AD-A058 193 DAYTON UNIV OHIO RESEARCH INST F/G 9/2
AFML SCIENTIFIC AND ENGINEERING COMPUTER SUPPORT.(U)
DEC 77 D L FORD, T WOOD, M DENNIS, D G LEET F33615-75-C-5191
UNCLASSIFIED UDRI-TR-77-10 AFML-TR-77-231 NL

1 OF 2
AD
A058193

AFML-TR-77-231

2

LEVEL II

ADA058193

AFML SCIENTIFIC AND ENGINEERING COMPUTER SUPPORT

UNIVERSITY OF DAYTON
DAYTON, OHIO

DECEMBER 1977

FINAL REPORT 15 MAY 1975 - NOVEMBER 1976

AD No.
DDC FILE COPY

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

AIR FORCE MATERIALS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

78 06 15 018

78 17 08 094

NOTICE

This technical report has been reviewed and is approved for publication.

[illegible]
ALAN R. MILLER, CAPT, USAF
Project Monitor

FOR THE DIRECTOR

[illegible]
WARREN P. JOHNSON
Chief, Operations Office

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(18) (19) REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER AFML-TR-77-231	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) AFML SCIENTIFIC AND ENGINEERING COMPUTER SUPPORT.		5. TYPE OF REPORT & PERIOD COVERED (9) Final Rep[redacted]t, 15 May 75-15 Nov[redacted] 76
		6. PERFORMING ORG. REPORT NUMBER (14) UDRI-TR-77-1[redacted]
7. AUTHOR(s) (10) Dale L. Ford, Thomas Wood, Michael Dennis Duane G. Leet		8. CONTRACT OR GRANT NUMBER(s) F33615-75-C-5191
9. PERFORMING ORGANIZATION NAME AND ADDRESS University of Dayton Research Institute 300 College Park Avenue Dayton, Ohio 45469		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Project Nr: 6106 Task Nr: 99930000 Work Unit Nr: 99930000
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Materials Laboratory (DOC) Air Force Systems Command Wright-Patterson AFB, Ohio 45433		12. REPORT DATE (11) Dec[redacted]77
		13. NUMBER OF PAGES 187
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) (15) F33615-75-C-5191		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for ~~Public Release~~; Distribution Unlimited

(16) 61062 9993 (17) 00

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

(12) 189p.

18. SUPPLEMENTARY NOTES

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

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

This report presents a summary of the work performed by the University of Dayton Research Institute under Contract No. F33615-75-C-5191 during 15 May 1975 to 15 November 1976. The work was accomplished under the Computer Activities Office. Air Force Materials Laboratory, Wright-Patterson Air Force Base, Ohio

78 17 08 094

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

105 400

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

FOREWORD

This final report was prepared by the University of Dayton Research Institute, Dayton, Ohio, under United States Air Force Contract F33615-75-C-5191. This contract was initiated under Project No. 6106. The work was administrated under the direction Computer Activities Office, Air Force Materials Laboratory, Wright-Patterson Air Force Base, Ohio.

This report covers scientific and engineering computer support supplied from 15 May 1975 to 15 November 1976.

Personnel who have contributed to this work are: Dale L. Ford, Thomas Wood, Michael Dennis, and Duane G. Leet.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF TABLES

SECTION I
INTRODUCTION

The application of computers in research programs is rapidly increasing due to the inherent advantages possible in the use of these machines. Experiments that were impractical or impossible with human operators are now feasible, mathematical models can be constructed and readily evaluated, sophisticated data reduction involving millions of calculations can be performed in seconds to enhance data analysis, computer graphics are providing an interactive dialogue with the computer to evaluate model building. All of these capabilities are being exploited to a greater and greater degree as one's awareness of the computer's capability increases and the relative time and cost per calculation continues to decrease. The Air Force Materials Laboratory use of computer has also been increasing. While the majority of the past computer experience of the Materials Laboratory has been with batch jobs to a large computer such as the CDC 6600 and high cost, specifically designed minicomputer systems, the introduction of the microprocessor has provided a flexible, low cost solution to obtaining maximum efficiency in in-house and on-site research programs.

This report summarizes the work performed in providing scientific and engineering computer support to various projects within AFML during this time period. This work includes exploitation of all computers available to AFML researchers. While the results of this work have been or are in the process of being published in either technical reports or scientific journals, completion of some projects and publication of the results will occur on a succeeding contract.

This report is divided into major topics: (1) Automated Experiments and Data Acquisition, (2) Computer Programs for Reentry and Laser Applications of Ablation/Thermal Effects, (3) Computer Analysis of Microstructures, and (4) General User Programs and Education.

SECTION II
AUTOMATED EXPERIMENTS AND DATA ACQUISITION

2.1 AFML DIGITAL COMPUTER HIERARCHY

It is the object of the Computer Activities Office of the AF Materials Laboratory to organize a digital computer hierarchy so that with the automation of laboratory experiments, data from the experiments can be transmitted to other digital computers within ASD for data reduction. A block diagram of this hierarchy is shown in Figure 1. This communications requires the use of telephone communications equipment such as modems, acoustic couplers, audio telephone lines, and hard-wired telephone lines. For the Materials Laboratory, the central point of this hierarchy is the SEL 86 located in Building 652. Data from the experiments is gathered by the mini and microcomputers and transmitted to the SEL via either low speed (110, 300 Baud) or high speed (9600 Baud) telephone lines.

The low speed links will communicate with SEL 86 through a program in the SEL called TSS (Terminal Support System). Thus the minicomputers must act as terminals when communicating with the SEL 86. The high speed links will communicate with the SEL 86 through a high speed multiplexer called a MAX. A high speed link is a dedicated link and is active at all times, so no "loggin" is necessary, but a great deal of data formulating is required. The SEL 86 driver for the high speed links allows such features as downstream loading, mass data storage for the experiment on the SEL, and transmission of both binary and ASCII data.

The mini and microcomputers in the laboratory consist of the following makes and models: HP 2100 (paper tape, cassette, mag tape, and disk operating systems), PDP-11 paper tape and floppy disk system), PDP-8 (disk system), Varian 620/L (disk system), and Control Logic (Intel 8080) (paper tape system).

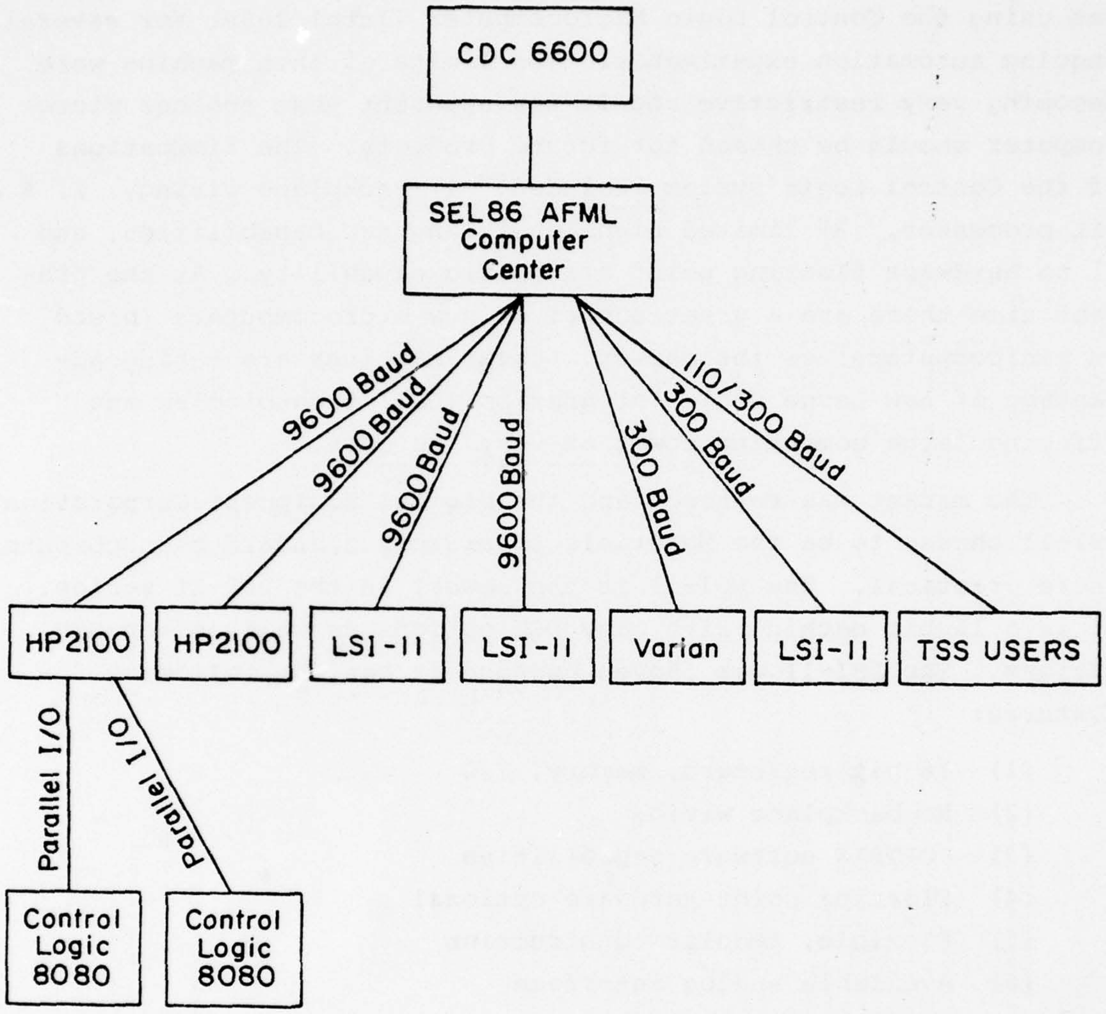


Figure 1. AFML Digital Computer Hierarchy.

2.2 SELECTION OF A LABORATORY STANDARD MICROCOMPUTER

At the beginning of this contract the Materials Laboratory was using the Control Logic Microcomputer (Intel 8080) for several ongoing automation experiments. The limits of this machine were becoming very restrictive and it was apparent that another microcomputer should be chosen for future projects. The limitations of the Control Logic system included: 1) Backplane wiring, 2) 8 bit processor, 3) limited high level language capabilities, and 4) no hardware floating point arithmetic capability. At the present time there are a great number of new microcomputers (based on minicomputers) on the market. These machines are taking advantage of new Large Scale Integration (LSI) technologies and offering large computing power at very low cost.

The market was reviewed and the Digital Equipment Corporation LSI-11 chosen to be the Materials Laboratory standard microcomputer where practical. The LSI-11 is the newest in the PDP-11 series. It is a 16 bit machine with many DEC options as well as non-DEC options. The LSI-11 was chosen because it has the following features:

(1) 16 bit registers, memory, I/O
(2) No backplane wiring
(3) FORTRAN software capabilities
(4) Floating point hardware optional
(5) Flexible, modular construction
(6) Available analog interface
(7) DEC software support
(8) DEC maintenance support
(9) Reasonable cost (less than Control Logic)

At the time that the LSI-11 was chosen there were requirements for five data acquisition systems, and since that time additional proposals and preliminary designs using the LSI-11 have been generated. Due to the fact that it is a true state-of-the-art machine, the LSI-11 should remain as the laboratory standard microcomputer (where practical) for many years to come.

2.3 SPECIFIC APPLICATIONS

One of the activities of this contract has been the specification, design, and implementation of data acquisition and process control systems to automate the experiments in the Materials Laboratory. This activity has involved three computers: Control Logic (Intel 8080), DEC LSI-11, and HP 2100. The automation of experiments with these machines is intended to become part of the Materials Laboratory Computer Hierarchy (see Figure 1).

2.3.1 Liquid Adsorption Experiment Automation

The liquid adsorption experiment of the Surface Interactions Branch studies the rate of adsorption and desorption of a fibrous sample with various concentrations of solvent and solute liquids passing over it. Using a valve network as shown in Figure 2 pumps cause various concentrations of solvent and solute liquids to flow into the sample chamber and the adsorption of the sample is measured with the defractometer.

The liquid adsorption experiment of the Surface Analysis Group (AFML/MBM) has been implemented using a Control Logic Microcomputer. Figures 2 and 3 show block diagrams of this experiment and control system. A Technical Report (AFML-TR-76-XX) describes the experiment and automation.[1] This system (hardware and software) was initially fabricated by AFML/DOC personnel, but due to changes in assignments and manpower, the debug, redesign, and final implementation (hardware and software) were done under this contract. The software flowchart is shown in Figure 4. A listing of the program is given in Appendix A. The control system is used for two purposes - first to take the place of a human operator in overnight experiments, and secondly to automate the data gathering so that data can be input to a data reduction program in a more convenient form. The software for this experiment was generated using PLM, a high order language developed by INTEL Corporation for use on their 8080 microprocessor. AFML/DOC had access to the PLM compiler on the GE time-share system. The program was written, stored, and compiled on the GE time-share system, and the object file punched on paper tape to load into the Control Logic System.

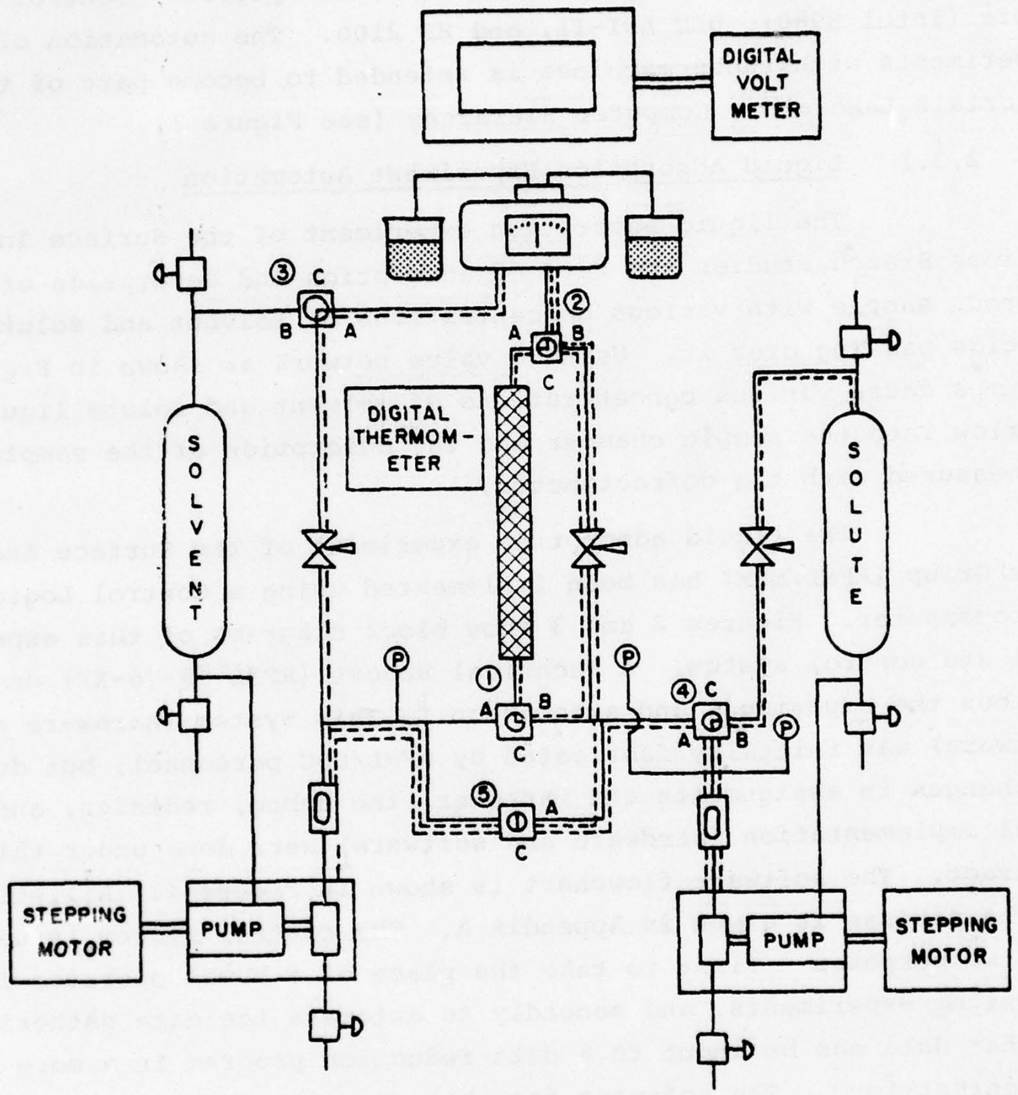


Figure 2. Surface Analysis Experiment Block Diagram.

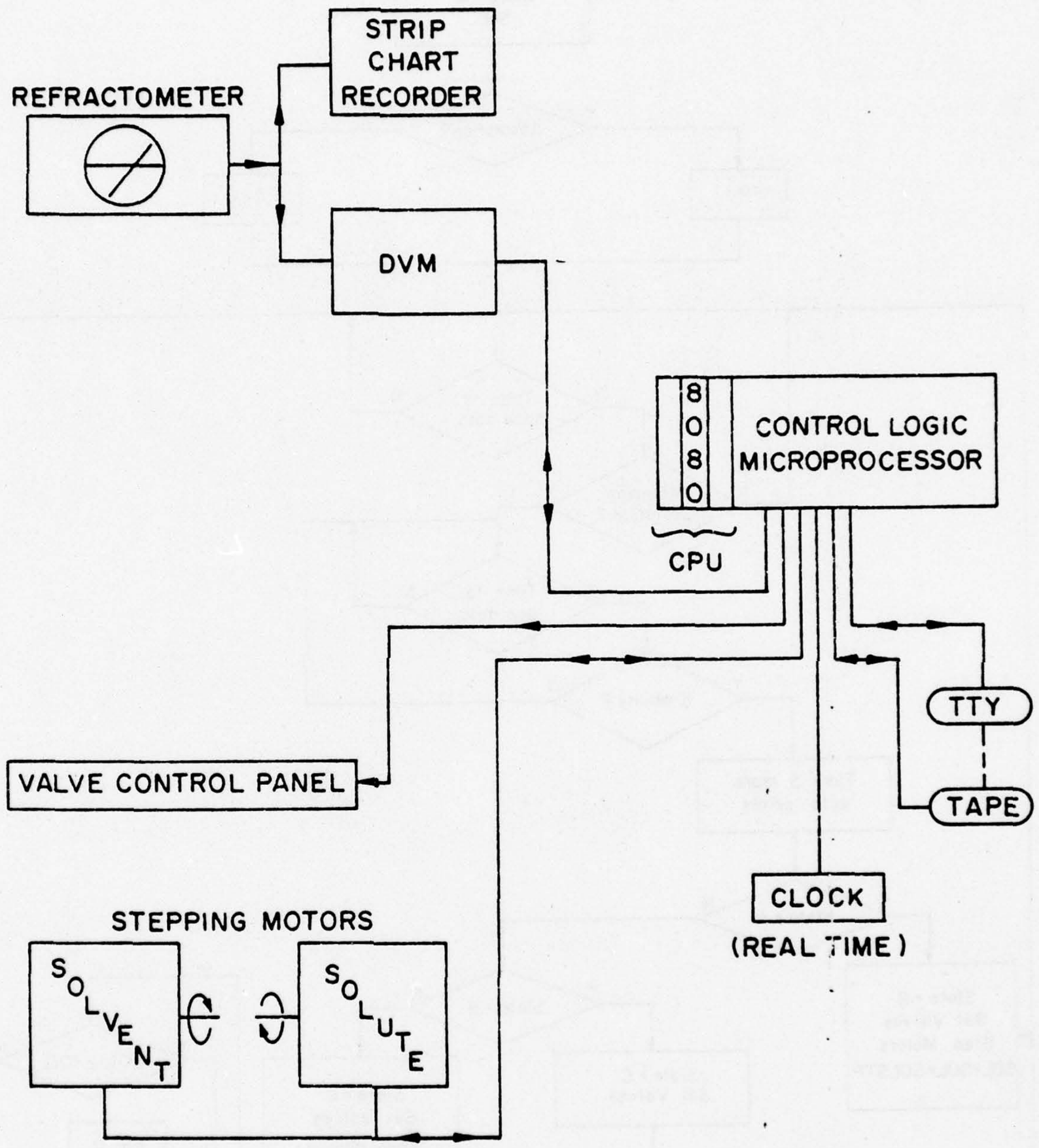


Figure 3. Control System for Surface Analysis Experiment.

