE MOTION SUBROUTINE -----
;
RSTP: MVI    B,40H    ; RETURN ON DAD*
      CALL   WDAD
      MVI    A,0FFH   ; IDLE
      OUT    ODEILL   ; STOP THE DRIVE
      RET

.PAGE
.SBTTL "{WFM}-WRITE FILE MARK CIRCUIT"
WFM:  CALL   WRDY    ; WAIT FOR DRIVE READY
      CALL   CWEN    ; CHECK WRITE ENABLED
      CALL   WSl     ; WRITE SEQUENCE 1
      MVI    A,0F5H  ; WEN* + FWD*
      CALL   DCAC    ; ISSUE CMD. AND CHECK
      CALL   WS2     ; WRITE SEQUENCE 2
      MVI    B,30    ; GET 2" CONSTANT
..B:  CALL   DLY2
      DCR    B
      JNZ   ..B
      CALL   WPRE    ; WRITE PREAMBLE
      MVI    A,FMKID ; GET FILE MK I.D.
      CALL   WBYT
      MVI    B,7     ; SEND EOR & ZEROS.
      LXI   H,AMBLE+1
      CALL   WS
      MVI    A,4     ; GET RXEN
      OUT   OPCICR  ; STOP THE WRITE
;
;----- HERE WE VERIFY THE FILE MARK -----
;
      MVI    B,0     ; SET RETURN ON DAD
      CALL   WDAD    ; LOOK FOR DATA DETECTED
      CALL   RBYT    ; CHECK FOR FILE MARK I.D.
      CPI    FMKID
      JNZ   ..A     ; IF NONE, GO ERROR
      CALL   FME     ; GO SET FMK IN DS
      MVI    B,30    ; GET 2" CONSTANT
..C:  CALL   DLY2    ; AND CLEAR A GAP
      DCR    B
      JNZ   ..C
      JMP   WSTOP   ; DO NORMAL WRITE STOP
..A:  MVI    B,E6    ; GET ERROR
      JMP   WEE     ; USE WRITE ERROR EXIT

.PAGE
.SBTTL "FWD./REV. SPACE/ERASE CIRCUIT"
;----- SPACE FORWARD -----
;
SRF:  ANI    0FDH    ; FWD*
SSTR: MOV    D,A     ; SAVE THE DIRECTION
      LDA    CA      ; GET THE COMMMAND
      MOV    E,A     ; SAVE IT FOR LATER
      MVI    H,0     ; CLEAR FMK FLAG
      CALL   WRDY    ; CHECK DRIVE RDY
S.A:  MOV    A,D     ; GET THE DIRECTION

```



```

CALL    DCAC    ; ISSUE CMD & CHECK TAKEN
MVI     B,0     ; SET RETURN ON DATA DETECT
CALL    WDAD    ; CALL DATA DETECT SUB
;
;----- HERE TO CHECK IF FILE MARK -----
;
CALL    DLY2    ; DELAY 4 M.S.
CALL    DLY2
IN      IPCISR  ; GET PCI STATUS
ADD     A       ; CHECK DAD*
JM      ..A     ; GO DO RECORD IF HI
MOV     H,E     ; NON-ZERO THE FMK FLAG
MOV     A,E     ; GET THE COMMAND
RRC     ; RECORD OR FILE SEARCH?
JNC     S.E     ; IF RECORD SEARCH, ABORT
..B:    CLA
ORA     C       ; CHECK COUNTER
JZ      SDONE   ; IF ZERO, GO STOP
DCR     C       ; ELSE DEC THE COUNTER
JNZ     S.A     ; IF NOT LAST, LOOP
MOV     A,D     ; GET THE COMMAND
ORI     4       ; SET SLOW SPEED
MOV     D,A     ; RESTORE COUNTER
JMP     S.A     ; GO SEARCH AGAIN
;
;----- HERE FOR RECORD -----
;
..A:    MVI     B,40H ; SET RETURN NO DATA DETECT
CALL    WDAD
MOV     A,E     ; GET THE COMMAND
RRC     ; RECORD OR FILE SEARCH?
JC      S.A     ; LOOP IF FILE SEARCH
JMP     ..B     ; IF REC, DEC THE COUNTER
;
;----- SEARCH EXIT -----
;
SDONE:  MOV     A,D ; GET THE DIRECTION BACK
ANI     1       ; SEE IF REVERSE
JNZ     S.E     ; IF NOT, SKIP RVRS THING
..F:    MVI     B,32 ; GET 16.0 MS. COUNT
..E:    CALL    DLY.5
IN      IPCISR  ; READ THE PCI STATUS
ANI     40H    ; CHECK DAD
JNZ     ..F     ; IF DATA, LOOP TO SKIP IT
DCR     B      ; LOOP FOR THE COUNT
JNZ     ..E
;
;----- STOP THE DRIVE AND EXIT LOGIC -----
;
S.E:    MVI     A,0FFH ; IDLE
OUT     ODEILL ; STOP THE DRIVE
CLA     ; CHECK FMK FLAG
ORA     H
RZ      ; RETURN OK IF LOW
CALL    FME     ; GO SET FMK STATUS

```

```

MOV     A,H      ; SEE IF REC. SEARCH
ANI     1
RNZ     ; IF NOT, RETURN OK
STC     ; ELSE SET CARRY
RET     ; FOR ERROR RETURN
;
;----- SPACE REVERSE -----
;
SRR:    ANI      0FEH ; REV*
        JMP      SSTRT ; GO DO IT
;
;----- ERASE -----
;
ERASE:  CALL     WRDY  ; WAIT FOR DRIVE READY
        CALL     CWEN  ; CHECK WRITE ENABLED
        CALL     WSL   ; WRITE SEQUENCE 1
        MVI     A,0F5H ; GET FORWARD COMMAND +WEN*
        CALL     DCAC  ; ISSUE CMD. & CHECK TAKEN
;
;----- ERASE 1/3 REC SIZE PER R.DBL -----
;
        MVI     B,(R.DBL>8)<2 ; B= APPROX COUNT
;
..A:    CALL     DLY2
        DCR     B
        JNZ     ..A
;
;----- WRITE STOP CIRCUIT -----
;
WSTOP:  MVI     A,0F7H ; WEN* ONLY
        OUT     ODEIL1 ; STOP THE DRIVE
        XRA    A      ; SIGNAL SUCCESSFUL ACTION
        RET
;
.PAGE
.SBTTL  "MISC. SUBROUTINES"
;----- WAIT FOR DRIVE READY -----
;
WRDY:   IN      IPCICR ; READ CR TO CLR TO MRL
        MVI     A,MSYN1 ; REVERT TO SINGLE SYN
        OUT     OPCIMR ; SET IT
        IN      IDEIS1 ; READ DRIVE STATUS
        ANI     81H    ; CHECK RDY + SLD
        CPI     81H
        JNZ     ..C    ; SIGNAL ERROR
..A:    IN      IDEIS1 ; GET STATUS AGAIN
        ANI     2      ; CHECK BSY
        JNZ     ..A    ; LOOP TILL NOT BUSY
        IN      IDEIS1 ; GET DRIVE STATUS
        ANI     10H    ; CHECK FLG BIT
        JZ      ..B    ; IF NOT CONTINUE
        IN      IMSR   ; GET MISC. STATUS
        ANI     10H    ; CHECK PTB
        JNZ     ..B    ; IF SET, ALLOW OPERATION
        MVI     B,E0   ; IF NOT, SIGNAL WARNING

```

```

        JMP     SEE
;
..B:   IN      IDEIS1 ; GET DEI STATUS
        ANI    4      ; CHECK IF WRITING
        JZ     ..D    ; IF NOT, NO UN-LATCH
        LDA    CSR    ; GET THE CMD ARG. AGAIN
        ANI    47H   ; ISOLATE REAL BITS
        CPI    2      ; WRITE-CMD. QUEUED?
        JZ     ..D    ; IF SO, NO UN-LATCH
        MVI    A,0FEH ; UNLATCH ERA F/F
        OUT    ODEIL1 ; BY PULSING REV.
        MVI    A,0FFH ; GET DRIVE IDLE
        OUT    ODEIL1 ; SET IT
        CALL   D.A    ; ALLOW COMMAND TO TAKE
..E:   IN      IDEIS1 ; WAIT TILL NOT BUSY
        ANI    2
        JNZ   ..E
;
..D:   CALL   DLY2    ; ALLOW SETTLEING
        LDA    DTLS   ; GET STORED SELECT WORD
        ORI    2      ; SET IT TO CLEAR FAULT LIGHT
        OUT    ODEIL2
        STA    DTLS   ; AND PRESERVE CODE
        RET
;
..C:   MVI    B,E2    ; CMD. TO BAD DRIVE
        JMP    SEE
.PAGE
;----- CHECK WRITE ENABLED -----
;
CWEN:  IN      IDEIS1 ; READ DRIVE STATUS
        ANI    8      ; CHECK FUP
        RNZ
        MVI    B,E1   ; WRITE NOT ENABLED
        POP    PSW    ; SET FOR ERROR RETURN
;
;----- SUBROUTINE ERROR EXIT -----
;
SEE:   MVI    A,0FFH  ; GET DRIVE IDLE CMD.
SEE1:  OUT    ODEIL1  ; SET IT
        CALL   DLY2   ; DELAY AT LEAST 2 MS.
        POP    PSW    ; POP TO CMD. LEVEL
        STC
        RET          ; RETURN TO CALLER
;
;----- DRIVE COMMAND AND CHECK -----
;
DCAC:  PUSH   PSW     ; SAVE THE COMMAND
        LDA    MA     ; CHECK THE TRACK
        ANI    1
        JZ     ..A    ; NO MOD TO TK 0 OR 2
        POP    PSW    ; GET THE COMMAND
        PUSH   PSW    ; BUT KEEP IT SAFE
        ANI    3      ; CHECK FOR FWD* OR REV*

```

```

CPI      3
JZ       ..A      ; NO MOD IF NEITHER
POP      PSW      ; GET THE COMMAND
XRI      3        ; EXCHANGE REV* AND FWD*
PUSH     PSW      ; SAVE MODIFIED CMD
..A:    POP      PSW      ; GET THE COMMAND
        OUT      ODEIL1  ; NOW DO DRIVE SELECT
        MVI      A,2     ; DELAY FOR SLEW
        CALL     D.A     ; DELAY 20 MICROS
        IN       IDEIS1  ; GET DRIVE STATUS
        ANI      2      ; CHECK BSY
        RNZ      ; RETURN TO CALLER -OK
        MVI      B,E3    ; DRIVE -CMD. REJECTED
        JMP      SEE     ; ERROR SIGNAL ETC.

.PAGE
;----- WAIT FOR DATA DETECTED -----
;
;----- B= 0 TO RETURN ON DAD TRUE
; B= 40H TO RETURN ON DAD FALSE
;
WDAD:   XRA      A      ; CLEAR DEADMAN
        STA      WCTU
        STA      WCTT
        STA      WCTH
..W:    IN       IPCISR  ; GET PCI STATUS
        ANI      40H    ; DCD=DAD*
        XRA      B      ; CALLER OPTION
        JNZ      ..A    ; IF GOOD, VERIFY AFTER WAIT
        LDA      WCTU   ; INCRAMENT DEADMAN
        ADI      1
        STA      WCTU
        JNC      ..B
        LDA      WCTT
        ADI      1
        STA      WCTT
        JNC      ..B
        LDA      WCTH
        INR      A
        STA      WCTH
        CPI      07H    ; TIMEOUT?
        JNZ      ..B    ; CONTINUE IF NOT
        LDA      CA     ; GET CALLING COMMAND
        ANI      07H    ; ISOLATE COMMAND BITS
        CPI      5      ; IS IT FILE SEARCH?
        JZ       ..W    ; IF SO, DON'T ABORT
        POP      PSW    ; DO STACK ADJUST
        JMP      ..C    ; AND SIGNAL ERROR
..B:    IN       IDEIS1  ; READ DEI STATUS
        ANI      2      ; CHECK BSY FALSE
        JNZ      ..W    ; LOOP IF STILL MOVING
..C:    MVI      B,E7    ; SIGNAL TRANSPORT ABORT
        JMP      SEE     ; USE SUBROUTINE ABORT EXIT
;
..A:    MVI      A,10    ; GET 100 MICRO-SEC. CONSTANT
        CALL     D.A

```

```

IN      IPCISR ; READ THE STATUS AGAIN
ANI     40H
XRA     B
RNZ     ; RETURN IF NOT FALSE ALARM
JMP     ..W   ; LET'S TRY IT AGAIN, FOLKS
;
;----- DELAY 2 MILLI-SECONDS -----
;
;-- ASSUMES 4MHZ Z-80 -NO WAIT STATES
;
DLY2:   MVI     A,205 ; GET 39 STATE CONSTANT
D.A:    NOP     ; KILL 9.75 MICRO-SECS.
        NOP
        NOP
        NOP
        DCR     A
        RZ
        JMP     D.A
;
;----- DELAY 0.5 MILLISECONDS -----
;
DLY.5:  MVI     A,52  ; GET CONSTANT FOR .5 MS.
        JMP     D.A   ; USE ABOVE CIRCUIT TO TIME
;
;----- WAIT READY AND RETURN CURRENT STATUS -----
;
WRRS:   CALL    ..A   ; LOWER PGM LEVEL FOR ABORT
        JC      AWOA  ; IF ABORT, SIGNAL SAME
        JMP     CPRA  ; SENT CURRENT DRIVE STATUS
;
..A:    CALL    WRDY  ; WAIT FOR DRIVE READY
        CLA     ; CLEAR CY FOR RETURN
        RET     ; RETURN TO PSEUDO CALLER

.PAGE
.SBTTL "{RAM VARIABLES}"
;*****
        .LOC     ATU+769H
;
;----- RESERVED RAM -----
;
WCTU:   .BYTE   0      ; WDAD DEADMAN COUNTER - UNIT
WCTT:   .BYTE   0      ; WDAD DEADMAN COUNTER - TEN
WCTH:   .BYTE   0      ; WDAD DEADMAN COUNTER - HUNDRED
R.AREA: .WORD    R.DBA  ; READ/WRITE RECORD AREA - START
WRDCNT: .WORD    R.DBL  ; READ/WRITE RECORD SIZE (BYTES)
MA:     .BYTE   0      ; MODE ARGUMENT
PA:     .BYTE   0      ; POSITIONAL ARGUMENT
CA:     .BYTE   0      ; COMMAND ARG.
CSR:    .BYTE   0      ; CA INTERMEDIATE STORAGE
DS:     .BYTE   0      ; DRIVE STATUS
IS:     .BYTE   0      ; INTERFACE STATUS
RETRY:  .BYTE   0      ; INTERNAL RETRY LITERAL
DRETRY: .BYTE   0      ; DYNAMIC RETRY COUNTER

```

```
LRC: .BYTE 0 ; LRC STORAGE
ODS: .BYTE 0 ; OLD DRIVE STATUS
DTLS: .BYTE 0 ; DRIVE/TRK/LED STORAGE
FMFLG: .BYTE 0 ; INTERNAL FMK COORD.
ECODE: .BYTE 0 ; ERROR CODE STORAGE
VER: .BYTE "1" ; VERSION NUMBER (LSB)
SSAVE: .WORD 0 ; STACK SAVE LOCATION
```

