AD-A093 649 PENNSYLVANIA STATE UNIV UNIVERSITY PARK IONOSPHERE R--ETC F/6 12/1
IMPROVED DATA REDUCTION AND ANALYSIS SYSTEM FOR BLUNT PROBE AND--ETC(U)
OCT 80 D C SCHRODER DAAD29-76-D-0100
UNCLASSIFIED ERADCOM/ASL-CR-80-0100-3 NL

1 OF 2
AD A
093649

AD A093649

ASL-CR-80-0100-3

11

AD

Reports Control Symbol
OSD 1366

LEVEL

IMPROVED DATA REDUCTION AND ANALYSIS SYSTEM FOR BLUNT PROBE AND GERDIEN CONDENSER MEASUREMENTS

OCTOBER 1980

JAN 7 1981

Prepared By
DARRYL C. SCHRODER
Ionosphere Research Laboratory
The Pennsylvania State University
University Park, Pennsylvania 16802

UNDER CONTRACT DAA–76-D-0100
This work was funded under DNA Task
S99QAxAD411 - Reaction Rates to Propagation
Work Unit 23 Variability of D-Region Ion Density and Conductivity
Contract Monitor: ROBERT O. OLSEN

DDC FILE COPY

US Army Electronics Research and Development Command
ATMOSPHERIC SCIENCES LABORATORY
White Sands Missile Range, NM 88002

81 1 08 062

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 18 ERADCOM /ASL-CR-80-0100-3 17	2. GOVT ACCESSION NO. AD A093649	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* IMPROVED DATA REDUCTION AND ANALYSIS SYSTEM FOR BLUNT PROBE AND GERDIEN CONDENSER MEASUREMENTS. 6		5. TYPE OF REPORT & PERIOD COVERED 9 [redacted] Final Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Darryl C./Schroder 10		8. CONTRACT OR GRANT NUMBER(s) 15 DAAD29 76-D-0100 ✓ ~~DAA-76-D-0100~~
9. PERFORMING ORGANIZATION NAME AND ADDRESS Ionosphere Research Laboratory ✓ The Pennsylvania State University University Park, Pennsylvania 16802		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS ~~DA Task 1L161102B5A/SA2~~
11. CONTROLLING OFFICE NAME AND ADDRESS US Army Electronics Research and Development Command Adelphi, MD 20783		12. REPORT DATE 11 Oct[redacted] [redacted]80
		13. NUMBER OF PAGES 12 121
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)* Atmospheric Sciences Laboratory White Sands Missile Range, New Mexico 88002		15. SECURITY CLASS. *(of this report)* UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

Contract Monitor: Robert O. Olsen

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Electrical conductivity	Gerdien condenser
Electrical Structure	Stratosphere
Mesosphere	Blunt probe

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

Measurements of electrical conductivity over the altitude range of 20 km to 80 km have been obtained by the deployment of a blunt probe or Gerdien condenser sensor from a small meteorological rocket. The sensors are deployed from the rocket vehicle at apogee and descend subsonically from altitudes as great as 80 km down to 20 km. This report discusses the development of a data reduction technique for the analysis of atmospheric electrical conductivity utilizing a computer software program and hardware components. A comparison test of the

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE 1

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

20. ABSTRACT (cont)

improved computer reduction technique with prior manual/computer reduction method indicates the improved reduction technique results in data with greater accuracy and increases the data yield to higher altitudes.

CONTENTS

INTRODUCTION

The purpose of this work was to explore the overall process of reducing electrical conductivity data using computer techniques and to make improvements or modifications necessary which would result in an improved data handling system producing both greater accuracy of the data resulting from a particular flight and making feasible much more complex analysis of data from many flights.

The basic experiment involves a rocket-borne instrument which is launched to altitudes typically in the range of 60 to 80 kilometers and then returns to earth using a stabilized parachute. During the descent of the instrument, conductivity measurements are made periodically and the resulting information is telemetered back to earth where it is monitored and recorded on magnetic tape. After appropriate reduction the final result of the experiment is a plot of altitude vs conductivity for the various species of positive and negative charge carriers.

The instrumentation payload essentially applies a linear voltage sweep to a pair of electrodes either in what is called the blunt probe or Gerdien condenser configuration (see figures 1 and 2). A sensitive electrometer measures the collected charged particle current and a current to frequency convertor produces a pulse train whose frequency is proportional to the collected current.

A linearly swept voltage waveform occurring typically every 6 to 8 seconds results in conductivity measurements of both

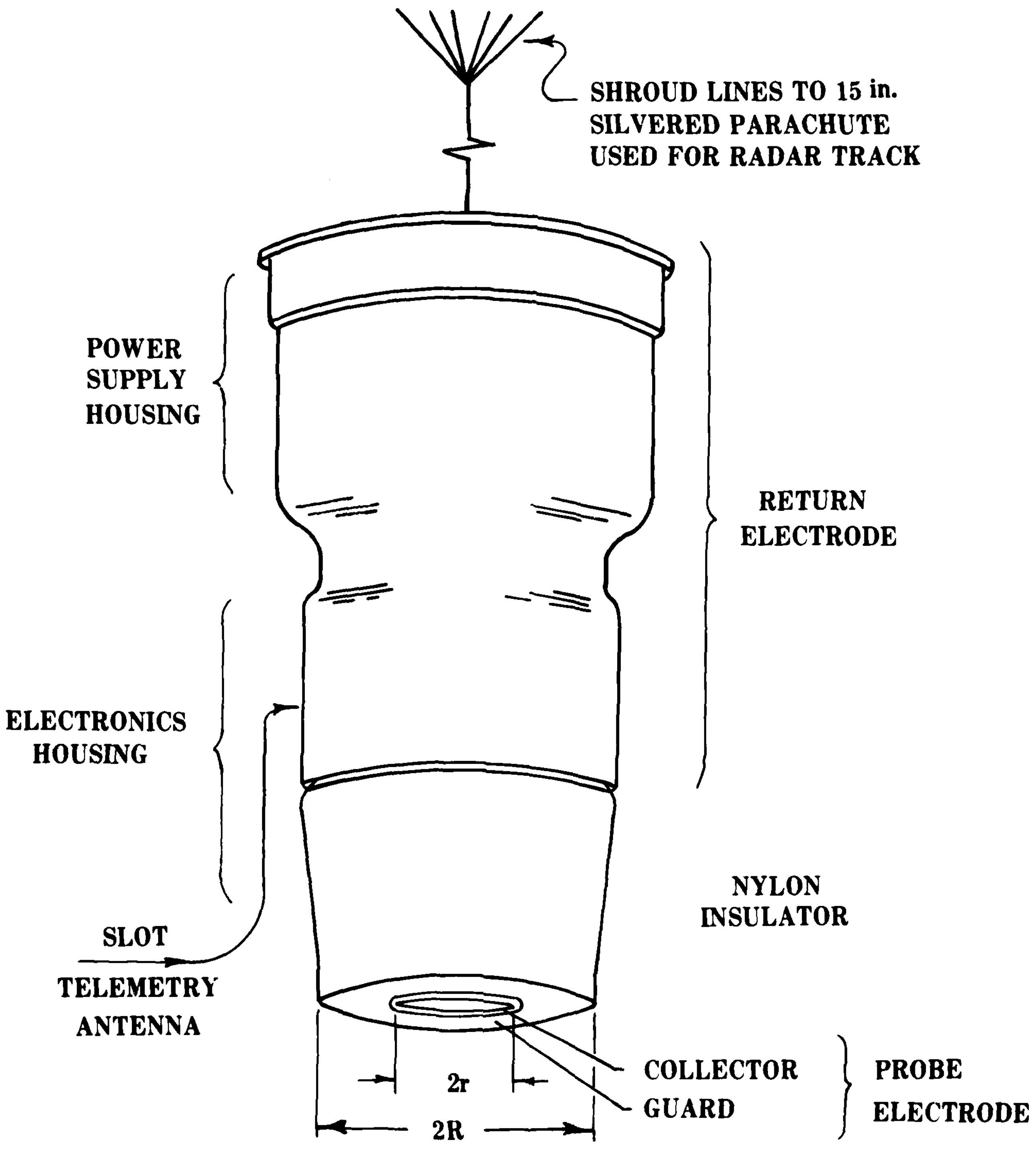


Figure 1. Blunt probe.

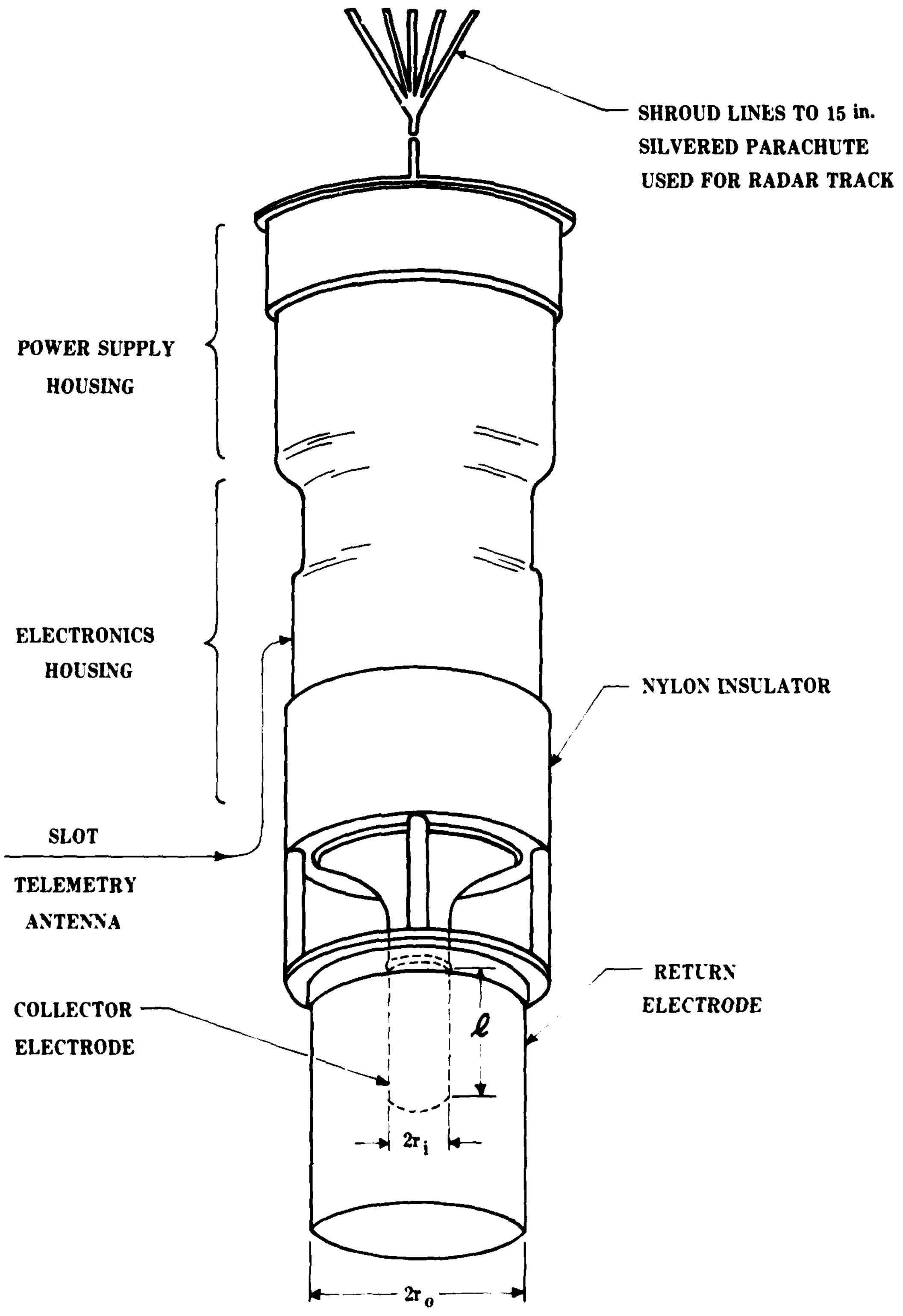


Figure 2. Gerdien condenser.

positive and negative species. The telemetered data pulse frequency is nominally in the range of 0 to 200hz thus making it compatible with the GMD telemetry system and the TMQ-5 plotter presently used by the Meteorological Rocket Network. The data resulting from a single voltage sweep will be referred to as a data waveform in the remainder of this report.

An idealized voltage waveform is shown in figure 3. At a time corresponding to the left side of the data waveform the sweep voltage drops abruptly from approximately +5 to -5 volts and then the voltage starts increasing linearly with time. Since a negative voltage is being applied to the collector electrode, positive charge carriers are collected and the pulse train frequency starts at a low value and begins increasing. As the sweep voltage increases the current due to the collection of positive charged particles also increases which results in an increasing frequency of the pulse train. This is the gradually rising portion of the data waveform to the left of the 4-second mark. The slope of the linear portion of the data waveform which occurs between, for this example, about 1 second and 4 seconds is proportional to the electrical conductivity due to positively charged particles.

At a time slightly greater than 4 seconds the voltage sweep passes through zero and then a positive voltage is applied to the collector electrode. This results in the collection of negative charge carriers and as for positive charge carriers the negative conductivity is determined by multiplying a constant by the slope of the linear portion of the curve occurring around the 5-second time mark. As the voltage sweep continues to increase, eventually all negative charge carriers are collected resulting in a constant charged particle current and a constant frequency of the pulse train at approximately 200hz.

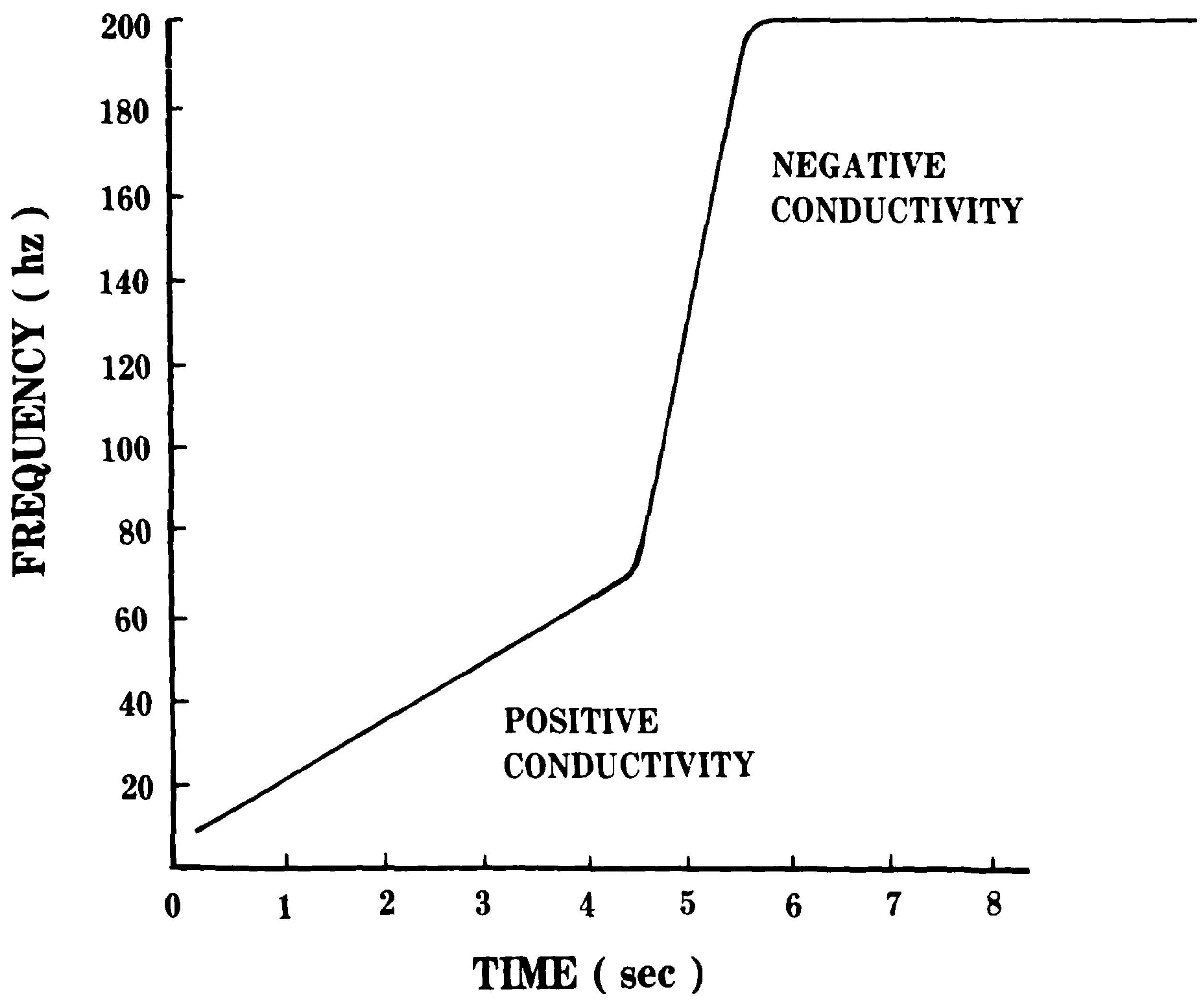


Figure 3. Typical waveform.

Current Data Reduction Techniques

The data waveforms resulting from a particular launch were historically obtained by playing back the recorded pulse trains through a data tachometer which converts the frequency of the incoming pulses to a proportional analog voltage. This voltage is then applied to a stripchart recorder which draws the data waveforms on chart paper while simultaneously recording timing ticks on the edge of the strip chart. Typical data waveforms resulting from the above process are shown in figure 4.

A straight edge is manually placed on the stripchart recording and a line is drawn through the portion of the data waveform whose slope is desired. The exact portion of the data waveform used and the straight line approximation to the actual data points are subject to the judgment of the person reducing the data. The slopes of the straight line segments are then determined and the time of occurrence of the waveform relative to launch is also found.

Prior to launch the instrument is calibrated by placing a known and large valued resistance between the instrument electrodes. Waveforms typical of those resulting from this procedure are shown in figure 5. The slope of these calibration waveforms in conjunction with other probe parameters is used to calculate a constant which is valid for a particular instrument and flight.(1) When the slopes of the data waveforms are multiplied by this constant the electrical conductivities result. Radar data which gives instrument height as a function of time is used

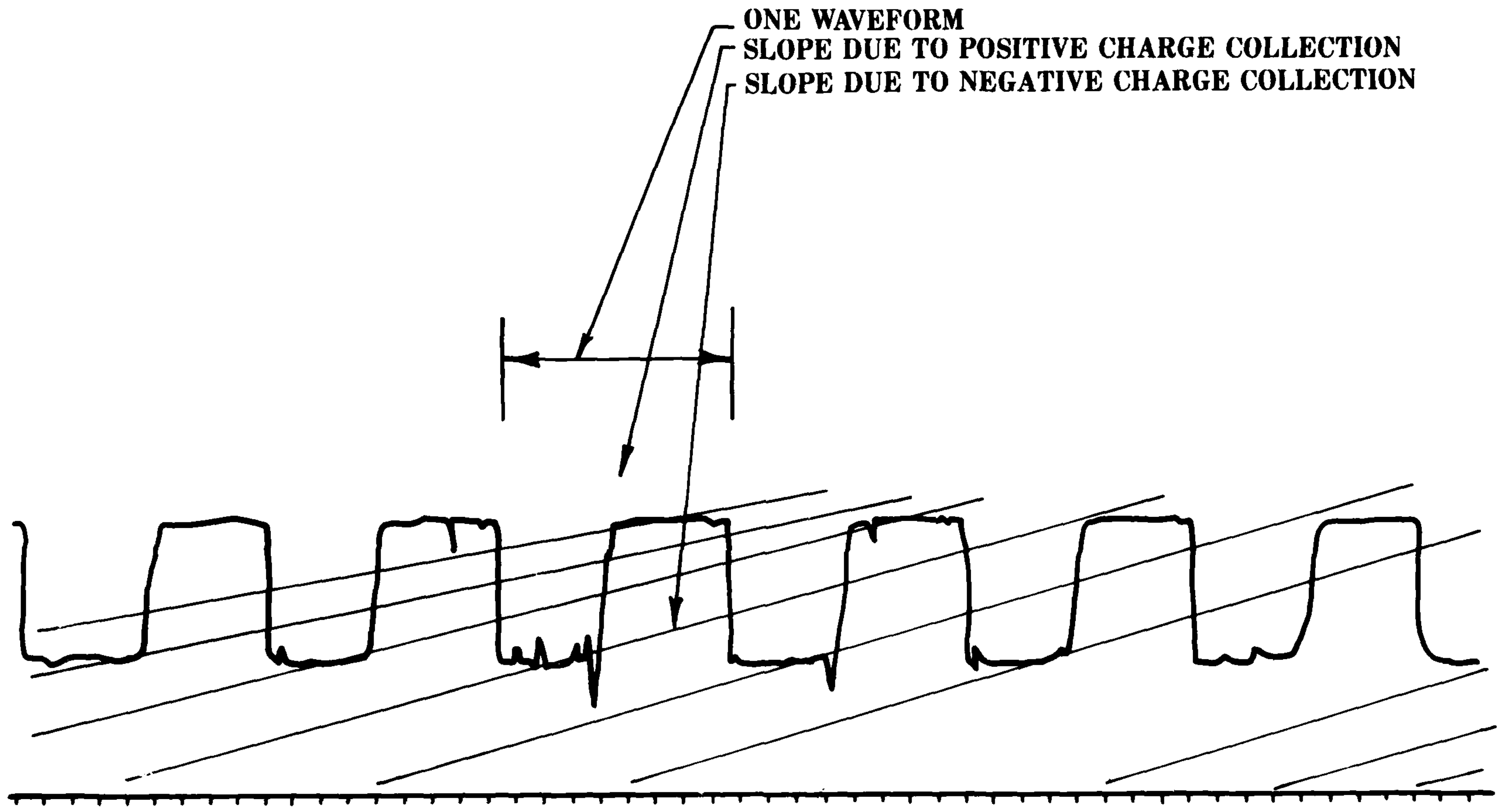


Figure 4. Typical blunt probe data waveform as displayed on a stripchart recorder.

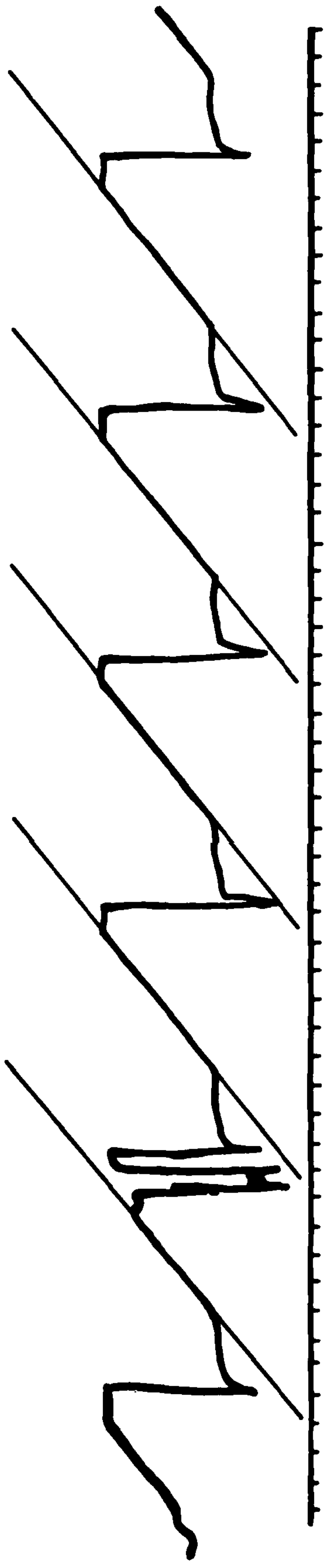


Figure 5. Calibration waveforms.

to determine the height at which a particular data waveform occured. The final result is a plot of the height vs the electrical conductivity as shown in figure 6.

For data waveforms which are relatively free of noise the data reduction process is straight forward although time consuming. A portion of a flight where substantial amounts of noise was present is shown in figure 7. Here the skill of the person reducing the data is extremely important, since it is not at all obvious how straight line fits should be drawn through the data points. In this case the inertia of the stripchart recorder pin plus the wide scatter of the data points makes the straight line approximation difficult. For this portion of the flight the positive ion conductivities were not determined.