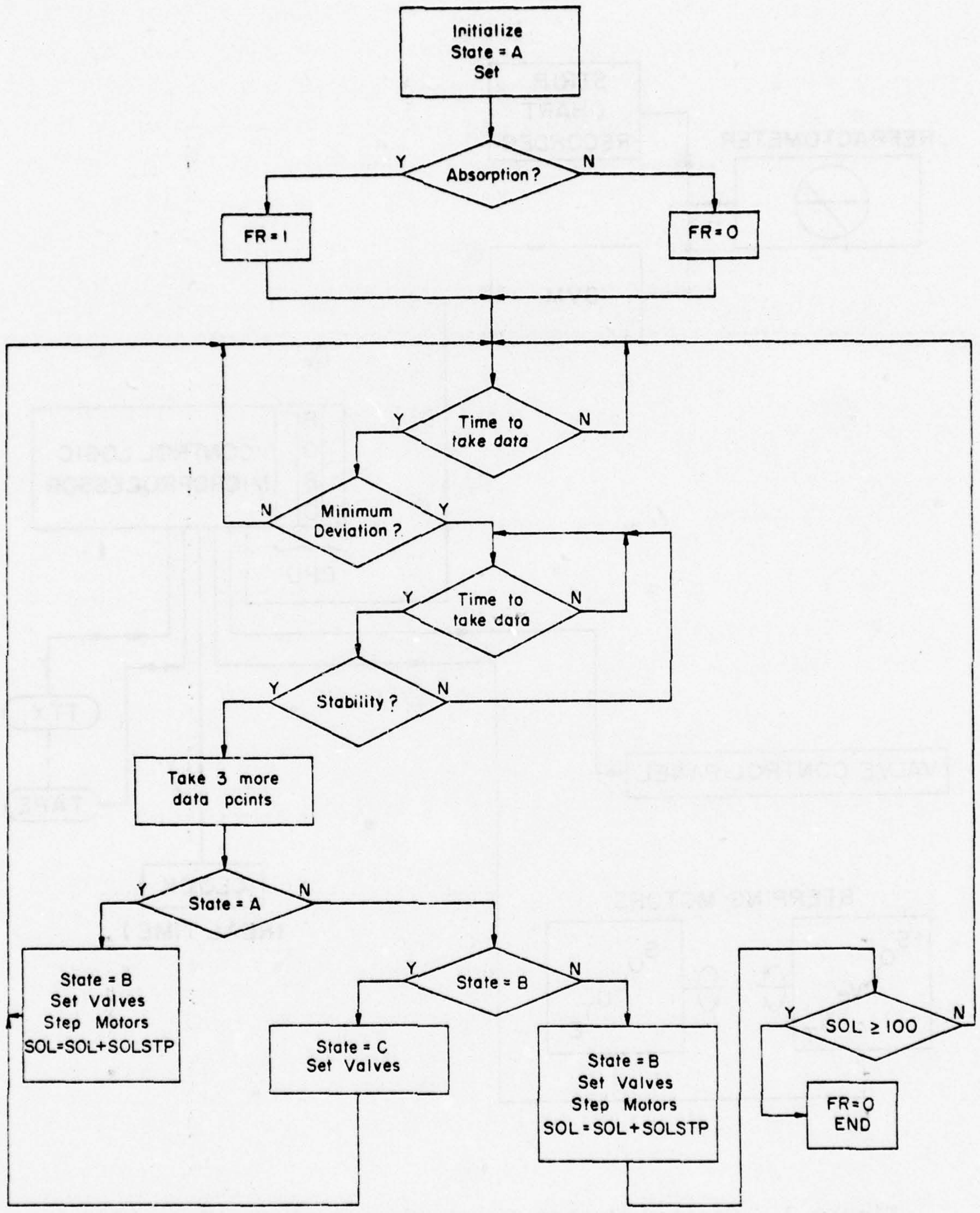


Figure 4. Software Flowchart.

The four states of the valves are called states A, B, C, and the rest state. These states relate to the quantity of solvent and solute passed over the sample. The rest state is the state in which the valves are not controlled by the computer, but rather by switches on the front panel. In state B a solution of solvent and solute is passed over the sample. In state C a solution of solvent only is passed over the sample. The solvent to solute ratio is stepped in increments from 0 to 1 and back to 0. The initial and final states of the experiment are state A. In state B and state C the interferometer measures the adsorbtion of the sample and the control system takes data from the interferometer, watching for the signal to stabilize. Once stability has been reached, the control system changes the state of the valves to the next state. Throughout this process the relative amounts of solvent and solute are changed by the control system through the stepping motors. In the past, one experiment required constant attention from an operator and the constant taking of data by an operator. Often one experiment could run for 48-72 hours. The computer control system has greatly relieved the operators task and made data gathering much easier.

2.3.2 Polymer-Surface Interaction Experiment Automation

The polymer-surface interaction experiment of the Surface Interactions Branch studies the gas volumetric adsorption properties of a material sample which is exposed to various concentrations of gases. The chamber containing the gases and sample is analyzed using a mass spectrometer. This automation required the control of a UTI-100C Mass Spectrometer by an HP 2100 Disk System minicomputer. The interface was configured using a Hewlett-Packard universal parallel interface and a Hewlett-Packard 8 channel Analog to Digital Converter. The block diagram of this automation system is shown in Figure 5. The parallel interface required some signal conditioning in order to drive some relays in the mass spectrometer. The cable and signal conditioner schematics are shown in Figure 6. The controls of the mass spectrometer such as operation mode, amplitude range, and sweep start are handled by the

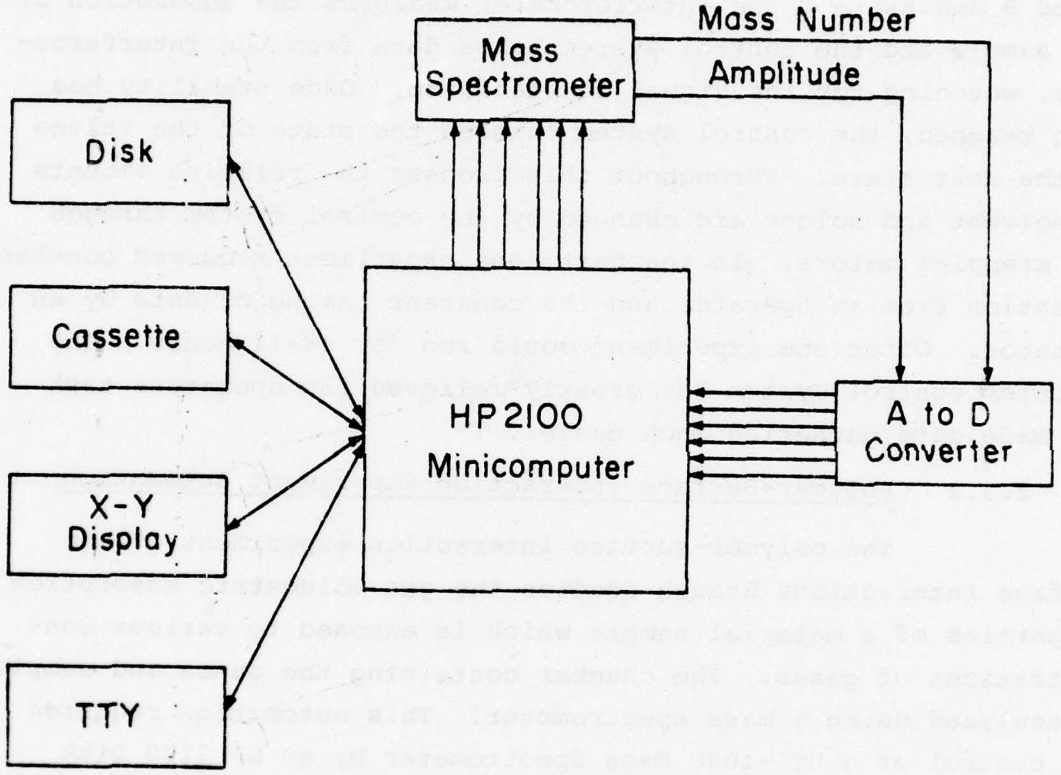


Figure 5. Automation of Polymer-Surface Interaction Experiment.

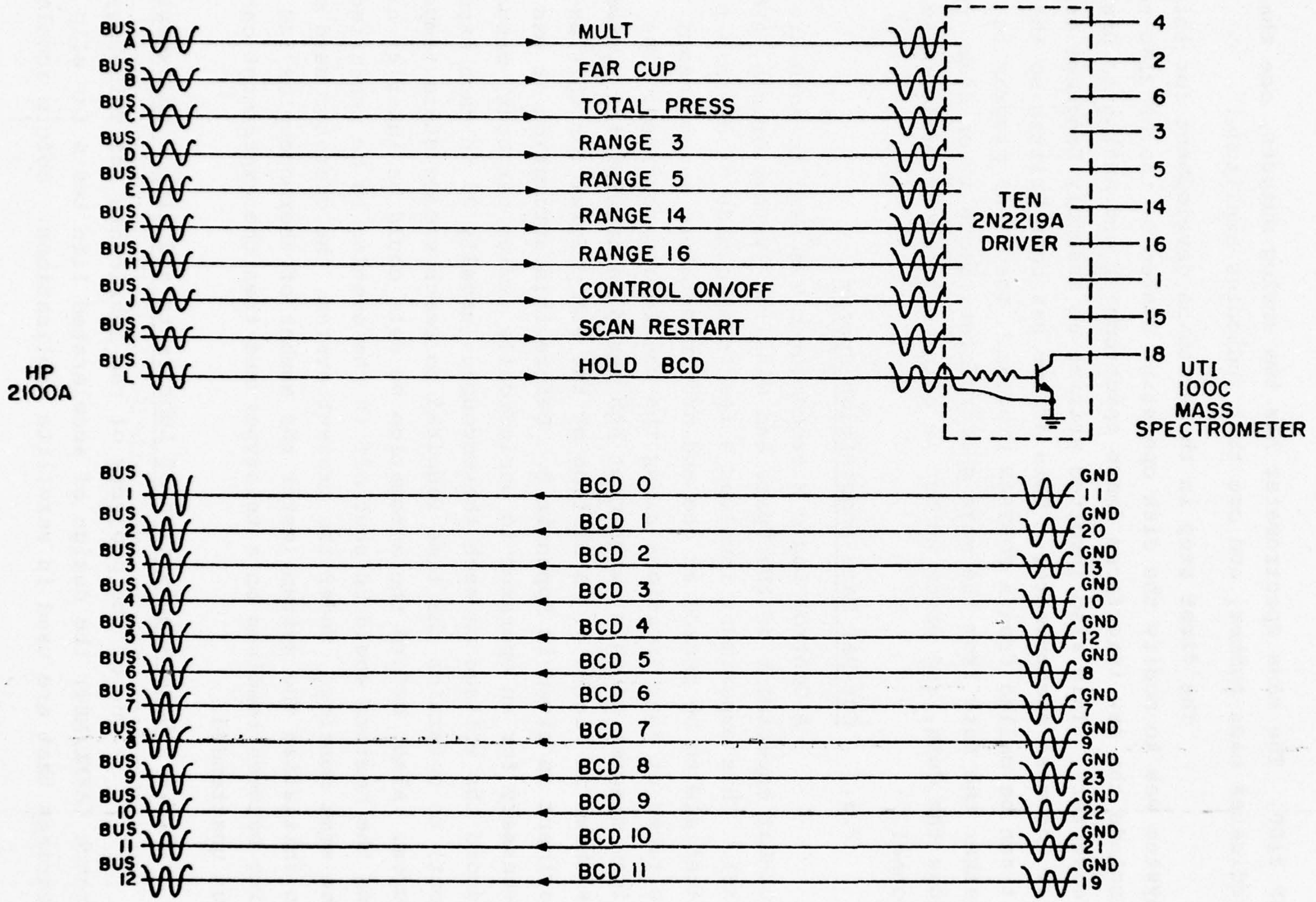


Figure 6. HP 2100 to Mass Spectrometer Interface.

HP 2100. The mass spectrometer has two analog outputs, one that indicates mass number, and one that indicates amplitude.

The first step in the software development for this system was to modify the disk operating system of the HP 2100 to include the new interfaces (mass spectrometer and Analog to Digital Converter). This also involved writing an Assembly Language driver for the mass spectrometer. This driver has been written so that it can be called from a FORTRAN program. The final program to gather the data from the mass spectrometer, store it on disk, reduce the data, and print it out is being written by AFML/DOC personnel.

2.3.3 Cut-Bar Data Acquisition System

A Control Logic Microcomputer was used to automate a Cut-Bar experiment of the Space and Missiles Systems Branch (AFML/MXS). This experiment involves a test sample made of different materials to be heated at one end and temperature measurements to be taken at many locations along the sample and recorded. The block diagram of this experiment and the data acquisition system is shown in Figure 7. A listing of the PLM program for this experiment is given in Appendix B. Before this automation it was necessary for an operator to periodically (twice per hour) manually record the voltage on each thermocouple (usually 20 in each experiment) to determine the time required to reach steady state temperatures. Also, before the automation no data could be taken at night and the heaters would be shut off in the evening to be restarted the next morning. Under the present system, the operator need only to initialize the system (enter the number of thermocouples and the time between readings on a teletype) and then the experiment can be run unattended.

2.3.4 Data Acquisition of Lubricants and Slip Ring Experiment

One of the projects of the Lubricants and Tribology Branch (AFML/MBT) the design of accelerated life tests for slip ring bearings that are used in satellite applications. Before accelerated

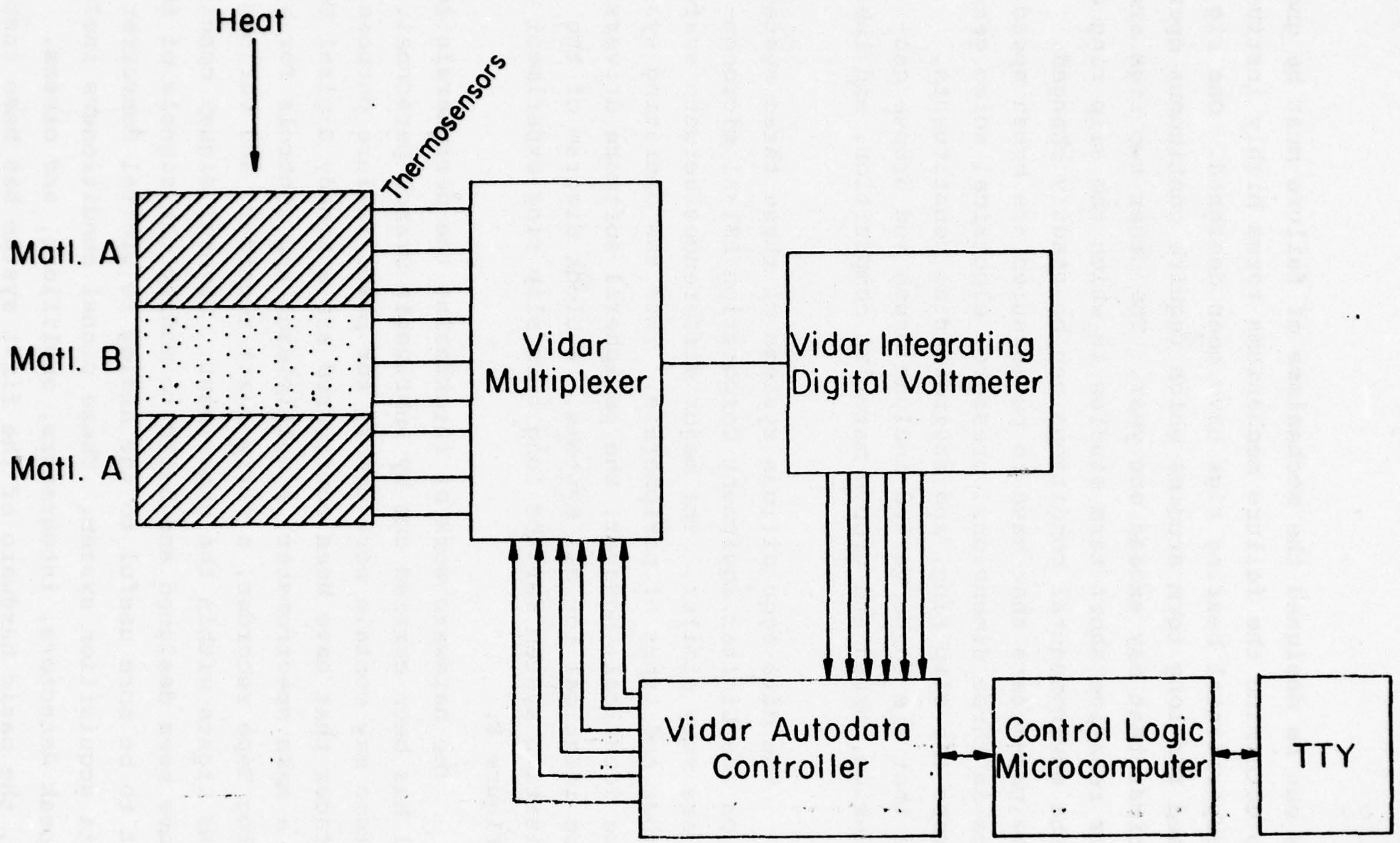

Heat
Thermosensors
Matl. A
Matl. B
Matl. A
Vidar Multiplexer
Vidar Integrating Digital Voltmeter
Vidar Autodata Controller
Control Logic Microcomputer
TTY

Figure 7. Block Diagram of the Cut-Bar Experiment.

life tests can be designed the mechanisms of failure must be understood. To determine the failure mechanisms three highly instrumentated experimental bearing rigs have been designed. One rig will be used for long term studies which require continuous operation for times that may exceed one year. The other two rigs are to be used for relative short term studies in which the slip ring design, or the environmental conditions can be readily changed. Some of the parameters that have to be measured are brush speed, brush force in three dimensions, pressure, electrical noise generated across the slip ring, and environmental constituents. Parameters that are to be varied include brush and groove geometrical design, brush and groove material composition, and lubricants.

The data acquisitions systems of these three systems are based on the Digital Equipment Corporation LSI-11 microcomputer and are very similar. The major differences between systems is the number and types of peripherals. Thus the operating system will be identical; however, the peripheral software drivers will not be identical for all systems. A block diagram of the data acquisition system for the long term slip ring experiment is shown in Figure 8.

The hardware work of interfacing the peripherals to the LSI-11 has been carried out by Lubricants Branch personnel. These systems may contain core memory for power failure purposes. The interfaces that have been configured are a Kenedy Digital Tape recorder, a mass spectrometer, a multiplexer and controls for an Ampex Analog Tape recorder, a strip chart recorder, and various sensors and lights within the experiment. Several signal conditioners have been designed and build to modify the signals of the experiment to be more useful to the Analog to Digital Converter of the data acquisition system. These signal conditioners include filters, peak detectors, integrators, amplifiers, and others. Presently, the basic hardware of the first system has been installed, and the software drivers for the peripherals are being written to checkout the hardware. When these systems are complete it will

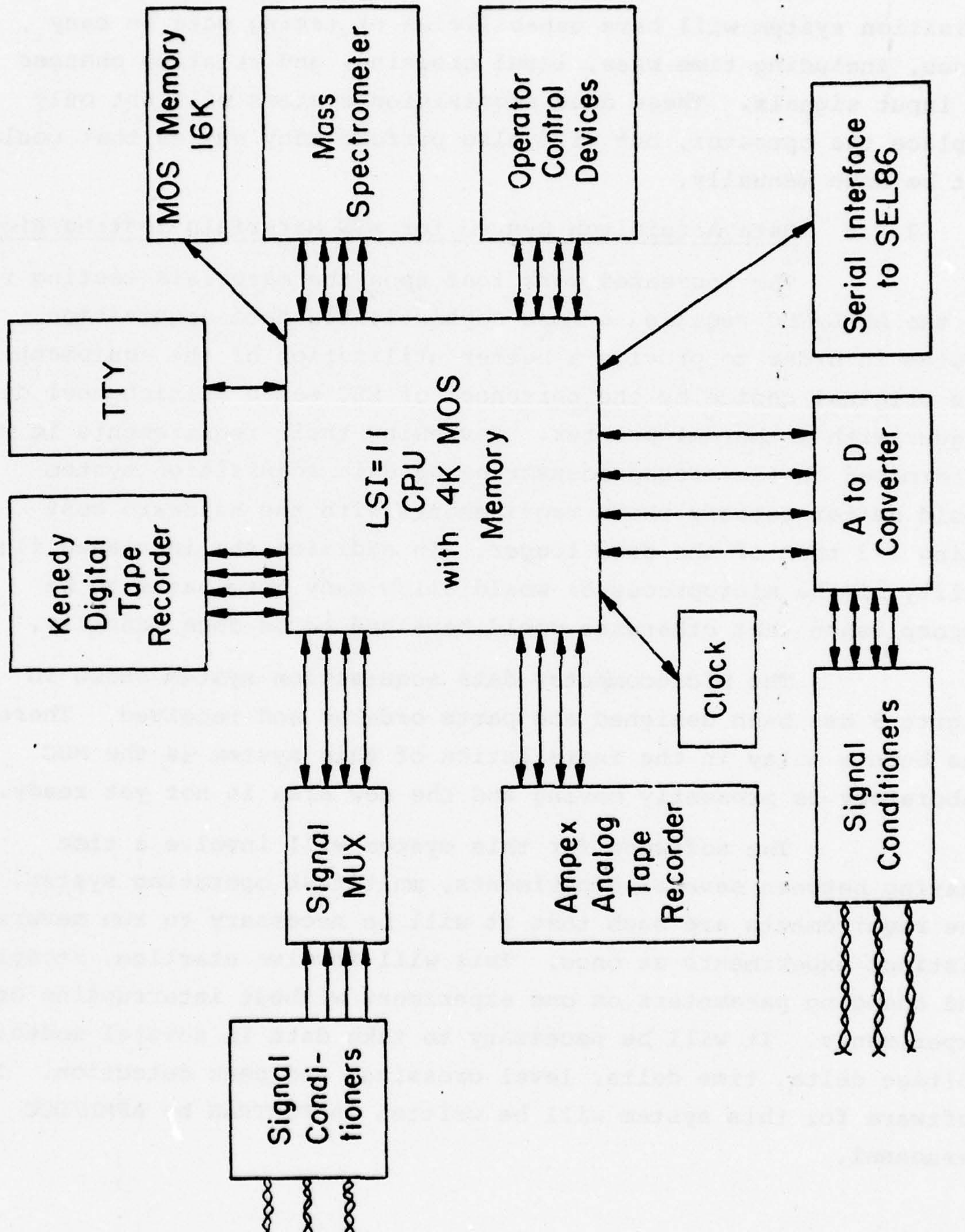


Figure 8. Lubricants Data Acquisition System Block Diagram.