```
;  
.END
```

APPENDIX III

TAPE SUMMARY

TAPE INDEX

Tape No.	Date	Time	1-Minute Record	Time Series Record	Remarks on Cartridge	General Remarks	Dates of Interest
1	10/28/82 10/30/82	5:50p 12:15p	no printouts		10-12 kn N. on first several hrs, large swell	6001 = GU, 6000 = OF 6011 = OF, 6013 = O2	10/29 10/31
2	10/30/82 11/01/82	1:00p	no printouts		Written over by accident - dead battery		
3					No tape	System removed 11/1-11/11	
4	11/11/82 11/12/82	4:51p 1:00p	**19 *9,10,13-19	**2 *9,10,13-19	Nothing on track 4	Tide gage not working caused tripping 1-min. rec. printout summary exists, no tape found - low wind speeds.	
5	11/12/82 11/15/82	3:00p	1-8	6,7	No power	Low wind speed	
6	11/15/82 11/17/82	3:15p 2:00p	1-10	1	20 mph, 1 T.S. at beginning of record	Instead of 1 min. Rec's 4,5,6 there is 3 file marks, T.S. for #6. Lost power question about WS staff. Tide gage not working caused tripping.	11/16
7	11/18/82 11/23/82	4:40p 2:30p	*15,5,6-10,12 **14	**3	1 T.S. 1 mph	Track 1 - last 3/4 is blank. Track 4 - prob. searching. Glitches on Track 4. Tide gage not working caused	

- tape stops when

8	11/23/82 3:30p 11/26/82 9:00a	45-61			6006 = 01 10 = 40 13 = 2d Appears to have over- written	11/26
9	11/26/82 9:40a 11/30/82 11:30a	57-97	4 *I don't think so	No reset, 6006 = 3 6010 = 00, 6011 = 61 6013 = 26	No reset	11/29
10	11/30/82	1	1	1 T.S., 1-min. Rec. (degaussed)	Looks like a test tape	
11	11/30/82 3:15p 12/06/82 1:36p	*122-142	129 at least *2	6009 = 26, 10 = 0, 11 = 8E, 13 = 53 (degaussed)	Copy over first records	
12	12/06/82 12/10/82				No printout, references, or tape may not be a tape.	
13	12/10/82 1:00p 12/15/82	3-35	2			
14	12/15/82 3:30p 12/16/82 3:00p	1-11	1-11 *7	50 + mph wind 06 = 3, 11 = 06, 12 = 06	Glitches	12/15
15	12/16/82 3:10p 12/23/82	30-58	8		10 mph wind speed Copy over 1st records	
16	12/23/82 3:02p 12/27/82 4:07p	1-48	4	06 = 02, 11 = 30, 13 = 09	Wind speed 25 mph + on SR 27-29, 31-36 or else zero	12/24
17	12/27/82 4:30p 12/29/82 1:00p	1-22	7,9 at least *3	06 = 02, 11 = 16, 13 = ?		
18	12/29/82 4:40p 01/06/83 11:00a *1:11p	54-94	56,64-69, 76-79	06 = 0, 11 = SE, 13 = 1d	Copy over 1st records. Try reading #79	

19	01/06/83 01/11/83	1:40p 11:50a	40-59	40-52	06 = 1, 11 = 36, 12 = 26	Wind speeds 25 mph + on this tape some some prob. with glitches	1/2 to 1 1/10/83
20	1/11/83 1/12/83					No tape	
21	01/12/83 1/19/83	3:40p	1-20	1&2	Low temperature cartridge		
22	01/19/83 01/25/83	2:30p	24-49	24-49		Low power. Copy over first records - Wind gage not working tripping	1/24
23	01/25/83 01/31/83	3:20p 11:00a	44-70				1/27
24	01/31/83				Boat wake test runs 1-8 see notes		
25	01/31/83				Boat wake test II		
26	01/31/83 02/03/83 *12/05/83	*3:00p 3:30p 12:00n	10-34			Removed system. Wind gage not working, tripping DAS	
27	03/16/83 03/30/83	7:00p				Sent to DC - good data on Ch. 18	
28	03/30/83 04/01/83	3:45p	1-11	11	1 manual T.S. Wind 12 - 15		
29	04/01/83 04/04/83	3:55p 1:00p				Don't know if this tape ever existed	
30	04/02/83	6:00p			Test tape, T.S. + 1-min. + T.S. WS 12-15		

31	04/04/83 04/05/83	6:15p 5:00p	1-19	19	No out date recorded Wind gage
32	04/05/83 04/06/83	5:15p 2:15p	1-20	20	I think this should really to to 4/17 Wind gage/tripped DAS
33	04/07/83	6:10p			WS 15 Test tape
34	04/07/83 04/11/83	5:00p 2:00p	1-34	14	Have gages 16,18,19,20 working
35	04/11/83 04/14/83	4:15p 2:20p	1-34	7 *0	Tr = 1, 1-min. Rec = 224 T.S. = A, N = 0800
36	04/14/83 04/17/83	5:15p 1:30p	1-34	21 *0	?Wind Gage
37	04/18/83 04/19/83	5:45p 3:15	1-10	10	Restarted - original date = 4/17 @ 5:30p
38	04/19/83		1	1	Test tape 1 T.S. & 1 1-min. rec.
39	04/19/83 04/21/83	5:30p 2:00p	1-22	6 *0	A006 = 0, 11 = 16, ? - 6
40	04/22/83 04/26/83	11:00a 4:45p	1-50	lots 16 *0	Lots of good data/storm 5 accelerometers working
41	04/21/83 4/22/83		1	1	Other note in box Test tape 1512, ITS
42	04/22/83			1-8	Boat Make 8 T.S.'s

43	04/26/83 05/03/83	5:00p 2:00p	1-82	7,66 *4 *2	Track 0, 1-min. rec. = 52, T.S. = 2
44	05/03/83 05/12/83	5:00p 11:00a	1-49	8 *9	Lower power
45	05/12/83 05/17/83	1:35p 2:50p	1-22	0	Batteries were dead good wind Saturday night
46	05/17/83 05/24/83	7:45p 3:30p	1-53	37 *1	Batteries were dead
47	05/26/83	5:00p	0	2	Restart tape, original date = 5/24 @ 5:00p Test Tape
48	05/26/83 06/03/83	6:00p 4:00p	1-33	8 11	Dead batteries
49	06/03/83 06/06/83	4:15p 9:30a			
50	06/08/83 06/13/83	1:55p 2:50p	1-21	13	Dead batteries Tr = 4 hrs
51	06/13/83	4:15p		2	Test tape 1-T.S., 1-T.S. stopped short
52	07/06/83 07/07/83	12:00n 10:00a	1-52 *1-10 & 38-52	24	10 rec. in this period overwrote rec. from previous
53	07/08/83 07/12/83	3:30p 2:30p	22-47	27	Wind gage wrong gain
54	07/12/83 07/14/83	5:00p 11:30a 11:00a	1-21	21	Wind gage/wrong gain Good tire BW loads

No T.S.

1 T.S. = 1-min.
Rec. # 7

2 rec. with boat
waves #1 ch's 45-60
org. gain. #2 Ch's
45-60 new gain.

Problem reading 1-min.
rec. on Macsym

10 rec. in this period
overwrote rec. from previous

Wind gage wrong gain

Wind gage/wrong gain
Good tire BW loads

55	07/14/83 07/16/83	3:00p 1:00p	1-23	23		Good tire BM loads
56	07/16/83 07/19/83	1:20p 10:00a			Met! (not the tape)	Good tire BM loads
57	07/19/83 07/20/83	2:30p 11:00a	1-10	10		Wind gage/wrong gain Good tire BM loads
58	07/20/83 07/21/83	4:15p 10:20a	1-9	9	I think I wrote over Track 1	Wind gage/wrong gain Good tire BM loads
59	07/21/83 07/22/83	12:00n 12:22p	1-12	12		Wind gage/wrong gain Good tire BM loads
60	08/18/83				REMOVED TIRE BREAKWATER 8/1/83 Pull test and boat wake 12 T.S. only	
61	08/27/83 08/29/83	12:12p 12:32p	1-23	23		Wind gage?
62	08/29/83 08/30/83 *11/50a	5:35p 10:50a 11:50a	1-9	9		Wind gage?
63	08/30/83 08/31/83	4:15p 10:30a *12:00p	1-9	9		Wind gage?
64	09/02/83 09/06/83	1:20p 12:00p	21-46	26	Bravely retrieved by Dan on 09/26/83	Wind gage?
65	08/31/83 09/02/83	6:00p 11:15a	1-20	20		Wind gage?

66	09/06/83 09/08/83	10:35a 9:40a	1-23	23		Wind gage?
67	09/08/83 09/09/83	1:50p 10:30a	1-20 *1-10	1		73 and 74 = 74