Limitations of Present System

The problems associated with the present system are perhaps obvious but for completeness will be enumerated. Most importantly the quality of the data reduction process depends on the ability of the person reducing the data to choose an appropriate portion of waveform and fit a straight line through it. Second the analog data tachometer does not have the ability to discriminate between extraneous ncise pulses and data waveform pulses. Third the data waveforms are distorted because of the response characteristics of the stripchart recorder. Fourth the time which is recorded for each data waveform is dependent on the speed characteristics of the tape recorder and the internal clock contained within the stripchart recorder which supplies the stripchart

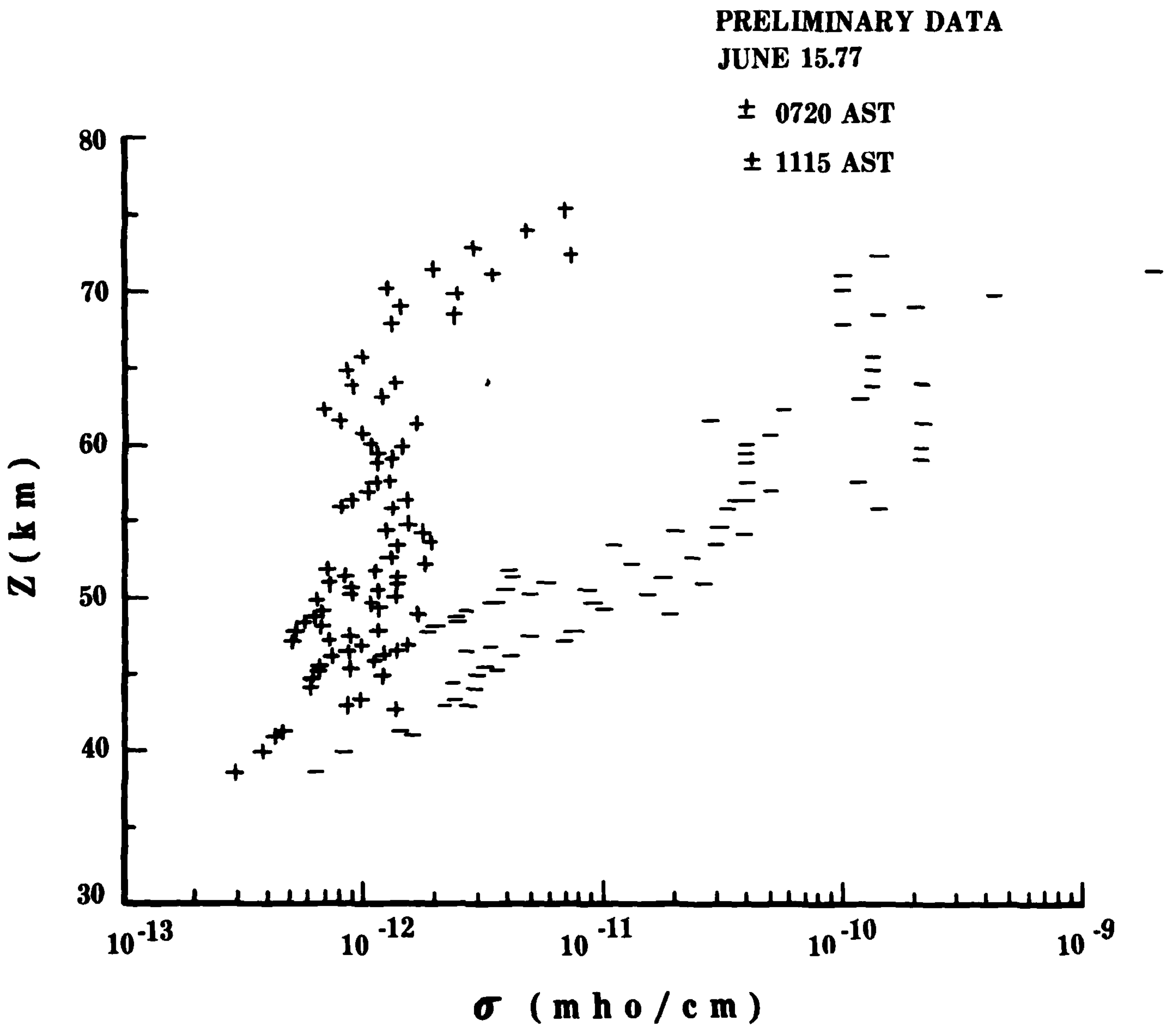


Figure 6. Typical results of an experiment.

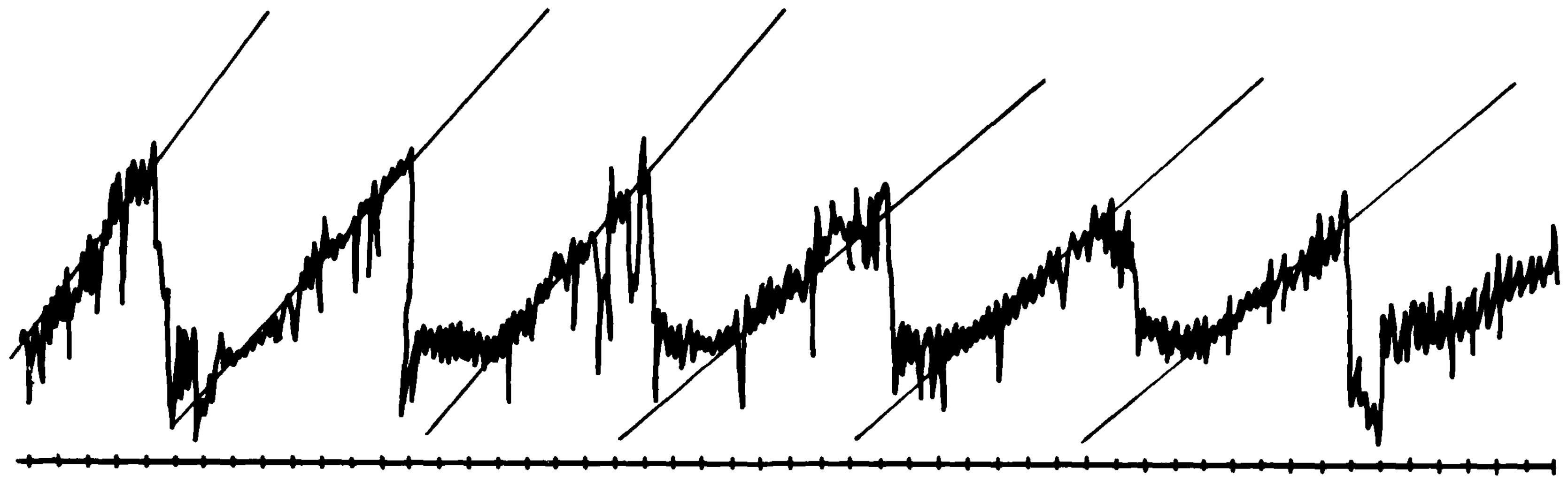


Figure 7. Data waveform with large amounts of noise.

timing ticks. Fifth a variety of problems exist in storage, retrieval and cataloging of the reduced data because of the volume which results. Finally the process is relatively slow.

Analysis of a Computer-Based Reduction System

It is appropriate to explore what tasks can and cannot be easily accomplished by computer in the overall data reduction process, and how particular tasks might logically be implemented.

Since the data is played back in real time any envisioned computer system must be able to process the data at a sufficient rate to meet this requirement. The task can however be partitioned into a number of passes through the computer system so that all work required to produce a plot such as that in figure 6 does not have to be accomplished at once.

A logical approach might be to implement a first pass which would perform only the minimum processing necessary to convert the incoming pulse trains into digital form and record the time of occurrence relative to launch for each data waveform. This would minimize the time constraints placed on the computer system and if sufficient time were available some digital filtering might be used to remove some of the noise contained in the data waveform.

The digitization of the data waveform might be accomplished as follows. It is necessary to determine the arrival of each pulse in the pulse train and also to accurately determine the elapsed time between pulses. This may be done using a Schmitt Trigger and a device called a real-time clock which consists of an oscillator driving a binary counter.

The Schmitt Trigger is a device which produces a pulsed output only when a pulse of sufficient amplitude arrives at the input. The level required to produce the output is adjustable so that by careful setting, low amplitude noise pulses are removed from the incoming pulse trains. Detection of the arrival of a pulse is accomplished normally by a computer interrupt generated at the output of the Schmitt Trigger. The computer then enables the real-time clock which starts to count at a frequency determined by the oscillator feeding the counter. The arrival of a second pulse at the Schmitt Trigger output again generates an interrupt and the computer then reads the accumulated count in the counter, stores this away, resets the counter to zero and restarts it in preparation for repeating the process.

As an example suppose that the oscillator driving the counter is set to a frequency of 100,000 hz and two pulses in the pulse train are separated by 0.01388 second. At the arrival of the first pulse the counter starts counting at a rate of 100,000 counts per second and since the second pulse stops the counting 0.01388 second later the counter will have recorded 100,000 counts/second x 0.01388 second = 1388 counts. This number in binary form would be stored as one data point in the waveform. Later we can convert back to the time between pulses by simply dividing 1388 by 100,000 which equals 0.01388 second. Finally, since frequency equals 1 divided by the time, the frequency of the pulse train at this point in time was 1/0.01388 = 72 hz.

Thus the conversion of the incoming pulse strings to a form which can be stored in a digital computer is relatively straight forward. A problem arises though because the pulse trains come in continuously and later in the reduction process it will be desirable to look at only one data waveform at a time. Thus a decision must be made concerning how to divide the digitized data into manageable segments.

An obvious place to divide the data waveforms would correspond to the left side of figure 3 where the frequency rapidly drops from about 200 hz to a low value. Unfortunately, dropout which is a temporary loss of signal from the instrument package produces an almost identical drop in frequency. The problem reduces to one of pattern recognition where a certain key characteristic is detected amid the incoming data stream. Computers do of course have this capability, but a human is also extremely good at pattern recognition and the problem of data waveform segmentation can be very easily accomplished by an operator simply signaling the computer when a new waveform has started. A good approach might be to play the pulse trains back through a speaker. The computer operator quickly establishes a sense of when the new data waveform is expected because the waveforms are received at fairly regular intervals. The sudden drop in audio tone would then alert the operator to signal that a new waveform is arriving. Because of the nature of the data waveform a time period approaching 2 seconds would be available for the operator to react, and this appears to be more than adequate.

The most critical portion of the reduction process from an implementation standpoint is the initial digitization and segmentation of the data waveforms. The reason for this, as mentioned previously, is the need to process the data in what corresponds to real time. Considerable effort is required to define exactly what must be done and in what sequence while making sure that the tasks can be accomplished with a particular computer system. Because of the speed requirements the computer programs which do the above would normally be written in assembly language and would be dedicated to a particular computer.

Once the data has been digitized and stored in a usable form the next task to be accomplished is determination of the two slopes of each data waveform. A wide variety of options are available involving not only selection of a particular algorithm for curve fitting assuming software is used, but also to what extent operator interaction should be incorporated in the process. The problem of pattern recognition is again encountered.

Essentially three tasks must be accomplished. First a decision must be made as to which portion of the data waveform is to be used. Second a straight line must be fitted through the selected points and finally the slope of this line must be computed and stored for future use.

Pattern recognition requires extensive programing and is time consuming for computers so this task might well be assigned to the human operator. Curve fitting requires many repetitive calculations and is best done by computer, and finally the computer should be used to calculate the line segment slope and store it.

A final aspect of the data reduction process is plotting the results such as those in figure 6. Since plotting routines and computer driven plotters are widely available this task should be accomplished by computer.

The way the data is stored in the computer has a great deal to do with the amount of effort required for the initial programing and how useful the system is once it is implemented. Since the data reduction task would normally be accomplished in several passes through the computer, the data structure could be changed after every pass or could maintain essentially the same form. If the data structure maintains the same form, initially much storage space is wasted since information is added to the data set as a result of each pass. If the data structure is allowed to vary, then the programing becomes more complex since each input and output operation incorporates a unique data configuration. Since the ultimate destination for the reduced data would normally be storage in a magnetic tape data library and magnetic tape storage is relatively cheap, the constant data structure approach is best for this class of problem.

The final result of the reduction process should be a series of data records stored of magnetic tape each of which contains all information known about a particular waveform. After initial digitization, the data structure would be fixed and any further passes would simply place additional information into previously storage locations. Upon retrieval from the library a particular record would be complete and independent, making additional analysis and/or comparison programs much simpler to implement.

Detailed Discussion of the Data Reduction Process and Design of the Data Management System

To quickly reduce the data resulting from a particular flight, make comparisons between various flights; to facilitate cataloging of all the available data, a computer-based information system is desirable. A typical flight results in approximately 200 waveforms each composed of several hundred data points. Currently, data from approximately 40 flights is available or awaiting reduction. Hand processing of this amount of data is an extremely laborious and expensive process.

The addition of a magnetic tape drive to existing computer facilities in the spring of 1979 provided the additional resource required to develop the needed information handling system. Prior to this time a computer program had been used to successfully reduce data from single flights, but because of limited mass storage capability, the information from only one single flight could be stored and processed at any one time. The problem of data reduction had been addressed but the overall process of data reduction, analysis and storage required substantial modifications. The way this early program stored the data was somewhat clumsy and wasteful but more importantly it was not structured in such a way that complete information for more than one flight was available. Thus comparison between various flights could not be accomplished directly.

The early programs were written in such a way that as the pulse trains entered the computer there was almost no free time to analyze what was being received. The program simply converted the time between the arrival of each pulse in the train to a digital number and dumped everything onto the disk. The reduction of one flight resulted in almost a full disk of data. By incorporating substantial modifications at this stage sufficient time could be obtained to apply some simple tests to the data. The portions of the waveform which contain the useful results are those where the frequency is between approximately 10 hz and 200 hz. Yet for substantial portions of the 6-8 second waveform the frequency is either below 10 hz or above 200 hz.

To illustrate what happens suppose that typically 2.5 seconds of a waveform consist of a constant pulse train of 200 hz. Processing of this portion would result in 500 numbers being stored on the disk which contain no useful information at all. They simply require processing time and storage space. A typical data waveform might consist of about 1000 points. So removal of the nonessential information produces significant savings in storage.

A major problem is then how to efficiently convert the information in the pulse train to digital form and store only what is useful. Another requirement is knowledge of the exact time of day when each pulse train is generated. This allows radar data to be used to determine the instrument altitude, and it turns out that the above two are related.

There are at least three fundamental approaches to obtaining correct timing information for each conductivity measurement which has been made. The first of these involves using a time decoder to reconstruct the time of occurrence of each waveform. The time code information is stored on one of two information channels recorded on the magnetic tape. This approach was not used because it required either a rather expensive piece of hardware or software decoding within the minicomputer which timing constraints did not permit.

The second approach was to use a device associated with the minicomputer called the laboratory peripheral system (LPS). The LPS contains a software programmable counter driven by a frequency selectable oscillator. This system operates as follows. An arbitrary but known point such as launch or payload separation is used as a reference and the computer keeps time from this point. When each data pulse is received by the computer the LPS counter is reset to zero by software and then the counter counts upward at a rate determined by the oscillator. Thus the computer keeps track of oscillator cycles between data pulses. Since the oscillator frequency is known the time between the occurrence of each pulse can be computed, and further computations produce the varying frequency of the pulse train. Since the time between reception of each pulse is known, the time of occurrence of any waveform within the total flight can be computed by simply adding all the times between each of the pulses and finally adding the reference launch time. This approach was used in a previous data reduction scheme and has several disadvantages.

The first results when dropouts or interruptions occur in the data pulse trains. In order to have good resolution in the measurement of the time interval required for the next data pulse to arrive the internal oscillator must run at 100,000 hz. Since the LPS counter is 16 bits it can count only to 65,535 counts so the counter will overflow after 0.65 second. Further this overflow does not generate an interrupt. The counter simply resets to zero and starts counting again. A data dropout can only be detected by periodically polling the counter output buffer and testing the results. This polling procedure must be inserted in the software which proved to be difficult and still have the computer receive data at a speed equivalent to real time. More importantly however upon resumption of data reception after a dropout the clock must be reset so that normal timing operations can be resumed. This resetting procedure and related testing requires a number of instructions to be executed while the clock is stopped and then a further correction must be made to the accumulated time. Otherwise all future waveforms will be incorrect as far as time and consequently altitude are concerned.

A third approach, and the one used for the present system, was to interface a time code generator with parallel BCD output to the digital I/O port of the LPS unit. This method has several distinct advantages over the others described previously. The time code generator can be preset with either the time of launch or payload separation, and once started the time code generator always has the correct time relative to data input available. Consequently corrections requiring additional computer time are unnecessary. This frees computer time for more sophisticated processing of the data as it enters the computer, such as digital filtering and suppression of nonessential data points. The drop-out problem is also solved since upon detection of resumption of the signal the time code generator can be immediately polled for correct timing information. Testing of the counter output buffer can also be eliminated since it is no longer necessary to know if the counter overflowed for correct timing information. Upon resumption of data transmission after a dropout the possibility exists for inclusion of one bad data point. This point would occur since the computer program reads the counter upon arrival of each data pulse and stores this as data without reinitialization of the clock. This potential source of error is tolerated since it greatly simplifies the required software and in a statistical sense will introduce very little error.

This fact can be seen by observing the following facts. Since the counter has run for an arbitrary period during dropout it will contain an arbitray count corresponding to frequencies varying between above 100,000 hz and 1/.65 hz. Thus the probability of getting a point within the frequency range of interest is small. Secondly even if by chance a point falls within the 10 - 200 hz range its effect will be minimal since the df/dt calculation for a particular conductivity is based on a substantial number of data points (typically 30 to 100). Thus one or even several bad data points due to dropout in a waveform will have a very minimal effect.

Ignoring the dropout problem by tolerating the possibility of minor errors and the use of an external clock reduces the complexity of the programs and drastically reduces the amount of mass storage required in the computer system.

Although not implemented in the present information system several possible modifications or extensions were carefully considered and whenever possible the present system was configured in such a way as not to preclude their future implementation.

The present system is based on the concept of data reduction being accomplished after the experiment is completed. There are however good reasons for having the data processed in such a way that it is available in final form in real time. The present system has been designed in such a way that an inexpensive microcomputer with dedicated I/O devices could replace the data reduction steps presently performed by the PDP 11/10 computer. This preprocessor in conjunction with a second microcomputer doing the curve fitting and driving a plotter would produce an inexpensive portable system which would operate in real time, but is not fundamentally different in concept from the present system. With the previous constraints and requirements in mind a computer based data management system was assembled and needed software was developed.

System Hardware

Figure 8 is a block diagram of the hardware configuration used for the data reduction and storage system. Pictures of the various devices are shown in figures 9 and 10. The computer used is a DEC PDP 11/10 minicomputer. Data is played back using a Hewlett Packard 3960 tape recorder. The tape recorder has a built-in amplifier which boosts the recorded pulses up to the 0-2 volt range which is satisfactory for the LPS Schmitt Trigger. The tape recorder also has a speaker which allows monitoring of the audio tone produced by the pulse train.

The pulse trains from the recorder are connected to Schmitt Trigger 2 of the LPS Unit. An adjustment knob is available on the front of the LPS Unit and is used to select the level of input required to fire the Schmitt Trigger. This level is adjusted before pass one of the data processing begins. An oscilloscope is connected to the LPS output jack labeled Schmitt Trigger 2 output. To observe the correct adjustment of the firing level of the Schmitt Trigger considerable care should be exercised in the adjustment since careful setting eliminates most of the extraneous pulses in noisy flights.

The analog to digital section of the LPS unit is used to enter operator commands since this is the most convenient available device to use with the assembly language programs. Channel 0 detects a 5-v to 0-v transition caused by the operator pushing a button which signals the start of a new data waveform. Channel 1 is reserved for the input of a specially processed analog reconstruction of the data waveform which may be implemented at a

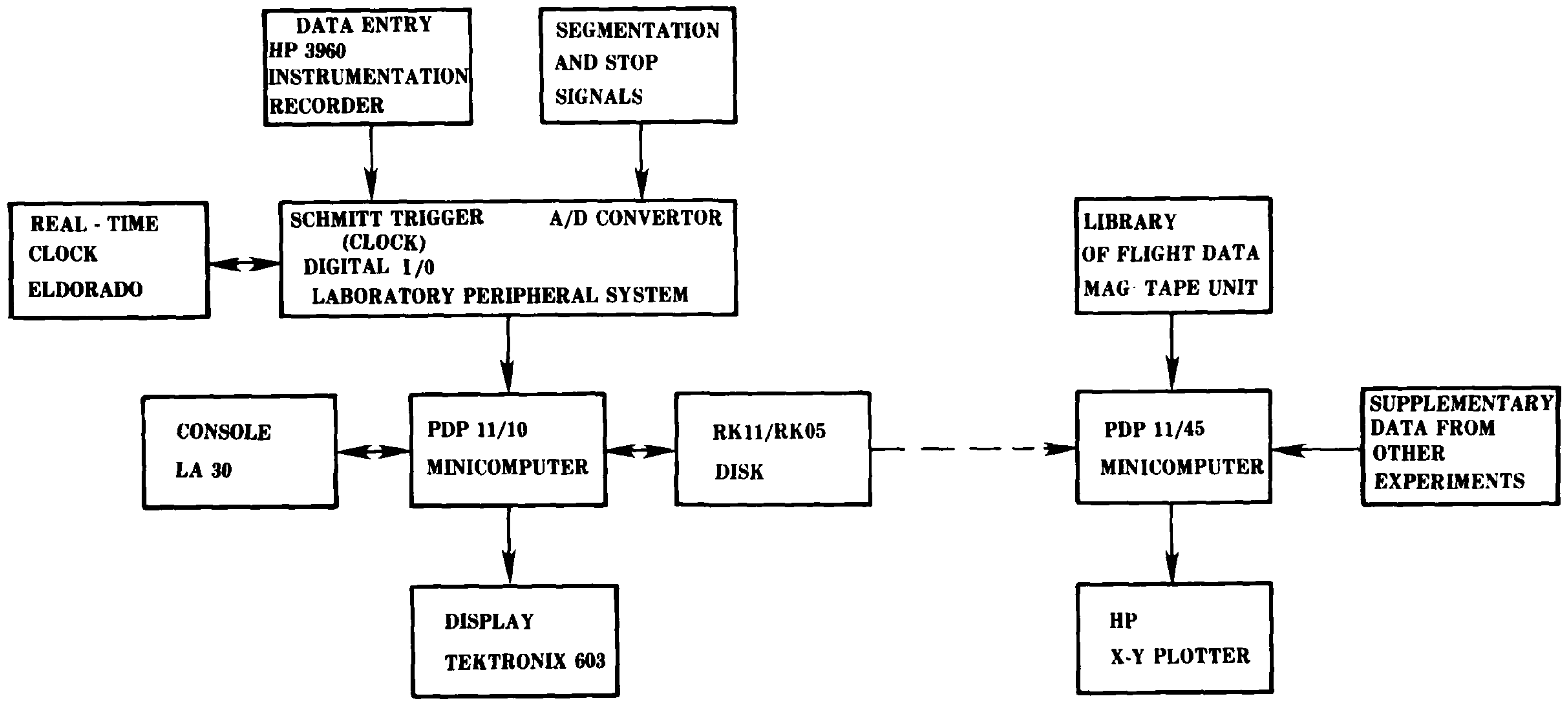


Figure 8. Hardware configuration of data reduction and storage system.

Figure 9. PDP 11/10 Minicomputer and Related Hardware.

Figure 10. PDP 11/45 Minicomputer.

future time. A 5-v to 0-v transition on channel 2 signals the end of processing for a flight.

The Eldorado 1710 real-time clock provides a nixe-tube display of the time. This device is preset to the time of launch using its own front panel controls and it is enabled simultaneously with the start of the flight recording on the tape recorder.

The 1710 provides parallel BCD outputs on the back of its chassis. It was however not directly compatible with the 16-bit digital I/O port of the LPS Unit so a special hardware interface was constructed to accomplish the task. Details and schematics of this device are included in appendix I. Briefly, the interface works as follows. Under program control the LPS Unit sends out a command to the interface initiating the transfer of a parallel 8-bit word containing the 4-bit BCD code for tens of seconds and unit seconds. Appropriate drivers are enabled and the computer reads in and stores the seconds. The process is repeated for minutes and hours and is accomplished extremely rapidly since communication is parallel and very few computer instructions are required. The 8-bit format was chosen since the hardware implementation was simplified and this form would be compatible with almost any computer system including a microcomputer.

As data enters during pass one a line is drawn on the Tektronix 603 storage scope which is also driven by the LPS Unit. This provides a visual indication that data is entering properly. The scope display is cleared at the beginning of each data waveform. This same display is used in pass three to display the data waveform. The presentation is a form similar to figure 3. During this pass the LED display of the LPS Unit is used to display the time into the flight of the waveform being displayed. The time displayed is in seconds and provides a unique identification of each waveform.

The LA 30 is the system console and is used to execute various programs as well as to instruct the system concerning which segments of the data waveform to use in computing the slopes.

Once a flight has been reduced the resulting data set is stored on a removable disk. This disk is transfered to a PDP 11/45 minicomputer where a magnetic tape drive and xy plotter are available for data cataloging and display.

The 11/45 computer is also equipped with time-sharing terminals that permit indirect entry of a previously reduced flight into the magnetic tape library. It should be mentioned that the magnetic tape unit is treated as a random access device by the various programs which greatly simplifies and speeds up search operations. The file structure is also fully compatible with the RT-11 operating system utility programs. Thus data storage can be accomplished on disk, Dec tape or mag tape as desired.

Overview of System Software

The computer programs for the PDP 11/10 minicomputer which is used for the initial data reduction are based on the RT-11 operating system and are written in Assembly language and Fortran. The programs used for the PDP 11/45 minicomputer are written in RT-11 Fortran and Unix Basic.

The data reduction process itself is based on a multipass philosophy as shown in the block diagram of figure 11. PASS1 digitizes the data, reads the external real-time clock, segments the data so that a waveform begins at the beginning of a block and insures that a waveform occupies an integral number of blocks. PASS2 calls a Fortran subroutine which requests from the system operator all necessary information concerning the instrument and launch and inserts this in the appropriate positions in each waveform record. This pass also decodes the time and performs certain other housekeeping functions.