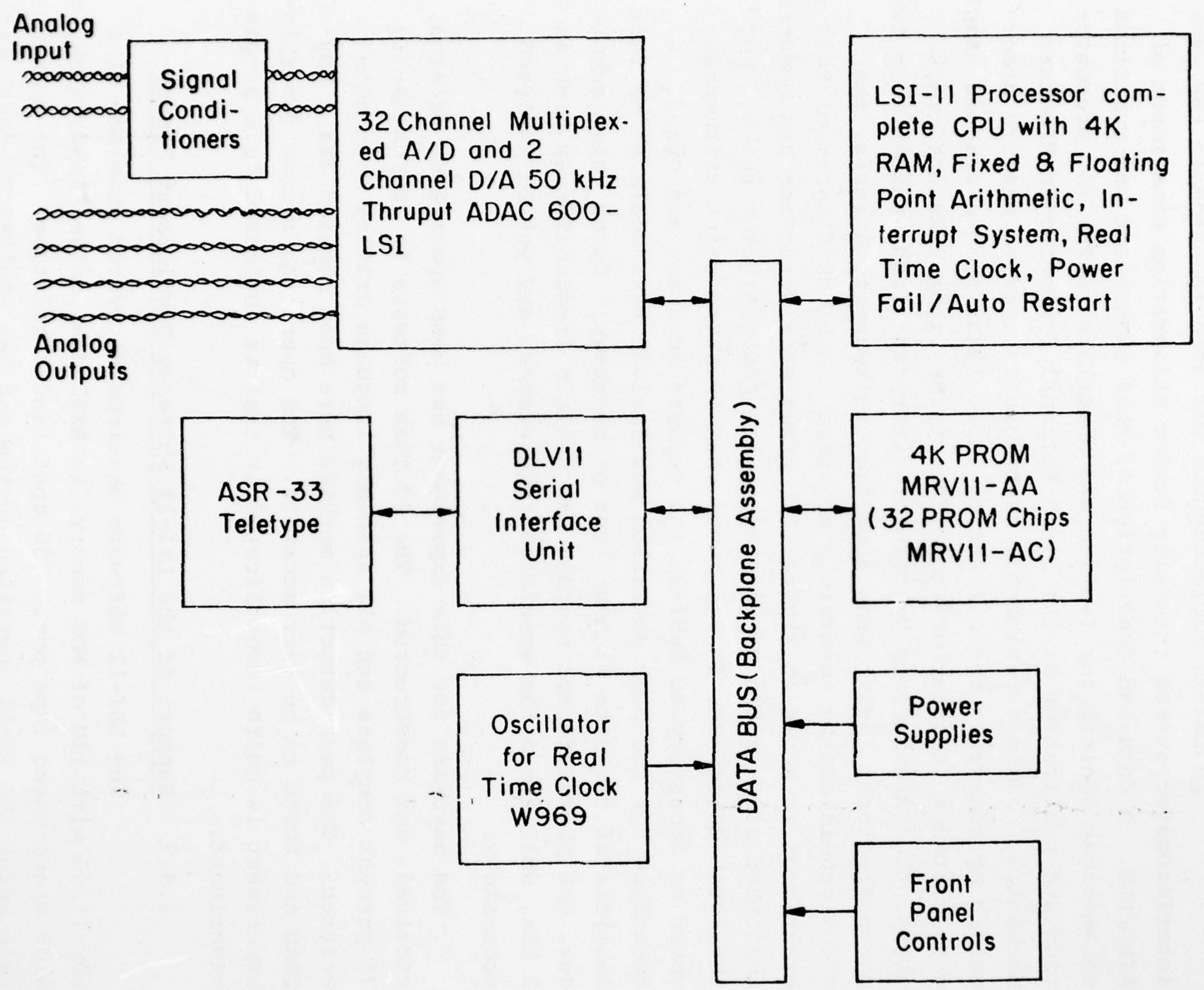
be possible for the data acquisition systems to take data in any of the several methods on a 24 hour per day basis. The data acquisition system will have capabilities of taking data in many modes, including time base, level crossing, and relative changes of input signals. These data acquisition systems will not only replace the operator, but will also perform many duties that could not be done manually.

2.3.5 Data Acquisiton System for MTS Materials Testing Rigs

The increased work load upon the materials testing rigs of the AFML/MBC required a more sophisticated data acquisition system in order to provide a better utilization of the equipment. The original choice by the personnel of MBC was a multichannel data logger with a thermal printer. Reviewing their requirements it was determined that a microprocessor based data acquisition system could better satisfy their requirements with the hardware cost being 2/3 that of the data logger. In addition the increased flexibility of the microprocessor would allow many more tasks to be accomplished that otherwise would have had to be done manually.

The microcomputer data acquisition system shown in Figure 9 has been designed and parts ordered and received. There has been a delay in the installation of this system as the MBC laboratory is presently moving and the new area is not yet ready.

The software for this system will involve a time sharing between several experiments, multitask operating system. The requirements are such that it will be necessary to run several distinct experiments at once. This will involve starting, stopping, and changing parameters on one experiment without interrupting other experiments. It will be necessary to take data in several modes: voltage delta, time delta, level crossing, and peak detection. The software for this system will be written in FORTRAN by AFML/DOC personnel.



Analog Input
Signal Conditioners
32 Channel Multiplexed A/D and 2 Channel D/A 50 kHz Thruput ADAC 600-LSI
Analog Outputs
LSI-11 Processor complete CPU with 4K RAM, Fixed & Floating Point Arithmetic, Interrupt System, Real Time Clock, Power Fail / Auto Restart
DATA BUS (Backplane Assembly)
ASR-33 Teletype
DLV11 Serial Interface Unit
4K PROM MRV11-AA (32 PROM Chips MRV11-AC)
Oscillator for Real Time Clock W969
Power Supplies
Front Panel Controls

Figure 9. Data Acquisition System for the MTS Testing Rigs.

2.3.6 Automation of a Stress Intensity Factor Calibration Experiment

An LSI-11 microcomputer was used to automate the user interferometer stress intensity factor calibration experiment of AFML/LLN. A detailed description of this experiment was submitted to American Society for Testing and Materials (ASTM) for presentation and publication at the Ninth National Symposium on Fracture Mechanics.[2] This experimental technique is used to measure crack growth by analyzing the diffracted laser light from a cracked sample with distinct indentations to diffract the light. As the crack grows, the interference pattern of diffracted light passes over two photodetectors. Using peak detection software techniques, the crack growth can be accurately measured. A block diagram of this experiment is shown in Figure 10. After peak detection has occurred the other signals are correlated to the photodetector peaks. For this peak detection, it was found that floating point arithmetic would be necessary to achieve the desired accuracy and speed. A technique for the peak detection was derived that would allow fast analysis of the data without loss of accuracy. Using this technique, the photodetector signals can contain frequencies as high as 2 kHz, well above the specified requirement, and yet be analyzed accurately.

The hardware for this experiment has been specified, ordered, received, and constructed. The FORTRAN software is approximately 75 percent complete and all assembly language drivers have been written. The peak detection methods have been modeled using FORTRAN and found to be very accurate. The operating system for this experiment is quite uncomplicated in that is dedicated to a single experiment.

2.3.7 Support of the LSI-11 Software Development System

The LSI-11 software development system consists of a PDP-11/03 with 12K of MOS memory, an RX01 dual drive Floppy disk, a high speed paper tape punch (30 cps), and a teletype. The system runs under the RT-11 operating system and has FORTRAN IV capabilities.

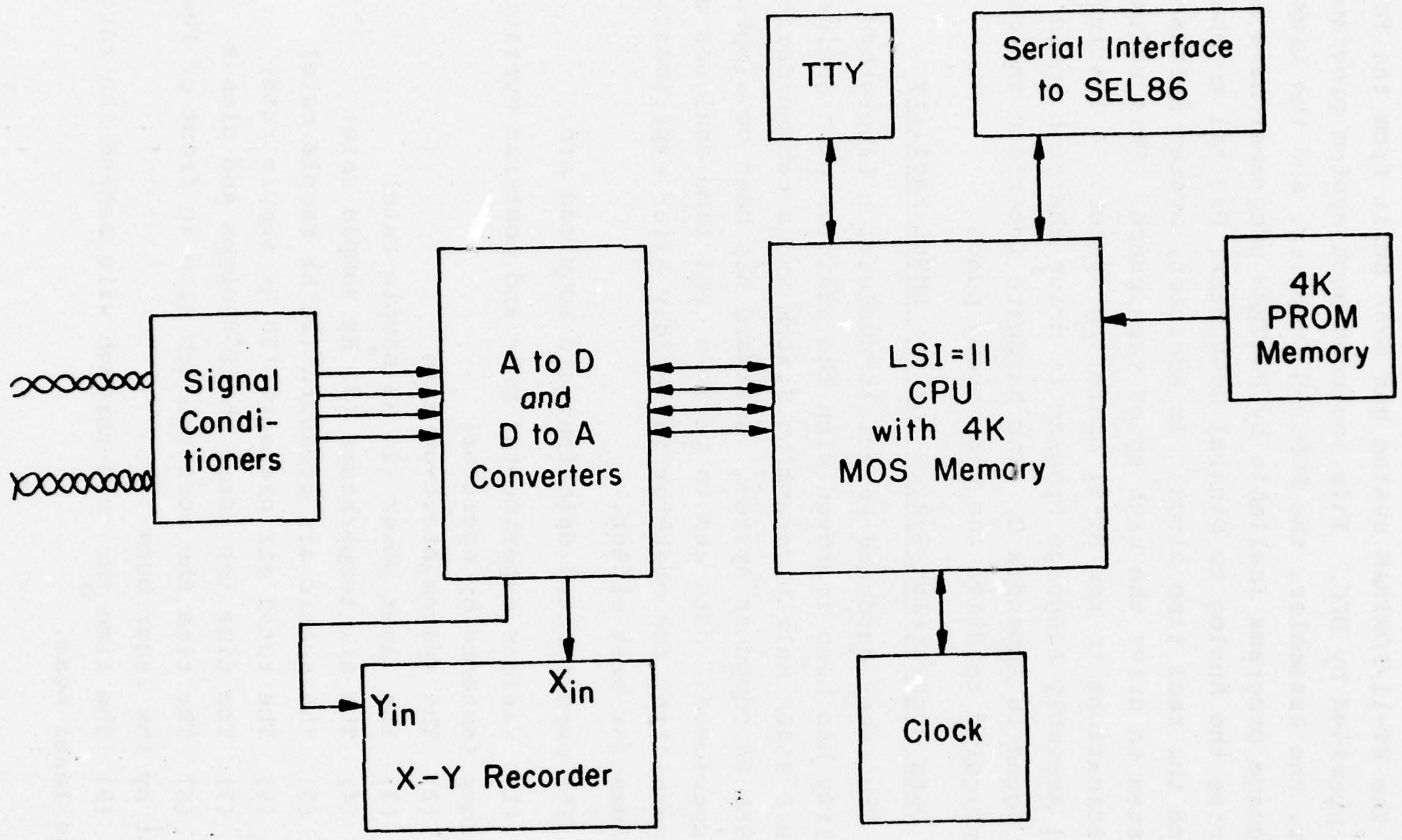

TTY
Serial Interface to SEL86
4K PROM Memory
LSI=11 CPU with 4K MOS Memory
Signal Condi-tioners
A to D and D to A Converters
X_{in}
Y_{in}
X-Y Recorder
Clock

Figure 10. Block Diagram of Stress Intensity Factor Experiment Automation.

Our usage of the RT-11/FORTRAN system has been built from the four master disks provided by DEC. This includes such system programs as the Editor, the Assembler, the FORTRAN Compiler, and the Linker. Assembly Language programs (callable by FORTRAN programs) have been written to drive the Analog to Digital Converter, Digital to Analog converter, and the real time clock. In addition, system software has been written to drive the high speed tape punch. This software included modifications to the RT-11 operating system. A listing of the PDP-11 Assembly Language Program to drive the high speed punch can be found in Appendix C. The hardware interface and cables were also configured to drive the high speed punch.

2.3.8 Data Acquisition System for the LHMEL Facility

The Laser Hardened Materials Evaluation Laboratory (LHMEL) facility has been improved with the addition of a dedicated Hewlett-Packard 9640A Multiprogramming System and a communication link to the SEL 86 computer system. Software has been developed to acquire experimental data and to perform real time analyses of these data, permitting the operator to rapidly alter experimental parameter values for best effect.

The experimental data that are acquired are:

(1) Various experimental pre- and post-run operating parameter values (entered by operator)

(2) The system start-up time

(3) The laser power (20 Hz sample rate)

(4) The air temperature (20 Hz sample rate)

(5) The static air pressure (20 Hz sample rate)

(6) The total air pressure (20 Hz sample rate)

(7) The time the laser shutter opens and closes

(8) The time the burn-through wire in front of the target is cut by the laser beam

(9) The time the burn-through wire behind the target is cut by the laser beam.

The principal component of the Hewlett-Packard system is a 21MX computer, operating under the Hewlett-Packard supplied RTE-C operating system. To begin an experimental run, the operator uses RTE-C to activate the program OZ, and enters the operating parameter values that have changed since the last experimental run. The operator then has the program OZGO activated to control the acquisition of the experimental data. Initialization of the laser control sequencer by the operator starts the program data acquisition sequence.

When the experiment terminates, the operator uses RTE-C to activate the program OXPST to enter the post-run operating parameters values. If desired the operator can have the program DADMP activated to inspect the raw data acquired during the run. If the data are acceptable, the SEL 86 computer program can be activated. Within this program the operator sends the raw data over a communication link to the SEL 86 computer and initiates the execution of the analysis program SELOZ (on the SEL 86). This program performs the following computations:

(1) The run time (time of shutter closure minus system start-up time)

(2) The shutter time (time of shutter closure minus time shutter opened)

(3) The burn-through time (back burn-through wire time minus front burn-through wire time)

(4) The laser power curve over the run time interval

(5) The laser power threshold time (the time when the laser power becomes less than (or equal to) the laser power threshold going down minus the time when the laser power becomes greater than (or equal to) the laser power threshold going up)

(6) The maximum, minimum, and average values of laser power over the shutter time interval

(7) The maximum and average values of laser power over the laser power threshold time interval

(8) The total energy over the laser power threshold time interval

(9) The diameter and area of the beam on-target

(10) The laser power on-target curve over the run time interval

(11) The average power on-target over the shutter time interval

(12) The average power density on-target over the shutter time interval

(13) The energy density on-target over the shutter time interval

(14) The average power on-target over the burn-thorugh time interval

(15) The average power density on-target over the burn-through time interval

(16) The energy density on-target over the shutter time interval

(17) The total temperature curve over the run time interval

(18) The minimum temperature over the shutter time interval

(19) The static air pressure curve over the run time interval

(20) The RMS value of static air pressure over the shutter time interval

(21) The total air pressure curve over the run time interval

(22) The RMS value of total air pressure over the shutter time interval, and

(23) The Mach number.

The data and analytical results are conveniently formatted and printed on a high speed printer. They are also archived on magnetic tape. CALCOMP quality plots are made of the power on-target, total temperature, static pressure, and total pressure curves. A summary of the analysis is printed for the operator on the LHMEL teletype. The operator can use this summary to select the experimental conditions for the next run. The users manual is found in Appendix D.

SECTION III
COMPUTER PROGRAMS FOR RE-ENTRY AND LASER APPLICATIONS OF ABLATION/THERMAL EFFECTS

3.1 INTRODUCTION

During the last fifteen years many computational techniques have been developed to simulate the ablation/thermal effects occurring in noise tips, heat shields, and rocket nozzles. With some modifications these computer programs can also be applied to analyze the ablation/thermal response of materials to laser heating and provide guidelines in the development of new materials. AFML has sponsored two research programs to select appropriate computational techniques for ablation/thermal analyses. These programs have led to two sets of computer programs; Aerotherm Prediction Procedure for Laser Effects (APPLE), and Unified Ablative Material Thermal Response Analysis Procedure (UNIFIED). While these two sets of computer programs contain many similar mathematical algorithms, each has special features designed for their particular application. For instance APPLE permits in-depth absorption of radiant heating, while UNIFIED has additional codes for boundary layer calculations of re-entry environments and also a discrete model capability.

This section is divided into two parts; each part describes the application of one set of computer codes to current AFML problems. The application of the UNIFIED codes has been used to determine the sensitivity of a one dimensional thermal response solution of carbon phenolic in an arc-jet environment to pertubations of its thermophysical properties. Specific applications of the APPLE code include the heating of ZnSe and KCl CO_2 laser output coupler windows and simulation of a quartz lampbank heating of painted aluminum in a windtunnel.

3.2 APPLICATIONS OF THE APPLE CODE

3.2.1 Introduction

APPLE consists of a set of computer codes developed to describe material response to low power ($<10^5$ kw/cm^2) laser radiation such as typified by an infrared CO_2 laser operating in the CW mode. The basis of these codes is the aerospace technology computer codes developed for combined radiative and convective heating conditions. The conversion of these codes to laser problems is feasible because low power heating rate is similar to that experienced by a high performance re-entry vehicle or a far planetary entry probe. Because these codes have been developed specifically for aerospace applications and not laser heating, a perfect transition is not expected in solving problems involving laser heating. Difficulties arising from the application of these codes are not due to numerical computational technique; but rather to the assumptions on thermophysical and thermochemical properties, necessitated by the lack of data on the material of interest.

The following descriptions are of the main routines comprising APPLE and are taken from the APPLE users manual.[3]

The Aerotherm Chemical Equilibrium (ACE) computer program is an extremely versatile code for calculating quantities of importance to a broad variety of thermochemical processes. The thermochemical processes treated may be divided into two categories; closed systems and open systems. Closed systems are those for which the relative amounts of each chemical elements in the system are constant if prespecified. Open systems are those for which the relative amounts of chemical elements depend on various mass transfer rates due, for example, to boundary layer convection or solid surface degradation. The ACE program can treat both systems in chemical equilibrium and systems for which certain reactions are kinetically controlled.* The most important objective of the

*The kinetic controlled reactions routine (KINET) was deleted for use in the APPLE computer program library.

the ACE program when used in conjunction with Aerotherm's CMA/CMAC and ASTHMA computer codes is to generate tables of gas-phase properties at the wall interface as a function of three independent variables: pressure, dimensionless gas blowing rate, and dimensionless char blowing rate (p, B_g', and B_c' respectively). In addition to calculating the chemical and thermodynamic state of a variety of systems, the ACE program also calculates and prints out some transport properties (e.g., viscosity, thermal conductivity) appropriate to that state.

The Charring Material Thermal Response and Ablation Program (CMA) is an implicit, finite-difference computational procedure for computing the one-dimensional transient transport of thermal energy in a three-dimensional isotropic material which can ablate from a front surface and which can decompose in depth. Decomposition reactions are based on a three-component model. The program permits up to eight different back-up materials of arbitrary thickness. The back wall of the composite material may transfer energy by convection and by radiation.