68	09/09/83 09/12/83	3:35p 11:55a	1-34	6		73 and 74 = 74
69	09/12/83 09/14/83	3:50p 9:45a	1-20	20		Wind speed not working
70	09/15/83 09/19/83	4:00p 9:00a	1-44	5	Restart was 9/14 @ 1:30p	Wind speed not working 9/18/83 dropped clump weight
71	09/19/83 09/21/83	3:35p 7:52a	1-20	3		Wind speed not working
71B	09/21/83				Test tape 3-T.S. Princess Margarette on 1st. 1 one-min. rec. which triggered T.S. North wind/white caps just beginning 10-12 mph	
72	09/22/83 09/23/83	6:32p 11:15a	1-8	no tser		Wind speed not working
73	09/23/83 09/27/83	3:35p 12:00n	2	1	Power out when removal	
74	09/29/83		1	8	Test tape, waves - white caps breaking over break- water, WS 18-25 mph from north.	Wind 18 - 25 mph
75	09/29/83 10/03/83	12:30p 1:00p	2	1		Batteries were low at first

88	10/21/83	12:15p	1-5	5	
	10/21/83	2:45p			
89	10/21/83	2:00p			
		2:45p			
90	10/24/83	5:00p	1-9	1	
	10/25/83	11:00a			
91	10/25/83	3:35p	1-11	3	Light north WS
	10/26/83	1:30p			4-5 mph
92	10/26/83	4:45p	1-12	5	S. WS 12 mph started
	10/27/83	2:45p			1-min. rec., T = 1
					T.S. = 05 ₁₆ , MR = 0C
93	10/27/83	*5:50p			
	10/28/83	5:05p			
	10/28/83	5:00p			
	10/29/83	8:43a		15	Reset system 11 = 6
	10/29/83				Boatwake ? 11 = 8
94	10/29/83				
95	10/29/83		*1-25	9	1-min. rec. = 19 ₁₆
	10/31/83	1:00p			T.S. = 9
					Started manually
96	10/31/83	*4:44a	1-14	14	Continuous hour
	11/01/83	4:40p			
		6:27a			
97	11/29/83			2	INSTALLED RUBBER CONNECTION
					Lab test tape
					T.S. #1, 1 volt fn
					T.S. #2 - sys.
98	12/01/83	*3:35a	1-8	1	
	12/02/83	3:35p			
		8:35p			
					00, 08, 01

Some problems at beginning
of tape* had to forward
space in 100 recs. to
get past it

Tape out of order
unable to read

110	12/29/83 01/05/84	1:45p 1:45p	21-63	20
111	01/06/84 01/11/84	1:30a 11:50a	1-57	14
112	01/05/84	16:40		
113	01/12/84 01/07/84	2:05p 10:30a	1-58	14
114	01/19/84 01/20/84	5:00p 10:40a	1-8	2
115	01/20/84 01/23/84	3:45a 11:40a	7-31	27
116	01/23/84 01/23/84	*12:15p 11:40a 12:00a	1-3	3
117	01/23/84 01/24/84	12:15p 11:45a	1-20	20
118	01/24/84 01/26/84	10:55a 11:30a	1-25	25
119	01/26/84 01/27/84	11:40a 9:55 9:45a	1-11	11
120	01/27/84 01/31/84	10:00a 9:45a	23-48	25
121	01/31/84 01/31/84	12:00a 12:00a	1	14
				Tow in

APPENDIX IV
SCALE FACTOR SUMMARY

TRANSDUCER INPUT SUMMARY SHEET

Anchor Forces

Channel No.	Location Code	Location
1	LNWC	Lower Northwest on Concrete Breakwater
2	LNEC	Lower Northeast on Concrete Breakwater
3	UNWC	Upper Northwest on Concrete Breakwater
4	USWC	Upper Southwest on Concrete Breakwater
5	UNEC	Upper Northeast on Concrete Breakwater
6	USEC	Upper Southeast on Concrete Breakwater
7	LSWC	Lower Southwest on Concrete Breakwater
8	LSEC	Lower Southeast on Concrete Breakwater
9	LNT	Lower North Tire Breakwater
10	UNT	Upper North Tire Breakwater
11	UST	Upper South Tire Breakwater
12	LST	Lower South Tire Breakwater

Fixed References

13	SOUTH
14	CENTER
15	NORTH

Waves

16	TIDE	Tide Gage
17	INC	Incident Wave Buoy
18	NW	Northwest Wave Buoy
19	NE	Northeast Wave Buoy
20	SW	Southwest Wave Buoy
21	SE	Southeast Wave Buoy

Dynamic Pressure

22	NU	North Upper
23	NL	North Lower
24	BNC	Bottom North Center
25	BEC	Bottom East Center
26	BEI	Bottom East Inner
27	BCC	Bottom Center
28	BWI	Bottom West Inner
29	BWO	Bottom West Outer
30	BSC	Bottom South Center
31	SCU	South Center Upper
32	SW1	Southwest Number 1
33	SW2	Southwest Number 2
34	SW3	Southwest Number 3
35	SCL	South Center Lower
36	SE3	Southeast Number 3
37	SE2	Southeast Number 2
38	SE1	Southeast Number 1
39	EP1	East Pontoon Number 1
40	EP2	East Pontoon Number 2

TRANSDUCER INPUT SUMMARY SHEET (Continued)

41	EP3	East Pontoon Number 3
42	EP4	East Pontoon Number 4
43	EP5	East Pontoon Number 5
44	EP6	East Pontoon Number 6

Concrete Strain

45	NULE	North Upper Longitudinal on East End of Pontoon
46	NBLE	North Bottom Longitudinal on East End of Pontoon
47	SBLE	South Bottom Longitudinal on East End of Pontoon
48	SULE	South Upper Longitudinal on East End of Pontoon
49	NT1	North Transverse Number 1
50	NT2	North Transverse Number 2
51	BT1	Bottom Transverse Number 1
52	BT2	Bottom Transverse Number 2
53	UT1	Upper Transverse Number 1
54	UT2	Upper Transverse Number 2
55	ST2	South Transverse Number 2
56	ST1	South Transverse Number 1
57	NULC	North Upper Longitudinal at Center of Pontoon
58	NBLC	North Bottom Longitudinal at Center of Pontoon
59	SULC	South Upper Longitudinal at Center of Pontoon
60	SBLC	South Bottom Longitudinal at Center of Pontoon

Accelerometer

61	WVA	West Vertical
62	WHA	West Horizontal
63	WRA	West Rotational
64	EVA	East Vertical
65	EHA	East Horizontal
66	ERA	East Rotational

Relative Motion

67	WVR	West Vertical Rotational Displacement
68	WHR	West Horizontal Rotational Displacement
69	EVR	East Vertical Rotational Displacement
70	EHR	East Horizontal Rotational Displacement
71	CLM	Center Longitudinal Motion
72	CRM	Center Rotational Motion

Wind Speed and Direction

73	WS1	Windspeed at Tide Gage
74	WS2	Windspeed at Instrument House
75	WD1	Windspeed Direction at Tide Gage
76	WD2	Windspeed Direction at Instrument House

Current Velocity

77	N-S	North South
78	E-W	East West

TRANSDUCER INPUT SUMMARY SHEET (Continued)

Not Committed

79
80

Single Scan Only
Not Committed

81
82

Internal System Voltages

83	-15
84	+15
85	-24
86	+24
87	+5
88	+5

Temperature

89	Water
90	Air
91	SCC
92	DAS

At Signal Conditioning Bar
At Data Acquisition System

Not Committed

93
94
95
96

GAIN SETTING

Channels	Scale Factors kips mv	9/29/82	10/4/82	2/21/83	5/26/83		7/14/83	8/31/83		11/30/83		Scale Factor = $\frac{\text{kips}}{\text{mv}} \times \frac{5000 \text{ mv}}{4095 \text{ cts}} \times \frac{1}{\text{Gain}}$
					Old	New		Old	New	Old	New	
1	2.293	300	1000	100	96	new card	1000	1000	1000	980	999	
2	2.284	300	1000	100	-	-	1000	1000	1000	955	1005	
3	4.112	300	1000	100	-	100	1000	1000	1000	1005	1005	
4	2.397	300	1000	100	-	24	1000	1000	1000	1005	1005	
5	2.623	300	1000	100	-	-	1000	1000	1000	1000	1000	