PASS3 accomplishes the curve fitting tasks. It requests from the operator the beginning and end points of the waveform segment required for the straight line fit and slope estimation. The algorithm used is a weighted least square fit. A straight line is initially fit through the waveform segment selected by the operator. Based on this initial estimate weights are assigned to the points with more significance attached to the points closer to the first straight line approximation. A least square fit is again computed this time using the weighted values. This process is repeated a third time resulting in a good linear approximation. PASS3 now determines the slope of the line and

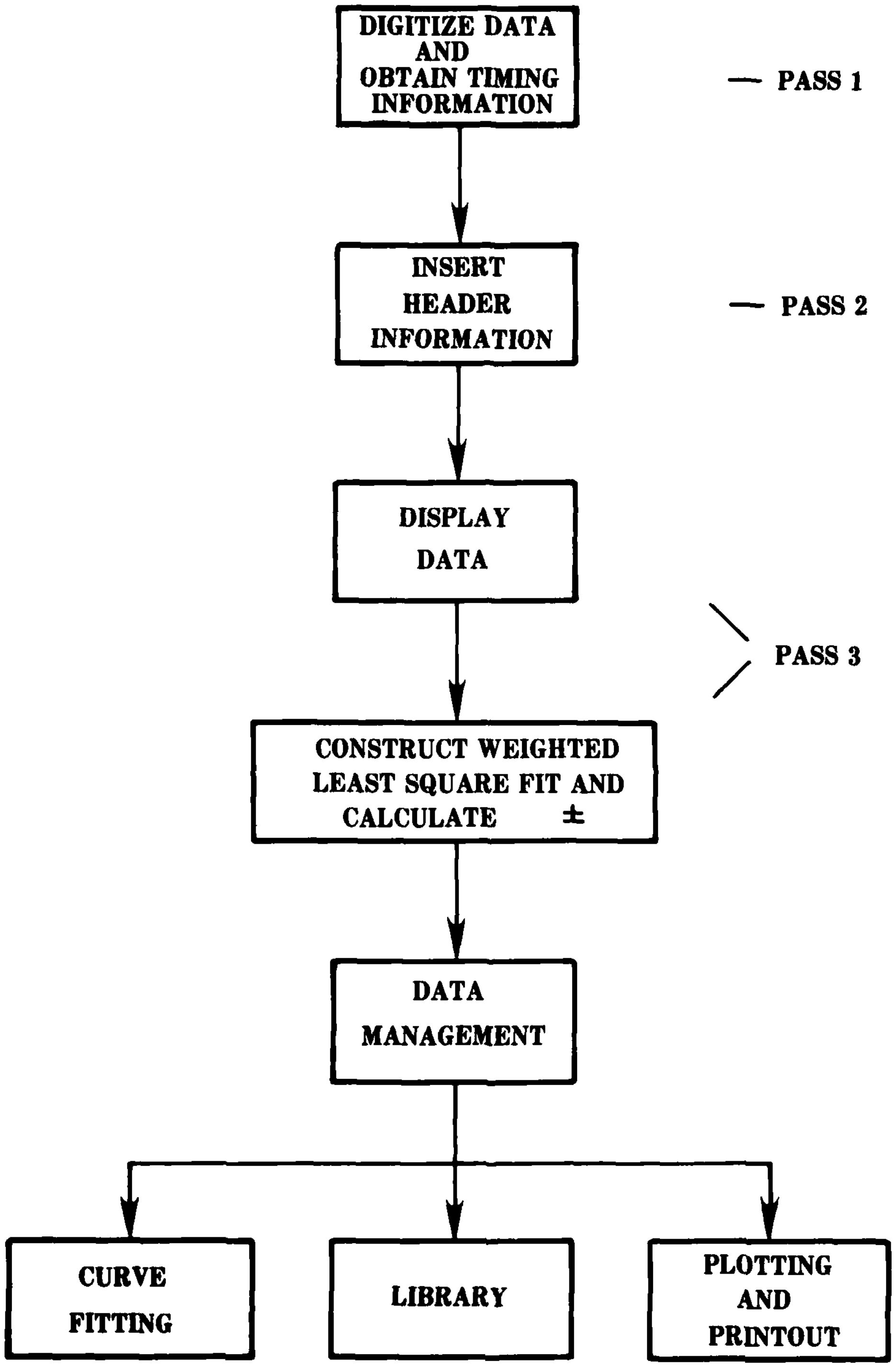

DIGITIZE DATA AND OBTAIN TIMING INFORMATION
— PASS 1
INSERT HEADER INFORMATION
— PASS 2
DISPLAY DATA
PASS 3
CONSTRUCT WEIGHTED LEAST SQUARE FIT AND CALCULATE ±
DATA MANAGEMENT
CURVE FITTING
LIBRARY
PLOTTING AND PRINTOUT

Figure 11. System Software configuration.

inserts the final computed values of the conductivities into the appropriate locations within the data record.

A number of utility programs are available on the PDP 11/45 computer for further data manipulation.

Operating system utilities are of course available to transfer files from one storage media to another. Programs were specially written to simplify the data management. One program extracts header information and conductivities from a flight data set existing in the library and prints out this information. Another extracts the altitude and conductivity information and puts it out on Dec tape in a form compatible with the Unix plotting routines. This program is used to produce plots similar to the one shown in figure 6. Another program inputs altitude and conductivity data from a terminal and converts this to the standard form required for the library. Thus data from previously hand reduced flights can be used in the present system as can data originating with other experimenters. Names of the principle programs used in the system and a brief description of their function are included in an appendix. The programs also appear in the appendix.

Computer Reduction of a Flight

In order to verify the correct operation of the computer data management system, a flight which had previously been reduced by hand was reduced by computer. The time required is of some interest. PASS1 took about 10 minutes, PASS2 required less than 2 minutes and PASS3 required about 1 hour. Production of the plots shown required approximately 1 additional hour. The above times do not include the time required for equipment hookup.

The system operator was familiar with the hand reduction method but figures 11 and 12 were his first attempts at reducing a complete flight by computer. Normally after viewing the final plots, a second iteration would involve reexamination of the particular waveforms producing data points whose validity might be questioned. This was not done for this flight to demonstrate the quality of the data reduction process without any fine tuning. In the figures the hand reduced data is indicated with "+" for positive conductivity values and "-" for the negative conductivities. The computer reduced values are plotted with "0" in both figures.

For the positive conductivities there is good agreement between the points resulting from the two techniques. The computer reduced set of the negative conductivities is superior to the hand reduced set. It shows in general much less scattering of the points. This is true because the computer can compute the slope of the rapidly rising portion of the waveform much more accurately than it can be scaled by hand.

The computer system has a distinct advantage over the hand method in the way the data is displayed. Comparison of the stripchart made during PASS1 with the storage slope display shows that the stripchart recorder, because of the pen inertia distorts and indeed surpresses some of the finer characteristics of the waveforms. Using the computer system the experimenter has the ability to see a much clearer image of the data waveform and as experience is gained in using the new system there is every reason to believe that both the speed and quality of the process will experience further improvement. Plots of each individual set of data points and a listing of the point pairs is included in appendix 12.

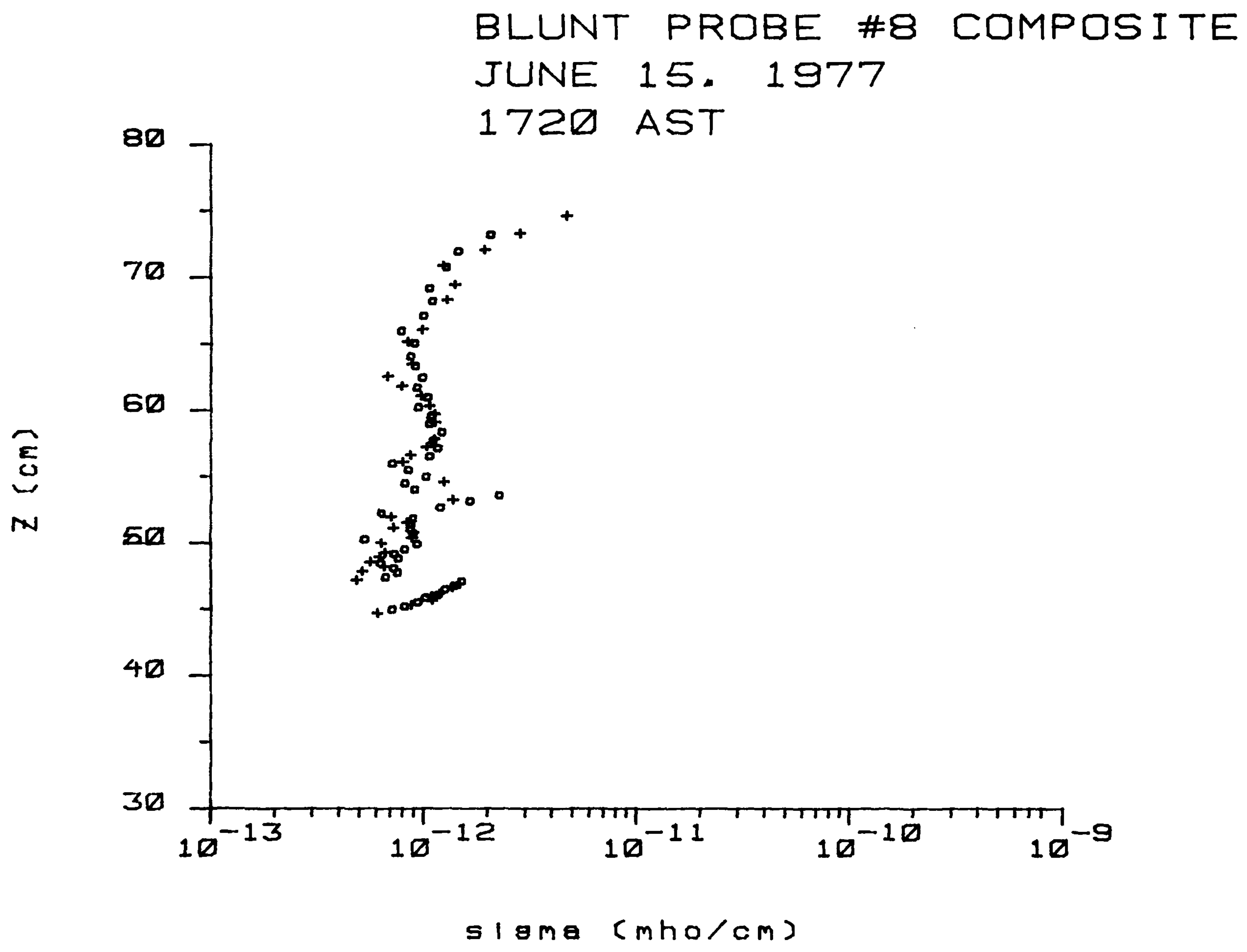


Figure 11. Comparison of Computer Reduced "0" and Hand Reduced "+" Positive Conductivities.

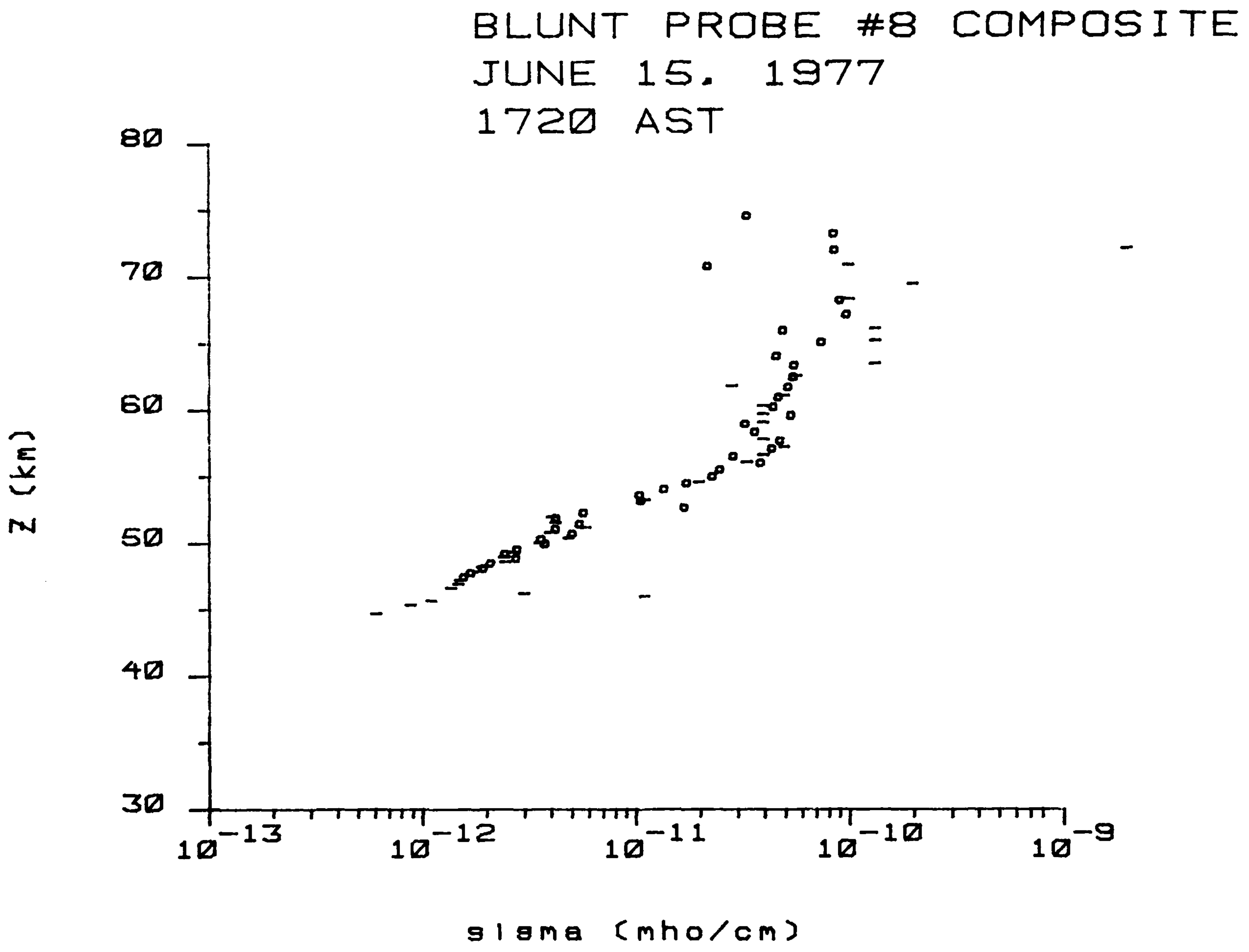


Figure 12. Compositive of Computer Reduced "0" and Hand Reduced "-" Negative Conductivities.

Validation of the Computer Reduction System

There are three possibilities for errors being introduced into the data reduction process. The first of these is from the instrument itself. Correct operation of the system is carefully checked during various phases of construction and upon completion calibration waveforms are recorded with a known prelision resistor connected between the electrodes. The second potential source of error is the telemetry system and tape recorder. Since the information is in digital form little possibility exists for changes in the period between pulses of the pulse train comprising the data. Noise may however introduce additional pulses. Overall timing is dependent upon the speed of the tape recorder and this is tied to power system frequency which for the purpose at hand is very stable.

Finally the computer system and operator are other sources of potential errors. The system itself has only one adjustment which is the firing level of Schmitt Trigger number This can be observed on an oscilloscope or on the Tektronix 603 scope. Either the signal is visible or it is not. Minor adjustments affect the noise levels but do not alter pulse timing.

The conversion from the pulse train to frequency was checked with laboratory quality instruments and the error is less than 1%. Again the system is digital and all timing is referenced to a 100 khz crystal oscillator which is very stable.

The primary source of potential error is the operator. He must make the decision as to which segment of the waveform is to be used for the slope computations. Once this is specified the least squares curve fitting is a standard numerical procedure, and of course the subsequent computations and plotting use proven software so very little error is introduced in the actual processing.

Several things might be done to reduce the possibility of errors. The first of these would be to modify the instrument in such a way that for a small portion of the sweep a resistor of known conductivity is connected across the probe electrodes. Because of the extremely high impedances involved and the desire to keep capacitance to a minimum, this might be an extremely difficult design task. A second desirable feature might be a special marker which would detect the zero voltage crossing point of the linear voltage sweep and key the transmitter to transmit a short burst of perhaps 200 hz pulses.

This would provide an easily recognized and decoded reference point in the waveform as far as the computer is concerned. It would not overlap any vital areas of the waveform and it would make the operators job much easier since he would have an absolute reference point to consider when the curve fitting is being accomplished.

The ultimate question of validity of the entire system has no simple answer. The instrument is measuring phenomena which are not well understood and cannot be duplicated in the laboratory under controlled conditions. Confidence in the system can be gained only with experience in using it. First this usage should be in parallel with the hand reduction techniques presently used.

BIBLIOGRAPHY

1) Mitchell, J. D., and Shih, S. A., A Computerized System for the Reduction of Middle Atmospheric Electrical Conductivity Data, Department of Electrical Engineering, University of Texas at El Paso, 1977.

2) Daniel, C. T., and F. S. Wood, Fitting Equations to Data, Wiley-Interscience, New York, 1971.

3) LPS11 Laboratory Peripheral System User's Guide, Digital Equipment Corporation, Maynard, Massachusetts, 1973.

4) PDP 11-04/05/10/35/40/45 Processor Handbook, Digital Equipment Corporation, Maynard, Massachusetts, 1975.

5) PDP 11 Peripherals Handbook, Digital Equipment Corporation, Maynard, Massachusetts, 1975.

6) RT-11 System Reference Manual, Digital Equipment Corporation, Maynard, Massachusetts, 1975.

7) Sagar, R. S., A Subsonic Gerdien Condenser Experiment for Upper Atmospheric Research, Master's Thesis, The University of Texas at El Paso, El Paso, Texas, 1976.

LIST OF APPENDICES

APPENDIX 1

HARDWARE INTERFACE FOR THE ELDORADO 1710 TIME CODE GENERATOR

Time Code Generator Circuit Description

A 40 pin connector cable brings parallel BCD signals from the back of the Eldorado 1710 time code generator to a wire wrapped custom made hardware interface. The time code generator outputs all signals simultaneously in parallel form, but since the LPS cannot receive over 16 bits at a time, the time code signals must be multiplexed before entry into the computer. This is accomplished by use of 4 LSI integrated circuits (8212) that operate as 8-bit latches.

Under software control an enable signal consisting of one bit is sent from the LPS digital output port to the interface. This signal is routed to one of the latches which when enabled supplies an 8-bit parallel code to the LPS digital input port. The latches are so connected that the first supplies unit and tens of seconds, the second supplies unit and tens of minutes, the third supplies unit and tens of hours, and the forth unit and tens of days.

The signals from the interface are routed through a dip 24 pin header to two 25 pin connectors which plug into the back of the LPS unit.

The LPS unit was designed in such a way that 0 volts are interpreted as a logic "1" and 5 volts is a logic "0" so the software must complement signals as necessary to take this fact into account.

Also included on this board is a simple circuit to convert a signal from the operator push button used to segment the waveforms into one and only one negative going pulse of 50 msec duration. The interface is assembled in a Honeywell Microblock inclosure and draws power from this source.

APPENDIX 2

DATA STORAGE STRUCTURE

The storage space allocation for a single data waveform consists of two parts. The first is a header section which contains all information concerning the instrument and experiment, except the actual digitized data waveforms. The header is 128 locations of 16 bits each. The second portion of the record consists of the data waveform stored as it was digitized in PASS1. Each waveform point occupies two 16-bit words. The first is the number of counts (100 khz oscillator) which occur between the arrival of two pulses within the pulse train. The second value is the 16-bit digital value of a 0-5 volt signal appearing on analog channel 1 of the LPS Unit. This second number in the pair was included so analog filtering techniques could be applied to the waveform before it was digitized or a second telemetry channel could be monitored for auxiliary signals such as the zero voltage crossing point. The waveform data is filled so that it occupies an integral number of blocks. The number of points in the waveform is contained in the header as is the number of blocks in the data portion of the record.

Each data waveform is a complete and self contained record. This allows access of any desired information with only one search. Some of the header information is redundant, but since the storage media is magnetic tape the cost for the redundancy is minimal.

A particular waveform is accessed by reference to the time in seconds that it occurred relative to launch. This time is a binary number stored in one 16-bit location and represents the time from launch in seconds. The conductivity values are also stored in the header portion and provision has been made for a total of 8 conductivities. The t s and t f locations store the starting and ending points of the waveform segment used for the least squares fit. $V\sigma$ is reserved for insertion of the zero crossing point.

A few locations in the header have been reserved for unique comments inserted by the operator during PASS2. Several other locations are available for future uses.

A table showing the exact layout of the header portion follows as does a listing of the Fortran variables used in the utility programs.

Single Waveform Data Storage Structure

0		32	Special Identification
1	DATE	33	
2	15 Mar 79	34	
3		35	
4	Sensor Type BP-G	36	
5	Sensor Number	37	
6	Launch	38	Relative Time for
7	Site	39	Zero Potential Crossing TVρ
8	R_F	40	Altitude
9		41	
10	R_{cal}	42	
11		43	Vertical Velocity
12	r for Blunt Probe	44	Saturation Current
13	r_i for Gerdien	45	ISAT
14	R for Blunt Probe	46	
15	r_o for Gerdien	47	
16	o for Blunt Probe	48	σ+1
17	λ for Gerdien	49	
18	$\frac{df}{d\tau_{cal}}$	50	$\tau_{s\sigma+1}$
19		51	
20		52	
21	$\Delta\tau$ sweep	53	$\tau_{f\ \sigma+1}$
22		54	Vσ+1
23	Δv sweep	55	
24		56	σ -1
25	v sweep	57	
26		58	$\tau_{s\sigma-1}$
27	v sweep+	59	
28		60	$\tau_{f\sigma-1}$
29		61	
30		62	
31	Special Identification	63	Vσ -1

Single Waveform Data Storage Structure(Cont.)

64 65	$\sigma+2$
66 67	$\tau_s \sigma+2$
68 69	$\tau_f \sigma+2$
70 71	$V\sigma+1$
72 73	$\sigma-2$
74 75	$\tau_s \sigma-2$
76 77	$\tau_f \sigma-2$
78 79	$V\sigma-2$
80 81	$\sigma+3$
82 83	$\tau_s \sigma+3$
84 85	$\tau_f \sigma+3$
86 87	$V\sigma+3$
88 89	$\sigma-+3$
90 91	$\tau_s \sigma+3$
92 93	$\tau_f \sigma+3$
94 95	$V\sigma+3$

96 97	$\sigma+4$
98 99	$\tau_s \sigma+4$
100 101	$\tau_f \sigma+4$
102 103	$V\sigma+4$
104 105	$\sigma-4$
106 107	$\tau_s \sigma-4$
108 109	$\tau_f \sigma-4$
110 111	$V\sigma-4$
112 113	
114 115	
116 117	
118 119	ϕ ALWAYS
120 121	Time
122 123	
124 125	N Points
126 127	N Blocks

DEFINITION OF VARIABLES IN FLIGHT HEADER

WORD	FORTRAN VARIABLE	TYPE	USE	FORMAT
0-1	DATE(1)	R	DDMM	A4
2-3	DATE(2)	R	MYY	A4
4	STYPE	I*2	Sensor type BP-GC	A2
5	SNUMB	I*2	Sensor Number	I3
6-7	LASITE	R	Launch Site	A4
8-9	RF	R	Feedback Resistor(OHMS)	F8.0
10-11	RCAL	R	Calibration Resistor(OHMS)	F8.0
12-13	R1	R	Collector Radius(CM) Small R-Blunt Probe Small RI-Gerdien	F4.1
14-15	R2	R	Outer Plate Radius(CM) or Guard R=Blunt Probe RO=Gerdien	F4.1
16-17	L	R	Electrode Length(CM) =Blunt Probe L=Gerdien	F4.1
18-19	DFDTCL	R	DF/DT Cal	F6.2
20-21	DTSW	R	Delta Time Sweep	F6.2
22-23	DVSW	R	Delta Voltage Sweep	F6.2
24-25	VSWN	R	Maximum Negative Value of Voltage Sweep	F6.2
26-27	VSWP	R	Maximum Positive Value of Voltage Sweep	F6.2
28-29	DUM1	R	Dummy Variable	
30-37	IDEN(8)	I*2	Room for 16 Characters of Special Identification	8A2

WORD	FORTRAN VARIALBE	TYPE	USE	FORMAT
38-39	TV	R	Time From Synchronization Pulse at Beginning of Waveform to Zero Potential Crossing Point	
40-41	ALT	R	Probe Altitude(KILOMETERS)	
42-43	VERVEL	R	Vertical Velocity(METERS/SECONDS)	
44-45	ISAT	R	Saturation Current(AMPS)	
46-47	DUM2	R	Dummy Variable	
			REDUCED DATA SECTION	
48-49	SIG(1,1)		Positive Ion Conductivity Species1	
50-51	SIG(2,1)		Starting Time for Slope Determination	
52-53	SIG(3,1)		Ending Time for Slope Determination	
54-55	SIG(4,1)		Probe Potential Estimate	
56-63			Same as Above But For First Negative Species	
64-71			Sigma Positive (2)	
72-79			Sigma Negative (2)	
80-87			Sigma Positive (3)	
88-95			Sigma Negative (3)	
96-103			Sigma Positive (4)	
104-111			Sigma Negative (4)	
112-113	DUM3	R	Dummy Variable	
114-115	DUM4	R	Dummy Variable	

WORD	FORTRAN VARIABLE	TYPE	USE	FORMAT
116-117	DUM5	R	Dummy Variable	
118-119	ZERO	R	Real Variable Always Zero. It is Marker for Beginning of Data Block	
120	TIME	I	Time from External BCD Clock	
121-123	DUM6	R	Dummy Variable	
124-125	NPTS	I*4	Number of Data Points in Waveform. Data is 2*NPTS Long	
126-127	NBLCKS	I*4	Number of Blocks Required for Waveform	

APPENDIX 3

LISTING OF NAMES OF PROGRAMS AND BRIEF DESCRIPTION

PRINCIPLE PROGRAMS USED WITH

THE DATA REDUCTION AND MANAGEMENT SYSTEM

PROGRAM	DESCRIPTION
PASS1	Digitizes the incoming data pulses grabs the time from the real-time clock at the beginning of the waveform puts in end of waveform marker.
PASS2	Adds header information to each waveform data set. Decodes timing. Puts everything into standard format.
PASS3	Does least squares curve fit through interaction with operator. Displays time of occurrence of waveform. Displays linear approximation along with waveform. Computes slopes and sigmas inserts final information into data set.
MOWS.FOR	Extracts flight information from the library and prints it on the line printer. The listing includes header information concerning launch location and instrument parameters plus all conductivities.
MXYI.FOR	Takes a Unix virtual array containing altitudes and conductivities, requests header information from the operator and combines the above into data sets compatible with the system library.
MXYO.FOR	Extracts a data set from the system library and converts it to a Unix virtual array for subsequent plotting.
MIWS.FOR	Creats a dummy data set and stores it in the system library. Header information comes from console. The rest is simulated. Used for debugging only.