The ablating-surface boundary condition may take one of three forms:

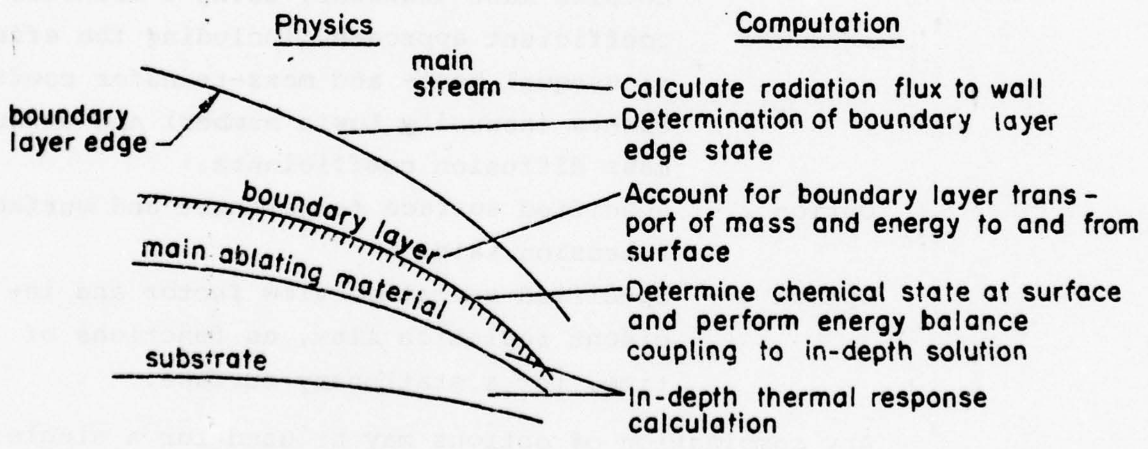
Option 1 - General convection-radiation heating with coupled mass transfer, using a transfer coefficient approach, including the effects of unequal heat- and mass-transfer coefficients (nonunity Lewis number) and unequal mass diffusion coefficients.

Option 2 - Specified surface temperature and surface recession rate.

Option 3 - Specified radiation view factor and incident radiation flux, as functions of time, for a stationary surface.

Any combination of options may be used for a single computation. Option 3 is appropriate to cool down after termination of convective heat input and is often useful in conjunction with Options 1 and 2.

The Axi-Symmetric Transient Heating and Material Ablation Version 3 (ASTHMA3) Computer Program is a transient heat conduction program for two-dimensional, axi-symmetric bodies. Multiple non-charring, anisotropic materials may be studied. The surface boundary condition has three options, including an unusually general thermochemical erosion or ablation condition as well as simplified radiation and specified temperature options. The in-depth temperature prediction is of the familiar explicit finite-difference type. It allows a completely general finite difference mesh layout relative to the physical r-z axes, and accounts for anisotropic heat conduction effects. The heated surface boundary condition is an unusually general thermochemical type. It accounts for two specific kinetically controlled carbon reduction reaction, one kinetically controlled water gas shift reaction, and any number of gas-phase equilibrium reactions for any combination of ablating materials and environments. Chief applications for the computer program are rocket nozzles and entry vehicles. The following sketch serves to clarify various physical aspects of the ablation problem treated.


Physics
Computation
main stream
boundary layer edge
boundary layer
main ablating material
substrate
Calculate radiation flux to wall
Determination of boundary layer edge state
Account for boundary layer transport of mass and energy to and from surface
Determine chemical state at surface and perform energy balance coupling to in-depth solution
In-depth thermal response calculation

The JANAF routine is a special purpose code that has been incorporated into the COUPLE computer program. It assembles a molecular species thermochemical data (JANAF) file in the appropriate format for input into the ACE program. JANAF performs these functions by searching in the JANAF data tape for the molecular species that can be formed from a prescribed list of atomic elements that is specified in the input.

The COUPLE preprocessor routine is a user oriented program specifically developed to execute singly or coupled a variety of Aerotherm surface state thermochemistry and in-depth ablation and thermal response computer programs. It generates appropriate input data for the computer programs that are to be executed from simple execution directives and very little input data. Version 1 couples the ACE and JANAF routines to obtain surface state thermochemistry solutions that are used as an input to the CMA and ASTHMA computer programs. Version 2 couples the ACE, JANAF, and CMA routines, while version 3 couples ACE, JANAF, CMA, and ASTHMA routines.

While running APPLE coupled decreases total turn around time for analysis as only a single job is submitted, it is usually desirable to run APPLE uncoupled. Thus no parameter values are assigned by default and the validity of assumptions made can be more readily determined. In addition, if the problem doesn't run correctly, debug is easier as the responsible routines can more easily be located. Brief descriptions of problems solved with the APPLE code under this contract are included below. A more detailed description of these problems can be found in a series of AFML technical memos[5-9], and in a technical report by Rondeau and Ford.[10]

3.2.2 Thermal Response of "Painted" Aluminum to Radiant Heating

Currently, various elastomeric coatings on aluminum plates mounted in a windtunnel are being exposed to radiant heating by a quartz lampbank. The objective of this experiment is the assessment of the nuclear flash protection afforded by the elastomeric coating applied to an aircraft's outer skin. A model of of this experiment consisting of the thermophysical properties of a polyurethane coating material and aluminum 2024T3, the physical dimension, and the boundary conditions (heating rate and heat transfer coefficient) were used with the APPLE code to calculate the back face temperature of the aluminum substrate as a function of time. The agreement with experimentally obtained back face temperature was quite good.[5] The use of APPLE does not supplant the need to do the experiment but rather compliments it. The experiment helps determine physical model to be used as input to APPLE, i.e., knowledge of whether the coating would peel, ablate, or blister during the experiment. Once agreement between experimental and calculated results is obtained, the APPLE code can then be used to predict the results for extrapolated experimental conditions.

In addition APPLE was used to parametrically study the effect of coating thermal conductivity on the surface and in-depth temperatures of a polyurethane coated aluminum sample subject to a thermal pulse in a windtunnel. The results show that temperature changes from a $\pm 20\%$ variation in the thermal conductivity of the coating are relatively minor.

The code was also used to model an in-flight scenario where the wing section of an aircraft, traveling at Mach 0.85 and 30,000 feet is exposed to the same heat flux as that produced in the laboratory experiment. Comparisons between the flight simulation and laboratory experiments have been presented along with the parametric study of the coating thermal conductivity in References 6 and 9.

3.2.3 Thermal and Stress Analysis of the AFWL Zinc Selenide Laser Window

Operating conditions of the Air Force Weapons Laboratory Electric Discharge Coaxial Laser (EDCL)[11] were used to calculate the axisymmetric (2D) temperature profiles in the zinc selenide output coupler CO_2 laser windows. These temperatures were then used to calculate the thermally induced stresses in the window. The calculated stresses were well within design expectations for the given operating conditions.[7]

3.2.4 Computed Temperature Response of the AFML Flat-Top Laser Window

AFML's 10 Kilowatt Flat-Top laser is based on the design of AFWL EDCL laser.[11] Because of the greater absorptance of the AFML zinc selenide windows, as determined by post-mortem examination, sufficiently high thermally induced stresses resulted in two window failures under normal operating conditions. The APPLE codes were used to verify that thermally induced stresses were a possible failure mechanism and to generate curves of maximum thermally induced stress versus time for various power levels. These calculations provide safe operating criteria to prevent failure of the laser coupler window. This work is described in detail in Reference 8.

3.3 UNIFIED COMPUTER PROGRAMS

3.3.1 Introduction

The primary objective in Unified set of computer programs as developed by Aerotherm is to apply the advanced computer technology developed over the past 15 years (for the ablation design analysis problem) to the far more difficult problem area of materials development guidance. The relative payoffs of altering the chemical composition of a composite material, reductions of density and thermal conductivity, and changing the composites configuration both on a macroscale and a microscale are all important areas that can be addressed with the Unified set of codes. The one dimensional computer program, CMA, primary use has been to determine the sensitivity of material response in an arc jet to changes in the thermophysical properties of carbon phenolic. This data coupled with the relative uncertainties of the individual, thermophysical properties illustrate where the most beneficial use of scarce resources can be used to reduce the uncertainty of the parameters that most effect the material response. Similar analyses can be done for a re-entry environment and using the two dimensional computer program ASTHMA4; however the more interesting results will be using ASTHMA4 to obtain material responses due to material configuration. The first study using CMA is described below.

3.3.2 Sensitivity Study Using CMA

A Carbon Phenolic button, backed by a stainless steel plate, placed in an arc jet was modeled for analysis with the CMA computer code. Fourteen nodes were placed in the Carbon Phenolic button and two nodes in the stainless steel backup plate. The environmental conditions used were those similar to a typical arc-jet experimental test run (i.e., pressure = 1.07 atm, heat input of 1050 BTU/(ft^2-sec) for 30 seconds and 0 for the remaining 70 seconds). The baseline thermophysical properties of carbon phenolic that were used are shown in Table I. The properties that were varied $\pm 10\%$ each are shown in Table II.

The response, as a function of time, of the carbon phenolic material was measured by seven parameters. These are the interface temperatures between the carbon phenolic and the stainless steel and 6 dimensionless variables shown in Table III.

Thirty-five computer runs were made (2 runs as each parameter was varied $\pm 10\%$ and the baseline case). Since the material response functions as defined in Table II are not calculated directly by the CMA computer program, the CMA computer program was modified to create a new file which contained all quantities necessary to calculate these response functions. This file was catalogued and then at a later time used as an input file to the computer program RES which then calculated the individual response function. This computer program is listed in Appendix E.

The sensitivity coefficients, defined as $\frac{\Delta R_j}{\Delta X_J}$ where ΔR_j is the change in the j th material response function due to a ΔX_i change made to the X_i material input parameter, are determined by the computer codes. The sensitivity coefficients for R_o (the interface temperature) and the experimental uncertainty in the materials input parameters are listed in Table IV.[12] The

TABLE I

THERMOPHYSICAL PROPERTIES OF CARBON PHENOLIC

(CCAZ CLOTH/5C 1008 RESIN)

MATERIAL	TEMPERATURE (^{0}R)	SPECIFIC HEAT (BTU/LB-^{0}R)	CONDUCTIVITY (BTU/LB-FT-^{0}R)	EMISSIVITY
Virgin Carbon Phenolic	530	0.25	1.45×10^{-4}	.85
(MX4926, 0^0 Layup)	800	0.34	1.65	.85
	1160	0.38	1.97	.85
	1500	0.38	2.34	.85
	6000	0.38	2.34	.85
Charred Carbon Phenolic	530	0.16	0.64×10^{-4}	.85
(MX4926, 0^0 Layup)	1000	0.33	0.64	.85
	1500	0.39	0.64	.85
	2000	0.42	0.64	.85
	2500	0.45	0.78	.85
	3000	0.47	0.95	.85
	3500	0.49	1.1	.85
	4000	0.51	1.3	.85
	5000	0.53	1.4	.85
	6000	0.55	1.5	.85
Stainless Steel	492	.11	22.2×10^{-4}	.14
(Backface Material)	672	.11	26.1	.14
	1032	.11	30.3	.14
	1400	.11	30.4	.14

PYROLYSIS GAS ENTHALPY		INTERNAL DECOMPOSITION KINETIC DATA				
TEMP ^{0}R	H BTU/LB	i	ρ_{v_i} lb/ft^3	ρ_{c_i} lb/ft^3	B_i sec^{-1}	$E_i/^0R$ ^{0}R
900	-1899.0					
1800	-761.9	1 (resin)	19.34	0.	1.4×10^3	1.54×10^4
2700	8868.6					
3600	2894.7	2 (resin)	58.00	29.24	4.8×10^9	3.68×10^4
4500	3953.5					
5400	5747.2	3 (reinforcement)	98.55	98.55	0.	0.

HEAT OF FORMATION OF PHENOLIC: -363 BTU/LB

VIRGIN MATERIAL RESIN MASS FRACTION: 0.330

VOLUME FRACTION: 0.378

TABLE II

THERMOPHYSICAL PROPERTIES THAT WERE VARIED ±10% EACH SEPARATELY

VIRGIN DENSITY OF FABRIC
CHAR DENSITY OF FABRIC
VIRGIN DENSITY OF RESIN 1
VIRGIN DENSITY OF RESIN 2
CHAR DENSITY OF RESIN 2
VOLUME FRACTION OF RESIN IN VIRGIN COMPOSITE
VIRGIN CONDUCTIVITY OF COMPOSITE AS f(TEMP)
CHAR CONDUCTIVITY OF COMPOSITE AS f(TEMP)
VIRGIN SPECIFIC HEAT OF COMPOSITE AS f(TEMP)

CLEAR SPECIFIC HEAT OF COMPOSITE AS f(TEMP)
EMISSIVITY OF VIRGIN COMPOSITE AS f(TEMP)
EMISSIVITY OF CHARRED COMPOSITE AS f(TEMP)
BLOWING RATE PARAMETER
PYROLYSIS GAS ENTHALPY AS f(TEMP)
PYROLYSIS REACTION GAS COUNTS FOR RESIN 1
PYROLYSIS REACTION GAS COUNTS FOR RESIN 2
HEAT OF FORMATION

TABLE III

MATERIAL RESPONSE PARAMETERS (R_j)

$R_0 = T_{BF}$

$$R_1 = \frac{\int_0^X \frac{k}{\rho C_p X^2}\, dx}{\int_0^X dx}$$

$$R_2 = \int_0^t R_1\, dt$$

$$R_3 = \frac{\int_0^t \dot{q}_{out}\, dt}{\int_0^t \dot{q}_{in}\, dt}$$

$$R_4 = \frac{\int_0^t h_g \dot{m}_g\, dt}{\int_0^t \dot{q}_{in}\, dt}$$

$$R_5 = \frac{\int_0^t h_x \dot{\rho}\, dt}{\int_0^t \dot{q}_{in}\, dt}$$

$$R_6 = \frac{\int_0^t H^T \dot{m}_g\, dt}{\int_0^t \dot{q}_{in}\, dt}$$

T_{BF}	- INTERFACE TEMPERATURE
k	- THERMAL CONDUCTIVITY
ρ	- DENSITY OF THE MATERIAL
C_p	- SPECIFIC HEAT OF THE MATERIAL
X	- THICKNESS OF THE MATERIAL
$\dot{q}_{in}$	- RATE OF HEAT ABSORBED BY THE MATERIAL
$\dot{q}_{out}$	- RATE OF HEAT LOST BY THE MATERIAL
h_g	- ENTHALPY OF THE GASEOUS MATERIAL
$\dot{m}_g$	- RATE OF MASS LOSS OF THE MATERIAL
h_x	- ENTHALPY OF THE SOLID MATERIAL
H^T	- HEAT OF FORMATION
t	- TIME.

TABLE IV

SENSITIVITY COEFFICIENTS FOR INTERFACE TEMPERATURE

i		PARAMETER	COEFFICIENT $\Delta Ro/\Delta X_i$	UNCERTAINTY in X_i	PRODUCT
1.	ρ	VIRGIN DENSITY OF FABRIC	1.0480	.02	.021
2.	ρ_{c_f}	CHAR DENSITY OF FABRIC	-1.2590	.02	.025
3.	ρ_{v_r}	VIRGIN DENSITY, RESIN 1	0.0284	.01	----
4.	ρ_{v_r}	VIRGIN DENSITY, RESIN 2	-0.0515	.01	----
5.	ρ_{c_r}	CHAR DENSITY, RESIN 2	0.0083	.02	----
6.	τ_r	RESIN VOLUME FRACTION	0.0256	.05	.001
7.	$k_v(T)$	VIRGIN CONDUCTIVITY TABLE	0.2626	.10	.026
8.	$k_c(T)$	CHAR CONDUCTIVITY TABLE	0.1570	.20-?	.031
9.	$C_{p_v}(T)$	VIRGIN SPECIFIC HEAT TABLE	-0.1080	.10	.010
10.	$C_{p_c}(T)$	CHAR SPECIFIC HEAT TABLE	0.0322	.15-?	.005
11.	$E_v(T)$	VIRGIN EMISSIVITY TABLE	-0.0009	.02	.002
12.	$E_c(T)$	CHAR EMISSIVITY TABLE	-0.0695	.02	.001
13.	λ	BLOWING RATE PARAMETER	-0.0110	.10-?	.001
14.	$H_g(T)$	PYROLYSIS GAS ENTHALPY TABLE	0.0109	.05	.001
15.	E_i, B_i	PYROLYSIS REACTION CONSTANTS, RESIN 1	0.1275	.05	.006
16.	E_i, B_i	PYROLYSIS REACTION CONSTANTS, RESIN 2	0.2270	.05	.01
17.	H^T	HEAT OF FORMATION	0.1287	.02	.002

total uncertainty of a material response function ΔR_j is given by

$$\Delta R_j = \sum_i \frac{\Delta R_j}{\Delta X_i} \Delta X_i \tag{1}$$

with $\frac{\Delta R_j}{\Delta X_i}$ bieng the sensitivity coefficient and ΔX_i the experimental uncertainty of the ith material input parameter. An appreciable reduction in ΔR_j can be made by reducing the uncertainty ΔX_i only if the product involving ΔX_i in Equation (1) is large in comparison to other terms in the sum. The parameters that most significantly effect the uncertainty of the interface temperature as determined by this study and listed in Table IV are:

(1)	CHAR CONDUCTIVITY	- MOD SENS/HIGH UNC
(2)	VIRGIN CONDUCTIVITY	- MOD SENS/HIGH UNC
(3)	CHAR FABRIC DENSITY	- HIGH SENS/MOD UNC
(4)	VIRGIN FABRIC DENSITY	- HIGH SENS/MOD UNC
(5)	VIRGIN SPECIFIC HEAT	- MOD SENS/MOD UNC
(6)	REACTION CONSTANTS	- MOD SENS/MOD UNC

Similar analyses are being conducted for the other response characteristics and are to be published.

SECTION IV
COMPUTER ANALYSIS OF MICROSTRUCTURES

4.1 COMPUTER RECOGNITION OF TITANIUM ALLOW MICROSTRUCTURES

4.1.1 Introduction

The objective of this project is to predict the mechanical properties of titanium alloys by computer examination of photomicrographs of their microstructures. Prior to this contract, computer codes were developed to effect the solution.[13] The results, however, were inconclusive; the correlations obtained did not consistently provide accurate predictions of the mechanical properties of samples processed. The previously developed solution was basically three steps.

1. A digital image of a microstructure is produced by digitizing a photomicrograph negative. The digital image consists of light intensity readings (pixels) taken at 40 micron intervals of the negative. The pixel (picture points) intensities are valued 0-255.

2. The digital image is operated on by computer software, the particles of the microstructure are extracted and their geometric characteristics determined.

3. These geometric characteristics of each image are correlated with the measured mechanical properties of the respective Ti-alloy samples, yielding a relation that will predict mechanical properties from the geometric characteristics of the particles in the microstructure of an alloy sample. Work performed on this project under this contract was concerned primarily with improvements in the computer processing involved in Part 2 above.

4.1.2 Initial Work

The first step taken in working on the project was to become familiar with the problem and gain a working knowledge of the use of the computer codes used to process the digital images.

Soon after work was underway on the project, a new set of photomicrographs of the Ti-alloy surfaces, which were of higher quality than photomicrographs used in the previous work, became available. It was theorized that processing these would yield better correlation results, and the processing of the new images was begun. Several of the programs that performed initial image processing were found to be inoperative. It was determined that the majority of the problems encountered in the attempt to execute these programs were due to changes in the operating system of the CDC 6600 computer that had been made since the programs were written and last used in a production mode. The computer codes were modified to make them compatible with the present CDC 6600 operating system and the processing of the new images continued.

4.1.3 Improvements of the Digital Processing

A close examination of the available intermediate results of the processing done during previous work on the project was undertaken in an effort to gain insight into possible improvements that could be made. Conclusions reached were that image thresholding and particle separation/extraction should be the areas of refinement. The digital image thresholding, in which the raw image pixel values of 0-255 are reduced to a value of 0 or 1, is the most critical stage of the processing; improvements in this process would cause significant improvement in the accuracy of the particle geometric characteristics determined at the final stage of the image processing. In the particle separation/extraction phase, it was observed that frequently particles that were very close together in the photomicrograph were extracted from the digital image and treated as one large particle. This was due to higher density levels of the pixels in the area of the junction of these particles, caused by light scattering and the resultant noise when the photomicrograph was digitized.

The two operations of image thresholding and particle separation, however, are not totally independent of each other. It was seen that the accuracy of particle separation/extraction would be greatly facilitated by a more accurate thresholding of the digital images. The approach taken to the improvement of the image processing was the development of a more accurate thresholding process, and the later development of a more precise method of particle separation/extraction, if necessary.

4.1.4 Improvement of Image Thresholding

The initial step in establishing an improved thresholding process was the collection of data of raw image characteristics. This data consisted primarily of computer printouts of the digital images and corresponding plots of pixel density versus frequency for a selection of photomicrographs.

In the previous method, the threshold density D_T was determined by

$$D_T = (D_H - D_L)/2$$

where D_L was the density below which five percent of the pixels of the image were contained, and D_H the density above which five percent of the pixels were contained. (This represents the standard technique for thresholding used in image analysis.) Through comparison of the original photomicrograph, the printout of the raw image, the density versus frequency plot, and the printout of the thresholded image for each sample used, it was concluded that this method did not consistently yield the best thresholded image.

Several alternate thresholding methods were examined. The techniques evaluated first were differential, line-by-line thresholding methods, and failed to produce accurately thresholded images. The poor results obtained are believed to be due to overall density variation across the photomicrograph, relatively high-density variation within the particles themselves, and general inaccuracies associated with applying one-dimensional operations to two-dimensional structures.