6	2.533	300	1000	100	-	-	1000	3000	2000	-	-	
7	2.353	300	1000	100	-	95	1000	-	-	1024	1003	
8	2.288	300	1000	100	-	85	1000	-	-	-	-	
9	1.504	300 (1000)	300	100	-	98	1000	no cards		.99	1.0	replaced by wave buoys 10/11/83
10	1.555	300 (1000)	300	100	-	-	1000			1.0	1.0	
11	1.501	300 (1000)	300	100	-	58	1000			.156	1.0	
12	1.524	300 (1000)	300	100	-	85	1000			.97	1.0	

Notes: 8/31/83 LC4-Ch 4 Rewired (by passed) two sides of bridge. Gain = $\frac{1}{2}$ x original Gain - not sure when changed back.

LC5-Ch 5 Added extra bridge to cut noise - not sure why or what this does to scale factor.

9/22/83 Ch 3 Rewired R1 & R2 Gain = $\frac{1}{2}$ x original Gain.

10/17/83 Ch 10 & 12 Reset Gain = 1000 not sure why.

12/5/82 Pull test [Ch's 9 x 12 G set to 400] [Ch's 11 & 12, changed resistor on 101 Amp] . [Ch' 11 G ~ 111, reset to 1000]

Ch's 9-12 were well under 1000, when recalibrated.

WAVES

Channels	Scale Factor (ft/count)	Orig.	2/21/83	11/83 Old New	10/31/83 Old New	8/24/83 These Were Off	8/29/83
16 Tide 4/83	.00611 .012	1.0	1.0	1.0 1.0	(9/22 reset)	reset to 1.0	8/31 1.0
17 -		1	1	.53 1.0			
18 NW	.0313	1	1	1.0 1.0		reset 1.0	
19 NE	.0313	1	1	.93 1.0			
20 SW	.0313	1	1	.92 1.0	0.9 0.9		new staff G = 1.0 G = 1.0
21 SE	.0313	1	1	1.07 1.0		reset 1.0	
9 A6	0.00195			.99 1.00	1.16 1.0		
10 A4	0.00195			1.0 1.0			
11 A3	0.00195			.156 1.0			
12 A2	0.00195			.97 1.0			
17 A8	.0313			.53 1.0	1.9 1.0		
19 A5	.0313			.93 1.0	.93 .93		
21 A1	.0313			1.07 1.0			
22 A7	.0313			.92 1.0			

$$\text{Scale Factor} = \frac{\text{ft}}{\text{count}} \times \frac{1}{\text{Gain}}$$

TIDE

10/30/84	Set at 20 ft = 0.5 volts
11/15/84	≈8 ft ≈ 0.58 volts
12/23/82	≈0.9v at 2:20 pm
12/29/84	≈0.69 at 3:40 pm
01/6/83	0.73 at 1:27 pm
01/11/83	.78 at 1:39 pm
01/14/83	Accidently moved offset 12:55 pm = 10½ ft from top of support set $V_o = 1.0$ volts 2:37 pm ≈ 10 ft tide $V_o = 0.85$
01/25/83	.75 at 2:18 pm
03/15/83	0.305v = 9.7 ft
03/16/83	1.10v ≈ 7 ft at 4:20 pm
03/30/83	1.98v ≈ 17 ft from top of bracket
04/ 5/83	2.11v ≈ 2:53 pm
04/08/83	1.45v at 4:30 pm
04/11/83	1.25v at 1:55 pm
04/14/83	1.45v at 2:42 pm
04/14/83	1.00v at 5:09 pm
04/26/83	4:40 pm tide gage at peir read 10.6 ft 4:49 pm Ch 6 = 0.91 ±0.01 volts
06/13/83	out

Mid April, 1983 S.F. changed from .006 to .01288

PRESSURE

Channels	Original Scale Factors psi/mv	Original Gains 02/21/83	Newer Scale Factors 3/84 Lab psi/volt	8/29/83 Old New	10/3/83	11/83 In Lab	12/6/83
22	.424	1000		-/2000		.92/1.0	
23	.471	1000		1000/2000		?/1020	
24	.425	1000		OK/2000		1698/2005	
25	.0471	1000		800		-	
26	.480	1000		2000/2000		2043/2002	
27	.538	1000		4000/2000		-	
28	.448	1000		1800/2000		-/1006	
29	2.09	1000		1000/2000		-/1016	
30	.556	1000		1000/2000		2025/2005	
31	.168	1000	0.334	2000/2000		2011/2011	2000/500
32	.588	1000	.909	2600/2000	2000/500	609/490	
33	.404	1000	1.68	470/2000	-	-	
34	.419	1000	-	50/2000	-	-	
35	.504	1000	0.957	2000/2000	2000/500	281/500	
36	.433	1000	1.199	64/2000	2000/500	-	
37	.433	1000	1.180	42/2000	2000/500	491/491	
38	.464	1000	1.044	1068/2000	2000/500	444/496	
39	.469	1000	1.015	1040/2000	2000/500	555/513	
40	.426	1000	0.280	4390/2000	-	-	
41	.418	1000	1.015	1000/2000	2000/500	530/504	
42	.508	1000	0.251	60/2000	-	-	
43	.433	1000	-	13000/2000	2000/500	530	
44	.374	1000	1.209	2900/2000	2000/500	483/504	

$$\text{Scale Factor} = \frac{\text{psi}}{\text{mv}} \times \frac{2500 \text{ mv}}{\text{count}} \times \frac{1}{\text{Gain}}$$

CONCRETE STRAIN

Channel	Scale Factor ($\frac{\mu s}{\text{count}}$)	2/21/83	5/26/83	8/30/83	11/83
45	1458.6	988	501/1000	2000	2015/2015
46	1458.6	988	478/1000	2150/2000	1951/2006
47	1458.6	493	509/1000	2000/2000	147/2017
48	1458.6	494	520/1000	2000/2000	-/2018
49	1458.6	248	267/1000	-	526/1998
50	1458.6	bad	543/1000	2000	1989/1989
51	1458.6	488	623/1000	2000	2097/1992
52	1458.6	480	419/1000	2000	1961/2016
53	1458.6	492	-	950/2000	1967/2010
54	1458.6	512	546/1000	2000/2000	1956/2006
55	1458.6	426	488/1000	2000	1997/1997
56	1458.6	439	520/1000	2000	2039/2008
57	1458.6	496	467/1000	2000	1955/1996
58	1458.6	508	542/1000	2000	2018/2018
59	1458.6	468	506/1000	2000	2017/2017
60	1458.6	491	539/1000	2000	2057/2012

5/3/83 Channels 45 and 46 Vex = 5.0 already

Channels 47 thru 60 reset Vex from 2.5 to 5.0 volts

$$\text{Scale Factor} = \frac{\mu s}{\text{count}} \times \frac{1}{\text{Gain}}$$

ACCELEROMETER

Channel	Scale Factor f	Gain Up to March 15, 1983	Gain 8/30/83
61	0.0631 fps ² /c	½	½
62	0.0631 fps ² /c	1	1
63	0.0098 fps ² /c	1	1
64	0.0631 fps ² /c	½	½
65	0.0631 fps ² /c	½	1
66	0.0098 fps ² /c	1	1
62	replacement horizontal		1
79	outer vertical		1
80	outer vertical		1

$$S.F._L = 0.0631 \div \text{Gain} =$$

$$S.F._R = 0.0098 \div \text{Gain}$$

RELATIVE MOTION

$$V_{ex} = 5v$$

$$\text{Gain} = 1$$

V_o = output voltage signal

C_o = output counts

Channel		11/83	12/2 - 12/16
67	wx	2.0	10
68	wz	2.0	10
69	Ex	2.0	10
70	Ez	2.0	10
71	Ext	$\frac{1}{2}$	$\frac{1}{2}$
72	Rot	2.0	10

12/2/84 Ch's 67-70 and 72 changed Gain from 2 to 10

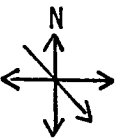
$$\text{S.F.} = 0 = \cos^{-1} (y) \quad y = \left(\frac{V_o - \bar{V}_o}{2.5} \right) = \left(\frac{C_o - \bar{C}_o}{127.5} \right)$$

$$\text{S.F.}_{(67-70)} = \cos^{-1} \left(\frac{V_o - \bar{V}_o}{2.5} \right) = \cos^{-1} \left(\frac{C_o - \bar{C}_o}{127.5} \right) \text{ degrees}$$

$$\text{Ch 71 S.F.} = \frac{8 \text{ inches}}{5 \text{ volts}} \times \frac{2.5 \text{ volts}}{255 \text{ count}}$$

$$\text{Ch 72 S.F.} = \frac{360^\circ}{5 \text{ volts}} \times \frac{2.5 \text{ volts}}{255 \text{ count}}$$

WIND SPEED AND DIRECTION (Cont.)

1/11/83	Ch 75 →		E 1.5 volts SE
		SE = 1.53 volts	
1/14/83	Ch 73	0.37 = North at 15 mph	
	75	$V_0 = 1.27$	
2/22/83		Calibrate WS and D 73, 74, 75 and 76 - see attached sheet Checked 75 and 76 with two anemometers	
3/15/83		6-10 mph = 0.214v	
3/16/83		19 mph = 0.25v	
5/26/83	Ch 73	- out	
	75	OK 2.09	
	74	= 0.17-0.19v ≈ 9.5 to 10 mph	
6/18/83	Ch 73	OK	