PROGRAM	DESCRIPTION
PLOT	Unix plotting routine creates the altitude vs conductivity plots from virtual arrays.
QUESTN	Inputs from the console device header information concerning launch location and date and instrument parameters.
PIP	RT-11 utility transfers files and catalogs storage devices
RTPIP	Unix utility that copies RT-11 files into the Unix system.
unixxyp.bas	Unix basic-plus program to input altitudes and sigmas from a terminal and/or correct previously input data. The output is a virtual file ready for plotting.
xylist.bas	Unix basic-plus program which lists out files which have been entered using unixxyp.bas.

ACKNOWLEDGMENT

Assembly Language Routines were coded by Mr. John Ho under the direction of Dr. D. C. Schroder.

APPENDIX 4

PASS1 INITIAL DIGITIZATION

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                 ;
;        PASS1 OF DATA PROCESSING                 ;
;                                                 ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;        THIS ROUTINE USES THE LPSKW TO COUNT
;        AT 100 KHZ AND THE SCHMIDT-TRIGGER TO
;        SAMPLE COUNTS BETWEEN PULSES FROM THE
;        TAPE RECORDER
;
;        FROM THE COUNTS, THE FREQUENCY OF THE
;        PULSE AND THE TIME FOR THE PULSE CAN
;        BE DETERMINED
;
;        THE ADCS PROVIDE COMMANDS FOR THE ROUTINE
;
;        CHANNEL 0 IS USED TO OBTAIN SIGNAL FOR
;        SEPARATION OF INDIVIDUAL WAVEFORMS
;        IT INSERTS A MARKER TO SIGNIFY THIS IS A
;        NEW WAVEFORM, AND PUTS BCD TIMING INFORMATION
;        OBTAINED FROM AN EXTERNAL TIME CODE GENERATOR
;        THROUGH THE DIGITAL I/O PORT OF THE LPS
;
;        CHANNEL 1 IS USED FOR THE INPUT OF AN ANALOG
;        INPUT WHICH IS A ANALOG FILTERED AND REBUILT
;        VERSION OF THE ORIGINAL SIGNAL
;        THIS INFORMATION IS STORED SIDE BY SIDE WITH
;        THE CLOCK COUNTS AS A PAIR AND CAN PROVIDE
;        ADDITIONAL INFORMATION ABOUT THE WAVEFORM
;
;        CHANNEL 2 PROVIDES COMMANDS FOR THE DATA
;        REDUCTION PROCESS
;        NORMALLY IT HAS A HIGH(5V) ON IT
;       IF IT DROPS TO A LOW 0v THIS SIGNIFIES THE
;        END OF PASS1
;       A MARKER IS INSERTED TO FILL UP THE REST OF THE
;       BUFFER AND ALL FILES ARE CLOSED
;
;
;
;        SOME CONSTANT DEFINITIONS
;
;        THIS ROUTINE IS HIGHLY PARAMETERIZED
;        CAN ADAPT TO OTHER SYSTEMS WITH MINIMAL CHANGES
;
;
        BUFSIZ = 4096.                ; BUFFER SIZE = 4K
        FILSIZ = 1000.                ; DEFAULT FILE SIZE = 1000 BLOCKS
        RECLEN = 16.                ; 4K BUFFER = 16 BLOCKS LONG
        CHAN0 = 1                 ; BIT PATTERN TO START A ADC IN CH0
        CHAN1 = 401                 ; START ADC AT CHANNEL 1
        CHAN2 = 1001                 ; START ADC AT CHANNEL 2
        KWENBL = 1505                 ; START LPSKW, 100KHZ, REPEAT MODE
```

```
                ERASE = 10000                ; ERASE THE SCOPE
                VCRDY = 2012                ; LPSVC Y-MODE, FAST INTENSIFY
                EOWF = 0                ; END OF WAVEFORM MARK
                NOINT = 340                ; BR7, NO INTERRUPT
                ZEROV = 5000                ; DIGITAL VALUE OF 0 V FOR ADC
                YVAL = 4000                ; MIDDLE OF SCREEN
                ICNT1 = 4000.                ; COUNT FOR IDLE LOOP1
                ICNT2 = 4000.                ; COUNT FOR IDLE
                XINC = 2                ; DISTANCE BETWEEN POINTS
        ;
        ;
        ;
        ;
        ;
        ;       LPS ADDRESS DEFINITION
        ;
                .NLIST
                ADST = 170400                ; LPSAD STATUS
                ADBF = 170402                ; LPSAD BUFFER
                KWST = 170404                ; LPSKW STATUS
                KWBP = 170406                ; LPSKW BUFFER/PRESET
                DRST = 170410                ; DIGITAL I/O STATUS
                DRIN = 170412                ; DIGITAL I/O INPUT
                DROUT = 170414                ; DIGITAL I/O OUTPUT
                VCST = 170416                ; LPSVC STATUS
                VCX = 170420                ; LPSVC X
                VCY = 170422                ; LPSVC Y
                KWIV = 324                ; LPSKW INTERRUPT VECTOR
                                ; NOTE THIS IS NON-STANDARD
        ;
        ;
        ;
                .LIST
        ;
        ;
        ;
        ;       SOME MACRO DEFINITIONS
        ;
        ;
                .MACRO SAVE A                ; SAVE A ON STACK
                MOV        A, -(%6)        ; PUSH ON STACK
                .ENDM
                .MACRO RESTOR A                ; RESTORE A FROM STACK
                MOV        (%6)+,A                ; POP A OFF STACK
                .ENDM
                .MACRO CALL A                ; JUMP TO SUBROUTINE A
                JSR        %7,A                ; THRU PC
                .ENDM
                .MACRO RETURN                ; RETURN FROM SUBROUTINE THRU PC
                RTS        %7
                .ENDM
                .MACRO WAIT        A,?B        ; IDLE LOOP TILL READY BIT SET
B:                TSTB        @#A
                BPL        B
                .ENDM
```

```
     ;
        ;
        ;
        ;
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ;                                                   ;
        ;                                                   ;
        ;                MAIN PROGRAM                           ;
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        ;
        ;
        .TITLE PASS1
        .SBTTL        PASS1 OF DATA PROCESSING
        .CSECT        PASS1
        .MCALL        ..V2..,.REGDEF
        .MCALL        .PRINT,.FETCH,.EXIT
        .MCALL        .ENTER,.CLOSE,.WRITE
        .MCALL        .WRITW,.WAIT              ; THESE ARE ALL THE SYSTEM
                                          ; MACRO CALLS
        ..V2..
        .REGDEF
        ;
        ;
        ;
PASS1:        ; START PASS1
        MOV         #ISR,@#KWIV          ; THE FIRST INTERRUPT SERVICE
                                          ; ROUTINE TO TAKE CARE OF PREFLIGHT
                                          ; DATA
        MOV         #NOINT,@#KWIV+2             ; BR7
        ;
        .FETCH        #HNDR,#DEVICE               ; GET THE DK HANDLER
        BCC         1$
        MOV         #FERR,R0                ; SOMETHING'S WRONG
        JMP         FAIL                        ; FATAL ERROR
1$:        .ENTER        #AREA,#0,#FILE,#FILSIZ        ; CREATE FILE PASS1.TMP
                                          ; ON CHANNEL 0
        BCC         2$
        MOV         #EERR,R0                ; CANNOT CREATE
        JMP         FAIL                        ; FATAL ERROR
2$:        ;         ERASE THE SCOPE AND GET READY
        ;
        MOV         #ERASE,@#VCST                 ; ERASE
        WAIT         VCST                         ; MAKE SURE IS ERASED
        MOV         #VCRDY,@#VCST           ; GET READY, Y MODE, FAST INTENSITY
        ;
        ; INITILIZE SOME PARAMETER
        ;
        CLR         X                        ; X-COOR OF THE LPSVC
        CLR         RECNO                        ; BLOCK NUMBER OF WRITE
        CLR         WBUF                        ; FLAG FOR DOUBLE BUFFER
                                          ; 0 MEANS BUFFER 1
                                          ; 1 MEANS BUFFER 2
        ;
        ;
        MOV         #KWENBL,@#KWST                 ; START THE CLOCK
```

```
13$:        ;
        MOV         #CHANO,@#ADST                  ; ISSUE TO SAMPLE A VALUE
                                           ; FROM CHANNEL 0 OF ADC
        WAIT         ADST                          ; WAIT FOR READY FLAG TO BE SET
        CMP         @#ADBF,#ZEROV                  ; IF IT IS A VOLTAGE DROP
                                           ; MEANS START DATA ACQUISITION
        BLE         START
        ; OTHERWISE
        ;       ENTER AN IDLE LOOP TO EAT UP SOME TIME
        ;       MAKE SURE EACH ROUND OF THE LOOP IS ABOUT 50 MS
        ;       SO THAT THE ONE SHOT WON'T OVER-SHOOT
        ;
        MOV         #ICNT1,IN1
3$:        DEC         IN1
        BNE         3$                          ; LOOP
        ;
        ;       NOW SEE IF THE SCREEN IS FULL
        ;       ERASE IF NECESSARY
        ;
        CMP         #4095.,X
        BGE         13$                          ; NOT YET, ALL RIGHT
        MOV         #ERASE,@#VCST                  ; ERASE SCREEN
        WAIT         VCST
        MOV         #VCRDY,@#VCST                  ; SET IT UP AGAIN
        CLR         X                          ; CLEAR X-COOR
        BR         13$                          ; LOOP AGAIN
        ;
        ;
        ;       THE ABOVE LOOP IS TO TAKE CARE OF PRE-FLIGHT DATA
        ;       IT CONTINUES LOOPING UNTIL A PULSE IS DETECTED
        ;       FROM CHANNEL 0 SIGNIFYING THAT DATA IS NOW COMING IN
        ;
        ;       THE REAL THING STARTS HERE
        ;
        ;
        ;
START:        CLR         @#KWST                          ; STOP THE CLOCK FIRST
        MOV         #KWSERV,@#KWIV                  ; A NEW INTERRUPT SERVICE ROUTINE
        ; DO NOT NEED TO CHANGE NEW PSW, IS STILL BR7
        MOV         #BUFSIZ,R2                  ; R2 IS USED AS COUNTER
        MOV         #BUFF1,R1                  ; R1 HAS ADDRESS FOR BUFFER
                                           ; START FILLING BUFFER1 FIRST
        CALL         POLL                          ; GET BCD TIME FOR THE FIRST WAVEFORM
        MOV         #ICNT1,IN1
1$:        DEC         IN1
        BNE         1$
        ;
        ;       THIS IS THE MAIN LOOP HERE
        ; THE ROUTINE WILL LOOP FOREVER UNTIL A VOLTAGE FALL IS DETECTED
        ; ON CHANNEL 2 WHICH MEANS THE END OF ALL THIS
        ; MEANWHILE, THE CLOCK COUNTS WILL BE COLLECTED FROM THE
        ; SCHMITT-TRIGGER AND THE LPSKW  AND STORED ON DISK
        ; USING DOUBLE BUFFER SCHEME
        ;
        ;
LOOP:        ; THIS IS THE MAIN LOOP
```

```
        ;
        MOV        #CHAN0,@#ADST             ; GET A READING FROM CHANNEL 0
        WAIT        ADST
        CMP        @#ADBF,#ZEROV             ; IS IT A VOLTAGE DROP
        BGT        7$                     ; NO
        CALL        POLL                     ; OTHERWISE POLL BCD TIME IN
7$:        ;
        ;         THIS IS AN IDLE LOOP  TO KILL TIME
        ;
        MOV        #ICNT2,IN1
4$:        DEC         IN1
        BNE        4$
        MOV        #CHAN2,@#ADST             ; GET A READING FROM CHANNEL 2
        WAIT        ADST                     ; IF IT IS A DROP, THAT'S ALL FOLKS
        CMP        @#ADBF,#ZEROV
        BGT        3$                     ; NO, GO ON
        JMP        EOT                     ; END OF TRANSMISSION
3$:        CMP         X,#4095.              ; SEE IF THE SCOPE IS FULL
        BLE        LOOP                     ; NO
        MOV        #ERASE,@#VCST             ; YES, ERASE SCREEN
        WAIT        VCST
        MOV        #VCRDY,@#VCST             ; SET UP AGAIN
        BR        LOOP
        ;
        ;         THE MAIN ROUTINE ENDS HERE
        ;
        .SBTTL        POLL TIME
        .CSECT        POLL
        ;
        ;         GET TIME CODE INFORMATION FROM THE EXTERNAL
        ;         TIME CODE GENERATOR
        ;         TIME CODE IS IN BCD COMES IN 4 BYTES
        ;         USES DIGITAL I/O ON THE LPS UNIT FOR INPUTING
        ;         TIME CODE GENERATOR INTERFACE HAS A LATCH
        ;         THAT PUT OUT THE FOUR BYTES SUCCESSIVELY ON RECEIVING
        ;         OF HAND SHAKING SIGNALS FROM THE DIGITAL OUTPUT
        ;         JUST PULL BCD IN, DO NOT BOTHER TO DECODE IT YET
        ;
POLL:        CLR        @#KWST                     ; STOP THE CLOCK FIRST
        MOV        #EOWF,R4               ; END OF WAVEFORM MARK
        CALL        STORE                     ; STORE THE END OF WAVEFORM MARK
        MOV        #ERASE,@#VCST             ; ERASE THE SCREEN AS WELL
        CLR        X                      ; RESET THE X-COOR COUNTER
        MOV        #VCRDY,@#VCST             ; SET THE SCOPE UP
        ;
        ;         THE DIGITAL I/O USES NEGATIVE LOGIC
        ;         THEREFORE HAVE TO SEND OUT THE COMPLEMENT BIT PATTERN
        ;
        MOV        #177776,@#DROUT             ; -1 TO PULL IN SECOND
        BIS        #2,@#DRST              ; TAKE TIME IN FROM THE LATCH
        MOV        @#DRIN,R               ; SECOND IN R
        CALL        STORE                     ; STORE IN BUFFER
        MOV        #177775,@#DROUT             ; -2 TO GET MINUTES
        BIS        #2,@#DRST
        MOV        @#DRIN,R
        CALL        STORE                       ; STORE MINUTES IN BUFFER
        MOV        #177773,@#DROUT              ; -4 TO GET DAY OF YEAR
        BIS        #2,@#DRST
        MOV        @#DRIN,R4
        CALL        STORE
        MOV        #177767,@#DROUT
        BIS        #2,@#DRST
        MOV        @#DRIN,R4
        CALL        STORE
        ;
        ;
        MOV        #KWENBL,@#KWST               ; RESTART THE CLOCK
        RETURN                                ; GO BACK
        ;
        ;
        ;        THE FIRST INTERRUPT SERVICE ROUTINE
        ;        IS JUST TO HANDLE PRE-FLIGHT DATA
        ;        NOT MUCH TO IT
        ;
        ;
        .SBTTL        INTERRUPT SERVICE ROUTINES
        .CSECT         ISR
ISR:        MOV          X,@#VCX
        MOV        #YVAL,@#VCY
        ADD        #XINC,X
        RTI                                ; JUST DISPLAY A POINT ON SCOPE
        ;
        ;
        ;        THE INTERRUPT SERVICE ROUTINE TO HANDLE FLIGHT DATA
        ;        IT IS INTERRUPTED THRU THE SCHMITT-TRIGGER 2 OF THE LPSKW
        ;        PULL IN THE CLOCK COUNT AND ALSO A SAMPLE FROM CHANNEL 2
        ;        OF THE ADC
        ;        STORE THEM IN BUFFER
        ;
        ;        DISPLAY BOTH THE CHANNELS ON THE SCOPE
        ;
        ;
KWSERV:        MOV        @#KWBP,R              ; COUNTS IN BUFFER/PRESET REGISTER
        CALL        STORE                      ; STORE IN BUFFER
        MOV        #CHAN1,@#ADST             ; SAMPLING ADC CHANNEL 1
        WAIT        ADST                       ; WAIT TILL READY
        MOV        @#ADBF,R                ; GET THE DIGITIZED VALUE
        CALL        STORE                      ; STORE IN BUFFER
        MOV        X,@#VCX                     ; LOAD X-COOR
        MOV        #YVAL##VCY               ; LOAD Y-COOR
        MOV        R,@#VCY                ; ANALOG CHANNEL TWO
        ADD        #XINC,X                     ; INCREMENT X
        RTI                                ; RETURN FROM INTERRUPT
        ;
        ;
        .SBTTL        FAIL
        .CSECT        FAIL
        ;        FATAL ERROR
        ;
        ;         PRINT ERROR MESSAGE AND EXIT
        ;
        ;
FAIL:        .PRINT                                  ; ERROR MESSAGE IN R0
        .EXIT                              ; LEAVE
        ;
        ;
        .SBTTL         END OF TRANSMISSION
        .CSECT         EOT
        ;         FILL THE REST OF THE CURRENT
        ;         BUFFER WITH MARKER AND WRITE OUT
        ;         THE LAST BUFFER AND CLOSE ALL FILES
        ;         AND EXIT
        ;
EOT:        CLR         @#KWST                        ; STOP THE CLOCK, NO MORE
        TST         R2                     ; IS THE CURRENT BUFFER JUST FULL
        BEQ         1$                     ; IF YES, MUST BE A BIG COINCIDENCE
        ;         FILL WITH MARKER
2$:        MOV         EOJ,(R1)+                ; MOVE END-OF-JOB MARK
        DEC         R2
        BNE         2$                     ; UNTIL WHOLE BUFFER FULL
1$:        TST         WBUF                       ; SEE WHICH BUFFER THIS IS
        BEQ         3$                     ; 0 MEANS THIS IS BUFFER 1
        ;         SHOULD WRITE OUT BUFFER 2
        .WAIT         #0                     ; IN CASE THE LAST WRITE IS NOT FINISHED
        .WRITW         #AREA,#0,#BUFF2,#BUFSIZ,RECNO
        BCC         5$                     ; WRITE OK
6$:        MOV         #WERR,R0                ; NO
        JMP         FAIL                    ; FATAL ERROR
5$:        BR          $                     ; EXIT
3$:        .WAIT         #0                      ; THIS WRITES OUT BUFFER 1
        .WRITW         #AREA,#0,#BUFF1,#BUFSIZ,RECNO
        BCS         6$                     ; ERROR
$:         .CLOSE         #0                      ; CLOSE FILE
        .PRINT         #ENDMSG                       ; PRINT ENDING MESSAGE
        .EXIT                              ; EXIT
        ;
        ;
        .SBTTL         STORE DATA
        ;         STORE DATA POINTS USING DOUBLE BUFFER SCHEME
        ;         DATA TO BE STORED IS PASSED FROM R
        ;         R1 IS A POINTER TO THE CURRENT AVAILABLE LOCATION
        ;         R2 CONTAINS NUMBER OF FREE LOCATIONS LEFT
        ;         WBUF IS A FLAG TO INDICATE WHICH BUFFER IS  CURRENTLY
        ;         BEING WRITTEN  0 = BUFFER 1, 1 = BUFFER 2
        .CSECT         STORE
STORE:        TST         R2                       ; ANY ROOM LEFT
        BNE         3$                     ; YES, EVERYTHING OK
        ;
        CLR         @#KWST                     ; ONE BUFFER IS FULL
                                       ; HAVE TO ISSUE A WRITE, STOP CLOCK
        TST         WBUF                    ; SEE WHICH BUFFER IS FULL
        BNE         1$                     ; 1 MEANS BUFFER 2
        .WAIT       #0                      ; IN CASE THE LAST WRITE IS NOT FINISHED
        .WRITE        #AREA,#0,#BUFF1,#BUFSIZ,RECNO
        ;         ISSUE AN ASYNCHRONOUS WRITE
        ;         CONTROL WILL PASS BACK TO PROGRAM RIGHT AFTER
        ;         WRITE REQUEST IS QUEUED
        ;
        BCS         4$                     ; ERROR
        MOV         #BUFF2,R1             ; RESET POINTER TO BUFFER 2
        INC         WBUF                      ; SET FLAG TO INDICATE  BUFFER 2 USED
        BR         2$
1$:        .WA         #0                     ; THIS  TIME IT IS BUFFER 2
        .WRITE         #AREA,#0,#BUFF2,#BUFSIZ,RECNO
        BCS         4$                     ; ERROR
        MOV         #BUFF1,R1             ; RESET POINTER TO R1
        CLR         WBUF                      ; INDICATE  BUFFER 1 IN USE
        BR         2$
4$:        MOV         #WERR,R0
        JMP         FAIL                      ; WRITE ERROR, FATAL
2$:        ADD         #RECLEN,RECNO              ; UPDATE BLOCK COUNTER
        MOV         #BUFSIZ,R2             ; RESET POINTS COUNTER
        MOV         #KWENBL,@#KWST             ; RESTART THE CLOCK
3$:        MOV         R4,(R1)+              ; STORE IN BUFFER
        DEC         R2
        RETURN
        ;
        ;
        .SBTTL        DATA DEFINITIONS
        .CSECT        DATA
        ;
        .NLIST
DEVICE:        .RAD50/DK /                    ; DEVICE NAME
FILE:        .RAD50/DK PASS1 TMP/          : FILE NAME
AREA:        .BLKW       10.                    ; AREA FOR WRITE
RECNO:        .WORD        0                  ; WRITE RECORD NUMBER
WBUF:        .BLKW       1                   ; FLAG TO INDICATE WHICH BUFFER
EOJ:        .WORD        "FI                  ; END OF JOB MARK
IN1:        .WORD        0                  ; FREE SLOT FOR COUNTING
X:        .WROD        0                    ; X-COOR
        ;
        ;         ERROR MESSAGES
        ;
FERR:        .ASCIZ/ NO DEVICE /             ; FETCH ERROR
EERR:        .ASCIZ/ CANNOT CREATE /             ; ENTER ERROR
WERR:        .ASCIZ/ WRITE ERROR /             ; WRITE ERROR
ENDMSG:        .ASCIZ/ END OF PASS1 /             ; ENDING MESSAGE
        .EVEN
BUFF1:        .BLKW       4096.                    ; BUFFER 1, K
BUFF2:        .BLKW       096.                   ; BUFFER 2, K
HNDR:        .+2                          ; PLACE TO PUT DK HANDLER
        .LIST
        .END PASS1
```

APPENDIX 5

PASS2 HEADER AND HOUSEKEEPING

```
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                               ;
;       PASS2 OF DATA PROCESSING                ;
;                                               ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       THIS PASS READS IN THE DATA ACQUIRED FROM
;       PASS1 IN 'PASS1.TMP' AND SPLITS THEM INTO
;       PROPER WAVEFORMS
;       IT DECODES THE TIMING INFORMATION FROM
;       BCD TO BINARY, INSERTS HEADER INFORMATION
;       AND RULES OUT UNWANTED DATA INFORMATION
;       FINALLY IT ROUNDS THE DATA TO 256 WORD BOUNDARIES
;
;
;
;       SOME MACRO DEFINITIONS:
        .NLIST
        MACRO       PUSH        ; SAVE ALL REGISTERS ON STACK
        MOV        %0,-(%6)
        MOV        %1,-(%6)
        MOV        %2,-(%6)
        MOV        %3,-(%6)
        MOV        %4,-(%6)
        MOV        %5,-(%6)
        .ENDM
        .MACRO      POP
        MOV        (%6)+,%5      ; POP ALL REGISTERS
        MOV        (%6)+,%4
        MOV        (%6)+,%3
        MOV        (%6)+,%2
        MOV        (%6)+,%1
        MOV        (%6)+,%0
        .ENDM
        .MACRO      SAVE        A        ; SAVE A ON STACK
        MOV        A,-(%6)
        .ENDM
        .MACRO      RESTOR       A        ; RESTORE A FROM STACK
        MOV        (%6)+,A
        .ENDM
        .MACRO      CALL        A        ; JUMP TO SUBROUTINE A THRU PC
        JSR        %7,A
        .ENDM
        .MACRO      RETURN                ; RETURN FROM SUBROUTINE
        RTS        %7
        .ENDM
        .LIST
;
; SOME CONSTANT DEFINITIONS
;
;
ERRWD = 52                          ; SYSTEM ERROR WORD
```