The thresholding method arrived at was essentially a modification to the original method. The thresholding density D_T was determined as

$$D_T = (D_H - D_L) *D_I$$

where D_H and D_L remained unchanged from the original method, and D_I is the percentage of the remaining image pixels to be considered as being within a particle. D_I is an input parameter to the thresholding program. Using a carefully selected value of D_I, determined by examining the density histogram for each image processed, the respective thresholded images were improved significantly.

4.1.5 Improvements in Particle Separation/Extraction

During the processing of the new photomicrographs, the particle separation/extraction portion of the computer program was found to be totally inoperative. Due to the program's complexity and lack of documentation, it was necessary to examine the entire code in detail in an effort to locate and correct the error(s). In doing so, it was discovered that numerous major programming errors were present; that in fact when operative the codes would not perform the operations as described in the furnished documentation. Because of these conditions, the design and development of new particle separation/extraction software was required. Also, the magnitude of the errors was certain to have adversely affected the results previously obtained using these codes.

4.1.6 Development of New Particle Separation/Extraction Software

Several methods of particle separation/extraction were investigated, in order to select the most accurate one for implementation. The first methods investigated employed circular and rectangular "windows" of various sizes which were "moved" over the raw digital images; while examining the portions of the image contained in the window. Particle edges are detected using this method by looking for large density variations occurring within the window. Once such a density change is found, the window is then moved slowly around the particle by maintaining the density differential interior to the window while shifting its position on the image, thus outlining a particle. This approach failed, however, due to the wide range of particle sizes, making the selection of the proper window area by the computer program extremely difficult at best. (Often a window would entirely enclose a small particle in one part of an image, and then be entirely enclosed itself by a large particle in a nearby option of the image.) Separation of two seemingly connected particles was unpredictable using this method, due to the varying sizes of these "bridges" connecting the particles in the digital image.

The method of particle separation/extraction chosen to be implemented employs the same basic concepts as the original theory.[13] In addition, the software incorporates the following:

1. input parameters should provide the capability of selecting the exact separation criteria for the case of erroneously connected particles;

2. actual separation should not be biased with respect to particle location in the image (previous method was biased in an upper-left sense, due to processing the image left to right and top to bottom);

3. the software should be much more efficient in terms of central memory required than its predecessor;

4. intermediate states of the image during separation/extraction should be available for precise evaluation of software performance.

The separation/extraction operation was broken into two main steps, to be performed by two individual programs:

1. Generation of interior particle pixel values (1,2, or 3, depending upon the location of a pixel with respect to the outer boundary of the particle).

2. Separation and extraction of the particles.

The design, development, and testing of a program (PARSEP) to perform the first step has been completed. This program requires much less central memory for its execution ($55,000_8$ words versus $165,000_8$ words for the previous program). This improvement was made possible by maintaining a much smaller portion of the image in central memory during processing. In addition, a utility program (IM6PRN) was written to print the image during and after the processing for software evaluation purposes.

The programming of the second step to process the image output by PARSEP (separation and extraction) was divided into five parts:

1. separation and extraction of particle cores (3-valued pixels);
2. determination, separation (when necessary), and extraction of the upper particle extremes (2-valued pixels);
3. determination, separation (when necessary), and extraction of the lower particle extremes (2-valued pixels);
4. determination, separation (when necessary), and extraction of the remaining left and right particle extremes (2-valued pixels); and
5. reconstruction and output of the particles.

The determination and separation operations stated above are effected by moving a window of dimensions 1 pixel by 1 pixel around in the stated area of a particle. This method locates particles from their interiors radially outward. In the case of seemingly "connected" particles, the user may specify the exact location at which these particles will be separated. This separation actually takes place at the user-specified relative location between the particle cores (3-valued pixels). For example, the user may specify that "connected" particles be separated at the location halfway between their cores. Software to perform parts 1, 2, and 3 has been designed, developed, and tested. Software to perform steps 4 and 5 has been designed and is presently in the development stage.

4.1.7 Additional Software Developed for Testing and Evaluation Purposes

In the process of testing and evaluating both previously existing and newly developed software, the following programs were written:

1. PICTURE - A program to print a "picture" of the raw digital images using 32 grey levels, obtained by overprinting of characters, for the purpose of quick, easy comparison of the raw digital image to the photomicrograph used to create it.

2, PARPLOT - A program to reconstruct and graphically display a microstructure from the geometric characteristics output by the MEAD program PROPERT.

3. SAMPLOT - A program to produce plots of frequency versus the log of area for particles extracted from images in previous work on the project.

4. FREQPLT - A program to produce plots of the image density histograms produced by the MEAD program HISTCDC.

In addition to the work described above, other work performed in association with this project consisted of the following:

1. Preparation of 10 image tapes used by the PAR Corporation in work related to this project.[14]

2. Familiarization with the use of the OLPARS (On-Line Pattern Analysis and Recognition System) of the Rome Air Development Center at Griffiss Air Force Base for the purpose of determining geometric characteristics-mechanical properties correlations. This system was determined suitable to perform correlation analysis when the new image geometric data becomes available.

4.2 COMPUTER MODELING OF CRACK PROPAGATION IN THE SURFACE TI-ALLOYS

4.2.1 Problem Description

In work performed with AFML/LLS, an effort to develop computer codes to model crack propagation across the surface of Titanium alloys was initiated as part of a continuing effort in this area. Given an alloy surface, the project objective is to, use the computer to predict the nature of a crack growing across it. At the present, this alloy surface is presented as an enlarged photograph of an alloy surface in which the features affecting crack propagation (grain structure, large particles, etc.) are easily discernable to the naked eye. Crack propagation is to be effected according to propagation-surface conditions relations specified by the software user.

4.2.2 System Requirements

Initial analysis of the problem yielded the following requirements for the developed software for the modeling process.

1. Capability of accurate representation of the alloy surface.
2. Capability of easy modification to that surface representation.
3. A high degree of flexibility in allowing for the specification of crack behavior criteria with respect to surface conditions, yet perform in strict accordance to that criteria.

Due to the desirability of a high degree of user-software interaction, the software developed functions in an interactive mode. There existed the need for graphics capability, for both establishing and modifying the surface representation, and for displaying intermediate and final results of the modeling process. AFML's TEKTRONIX 4014 graphics terminal and 4954 graphics tablet in conjunction with the INTERCOM system of the CDC 6600 was selected to be used since this configuration satisfies the interactive graphics capabilities required.

4.2.3 Computer Representation of the Alloy Surface

The photograph of an alloy surface to be processed is used to establish the digital surface representation in the computer. This consists, basically, of five steps:

1. Prior to execution of the modeling program, the boundaries of areas of homogeneous grain orientation on the photograph are manually outlined. Within each outlined area, a line determining the grain orientation to be used for that area is drawn.

2. Each boundary and respective grain orientation line on the photograph are digitized using the TEKTRONIX 4954 graphics tablet.

3. The output from the digitizer is stored in the computer. The computer surface representation then, in essence, consists of a set of lines.

4. The surface representation is displayed graphically on the terminal screen. Any bad points resulting for erroneous digitized input are corrected by editing.

5. The final step is to eliminate gaps between points on the boundary lines, to insure that a crack cannot pass between two points on a boundary line and not be recognized by the computer as having intersected that boundary line. This is done by converting boundary line point values to integers and interpolating points between the original points comprising a boundary line such that for two consecutive points on a boundary line (X_1, Y_1), (X_2, Y_2),

$$|X_2 - X_1| = 1 \text{ or } 0,$$

and

$$|Y_2 - Y_1| = 1 \text{ or } 0.$$

In any case, one of the above relations must have a value of 1.

The above operations, with the exception of 1, are performed by programs of the CRAKMOD software system.

4.2.4 Linkage Table Generation

Once the user is satisfied with the established surface representation, the table of point-to-point links used to determine crack propagation is generated by CRAKMOD. At the present stage of development, these links are determined solely from the grain orientations of each boundary. (The linkage table structure, however, is designed to provide for additional links to be determined by additional crack propagation-surface condition criteria, as these relations become known.)

4.2.5 Crack Propagation

The propagation by the computer of a crack across the represented surface is initialized by the CRAKMOD user specifying its starting location. The software then determines the path of the crack by chaining together the points specified by the linkage table.

The main output from CRAKMOD is the graphical display of the surface and the crack grown across it. (Optional display capabilities include magnified views of one or more user selected surface areas, with or without the segments of the crack (if any) across them displayed.)

At the present stage of development, the CRAKMOD user may effect different crack configurations by specifying changes in the starting location of the crack and the grain orientations for the areas enclosed by the boundary lines. As other surface condition/crack propagation relations are recognized, all that will be necessary to modify CRAKMOD for their implementation will be the addition of program modules to generate the proper links, and provide for the input of any additional necessary features from the photograph (via digitizer or terminal). CRAKMOD is completely modular in structure, such that the implementation of the above features will require a minimum amount of modification to the existing software.

4.2.6 Additional Capabilities

In addition to the generation of cracks across the surface representation, CRAKMOD is capable of processing a crack digitized from the alloy surface photograph. Thus, crack generated by CRAKMOD may be compared to an actual crack in the surface processed.

There is the capability of measuring the length of the crack very accurately; much more accurately than is possible by measuring directly from a photograph or an actual sample, because the measurement may be made along the length of the actual crack using CRAKMOD, rather than along a straight line from its initial point to its end.

A listing of all FORTRAN programs developed for computer analysis of microstructures is enclosed in Appendix F.

SECTION V
GENERAL USER PROGRAMS AND EDUCATION

5.1 INTRODUCTION

A number of general user programs were written for use by AFML personnel. These include programs designed for graphics, digitizing, data analysis, support of the LSI-11 microprocessors and support of the Control Logic microprocessors. In addition, classes for AFML CDC 6600 and LSI-11 users were held to improve the utilization of these computers.

5.1.1 General User Programs for the CDC 6600

DIGIPLT is an easy-to-use, completely interactive software package that provides the capability of digitizing, displaying in graphic and tabular form, editing, and interpolating, data curves. Digitization and Graphics require a Tektronix 4394 or 4395 graphics tablet and 4014 Terminal respectively. The other tasks can be done at any terminal.

A complete description and instruction on DIGIPLT is given in the DIGIPLT USERS GUIDE in the Appendix G of this report.

In work performed under this contract, two relatively small, specific-purpose data processing programs were written. These programs are listed here along with a brief description of each.

DATPLOT -	A program to produce log-log plots, with axis systems, of crack growth rate versus stress intensity.
CONVRT -	A program to convert data output from the electron microprobe to usable format; edit erroneous points; and compute the mean and standard deviation for each data set processed.

These programs are listed in Appendix H.

5.1.2 LSI-11 Peripheral Driver Programs

The DEC RT-11 operating system can be used to generate stand alone FORTRAN program which contain a driver for a terminal. To drive other peripherals (such as A/D converter, D/A converter, real time clock, and modem interface) it is necessary to write Assembly Language programs that can be called from FORTRAN programs. There is a linking scheme used by RT-11 FORTRAN for Assembly Language subroutines. This scheme was used in writing these programs. The input and output parameters of the subroutines and the FORTRAN calls are shown in Table V. Listings of these PDP-11 Assembly Language programs can be found in Appendix I.

5.1.3 PLM Support Programs

The PLM compiler (Intel 8080 high-level language) originally used by AFML/DOC was the compiler installed on the General Electric Time-Share System. This compiler had two steps of operation. The first step had input of an ASCII source PLM program file and output of an ASCII file that contained Assembler Language and compiler error information. The second step had as input the output file from the first step and output of an ASCII file that contained memory and symbol information as well as the memory laoding information. It was necessary to punch an ASCII paper tape of the memory loading information and convert that information to a binary paper tape suitable for loading by the paper tape loader supplied by Control Logic. This conversion had been done previously on a Control Logic System with low-speed tape reader and was very time consuming (45 minutes for a 4K program). In addition, there was noise on the telephone lines which made punching a correct ASCII tape very unlikely. Many times the tape would have to be punched four or five times to get an error free tape. A FORTRAN program was written for the GE system that used the ASCII memory loading file as input and created a binary output file compatible with the paper tape loader. A listing of this program is in Appendix J. The binary output file was then dumped to the terminal tape punch (30 cps). This process greatly improved

TABLE V

CALLING ASSEMBLY LANGUAGE SUBROUTINES FROM RT-11 FORTRAN

Analog to Digital Converter:

```
    CALL ADC (ICH, IN)

            where ICH   =  Channel number to be converted (0 to 31)
                  IN    =  Integer equivalent of analog input on
                           A/D channel number ICH
```

Digital to Analog Converter:

```
    CALL DAC (ICH, IOUT)

            where ICH   =  Digital to Analog converter number (1 or 2)
                  IOUT  =  Integer equivalent of desired voltage
                           on D/A channel number ICH
```

Clock:

```
    CALL CLOCK (IFLG, NHR, NMIN, NSEC, NMSEC)

            where IFLG  =  0 for read clock,
                           1 for set clock
                  NHR   =  Integer number of hours
                  NMIN  =  Integer number of minutes
                  NSEC  =  Integer number of seconds
                 NMSEC  =  Integer number of milliseconds
```

turnaround time for minor program changes, especially in the debug procedure. An option was added to print an octal symbol table rather than the standard hexadecimal symbol table.

Also a FORTRAN program was written to scan through the output file of the first compiler step and print errors. A listing of this program is in Appendix J. Previously it has been necessary to search this file for errors line by line, which was not only slow and difficult, but inaccurate. This program would print each line with an error, the error diagnostic, and the total number of errors. Both of these utility programs will be adopted to either the SEL 86 or the CDC 6600 for future PLM use.

5.2 AFML USER EDUCATION

5.2.1 CDC 6600 Users

To determine methods AFML could employ to realize more efficient utilization of allocated computer resources, an analysis of AFML's use of the CDC 6600 was made. The results of this analysis indicated the following would yield a significant increase in efficiency and throughout for prime shift processing:

1. Limiting the Computer Resource Units available per AFML prime shift job to 100 CRUS. This would prevent a small number of large jobs expending a large percentage of AFML's allocated resources, which then results in low throughout long turnaround AFML jobs for the remainder of the shift.

2. AFML computer users requesting only the resources required for their computer jobs versus gross overestimates, resulting in unnecessary restrictions on computer resources available to AFML while such jobs are in the input queue or in execution.

3. More efficient use of permanent file space by using all five available cycles per permanent file; and by maintaining on permanent file only information currently being processed.

To effect the above, it was seen that a user education effort was in order to explain and demonstrate to AFML users the ways such practices caould be carried out and how their use would benefit the individual user. A class was conducted during which methods of efficient use of the computer were demonstrated and discussed.

A second analysis of AFML computer use was made to evaluate the effectiveness of the effort. The results showed an increase of 50% in prime shift throughout, and a significant increase in available permanent file space.

5.2.2 LSI-11 Users

A two day class attended by 14 people was held on the applications of the LSI-11 microcomputer for potential LSI-11 users. A wide variety of subjects were discussed ranging from very general to very specific. A large emphasis was placed on I/O as this applies to all users.