9/8/83	Ch 74	Hard wired to Ch 73 ∴ Ch 73 = Ch 74	
10/11/83	Ch 75	Gain at $\frac{1}{2}$ offset = 0	
	76	Gain at $\frac{1}{2}$ offset = 0	
1/14/84	Offset 73, 74 to 1.0 volts Spiral notebook shows this on 9/18/83		
9/15/83	rewired Ch 73 recalibrated to original		

TEMPERATURE

Read Direct

1 Degree
count

CONNECTOR LOAD BOLTS

12/83 - 1/31/84

Channel		Gain 11/83	S.F. $\frac{1\text{bs}}{\text{mv}} \frac{2500\text{ mv}}{255}$
23	N outer	1020	15.7 = 153.9 $\frac{1\text{b}}{\text{c}}$
28	N inner	1006	15.9 = 155.9
29	S inner	1016	15.7 = 154
43	S outer	503	31.8 = 312

$$\frac{100000\text{ lbs}}{\frac{1.250\text{ mv}}{\text{Vex} = 5}} = \frac{100000\text{ lbs}}{6.25\text{ mv}} = \underline{\underline{16000\text{ lbs/mv}}}$$

12/12/83

Change Gain back to 1000

Not sure what it was

$$\text{S.F.} = \frac{16000\text{ lbs}}{\text{mv}} \times \frac{2500\text{ mv}}{255\text{ correct}} \times \frac{1}{\text{Gain}}$$

APPENDIX V

BOAT WAKE AND PULL TEST

APPENDIX V

BOAT WAKE AND PULL TESTS

VII.1 Boat Wake Tests

Boat wake tests were conducted on April 4, August 18, and October 29, 1983. Vessel characteristics for each of the three tests are listed below. Then, for each test, the individual run characteristics are listed. Vessel directions (sailing line) are given in approximate azimuth bearings; distances are approximately minimum distances between the vessel and the closest corner at the west end of the pontoon breakwater.

Table V-1. Vessel Characteristics

<u>Date</u>	<u>April 4</u>	<u>August 18</u>	<u>October 27</u>
Vessel Type	Coast Guard Cutter	Marine Tug	Harbor Tug
Length, ft.	40	110	73
Beam, ft.	10	34	17
Draft, ft.	3.5	10	11
Displacement, tons	10 (approx.)	193 (gross)	67
Speed, knots	12, 16, 20	11	10
Wave period range, seconds	2.5 - 3.5	2.8	2.8
Wave height max, feet	2.2	2.8	2.8

Table V-3. August 18 Test Conditions

<u>Run No.</u>	<u>Sailing Line</u>	<u>Speed-kn</u>	<u>Distance-ft.</u>
1	345°	5	50
	135°	8	50
2	345°	11	50
	345°	11	50
	135°	11	50
3	330°	11	50
	150°	11	60
4	325°	11	50
	135°	8	50
	345°	11	200
5	345°	8	200
6	0°	11	150

Table V-4. August 29 Test Conditions

<u>Run No.</u>	<u>Sailing Line</u>	<u>Speed-kn</u>	<u>Distance-ft.</u>
0	90°	6-7	200
1	90°	6-7	250
2	315°	10	250
3	315°	not recorded	not recorded
4	315°	10	400
5	315°	10	250
6	330°	10	50-100
7	180°	10	100
8	315°	10	150
9	180°	10	100
10	315°	10	150
11	200°	10	150
12	315°	10	100
13	215°	10	250
14	130°	10	200
15	215°	10	200
16	130°	10	not recorded
17	225°	10	250-300
18	130°	8	150
19	225°	10	150-200

Note: Vessel sailing lines for each of the three tests were dictated in part by vessel draft and water depth. Limited serial photographs were taken during the April 4 and August 18 tests, and 8-mm motion pictures of the breakwater response were taken during the October 29 tests.

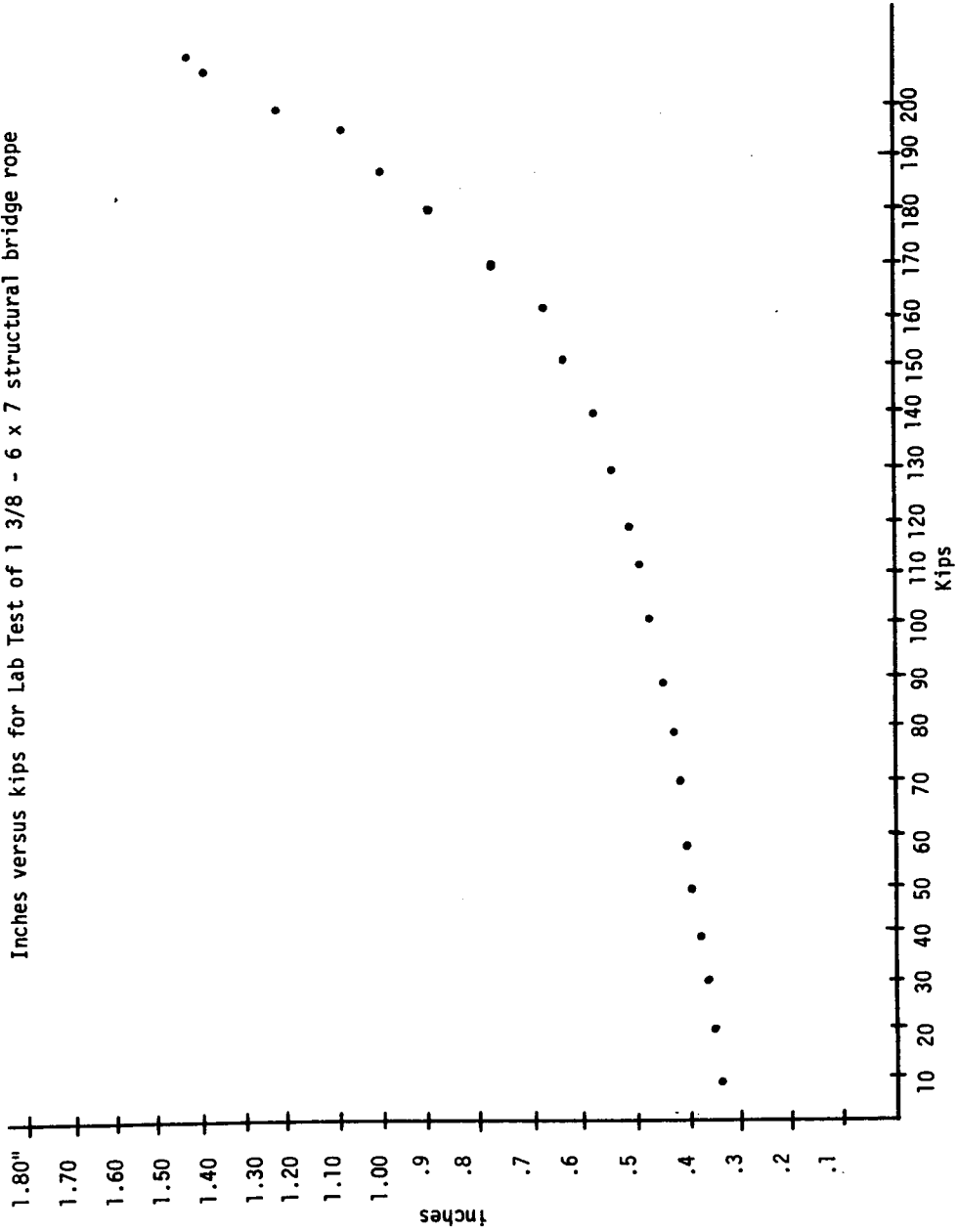
V.2 Pull Tests

APPENDIX VI
LABORATORY TESTING

UW Laboratory Test - 3/8", 6 x 7 Structural Bridge Rope

Kips	Inches
0	
10	.350
20	.353
30	.366
40	.387
50	.397
60	.412
70	.430
80	.443
90	.458
100	.474
110	.492
120	.507
130	.536
140	.567
150	.611
160	.660 load popping sound
170	.741
180	.860
185	.945
190	.059
195	1.188
198	1.289
200	1.372
201	1.427
204	2.162 failure

Inches versus kips for Lab Test of 1 3/8 - 6 x 7 structural bridge rope



Laboratory Test 1 3/8 - 6 x 7 Structured Bridge Rope
 Scale Factor = 2.1575 ft/volt x 12/ft = 25.89 inches/volt

	up	down	up volt	down	up	inches
0	0					
10	.0080	.0104				0.2693
20	.0090	.0111	.0105			0.2718
30	.0101	.0118	.0110			0.2848
40	.0112	.0121	.0118			0.3055
50	.0120		.0122	.0162		0.3159
60			.0134	.0168	.0159	0.4117
70			.0142	.0174	.0166	0.4298
80			.0222	.0179	.0171	0.4427
90			.0160	.0185	.0177	0.4583
100			.0172	.0192	.0183	0.4738
110			.0182	.0194	.0190	0.4919
120			.0193		.0196	0.5074
130					.0207	0.5359
140					.0219	0.5670
150					.0236	0.6110
160	popping sound				.0255	0.6602
168						
170					.0286	0.7405
174						
176						
178						
179						
180					.0332	0.8596
181						
182						
183						
184						
185					.0365	0.9450

186		
187		
188		
189		
190	.0409	1.0590
191		
192		
194		
195	.0459	1.1883
196		
197		
198	.0498	1.2893
199		
200	.0530	1.3722
201	.0551	1.4265
204	.0835	2.162

1 3/8 - 6 x 7 Structured Bridge Rope

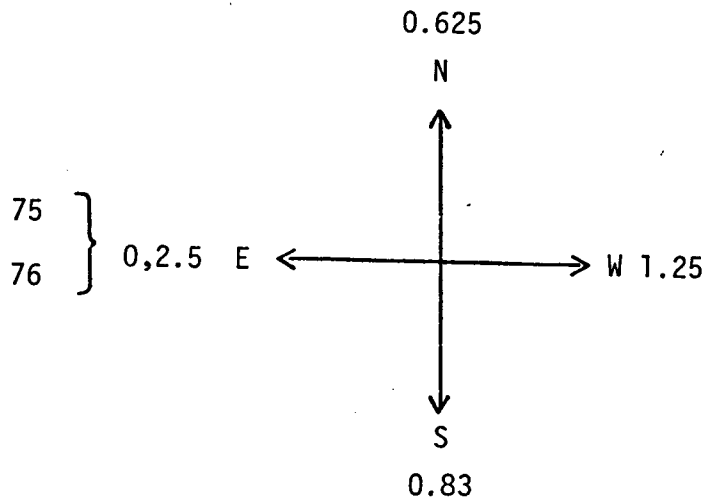
Load Kips	Volts (-1,3944)	Inches	Inches
0	0		
10	.0104	0.269	
20	.0105	.272	.003
30	.0110	.285	.013
40	.0118	.306	.021
50	.0122	.316	.010
60	.0159	.412	.096
70	.0166	.430	.018
80	.0171	.443	.013
90	.0177	.458	.015
100	.0183	.474	.016
110	.0190	.492	.018
120	.0196	.507	.015
130	.0207	.536	.029
140	.0219	.567	.031
150	.0236	.611	.044
160	.0255	.660	.049
168			
170	.0286	.741	.081
174			
176			
178			
179			
180	.0332	.860	.119
181			
182			
183			
184	.0365	.945	.085
185			
186			

187			
188			
189			
190	.0409	1.059	.114
191			
192			
194			
195	.0459	1.188	.129
196			
197			
198	.0498	1.289	.101
199			
200	.0530	1.372	.083
201	.0551	1.427	.055
204	.0835	2.162	.735

WIND SPEED AND DIRECTION

Channel

$$\left. \begin{array}{l} 73 \\ 74 \end{array} \right\} \frac{100 \text{ mph}}{2.5 \text{ volts}} \times \frac{2.5 \text{ volts}}{255 \text{ c}} = 0.392 \frac{\text{mph}}{\text{c}}$$



WIND SPEED AND DIRECTION

<p>10/30/82 Sat. S.F. = 40 $\frac{\text{mph}}{\text{volt}}$</p>
<p>11/3/82 Ch 73 Gain set = 1.30 calibration in field but have changed $\pm V_{ax}$ from 10 to 15 volts</p>
<p>11/11/82 Ch 74</p> <div style="text-align: center;"> </div> <p>Ch 73 ground input and set zero</p>
<p>11/15/82 10-12 mph = 0.234 volts</p>
<p>11/23/82 Ch 73 5 mph = 0.18v 4 mph = 0.173 \pm .003 Ch 74 reset 1.6 volts = NW was out</p>
<p>11/30/82 0.22 v \approx 12 mph 0.27 v = 13 mph 0.31 v = SE</p> <div style="text-align: center;"> </div>
<p>12/23/83 Recalibrate W.S. at lab 5 mph = 0.25 volts S.F. = 40 mph/volt</p>
<p>12/27/83 10 mph \approx 0.25v 1.50 = N</p> <div style="text-align: center;"> </div>
<p>12/29/83 1.3 \Rightarrow</p> <div style="text-align: center;"> </div>