```
        INSIZ = 4096.                           ; INPUT BUFFER SIZE = 4K
        OUTSIZ = 4096.                           ; OUTPUT BUFFER SIZE = 4K
        F180 = 560.                           ; COUNT CORR TO 180 HZ
        FEQLIM = 8.                           ; MORE THAN 8 PTS  180 HZ
        SLPLIM = 8.                           ; MORE THAN 8 PTS HAVE NON-RISING SLOPE
        ADBF = 170402
        INCNT = 16.                           ; INPUT BUFFER BLOCK LENGTH
                                           ; 4K = 16. BLOCKS
        HEADLN = 256.                           ; HEADER = 128 WORDS LONG
        ;
        ;
        ;
        .TITLE PASS2
        .SBTTL PASS2 OF DATA PROCESSING
        ;        THIS PASS READS IN DATA FROM THE DATA OBTAINED FROM PASS1
        ;       INSERTS IT
        ;        INTO THE INPUT BUFFER, DECODES 4 WORDS( ONLY THE LOWER BYTE
        ;        OF EACH WORD) OF TIMING INFORMATION INTO BINARY FORM
        ;        SPLIT THE INCOMING DATA INTO SEPARATE WAVEFORMS
        ;        AND THROWS OUT UNNECESSARY DATA ( DATA THAT HAS A MAJORITY
        ;        OF NON-RISING SLOPES AND MAJORITY OF FREQUENCIES  180 HZ)
        ;        FINALLY IT ROUNDS THE DATA INTO CLOSEST 256 WORD BOUNDARIES
        ;
        ;
        ; ALL SYSTEM MACRO CALLS
        .MCALL        ..V2..,.REGDEF
        .MCALL        .FETCH,.ENTER,.LOOKUP
        .MCALL        .READW,.WRITW
        .MCALL        .EXIT,.CLOSE,.PRINT
        .GLOBL        QUESTN                          ; ENTRY POINT FOR QUESTION
        ;
        ;
        ..V2..
        .REGDEF
        ;
        ;
        .CSECT        PASS2
PASS2:        .FETCH        #HNDR,#DEVICE
        BCC        1$                           ; FETCH OK
        MOV        #FERR,R0
        JMP        FAIL                           ; FETCH FAIL
1$:        .LOOKUP        #INAREA,#0,#INFIL        ; OPEN 'PASS1.TMP' ON CHANNEL 0
                                           ; FOR INPUT
        BCC        2$
        TSTB        @#ERRWD                           ; WHAT'S WRONG
        BEQ        3$                           ; 0 MEANS CHANNEL ACTIVE
        MOV        #NERR,R0                   ; NO FILE
        JMP        FAIL
3$:        MOV        #AERR,R0                   ; CHANNEL ACTIVE
        JMP        FAIL
2$:        .ENTER        #OAREA,#1,#OUTFIL,#-1        ; CREATE OUTPUT FILE
        BCC        4$                           ; ENTER OK
        MOV         #EERR,R0                  ; CREATE ERROR
        JMP         FAIL
        ;
        ;
4$:
        ;         INITIALIZE SOME PARAMETERS
        .SBTTL         INIT PARAM                  ; INIT PARAMETERS
        ;
        CLR         INBLK                          ; INPUT BLOCK CNT
        CLR         OUTBLK                          ; OUTPUT BLOCK CNT
        MOV         #INBUF,R1                  ; R1 HAS ADDRESS OF INPUT BUFFER
        MOV         #INSIZ,R2                  ; R2 HAS SIZE OF INPUT BUFFER
        MOV         #OUTSIZ - 128.,OUTCNT         ; 4K - 128. WORDS
        CLR         ENDFLG                          ; END FILE FLAG
        ;
        ;
        .SBTTL         PROCESS HEADER
        ;
        PUSH                                   ; SAVE ALL REGISTERS
        CALL         QUESTN                          ; FORTRAN SUBROUTINE TO GET
                                            ; HEADER INFORMATION
        POP                                   ; RESTORE ALL REGISTERS
        ;
        ; GET THE ZERO IN THERE
        ;
        ;
        MOV         #OUTBUF,R3                  ; R3 POINTS TO BEGINNING OF OUTBUF
        ; BEGIN AT WHERE A ZERO IS INSERTED
        ;
        ADD         #236.,R3                  ; SKIP HEADER
        CLR         (R3)+                          ; PUT ZERO IN THERE
        ; BADDR HOLD BEGINNING ADDRESS
        MOV         R3,BADDR                  ;SAVE THIS ADDRESS FOR LATER USE
        ;
        .SBTTL         READ IN 1'ST BUFFER
        ; READ IN THE FIRST BUFFER JUST TO GET STARTED
        .READW         #INAREA,#0,R1,R2,INBLK
        BCC         6$
        MOV         #RERR,R0
        JMP         FAIL                          ; READ ERROR
6$:        ADD         #INCNT,INBLK                  ; UPDATE INBLK
        ;
        ; THE REAL PROCESSING STARTS HERE
        .SBTTL START                          ; START PROCESSING
        ;
        ;
        ;
        .CSECT START
LOOP:         CALL         READ                          ; READ IN 1'ST PT IN R4
NOREAD:         TST         R4                          ; TILL IT FINDS A ZERO
                                            ; WHICH IS THE END-OF-WAVEFORM MARK
        BEQ         1$
        BR         LOOP
1$:        MOV         R4,(R3)+                  ; PUT IT IN THE OUTPUT BUFFER FIRST
```

```
        CLR       ZFLAG                       ; CLEAR THIS FLAG
        CALL       READ                       ; 1'ST TIMING WORD IN R4
                                      ; ONLY THE LOWER BYTE IS USED
        CALL       DECODE                       ; DECODE INTO BINARY SECOND
        MOV       R4,SEC                       ; SAVE IT
        CALL       READ                       ; THIS TIME IS MINUTES
        CALL       DECODE
        TST       R4                        ; IS IT ZERO
        BEQ       2$
3$:        ADD       #60.,SEC                  ; CONVERT MINUTES TO SECONDS
        DEC       R4
        BNE       3$
2$:        CALL       READ                        ; HOUR
        COMB       R4                       ; IN NEGATIVE LOGIC
        RORB       R4                       ; IS IT SET
        BCC       4$
        ADD       #3600.,SEC                ; YES, ADD 3600 SECONDS
4$:        MOV       SEC,R4               ; NOW SEC HAS THE BINARY EQUIVALENT OF BCD
        MOV       R4,(R3)                       ; PUT TIMING INFO INTO OUTPUT BUFFER
        ADD       #16.,R3
        CALL       LED
        CALL       READ
        ;
        ;
        ;
        ; THE FOLLOWING SECTION OF CODE IS USED TO READ IN 16 PAIR
        ; OF DATA POINTS AND STORE THEM INTO OUTPUT BUFFER 16 AT A TIME
        ; IT THEN COMPARES LIMITS ON NON-RISING SLOPES AND FREQ.  180HZ
        ; IF BOTH LIMITS EXCEEDED, IT WILL TREAT THIS POINT AS THE END
        ; OF THIS WAVEFORM AND THROW OUT THE REST OF THE DATA
        ; AND GO-ON FOR THE NEXT
        ;
READ10:        CLR        FEQFLG                        ; THIS FLAG USED TO INDICATE
                                                          HOW MANY
                                      ; POINTS ARE ABOVE 180 HZ
        CLR       SLPFLG                         ; THIS FLAG USED TO INDICATE
                                      ; NON-RISING SLOPE
        MOV       #31.,TSTCNT                 ; READ IN 1 ALREADY, 31 LEFT
        CALL       READ                       ; THIS IS THE ANALOG
        TST       R4
        ; IF IT EQUALS  0 THIS MEANS THATS THE END OF THE WAVEFORM
        ; ALL POINTS WERE PROCESSED
        BEQ       SAVE10                        ; TAKE CARE OF THE ROUNDING
        MOV       R4,(R3)+                ; SAVE IT IN OUTPUT BUFFER
        DEC       OUTCNT                        ; UPDATE COUNTER
        MOV       R4,R5                       ; GET READY TO DETERMINE SLOPE
                                      ; HAVE TO GET THE 1'ST POINT IN
        CMP       #F180,R4                ; SEE IF IT IS  180 HZ
        BLT       5$
        INC       FEQFLG                        ; IF  INCREMENT FLAG
        ; NOW ENTER THE LOOP FOR THE REST OF THE 31 POINTS
5$:        CALL       READ                        ; READ IN R
        MOV       R4,(R3)+                ; SAVE IN OUTBUF
        DEC       OUTCNT
        DEC       TSTCNT                        ; UPDATE BOTH COUNTERS
        CALL       READ
        TST       R
        BEQ       SAVE10
        MOV         R4,(R3)                         ; ANALOG TOO
        DEC         OUTCNT
        ; COMPARE FREQUENCY AND SLOPE
        CMP         #F180,R4
        BLT         6$
        INC         FEQFLG
6$:        SUB         R5,R4                  ; IF LAST COUNT  PRESENT COUNT
                                ; = RISING SLOPE
        BLT         7$
        INC         SLPFLG
7$:        MOV         (R3)+,R5
        DEC         TSTCNT
        BNE         5$
        ; SEE IF BOTH LIMITS ARE OVER, LOGICAL AND TO GET IT OUT
        CMP         #FEQLIM,FEQFLG
        BGE         READ10
        CMP         #SLPLIM,SLPFLG
        BGE         READ10
        ; WHEN ARRIVE HERE, BOTH LIMITS HAVE BEEN EXCEEDED AND THE
        ; REST OF THE DATA IS TO BE THROWN OUT
        ; THE ZERO FLAG IS USED TO INDICATE WHETHER IT NEEDS A READ OR NOT
        ; JUST BE SURE TO HAVE THE RIGHT 0 IN THERE
        INC         ZFLAG                 ; BACK SPACE 0 OR NOT
        ; THE FOLLOWING SECTION OF CODE WRAPS UP THE PROCESS
        ; IT ROUNDS UP THE REST OF THE BUFFER TO THE CLOSEST
        ; 256 WORD BOUNDARY AND WRITES IT OUT ONTO THE DISK
SAVE10:
        MOV         #OUTSIZ,R5                 ; SEE HOW MANY POINTS IT HAS GOT
        SUB         OUTCNT,R5
        MOV         R5,R4
        SUB         #128.,R4
        ASR         R4                         ; THIS IS THE NUMBER OF PAIRS OF POINTS
                                      ; THEREFORE HAVE TO DIVIDE BY 2
        MOV         R4,NPTS                         ; SAVE IT
        CLR         R4
1$:        INC         R4                     ; THIS IS TO CAL THE NUMBER OF 256 WORDS
                                      ; BLOCKS TO THE NEAREST BLOCK BOUNDARY
        SUB         #256.,R5
        BGT         1$
        TST         R5
        BEQ         2$                         ; JUST HAPPENS TO BE AT BLOCK BOUNDARY
3$:        CLR         (R3)+                        ; CLEAR THE REST
        INC         R5
        BNE         3$
2$:        MOV         R4,NBLK                          ; SAVE NUMBER OF BLOCKS
        CLR         R5
4$:        ADD         #256.,R5                  ; GET BACK NBLKS*256 TO WRITE
        DEC         R4
        BNE         $
        ; WRITE NBLKS AND NPTS BACK TO THE BUFFER
        MOV         #OUTBUF,R
        ADD         #248.,R
        MOV         NPTS,(R)
        ADD         #4,R4
        MOV         NBLK,(R4)
        ; WRITE OUT WAVEFORM COMPLETELY
        .WRITW         #OAREA,#1,#OUTBUF,R5,OUTBLK
        ADD         NBLK,OUTBLK                  ; UPDATE COUNTER
        MOV         #OUTSIZ-128.,OUTCNT          ; RESET OUTPUT SIZE CNT
        MOV         BADDR,R3                  ; RESET BEGIN ADDRESS
        TST         ZFLAG                  ; SEE IF I NEED TO BACK SPACE 0
        BEQ         5$
        JMP         LOOP
5$:        CLR         R
        JMP         NOREAD
        ;
        ; END
        ;
        ; END PASS2 MAIN ROUTINE
        ;
        ;
        .SBTTL READ                           ; RETURN A DATA PT FROM R
                                           ; KEEP TRACK OF BUFFER CONTENT
        ;
        ;
        ; THIS ROUTINE ALWAYS RETURNS THE NEXT POINT IN R4
        ; IF CURRENT BUFFER IS EMPTY, READ IN ANOTHER BUFFER
        ; UNTIL EOF
        .CSECT         READ
READ:        TST         R2                       ; R2 HAS INPUT BUFFER CNT
        BNE         1$                       ; NOT EMPTY YET
        TST         ENDFLG
                                             ; IS THIS THE LAST BUFFER OF THE FILE
        BEQ         2$
        JMP         EOF                       ; YES, LAST BUFFER
2$:        MOV         #INBUF,R1
        .READW         #INAREA,#0,R1,#INSIZ,INBLK
        BCC         3$                       ; READ IN ANOTHER BUFFER
        TSTB         @#ERRWD                       ; WHAT'S WRONG
                                           ; COULD BE EOF OR READ ERROR
        BEQ         4$
        MOV         #RERR,R0
        JMP         FAIL                       ; READ ERROR, FATAL
4$:        TST         R0
        BNE         5$
        JMP         EOF
5$:
        MOV         R0,R2                       ; END OF FILE, R0 CONTAINS ACTUAL
                                           ; NO OF POINTS READ
        INC         ENDFLG                       ; SET END-OF-FILE FLAG
        BR         1$                       ; GO ON WITH LAST BUFFER
3$:        ADD         #INCNT,INBLK              ; UPDATE INPUT BLOCK CNT, K = 16 BLOCKING
        MOV         #INSIZ,R2                  ; REFRESH COUNTER
1$:        MOV         (R1)+,R                  ; R IS THE DATA RETURNED
        DEC         R2                       ; UPDATE CNT
        CMP         R,EOJ                       ; IS IT THE END OF FLIGHT MARK
        BNE         6$
        JMP         EOF                       ; YES, THE END
```

```
6$:
        RETURN
        ;
        ;
        ;
        ; DECODE BCD TIME
        ;
        .SBTTL DECODE
        ; DECODE THE LOWER BYTE OF R4 WHICH IS TO CONTAIN 2
        ; BCD DIGITS AND THE RESULTANT BINARY IS RETURNED IN R4 ALSO
        ; THE INCOMING BCD NUMBER IS IN NEGATIVE LOGIC
        ; THEREFORE HAVE TO COMPLEMENT IT FIRST
        .CSECT       DECODE
DECODE:       CLR        R5                 ; R5 IS TO HOLD THE RESULT TEMPORARILY
        COM        R4
        RORB        R4                       ; R4 HAS BCD NO
        BCC        1$
        INC        R5                       ; R5 WILL HAVE DECIMAL NO
1$:        RORB        R4
        BCC        2$
        ADD        #2,R5
2$:        RORB        R4
        BCC        3$
        ADD        #4,R5
3$:        RORB        R4
        BCC        4$
        ADD        #8.,R5
4$:        RORB        R4
        BCC        5$
        ADD        #10.,R5
5$:        RORB        R4
        BCC        6$
        ADD        #20.,R5
6$:        RORB        R4
        BCC        7$
        ADD        #40.,R5
7$:        RORB        R4
        BCC        8$
        ADD        #80.,R5
8$:        MOV        R5,R4                        ;RETURN RESULT IN R4
        RETURN
        ;
        ;
        ; CLOSE ALL FILES AND PRINT OUT END MESAGE
        .SBTTL        EOF                        ; END-OF-FILE
EOF:
        .CLOSE        #0                        ; CLOSE INPUT FILE
        .CLOSE        #1                        ; CLOSE OUTPUT FILE
        .PRINT        #ENDMSG                        ; END MESSAGE
        .EXIT
        ;
        ;
        .SBTTL        FATAL ERROR
        ; PRINT ERROR MESSAGE AND EXIT
        ;
FAIL:       .PRINT                                  ; MESSAGE IN R0
        .EXIT
        ;
        .SBTTL LED
        .CSECT LED
        ; R4 CONTAINS THE BINARY NUMBER TO BE DISPLAYED
        ; DIVIDE IT INTO POSITIONAL FORM AND PUT IT OUT IN THE LED
LED:        CLR         LED1
        ; CLEAR ALL 6 DIGITS FIRST
        MOV         #17,@#ADBF
        MOV         #17,@#ADBF
        MOV         #1017,@#ADBF
        MOV         #1417,@#ADBF
        MOV         #2017,@#ADBF
        MOV         #2417,@#ADBF
1$:         MOV         #-1,LED2
2$:         INC         LED2
        SUB         #10.,R4                         ; GET REMAINDER
        BGE         2$
        ADD         #10.,R4
        ; SET UP A WHOLE WORD AND GO
        MOVB         R4,LED3
        MOVB         LED1,LED3+1
        MOV         LED3,@#ADBF                 ; ILLUMINATE
        INC         LED1
        MOV         LED2,R4
        TST         R4
        BNE         1$
        RETURN
        ; END LED
LED1:       .WORD 0
LED2:       .WORD 0
LED3:       .WORD 0
        ;
        .SBTTL         DATA DEFINITIONS
        .CSECT         DATA
EOJ:        .WORD         "FI                        ; FINISH MARK
INBLK:        .WORD         0                        ; INPUT BLOCK CNT
OUTBLK:        .WORD         0                        ; OUTPUT BLOCK CNT
OUTCNT:        .WORD         0                        ; OUTPUT BUFFER WORD CNT
TSTCNT: .WORD 0                                 ; TEST COUNT FOR 31 POINTS
BADDR:        .WORD 0                           ; TO HOLD BEGINNING ADDRESS
SEC:        .WORD         0                           ; STORE SECOND
NBLK:        .WORD 0                                ; NUMBER OF BLOCKS
NPTS:        .WORD 0                                ; NUMBER OF POINTS
ZFLAG:        .WORD         0                       ; FREQ/SLOPE FLAG
ENDFLG:        .WORD         0                      ; END-OF-FILE FLAG
SLPFLG:        .WORD         0                      ; SLOPE FLAG
FEQFLG:        .WORD         0                      ; FREQUENCY FLAG
INAREA:        .BLKW         10.                     ; FOR INPUT EMT LIST
OAREA:        .BLKW         10.                      ; FOR OUTPUT EMT LIST
INBUF:        .BLKW         096.                     ; INPUT BUFFER K
OUTBUF:        .BLKW         096.                    ; OUTPUT BUFFER K
        .NLIST
INFIL:        .RAD50/DK PASS1 TMP/                 ; INPUT FILE NAME
```

```
OUTFIL:        .RAD50/DK PASS2 TMP/               ; OUTPUT FILE NAME
DEVICE:        .RAD50/DK /
FERR:        .ASCIZ/NO DEV/                       ; FETCH ERROR
NERR:        .ASCIZ/NO FILE/                        ; NO FILE
AERR:        .ASCIZ/CHAN ACT/               ; CHANNEL ACTIVE
EERR:        .ASCIZ/CANT CREATE/              ; ENTER ERROR
RERR:        .ASCIZ/READ ERR/               ; READ ERROR
OERR:        .ASCIZ/NO ROOM/                        ; OUTPUT BUFFER FULL
WERR:        .ASCIZ/WRITE ERR/               ; WRITE ERROR
ENDMSG:        .ASCIZ/END PASS2/              ; END MESSAGE
        .EVEN
HNDR:        .+2
        .LIST
        .END PASS2
```

APPENDIX 6

PASS3 CURVE FITTING CONDUCTIVITIES

```
C
C        MAIN PROGRAM FOR PASS3 OF DATA PROCESSING
C
C        PROGRAM LAY OUT
C                THIS PROGRAM WILL CALL THE FOLLOWING SUBROUTINES
C
C        SUBROUTINE INIT - INITIALIZES ALL THE PROGRAM PARAMETERS INCLUDING
C                I/O LOGICAL UNIT NUMBERS AND CONSTANTS, OPENS INPUT
C                AND OUTPUT FILES.
C        SUBROUTINE LED - DISPLAYS TIMING INFORMATION ON THE LED DISPLAY
C                OF THE LPS UNIT.
C        SUBROUTINE READIN - READS IN THE NEXT WAVEFORM. SINCE WAVEFORMS
C                ARE OF VARIABLE LENGTH, IT HAS TO READ IN THE 1'ST
C                256 WORDS, AND FIND OUT THE LENGTH OF THAT WAVEFORM IN THE HEADER
C
C        SUBROUTINE DISP - DISPLAY DATA ON THE TEKTRONIX 603 SCOPE
C                ACCORDING TO THE FREQUENCY LIMITS. A FLAG IS USED TO
C                INDICATE WHETHER THE ANALOG CHANNEL IS TO BE DISPLAYED OR NOT.
C
C        SUBROUTINE EXPAND - DETERMINE WHETHER THE WAVEFORM IS TO BE EXPANDED
C                TO DIFFERENT LIMITS.
C
C        SUBROUTINE SLOPE - GET THE BREAK POINTS OF THE WAVEFORM FROM THE
C                TERMINAL AND PERFORM A WEIGHTED LEAST SQUARE STRAIGHT LINE
C                FIT FOR THE SLOPE OF THE WAVEFORM, THEN DISPLAY THE FITTED
C                STRAIGHT LINE ON THE SCREEN
C
C        SUBROUTINE ENDW - END OF WAVEFORM ROUTINE. CALCULATE THE CONDUCTIVITY
C                VALUE FROM THE SLOPE, RESET THE PARAMETERS, WRITE THE HEADER
C                BACK TO THE INPUT FILE AND WRITE CONDUCTIVITY INFO TO OUTPUT FILE
C
C
C
C        DATA BASE DEFINITIONS
C
C
C        COMMON DATA - CONTAINS INTEGER ARRAY IDATA(4096) WHICH
C                IS THE INPUT BUFFER TO HOLD ALL THE DATA POINTS
C        COMMON PARM - CONTAINS ALL THE PARAMETERS AND CONSTANTS
C                IIN - LOGICAL UNIT NUMBER FOR INPUT FILE (19)
C                IOUT - LOGICAL UNIT NUMBER FOR OUTPUT FILE (20)
C                INEXT - RECORD NUMBER IN INPUT FILE TO HOLD NEXT RECORD NO.
C                ILAST - RECORD NUMBER FOR THE BEGINNING OF THE LAST WAVEFORM
C                ISIZ - MAXIMUM SIZE FOR THE INPUT FILE ( REQUIRED BY THE
C                        RT-11 DEFINE FILE )
C                A - SLOPE FOR DEVICE COORDINATE CONVERSION
C                B - INTERCEPT FOR DEVICE COORDINATE CONVERSION
C                C - CONSTANT TO CALCULATE CONDUCTIVITIES
C
C
C
C
C********************************************************
C                                                       *
C        PASS3 OF DATA PROCESSING                       *
C                                                       *
C                                                       *
C********************************************************
```

```
C
C        THE MAIN PROGRAM WILL SET LOGICAL UNIT NUMBER 5 AS THE STANDARD
C        INPUT ( TERMINAL ) AND 6 AS THE STANDARD OUTPUT ( TERMINAL )
C
C        COMMON DECLARATION
C
        COMMON /DATA/ IDATA(4096)        ! HOLD INPUT DATA
        COMMON /PARM/ IIN,IOUT,INEXT,ILAST,ISIZ,A,B,C        ! PARAMETERS
C        EQUIVALENCE PART OF THE HEADER INFORMATION
C        TIMING, NUMBER OF (PAIR) POINTS, NUMBER OF 256 WORD BLOCKS
        EQUIVALENCE (ITIME,IDATA(121)),(NPTS,IDATA(125)),(NBLK,IDATA(127))
        DATA ICHAR,JCHAR /'  ','Y '/        ! NOTE UNIX WILL PUT 'Y' IN LOWER BYTE
C
C        START
C
        CALL ASSIGN(6,'TT:/N')                ! FOR RT-11 ASSIGN 6 AS TERMINAL
C        UNIX DOES NOT REQUIRE THIS, THIS IS DEFAULT
C
        NWAVE = 1                        ! WAVEFORM NUMBER
        WRITE(6,1)                        ! WRITE OUT MESSAGE
1        FORMAT('**** PASS3 OF DATA PROCESSING ****')
C
C       INITIALIZE PARAMETERS
        CALL INIT
C        DETERMINE WHICH WAVEFORM TO START WITH
        WRITE(6,2)
2        FORMAT('INPUT STARTING WAVEFORM,I4 FORMAT')
        READ(5,3) NSTART
3        FORMAT(I4)
        IF( NSTART .LE. 1 ) GOTO 10        ! A -R IS INTERPRET AS 0
C        OTHERWISE SKIP WAVEFORMS
        DO 100 I = 1,NSTART-1                ! SKIP NSTART - 1 WAVEFORMS
100        CALL READIN
        NWAVE = NSTART                        ! UPDATE WAVEFORM NUMBER
C
C        THE MAIN LOOP IS HERE
C
10        CALL READIN                        ! READ IN THE DESIRED WAVEFORM
C        THIS IS WHY NSTART-1 WAVEFORM IS SKIPPED -- THIS CALL TO
C        READIN WILL GET TO THE RIGHT WAVEFORM
        CALL LED(ITIME)                        ! DISPLAY THE TIME OF THIS WAVEFORM
C        WRITE SOME INFORMATION OUT
        WRITE(6,4) NWAVE,ITIME,NPTS,NBLK
4        FORMAT('WAVEFORM',I4,5X,'TIME',I5,5X,
     1        I5,5X,'POINTS',I4,5X,'BLOCKS')
C        IF THIS IS AN EMPTY WAVEFORM, SKIP TO THE NEXT
        IF( NPTS .LE. 1) GOTO 11        ! UPDATE COUNTER AND READ AGAIN
C        DISPLAY WAVEFORM WITH ANALOG CHANNEL
        CALL DISP(1,NPTS,0,200,1)        ! ALL POINTS, 0-200 HZ, FLAG = 1
                                        ! MEANS DISPLAY ANALOG CHANNEL TWO
        CALL EXPAND                        ! SEE IF IT REQUIRES EXPANSION
        CALL SLOPE                        ! GET BREAK POINTS AND FIT SLOPE
        CALL ENDW                        ! END OF WAVEFORM
C        SEE IF THERE IS ANY MORE DATA FROM THE INPUT FILE
```