APPENDIX A
LIQUID ADSORPTION EXPERIMENT PLM PROGRAM

```
GO TO STEPL;
SS$OK:
SS(0) = SHL(SS(0),4);
SS(1) = SS(1) - 260Q;
SS(0) = SS(0) + SS(1);
/* TIME STEP SETUP  */
TIME$S:  CALL CRLF;
       DO P = 0 TO LAST (OT);
             CALL OUTWAIT;
             OUTPUT (8) = OT (P);
       END;
DO P = 0 TO 1;
       A(P) = TS(P) + 260Q;
       CALL OUTWAIT;
       OUTPUT (8) = A(P);
END;
CALL CRLF;
DO P = 0 TO 1;
       CALL INWAIT;
       A(P) = INPUT (0);
       CALL OUTWAIT;
       OUTPUT (8) = A(P);
       IF A(P) = SLASH THEN GO TO SET$DUN;
END;
DO P = 0 TO 1;
       TS(P) = A(P) - 260Q;
END;
GO TO TIME$S;
SET$DUN:
        DO P = 0 TO 4;
              CALL CRLF;
       END;
I=0;
DO P = 0 TO 2;
       DEFSL(P) = 0;
END;
  JFLG=1;
  CALL DVM$IO;
  JFLG=0;
  D1=(DEF(0)/16)*1000;
   D2=(DEF(0) AND 17Q)*100;
  D3=(DEF(1)/16)*10;
   D4=DEF(1) AND 17Q;
  LASTR=D1+D2+D3+D4;
   SSABS=(LASTR+4)/8;
   NTS=TS(0)*10+TS(1);
     B1(2)=72Q;
     B1(5)=72Q;
     B1(8)=40Q;
     B1(9)=40Q;
RETURN;
END SETUP;
 /*  DIGITAL VOLT METER SUBROUTINE...................................*/
 /*        DATA IS GATHERED WITHE THE MOST SIGNIFICANT BCD DIGIT
   PLACED IN DVM(0), AND THE LEAST SIGNIFICANT DIGIT IN DVM(2).
```

```
DECLARE (I,M) BYTE,
   DEF(3) BYTE,    /*  PRESENT DEFRACTOMETER READING  */
   DEFSL(3) BYTE,  /*  LAST DEFRACTOMETER READING  */
   SOL(3) BYTE,    /*  SOLUTE LEVEL */
   SS(3) BYTE,     /*  SOLUTE STEP */
   TS(3) BYTE,     /*  TIME BETWEEN SAMPLES 00-99 MINUTES  */
  BI(10) BYTE,
   SLASH LITERALLY '257Q',
   STBL BYTE,     /* 1=STABLIZED DU4 READING, 0=ERROR  */
   AA LITERALLY '301Q',
   BB LITERALLY '302Q',
   CC LITERALLY '303Q',
  /*  NEW VALVE CONTROL TECHNIQUE  */
  /*     PORT 32 BITS 1 AND 0(LSB'S)=VALVE #1
         PORT 32 BITS 3 AND 2        =VALVE #2
         PORT 32 BITS 5 AND 4        =VALVE #3
         PORT 32 BITS 7 AND 6(MSB'S)=VALVE #4
         PORT 33 BITS 1 AND 0(LSB'S)=VALVE #5
           VALVE STATE A=10
           VALVE STATE B=01
           VALVE STATE C=00
 VALVE NUMBER           1      2      3      4      5
  STATE
 INITIAL                C      C      B      B      C
    A                   B      B      A      B      A
    B                   C      B      C      A      A
    C                   B      A      C      A      A      */
   INIT32 LITERALLY '01010000B',
   INIT33 LITERALLY '00000000B',
   AA32 LITERALLY '01100101B',
   AA33 LITERALLY '00000010B',
   BB32 LITERALLY '10000100B',
   BB33 LITERALLY '00000010B',
   CC32 LITERALLY '10001001B',
   CC33 LITERALLY '00000010B',
     NTS BYTE,
   F$R BYTE, /* 1=FOWARD (ADSORBTION), 0=DESORBTION.  */
   DU4 BYTE,
   (D1,D2,D3,D4,DT,LASTR) ADDRESS,
   IFLG BYTE,
    (M4,CTR) BYTE,
   (PV,LV) ADDRESS,
   DELTA BYTE,
   (A1,A2,SSABS,STBLCHK) BYTE,
   STATE BYTE,  /* KEEPS PRESENT STATE OF VALVES. */
    FLAG BYTE,
   JFLG BYTE,
   POS LITERALLY '255Q',
   NEG LITERALLY '255Q',
   E BYTE,  /* E=1 IF ENTERING DESB., E=2 IF PROG. FINISHED. */
   POLR BYTE;
/*  TTY I/O WAITING SUBROUTINES  ....................................*/
/*  IT IS NECESSARY TO CHECK THIS BIT SO THE PROCESSOR
WILL KNOW WHEN A CHARACTER HAS BEEN PUNCHED ON THE TTY.
```

```
THE SECOND BIT IS CHECKED TO SEE IF THE TTY IS READY TO
ACCEPT THE NEXT CHARACTER.                                     */
INWAIT:  PROCEDURE;
      DO WHILE NOT ROL (INPUT (1),8);
      END;
      RETURN;
END INWAIT;
/*000*/
OUTWAIT:  PROCEDURE;
      DO WHILE NOT ROR (INPUT (1),1);
      END;
      RETURN;
END OUTWAIT;
/*  CR AND LF SUBROUTINE  .......................................*/
/*      THIS OUTPUTS THE PROPER CHARACTER TO THE TTY FOR A
 CARRIAGE RETURN AND A LINE FEED.                                  */
CRLF:  PROCEDURE;
      CALL OUTWAIT;
      OUTPUT (8) = 150;  /* CR  */
      CALL OUTWAIT;
      OUTPUT (8) = 212Q; /* LF  */
      RETURN;
END CRLF;
/*  250 MS WAITING SUBROUTINE  ..................................*/
/*      THIS IS A GENERAL UTILITY SOBROUTINE.  FOR THIS
   PROGRAM IT IS USED TO ALLOW THE HP DVM DATA TO SETTLE
    AFTER ENCODING, BEFORE READING THE NEW VALUE.              */
QTRSSEC:  PROCEDTRE;
DECLARE II BYTE;
      DO II = 1 TO 10;
         CALL TIME (250);
      END;
      RETURN;
END QTRSSEC;L980  /*   SETUP SUBROUTINE.............................................*/
/*        SETUP IS THE FIRST ROUTINE CALLED UP UNDER PROGRAM
   CONTROL.  IT ARKS THE OPERATOR FOR THE VALUES NAMED:
   SOL= (SOLUTE LEVEL THE PROGRAM IS TO START AT)
   SS= (SOLUTE STEP, OR WHAT PERCENTAGE OF 100 WE WISH
        THE STEPPING MOTORS TO JUMP EACH TIME.)
   TS= (NUMBER OF MINUTES TO WAIT BEFORE TAKING NEXT DATA POINT.)
   A SLASH MARK MEANS (VALUE IS OK, CONTINUE)                      */
DVMSIO:  PROCEDURE;
DECLARE (J,K) BYTE, PER LITERALLY '256Q',
      A(3) BYTE,
      B(3) BYTE,
      SUM(3) BYTE,
      (SUM1, SUM2, SUM3) BYTE,
      (A1, A2, A3) BYTE;
/*  GATHERING 10 VALUES FROM DVM FOR AVERAGE.  */
   DEFEL(0)=DEF(0);
   DEFEL(1)=DEF(1);
   DEFEL(2)=DEF(2);
S1:  DO J = 0 TO 9;
        OUTPUT (16) = 1;
        CALL QTRSSEC;
```

```
1200          CALL TIME(250);
1210          CALL TIME(250);
1220          CALL TIME(250);
1230         DO K = 0 TO 2;
1240             DO CASE K;
1250                  A(K) = 0;
1260                  A(K) = INPUT (18);                /* HUNDREDS  */
1270                  A(K) = INPUT (19);                /* UNITS     */
1280             END;
1290             A(K) = SHL(A(K),4);
1300             DO CASE K;
1310                  B(K) = INPUT (17);                /* THOUSANDS */
1320                  B(K) = INPUT (18);                /* TENS      */
1330                  B(K) = INPUT (19);                /* SUB-UNITS */
1340            END;
1250            B(K) = SHR(B(K),4);
1360            A(K) = A(K) + B(K);
1370       END;
1380       HF J <> 0 THEN GO TO ADD;
1390       DO K = 0 TO 2;
1400            SUM (K) = A(K);
1410       END;
1420       GO TO NOJ;
1430 /*  TAKING THE AVERAGE DVM READING   */
1440 ADD:  A1 = A(0);
1450       A2 = A(1);
1460       A3 = A(2);
1470       SUM1 = SUM(0);
1480       SUM2 = SUM(1);
1490       SU43 = SUM(2);
1500 SUM(2) = DEC(A3 + SUM3);    /*  DECIMAL ADDITION  */
1510 SUM(1) = DEC(A2 PLUS SUM2);L1520 SUM(0) = DEC(A1 PLUS SUM1);
1530 NOJ:
1540      END;
1550 /*  NOW PLACE THE AVERAGE DVM READING INTO DEF. */
1560 DO J = 0 TO 2;
1570      DEF(J) = SUM(J);
1580 END;
1590    LV=PV;
1600    D1=(DEF(0)/16)*1000;
1610    D2=(DEF(0) AND 17Q)*100;
1620    D3=(DEF(1)/16)*10;
1630    D4=DEF(1) AND 17Q;
1640    PV=D1+D2+D3+D4;
1650 /*  OUTPUT SUM ONTO TTY.  */
1660   IF JFLG=1 THEN GO TO DONE;
1670 DO J = 0 TO 2;
1680      B(J) = SUM(J);   /*VERY NECESSARY STEPS AFTER EXIT FROM     */
1690      SUM(J) = B(J);   /*   THE DEC COMMAND. */
1700      B(J) = SHR(B(J),4);
1710      B(J) = B(J) + 260Q;
1720      CALL OUTWAIT;
1730      OUTPUT (8) = B(J);
1740           IF J = 0 THEN
1750           DO;
```

```
            CALL OUTWAIT;
            OUTPUT (8) = PER;
        END;
        SUM(J) = SHL(SUM(J),4);
        SUM(J) = SHR(SUM(J),4);
        SUM(J) = SUM(J) + 2600;
        CALL OUTWAIT;
        OUTPUT (8) = SUM(J);
 END;
 CALL OUTWAIT;L1960 OUTPUT (8) = SLASH;
  DONE: RETURN;
 END DVMSIO;
   SETUP: PROCEDURE;
  DECLARE P BYTE,
         A(3) BYTE,
         QL DATA ('SOL= '),
         QS DATA ('SS= '),
          QT DATA ('TS= '),
            MES DATA ('SWITCH VALVE CONTROLLER TO REMOTE');
  /*..........  SOLUTE LEVEL CHANGES  ......*/
  /* PREPARE CORE VALUE OF SOL FOR TTY */
   OUTPUT(26)=INIT32;
   OUTPUT(27)=INIT33;
   DO P=0 TO LAST(MES);
      CALL OUTWAIT;
      OUTPUT(8)=MES(P);
    END;
    CALL CRLF;
    CALL CRLF;
 SOL(2) = SOL(1);
 SOL(1) = SHR(SOL(1),4);
 SOL(2) = SHL(SOL(2),4);
 SOL(2) = SHR(SOL(2),4);
 DO P = 0 TO 2;
       SOL(P) = SOL(P) + 2600;
 END;
 IO$SOL:  CALL CRLF;
 DO P = 0 TO LAST(QL);
       CALL OUTWAIT;
       OUTPUT (8) = QL(P);
 END;
 DO P = 0 TO 2;
       CALL OUTWAIT;
       OUTPUT (8) = SOL(P);
 END;
 CALL CRLF;
 CALL CRLF;
 DO P = 0 TO 2;
       CALL INWAIT;
       A(P) = INPUT (0);
       CALL OUTWAIT;
       OUTPUT (8) = A(P);
       IF A(P) = SLASH THEN GO TO SOL$OK;
 END;
 DO P = 0 TO 2;
```

```
      SOL(P) = A(P);
END;
GO TO IO$SOL;
/*  RETURN FINAL VALUE OF SOL TO CORE FORMAT.  */
SOL$OK:
DO P = 0 TO 2;
     SOL(P) = SOL(P) - 260Q;
END;
SOL(1) = SHL(SOL(1),4);
SOL(1) =SOL(1) + SOL(2);
/* SOL IS NOW STORED IN BCD FORM IN SOL(0) AND SOL(1) */
/* PREPARES SS FROM CORE FOR TTY I/O  */
SS(1) = SS(0);
SS(0) = SHR(SS(0),4);
SS(1) = SHL(SS(1),4);
SS(1) = SHR(SS(1),4);
SS(0) = SS(0) + 260Q;
SS(1) = SS(1) + 260Q;
/*..........  SOLUTE STEP CHANGES  .........*/
/* DIRECTION OF STEPPING MOTORS CHECKED */
STEPL:  FSR = INPUT (17);
     FSR = SHL (FSR,7);
     FSR = SHR (FSR,7);
     IF FSR = 1 THEN GO TO ADS;
     POLR = NEG;
     GO TO NXT1;
ADS:  POLR = POS;
NXT1:  CALL CRLF;
CALL CRLF;
DO P = 0 TO LAST(QS);
     CALL OUTWAIT;
     OUTPUT (8) = QS(P);
END;
/* I/O OF PREVIOUS VALUE  */
CALL OUTWAIT;
OUTPUT (8) = POLR;
DO P = 0 TO 1;
     CALL OUTWAIT;
      OUTPUT (8) = SS(P);
END;
CALL CRLF;
CALL INWAIT;
A(0) = INPUT (0);
CALL OUTWAIT;
OUTPUT (8) = A(0);
IF A(0) = POLR THEN GO TO SGN$OK;  /* GOES TO SS INPUT */
IF A(0) = SLASH THEN GO TO SS$OK;  /* TO NEXT PARAMETER SETUP */
OUTPUT (25) = 1;   /* CHANGES STEPPING MOTOR DIRECTION **/
SGN$OK:
DO P = 0 TO 1;
     CALL INWAIT;
     SS(P) = INPUT (0);
CALL OUTWAIT;
OUTPUT (8) = SS(P);
END;
```

```
  THE EIGHT BIT BYTE, (ONE WORD), HOLDS TWO BCD DIGITS.          */
CHK: PROCEDURE;
 IFLG=0;
D1=DEF(0)/16*1000;
D2=(DEF(0) AND 17Q)*100;
D3=DEF(1)/16*10;
D4=DEF(1) AND 17Q;
DT=D1+D2+D3+D4;
 IF LASTR+SSABS<DT THEN IFLG=1;
 IF LASTR>DT+SSABS THEN IFLG=1;
 IF IFLG=0 THEN GO TO DONE;
[illegible]=1;
DONE:  RETURN;
END CHK;
WAITIO: PROCEDURE;
        DECLARE A(4) BYTE,
        (NS,NM,NH) BYTE,
        (TVNM,TVNS,TVNH,J)BYTE;
        OUTPUT(20)=1;
        CALL QIR$SEC;
        A(1)=INPUT(21);
        A(2)=INPUT(22);
        A(3)=INPUT(23);
        NH=(A(1) AND 17Q)+(A(1)/16)*10;
        NM=(A(2) AND 17Q)+(A(2)/16)*10;
        NS=(A(3) AND 17Q)+(A(3)/16)*10;
        TVNH=NH;
        TVNM=NM;
        TVNS=NS+NTS;
        IF TVNS>59 THEN
         DO;
           TVNS=TVNS-60;
           TVNM=TVNM+1;
         END;
        IF TVNM>59 THEN
        DO;
         TVNM=TVNM-60;
         TVNH=TVNH+1;
        END;
        B1(0)=(A(1)/16)+60Q;
        B1(1)=(A(1) AND 17Q)+60Q;
        B1(3)=(A(2)/16)+60Q;
        B1(4)=(A(2) AND 17Q)+60Q;
        B1(6)=(A(3)/16)+60Q;
        B1(7)=(A(3) AND 17Q)+60Q;
        CALL DVM$IO;
        DO J=0 TO 9;
         CALL OUTWAIT;
         OUTPUT(8)=B1(J);
        END;
        I=I+1;
        IF I>3 THEN
        DO;
         I=0;
         CALL CRLF;
```

```
               END;
     LOOP:     OUTPUT(20)=1;
               CALL QTR$SEC;
               A(1)=INPUT(21);
               A(2)=INPUT(22);
               A(3)=INPUT(23);
               NH=(A(1) AND 17Q)+(A(1)/16)*10;
               NM=(A(2) AND 17Q)+(A(2)/16)*10;
               NS=(A(3) AND 17Q)+(A(3)/16)*10;
               IF NH<TVNH THEN GO TO LOOP;
               IF NH>TVNH THEN GO TO DONE;
               IF NM<TVNM THEN GO TO LOOP;
               IF NM>TVNM THEN GO TO DONE;
               IF NS<TVNS THEN GO TO LOOP;
     DONE:     RETURN;
               END WAITIO;
     ERROR:    PROCEDURE;
               DELTA=(PV+64)/128;
               IF LV>DELTA+PV THEN GO TO DONE;
               IF LV+DELTA<PV THEN GO TO DONE;
               STBL=1;
     DONE:     RETURN;
               END ERROR;
     /*  MARK SUBROUTINE................................................*/
     /*         THIS ROUTINE PRINTS THE PRESENT STATE OF THE PROGRAM
       ON THE TTY IN THE FORM "STATE = (FLAG) ".  WHERE FLAG IS
       EITHER A, B, OR C.  THE ACTUAL ASCII CHARACTERS ARE STORED
       IN THE LITERALS LABELED AS AA, BB, AND CC.                    */
    MARK:  PROCEDURE;
    DECLARE J BYTE,
        FLG DATA ('STATE= ');
    DO J = 0 TO 3;
        CALL CRLF;
    END;
    DO J = 0 TO LAST(FLG);
        CALL OUTWAIT;
        OUTPUT (8) = FLG(J);
    END;
    CALL OUTWAIT;
    OUTPUT (8) = FLAG;
    DO J = 0 TO 1;
        CALL CRLF;
    END;
    RETURN;
    END MARK;
     /*  ADD$SS SUBROUTINE..............................................*/
     /*        IF THE PROGRAM IS ADSORBING, F$R WILL BE EQUAL TO ONE
       AND THE VALUE OF SS (GIVEN BY OPERATOR) WILL BE ADDED TO SOL.
       IF DESORBING, SS WILL BE SUBTRACTED FROM SOL.                 */
    ADD$SS:  PROCEDURE;
    DECLARE (A1,A2,B1,B2) BYTE;
    /* CHECK STEPPING MOTOR DIRECTION */
    F$R = INPUT (17);
    F$R = SHL(F$R,7);
    F$R = SHR(F$R,7);
```

```
IF FSR = 1 THEN GO TO ADDS;
 /*  1 = ADSORBTION, 0 = DESORBTION                                      */
POLR = NEG;
GO TO GOSUN;
ADDS:
      POLR = POS;
GOSUN:
      A2 = SOL(1);
      A1 = SOL(0);
      B2 = SS(0);
      B1 = 0;
      IF FSR = 0 THEN
 /*   SUBTRACTING SS FROM SOL                                            */
          DO;
      B2 = 99H - B2;
      B2 = B2 + 1;
      B1 = 99H - B1;
      B1 = B1 +1;
      A2 = DEC(A2 + B2);
      A1 = DEC(A1 + B1);
          END;
      IF FSR = 0 THEN GO TO UKPT;
 /*   ADD SS TO SOL  */
      A2 = DEC(A2 + B2);
      A1 = DEC(A1 PLUS B1);
  UKPT:   SOL(1) = A2;  /* 4 VERY IMPORTANT EQUALITIES. */
      A2 = SOL(1);
      SOL(0) = A1;
      A1 = SOL(0);
RETURN;
END ADDSSS;
/*...............................................................*/
/*...............................................................*/
/*...................MAIN PROGRAM................................*/
              CALL SETUP;
              STATE=0;
              FLAG=AA;
              CALL MARK;
              OUTPUT(26)=AA32;
              OUTPUT(27)=AA33;
              IF SOL(1)>0 THEN FSR=0;
      START:  STBLCHK=0;
              STBL=0;
              I=0;
      LOOP1:  CALL WAITIO;
              CALL CHK;
              IF STBLCHK=0 THEN GO TO LOOP1;
      LOOP2:  CALL WAITIO;
              CALL ERROR;
              IF STBL=0 THEN GO TO LOOP2;
              STBL=0;
              CALL WAITIO;
              CALL ERROR;
              IF STBL=0 THEN GO TO LOOP2;
              DO M=0 TO 2;
```

```
04MAY77      16:04:06     PICARD2@              SYSTEMS REAL-TIME MONITOR-5.0                    PAGE  10

5070              CALL  WAITIO;
5080            END;
5090            STBLCHK=0;
5100            STBL=0;
5110            D1=(DEF(0)/16)*1000;
5120            D2=(DEF(0) AND 17Q)*100;
5130            D3=(DEF(1)/16)*10;
5140            D4=DEF(1) AND 17Q;
5150            LASTR=D1+D2+D3+D4;
5160            SSABS=(LASTR+8)/16;
5170            IF STATE=0 THEN GO TO STATEB;
5180            IF SOL(0)=0 THEN GO TO TEST;
5190    NEXT:   IF STATE=1 THEN GO TO STATEC;
5200    STATEB: STATE=1;
5210            FLAG=BB;
5220            CALL MARK;
5230            OUTPUT(26)=BB32;
5240            OUTPUT(27)=BB33;
5250            DO CIR=1 TO 25;
5260              CALL QIRSEC;
5270            END;
5280            OUTPUT(24)=1;
5290            CALL ADD$SS;
5300            IF SOL(0)>0 THEN FSR=0;
5305            OUTPUT(25)=1;
5310            GO TO START;
5320    TEST:   IF SOL(1)>0 THEN GO TO NEXT;
5330            STATE=0;
5340            FLAG=AA;
5350            CALL MARK;
5360            OUTPUT(26)=AA32;
5370            OUTPUT(27)=AA33;
5380            HALT;
5390    STATEC: STATE=2;
5400            FLAG=CC;
5410            CALL MARK;
5420            OUTPUT(26)=CC32;
5430            OUTPUT(27)=CC33;
5440            GO TO START;
5450            EOF;


READY
```

APPENDIX B
CUT-BAR EXPERIMENT PLM PROGRAM

```

           DECLARE (N,VAL,MIN,CH) BYTE,
           EXP BYTE,
           (I,J,K,TENS,ONES,FIN) BYTE,
           (NM1,NM2,NM3,TENX,HUNX) BYTE,
           PRINT(5) BYTE,
           CHMES DATA ('CH NO '),
           MVMES DATA (' MV'),
           VIDAROUT(8) BYTE,VIDARIN(8) BYTE,
           INITMES1 DATA ('ENTER NUMBER OF MINUTES BETWEEN SAMPLES'),
           TEMP(5) BYTE,
           INITMES2 DATA ('ENTER LAST CHANNEL NUMBER TO BE SAMPLED');
  TTYINWAIT:  PROCEDURE;
           DO WHILE NOT ROL(INPUT(1),8);
           END; RETURN; END TTYINWAIT;
  TTYOUTWAIT:  PROCEDURE;
           DO WHILE NOT ROR(INPUT(1),1);
           END; RETURN; END TTYOUTWAIT;
  SIOINWAIT:  PROCEDURE;
           DO WHILE NOT ROL(INPUT(7),8);
           END; RETURN; END SIOINWAIT;
  SIOOUTWAIT:  PROCEDURE;
           DO WHILE NOT ROR(INPUT(7),1);
           END; RETURN; END SIOOUTWAIT;
  CRLF:    PROCEDURE;
           CALL TTYOUTWAIT; OUTPUT(8)=215Q;
           CALL TTYOUTWAIT; OUTPUT(8)=212Q;
           RETURN; END CRLF;
  PRINTOUT:  PROCEDURE (NUM); DECLARE NUM BYTE, A(3) BYTE,J BYTE;
           A(0)=(NUM AND 7Q)+260Q; A(1)=SHR(NUM,3);
           A(1)=(A(1) AND 7Q)+260Q; A(2)=SHR(NUM,6);
           A(2)=(A(2) AND 7Q)+260Q; CALL CRLF;
           DO J=0 TO 2;
           CALL TTYOUTWAIT; OUTPUT(8)=A(2-J);
           END; RETURN; END PRINTOUT;
  MINUTE.  PROCEDURE;
           DECLARE I ADDRESS;
           DO I=1 TO 27056;
             CALL TIME(10);
           END;
           RETURN;END MINUTE;
  VALIN:   PROCEDURE;
           DO N=0 TO 4;
             CALL TTYINWAIT;
             TEMP(N)=INPUT(0);
             IF TEMP(N)=215Q THEN GO TO NEXT;
             CALL TTYOUTWAIT;
             OUTPUT(8)=TEMP(N);
             TEMP(N)=TEMP(N)-260Q;
           END;
  NEXT:    CALL CRLF;
           NM1=N-1; NM2=N-2; NM3=N-3;
           VAL=TEMP(NM1);
           IF N=1 THEN GO TO RET;
           TENX=10*TEMP(NM2);
```

```
            VAL=VAL+TENX;
            IF N=2 THEN GO TO RET;
            HUNX=100*TEMP(NM3);
            VAL=VAL+HUNX;
     RET:   RETURN;
            END VALIN;
     START: VIDAROUT(0)=322Q;
            VIDAROUT(1)=260Q;
            VIDAROUT(2)=260Q;
            VIDAROUT(3)=260Q;
            CALL CRLF;
            DO N=0 TO LAST(INITMES1);
              CALL TTYOUTWAIT;
              OUTPUT(8)=INITMES1(N);
            END; CALL CRLF; CALL CRLF;
            CALL VALIN;
            MIN=VAL;
            DO N=0 TO LAST(INITMES2);
              CALL TTYOUTWAIT;
              OUTPUT(8)=INITMES2(N);
            END; CALL CRLF; CALL CRLF;
            CALL VALIN;
            CH=VAL;
            TENS=CH/10;
            ONES=CH-(TENS*10);
     RES:   DO N=0 TO 3;
              CALL SIOOUTWAIT;
              OUTPUT(15)=VIDAROUT(N);
            END;
            DO N=0 TO 7;
              CALL SIOINWAIT;
              VIDARIN(N)=INPUT(6);
            END;
            DO N=0 TO 100;
              CALL TIME(250);
            END;
            IF VIDARIN(0)=260Q THEN GO TO RES;
      LOOP: CALL CRLF; CALL CRLF; CALL CRLF;
             FIN=9;
             DO I=0 TO TENS;
               VIDAROUT(2)=I+260Q;
               IF I=TENS THEN FIN=ONES;
               DO J=0 TO FIN;
                 VIDAROUT(3)=J+260Q;
     P1:
                 DO K=0 TO 3;
                   CALL SIOOUTWAIT;
                   OUTPUT(15)=VIDAROUT(K);
                 END;
     P2:
                 DO K=0 TO 7;
                   CALL SIOINWAIT;
                   VIDARIN(K)=INPUT(6);
                 END;
     P3:
             EXP=VIDARIN(0)-260H;
             VIDARIN(0)=240H;
             IF VIDARIN(1)=253H THEN VIDARIN(1)=240H;
      P4:
             DO K=2 TO 7;
               IF VIDARIN(K)=260H THEN VIDARIN(K)=240H;
               ELSE GO TO CONT;
             END;
      CONT:  DO K=0 TO (11-EXP);
              PRINT(K)=VIDARIN(K);
             END;
      P5:
             PRINT(11-EXP)=256H;
             DO K=(12-EXP) TO 8;
              PRINT(K)=VIDARIN(K-1);
             END;
      P6:
             DO K=0 TO LAST(CHMES);
               CALL TTYOUTWAIT;
               OUTPUT(8)=CHMES(K);
             END;
      P7:
             DO K=1 TO 3;
               CALL TTYOUTWAIT;
               OUTPUT(8)=VIDAROUT(K);
             END;
      P8:
             DO K=0 TO 8;
               CALL TTYOUTWAIT;
               OUTPUT(8)=PRINT(K);
             END;
      P9:
             DO K=0 TO LAST(MVMES);
               CALL TTYOUTWAIT;
               OUTPUT(8)=MVMES(K);
             END;
             CALL CRLF;
           END;
         END;
      P10:
         DO K=1 TO (MIN-1);
           CALL MINUTE;
         END;
         GO TO LOOP;
         EOF

READY
```