Summary of Anchor Cable Test 6/21/82

Cable 4.32 ft/volt nylon
2.16 ft/volt wire rope

Nylon Hard lay Dry	45900#/4.37 ft.	
Nylon Hard lay Dry	46600#/4.62 ft.	
Nylon Hard lay Wet	39750#/4.38 ft.	(14% reduction)
Nylon Soft lay Dry	37450#/4.0 ft.	
Nylon Soft lay Dry	36800#/ 4.5 ft.	(1 month)
Nylon Soft lay Wet	33400#/3.3 ft.	(11% reduction)

18.4% difference between Hard and Soft lay.

20400 lb for Bridge stress

Calibration - Cable Transducer

$V_{in} = 5.000 \text{ V}$ (HP power supply)

Extension (ft)	V_o	4.3197 ft/volt
8.0	1.858	4.3197
7.0	1.624	4.326
6.0	1.392	4.329
5.0	1.163	
4.	.929	
3.	.698	
2.	.463	
1.	.231	
0.	.006	

Run 2

1 1/4 Nylon Rope
Nylon Rope Test
Hard Lay - Dry

6/21/82

	Load (Kips)	Vo (Volts)		
Initial setting	3.000	4.32 ft/volt	10	
	4.000	1.408	12	.842
	5.000	1.327	14	.823
	6.000	1.222	16	.807
	7.000	1.158	18	.790
	8.000	1.099	20	.767
	9	1.050	22	.740
	10	1.007	24	.719
	11	965	26	.699
	12		28	.675
	13	913	30	.653
	15	872	32	.630
	17	833	34	.607
	19	.797	36	.584
	20	.777	38	.559
	18	.784	40	.534
	16	.795	42	.508
	14	.808	44	.477
	12	.829	46	.445
	10	.853	48	.403
			48200	.397
			2300	failure

Run 3 Soft Lay - Dry corrected failure 37,450

Load (kips)	Vo (volts)	Time (minutes)	Load (kips)	Vo (volts)	Time
----------------	---------------	-------------------	----------------	---------------	------

initial setting 0

3000	1.633	0	30	.867	15.5
5000	1.471				
6000	1.361		32	.847	16.5
8000	1.263				
10000	1.178		34	.828	17.5
12000	1.134				
14000	1.082		36	.803	
16000	1.043	3m	38		
18000	1.005		40		
20000	.976	4m			
18000	.984				
16000	.995				
14000	1.008				
12000	1.026	5m	failure at 36 (one strand)		
10.	1.049	6m	at 36 drop to 22.500		
12	1.031	7m			
14	1.023		26	.786	19
16	1.006	8	28	.776	20
18	.989	9m	30	.765	20.75
20	.969	10	32	.755	
22	.947	11	34	.744	22
24	.925	12	36	.731	22.5
26	.905	13.5	*38650	.707	24
28	.885	14.5	-1200		
			37450		

Run 4

Hard Lay - Wet

6/21/82

Load (kips)	Vo (volts)	Time (min)	Load (kips)	Vo (volts)	Time (min)
Initial load 0					
3000					
4000	1.208	1	26	.385	7.0
6	.942	1.5	28	.353	8.0
8	.807	2.0	30	.328	9.0
10	.714	2.5	32	.301	9.5
12	.650		34	.277	10.0
14	.600	3.0	36	.250	11.0
16	.550	3.25	38	.223	11.75
18	.512	3.50	40	.193	12.5 failure
20	.475	3.75	42		
18	.483	4.00	44		
16	.499	4.25	46		
14	.519	4.50	48		
12	.543	4.75			
10	.576				
12	.555				
14	.532	5.5			
16	.509				
18	.487	6.0			
20	.463				
22	.437				
24	.411				

Run 5

Soft Lay - Wet

6/21/84

Load (kips)	Vo (volts)	Time (min)	Load (kips)	Vo (volts)	Time (min)
Initial load 0					
3000		0			
4000	1.621	1	24	.979	6.5
6000	1.412	1.5	26	.959	
8	1.304	1.75	28	.933	9.0
10	1.230	2.00	30	.912	9.5
12	1.173	2.25	32	.891	10.25
14	1.126	2.50	34	.869	
16	1.086	2.80	34500	.859	11.00
18	1.054	3.00	-1100		
20	1.028	3.50	33400		
18	1.036	4.00			
16	1.048				
14	1.065	4.5			
12	1.087				
10	1.111	5.0			
12	1.093				
14	1.075				
16	1.058	5.5			
18	1.041				
20	1.021	6.0			
22	1.001				

Run 6

Hard Lay - Dry

Load	Vo	Time	Load	Vo	Time
3000					
4	1.290	.5	24	.539	7.5
6	1.052		26	.516	8.0
8	.922	1.75	28	.484	9.0
10	.835		30	.461	10.5
12	.769		32	.440	10.5
14	.710	2.5	34	.416	12.0
16	.664	3.0	36	.393	13.75
18.	.628		38	.368	14.5
20	.596	3.5	40	.341	15.5
18	.602		42	.314	16.5
16	.613	4.0	44	.285	17.5
14	.629		48		
12	.650	4.5	50		
10	.675	5.0	failure:		
12	.664		47,400	.221	19.75
14	.644	5.5	-800		
16	.625	6.0	46,600		
18	.606	6.5			
20	.585				
22	.564	7.0			

1 3/8 - 6 x 7 Structural Bridge Rope

V_{in} = 2.500

Load	Volt	Time	Load	Volt	Time
0.	1.3944		110	1.3762	
2.0	1.3918		120	1.3751	
10	1.3864		110	1.3750	
20	1.3854		100	1.3752	
30	1.3843		90	1.3759	
40	1.3832		80	1.3765	
50.	1.3824		70	1.3770	
40	1.3823		60	1.3776	
30	1.3826		50	1.3782	
20	1.3833		60	1.3785	
10	1.3840		70	1.3778	
20	1.3839		80	1.3773	
30	1.3834		90	1.3767	
40	1.3826		100	1.3761	
50	1.3822		110	1.3754	
60	1.3810		120	1.3748	
70	1.3802		130	1.3737	
80	1.3722		140	1.3725	
90	1.3784		150	1.3708	
100	1.3772		160	1.3689	(popping sound)

Br to displ device change from 5V to 2.5 volts

168	1.3663	200	1.3414
170	1.3658	201	1.3393
174	1.3642	*failure	
176	1.3632	204	1.3109
178	1.3622		
179	1.3616		
180	1.3612		
181	1.3605		
182	1.3600		
183	1.3593		
184	1.3587		
185	1.3579		
186	1.3571		
187	1.3562		
188	1.3555		
189	1.3544		
190	1.3535		
191	1.3526		
192	1.3516		
193	1.3506		
194	1.3495		
195	1.3485		
196	1.3471		
197	1.3456		
198	1.3446		
199	1.3430		

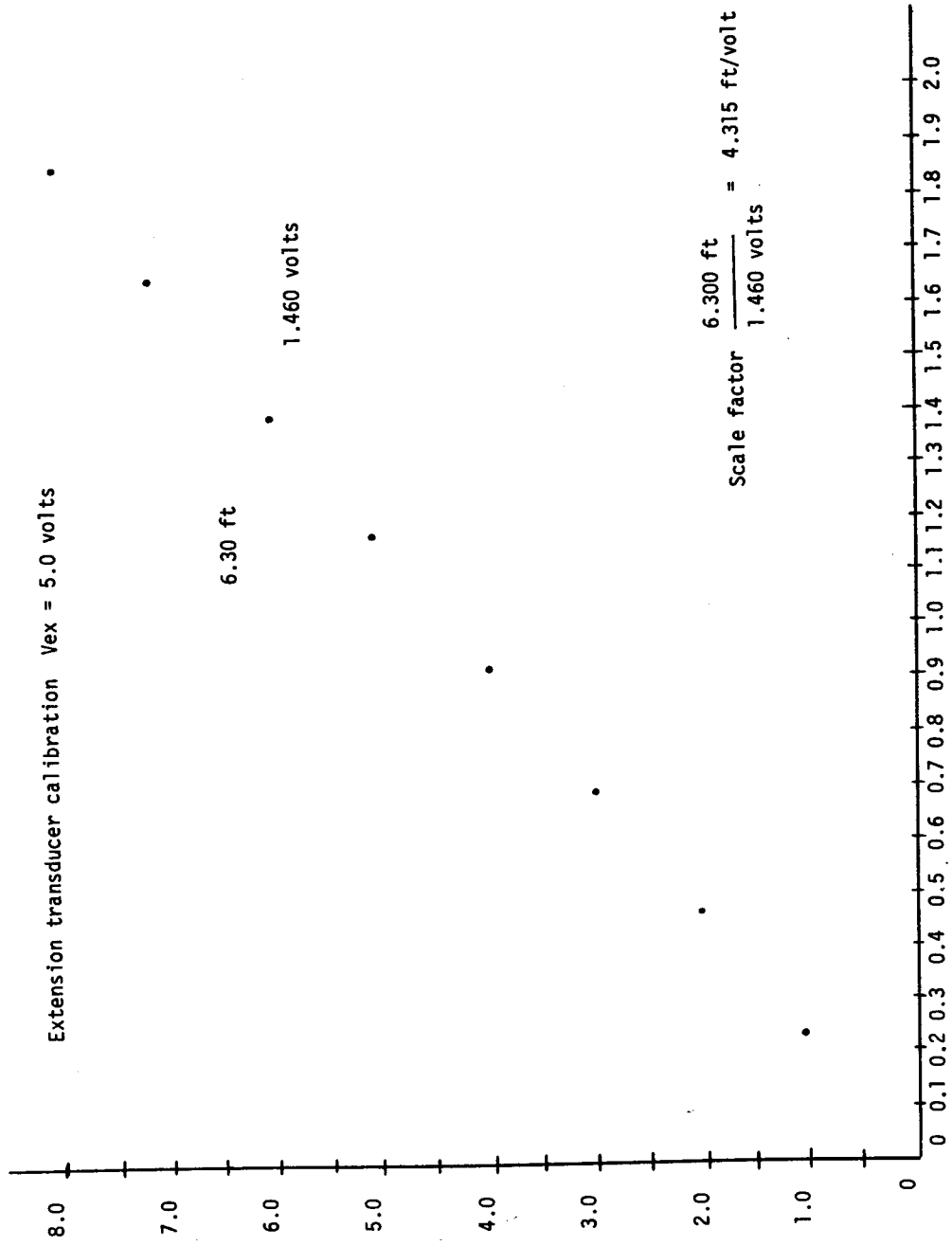
Br to displ device change from 5V to 2.5 volts

168	1.3663	200	1.3414
170	1.3658	201	1.3393
174	1.3642	*failure	
176	1.3632	204	1.3109
178	1.3622		
179	1.3616		
180	1.3612		
181	1.3605		
182	1.3600		
183	1.3593		
184	1.3587		
185	1.3579		
186	1.3571		
187	1.3562		
188	1.3555		
189	1.3544		
190	1.3535		
191	1.3526		
192	1.3516		
193	1.3506		
194	1.3495		
195	1.3485		
196	1.3471		
197	1.3456		
198	1.3446		
199	1.3430		

Calibration

Distance from top of displacement unit to clip (ft)

Disp.	0.00	.01	.008
	1.00	.29	.287
	2.00	.57	.569
(5)	3.00	.86	.852
(4)	4.00	1.14	1.133
(3)	5.00	1.42	1.416
(2)	6.00	1.70	1.697
(1)	7.00	1.99	1.982
(0)	8.00	2.27	2.264



Disp Trans Reading	Load	Disp Trans Reading	Load
2.262		1.166	21000
2.205 (zero)	5.000	1.152	22000
1.659	8.000	1.138	23000
1.361	14.000	1.125	24000
1.307	15.000	1.108	25000
1.365	10.000	1.089	26000
1.304	15.000	1.072	27000
1.260	17.500	1.050	28000
1.224	18.000	1.042	29000
1.212	19.500	1.030	30000
1.190	20.000	1.015	31000
1.302	10.000	1.002	32000
1.267	14.000	.985	33000
1.252	15.000	.972	34000
1.240	16.000	.958	35000
1.227	17.000	.942	36000
1.215	18.000	.929	37000
1.200	19.000	.917	38000
1.185	20.000	.900	39000
1.166	21.000	.882	40000
			break

$$V_o = 6.050$$

NOTE

1. Started zero at 5,000 - actual reading 1,800 pounds.
2. Noted rotation at 8,000 - several revolutions.
3. After relaxation (20,000 10,000) load would creep up with fixed strain.