```
        IF( INEXT .GE. ISIZ ) GOTO 1000          ! END
11        NWAVE = NWAVE + 1                    ! UPDATE WAVEFORM NUMBER
        WRITE(6,5)
5        FORMAT('CONTINUE ?')
        READ(6,6) JCHAR                          ! GET ANSWER FROM TERMINAL
6        FORMAT(A1)
        IF( ICHAR .EQ. JCHAR ) GOTO 10         ! YES
C        OTHERWISE
1000        WRITE(6,7)
7        FORMAT('**** END OF PASS3 ****')         ! END MESSAGE
        END
```

```
C
C        THIS PROGRAM MODULE CONTAINS MOST OF THE SUBROUTINES
C        PASS3 CALLS
C
C
C
C
C
C        SUBROUTINE INIT INITIALIZES PARAMETERS AND OPENS INPUT AND
C        OUTPUT FILES
C
        SUBROUTINE INIT
C        COMMON DEFINITIONS
        COMMON /DATA/ IDATA(4096)
        COMMON /PARM/ IIN,IOUT,INEXT,ILAST,ISIZ,A,B,C
C       INITIALIZE I/O LOGICAL UNIT NUMBERS
        IIN = 19
        IOUT = 20
C        OPEN I/O FILES
C
C        ASSIGN IS AN RT-11 SYSTEM ROUTINE
C        UNIX WILL CALL SETFIL WHICH WILL DO THE SAME THING
C
        CALL ASSIGN(IIN,'PASS2.TMP',9,'OLD')          ! INPUT FILE
        CALL ASSIGN(IOUT,'PASS3.TMP',9,'NEW')          ! CREATE OUTPUT FILE
C        A AND B ARE CONSTANTS TO CONVERT FREQUENCY TO DEVICE COORDINATE
C        ON THE TEKTRONIX 603
        A = 2.057789
        B = -A
C
C        DEFINE FILE
        WRITE(6,1)
1        FORMAT('NUMBER OF BLOCKS IN PASS2.TMP?,I4 FORMAT')
        READ(5,2) N
2        FORMAT(I4)
        ISIZ = N                            ! STORE IT AWAY
C        THIS IS A STANDARD RT-11 FORMAT, OTHER SYSTEMS MIGHT NEED SOME
C        MODIFICATIONS
C        EACH RECORD IS 256 WORDS LONG
        DEFINE FILE IIN( N, 256, U, INEXT )          ! INEXT WILL POSITION TO THE
                                            ! NEXT AVAILABLE RECORD
        WRITE(6,3)
3        FORMAT('CALCULATION CONSTANT?')
        READ(6,4) C                            ! SIGMA = C * DF/DF
4        FORMAT(E12.5)
        INEXT = 1                            ! MAKE SURE IT STARTS READING FROM
C                                            BLOCK 1
        RETURN
        END
C
C
C
```

```
C        THIS ROUTINE READS IN THE NEXT WAVEFORM
C        INEXT ALWAYS POINTS TO THE NEXT RECORD AVAILABLE
C        SET ILAST - INEXT TO REMEMBER WHERE THE LAST WAVEFORM IS
C        SO THAT AFTER THE CALCULATIONS THE INFORMATION CAN BE
C        WRITTEN BACK TO THE HEADER
C        READ IN THE 1'ST BLOCK FIRST BECAUSE EACH WAVEFORM IS AT
C        LEAST 1 BLOCK LONG
C        FIND OUT HOW LONG THIS WAVEFORM IS AND READ IN THE REST
C
        SUBROUTINE READIN
C        COMMON DEFINITIONS
        COMMON /DATA/ IDATA(096)
        COMMON /PARM/ IIN,IOUT,INEXT,ILAST,ISIZ,A,B,C
C        EQUIVALENCE THE 1'ST 256 WORDS
        INTEGER IBUF(256),JBUF(256)
        EQUIVALENCE (IBUF(1),IDATA(1)),(NBLK,IDATA(127))
C        SET ILAST = INEXT
        ILAST = INEXT
C        READ IN THE 1'ST BLOCK
        READ(IIN'INEXT) IBUF                ! READ IN 256 WORD RECORD
        IF( NBLK .EQ. 1 ) RETURN        ! ONLY 1 BLOCK LONG
C        OTHERWISE HAVE TO READ THE REST
        ISTART = 256                        ! THERE ARE 256 POINTS ALREADY
                                        ! START AT 257 POINT
        DO 100 I = 1,NBLK - 1                ! DO THE REST
        READ(IIN'INEXT) JBUF                ! READ IN 256 WORDS AT A TIME
C        COPY JBUF INTO IDATA AT THE RIGHT BOUNDARY
        DO 101 J = 1,256
101        IDATA(J + ISTART) = JBUF(J)
        ISTART = ISTART + 256                ! SET FOR THE NEXT BOUNDARY
100        CONTINUE
C        THE WHOLE WAVEFORM IS IN IDATA() NOW, CAN RETURN
        RETURN
        END
C
C        THIS ROUTINE CONVERTS THE COUNTS IN IDATA() INTO FREQUENCY
C        AND CONVERTS FREQUENCY INTO DEVICE COORDINATE BETWEEN
C        0 - 4095
C
        INTEGER FUNCTION ICONV(ICNT)
        INTEGER ICNT        ! ICNT = CLOCK COUNTS = 100KHZ/FREQ
        COMMON /PARM/ IIN,IOUT,INEXT,ILAST,ISIZ,A,B,C
        IF( .NOT.( ICNT .LE. 500 )) GOTO 1        ! FREQ.  200 HZ
        ICONV = 4095                        ! RETURN 4095 FOR  200 HZ
        RETURN
1        FREQ = 1.0E5/FLOAT(ICNT)        ! FREQ = 100KHZ/COUNTS
        I = INT(( A * FREQ + B + 0.5 ))        ! CONVERT TO DEV. COOR.
        IF( I .LT. 0 ) GOTO 2                ! OUT OF BOUNDS
        ICONV = I                        ! RETURN DESIRED VALUE
        RETURN
2        ICONV = 0
        RETURN
        END
C
C
C
```

```
 C    THIS ROUTINE EXPANDS THE WAVEFORM TO A SET OF NEW LIMITS
C        AND DISPLAYS WITHOUT THE ANALOG DATA
C
        SUBROUTINE EXPAND
        COMMON /PARM/ IIN,IOUT,INEXT,ILAST,ISIZ,A,B,C
        DATA ICHAR,JCHAR /'  ','Y '/        ! UNIX PUTS 'Y' IN LOWER BYTE
10        WRITE(6,1)
1        FORMAT('EXPAND ?')
        READ(5,2) ICHAR                        ! A -R MEANS NO
2        FORMAT(A1)
        IF(ICHAR .NE. JCHAR) RETURN        ! DO NOT EXPAND
        WRITE(6,3)                        ! GET THE LIMITS
3        FORMAT('LOW-X, HIGH-X, LOW-Y, HIGH-Y ?')
        READ(5,4) ILX,IUX,ILY,IUY
4        FORMAT(4I10)
C        CALCULATE NEW A AND B
        A = 4095./(FLOAT(IUY-ILY))
        B = -A * FLOAT(ILY)
C        DISPLAY THE EXPANDED WAVEFORM WITHOUT ANALOG
        CALL DISP(ILX,IUX,ILY,IUY,0)
        GOTO 10                                ! AGAIN
        RETURN
        END
C
C        THIS ROUTINE CALCULATES THE CONDUCTIVITY AND OUTPUTS IT TO
C        THE OUTPUT FILE
C        WRITE THEM BACK TO THE HEADER AND RESET ALL THE NECESSARY
C        PARAMETERS
        SUBROUTINE ENDW
        COMMON /DATA/ IDATA(4096)
        COMMON /PARM/ IIN,IOUT,INEXT,ILAST,ISIZ,A,B,C
        INTEGER IBUF(256)
        EQUIVALENCE (IBUF(1),IDATA(1))
        I = INEXT
        WRITE(IIN'ILAST) IBUF                ! WRITE THE HEADER BACK
        INEXT = I
        FIND(IIN'I)                        ! POSITION BACK TO THE NEXT RECORD
        A = 2.0577889                        ! RESET A AND B
        B = -A
        RETURN
        END
        ; THE FOLLOWING TWO ROUTINES PROVIDE LED DISPLAY AND
        ; TEKTRONIX 603 DISPLAY
        ;
        ; LED AND LEDD ARE TWO ENTRY POINTS FOR DISPLAYING NUMERIC
        ; VALUES ON THE LED DISPLAY
        ; LED IS CALLED FROM FORTRAN AND LEDD EXPECTS A NUMBER IN R4 ON CALLING
        ;
        ; DISP IS CALLED FROM FORTRAN AND EXPECTS FIVE(5) ARGUMENTS
        ; FIRST TWO(2) ARE THE X LIMITS
        ; SECOND TWO(2) ARE THE Y LIMITS
        ; FIFTH IS A FLAG - IF SET, DISPLAY BOTH CHANNELS
        ;        IF CLEARED, DISPLAY ONLY THE DIGITIZED VALUE
        ;
        ;
        ;
        ;
        ; ASSEMBLER ROUTINES FOR PASS3 DATA PROCESSING
        ;
        .MACRO       PUSH
        MOV        %0,-(%6)
        MOV        %1,-(%6)
        MOV        %2,-(%6)
        MOV        %3,-(%6)
        MOV        %4,-(%6)
        MOV        %5,-(%6)
        .ENDM
        .MACRO POP
        MOV        (%6)+,%5
        MOV        (%6)+,%4
        MOV        (%6)+,%3
        MOV        (%6)+,%2
        MOV        (%6)+,%1
        MOV        (%6)+,%0
        .ENDM
        ;
        .MACRO RETURN
        RTS %7
        .ENDM
        ;
        .MACRO CALL A
        JSR        %7,A
        .ENDM
        ;
        ;
        .MCALL ..V2..,.REGDEF
        .MCALL .PRINT
        ..V2..
        .REGDEF
        .GLOBL LED LEDD
        .GLOBL ICONV
        .TITLE LED DISPLAY
        ADBF=170402
        .CSECT LEDIS
LED:        PUSH
        MOV        @2(R5),R4
        BR         START
LEDD:        PUSH                ; IN R4 ALREADY
START:
        CLR         LED1
        MOV         #17,@#ADBF
        MOV         #417,@#ADBF
        MOV         #1017,@#ADBF
        MOV         #1417,@#ADBF
        MOV         #2017,@#ADBF
        MOV         #2417,@#ADBF
1$:        MOV         #-1,LED2
2$:        INC         LED2
        SUB         #10.,R4
        BGE         2$
        ADD         #10.,R4
        MOVB         R4,LED3
        MOVB         LED1,LED3+1
        MOV         LED3,@#ADBF
        INC         LED1
        MOV         LED2,R4
        TST         R4
        BNE         1$
        POP
        RETURN
LED1:        .WORD 0
LED2:        .WORD 0
LED3:        .WORD 0
;        .END LED
        ;
        ;
        .SBTTL DISPLAY
        .CSECT DISPLAY
        .GLOBL DISP
        VCST=170416
        VCX =170420
        VCY =170422
        VCRDY = 2010
        ERASE = 10000
DISP:        PUSH                        ; SAVE ALL REGISTER
        MOV         @2(R5),IX1
        MOV         @4(R5),IX2          ; X - AXIS
        MOV         @6(R5),IY1
        MOV         @10(R5),IY2          ; Y - AXIS
        MOV         @#12(R5),FLAG
        TST         IY1
        BNE         5$
        INC         IY1
5$:
        MOV         #PLIST,R5          ; PARM LIST
        MOV         #P,2(R5)
        MOV         #1,XINC
        ;
        CMP         IX1,IX2                ; CHECK LIMITS
        BLT         1$
3$:        .PRINT        #ERR1                ; LIMIT ERROR
        RETURN
1$:        CMP         IY1,IY2
        BGE         3$
        MOV         #ERASE,@#VCST
4$:        TSTB        @#VCST
```

```
        BPL         4$
        MOV         #VCRDY,@#VCST
        ;
        ; WRITE X AXIS
        ;
        MOV         #XLABEL,R1
XAXIS:        MOV         (R1)+,R2
        TST         R2
        BEQ         YAXIS
        MOV         R2,@2(R5)
        PUSH                 ; SAVE ALL REGISTERS
        CALL         ICONV                ; RETURN IN RO
        MOV         RO,Y         ; RESULT IN RO
        POP                          ; RESTORE ALL REGISTERS
        MOV         Y,RO
        ;
        TST         RO
        BLT         XAXIS
        CMP         RO,#7777
        BGT         YAXIS
        ;
        CLR         R4         ; USE AS SCOPE COUNTER
6$:        MOV         R4,@#VCX
        MOV         RO,@#VCY
        ADD         #20.,R4
        CMP         #7777,R4
        BGT         6$
        JMP         XAXIS
YAXIS:        ; DRAW Y AXIS
        ;
1$:        MOV         IX2,R1
        SUB         IX1,R1                ; CAL X INCREMENT
        MOV         #4096.,R2
        CLR         R3
2$:        INC         R3
        SUB         R1,R2
        BGT         2$
        CMP         R3,#10.                ; IF  10 USE 10
        BLE         3$
        MOV         #10.,XINC
        BR         4$
3$:        MOV         R3,XINC
$:        MOV         #IDATA,R1         ; R1 HAS ADDR
        ADD         #248.,R1         ; SKIP HEADER
        MOV         (R1),NPTS         ; FIND NO OF POINTS
        ;
        ADD         #8.,R1
        MOV         IX2,R2
        SUB         IX1,R2
        INC         R2
        CMP         R2,NPTS
        BLE         5$
        MOV         NPTS,R2
5$:        DEC         IX1
        ASL         IX1
        ASL         IX1
        ADD         IX1,R1
        CLR         X
6$:         MOV         #50.,R3
        MOV         #2010,@#VCST
7$:         MOV         (R1)+,@2(R5)
        PUSH
        CALL         ICONV
        MOV         R0,Y
        POP
        MOV         Y,R0
10$:          MOV          X,@#VCX
        MOV         R0,@#VCY
11$:          TSTB         @#VCST
        BPL         11$
        TST         FLAG
        BNE         9$
        ADD         #2,R1
        BR         13$
9$:
        MOV         (R1)+,@#VCY
12$:          TSTB         @#VCST
        BPL         12$
13$:          ADD          XINC,X
        CMP         #7777,X
        BLT         EXIT
        DEC         R2
        BEQ         EXIT
        DEC         R3
        BNE         7$
        CLR         Y
8$:         MOV         #VCRDY,@#VCST
        MOV         Y,@#VCY
        ADD         #20.,Y
        CMP         #7777,Y
        BGT         8$
        JMP         6$
EXIT:          POP
        RETURN
        .CSECT DATA
IDATA:          .BLKW         4096.
        ;
        ;
        .CSECT DISPLAY
PLIST:          .WORD         1
        .WORD         0
P:         .WORD         0
IX1:          .WORD         0
IX2:          .WORD         0
IY1:          .WORD         0
IY2:          .WORD         0
XLABEL:          .WORD         500.,250.,167.,125.,100.,83.,71.,63.,56.,50.,0
XINC:          .WORD         0
NPTS:          .WORD         0
X:         .WORD         0
Y:         .WORD         0
FLAG:          .WORD         0
```

```
        .NLIST
ERR1:        .ASCIZ/ERR IN LIMITS/
        .LIST
        .EVEN
        .CSECT INTE
        .SBTTL         INTENSIFY A POINT
        .GLOBL         INTEN
INTEN:        PUSH
        MOV         @#VCST,VCSTAT
        MOV         #2010,@#VCST
1$:        TSTB         @#VCST
        BPL         1$
        MOV         @2(R5),@#VCX
        MOV         @4(R5),@#VCY
2$:        TSTB         @#VCST
        BPL         2$
        MOV         VCSTAT,@#VCST
        POP
        RETURN
VCSTAT:        .WORD         0
        .END
```

APPENDIX 7

MOWS.FOR
LISTS OUT HEADER INFORMATION AND CONDUCTIVITIES ON LINE PRINTER

```
C  MAIN PROGRAM TO DUMP DATA FILES FROM PREVIOUSLY REDUCED FLIGHTS
      COMMON DATE(2),STYPE,SNUMB,LASITE,RF,RCAL,R1,R2,L,DFDTCL,
     1DTSW,DVSW,VSWN,VSWP,DUM1,IDEN    ,TVO,ALT,VERVEL,ISAT,DUM2,
     2SIG(4,8),DUM3,DUM4,DUM5,ZERO,TIME(2),NPTS,NBLKS,
     3LUN,NREC,IN,IOUT,LP
        INTEGER*2 IBUF(128),IDEN(8)
      INTEGER*2 STYPE,SNUMB
        INTEGER*2 JBUF(128)
        INTEGER*4 NPTS,NBLKS
      EQUIVALENCE (DATE(1),IBUF)
      REAL LASITE,L,ISAT
C
C  SET COMMON BLOCK EQUIVALENT TO IBUF FOR MAG TAPE I/O
C
C


        IN=5
        IOUT=7
        LP=6
        WRITE(IOUT,10)
10        FORMAT(' ENTER LOGICAL UNIT NUMBER FOR MAG TAPE DRIVE I1 FMT')
        READ(IN,11) LUN
11        FORMAT(I1)
        WRITE(IOUT,20)
20        FORMAT(' ENTER MAG TAPE DEV & FILE NAME MTO:FILE1.DAT')
        CALL ASSIGN(LUN,'MTO:FILE01.DAT',-1)
        WRITE(IOUT,30)
30        FORMAT( ' ENTER NUMBER OF RECORDS IN DATA FILE -I5 FORMAT')
        READ(IN,40) NREC
40        FORMAT(I4)
        IVAR=1
        DEFINE FILE LUN(NREC,256,U,IVAR)
        READ(LUN'1) IBUF
        CALL HEADR
        CALL WDATA
        STOP
        END
      SUBROUTINE HEADR
      COMMON DATE(2),STYPE,SNUMB,LASITE,RF,RCAL,R1,R2,L,DFDTCL,
     1DTSW,DVSW,VSWN,VSWP,DUM1,IDEN    ,TV0,ALT,VERVEL,ISAT,DUM2,
     2SIG(4,8),DUM3,DUM4,DUM5,ZERO,TIME(2),NPTS,NBLKS,
     3LUN,NREC,IN,IOUT,LP
      INTEGER*2 IBUF(128),IDEN(8)
      INTEGER*2 STYPE,SNUMB
        INTEGER*4 NPTS,NBLKS
      EQUIVALENCE (DATE(1),IBUF)
      REAL LASITE,L,ISAT
C
C  WRITE FLIGHT HEADER
C
        WRITE(LP,1)
        WRITE(LP,2)
C
C  GET A RECORD
C  WRITE OUT FLIGHT INFORMATION
C
      WRITE(LP,15) DATE(1),DATE(2),STYPE,SNUMB,LASITE,RF,RCAL,
     1R1,R2,L,DFDTCL,DTSW,DVSW,VSWN,VSWP,(IDEN(I9),I9=1,8)
   15 FORMAT('0',2A4,4X,A2,5X,I3,5X,A4,1X,E8.2,1X,E8.2,3X,F4.1,6X,
     1F4.1,6X,F4.1,6X,F6.2,2X,F6.2,3X,F6.2,3X,F6.2,3X,F6.2//
     2' COMMENTS-'//' ',8A2)
C
C  WRITE OUT DATA HEADER
C
C
C  HEADER FOR EACH FLIGHT
C
    1 FORMAT('1',3X,'DATE',4X,'SENSOR',2X,'SENSOR',2X,'LAUNCH',4X,
     1'RF',5X,'RCAL',2X,'COLLECTOR',4X,'GUARD',4X,'ELECTRODE',2X,
     2'DF/DTCAL',2X,'DELTA T',2X,'DELTA V',2X,'VSWEEP',3X,'VSWEEP')
    2 FORMAT(' ',12X,'TYPE',5X,'NO',5X,'SITE',18X,'RADIUS(CM)',1X,
     1'RADIUS(CM)',1X,'LENGTH(CM)',12X,'SWEEP',4X,'SWEEP',6X,
     2'POS',5X,'NEG')
C
C  HEADER FOR VARIOUS MEASUREMENTS WITHIN FLIGHT
C
      WRITE(LP,3)
    3 FORMAT('0',3X,'TIME',6X,'ALTITUDE',2X,'VERTICAL',4X,'I',4X,
     1'ZERO VOLT',4X,'SIG(1)',1X,'SIG(1)',1X,'SIG(2)',1X,'SIG(2)',1X,
     2'SIG(3)',1X,'SIG(3)',1X,'SIG(4)',1X,'SIG(4)' )
      WRITE(LP,4)
    4 FORMAT(' ',1X,'HR MIN SEC',5X,'KM',5X,'VELOCITY',3X,'SAT',
     13X,'CROSSING',6X,'POS',4X,'NEG',4X,'POS',4X,'NEG',4X,'POS',
     14X,'NEG')
      RETURN
      END
```

```
       SUBROUTINE WDATA
       COMMON DATE(2),STYPE,SNUMB,LASITE,RF,RCAL,R1,R2,L,DFDTCL,
      1DTSW,DVSW,VSWN,VSWP,DUM1,IDEN    ,TVO,ALT,VERVEL,ISAT,DUM2,
      2SIG(4,8),DUM3,DUM4,DUM5,ZERO,TIME(2),NPTS,NBLKS,
      3LUN,NREC,IN,IOUT,LP
       INTEGER*2 IBUF(128),IDEN(8)
       INTEGER*2 STYPE,SNUMB
         INTEGER*4 NPTS,NBLKS
       EQUIVALENCE (DATE(1),IBUF)
       REAL LASITE,L,ISAT
       WRITE(LP,20) (TIME(I9),I9=1,2),ALT,VERVEL,ISAT,TVO,
      1(SIG(1,I8),I8=1,8)
    20 FORMAT(' ',2A4,2X,F6.2,4X,F7.2,2X,F6.2,3X,F6.2,6X,
      18(1X,F6.2))
       RETURN
       END
```

APPENDIX 8

MXYI.FOR TRANSFER OF DATA FROM UNIX TO LIBRARY

```
C  MXYI.FOR
C  MAIN PROGRAM TO CREATE DATA FILES FROM PREVIOUSLY REDUCED FLIGHTS
C  WHERE DATA IS ALREADY STORED IN UNIX VIRTUAL ARRAYS
C  VIRTUAL ARRAYS ARE DOUBLE PRECISION IN UNIX BASIC
C  AND ARE IN EXACTLY THE SAME FORM AS FORTRAN DIRECT ACCESS FILES
C  THIS PROGRAM STARTS WITH A SET OF X AND Y COORDINATE POINTS
C  WHICH MIGHT HAVE BEEN CREATED AS A VIRTUAL FILE UNDER UNIX OR READ
C  IN FROM A CARD READER.
C  THESE FILES ARE THEN USED TO CONSTRUCT AN OUTPUT FILE
C  WHICH IS IN THE STANDARD FORMAT OF THE DATA REDUCTION PROGRAMS
C  EACH POINT PAIR IS STORED IN ONE BLOCK OF 256 WORDS WITH DUMMY
C  DATA ADDED TO REPLACE THE USUAL TIMING INFORMATION
      COMMON DATE(2),STYPE,SNUMB,LASITE,RF,RCAL,R1,R2,L,DFDTCL,
     1DTSW,DVSW,VSWN,VSWP,DUM1,IDEN    ,TV0,ALT,VERVEL,ISAT,DUM2,
     2SIG(4,8),DUM3,DUM4,DUM5,ZERO,TIME(2),NPTS,NBLKS,
     3LUN,NREC,IN,IOUT,LP,X(301),Y(301)
        REAL*8 X,Y
        DATA TIME/4H    ,4H    /
        INTEGER*2 IBUF(128),IDEN(8)
      INTEGER*2 STYPE,SNUMB
        INTEGER*2 JBUF(128)
        INTEGER*4 NPTS,NBLKS
      EQUIVALENCE (DATE(1),IBUF)
      REAL LASITE,L,ISAT
C
C  SET COMMON BLOCK EQUIVALENT TO IBUF FOR MAG TAPE I/O
C
C


        IN=5
        IOUT=7
        LP=6
        WRITE(IOUT,10)
10        FORMAT(' ENTER LOGICAL UNIT NUMBER FOR MAG TAPE DRIVE I1 FMT')
        READ(IN,11) LUN
11        FORMAT(I1)
        WRITE(IOUT,20)
20        FORMAT(' ENTER MAG TAPE DEV & FILE NAME MT0:FILE1.DAT')
        CALL ASSIGN(LUN,'MT0:FILE01.DAT',-1)
        WRITE(IOUT,30)
30        FORMAT( ' ENTER NUMBER OF RECORDS IN DATA FILE -I5 FORMAT')
        READ(IN,40) NREC
40        FORMAT(I4)
        IVAR=1
        DEFINE FILE LUN(NREC,256,U,IVAR)
C  CREATE DUMMY SET OF TIMING DATA TO MAKE EACH RECORD
C  256 16-BIT WORDS LONG
        DO 100 I=1,128
100        JBUF(I)=0
        CALL QUESTN
C
C  BRING IN ARRAY CONTAINING SIGMA'S AND ALTITUDES AND
C  STORE IN X AND Y.  X(1) CONTAINS NUMBER OF POINTS IN DATA SET
C
C  REMEMBER THAT  X(1) IN FORTRAN IS THE EQUIVALENT OF
C
```