APPENDIX C
DEVELOPMENT SYSTEM PDP-11 ASSEMBLY LANGUAGE PROGRAM

```
        .MCALL  ..V2..,.REGDEF
        .MCALL  .EXIT
        ..V2..
        .REGDEF
        .MCALL  .CSIGEN,.READW,.PRINT,.TTYIN
START:  BIS     #400,@#167772
        .CSIGEN #HAND,#DEXT,#0
        MOV     R0,BUFF
        CLR     INBLK
        MOV     #LIST,R5
        CLR     COL
        .PRINT  #MSG
        BIC     #10000,44
        .TTYIN
        BIC     #100,@#167564
        CLR     R1
        CLR     R2
LOOP:   JSR     PC,TYPE
        INC     R1
        CMP     R1,#200
        BLT     LOOP
        CLRB    B1
        COMB    B1
READ:   .READW  R5,#3,BUFF,#256.,INBLK
        ADD     R0,R0
        MOV     BUFF,R1
PUNCH:  MOVB    (R1)+,R2
        MOVB    B3,B4
        MOVB    B2,B3
        MOVB    B1,B2
        MOVB    R2,B1
        CMPB    B4,#1
        BNE     CONT
        CMPB    B3,#0
        BNE     CONT
        CMPB    B2,#6
        BNE     CONT
        CMPB    B1,#0
        BEQ     DONE
CONT:   JSR     PC,TYPE
        DEC     R0
        TST     R0
        BGT     PUNCH
        INC     INBLK
        BR      READ
DONE:   MOV     #4,R0
LOOP1:  JSR     PC,TYPE
        MOVB    (R1)+,R2
        DEC     R0
        TST     R0
        BGT     LOOP1
        CLR     R0
        CLR     R2
LOOP3:  JSR     PC,TYPE
```

```
         INC     R0
         CMP     R0,#400
         BLT     LOOP3
         BIS     #100,@#177560
         BIS     #100,@#177564
         .TTYIN
         JMP     START
TYPE:    CLR     R3
LOOP4:   INC     R3
         CMP     #4100,R3
         BGT     LOOP4
         MOVB    R2,@#167772
         BIC     #400,@#167772
         BIS     #400,@#167772
         RTS     PC
MSG:     .ASCII  /TURN ON TAPE PUNCH/
         .BYTE   15,12
         .ASCIZ  /AND STRIKE ANY KEY./
         .EVEN
B1:      .BYTE   0
B2:      .BYTE   0
B3:      .BYTE   0
B4:      .BYTE   0
DATA:    .BYTE   0
         .EVEN
TEMP:    .WORD   0
COL:     .WORD   0
LIST:    .BLKW   5
INBLK:   0
BUFF:    0
DEXT:    .RAD50  /LDA/
HAND:    .
         .END    START
```

APPENDIX D
USERS MANUAL FOR THE DATA ACQUISITION IN THE LHMEL FACILITY

SECTION 1

PREVIEW

The Laser Hardened Materials Evaluation Laboratory (LHMEL) facility includes a dedicated Hewlett-Packard 9640A Multiprogramming System and a communication link to the SEL86 computer system. The OZ software system is a collection of programs residing both on the Hewlett-Packard system and the SEL86 system that acquires experimental data and performs real time analysis of data, permitting the LHMEL operator to rapidly alter experimental parameter values for best effect.

The experimental data that are acquired are:

1) Various experimental pre- and post-run operating parameter values (entered by the operator).
2) The run startup time.
3) The laser power (20 Hz sample rate).
4) The air temperature (20 Hz sample rate).
5) The static air temperature (20 Hz sample rate).
6) The total air pressure (20 Hz sample rate).
7) The time the laser shutter opens and closes.
8) The time the burn-through wire in front of the target is cut by the laser beam.
9) The time the burn-through wire behind the target is cut by the laser beam.

The principle component of the Hewlett-Packard system is a 21MX computer, operating under the Hewlett-Packard supplied RTE-C operating system. To begin an experimental run the operator uses RTE-C to activate the program OZ and enters the pre-run operating parameter values that have changed since the last experimental run. The operator then has the program OZGO activated to control the acquisition of the experimental data. Initialization of the laser control sequencer by the operator starts the program data acquisition sequence.

When the run terminates, the operator can use RTE-C to activate the program OZPST to enter the post-test operating parameter values. If desired, the operator can have the program DADMP activated to inspect the raw data acquired during the run. If the data are acceptable, the SEL86 program can be activated. Within this program the operator sends the raw data over the communication link to the SEL86 computer and initiates the execution of the analysis program SELOZ (on the SEL86). SELOZ computes the following:

1) The run time interval (time of shutter closure minus run startup time).
2) The shutter time interval (time of shutter closure minus time shutter opened).
3) The burn-through time interval (back burn-through wire time minus front burn-through wire time).
4) The laser power curve over the shutter time interval.
5) The laser power threshold time interval (the time when the laser power becomes less than, or equal to, the laser power threshold going down minus the time when the laser power becomes greater than, or equal to, the laser power threshold going up).
6) The maximum, minimum, and average values of laser power over the shutter time interval.
7) The maximum and average values of laser power over the laser power threshold time interval.
8) The total energy over the laser power threshold time interval.
9) The diameter and area of the beam on-target.
10) The laser power on-target curve over the shutter time interval.
11) The average power on-target over the shutter time interval.
12) The average power density on-target over the shutter time interval.
13) The energy density on-target over the shutter time interval.
14) The average power on-target over the burn-through time interval.
15) The average power density on-target over the burn-through time interval.
16) The energy density on-target over the shutter time interval.
17) The total temperature curve over the shutter time interval.
18) The minimum temperature over the shutter time interval.
19) The minimum temperature over the shutter time interval.
20) The RMS value of static air pressure over the shutter time interval.
21) The total air pressure curve over the shutter time interval.
22) The RMS value of total air pressure over the shutter time interval.
23) The MACH number.

The data and analytical results are conveniently formatted and printed on a high-speed printer at the SEL86 site. They are also archived on magnetic tape. Programs are provided to produce CALCOMP quality plots of the power on-target, total temperature, static pressure, and total pressure curves either in real-time or from the archived data. A summary of the analysis is printed for the operator on the LHMEL teletype. The operator can use this summary to select the experimental conditions for the next run.

SECTION 2
PROGRAM SUMMARIES

2.1 INTRODUCTION

There are five sets of routines within the OZ system:

1) RTE-C: the operating system on the 21MX computer.
2) The OZ programs: the set of 21MX resident routines that acquire the pre-run and post-run operating parameter values and experimental data.
3) SEL86: the 21MX resident routines that communicate with the SEL86 computer, sending data to it, starting programs on it, and receiving ASCII data from it for output on the LHMEL teletype.
4) SELOZ: the set of data analysis programs resident on the SEL86.
5) TPOUT: the program resident on the SEL86 that extracts data archived on magnetic tape and offers a plotting option.

Each of these sets of routines will be briefly described in the following sections. A more detailed description is found in Reference 15.

2.2 RTE-C

RTE-C is Hewlett-Packard's acronym for Real-Time Core-Based Software System. The procedure for initializing this system is found in Section 3.2. Once RTE-C is running, and any time it has control of the teletype, entering any character through the teletype will cause RTE-C to respond by typing an asterisk (*). At this time, the operator is free to enter any command described in the RTE-C operating manual.*

The operator is required to enter the time of day every time RTE-C is initialized. The format for this command is:

```
TM,day,h,min,sec
```

where day is a three-digit day-of-the-year (see Table 2-5 of the RTE-C operating manual), and h,min,sec is the current time on a 24-hour clock.

*Real-Time Executive Core-Based Software System Programming and Operating Manual, 1 August 1978, Part No. 29101-93001.

Among the many functions of RTE-C is the control of the operator's access to the programs stored in the 21MX core. In order to execute a program, the operator types:

```
ON,pgnam
```

where pgnam is the name of the program to be executed.

Error messages from RTE-C are discussed on page 3-12 of the operating manual.

It should be noted that if a wrong character is typed "control-A" can be used to delete that character. RTE-C types a left-facing arrow to indicate that the character has been deleted. To delete an entire line, type "rubout". RTE-C types a back slash to indicate that the line has been deleted.

2.3 THE OZ ROUTINES

2.3.1 Introduction

The following OZ routines can be called using the RTE-C ON,pgnam command:

OZ
OZGO
DADMP
OZPST.

In addition, two other routines are called internally by RTE-C:

OZAD
OZUNI.

2.3.2 OZ

The OZ routine permits entry of pre-run operating parameter values. When the program is started by RTE-C, this heading is typed on the LHMEL teletype:

```
OZ, VERSION xx/xx/xx.
```

After this, the system waits for the operator to enter a two-letter code corresponding to a pre-run operating parameter. (These codes are listed in Table I.) The code should be followed by a carriage return.

OZ searches its internal table of codes. If it finds a match it types:

ENTER XX

where XX is an echo of the operator-supplied code. A carriage return is executed and OZ waits for the operator to enter the appropriate parameter value followed by a carriage return.

If no match is found, OZ types:

NO MATCH

followed by a carriage return and waits for the code to be reentered.

Any number of parameters and their values can be entered during an OZ execution, and their order is arbitrary. This permits entry of only those parameters that have changed value since the previous run.

If a parameter value is entered incorrectly, the parameter code and value can be reentered.

All numeric parameter values should be entered in F7.3 format (i.e., in the form 999.999). For alphanumeric parameter values, the length of the field varies with parameters as defined in Table I.

To notify OZ that no more parameters are to be entered, type XX for the parameter code. OZ types:

END OZ

and returns control to RTE-C.

2.3.3 <u>OZGO</u>

The OZGO routine initiates the data acquisition process. Upon entering the routine, all the flags and pointers are initialized and the rear burn-through wire clock is zeroed.* The following message is printed on the teletype:

*Upon completion of a run a zero rear burn-through wire time indicates that the target did not burn through.

TABLE I

PRE-RUN OPERATING PARAMETERS

PARAMETER	CODE	FORMAT	UNITS
sample id	SI	A6	
test number	TN	A4	
distance between mirror and sample	DI	F7.3	cm
HE dome pressure	HD	F7.3	psig
N2 dome pressure	ND	F7.3	psig
CO2 dome pressure	CD	F7.3	psig
laser pressure	LA	F7.3	mm Hg
exhaust pressure	EP	F7.3	mm Hg
high voltage	KV	F7.3	kilovolts
run time	RT	F7.3	sec
HE line pressure	HL	F7.3	psig
N2 line pressure	NL	F7.3	psig
CO2 line pressure	CL	F7.3	psig
WT back pressure	WB	F7.3	psig
WT line pressure	WL	F7.3	psig
WT plenum temperature	WP	F7.3	deg
WT nozzle design, x dim	WX	F7.3	cm
WT nozzle design, y dim	WY	F7.3	cm
initial load	IL	F7.3	kilopounds

READY..AUTO?

The operator should respond by entering a "Y" or an "N" followed by a carriage return.

<u>If a "Y" is entered</u>, the system waits up to 20-seconds for a signal from the laser control system (on channel 0 of the universal interface card data buffer).

(a) If the pulse does not occur within 20-seconds, this message is typed on the teletype:

TIME UP

and control is returned to RTE-C.

(b) If the pulse does occur within the time limit, the OZUNI routine sets the experiment initialize flag IF4 and records the time of the pulse as the "start-up" time. Control is returned to RTE-C.

<u>If an "N" is entered</u>, the experiment initialize flag IF4 is set, the current time is recorded as the "start-up" time, and OZAD and OZUNI are started. Control is returned to RTE-C.

2.3.4 <u>DADMP</u>

The DADMP routine generates a teletype listing of the values of pre-run and post-run parameters, the flags, the pointers, the data arrays, and the time buffers.

The numbers under the "FLAGS AND POINTERS" banner are the values of the flags and pointers in this sequence:

IF4- experiment initialize flag; incremented each time the experiment is initialized during a run.

IF0- front burn-through wire flag, incremented each time a front burn-through wire signal is detected during a run.

IF1- rear burn-through wire flag; incremented each time a rear burn-through wire signal is detected during a run.

IF2- shutter open flag; incremented each time a shutter open signal is detected during a run.

IF3- shutter close flag; incremented each time a shutter close signal is detected during a run.

IFER - previous universal interface data register contents (decimal equivalent).

IRBUF- current universal interface data register contents (octal value).

IFIN - set when flags initialized and reset when run completed.

IPNTR- the number of filled entries in the data arrays (decimal value).

The sequence of numbers labeled "START UP=" down through "SHUTTER CLOSE=" are associated with the clock buffer entries. The order of these entries on a single line is: tens of msec, seconds, minutes, hours, days (all in decimal).

The values in the data arrays are in units of volts.

2.3.5 OZPST

The OZPST routine is used to enter the post-run operating parameter values. When the program is started by RTE-C, the following heading is printed on the LHMEL teletype:

```
OZPST,VERSION XX/XX/XX.
```

At this point the system idles, waiting for the operator to enter one of the two-letter codes corresponding to a post-run operating parameter listed in Table II. The code should be followed by a carriage return.

OZPST searches its internal table of codes. If a match is found, it types

```
ENTER XX
```

where XX is the echo of the operator-supplied code. A carriage return is executed and OZPST waits for the operator to enter the appropriate parameter value followed by a carriage return.

If no match is found, OZPST types

```
NO MATCH
```

followed by a carriage return and waits for the code to be reentered.

All the parameter values are numeric and should be entered in an F7.3 format (i.e., in the form 999.999).

TABLE II

POST-RUN OPERATING PARAMETERS

PARAMETER	CODE	FORMAT	UNITS
HE upstream pressure	HU	F7.3	psig
N2 upstream pressure	NU	F7.3	psig
CO2 upstream pressure	CU	F7.3	psig
laser pressure	LP	F7.3	mm Hg
exhaust pressure	EX	F7.3	mm Hg
current	CR	F7.3	milliamps
WT exhaust pressure	WE	F7.3	psig

To notify OZPST that no more parameters are to be entered, the operator should type "XX" for the parameter code. OZPST types:

END OZPST

and returns control to RTE-C.

2.3.6 OZAD

The purpose of OZAD is to control the A/D converter.

When OZAD is started by OZUNI, it checks the IF4 flag (experiment initialize) to determine if the run has been initialized by the OZGO routine. If it has not, this message is typed on the teletype:

OZAD NOT INIT.

Control is returned to RTE-C.

If the run has been initialized, the IF3 flag (shutter closed) is examined to determine if the run has been completed. If it has, the message

A/D DONE

is typed on the teletype and control is returned to RTE-C. If the run is still in progress, a single reading is taken on each of the first four channels (CH0-CH3) of the A/D converter. These readings are stored in the next available location of the four arrays PMETR(CH0), TTEMP(CH1), SPRES(CH2), and TPRES(CH3). Then the pointer used to indicate the next available location, IPNTR, is incremented and that value compared with the maximum dimension of the four arrays (101). If the arrays are not full, OZAD reschedules itself to execute 50 msec later and control is returned to RTE-C.

If the arrays are full, which occurs five seconds after the run begins, this message is printed on the teletype:

ARRAYS FULL
A/D DONE

Control is returned to RTE-C.

2.3.7 OZUNI

OZUNI is the program that processes the signals input to the Universal Interface Card (12930A). It is started by the OZGO program and it reschedules itself to run every 20 msec until either the run is completed, an error has occurred, or 20 seconds has passed without a "start-up" signal. This program should not be called by the operator using the RTE-C "ON,OZUNI" command.

The Universal Interface Card data channel input register is assumed by OZUNI to be wired as follows:

bit 0:	run start-up signal wire
bit 1:	shutter signal wire
bit 2:	front burn-through signal wire
bit 3:	back burn-through signal wire.

All signals are assumed to be single-ended.

The logical meaning of each of the signals is assumed to be as follows:

Run start-up signal: the occurrence of the 0 to 5v transition indicates start of run. The signal remains at 5v for the duration of the run.

Shutter signal wire: the occurrence of the first 0 to 5v transition after the start of a run indicates the opening of the shutter. Occurrence of the next 5v to 0v transition indicates the closing of the shutter. The signal remains at 5v in between.

Front burn-through signal: the occurrence of the first 0 to 5v transition after the start up of a run indicates that the front burn-through wire has burned through. The signal remains at 5v for the duration of the run.

Back burn-through signal: the occurrence of the first 0 to 5v transition after the start of a run indicates that the back burn-through wire has burned through. The signal remains at 5v for the duration of the run.

Upon the first entry into the program during a run, the FINIT flag is examined to determine if the OZGO program initiated OZUNI. If not, the message is typed on the teletype and control is returned to RTE-C. This prevents the operator from starting OZUNI using the RTE-C "ON,OZUNI" command. Otherwise, each time OZUNI runs, the first thing it does is load the Universal Interface Card's data register into the A register, where it is exclusive-ored with the contents of the data register obtained the last time OZUNI ran. The resulting word has a bit set in every position where the level of the data register has changed since the last time OZUNI ran. This word is processed as shown in the accompanying flowchart in Figure 1. During the course of processing, if a logical error occurs, (e.g., two-level changes in the front burn-through wire signal during the course of one run), the message "ERROR IN UNIV INT" is printed on the teletype and data acquisition for the run is aborted. The source of the error can be determined by running DADMP and examining the values for IFER and IRBUF. If no logical error occurs, OZUNI checks the shutter-closed flag to determine if the shutter-closed event has occurred. If it has, the program terminates. If it hasn't the program reschedules itself to run 20 msec later.

After OZGO turns OZUNI on, OZUNI runs every 20 msec, checking for the start-up signal. OZGO times the event. If the signal is not generated in 20 seconds, OZGO notifies OZUNI through the IFIN flag, and OZUNI does not reschedule itself.