4. Noted "snapping" at 10K 15K (appeared to be in splice)

End reading 3,200
Zero

initial load
= 5,000 - 3,200 = 1,800

Adjusted "zero" to (5,000 dial) (No actual load)

$h = 5$

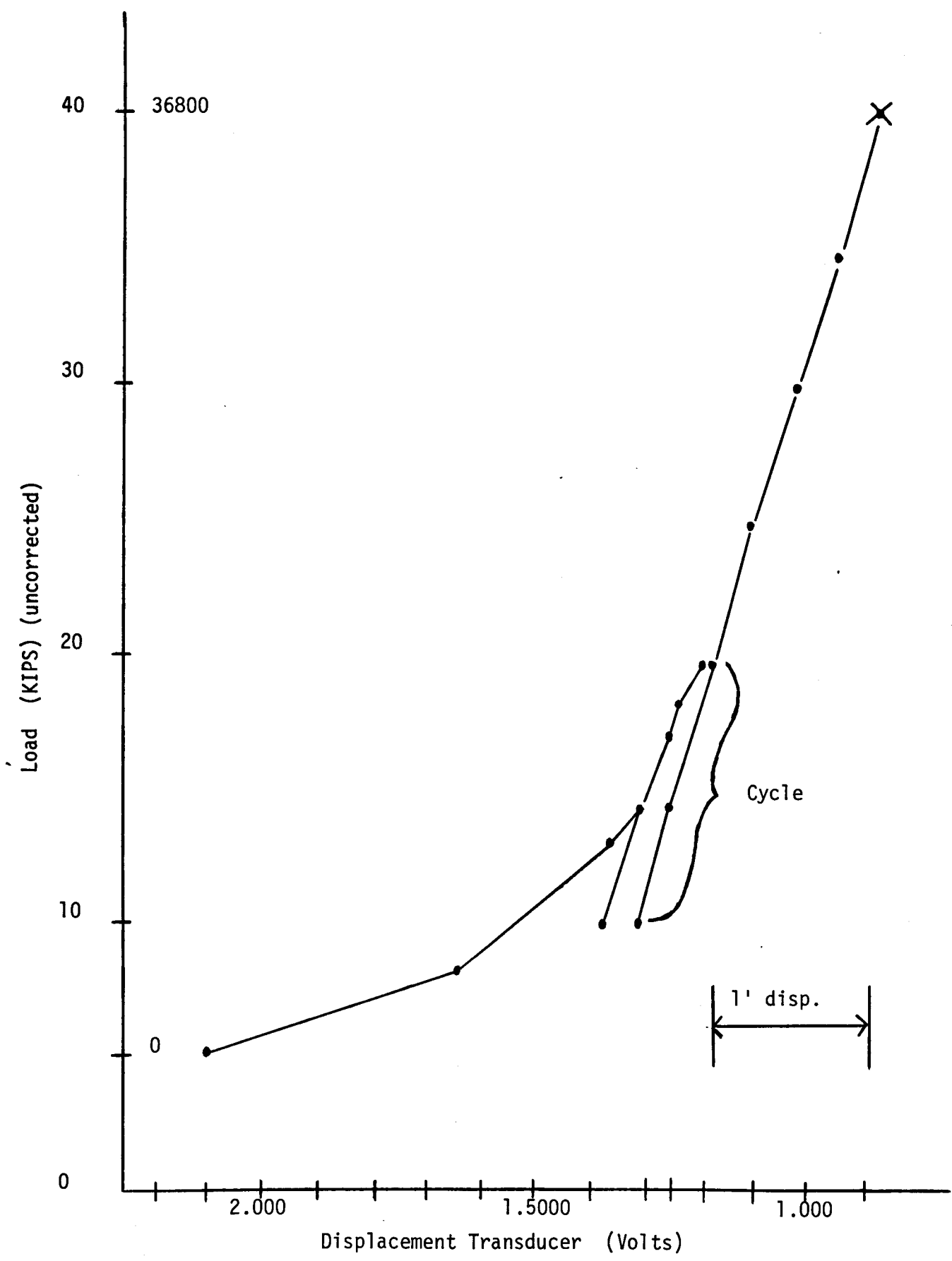
At breaking (40,000 dial)

No load = 3,200 dial

Add 33#/inch

From reading sub 5,000 + 33#/inch

Broke at $40,000 - 5,000 + 1,800 = 36,800$



APPENDIX VIIB

DYNAMIC RUBBER CONNECTOR TESTING

Laboratory Test on Rubber Connector
(Tension and compression MTS 110 Kip Machine)

Test Procedure

A. LOAD CONTROL

1. Constant peak loads versus frequency.

Fix loads at 5, 10, 15 and 20 kips.

2. Constant frequencies versus varying peak loads.

3. Constant load rates.

Alter peak loads and frequencies such that the plus to minus peak slope stays constant



Vary load such that

B. DISPLACEMENT CONTROL

1. Constant peak displacement versus frequency.

2. Constant frequency versus varying peak displacement.

3. Constant load rate.

Dynamic Testing Sequences

6/29/82

Set Zero at $-1.00 \times \text{DCR2}$ (1K comp)
adjust = 4.50

Span	Kips	Adjust	Am at .4V/am
------	------	--------	-----------------

5	=		
10	= 5.00		
15			
20			

IV		.am = 4K	
1.98 2V 0			
4 am = .4.4 =			

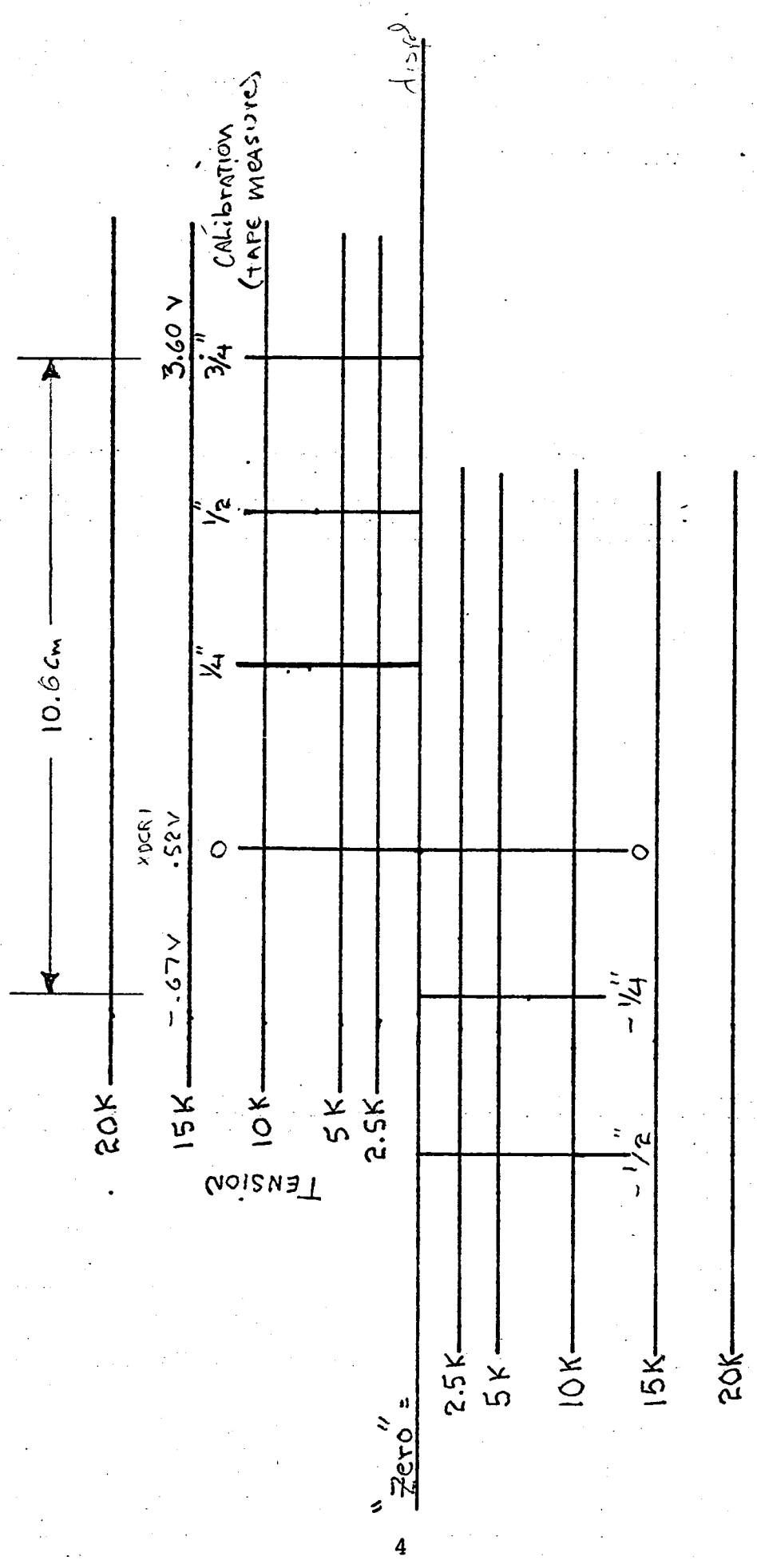
Note: Strain value increases with load in compression

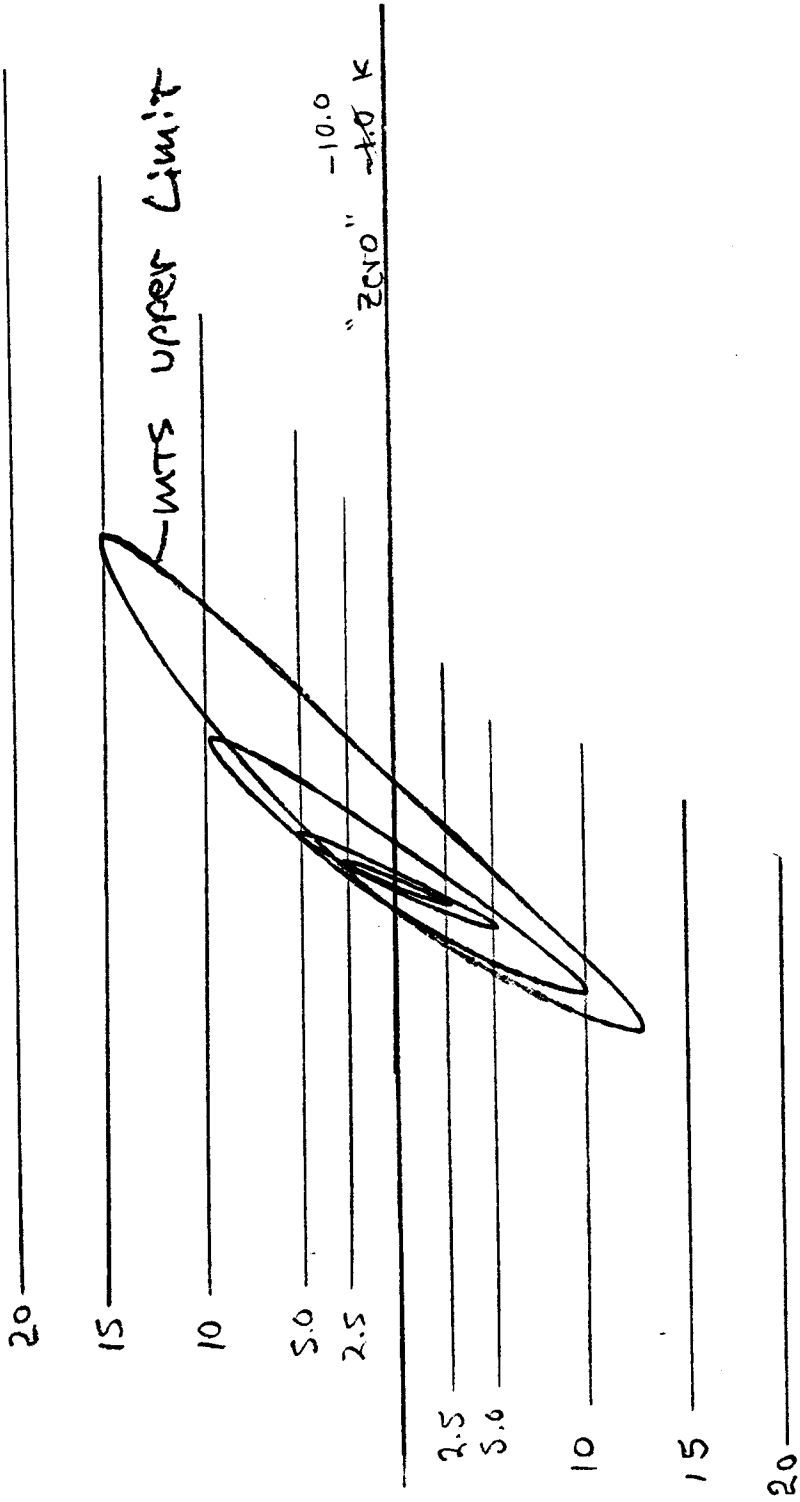
Constant Loading

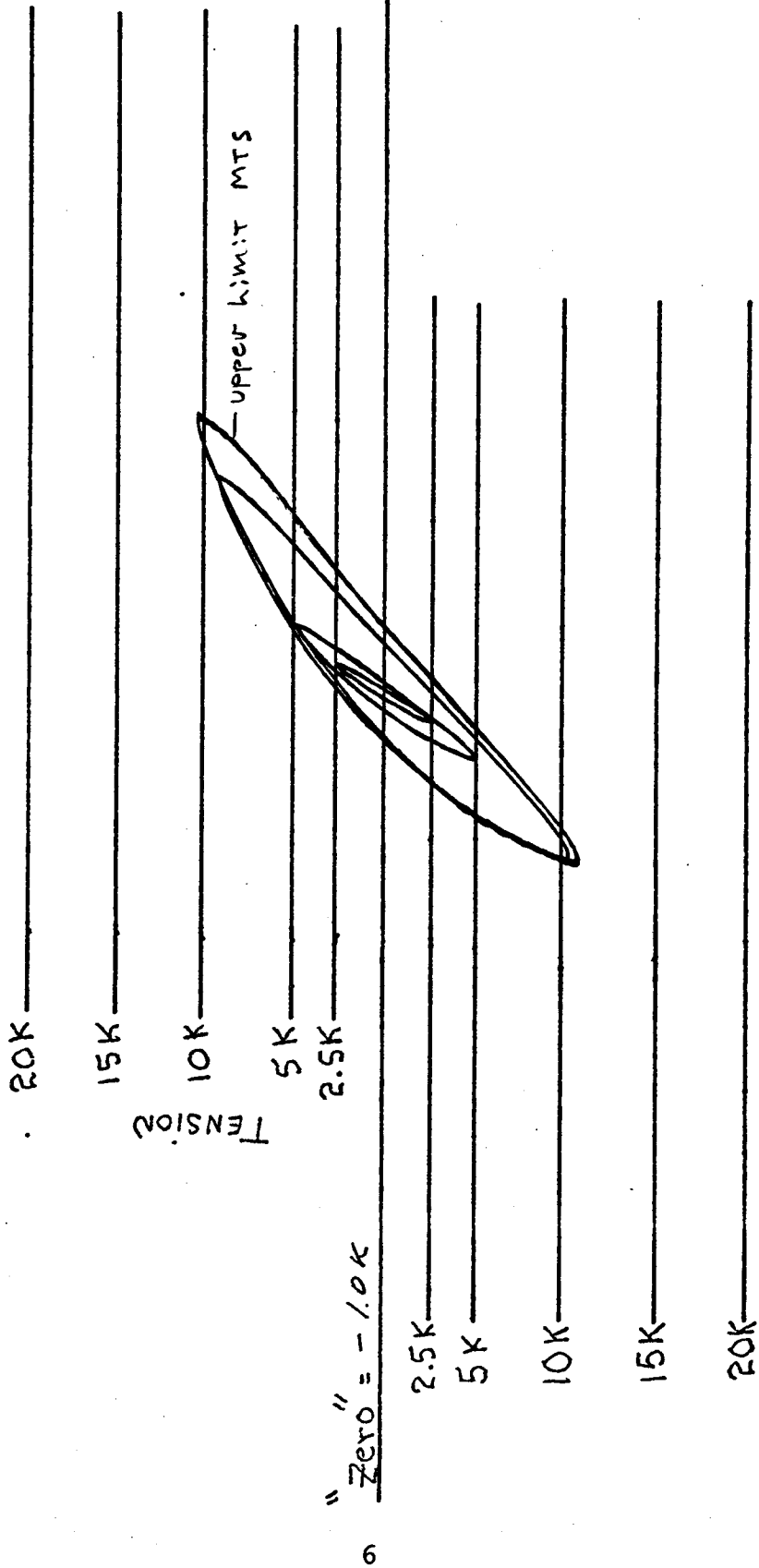
A/T = Constant

+	2.5K	at	2.5 sec	(.4)
+	5K	at	5 sec	(.2)
+	10K	at	10 sec	(.1)
+	15K	at	15 sec	(.067)
+	20K	at	20 sec	(.05)

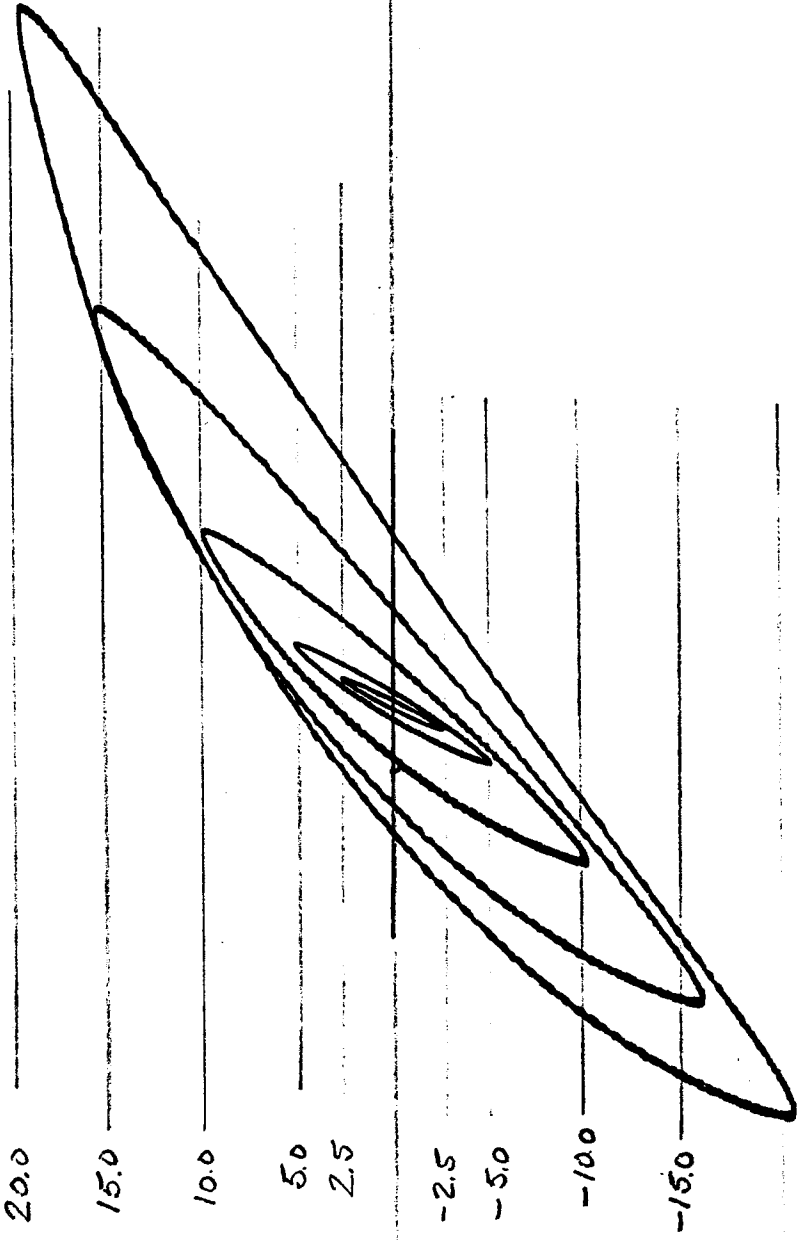
CAL. $2.5'' = 10V$
 $.4V = 1cm$
 $\therefore 2.5'' = 25$







"Zero" = -1.0K

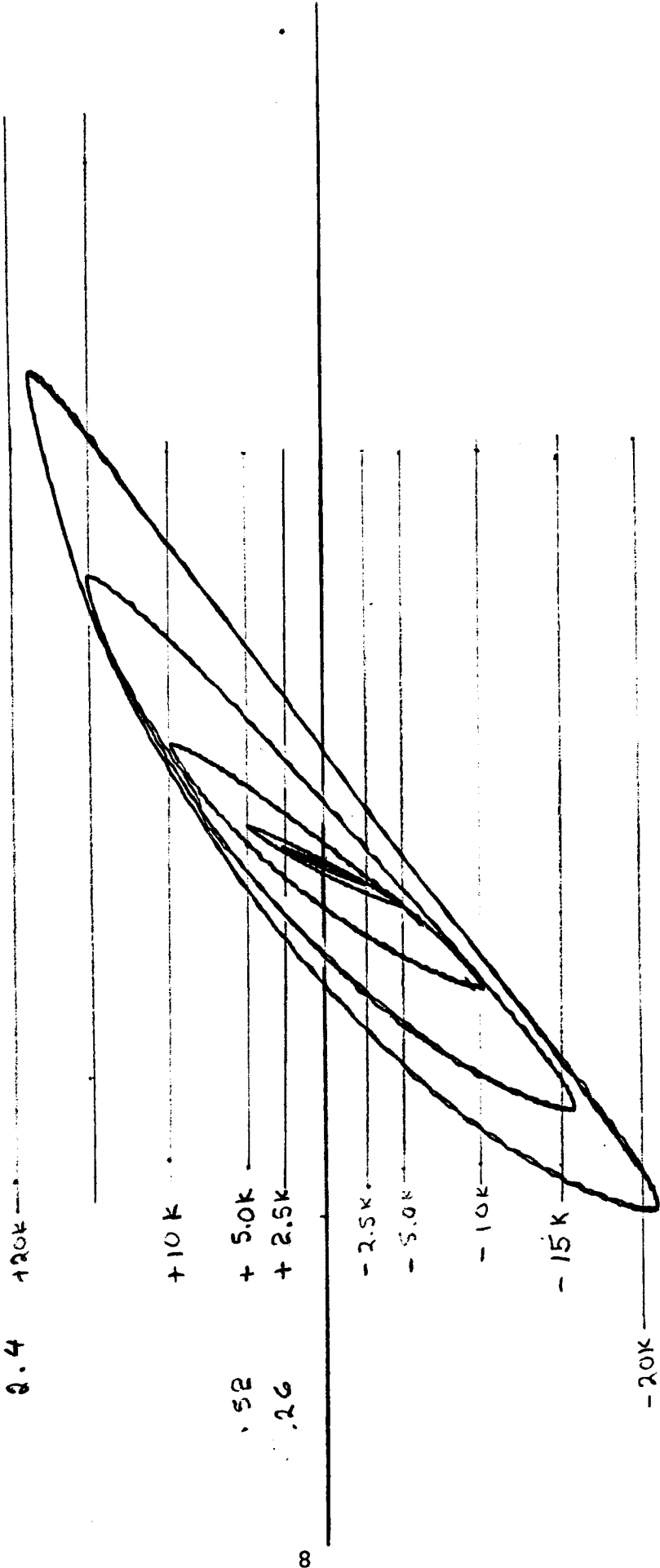


zero
ATK

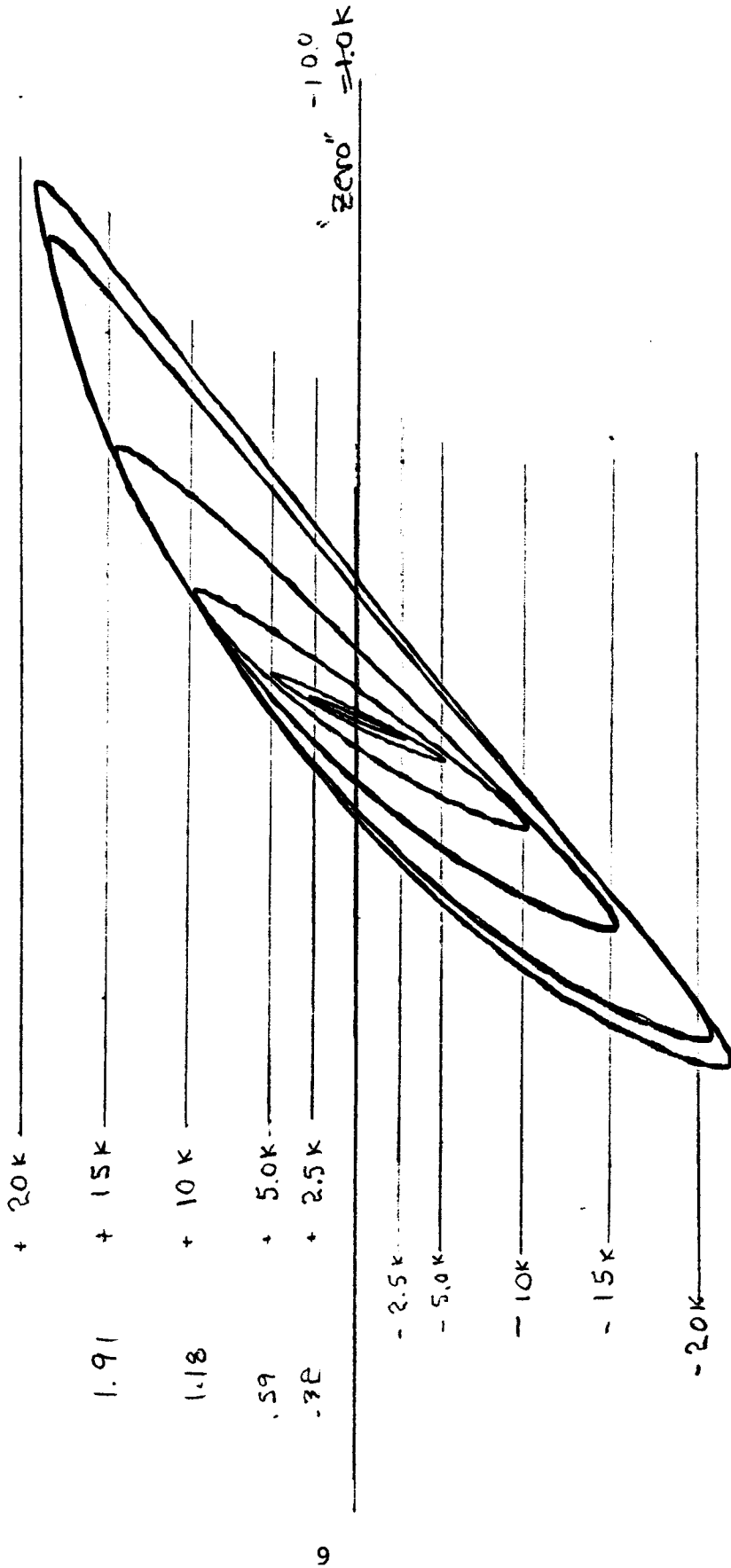
19

f = 1.103

SPAN
ADJUST
2.4

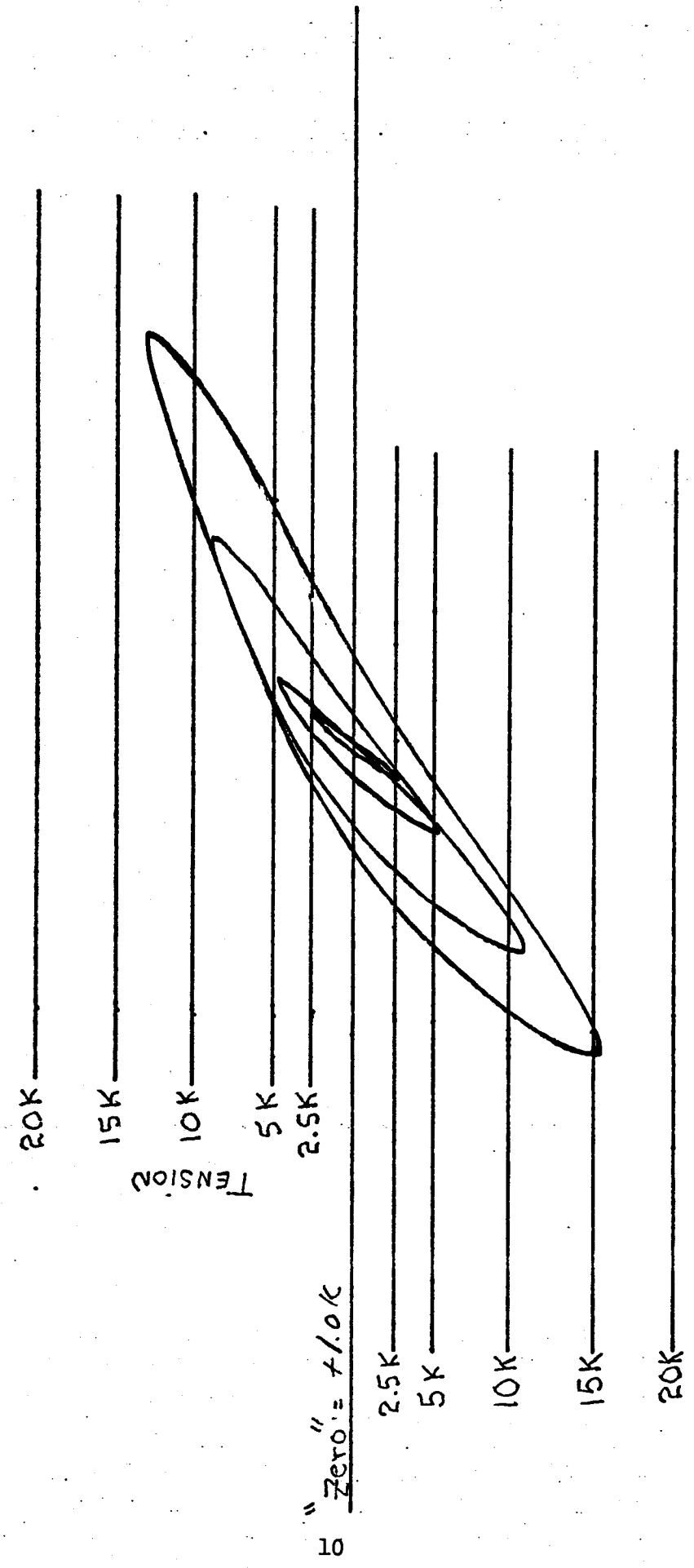


SPAW
MAY 1958



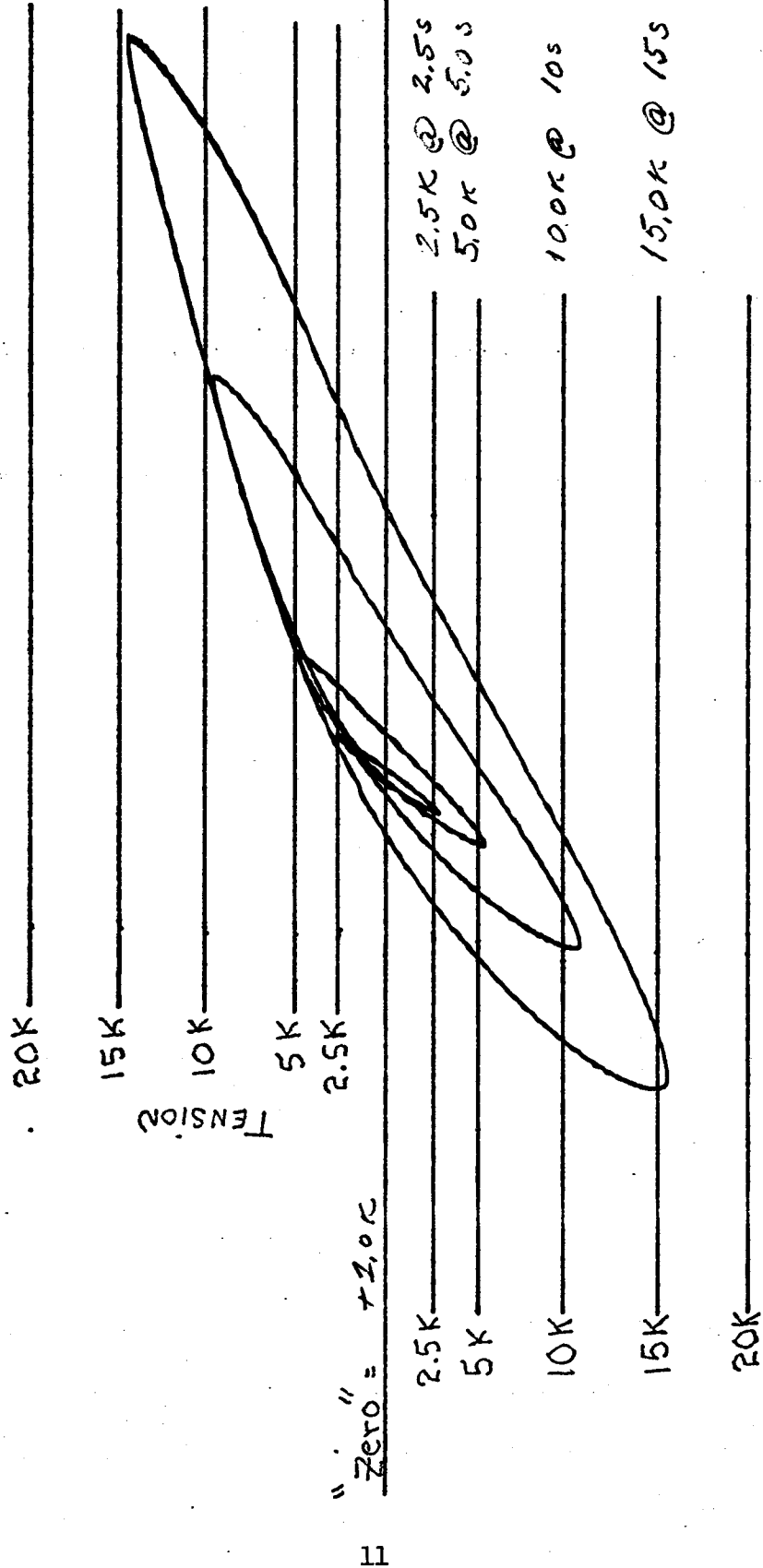
2010N02
Attempt #2

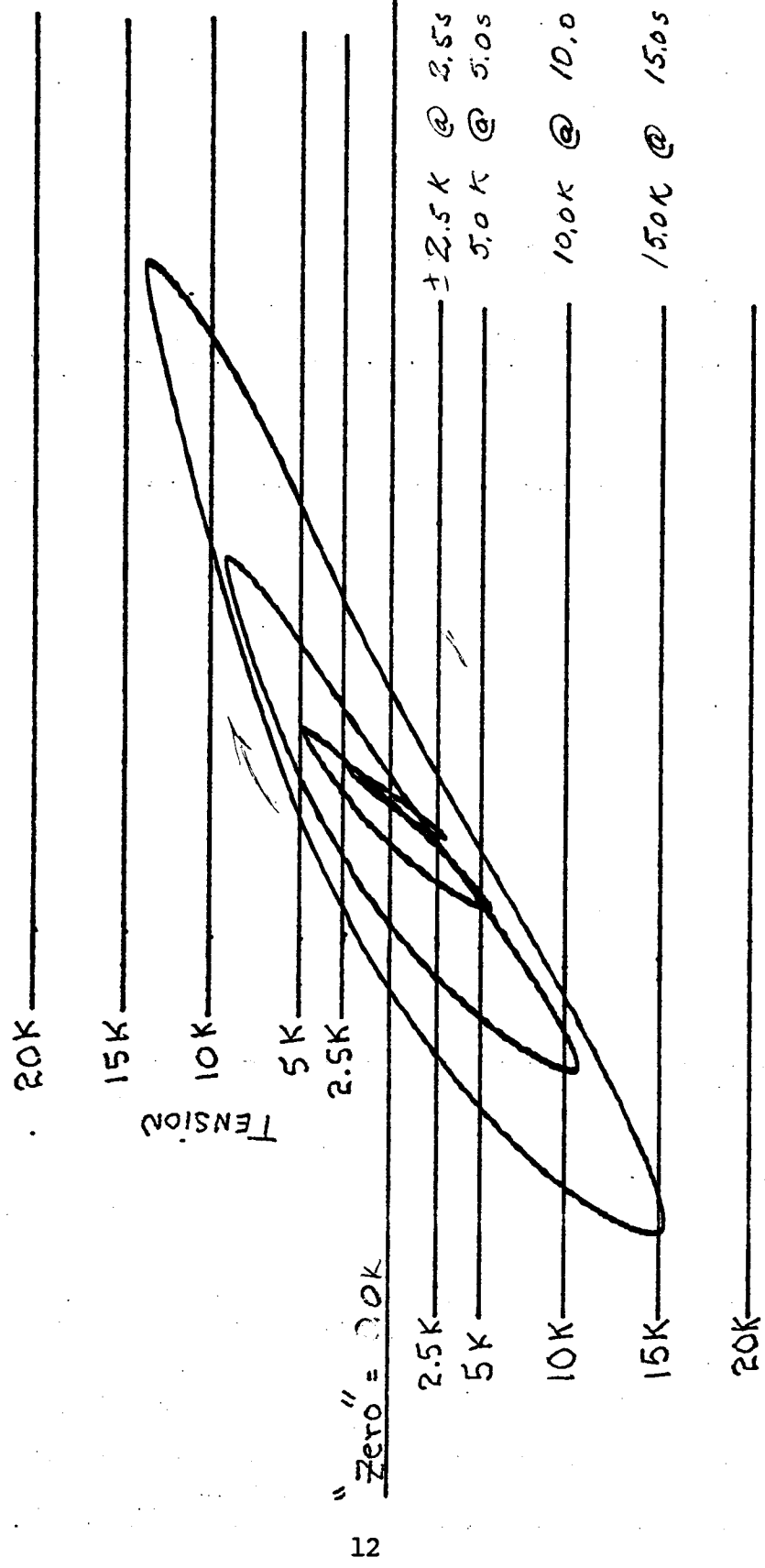
7-5-42

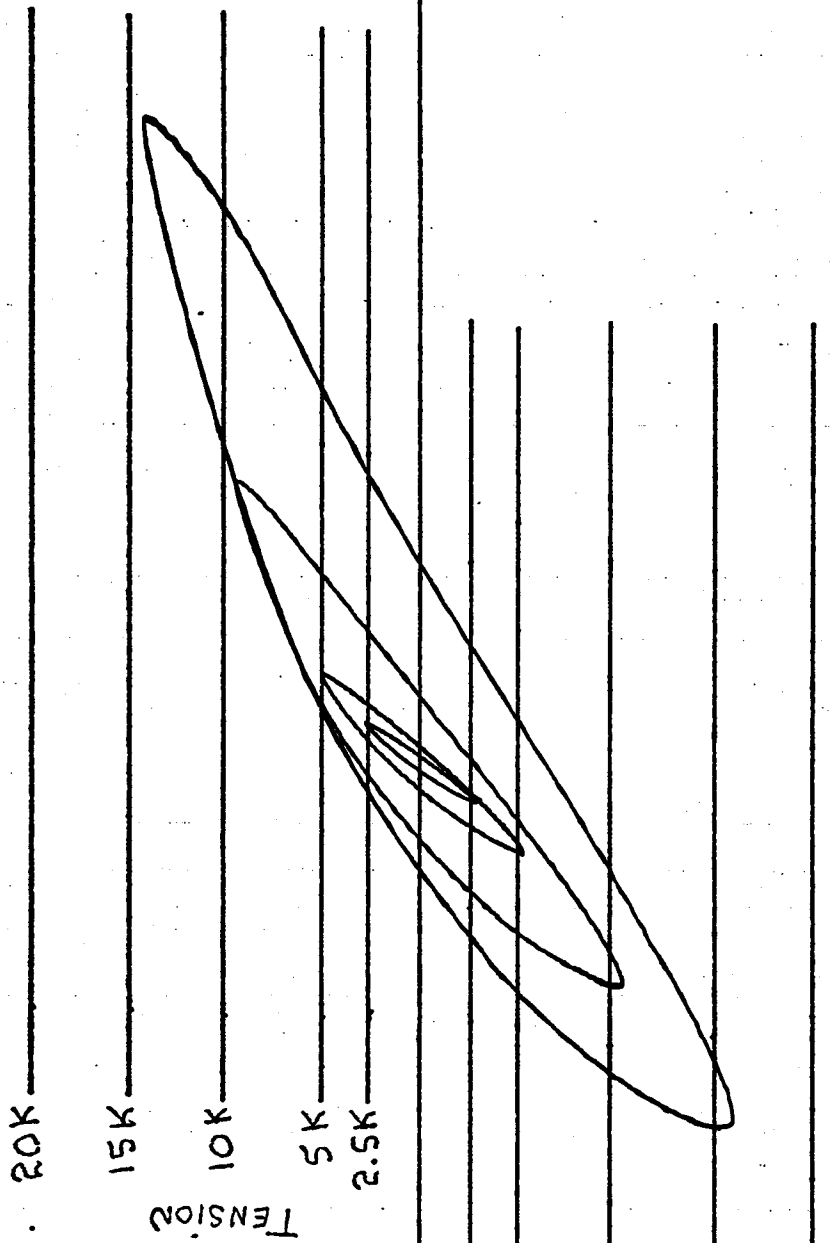


175

18



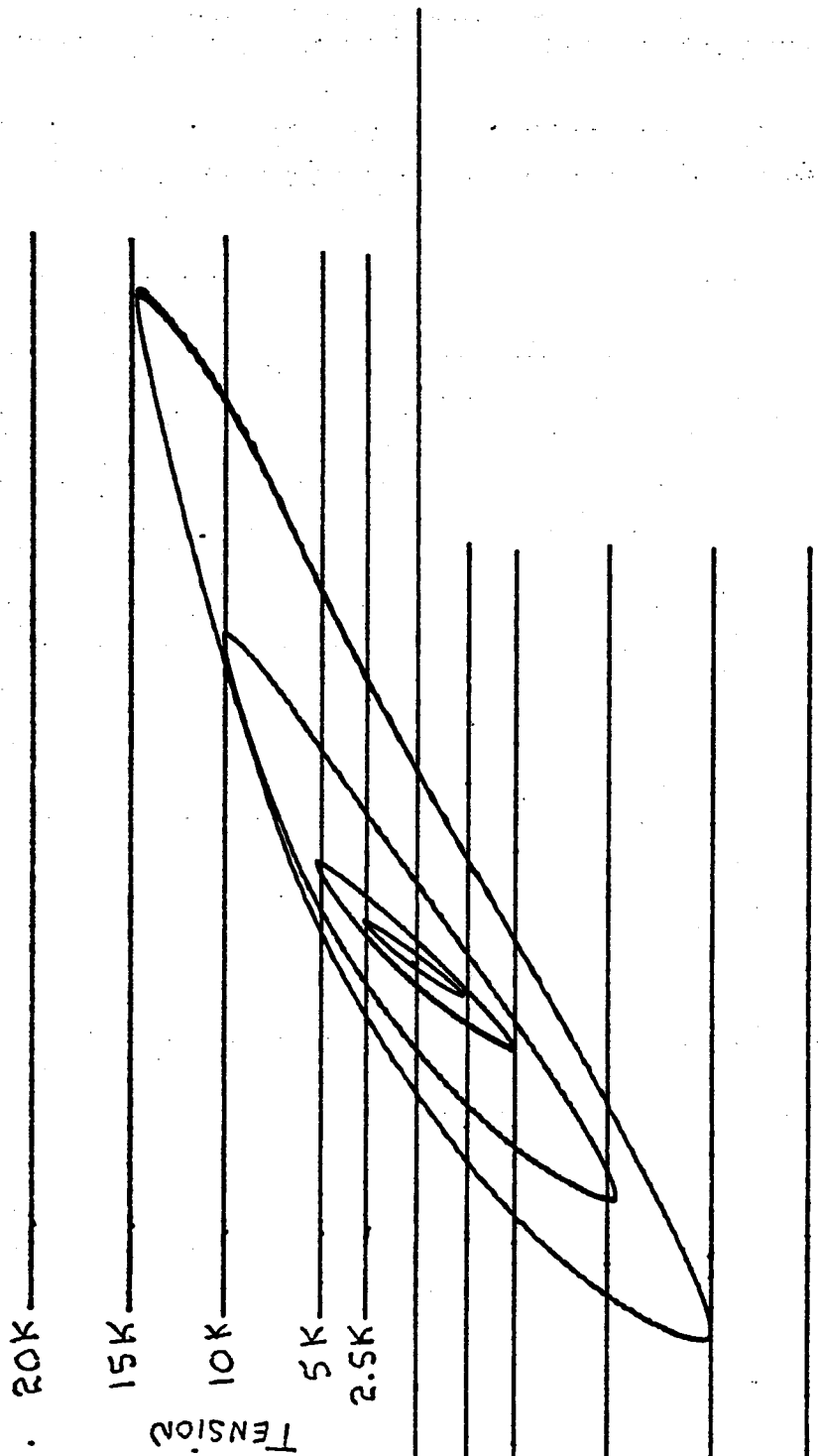


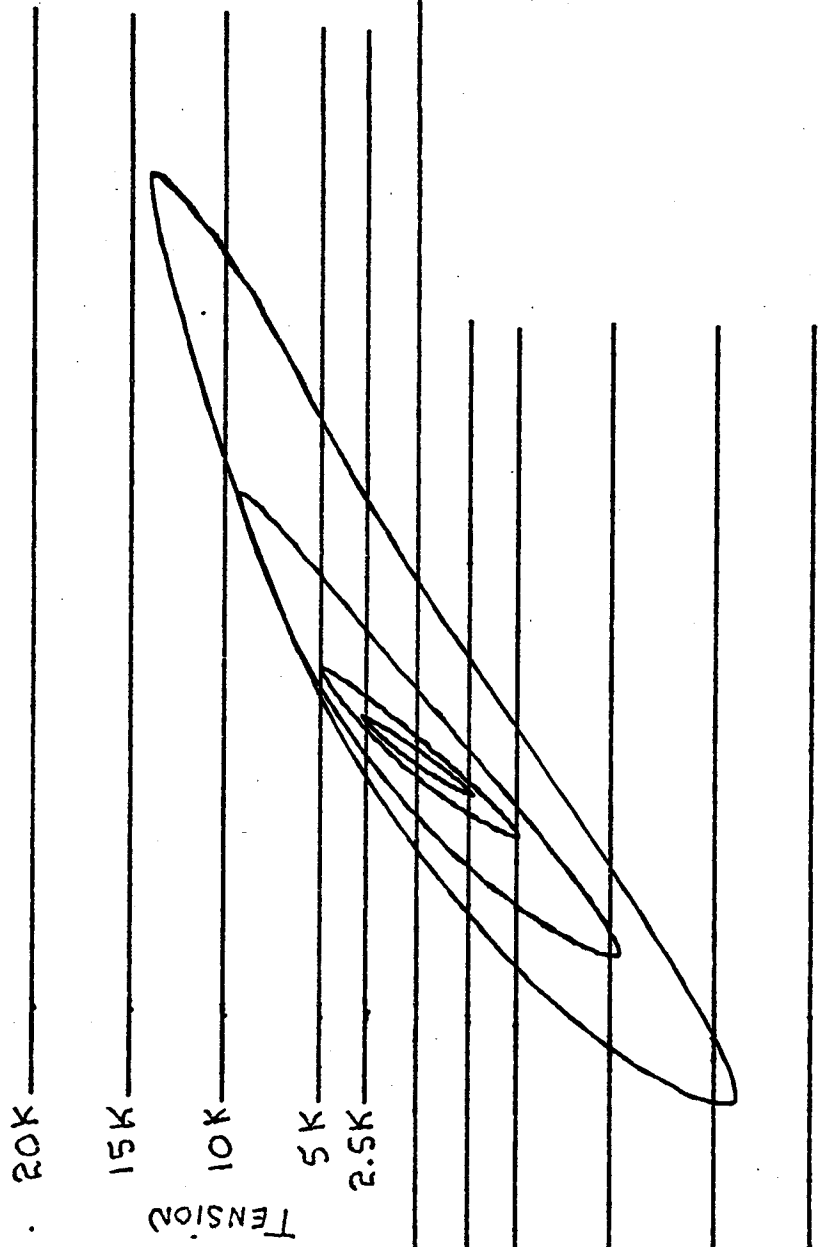


"Zero" = -1.0K

30 JUN 82

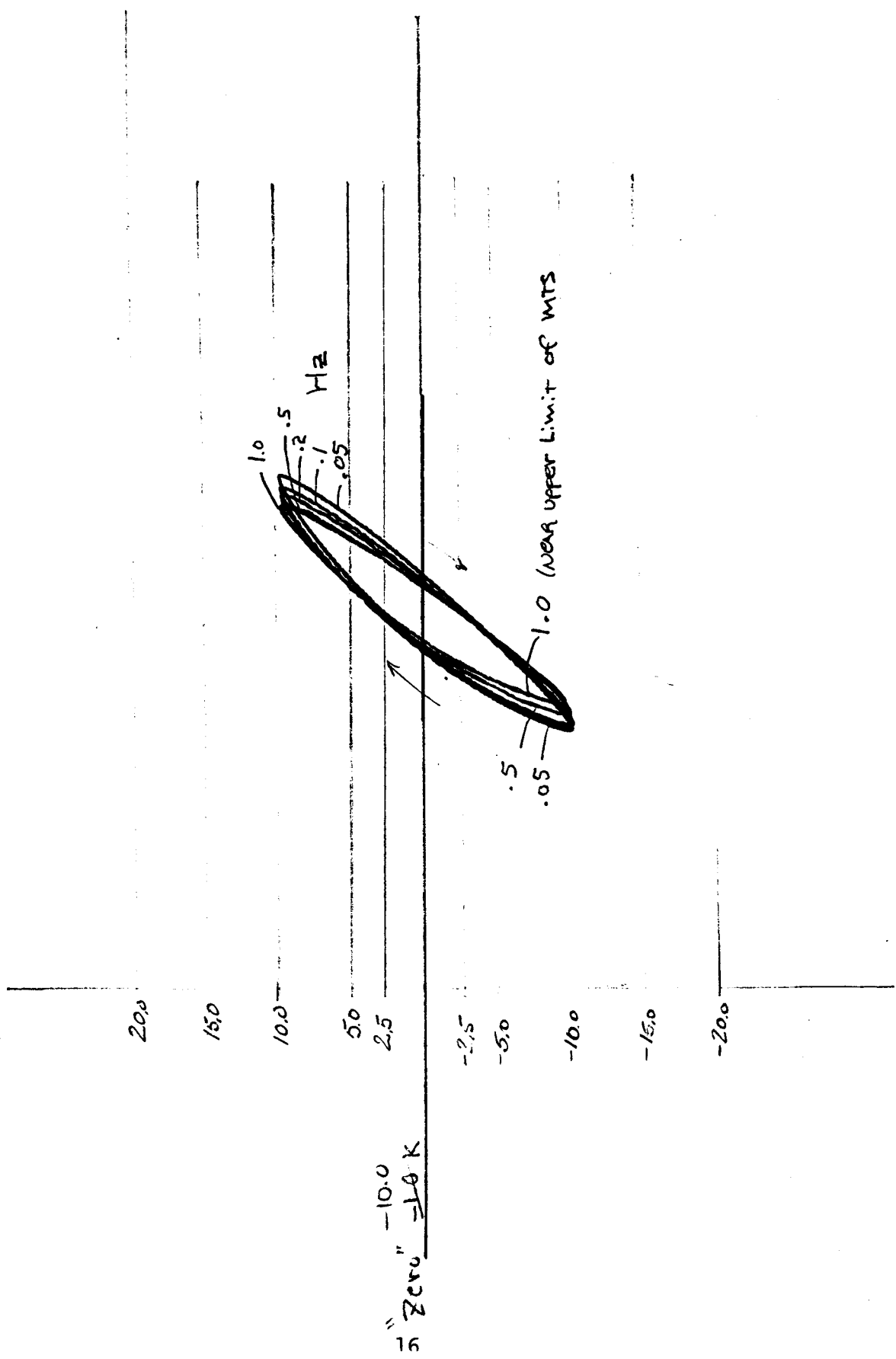
$f = .05 \text{ Hz}$





15 "Zero" = -1.0K

6/29/82

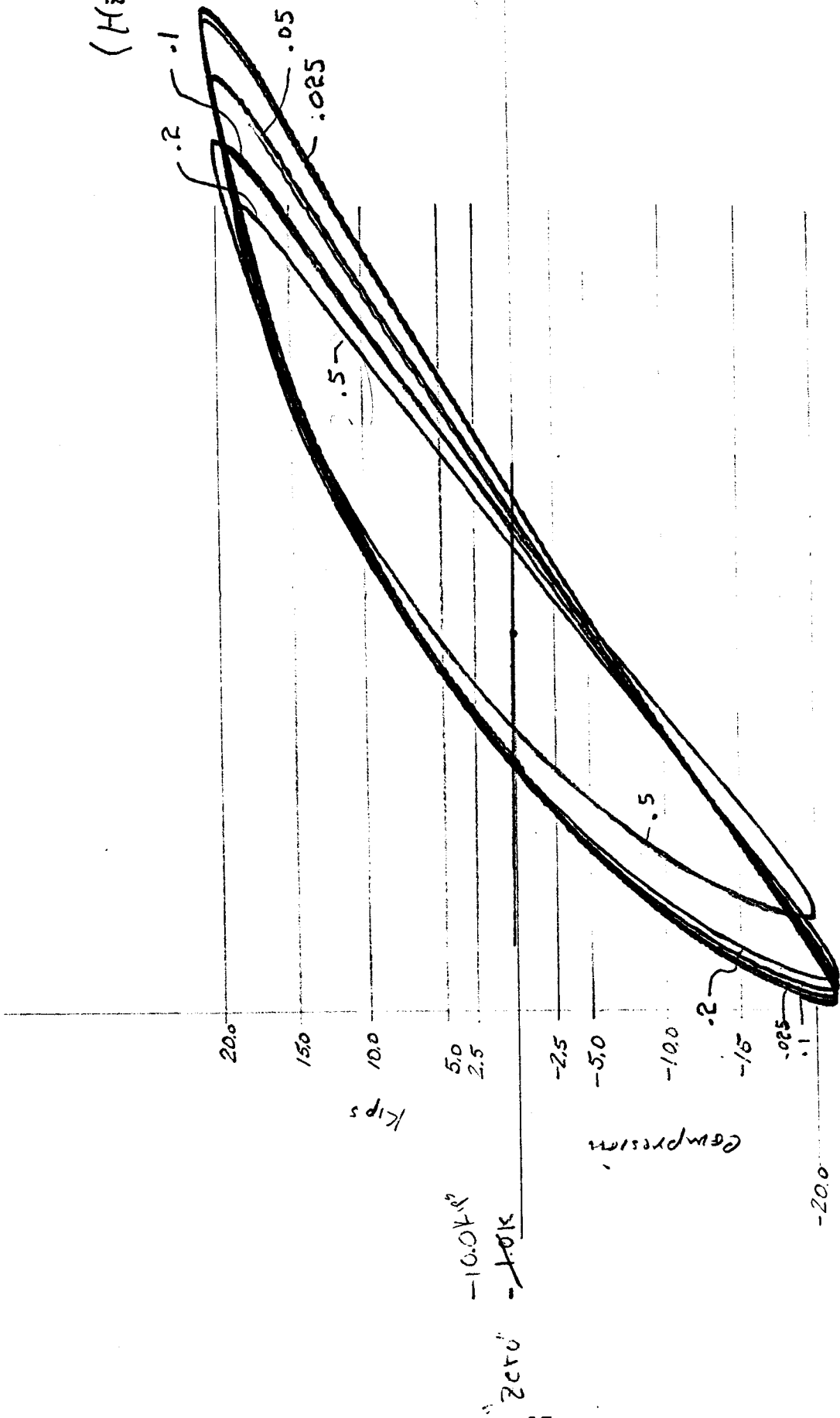


5 "Zero" = 10 K

1.0 (NEAR UPPER LIMIT OF MTS)

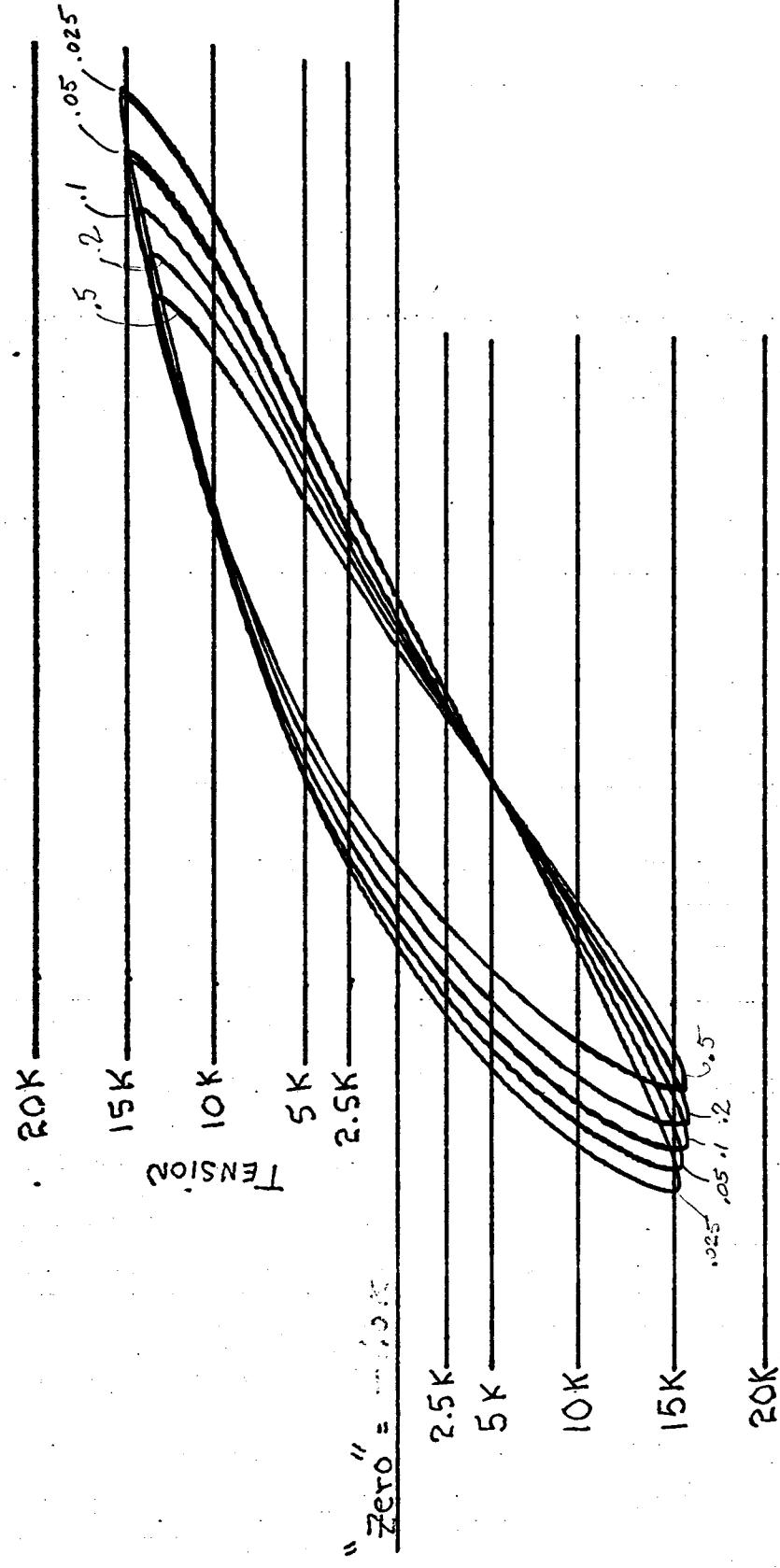
288/21782

(Hz)

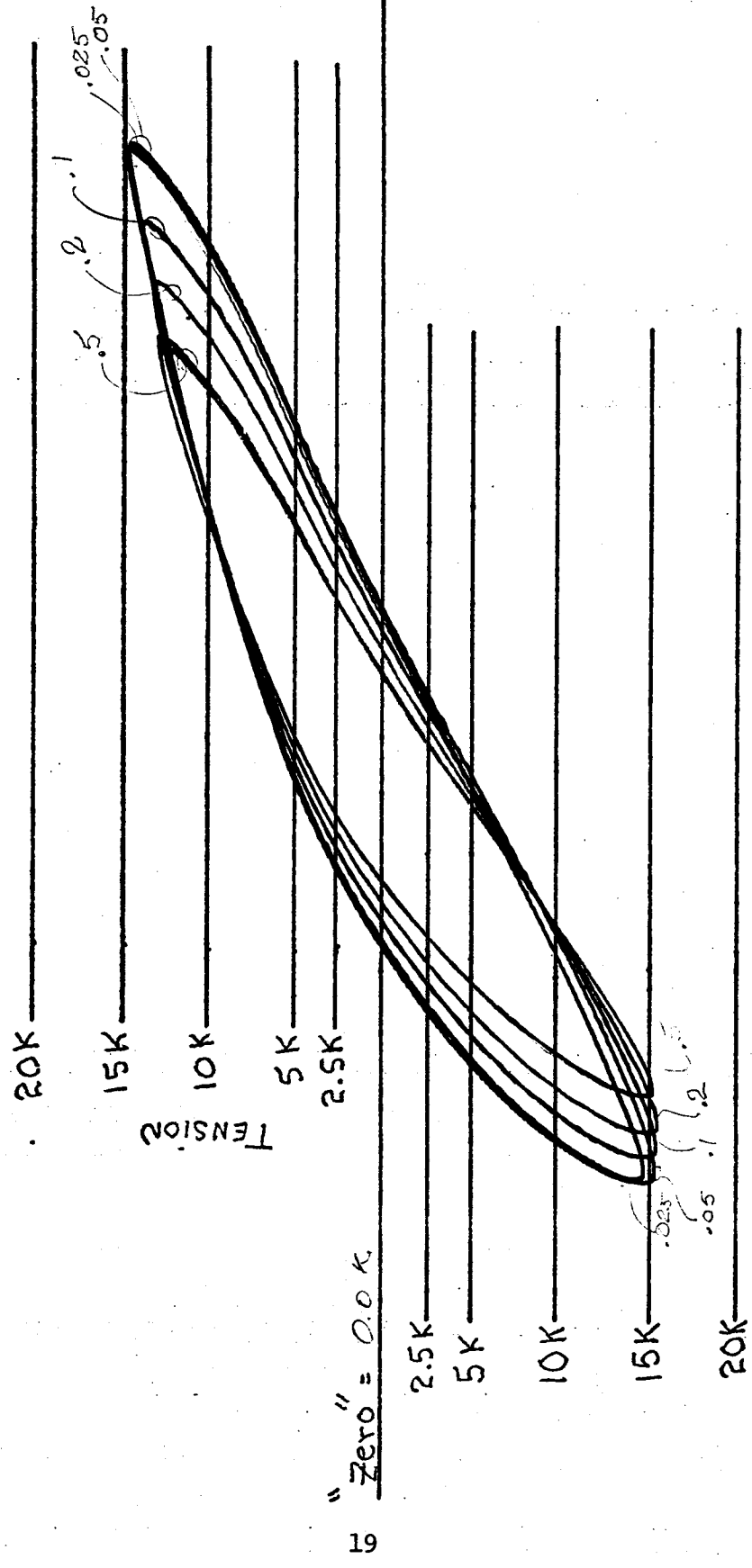


f_o variable

Constant load ampl.

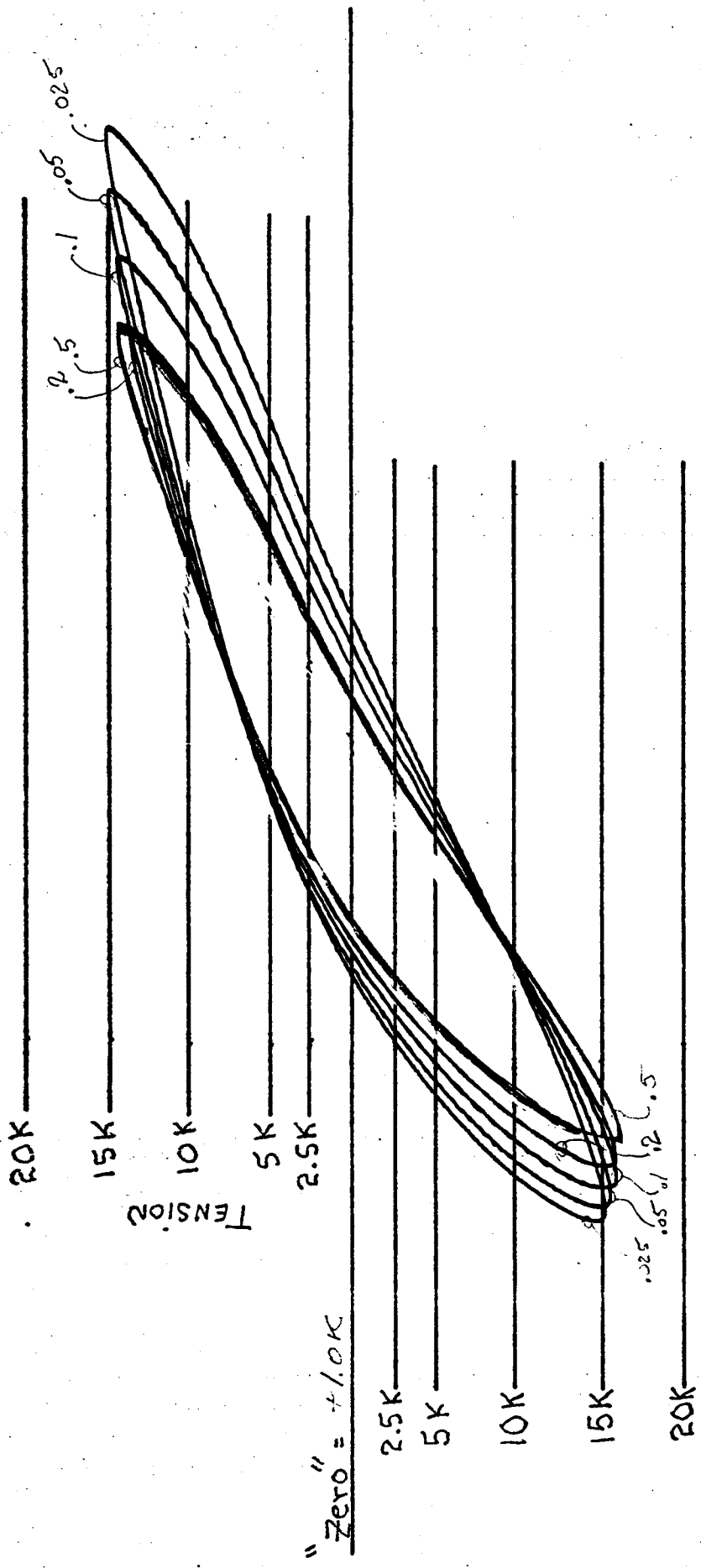


constant load appl.

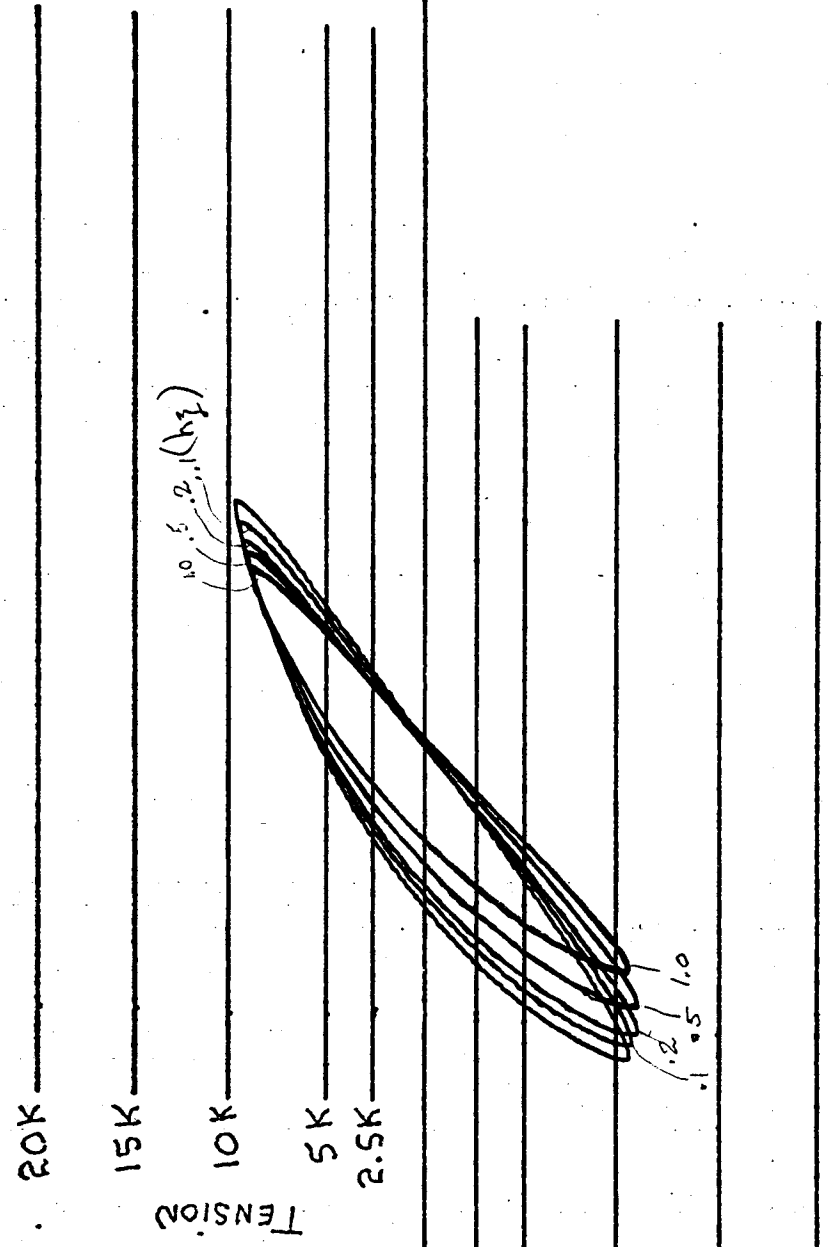


30 JUN 62

f = variable
Constant load ampl.



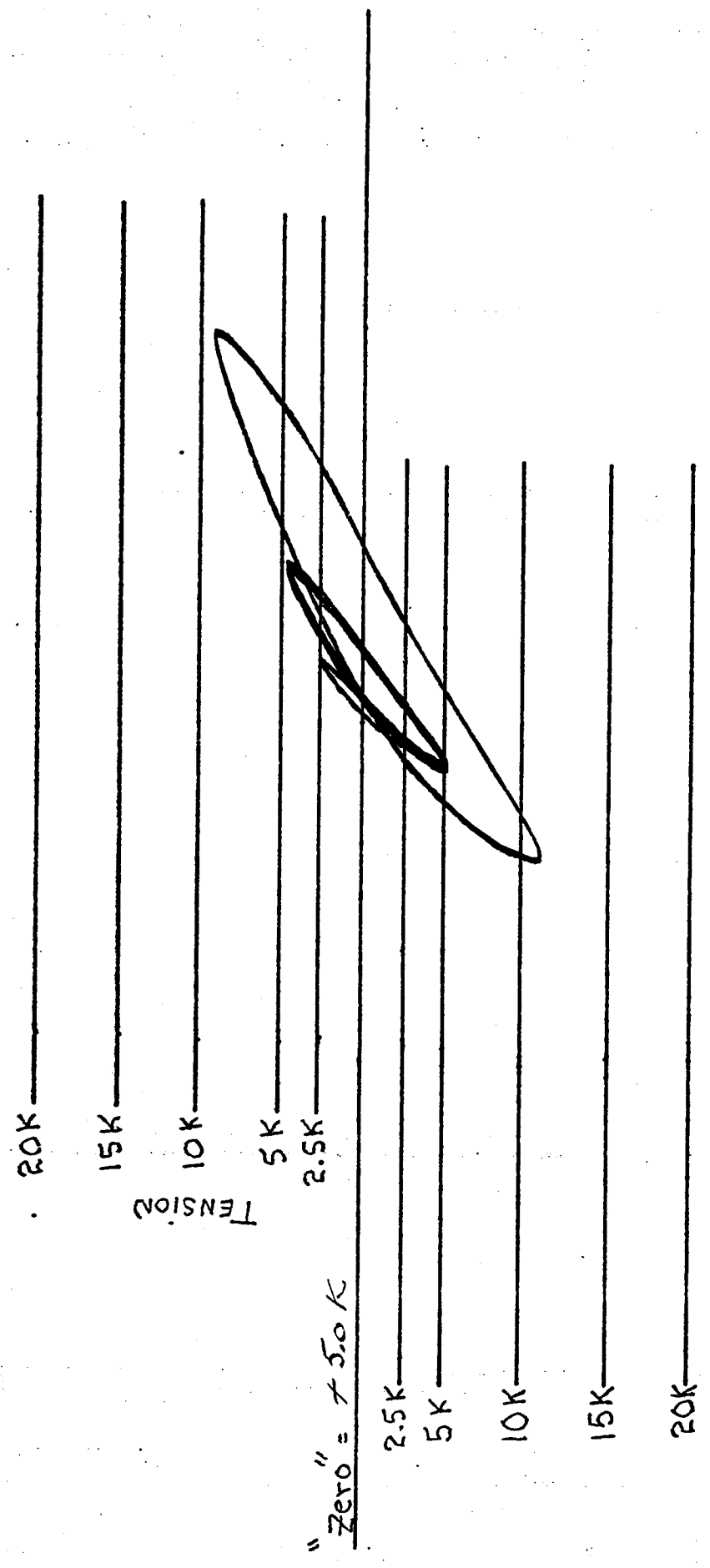
Variable!
Constant force ampl.



"Zero" = -1.0K

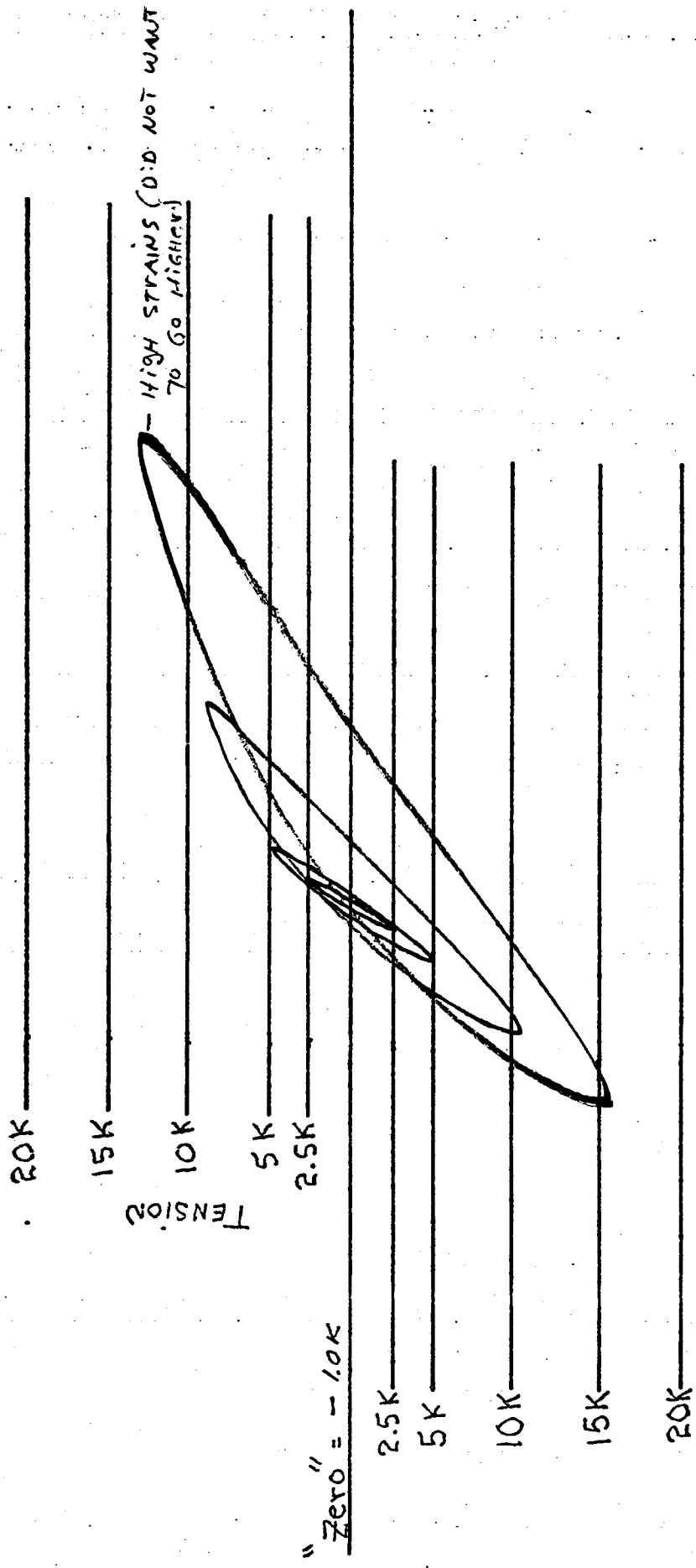
f = 0.5 Hz

50 JEN 82



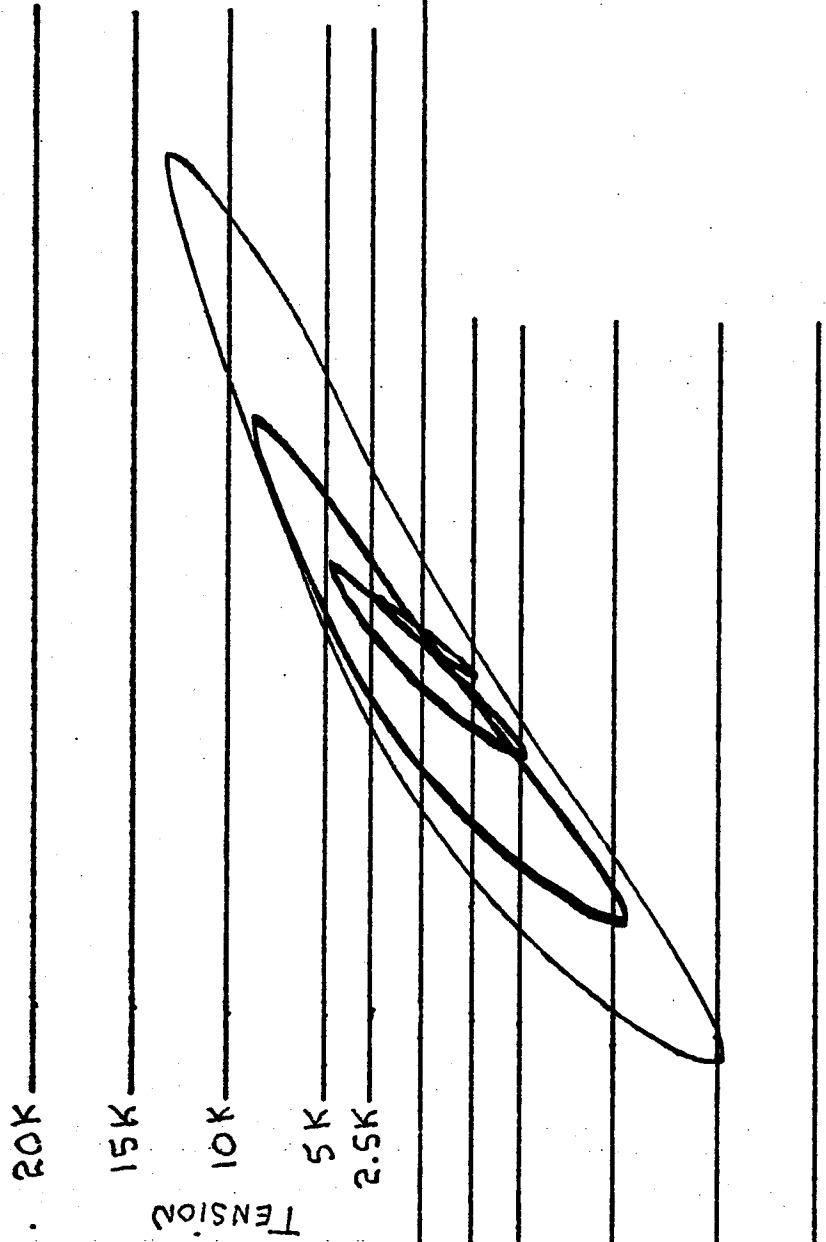
"Zero" = 7.50 K

5/12



30124 62
RHS 1

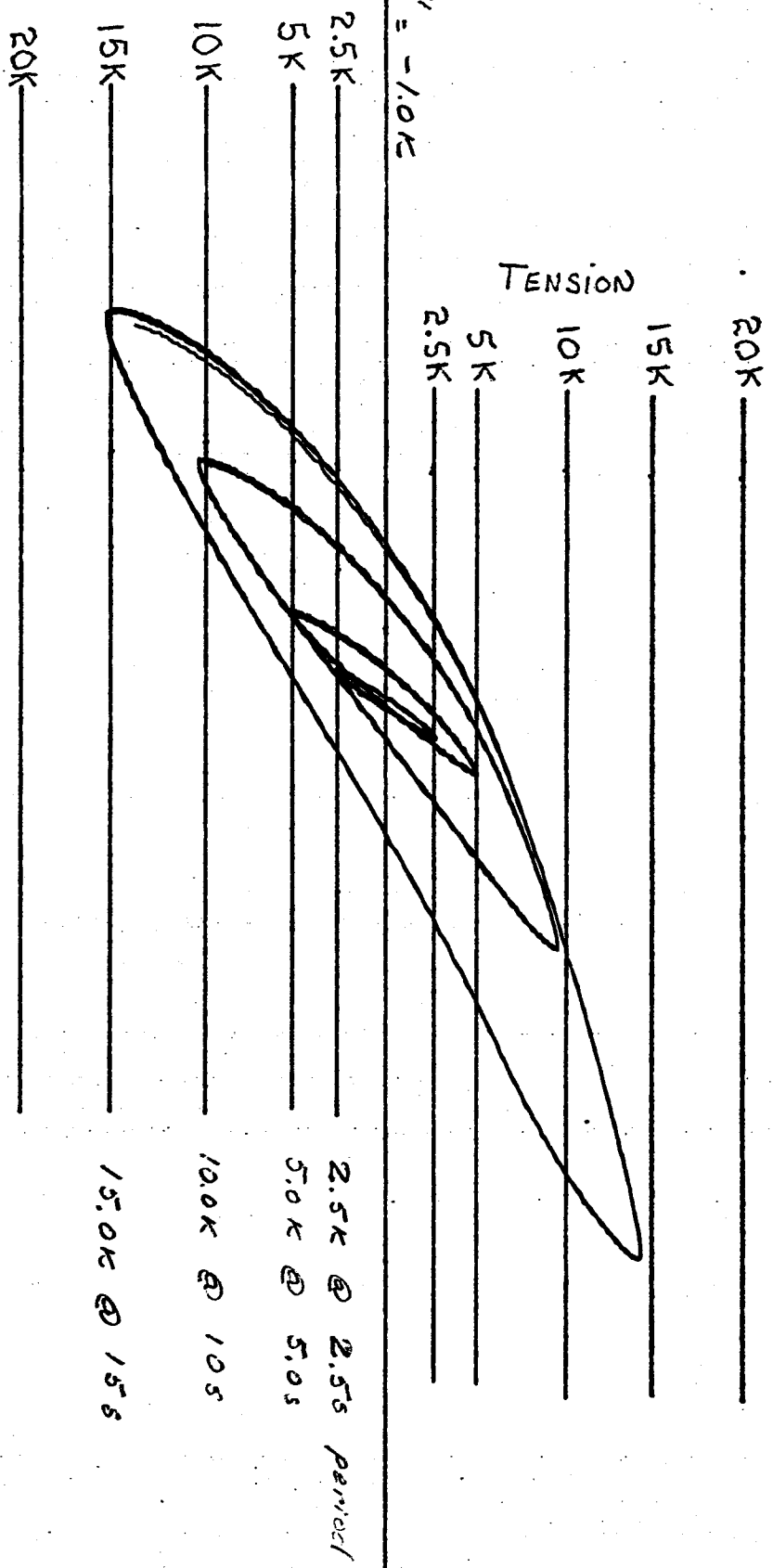
$f = 1.5 Hz$



"Zero" = 7.5K

25

"Zero" = -1.0K



2.5K @ 2.5s period

5.0K @ 5.0s

10.0K @ 10s

15.0K @ 15s

20 @ 20%

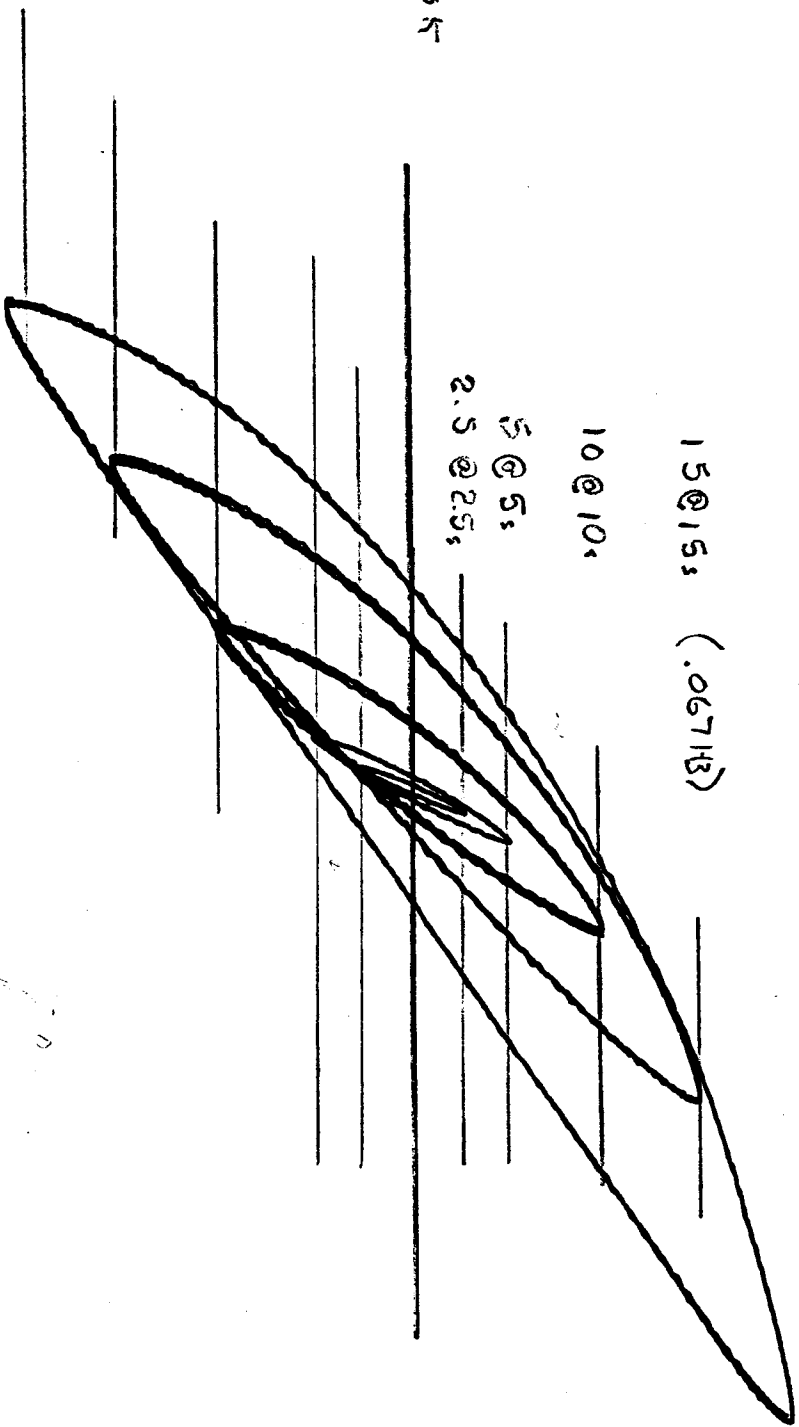
15 @ 15% (.06718)

10 @ 10%

5 @ 5%

2.5 @ 2.5%

-10.0
ZERO = -10K



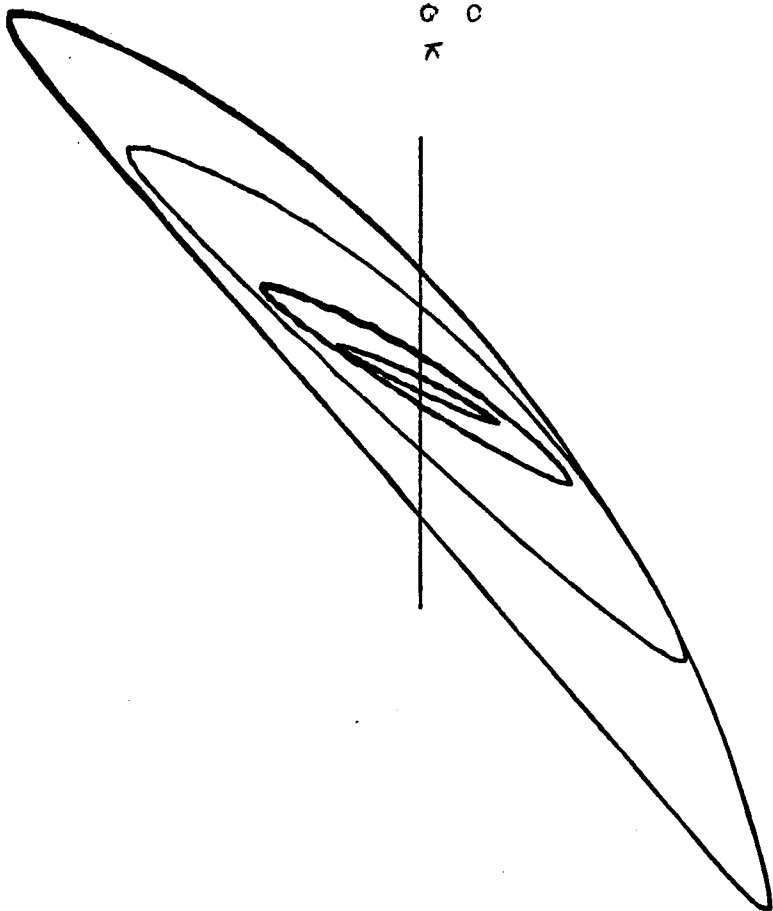
1.0

.25 H3 50M+5K, 10K, 20K, 30K



27

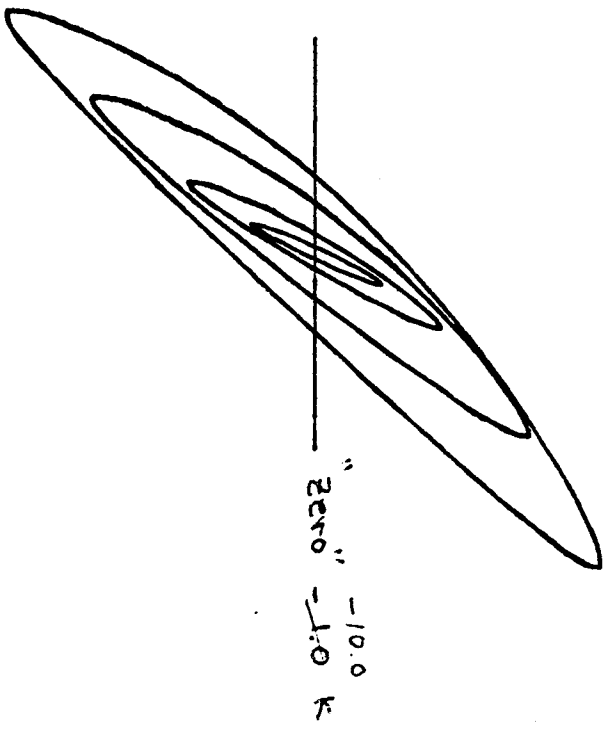
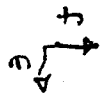
"zero" -10.0
-1.0 K

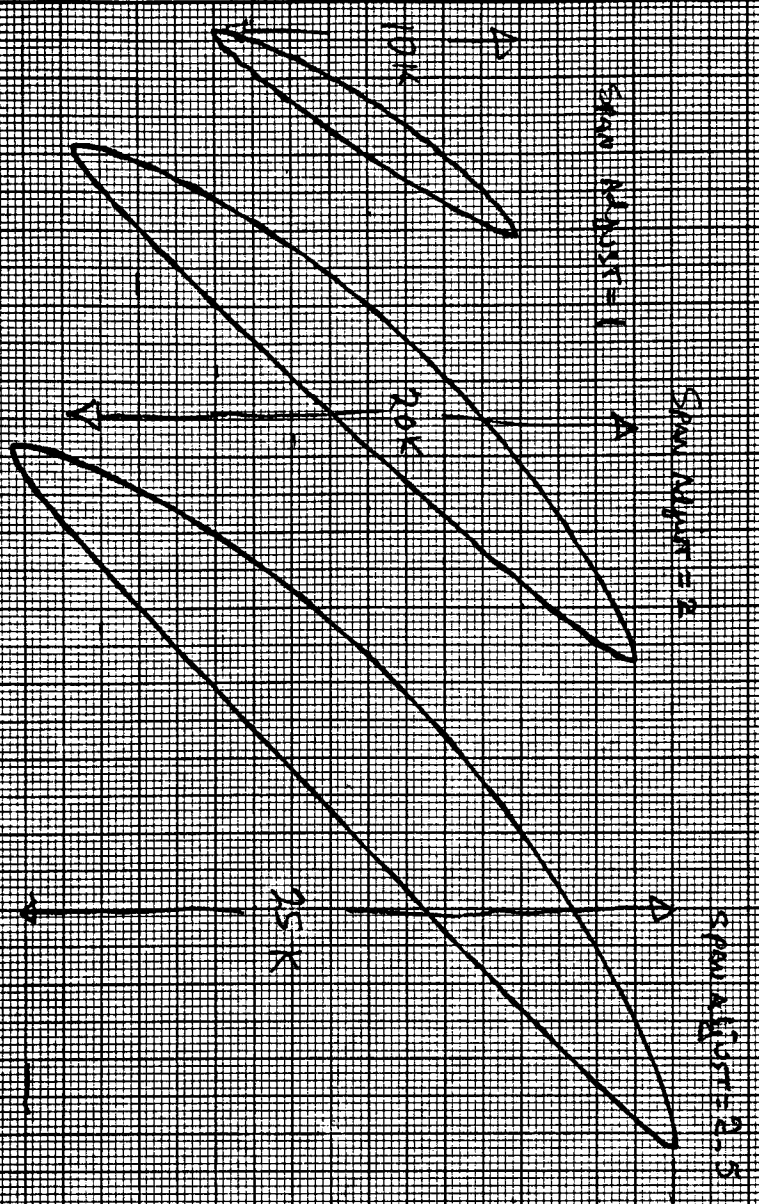
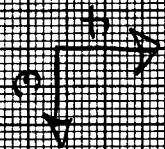


6/29/82

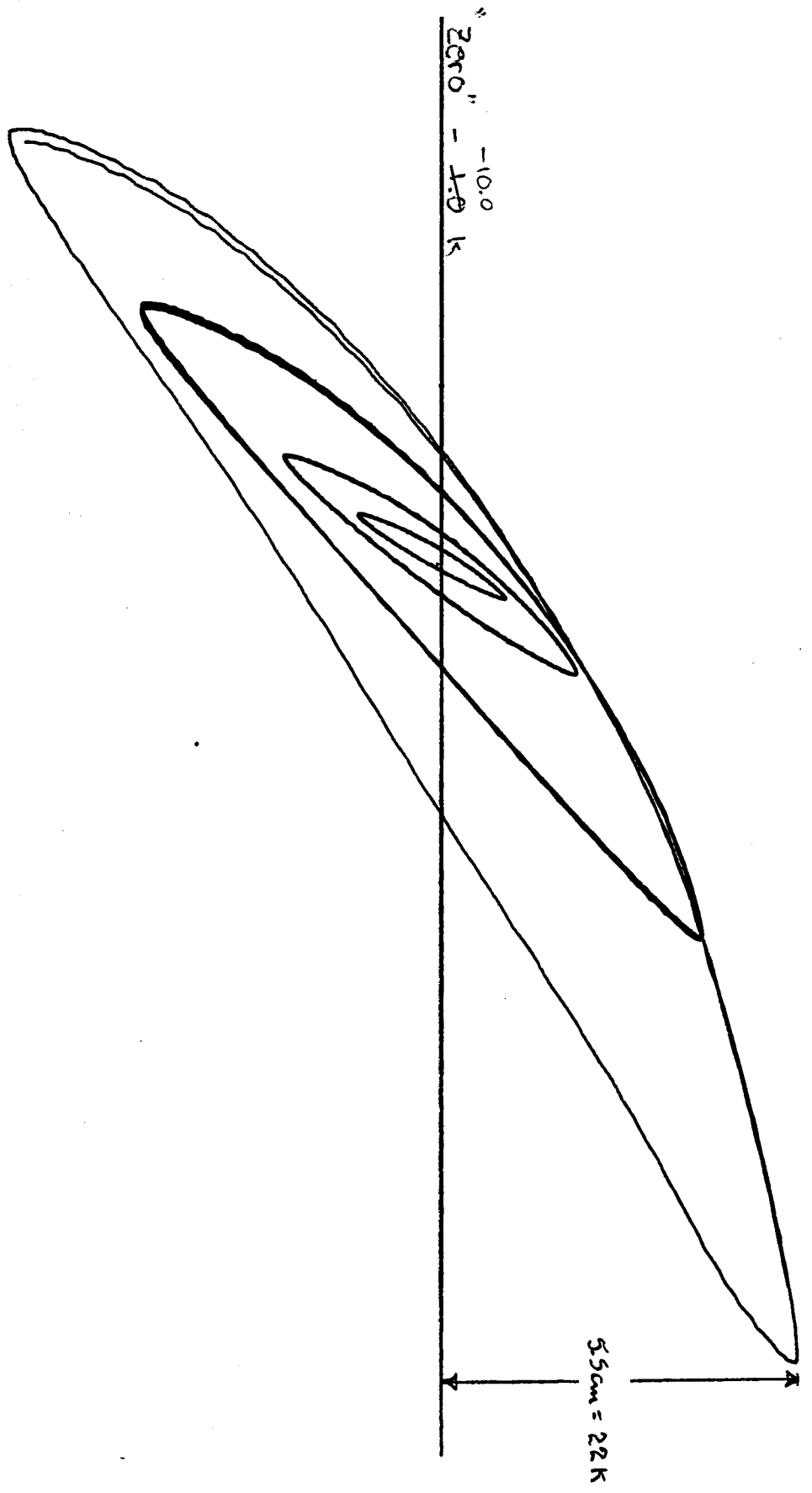


6/29/52

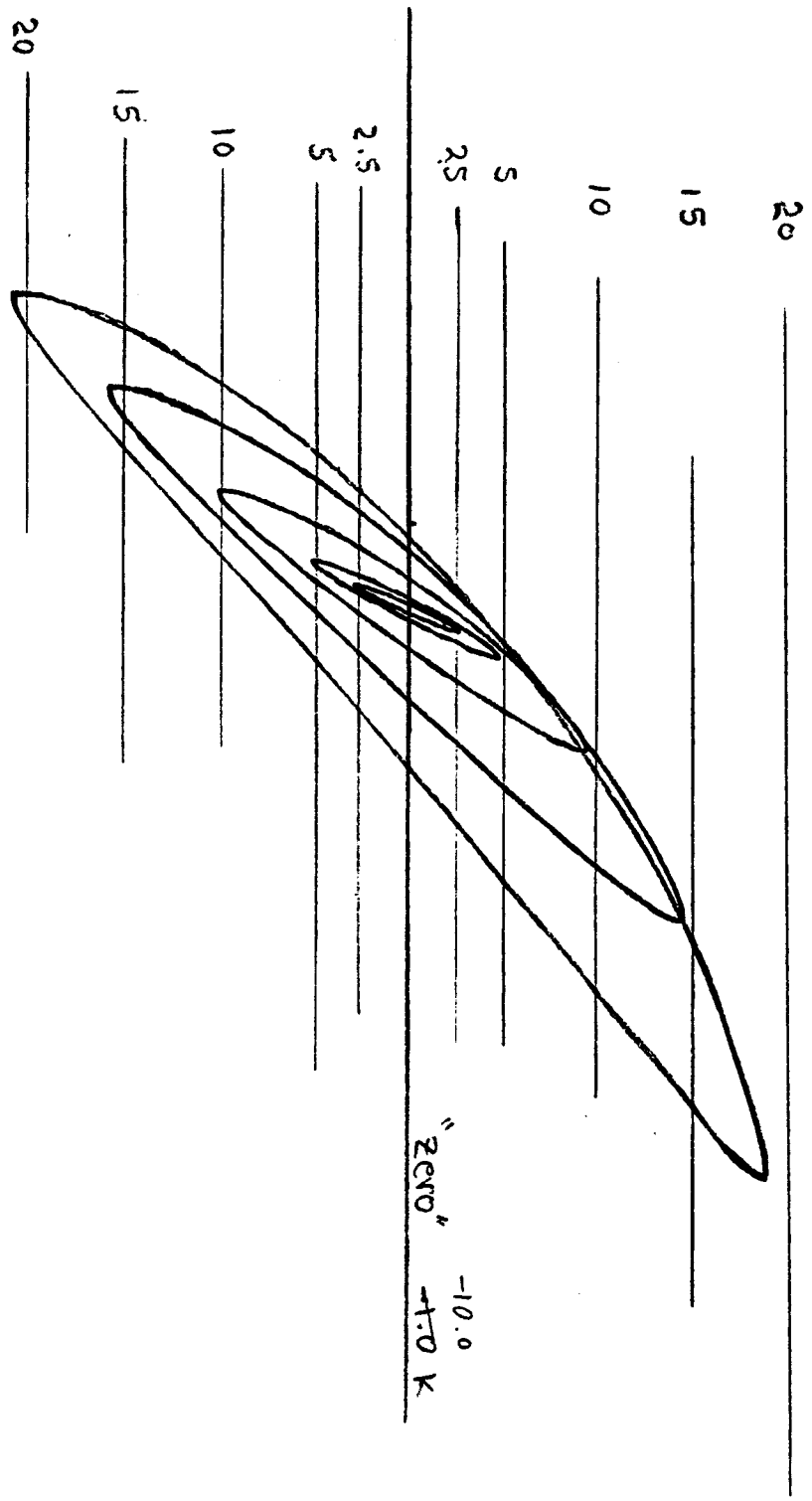




1 M K, 20 K, 29 2



31





200K

150K

100K

50K

25K

-25K

-50K

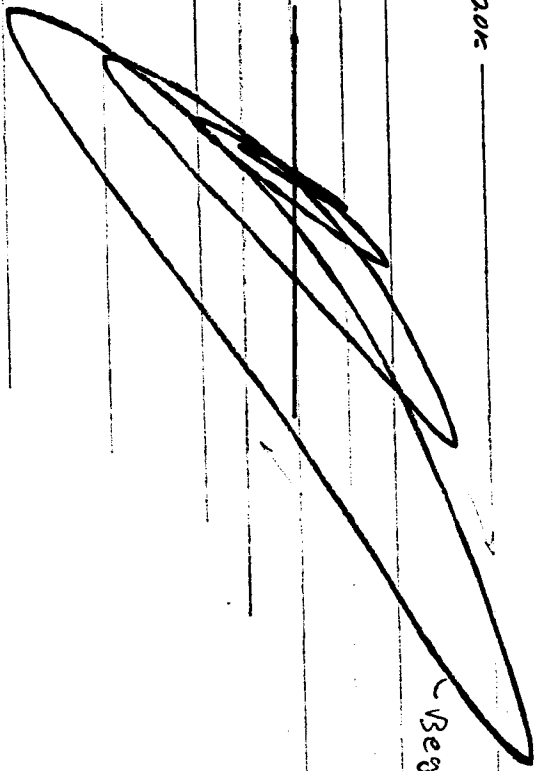
-100K

-150K

-200K

32

"ZERO" = 0.0K



Began to notice large strains in TENSION

NA...

Rubber fender - Tension/Compression 6/23/82

Load V (XDCR2)	Displ (tape measure) (in)	Load V	Displ
	33 1/4	+1000	33 1/16
-2040	33.25	0	33 15/16
-4020	33 7/32	-1000	33 3/4
-6000	33 3/16	-2000	33 11/16
-8000	33 1/8	-4000	33 9/18
-10130	33 1/16	-6000	33 7/16
-12000	32 31/32	-8000	33 5/16
-14110	32 13/16	-1000	33 3/16
-16000	32 3/4	-12000	33 1/32
-18020	32 5/8	-14000	32 7/8
-20300	32 1/2	-16000	32 3/4
-22000	32 7/16	-18000	32 5/8
-24000	32 3/8	-20000	32 1/2
-26000	32 1/4	-22000	32 7/16
-24000	32 1/4	-24000	32 11/32
-22000	32 9/32	-26000	33 1/4
-20000	32 5/16	-24000	32 1/4
-18000	32 11/32	-22000	32 1/4
-16000	32 3/8	-20000	32 5/16
-14000	32 7/15	-18000	32 9/32
-12000	32 1/2	-16000	32 3/8
-10000	32 17/32	-14000	32 13/32
-6000	32 11/16	-12000	32 1/2
-4000	32 3/4	-1000	32 17/32
-2000	32 27/32	-8000	32 5/8
-1000	32 29/32	-6000	32 11/16
0	32 15/16	-4000	32 3/4
+1000	33 1/6	-2000	32 13/16
+4000	33 1/8		32 9/32 1000
+6000	33 1/4		32 15/16 0
+8000	33 1/2	+2000	33 1/16

+10000	33 23/32	+4000	33 1/8
+12000	34	+6000	33 11/32
+14000	34 1/4	+8000	33 1/2
+16000	34 15/32	+10000	33 13/16
+18000	34 23/32	+12000	34 1/16
+20000	34 15/16	+14000	34 5/16
+22000	35 1/4	+16000	34 1/2
+24000	35 7/16	+18000	34 13/16
+22000	35 7/16	+20000	35
+20000	35 3/8	+22000	35 1/4
+18000	35 1/4	+24000	35 15/32
+16000	35 5/32	+22000	35 15/32
+14000	35	+20000	35 15/32
+12000	34 7/8	+18000	35 9/32
+10000	33 23/32	+16000	35 3/16
+8000	34 17/32	+14000	35 1/32
+6000	34 11/32	+12000	34 29/32
+4000	34 1/8	+10000	34 3/4
+8000	34 8/16		
+6000	34 13/32		
+4000	34 7/32		
+2000	34 1/32		
+1000	33 29/32		

Rubber Fender - Vertical Shear Test

Calibration

$$V_{in} = 5.0000V$$

Distance Y	V out
5' 10"	1.3537
5' 9"	1.3344
5' 0"	1.1602
4' 0"	.9282
4' 6"	1.0446
4' 9"	1.1022
5' 3"	1.2183
5' 6"	1.2755

Rubber Fender Shear, Vert. Without Parallel

$V_{in} = 5.000$

	0		
	100#	1.2802	
	500	1.2796	
	1000	1.2787	
	1500	1.2777	
	2000	1.2766	
	2500	1.2755	
	3000	1.2740	
	4000	1.2703	
	4500	1.2681	
	5000	1.2655	
	4000	1.2651	
	3000	1.2662	
	2000	1.2683	
	1000	1.2707	
	0	1.2740	
cycle	0	1.2776	
	5	1.2660	
	0		
	5	1.2658	
	0	1.2748	
	5	1.2658	0
	5	1.2648	15 secs
	5	1.2644	45
	5	1.2640	1.5 m
load	6	1.2603	
	7	1.2555	
	8	1.2491	

9

1.2408

6.500

approx 3 degrees
rotation

5

1.2464

4

1.2492