```
C  X(0) IN UNIX BASIC
        CALL XYIN
        TVO=0.
        VERVEL=0.
        ISAT=0.
        DUM1=0.
        DUM2=0.
        DUM3=0.
        DUM4=0.
        DUM5=0.
        ZERO=0.
        NPTS=64
        NBLKS=1
        DO 300 J=1,8
        DO 300 K=1,4
C  CONVERT DOUBLE PRECISION TO INTEGER
300        SIG(K,J)=0.
        IX=IDINT(X(1))
C  WRITE OUT ONE WAVEFORM FOR EACH SET OF POINTS
        DO 200 I=2,IX
        SIG(1,1) =SNGL(X(I))
C  CONVERT DOUBLE PRECISION TO SINGLE PRECISION
        ALT=SNGL(Y(I))
        I2=I-1
        WRITE(LUN'I2) IBUF,JBUF
200        CONTINUE
        STOP
        END
```

```
       SUBROUTINE XYIN
       COMMON DATE(2),STYPE,SNUMB,LASITE,RF,RCAL,R1,R2,L, FDTCL,
      1DTSW,DVSW,VSWN,VSWP,DUM1,IDEN    ,TVO,ALT,VERVEL,ISAT,DUM2,
      2SIG(4,8),DUM3,DUM4,DUM5,ZERO,TIME(2),NPTS,NBLKS,
      3LUN,NREC,IN,IOUT,LP,X(301),Y(301)
       INTEGER*2 IBUF(128),IDEN(8)
       INTEGER*2 STYPE,SNUMB
         INTEGER*4 NPTS,NBLKS
       EQUIVALENCE (DATE(1),IBUF), (X(1),Z(1))
         REAL*8 X,Y,BUF(64), Z(640)
         REAL LASITE,L,ISAT
C
C  READ FROM DIRECT ACCESS FILE X=SIGMA AND Y= ALTITUDE DATA
C
        WRITE(IOUT,20)
20        FORMAT(' ENTER DEVICE AND FILE DESIGNATION WHERE X,Y DATA IS'//)
        CALL ASSIGN(2,'DK0:XY.DAT',-1)
        IVAR=1
        DEFINE FILE 2(10,256,U,IVAR)
C
C  INPUT AND UNPACK READ BUFFER BUF
C
        DO 100 I=1,10
        READ(2'I) BUF
        J1=64*(I-1)
        DO 200 J=1,64
        J2=J1+J
200        Z(J2)=BUF(J)
100        CONTINUE
        RETURN
        END
```

APPENDIX 9

MXYO.FOR TRANSFER DATA FROM SYSTEM LIBRARY TO UNIX

```
C  MXYO.FOR
C  MAIN PROGRAM TO SEARCH THROUGH MAG TAPE OR OTHER DIRECT ACCESS
C  DATA SETS IN STANDARD FORMAT AND EXTRACT TWO ARRAYS X AND Y WHICH
C  CONTAIN A DESIRED SIGMA AND CORRESPONDING ALTITUDE  DATA.
C  SUBROUTINE XYOUT THEN DUMPS EXTRACTED DATA POINTS ONTO ANOTHER
C  DIRECT ACCESS DEVICE IN A FORM SUITABLE FOR PLOTTING UNDER
C  THE UNIX PLOT ROUTINE.
      COMMON DATE(2),STYPE,SNUMB,LASITE,RF,RCAL,R1,R2,L,DFDTCL,
     1DTSW,DVSW,VSWN,VSWP,DUM1,IDEN    ,TVO,ALT,VERVEL,ISAT,DUM2,
     2SIG(4,8),DUM3,DUM4,DUM5,ZERO,TIME(2),NPTS,NBLKS,
     3LUN,NREC,IN,IOUT,LP,X(301),Y(301)
        REAL*8 X,YBUF(64),Z(640)
        DATA TIME/4H    ,4H    /
        INTEGER*2 IBUF(128),IDEN(8)
      INTEGER*2 STYPE,SNUMB
        INTEGER*2 JBUF(128)
        INTEGER*4 NPTS,NBLKS
      EQUIVALENCE (DATE(1),IBUF), (X(1),Z(1))
      REAL LASITE,L,ISAT
        IN=5
        IOUT=7
        LP=6
        WRITE(IOUT,10)
10        FORMAT(' ENTER LOGICAL UNIT NUMBER FOR MAG TAPE DRIVE I1 FMT')
        READ(IN,11) LUN
11        FORMAT(I1)
        WRITE(IOUT,20)
20        FORMAT(' ENTER MAG TAPE DEV & FILE NAME MTO:FILE1.DAT')
        CALL ASSIGN(LUN,'MTO:FILE01.DAT',-1)
        WRITE(IOUT,30)
30        FORMAT( ' ENTER NUMBER OF RECORDS IN DATA FILE -I5 FORMAT')
        READ(IN,40) NREC
40        FORMAT(I4)
        IVAR=1
        DEFINE FILE LUN(NREC,256,U,IVAR)
        WRITE(IOUT,150)
150         FORMAT(' WHICH SIGMA DO YOU WANT OUTPUT?'/
     1' ENTER A VALUE BETWEEN 1 AND 8 FOR THE SPECIES DESIRED')
        READ(IN,151)  NSIG
151         FORMAT(I1)
C  GET FIRST BLOCK OF A WAVEFORM DATA SET
C  VARIABLE NBLKS WILL TELL HOW MANY BLOCKS OF
C  DATA GO WITH THIS WAVEFORM
C  ALT WILL CONTAIN THE VALUE OF THE Y COORDINATE
C  SIG(1,NSIG) WHERE NSIG=1,8 WILL CONTAIN THE CONDUCTIVITY OF
C  THE NSIG TH SPECIES
C  SET RECORD COUNTER TO 0 INITIALLY
        IRECNT =0
        IXYCNT =1
500         INEXT =IRECNT+1
        READ(LUN'INEXT) IBUF,JBUF
        IXYCNT =IXYCNT+1
C  CONVERT ALT AND SIG TO DOUBLE PRECISION AND STORE AWAY
        X(IXYCNT)=DBLE(SIG(I,NSIG))
        Y(IXYCNT)=DBLE(ALT)
C  FLUSH OFF REST OF RECORDS ASSOCIATED WITH THIS
C  WAVEFORM IRECNT BY NUMBER OF BLOCKS ASSOCIATED WITH THIS WAVEFORM
        IRECNT=IRECNT+NBLKS
      IF(IRECNT.LT.NREC) GO TO 500
C  ALL DONE SO PUT NUMBER OF WAVEFORMS FOUND IN SEARCH IN
C  X(1) AND CALL SUBROUTINE TO PUT OUT PLOTTING  FILE.
      X(1) =DBLE(FLOAT(IXYCNT))
      Y(1)=0.
      CALL XYOUT
      STOP
      END
      SUBROUTINE XYOUT
C  SUBROUTINE TO TAKE DATA STORED IN X AND Y ARRAYS AND DUMP
C  IT OUT IN A FORM WHICH IS SUITABLE FOR PLOTTING USING THE UNIX
C  %PLOT ROUTINE.
      COMMON DATE(2),STYPE,SNUMB,LASITE,RF,RCAL,R1,R2,L,DFDTCL,
     1DTSW,DVSW,VSWN,VSWP,DUM1,IDEN    ,TV0,ALT,VERVEL,ISAT,DUM2,
     2SIG(4,8),DUM3,DUM4,DUM5,ZERO,TIME(2),NPTS,NBLKS,
     3LUN,NREC,IN,IOUT,LP,X(301),Y(301)
      INTEGER*2 IBUF(128),IDEN(8)
      INTEGER*2 STYPE,SNUMB
        INTEGER*4 NPTS,NBLKS
      EQUIVALENCE (DATE(1),IBUF), (X(1),Z(1))
        REAL*8 X,Y,BUF(64), Z(640)
        REAL LASITE,L,ISAT
        WRITE(IOUT,20)
20        FORMAT(' ENTER DEVICE AND FILE DESIGNATION WHERE X AND Y '/
     1' DATA IS TO BE STORED'//)
        CALL ASSIGN(2,'DK0:XY.DAT',-1)
        IVAR=1
        DEFINE FILE 2(10,256,U,IVAR)
C  BACK FILL ARRAY TO BRING UP TO 2560 WORDS FROM 2408 IN X AND Y
C  THIS MAKES EXACTLY TEN 256-WORD BLOCKS
        DO 100 I= 603,640
100       Z(I)=0.
        DO 300 I= 1,10
        J1=64*(I-1)
        DO 200 J= 1,64
        J2=J1+J
200        BUF(J)=Z(J2)
C  WRITE OUT 256-WORD BLOCK TO PLOTTING FILE
300        WRITE(2'I) BUF
        RETURN
        END
```

APPENDIX 10

MIWS.FOR CREATES DUMMY DATA FILES OR DEBUGGING

```
C  MAIN PROGRAM TO CREATE DATA FILES FROM PREVIOUSLY REDUCED FLIGHTS
      COMMON DATE(2),STYPE,SNUMB,LASITE,RF,RCAL,R1,R2,L,DFDTCL,
     1DTSW,DVSW,VSWN,VSWP,DUM1,IDEN    ,TVO,ALT,VERVEL,ISAT,DUM2,
     2SIG(4,8),DUM3,DUM4,DUM5,ZERO,TIME(2),NPTS,NBLKS,
     3LUN,NREC,IN,IOUT,LP
        INTEGER*2 IBUF(128),IDEN(8)
      INTEGER*2 STYPE,SNUMB
        INTEGER*2 JBUF(128)
        INTEGER*4 NPTS,NBLKS
      EQUIVALENCE (DATE(1),IBUF)
      REAL LASITE,L,ISAT
C
C  SET COMMON BLOCK EQUIVALENT TO IBUF FOR MAG TAPE I/O
C
C

        IN=5
        IOUT=7
        LP=6
        WRITE(IOUT,10)
10        FORMAT(' ENTER LOGICAL UNIT NUMBER FOR MAG TAPE DRIVE I1 FMT')
        READ(IN,11) LUN
11        FORMAT(I1)
        WRITE(IOUT,20)
20        FORMAT(' ENTER MAG TAPE DEV & FILE NAME MT0:FILE1.DAT')
        CALL ASSIGN(LUN,'MT0:FILE01.DAT',-1)
        WRITE(IOUT,30)
30        FORMAT( ' ENTER NUMBER OF RECORDS IN DATA FILE -I5 FORMAT')
        READ(IN,40) NREC
40        FORMAT(I4)
        IVAR=1
        DEFINE FILE LUN(NREC,256,U,IVAR)
        DO 100 I=1,128
100        JBUF(I)=0
        CALL QUESTN
200        K=K+2
        L=K+1
        CALL DDATA
        WRITE(LUN'K) IBUF
        WRITE(LUN'L) JBUF
        WRITE(IOUT,50)
50        FORMAT(' ANY MORE WAVEFORMS?  1=YES,0=NO')
        READ(IN,60) IQ
60        FORMAT(I1)
        IF(IQ.EQ.1) GO TO 200
        STOP
        END
```

AD-A093 649 PENNSYLVANIA STATE UNIV UNIVERSITY PARK IONOSPHERE R--ETC F/G 12/1
IMPROVED DATA REDUCTION AND ANALYSIS SYSTEM FOR BLUNT PROBE AND--ETC(U)
OCT 80 D C SCHRODER DAAD29-76-D-0100
UNCLASSIFIED ERADCOM/ASL-CR-80-0100-3 NL

2 OF 2
AD A
093649

END
DATE
FILMED
2-81
DTIC

```
      SUBROUTINE QUESTN
      COMMON DATE(2),STYPE,SNUMB,LASITE,RF,RCAL,R1,R2,L,DFDTCL,
     1DTSW,DVSW,VSWN,VSWP,DUM1,IDEN    ,TVO,ALT,VERVEL,ISAT,DUM2,
     2SIG(4,8),DUM3,DUM4,DUM5,ZERO,TIME(2),NPTS,NBLKS,
     3LUN,NREC,IN,IOUT,LP
      INTEGER*2 IBUF(64),IDEN(8)
      INTEGER*2 STYPE,SNUMB
        INTEGER*4 NPTS,NBLKS
      REAL LASITE,L,ISAT
      EQUIVALENCE (DATE(1),IBUF)
C
C  SUBROUTINE TO FILL INFORMATION BLOCK FOR HEADER OF EACH  RECORD
C
      WRITE(IOUT, 1)
    1 FORMAT(' PROGRAM TO INPUT HEADER INFORMATION FOR EACH FLIGHT')
C
C  GET DATE
C
      WRITE(IOUT, 2)
    2 FORMAT(' DATE--DDMMYY')
      READ(IN,3) DATE(1),DATE(2)
    3 FORMAT(2A4)
C
C  GET SENSOR TYPE
C
      WRITE(IOUT, 4)
    4 FORMAT(' SENSOR TYPE BP FOR BLUNT PROBE-- GC FOR GERDIEN')
      READ(IN,5)  STYPE
    5 FORMAT(A2)
C
C  GET SENSOR NUMBER
C
      WRITE(IOUT, 6)
    6 FORMAT(' SENSOR NUMBER-- I3')
      READ(IN,7) SNUMB
    7 FORMAT(I3)
C
C  GET LAUNCH SITE
C
      WRITE(IOUT,8)
    8 FORMAT(' LAUNCH SITE -- XXXX')
      READ(IN,9) LASITE
    9 FORMAT(A4)
C
C  GET FEEDBACK RESISTOR
C
      WRITE(IOUT,10)
   10 FORMAT(' FEEDBACK RESISTOR SIZE -RF- F7.2')
      READ(IN,11) RF
   11 FORMAT(E7.2)
C
C  GET CALIBRATION RESISTOR
C
      WRITE(IOUT,12)
   12 FORMAT(' CALIBRATION RESISTOR SIZE -RCAL-F7.2')
      READ(IN,13) RCAL
   13 FORMAT(E7.2)
C
C  COLLECTOR RADIUS
C
      WRITE(IOUT,14)
   14 FORMAT(' COLLECTOR RADIUS-R FOR BP--RI FOR GC  F4.1')
      READ(IN,15) R1
   15 FORMAT(F4.1)
C
C  GUARD RADIUS
C
      WRITE(IOUT,16)
   16 FORMAT(' GUARD OR OUTER PLATE RADIUS- R FOR BP-RO FOR GC  F4.1')
      READ(IN,17) R2
   17 FORMAT(F4.1)
C
C  ELECTRODE LENGTH
C
      WRITE(IOUT,18)
   18 FORMAT(' ELECTRODE LENGTH IN CM - ENTER 0 FOR BP  F4.1')
      READ(IN,19) L
   19 FORMAT(F4.1)
C
C  DF/DT CAL
C
      WRITE(IOUT,20)
   20 FORMAT(' DF/DT CAL  F6.2')
      READ(IN,21) DFDTCL
   21 FORMAT(F6.2)
C
C  DELTA TIME SWEEP
C
      WRITE(IOUT,22)
   22 FORMAT(' DELTA TIME SWEEP  F6.2')
      READ(IN,23) DTSW
   23 FORMAT(F6.2)
C
C  DELTA VOLTAGE SWEEP
C
      WRITE(IOUT,24)
   24 FORMAT(' DELTA VOLTAGE SWEEP  F6.2')
       READ(IN,25) DVSW
   25 FORMAT(F6.2)
C
C  NEGATIVE MAXIMUM VALUE OF VOLTAGE SWEEP
C
      WRITE(IOUT,26)
   26 FORMAT(' NEGATIVE MAXIMUM VALUE OF SWEEP VOLTAGE  F6.2')
      READ(IN,27) VSWN
   27 FORMAT(F6.2)
C
C  POSITIVE MAXIMUM VALUE OF VOLTAGE SWEEP
C
```

```
      WRITE(IOUT,28)
   28 FORMAT(' POSITIVE MAXIMUM VALUE OF SWEEP VOLTAGE  F6.2')
      READ(IN,29) VSWP
   29 FORMAT(F6.2)
C
C  SPECIAL IDENTIFICATION OR INFORMATION
      WRITE(IOUT,30)
   30 FORMAT(' SPECIAL IDENTIFICATION OR INFORMATION  8A2')
      READ(IN,31) (IDEN(I9),I9=1,8)
   31 FORMAT(8A2)
C
C  END OF QUESTION AND ANSWER SUBROUTINE
      WRITE(IOUT,32)
   32 FORMAT(' END OF QUESTION AND ANSWER SECTION OF PROGRAM')
      RETURN
      END
      SUBROUTINE DDATA
      COMMON DATE(2),STYPE,SNUMB,LASITE,RF,RCAL,R1,R2,L,DFDTCL,
     1DTSW,DVSW,VSWN,VSWP,DUM1,IDEN    ,TV0,ALT,VERVEL,ISAT,DUM2,
     2SIG(4,8),DUM3,DUM4,DUM5,ZERO,TIME(2),NPTS,NBLKS,
     3LUN,NREC,IN,IOUT,LP
      INTEGER*2 IBUF(128),IDEN(8)
      INTEGER*2 STYPE,SNUMB
        INTEGER*4 NPTS,NBLKS
      EQUIVALENCE (DATE(1),IBUF)
C
      REAL LASITE,L,ISAT
C  SUBROUTINE TO CREATE DATA SET FOR PREVIOUSLY REDUCED DATA
C
        WRITE(IOUT,10)
10        FORMAT(' TIME FROM SYNC PULSE TO ZERO POTENTIAL CROSSING POINT')
        READ(IN,11) TV0
11        FORMAT(F10.5)
        WRITE(IOUT,20)
20        FORMAT(' ALTITUDE IN KM-- F6.2')
        READ(IN,21) ALT
21        FORMAT(F6.2)
        WRITE(IOUT,30)
30        FORMAT(' VERTICAL VELOCITY -- F6.2')
        READ(IN,21) VERTICAL
        WRITE(IOUT,40)
40        FORMAT(' SATURATION CURRENT -- F6.2')
        READ(IN,21) ISAT
C
C  FILL DUMMY LOCATIONS
C
      DUM1=0.
      DUM2=0.
      DUM3=0.
      DUM4=0.
      DUM5=0.
C
C  ZERO POINTER
C
      ZERO= 0.0
C
C  NOW DEFINE TIMES
        WRITE(IOUT,80)
80        FORMAT(' ENTER TIME DDHRMMSS  2A4')
        READ(IN,81) TIME(1),TIME(2)
81        FORMAT(2A4)
        NPTS = 64
        NBLKS=1
C
C
        WRITE(IOUT,50)
50        FORMAT(' ENTER VALUES FOR SIGMAS')
        DO 200 J=1,8
        WRITE(IOUT,60) J
60        FORMAT(' SPECIES NUMBER IS ',I5)
        WRITE(IOUT,61)
61        FORMAT(' SIGMA=? --F6.2')
```

```
        READ(IN,21) SIG(1,J)
        WRITE(IOUT,62)
62        FORMAT(' POTENTIAL FOR ABOVE SIGMA=? -- F6.2')
        READ(IN,21) SIG(4,J)
        SIG(2,J)=0.
        SIG(3,J)=0.
200        CONTINUE
        RETURN
        END
```

APPENDIX 11

DOCUMENTATION OF UNIX UTILITY PROGRAMS

USED WITH DATA REDUCTION SYSTEM

```
5 rem program name unixxyp.bas
10 rem unix program to input from 11/45 terminal previously
20 rem reduced data sets
30 rem x(0) contains number of (sigma, altitude) pairs of points
40 rem rtpip is a unix program that can be used to convert unix virtual
50 rem files to rt-11 files or conversely
60 input "what file name do you want to store the data in",f$
100 open f$ as file #4%
110 dim #4%, x(300),y(300)
115 print "to update a file enter u"
116 print "to enter a new set of data just hit carriage return"
117 input q$
118 if q$="u" then 260
120 c%=o
130 print "enter altitude and corresponding sigma."
140 print "Terminate with a negative altitude."
150 input a,s
160 if a0 then 210
170 c%=c%+1
180 x(c%)=s
190 y(c%)=a
200 go to 150
210 x(0)=c%
220 print "End of data has been detected."
230 print "Do you wish to go back and correct any values."
240 input "Answer with y or n", q$
250 if q$="y" then 260 else 340
260 rem start here to update
270 print "Type a y to update a data pair else push cr"
275 print "altitude,   sigma"
280 for I=1 to x(0)
290 print y(i), x(i)
300 input q$
310 if q$  "y" go to 330
320 input "enter altitude and sigma", y(I), x(I)
330 next I
340 close #4%
350 end
```

```
1 rem  xylist.bas
2 remthis program prints out altitudes and sigmas contained in
3 rem virtual array ready for plotting
5 input "what is name of virtual file containing data",f$
10 open f$ as file #4%
20 dim #4%, x(300),y(300)
25 print "altitude       sigma"
26 print
30 for i = 1 to x(0)
40 print y(i),x(i)
50 next i
60 end
```

NAME
    plot ---- plot points stored in a Basic-Plus virtual array file

SYNOPSIS
    plot

DESCRIPTION
This program will plot points stored in a Basic-Plus virtual array file of the form described int eh following Basic dimension statement:

```
Dim#n, X(300), Y(300)
```

where `n' is the logical unit number of the virtual file. `X' is the array containing the absicca values and `Y' is an array containing the corresponding ordinate values. The virtual arrays must be dimensioned exactly as shown.

The data in the arrays is assumed to begin at element 1 (instead of element 0). Element 0 of the first array (X(0)) is interpreted as the number of points to be plotted (300 maximum). It is important that the user remembers to set this element.

For `C' users who wish to take advantage of this program, the data may be stored on disk with a `write' statement using a buffer with the following structure:

```
struct {
 double X[301];
 double Y[301];
} buffer;
```

The program `plot' asks a number of questions from the user to determine such information as the scaling, title name, axis divisions, etc. The program uses the routine `plot2d' (see PLOT2D (IX)) to perform the plotting. Replies typed by the user are translated to actual parameters of `plot2d'.

Most of the questions asked by `plot' are self-explanatory; however, for the sake of completeness, an explanation of each question will be given. Any reply to a yes or no question will be interpreted as `no' if the first character typed does not begin with `y' or `Y'.

Virtual array file?
    The user responds with the name of the virtual file on which his data is located.

Log X?
    If the X axis is to be logarithmic, then reply yes.

Minimum value of X?
Maximum value of X?

The user is to reply with numerical values. The horizontal scaling of the plot is determined by these replies.

Log Y?

If the Y axis is to be logarithmic, then reply yes.

Minimum value of Y?
Maximum value of Y?

Numerical values are expected. This information is used to determine the vertical scaling.

Do you wish axes to be marked?

If the user replies negativelythe axes willl not be marked and the following 4 questions will be skipped.

Distance between each tick mark on X axis?

The numerical value recieved is interpreted as the length of the intervals at which major ticks marks are to be drawn. (see PLOT2D(IX)). Each major tick mark will be accompanied on the plot by its numerical value along the axis.

Number of minor tick marks per scale division on X axis?

An integer is expected. Between each amjor tick mark may appear an arbitrary number of minor scale divisions (such as the 1/8th inch divisions on a foot ruler). The non-negative integer recieved from the user will be interpreted as the number of such minor scale divisions to appear at each major interval.

Distance between each tick mark on Y axis?

Number of minor tick marks per scale division on Y axis? Analogous to two preceding questions.

Grid?

If the user answers with yes, then a rectangular grid will be drawn over the plot. Otherwise, axes will be drawn with small dashes at the major and minor scale divisions.

Axes at margin?

By default the axes of a plot will intersect at the point (0,0) if it is visible. If the origin is not within the range of the screen, the axes will intersect at the edge nearest to the origin. Often, it is desired that the axes intersect in the lower left hand corner regardless of where the actual origin is. This is especially desirable if the user wishes to superimpose a curve which requires a different vertical scaling. A reply of yes will place the intersection of the axes in the lower left corner.

Type of curve (smooth, dashed, point)?

The user has an option of a smooth curve, dashed curve, or point plot. (S)he need only type 'carriage return', 'd', or 'p' for smooth, dashed, and point, respectively.

Name of horizontal axis?
Name of vertical axis?

The user types in the name of each axes. If no axis labels are desired then the user should reply with a carriage return. Special characters (greek) are supported (see text IX).

Do you wish the plot to be titled?

Yes or no.

Type in title line by line; terminate with \\

Special (greek) characters are supported (see text IX).

Output file ?

If either a carriage return or the characters 'gr' are typed, then the plot will be sent directly to the Tektronix 4010 graphics terminal. If the characters 'pl' are typed, then the plot will be sent directly to the HP plotter. Any other characters typed will be interpreted as a file name to where the pen moving codes will be stored. A plot stored in this manner can be retrieved on the graphics terminal by typing the command:

```
cat filename ^ tek
```

or on the HP plotter with the command:

```
cat filename ^ hp
```

Superimpose another plot?

A user may superimpose the points stored in another virtual array file on the existing plot. If required the second plot may have a different vertical scaling (as in the case of magnitude-phase plots). The new vertical axis of the different scale will appear at the far right. Any number of superimpositions are allowed; however, no more than two different vertical scalings are allowed.

When a user is in the process of debugging his program that produces the virtual array file, it can be tedious answering all of the above questions repeatedly in order to test the data. To over come this difficulty, the user could keep an account of his answers and put them in a text file (each response seperated by a carriage return). Then the user could view his plot by simply typing:

```
plot <text,file > /dev/null
```

The standard output is redirected to the null device to the question prompts from appearing on the user's terminal.

Example
The following Basic Plus program segment will plot a circle.

```
1000 OPEN "plot.dat" AS FILE 1
1100 DIM#1, X(300), Y(300)
1200 X(0) = 100    !100 POINTS TO BE PLOTTED
1300 FOR I% = 1% TO 100%
1400              T = (I% - 1.) * .0628
1500              X(I%) = COS(T)
1600              Y(I%) = SIN(T)
1700 NEXT I%
32767 END
```

Dialogue with `plot' takes place as follows:

```
Virtual array file name? plot.dat
Log X?
Minimum value of X? -1
Maximum value of X? 1
Log Y?
Minimum value of Y? -1
Maximum value of Y? 1
Do you wish axes to be marked? yes
Distance between each tick mark on X axis? .2
Number of minor tick marks per scale division on X axis? 3
Distance between each tick mark on Y axis? .2
Number of minor tick marks per scale division on Y axis? 1
Grid?
Axes at margin?
Type of curve (smooth, dashed, point)?
Name of horizontal axis? X Axis
Name of vertical axis? Y Axis
Do you widh plot to be titled? yes
Type in title line by line; terminate with `\\'
This is a Circle
July 17, 1980\\

Output file? plot.out
Superimpose another plot? no
```

In order to view the plot on the Tektronix Graphics Terminal, the following is typed:

```
cat plot.out ^ tek
```

To get a hard copy on the HP plotter, the following command is typed:

```
cat plot.out ^ hp
```

SEE ALSO
plot_library(IX), tekhp(IX), plot2d(IX)

BUGS
It is difficult to change pens on the HP plotter when super imposing plots. The first plot halts before completion until the user responds to the question of superimposing addition- al plots.

DIALOG REQUIRED FOR PLOT LIKE FIGURE 6

```
% plot
Virtual array file name? PLOT.DAT
log X? y
Minimum value of X? 1.E-13
Maximum value of X? 1.E-9
log Y? n
Minimum value of Y? 30.
Maximum value of Y? 80.
Do you wish axes to be marked? y
Distance between each tick mark on Y axis? 10.
Number of minor tick marks per scale division on Y axis? 1
Grid? n
Axes at margin?
Type of curve (smooth, dashed, point)? p
Character to appear at each point? +
Name of horizontal axis? sigma (mho/cm)
Name of vertical axis? z (km)
Do you wish plot to be titled? y
Type in title line by line; terminate with '\\'
BLUNT PROBE #8 HAND REDUCED
JUNE 15, 77
1720 AST\\

Output file? pl
Superimpose another plot?
%
```

NAME
rtpip - manipulate RT-11 files from UNIX

SYNOPSIS
rtpip key [file ...]

DESCRIPTION
key consists of two and only two characters; file is either a UNIX filename or an RT-11 filename. RT-11 filenames must be preceded immediately by a '+' and UNIX filenames must have standard shell formats. The meanings of the key characters are:

The first character is the function to be performed:

r copy the file(s) to the RT-11 device.

x copy the file(s) from the RT-11 device.

l list the directory of the RT-11 device on the standard output. If no filename is given, the entire directory will be listed.

e same as l, but list only filenames with extension as given by the filename argument(s) (+EXT or +EXT.).

n same as l, but list only filenames with name-part as given by the filename argument(s) (+.FILNAM or +FILNAM).

f list, on the standard output, the sizes of all empty entries on the RT-11 device.

d delete the RT-11 file(s).

The second character identifies the RT-11 device:

r RK1

0 tap0

...

7 tap7

SEE ALSO
rtmount(II), rtumount(II), rtopen(II), rtclose(II), rtread(II), rtwrite(II), rtdelete(II).

AUTHOR
T. Forgacs, Informatics Dept. Nymegen Univ. the Netherlands.

DIAGNOSTICS
If appropriate, rtpip tries to match pairs of UNIX and RT-11 filenames in the sequence in which they occur in the file argument. If there is a UNIX or RT-11 filename without corresponding filename from the other side, the same name is taken. If the UNIX name contains slashes, all characters up

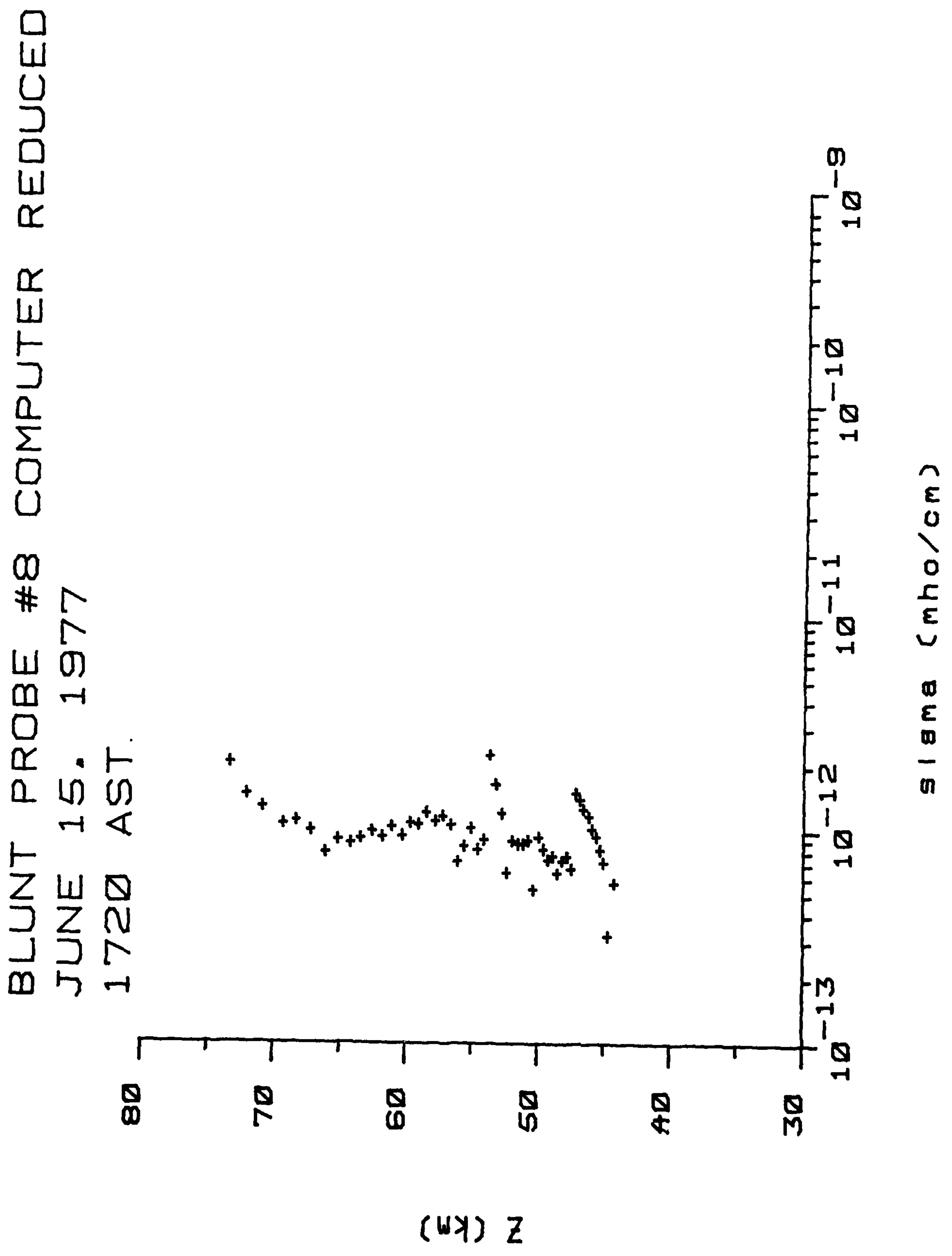
to and including the last slash are disregarded, and the result is taken as the RT-11 filename.

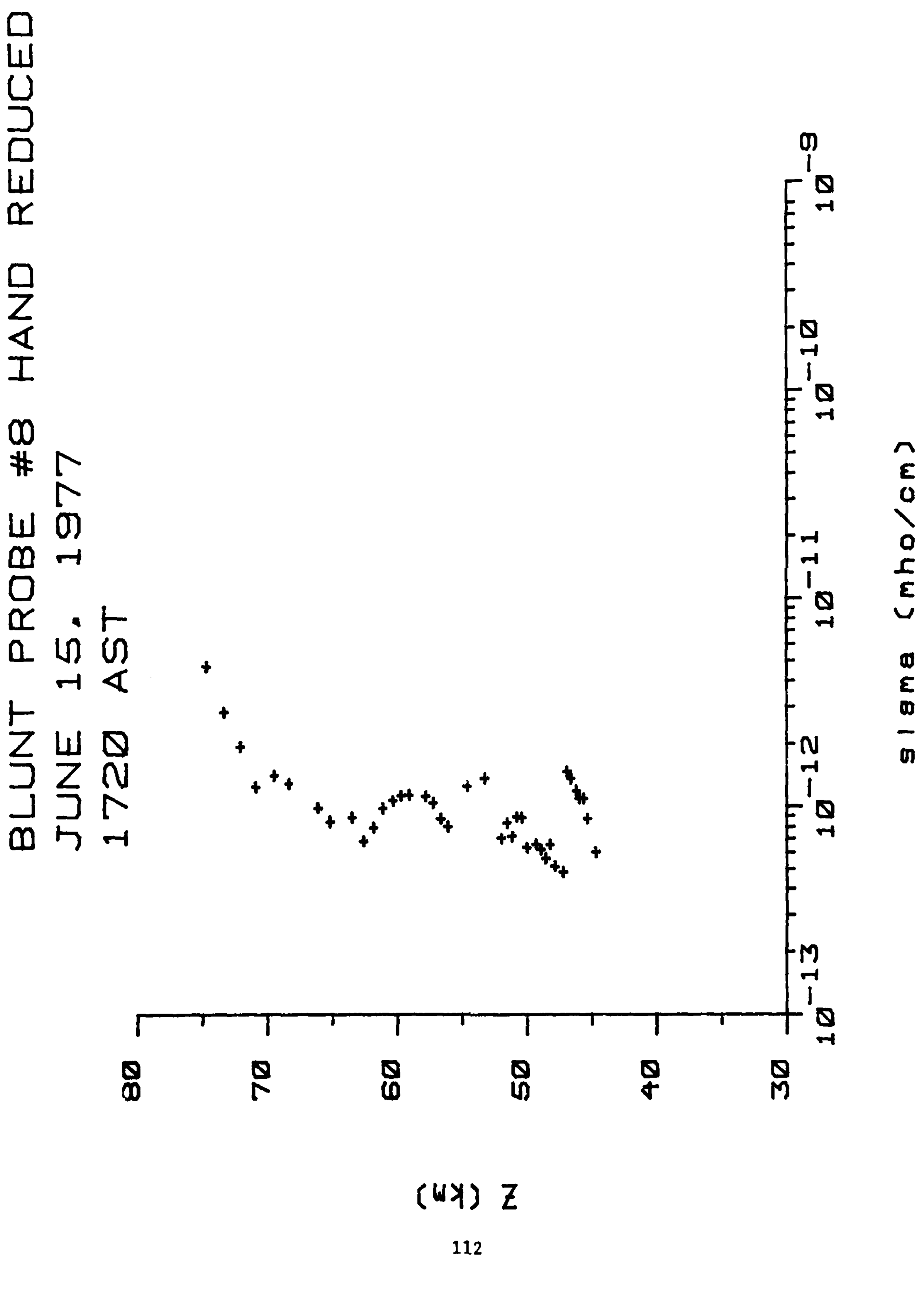
BUGS

APPENDIX 12

PLOTS RESULTING FROM HAND AND COMPUTER

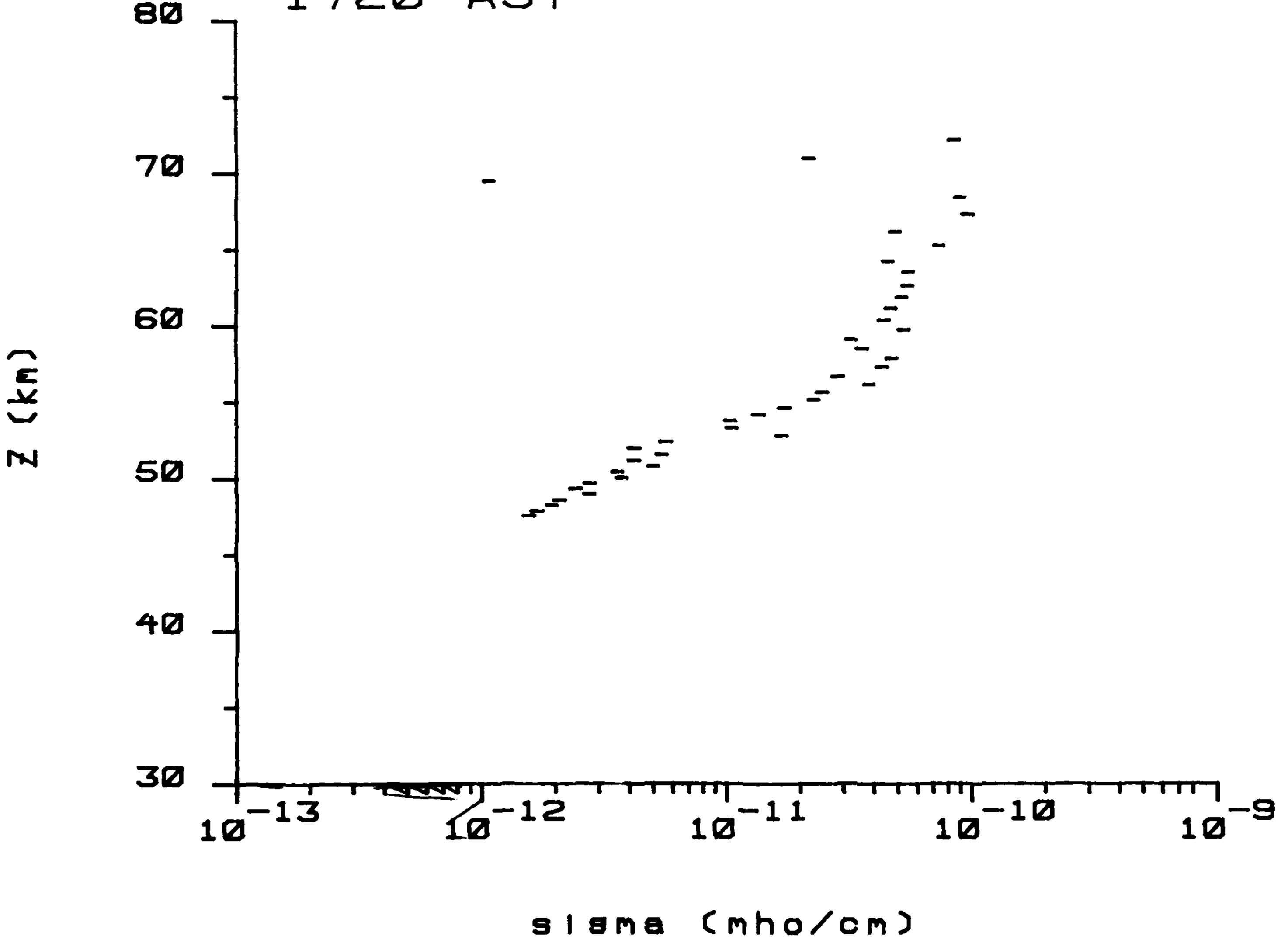
REDUCTION OF THE SAME FLIGHT



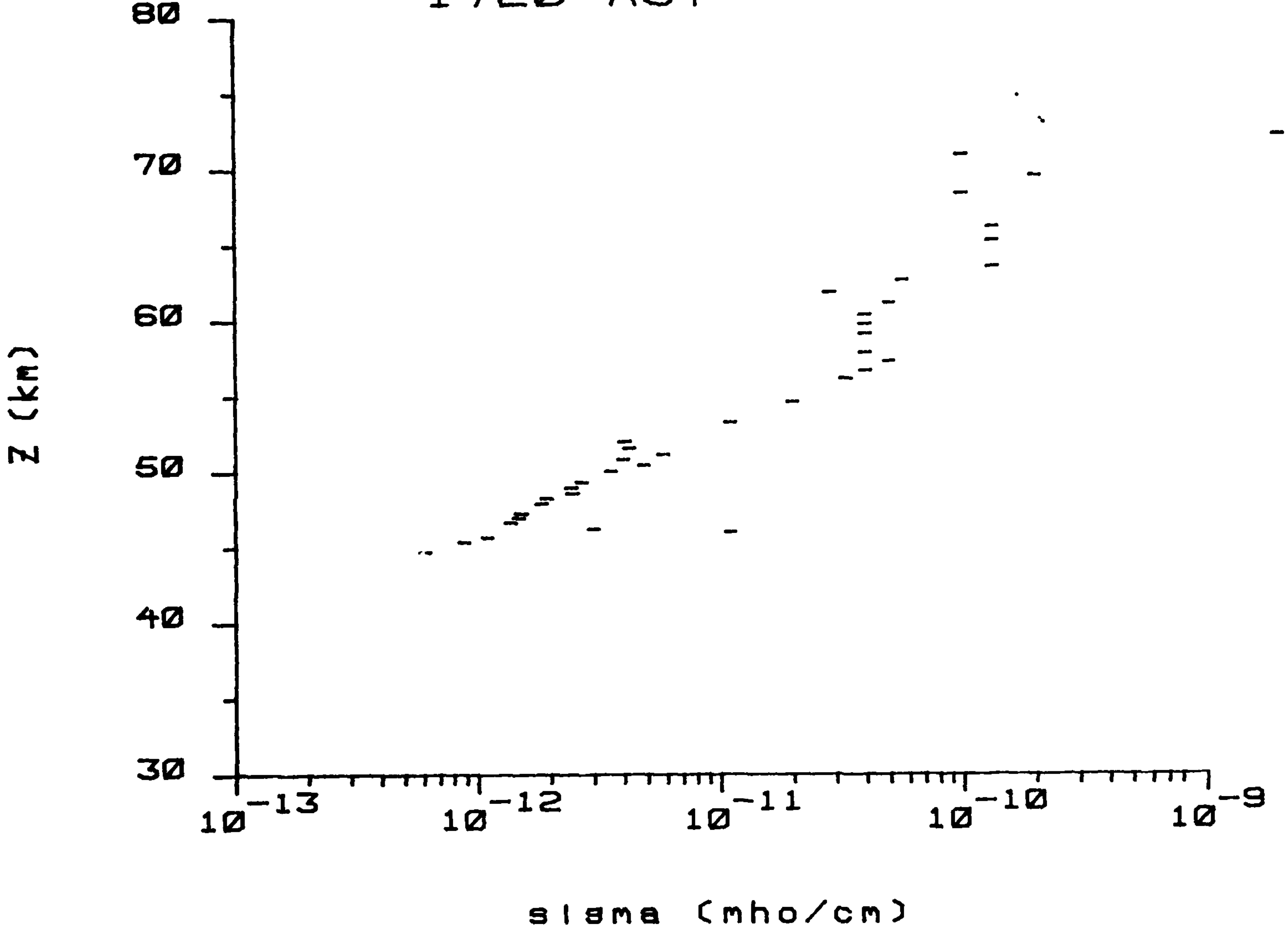


BLUNT PROBE #8 HAND REDUCED
JUNE 15, 1977
1720 AST
Z (km)
80
70
60
50
40
30
10^{-13}
10^{-12}
10^{-11}
10^{-10}
10^{-9}
sigma (mho/cm)

BLUNT PROBE #8 COMPUTER REDUCED
JUNE 15, 1977
1720 AST



BLUNT PRØBE #8 HAND REDUCED
JUNE 15. 1977
172Ø AST



```
run xylist
what is name of virtual file containing data              ? sigcomp.dat
altitude         sigma

 73.38           .2098E-11
 72.16           .1489E-11
 70.95           .13127E-11
 69.35           .1092E-11
 68.39           .1129E-11
 67.29           .1023E-11
 66.15           .807E-12
 65.24           .9258E-12
 64.24           .8907E-12
 63.52           .9383E-12
 62.64           .1015E-11
 61.88           .9565E-12
 61.15           .1071E-11
 60.39           .9676E-12
 59.75           .1113E-11
 59.13           .1091E-11
 58.53           .1246E-11
 57.87           .1134E-11
 57.3            .1196E-11
 56.69           .1095E-11
 56.17           .7326E-12
 55.67           .866E-12
 55.16           .105E-11
 54.65           .8381E-12
 54.21           .9301E-12
 53.76           .2308E-11
 53.31           .1683E-11
 52.84           .1231E-11
 52.44           .6495E-12
 52.04           .9157E-12
 51.6            .8902E-12
 51.22           .8857E-12
 50.85           .9192E-12
 50.44           .5434E-12
 50.08           .9583E-12
 49.7            .837E-12
 49.36           .7472E-12
 48.99           .7806E-12
 48.63           .6448E-12
 48.27           .7436E-12
 47.91           .7749E-12
 47.6            .6788E-12
 47.26           .1537E-11
 46.95           .1429E-11
 46.67           .1292E-11
 46.24           .1197E-11
 46.02           .1048E-11
 45.7            .9602E-12
 45.38           .8334E-12
 45.15           .7275E-12
 44.75           .3291E-12
 44.32           .5797E-12

Ready
```

```
  basic
Revised 02.11.76

Ready

run xylist
what is name of virtual file containing data              ? sigphand.dat
altitude        sigma

 74.74          .48033E-11
 73.38          .2882E-11
 72.16          .19778E-11
 70.95          .12688E-11
 69.53          .1441E-11
 68.39          .13186E-11
 66.15          .1008E-11
 65.24          .8621E-12
 63.52          .90466E-12
 62.64          .69565E-12
 61.88          .8102E-12
 61.15          .99871E-12
 60.39          .10905E-11
 59.75          .11528E-11
 59.13          .11594E-11
 57.87          .11462E-11
 57.3           .10618E-11
 56.69          .89265E-12
 56.17          .82008E-12
 54.65          .12768E-11
 53.31          .1401E-11
 52.04          .7205E-12
 51.6           .85483E-12
 51.22          .73627E-12
 50.85          .91285E-12
 50.44          .90466E-12
 50.08          .64868E-12
 49.36          .67698E-12
 48.99          .6344E-12
 48.63          .57475E-12
 48.27          .67246E-12
 47.91          .52811E-12
 47.26          .49569E-12
 46.95          .15055E-11
 46.67          .13913E-11
 46.24          .12153E-11
 46.02          .11146E-11
 45.7           .11208E-11
 45.38          .89662E-12
 44.75          .61694E-12

Ready

bye
%
```

```
run xylist
what is name of virtual file containing data              ? sigecomp.dat
altitude        sigma

 74.74          .3369E-10
 73.38          .8621E-10
 72.16          .8687E-10
 70.95          .221E-10
 69.53          .1092E-11
 68.39          .9172E-10
 67.29          .988E-10
 66.15          .4979E-10
 65.24          .752E-10
 64.24          .4645E-10
 63.52          .5643E-10
 62.64          .5578E-10
 61.88          .5281E-10
 61.15          .4753E-10
 60.39          .4506E-10
 59.75          .5436E-10
 59.13          .3317E-10
 58.53          .367E-10
 57.87          .4815E-10
 57.3           .4407E-10
 56.69          .2909E-10
 56.17          .3891E-10
 55.67          .2506E-10
 55.16          .2313E-10
 54.65          .1762E-10
 54.21          .1379E-10
 53.76          .1055E-10
 53.31          .1074E-10
 52.84          .1713E-10
 52.44          .5769E-11
 52.04          .4279E-11
 51.6           .5554E-11
 51.22          .4284E-11
 50.85          .5121E-11
 50.44          .365E-11
 50.08          .3817E-11
 49.7           .2823E-11
 49.36          .2472E-11
 48.99          .2797E-11
 48.63          .212E-11
 48.27          .1969E-11
 47.91          .1721E-11
 47.6           .1591E-11

Ready
```

```
  basic
Revised 02.11.76

Ready

run xylist
what is name of virtual file containing data          ? sigehand.dat
altitude          sigma

 74.74            .16812E-9
 73.38            .20174E-9
 72.16            .2017E-8
 70.95            .10087E-9
 69.53            .20174E-9
 68.39            .10087E-9
 66.15            .13449E-9
 65.24            .13449E-9
 63.52            .13449E-9
 62.64            .5764E-10
 61.88            .2882E-10
 61.15            .50435E-10
 60.39            .40348E-10
 59.75            .40348E-10
 59.13            .40348E-10
 57.87            .40348E-10
 57.3             .50435E-10
 56.69            .40348E-10
 56.17            .33623E-10
 54.65            .2017E-10
 53.31            .11208E-10
 52.04            .41171E-11
 51.6             .42923E-11
 51.22            .59335E-11
 50.85            .40348E-11
 50.44            .492E-11
 50.08            .3593E-11
 49.36            .27262E-11
 48.99            .24602E-11
 48.63            .24906E-11
 48.27            .19474E-11
 47.91            .1868E-11
 47.26            .154E-11
 46.95            .15055E-11
 46.67            .13913E-11
 46.24            .30567E-11
 46.02            .11146E-10
 45.7             .11208E-11
 45.38            .89662E-12
 44.75            .61694E-12

Ready
```

```
% plot
Virtual array file name? sigphand.dat
log X? y
Minimum value of X? 1.E-13
Maximum value of X? 1.E-9
log Y? n
Minimum value of Y? 30.
Maximum value of Y? 80.
Do you wish axes to be marked? y
Distance between each tick mark on Y axis? 10.
Number of minor tick marks per scale division on Y axis? 1
Grid? n
Axes at margin?
Type of curve (smooth, dashed, point)? p
Character to appear at each point? +
Name of horizontal axis? sigma (mho/cm)
Name of vertical axis? Z (km)
Do you wish plot to be titled? y
Type in title line by line; terminate with `\\'
BLUNT PROBE #8 HAND REDUCED
JUNE 15, 1977
1720 AST\\

Output file? p1
Superimpose another plot?
%
```

☆U.S. GOVERNMENT PRINTING OFFICE: 1980-0-777-026/3