3

1.2517

2

1.2455

1

1.2590

0

1.2980

Rubber Fender Shear, Vert. With Parallel Bars

$$V_{in} = 5.000$$

Load (K)	Vo
0	1.3049
1	1.3043
2	1.3029
3	1.3012
4	1.2982
5	1.2953
4	1.2950
5	1.2940
6	1.2918
7	1.2875
8	1.2800
8	1.2787
9	1.2749
10	1.2702
6	1.2738
7	1.2730
8	1.2720
9	1.2707
10	1.2688
11	1.2642
12	1.2591

Rubber Fender Horizontal Shear With Parallel Bars

$$V_{in} = 5.000 \text{ V}$$

Load	Vo
0	1.1761
500	1.1755
1.000	1.1746
1.500	1.1738
2.000	1.1725
2.500	1.1710
3.0	1.1696
3.5	1.1678
4.0	1.1660
4.5	1.1640
5.0	1.1614
5.0	1.1596
5.5	1.1582
6.0	1.1564
6.0	1.1540
7.0	1.1518
7.5	1.1494
8.0	1.1468
8.5	1.1442
9.0	1.1413
10.0	1.1352
9.0	1.1328
8.0	1.1334
7.0	1.1349
6.0	1.1365
5.0	1.1385

4.0	1.1410
3.0	1.1434
2.0	1.1464
1.0	1.1496
0.0	1.1542
0.0	1.1562
1.0	1.1556
2.0	1.1539
3.0	1.1519
4.0	1.1498
5.0	1.1472
6.0	1.1449
7.0	1.1422
8.0	1.1395
9.0	1.1369
10.0	1.1334
9.0	1.1311
8.0	1.1321
7.0	1.1333
6.0	1.1350
5.0	1.1369
6.0	1.1379
7.0	1.1367
8.0	1.1350
9.0	1.1332
10.0	1.1310
11.0	1.1277
12.0	1.1226
13.0	1.1160
14.0	1.1080
15.0	1.1009
16.0	1.0934
17.0	1.0859
16.0	1.0853

15.0	1.0850
14.0	1.0864
13.0	1.0881
12.0	1.0899
11.0	1.0920
10.0	1.0945
9.0	1.0967
8.0	1.0994
7.0	1.1018
6.0	1.1053
7.0	1.1089
4.0	1.1130
3.0	1.1175
2.0	1.1223
1.0	1.1274
0.0	1.1324
0.0	1.1393
1.1	1.1377
2.0	1.1360
3.0	1.1341
4.0	1.1320
5.0	1.1292
6.0	1.1261
7.0	1.1231
8.0	1.1194
9.0	1.1157
10.0	1.1115
11.0	1.1077
12.0	1.1036
13.0	1.1002
14.0	1.0959
15.0	1.0920
16.0	1.0876
17.0	1.0835
16.0	1.0816

15.0	1.0814
14.0	1.0829
13.0	1.0846
12.0	1.0864
11.0	1.0886
10.0	1.0912
9.0	1.0935
8.0	1.0962
7.0	1.0994
6.0	1.1021
5.0	1.1060
4.0	1.1096
3.0	1.1141
2.0	1.1187
1.0	1.1239
0.0	1.1297

Rubber Fender Tension/Compression

6/23/84

7:39 pm

110k MTS

Load V	Tape Measure
0	33 1/8
1.30	
-24k	32 3/8
+20.5	34 7/8
+21.3	35.0
+24	35 7/16
-24	32 3/8

Load (#)	Disp. (volts)
1010	1.026
2040	.954
4070	.788
6000	.577
8000	.347
10130	.016
	-.295
13110	-.805
16000	-1.198
18020	1.680
20300	2.169
22000	2.472
24000	-2.790
26000	-3.157
24000	-3.169
22000	-3.082
20000	-2.981
18000	-2.854
16000	-2.701
14000	-2.507
12000	-2.292
10000	-2.053
6000	-1.458
4000	-1.085
2000	-.724
1000	-.474
000	-.266
+2000	.239
4000	.558
6000	1.247
8000	2.055
10000	3.048

12000	4.180
14000	5.141
16000	6.083
18000	7.061
20000	8.042
+22000	9.103
24000	10.077
22000	9.889
20000	9.600
18000	9.214
16000	8.751
14000	8.203
12000	7.580
10000	7.033
8000	6.211
6000	5.518
4000	4.661
2000	3.924
0	3.334
-1000	2.972
-2000	
-4000	2.187
-6000	1.600
-8000	1.060
-10000	.611
-12000	.066
-14000	.680
-16000	1.199
-18000	1.737
-20000	2.180
-24000	3.218
-22000	3.140
-20000	3.026
-18000	2.888
16000	2.715

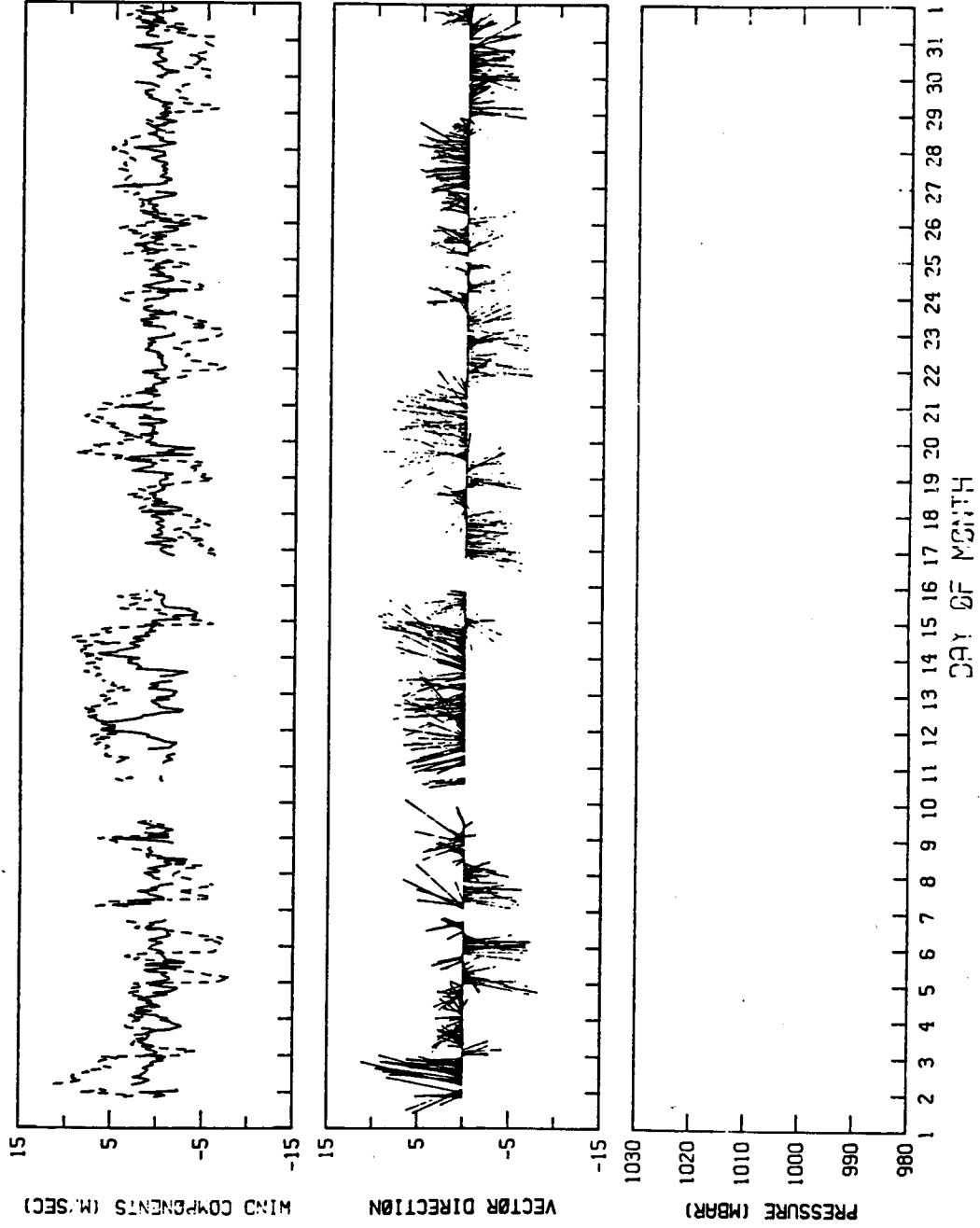
14000	2.546
12000	2.236
10000	2.040
8000	1.741
6000	1.432
4000	1.107
2000	.742
1000	.472
000	.290
2000	.221
4000	.750
6000	1.437
8000	2.260
10000	3.442
12000	4.463
14000	5.383
16000	6.380
18000	7.445
20000	8.351
22000	9.254
24000	10.073
22000	10.026
20000	9.720
18000	9.298
16000	8.851
14000	8.317
12000	8.779
10000	7.097
8000	6.380
6000	5.607
4000	4.771
2000	4.149
000	3.524
-1000	3.129

Calibration (proving ring) Zero from Rubber Fender Tests = .008
Re-Zero to .002

Load XDCR2	Kips	Proving Ring
.100	1	14.5
.200	2	27.3
.400	4	53.8
.600	6	80.0
.800	8	106.0
1.000	10	131.5
1.200	12	157.4
1.400	14	183.0
1.600	16	208.8
1.800	18	235.2
2.000	20	262.0
2.200	22	288.4
2.400	24	314.5
2.600	26	340.0
2.200	22	289.0
1.800	18	236.5
1.400	14	183.0
1.000	10	132.5
.600	6	80.5
.200	2	28.0

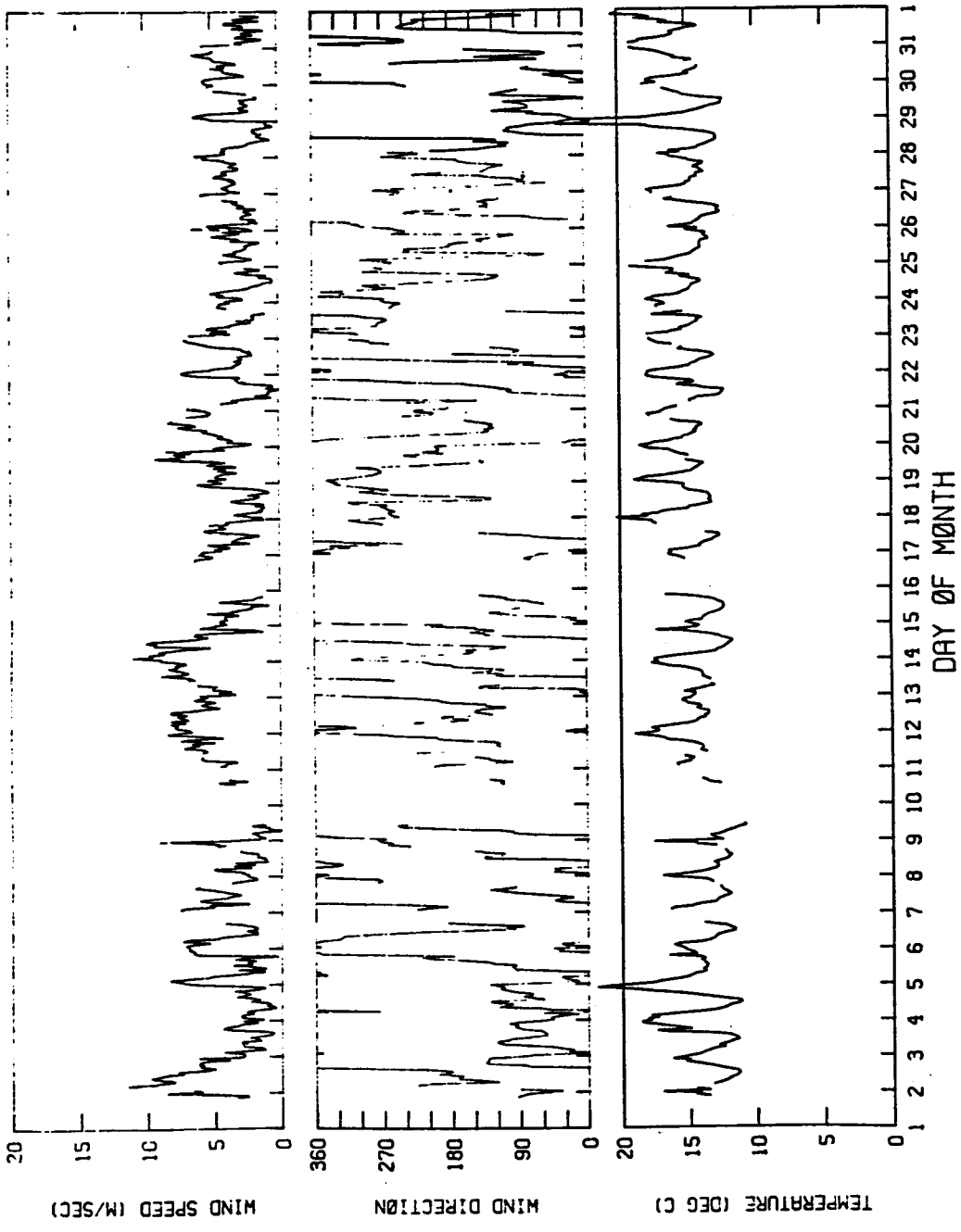
APPENDIX VII
SUPPLEMENTAL DATA

TIME SERIES PLOT ØR ALKI PØINT BUØY
 1 JUL 83 TØ 1 AUG 83 INTERVAL= 60.0MINS



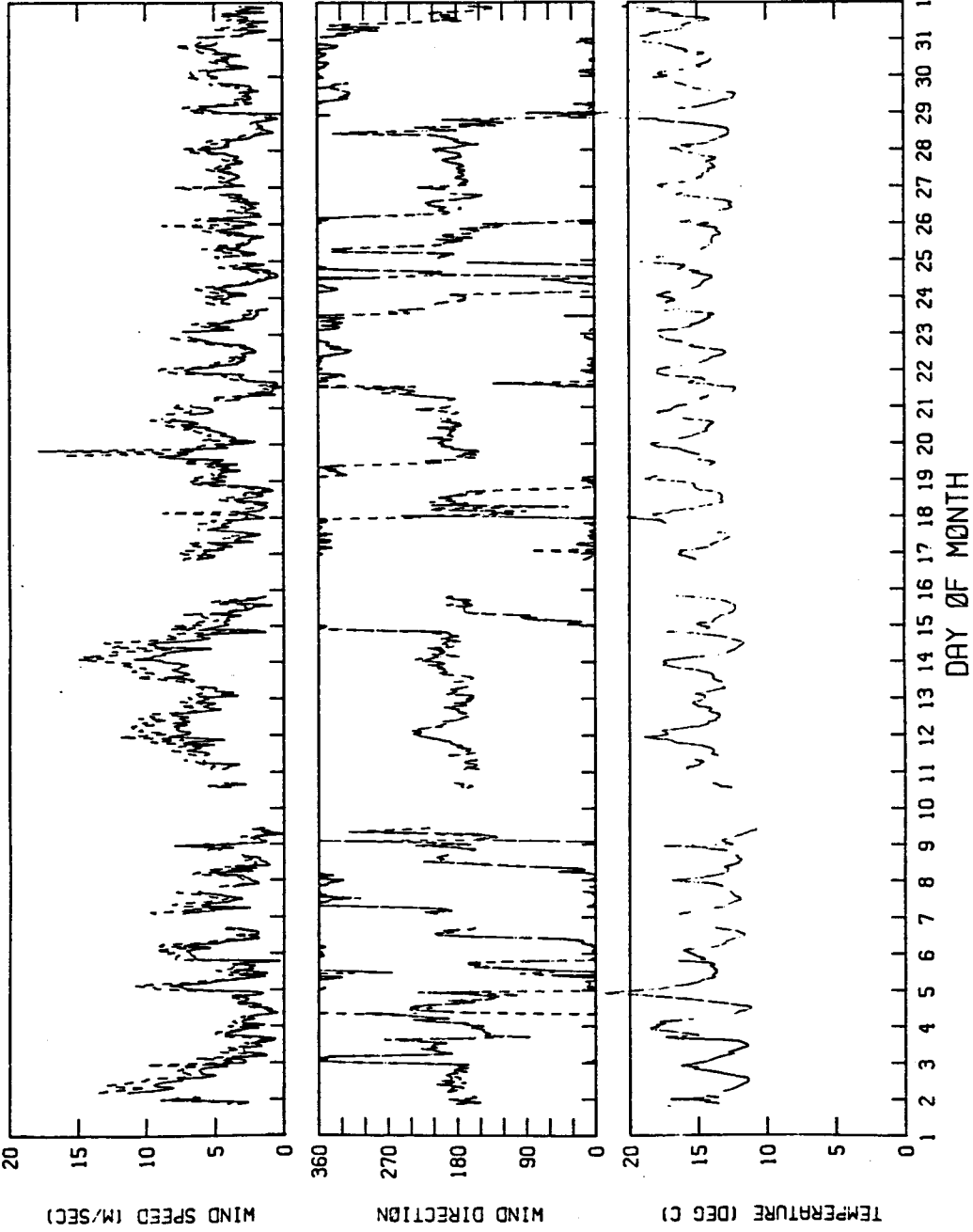
TIME SERIES PLOT FOR ALKI PØST BUDØY

1 JUL 83 TO 1 AUG 83 INTERVAL = 60.0MINS



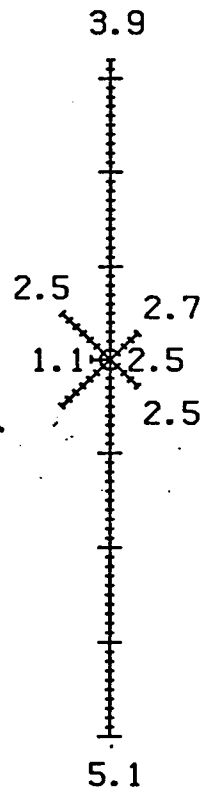
TIME SERIES PLOT FOR ALKI PONT BUOY

1 JUL 83 TO 1 AUG 83 INTERVAL = 60.0MINS



WIND RØSE

TIME SERIES PLØT FØR ALKI PØINT BUØY
1 JUN 83 TØ 1 JUL 83 INTERVAL= 60.0MINS



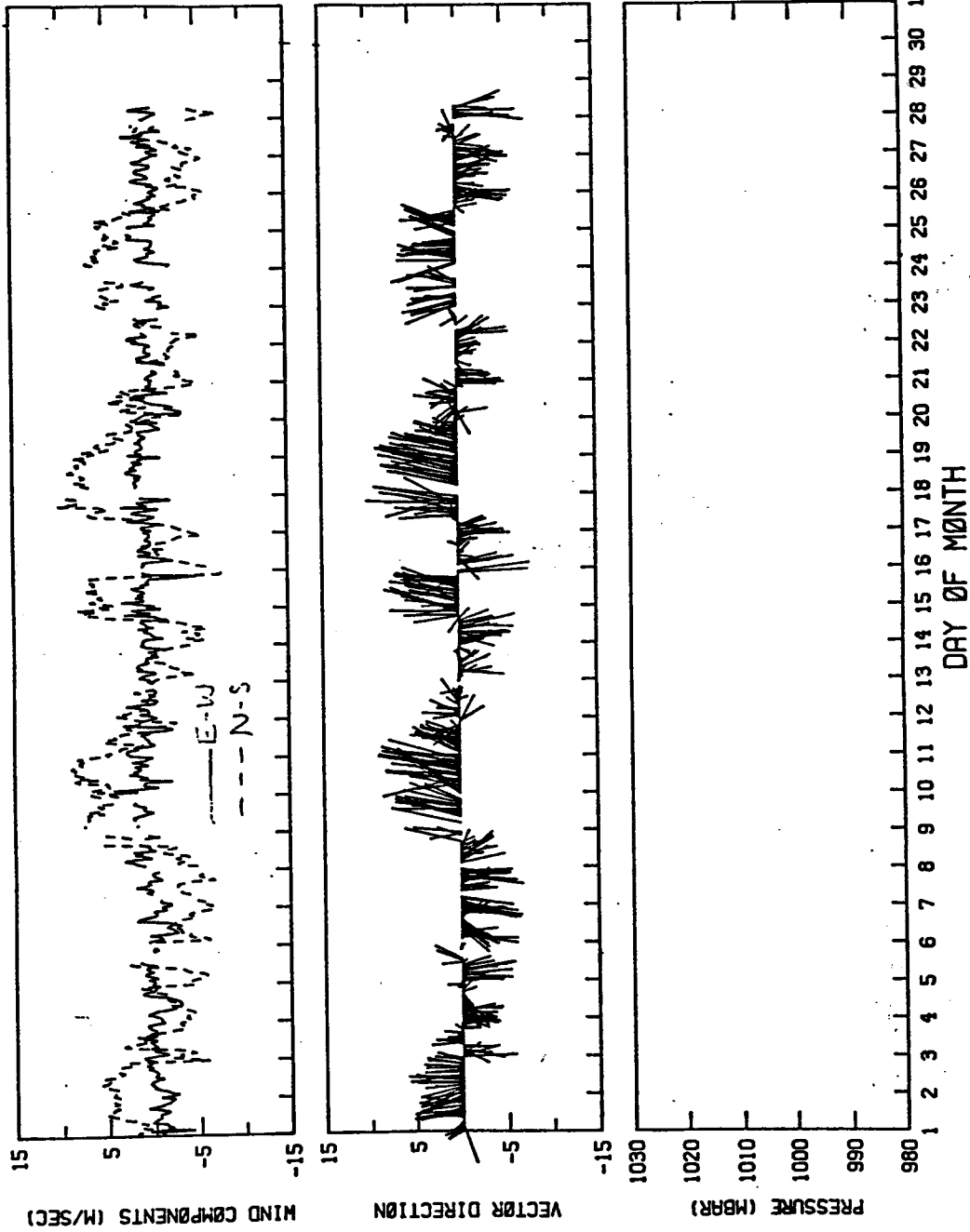
NPTS = 586

CALM = 0.0

BAD = 114

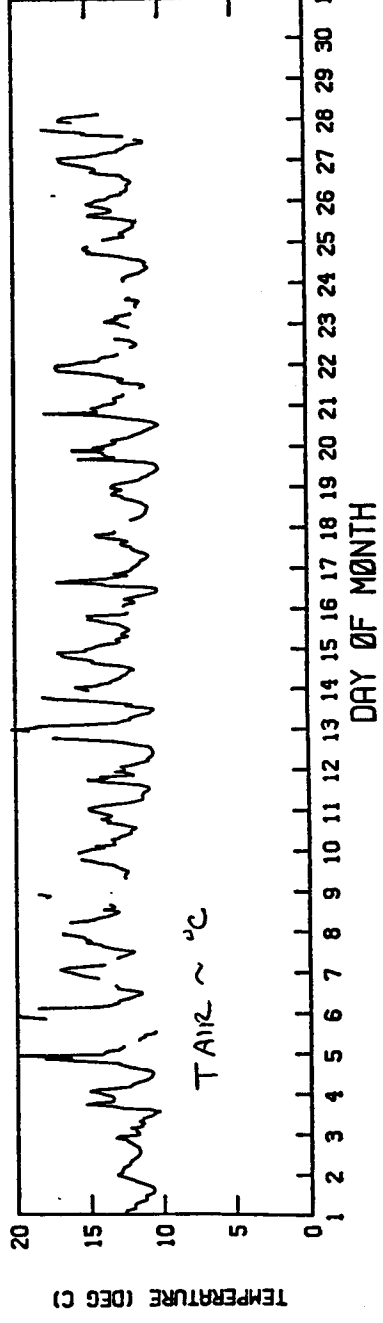
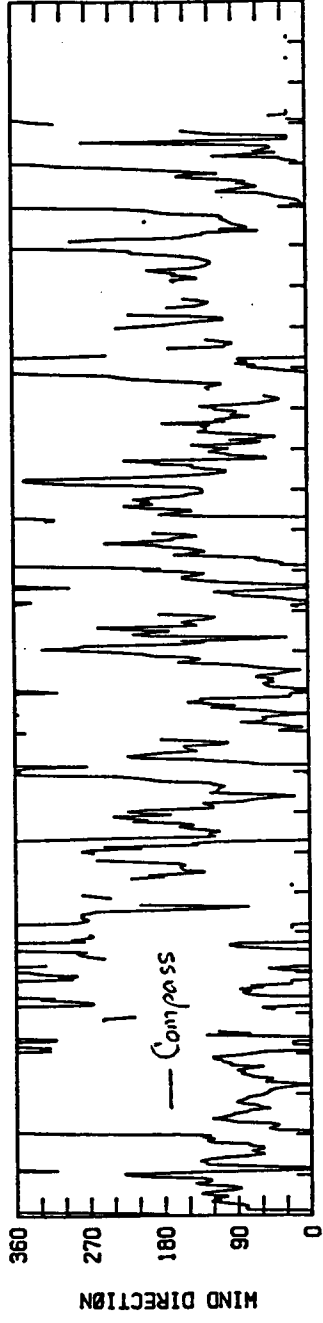
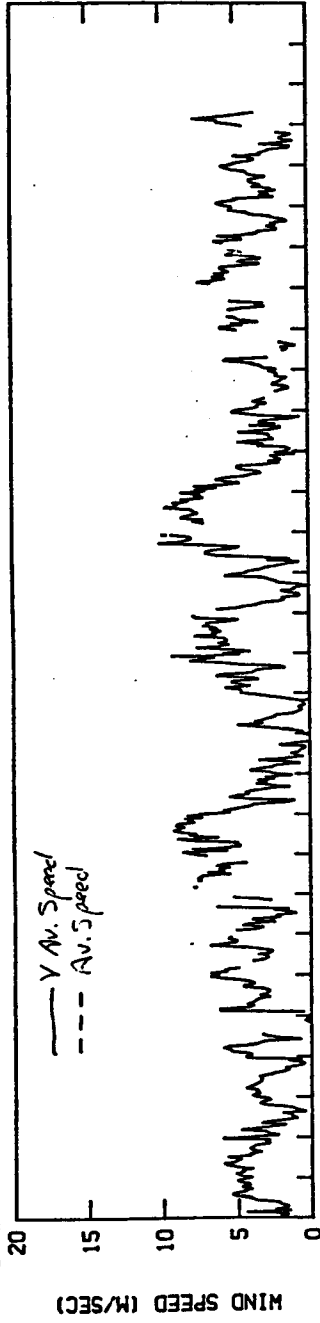
TIME SERIES PLOTTING FOR ALKI POINT BUOY

1 JUN 83 TO 1 JUL 83 INTERVAL= 60.0MINS



TIME SERIES PLOT FOR ALKI PØ...T BUØY

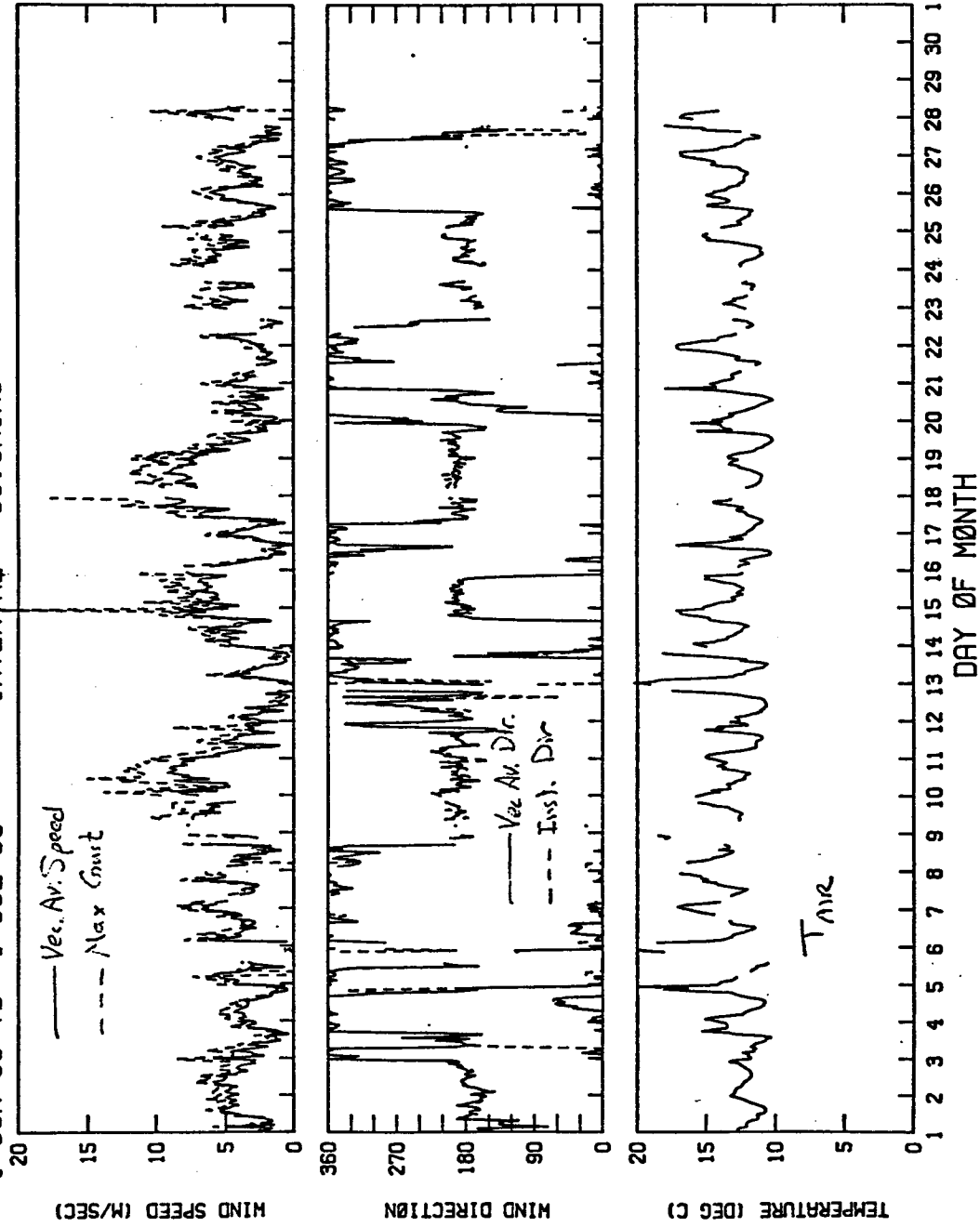
1 JUN 83 TØ 1 JUL 83 INTERVAL= 60.0MINS



0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 1
DAY OF MONTH

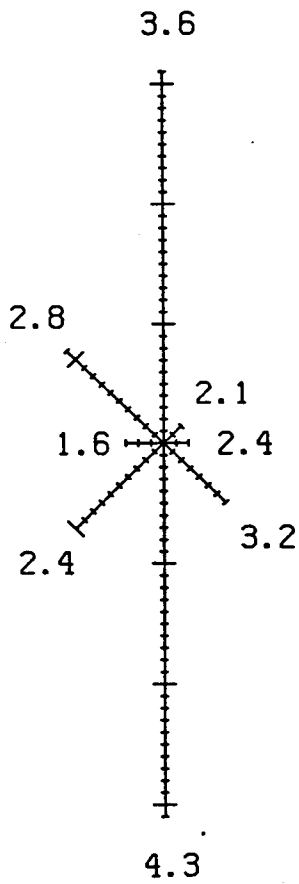
TIME SERIES PLOT FOR ALKI PØINT ØJØY

1 JUN 83 TØ 1 JUL 83 INTERVAL= 60.0MINS



WIND RØSE

TIME SERIES PLOT FOR ALKI POINT BUØY
1 MAY 83 TO 1 JUN 83 INTERVAL= 60.0MINS



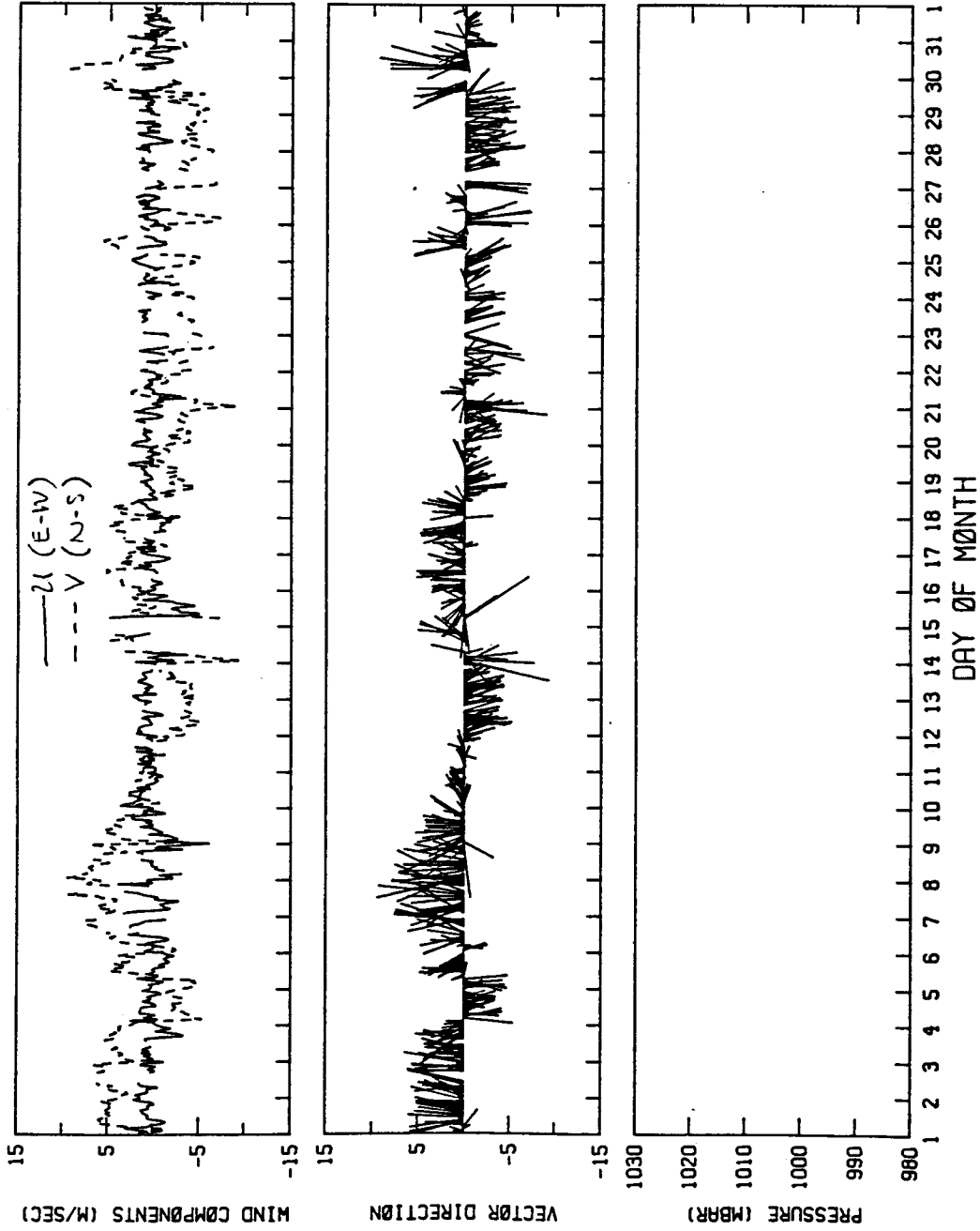
NPTS = 632

CALM = 0.0

BAD = 110

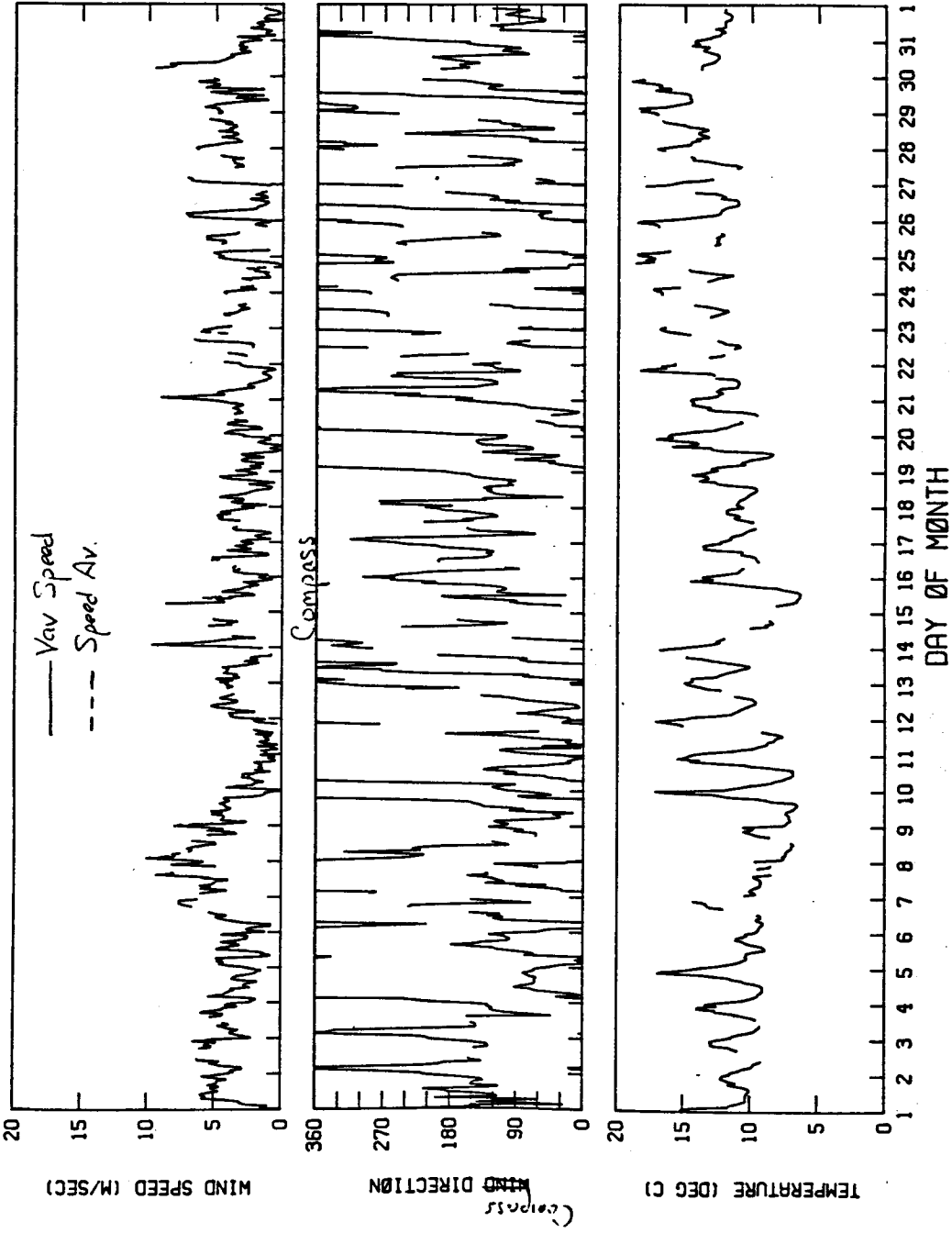
TIME SERIES PLOT FOR ALKI PØINT BUØY

1 MAY 83 TO 1 JUN 83 INTERVAL= 60.0MINS



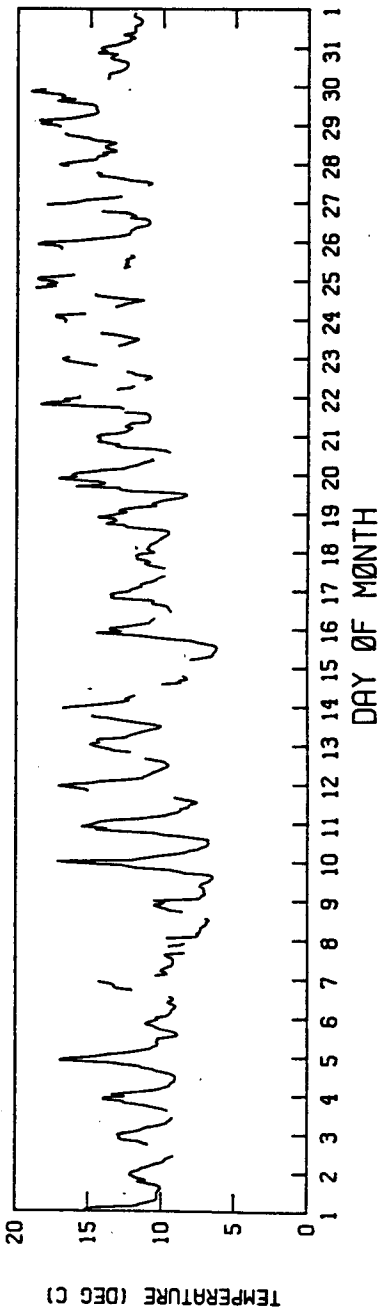
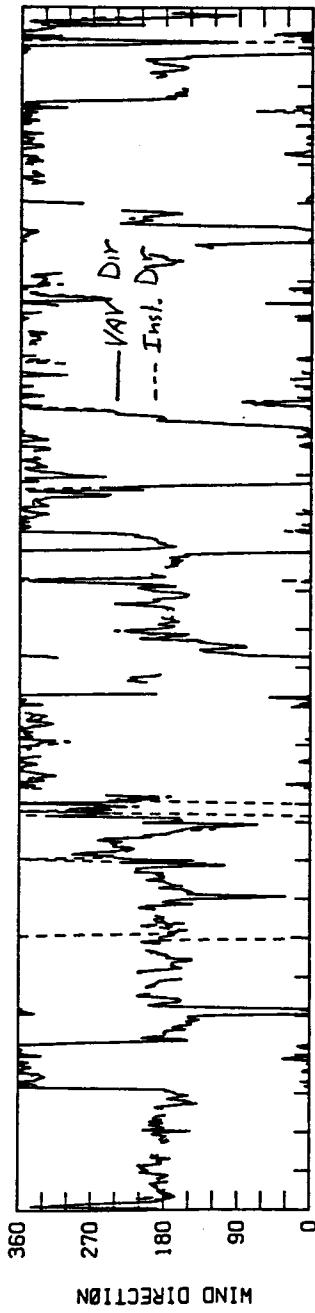
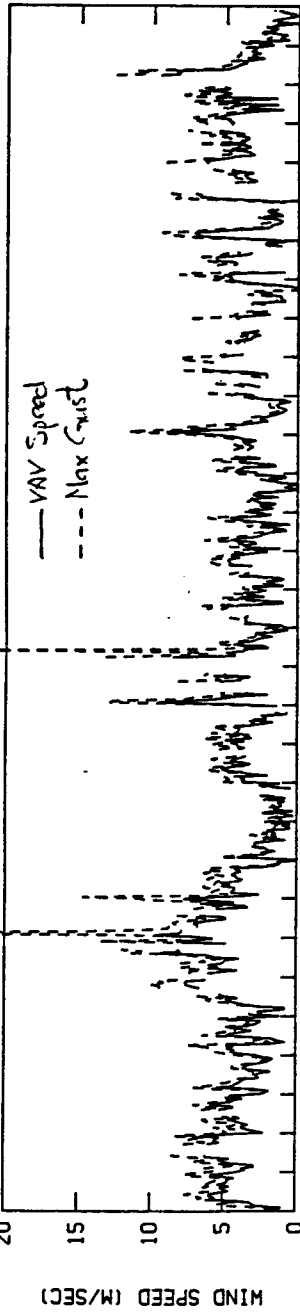
TIME SERIES PLOT FOR ALKI PØI.NI BUØY

1 MAY 83 TO 1 JUN 83 INTERVAL= 60.0MINS



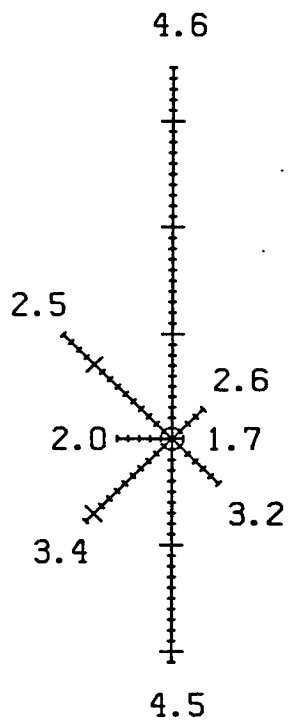
TIME SERIES PLOT FOR ALKI PØI BUØY

1 MAY 83 TØ 1 JUN 83 INTERVAL = 60.0MINS



WIND RØSE

TIME SERIES PLOT FØR ALKI PØINT BUØY
1 APR 83 TØ 1 MAY 83 INTERVAL= 60.0MINS

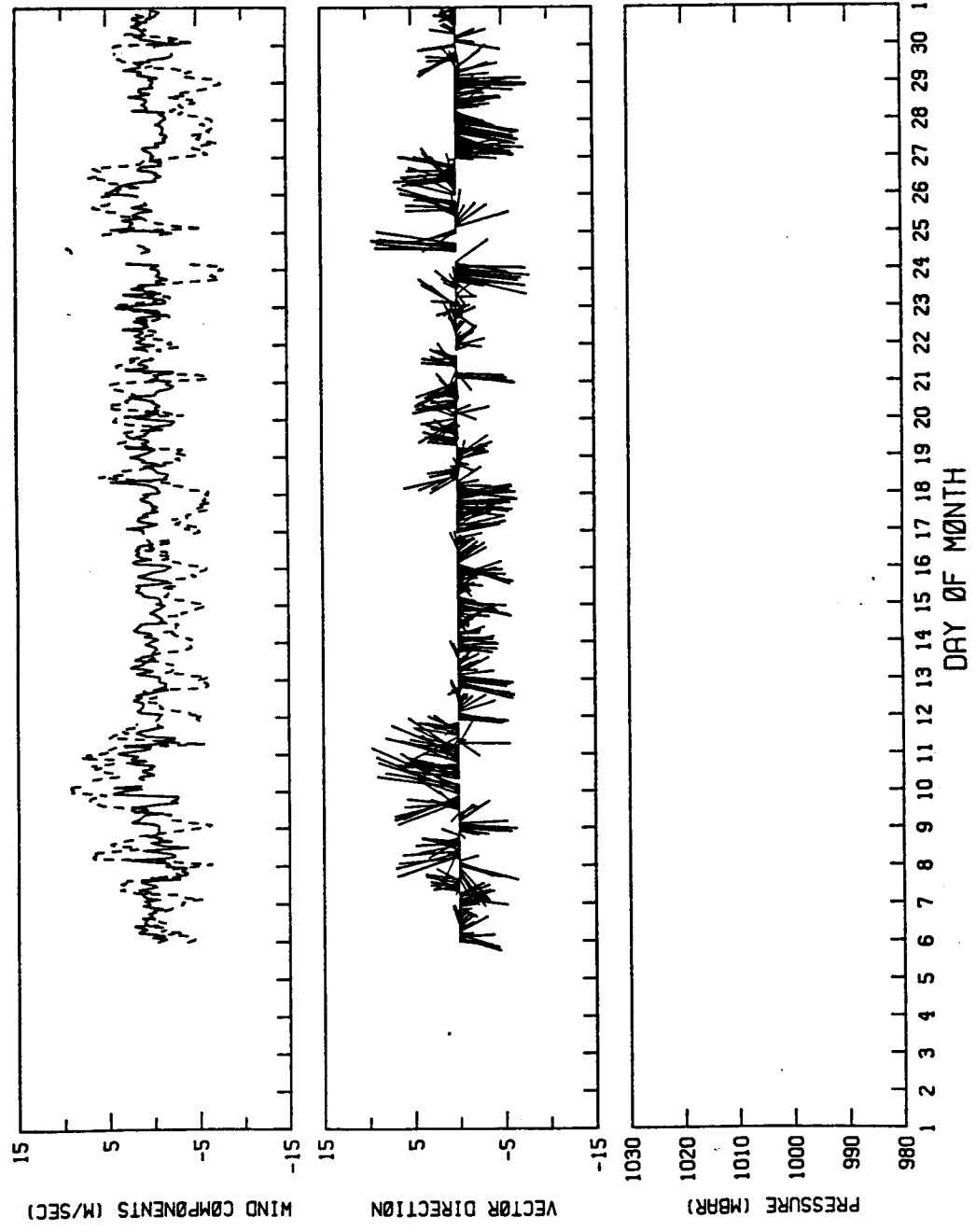


NPTS = 571

CALM = 0.0

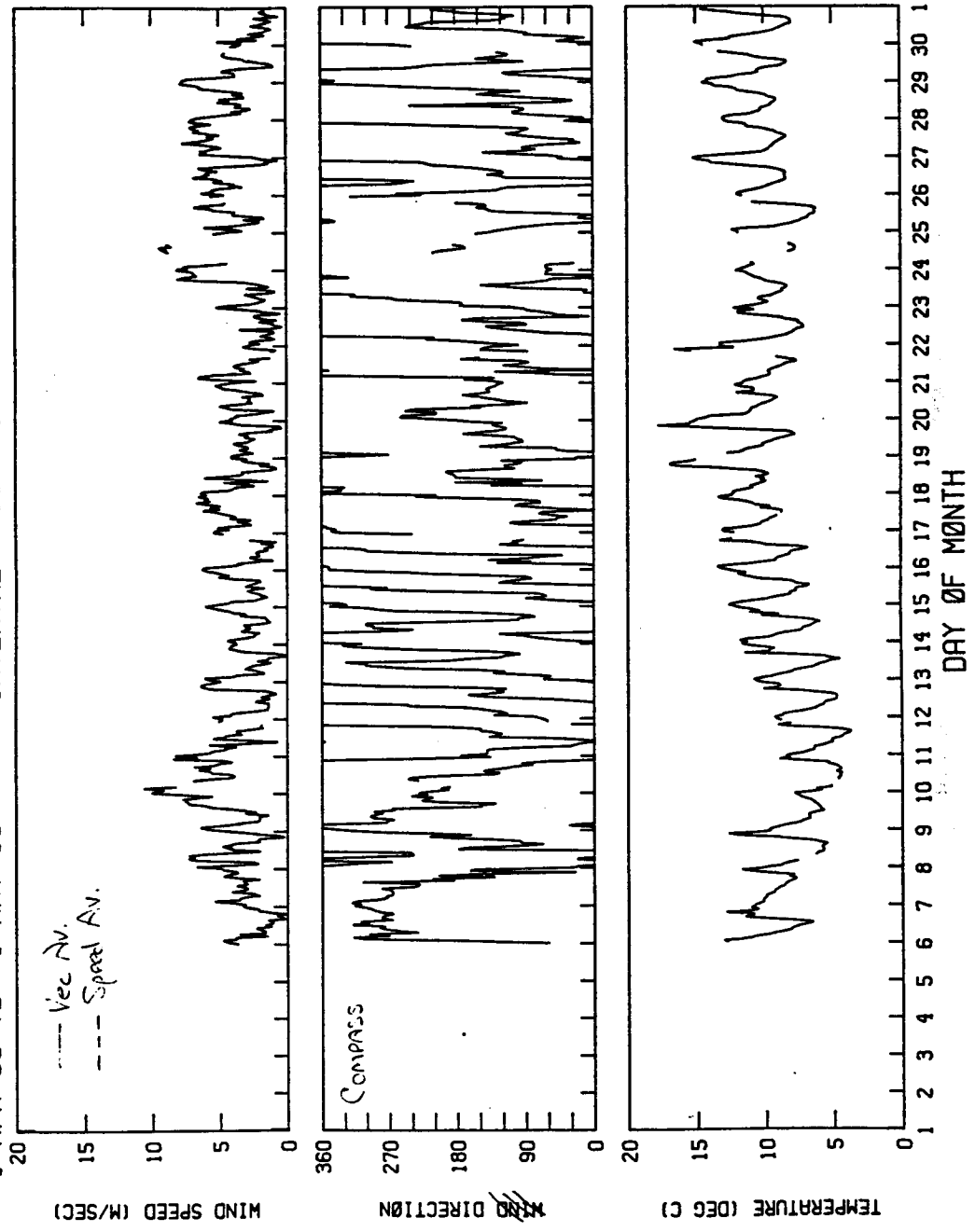
BAD = 28

TIME SERIES PLOT FOR ALKI POINT BUOY
 1 APR 83 TO 1 MAY 83 INTERVAL= 60.0MINS



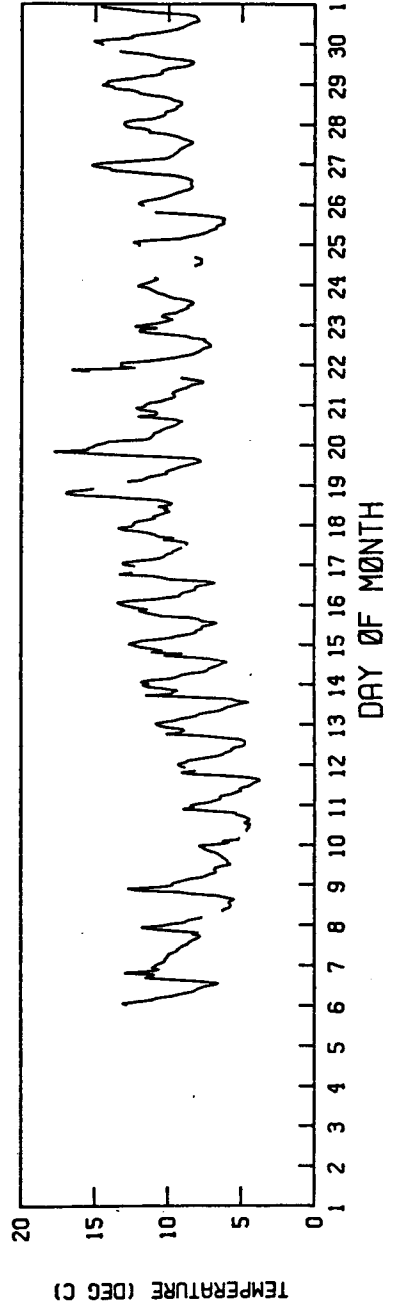
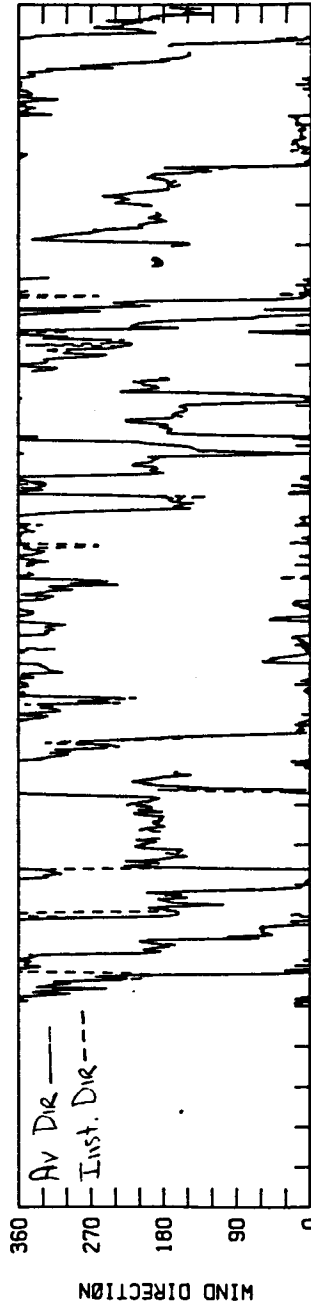
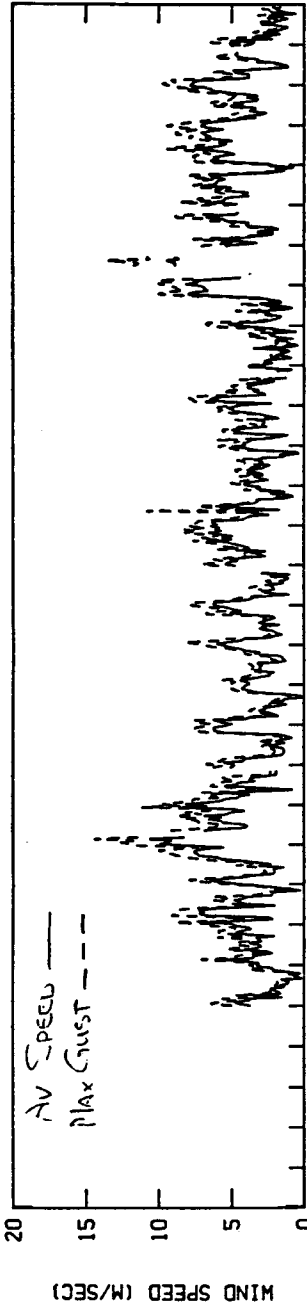
TIME SERIES PLOT FOR ALKI PØI... BUDY

1 APR 83 TO 1 MAY 83 INTERVAL= 60.0MINS



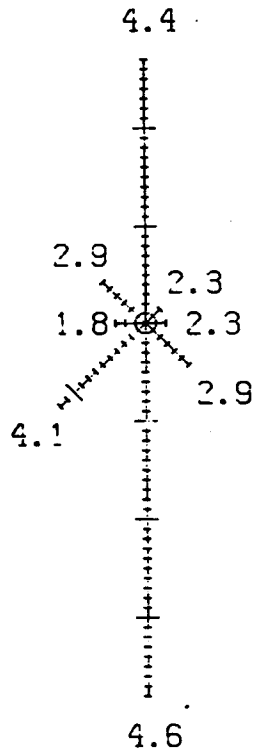
TIME SERIES PLOT FOR ALKI PØ...T BUDY

1 APR 83 TO 1 MAY 83 INTERVAL= 60.0MINS



WIND ROSE

TIME SERIES PLOT FOR ALKI POINT BUOY
1 JUL 83 TO 1 AUG 83 INTERVAL = 60.0MINS



NPTS = 629

CALM = 0.0

BAD = 93