2.4 SEL86

The SEL86 routine controls the user's communication with the SEL 86 computer. SEL86 is brought up by typing:

```
ON,SEL86.
```

The response of the terminal is

```
-SEL TERMINAL SUPPORT SYSTEM, TERMINAL TYØØ -
ENTER FUNCTION CODE, ?, OR CR TO TERMINATE:
```

The informed user will recognize this as the normal mode of entry into the SEL 86 Terminal Support System (TSS). All the options

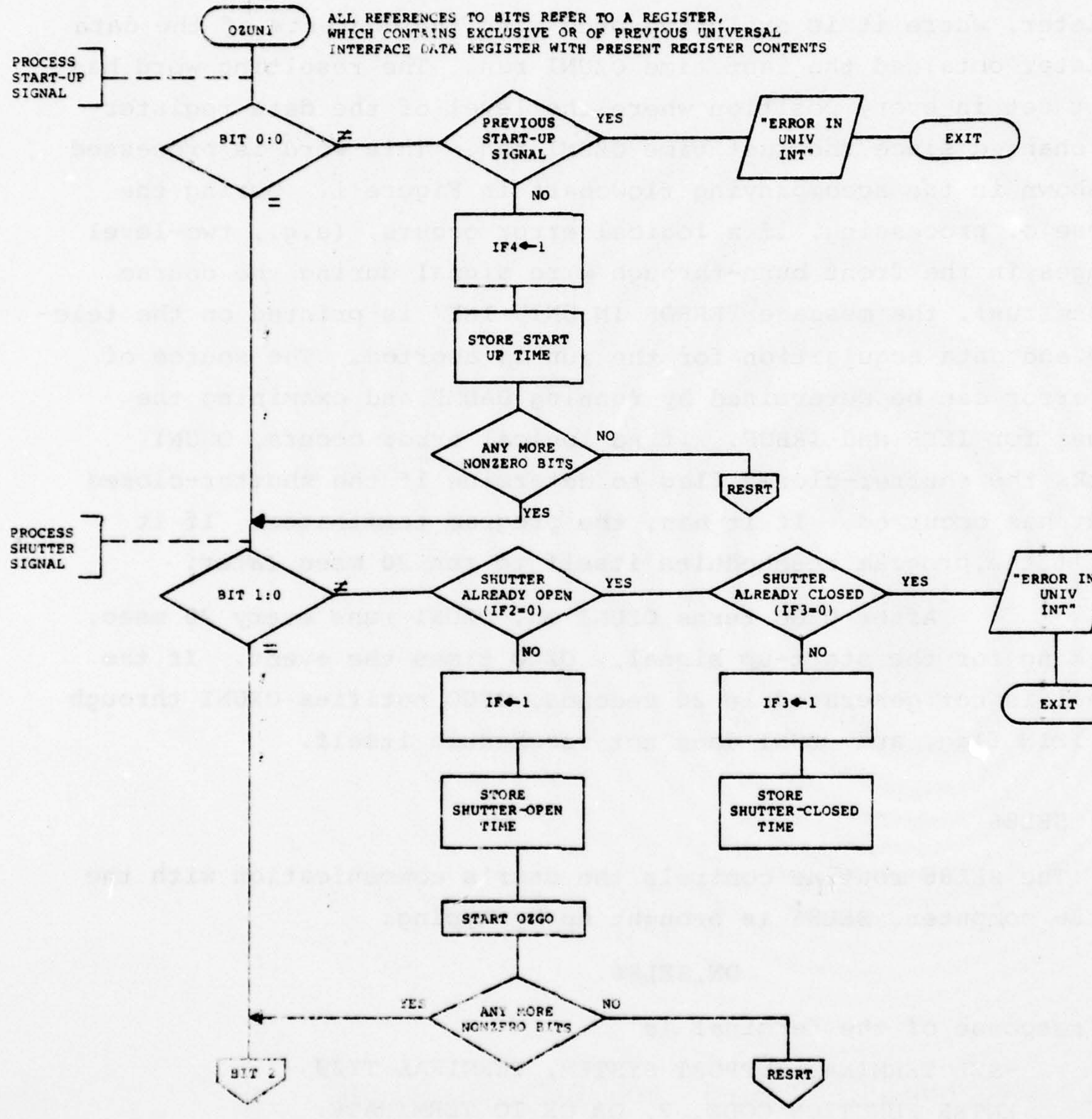


Figure 1. Flow-chart of OZUNI.

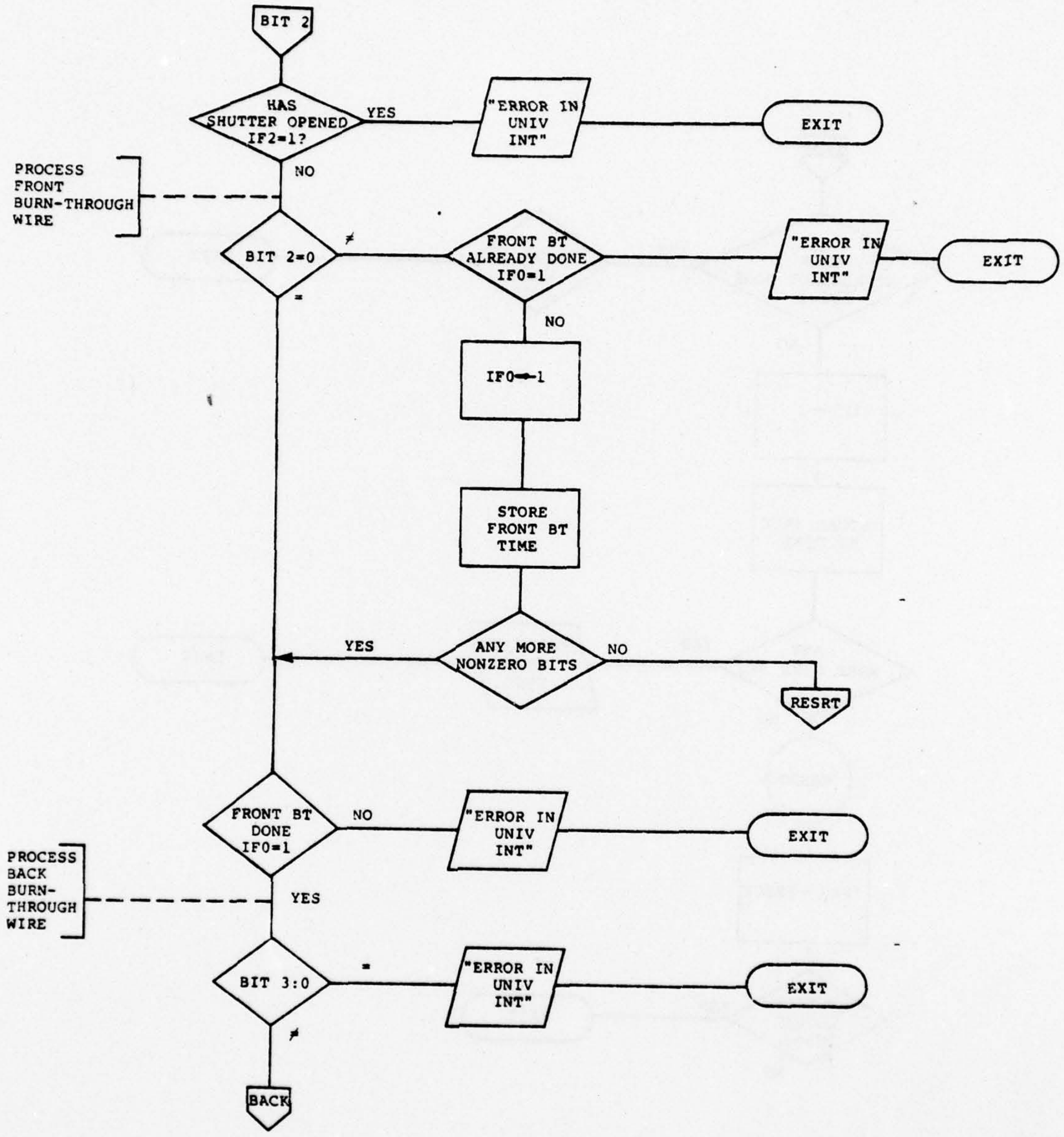


Figure 1. Flowchart of OZUNI. (Continued)

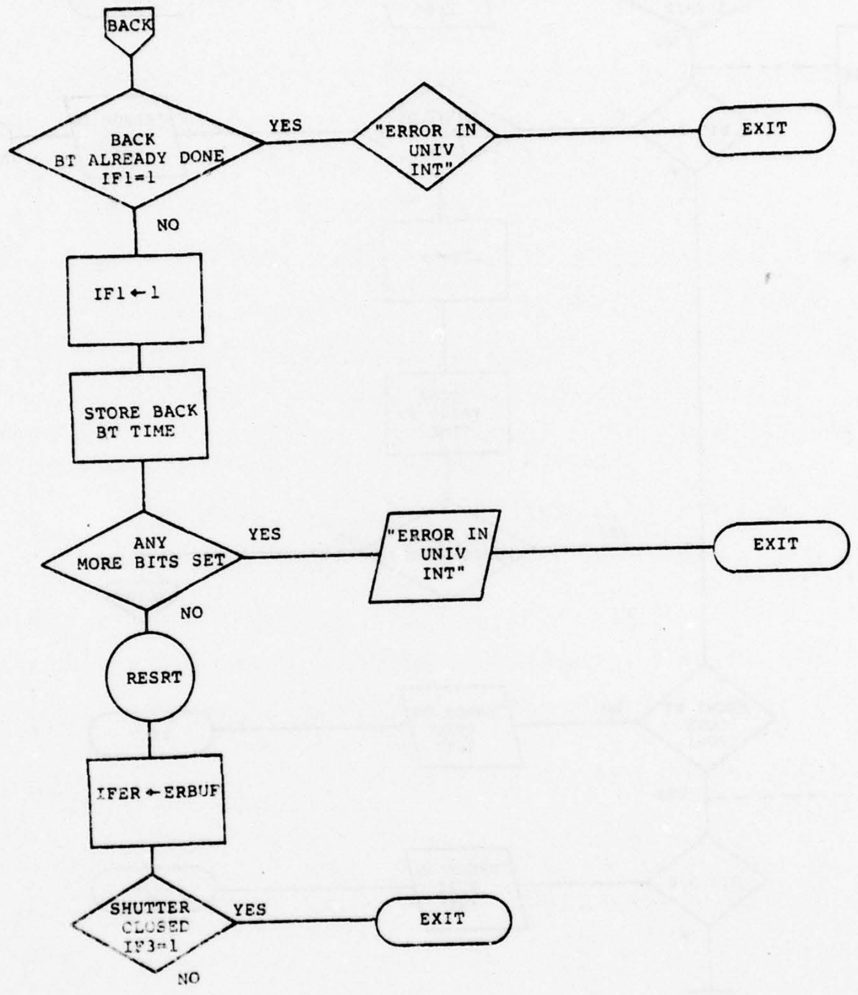


Figure 1. Flowchart of OZUNI. (Continued)

of this system are now open to the user. The normal protocol for an experiment is to use TSS to transfer the COMMON area of RTE-C to a disk file on the SEL 86 and then to run SELOZ.

To transfer the COMMON area, type "RM" in response to the above request:

```
        ENTER FUNCTION CODE,?, OR CR TO TERMINATE: RM
```

The terminal will respond with a question mark and the user should type Y:

```
                  ?Y
```

The terminal will follow this with another

```
        ENTER FUNCTION CODE,?, OR CR TO TERMINATE:
```

The user should respond with:

```
        ENTER FUNCTION CODE,?, OR CR TO TERMINATE: PM.
```

The terminal will type:

```
        - BEGIN PROGRAM MONITOR -
        PROGRAM (, #SECONDS):
```

The proper user's response is:

```
        PROGRAM (, #SECONDS): MØSELOZ,200.
```

This will start the execution of SELOZ.

When SELOZ has completed execution, the normal sequence of interaction is

```
        PROGRAM(,#SECONDS): control-D
        ENTER FUNCTION CODE,?, OR CR TO TERMINATE:  control-D
        TERMINAL TYØØ LOGGED OUT
```

At this point control is returned to RTE-C.

<u>At the end of a day's runs</u>, provide the following command to RTE-C:

```
                        *OF,SEL86,1.
```

A more technical description of SEL86 is found in the Programmer Comments section of Reference 15.

AD-A058 193 DAYTON UNIV OHIO RESEARCH INST F/G 9/2

AFML SCIENTIFIC AND ENGINEERING COMPUTER SUPPORT.(U)

DEC 77 D L FORD, T WOOD, M DENNIS, D G LEET F33615-75-C-5191

UNCLASSIFIED UDRI-TR-77-10 AFML-TR-77-231 NL

2 OF 2

AD
A058193

END
DATE
FILMED
10-78
DDC

2.5 SELOZ

SELOZ and its supporting routines analyze the data obtained by OZ, generate summaries of the analysis produce plots, and store the data on digital magnetic tape.

When SELOZ is initiated through SEL86 it performs the following procedure:

1) Send the message "BEGINNING SELOZ" to the LHMEL teletype.

2) Input the Calibration Deck (MØ$CALDK).

3) Input the experimental data (MØ$OZFIL).

4) Calculate the time the shutter was open (TS):

   TS = shutter close time (ISHC) - shutter open time (ISHO).

5) Calculate the time required to burn through the sample (TB):

   TB = back burn through wire time (IBTB) -
   front burn-through wire time (IBTF).

6) Calculate the time between the shutter open time and the time the front burn-through wire burned through (TD):

   TD = front burn-through wire time (IBTF) -
   shutter open time (ISHO).

7) If TD is greater than 30 seconds or negative, dump results, which will consist of zero values, and exit.

8) Convert the power meter readings to floating point volts:

```
CHØ(I) = FLOAT(IAND(IMETR(I), B177760))*.0003125 .
```

   (Note: This procedure is documented in the HP 91000A A/D Converter Manual.)

9) Convert power meter volts to laser power (PL(I)):

   PL(I) = power volts + (power responsivity/beamsplitter reflectivity)

   or

```
PL(I) = CHØ(I) + (C(2)/C(1))
```

   (See Table III for Calibration Table parameter definitions as stored in the program's arrays.)

10) Convert laser power to laser power on target (PT(I)):

   PT(I) = (1. - beamsplitter reflectivity)*
   (combined reflectivity of 3 mirrors)*
   (laser power)

or

$$PT(I) = (1.-C(L))* C(3) * PL(I)$$

11) Find the time (TL1) at which laser power (PL) first equals or exceeds the laser power threshold (C(4)).
12) Find the first time (TL2) after TL1 at which the laser power (PL) decreases below the laser power threshold (C(4)).
13) Calculate the difference between TL1 and TL2 (TL):

$$TL = TL2-TL1 .$$

14) Define the TL time interval as the time interval from TL1 to TL2 inclusive.
IBINL1 is the element in any of the four data arrays corresponding to TL1 and IBINL2 is the element corresponding to TL2. Then NL = IBINL2-IBINL1+1 is the number of elements in any array over the TL time interval. Calculate the average laser power over the TL time interval (PLLA):

$$PLLA = \frac{1}{NL} \sum_{I=IBINL1}^{IBINL2} PL(I) .$$

15) Calculate the maximum laser power (PLLMX) over the TL time interval:

$$PLLMX = \max\{PL(I) \mid I=IBINL1, IBINL2\} .$$

16) Define the TS time interval as the time interval from shutter open time (ISHO) to shutter close time (ISHC) inclusive. Define IBINS1 as the element in any of the four data arrays corresponding to ISHO and IBINS2 as the element corresponding to ISHC. NS=IBINS2-IBINS1+1 is the number of elements in any data array over the TS time interval. Calculate the average laser power over the TS time interval (PLSA):

$$PLSA = \frac{1}{NS} \sum_{I=IBINS1}^{IBINS2} PL(I) .$$

17) Calculate the maximum laser power over the TS time interval (PLSMX):

$$PLSMX = \max\{PL(I)\,|\,I=IBINS1, IBINS2\} .$$

18) Calculate the minimum laser power over the TS time interval (PLSMN):

$$PLSMN = \min\{PL(I)\,|\,I=IBINS1, IBINS2\} .$$

19) Calculate the integral of laser power over the TL time interval (WJ):

$$WJ = .050* \sum_{I=IBINL1+1}^{IBINL2} PL(I) .$$

The value .050 is in units of seconds and is the bin width.

20) Calculate the average laser power on target over the TS time interval (PTSA):

$$PTSA = \frac{1}{NS} \sum_{I=IBINS1}^{IBINS2} PT(I) .$$

21) Define the TB time interval as the interval from the time the front burn-through wire burned through to the time the back burn through wire burned through. Define IBINB1 as the element in any of the four data arrays corresponding to IBTF and IBINB2 as the element corresponding to IBTB. NB - IBINB2-IBINB1 is the number of elements in any data array over the TB time interval. Calculate the average laser power on target during the TB time interval (PTBA):

$$PTBA = \frac{1}{NB} \sum_{I=IBINB1}^{IBINB2} PT(I) .$$

22) Compare the mirror-to-sample length supplied in the experimental data (FID(1)) with the three sets of distance bounds in the Calibration Table. Select the correct slope (C7) and intercept value (C8) and compute the area of the laser beam

at the target:

DIAM = C6+C7*FID(1)

$$AREA = (\frac{\pi}{4})*DIAM^2 .$$

23) Calculate the average laser power on target per unit area of the target over the TS time interval (PDSA):

PDSA = PTSA/AREA .

24) Calculate the average laser power on target per unit area of the target over the TB time interval:

PDBA = PTBA/AREA .

25) Calculate the integral of laser power on target over the TS time interval (EDSA):

$$EDSA = .050* \sum_{I=IBINS1+1}^{IBINS2} PT(I).$$

26) Calculate the integral of laser power on target over the TB time interval (EDBA):

$$EDBA = .050* \sum_{I+IBINB1+1}^{IBINB2} PT(I) .$$

27) Convert the air temperature readings to floating point volts:

CH1(I) = FLOAT(IAND(ITEMP(I), B177760))*.0003125 .

28) Convert air temperature volts to degrees

air temperature, degrees = (air temperature, volts) + air temperature channel responsivity, degrees/volt)

or

CT(I) = CH1(I)*C(12) .

29) Convert the static pressure readings to floating point volts:

CH2(I) = FLOAT(IAND(IPRES(I), B177760))*.0003125 .

30) Convert static pressure volts to psi:

static pressure, psi = (static pressure, volts)* (static pressure channel responsivity, psi/volt)

or

CT2(I) = CH2(I)*C(13) .

31) Convert the total pressure readings to floating point volts:

CH3(I) = FLOAT(IAND(JPRES(I),B177760))*.0003125 .

32) Convert total pressure volts to psi:

total pressure, psi = (total pressure, volts)* (total pressure channel responsivity, psi/volt)

or

CT3(I) - CH3(I)*C(14) .

33) Compute the minimum air temperature over the TS time interval (CTMN):

CTMN = Min{CT(I) | I=IBINS1, IBINS2} .

34) Compute the RMS value of the static pressure over the TS time interval (PSR):

$$\mathrm{PSR} = \sqrt{\frac{1}{\mathrm{NS}} \sum_{\mathrm{I=IBINS1}}^{\mathrm{IBINS2}} \mathrm{CT2(I)}^2} \quad .$$

35) Calculate the RMS value of total pressure over the TS time interval (PTR):

$$\mathrm{PTR} = \sqrt{\frac{1}{\mathrm{NS}} \sum_{\mathrm{I=IBINS1}}^{\mathrm{IBINS2}} \mathrm{CT3(I)}^2} \quad .$$

36) Calculate the mach number (CMACH):

$$\mathrm{CMACH} = \sqrt{5*[(\frac{\mathrm{PTR}}{\mathrm{PSR}})^{.285714} - 1.]} \quad .$$

37) Print the Calibration Table.

38) Print the evaluation sheets and dumps of various arrays.

39) Send the post-test summary to the LHMEL teletype.

40) Send "SAVE DATA?" message to LHMEL teletype. Wait for an answer. If the answer is not "N", sent the message "NEW TAPE?" to the LHMEL teletype. If the response to "NEW TAPE?" is "N", send the response to "FIRST RUN?" to the LHMEL teletype. If the response to "FIRST RUN?" is "N", then this is not the first run of the day and the run data are added to the tape. If the response to "FIRST RUN?" is anything else, then a search of the magnetic tape is made for the last run on the tape before the run data are added to the tape. If the response to "NEW TAPE?" is anything other than "N" then the tape is assumed to be empty and the run data is written without checking whether or not the tape is at the end of useful information.

41) Send "PLOTS?" message to LHMEL teletype. Wait for an answer. If the answer is not "N", generate the plots.

TABLE III

CALIBRATION TABLE PARAMETERS IN ARRAYS

ARRAY LABEL	CONTENTS		
DATE	calibration table date		
C(1)	beamsplitter reflectivity		
C(2)	power responsivity (watts/v)		
C(3)	combined reflectivity of 3 mirrors		
C(4)	laser power threshold (watts		
C(5)	mirror focal length (m)		
D1(1)	lower mirror distance bound	(cm)	diameter parameters used for sample palced much less than focal lengths
D2(1)	upper mirror distance bound	(cm)	diameter parameters used for sample palced much less than focal lengths
C(6)	slope		
C(7)	intercept		
D1(2)	lower mirror distance bound	(cm)	diameter parameters used for sample placed near focal lengths
D2(2)	upper mirror distance bound	(cm)	diameter parameters used for sample placed near focal lengths
C(8)	slope		
C(9)	intercept		
D1(3)	lower mirror distance bound	(cm)	diameter parameters used for sample placed much greater than focal lengths
D2(3)	upper mirror distance bound	(cm)	diameter parameters used for sample placed much greater than focal lengths
C(10)	slope		
C(11)	intercept		
C(12)	air temperature channel responsivity (degree/volt)		
C(13)	static pressure channel responsivity (psi/volt)		
C(14)	total pressure channel responsivity (psi/volt)		

SECTION 3
STANDARD OPERATING PROCEDURES

3.1 BASIC PROCEDURE

1. Before each run in which the tape drive or plotter is to be used, or before each series of runs if the runs are going to follow one after another, call the SEL operator (Cindy) at 53778 and inform her of the equipment requirements. Also inform her whether or not the tape is to be rewound at the end of the run.
2. Start RTE-C using Section 3.2.
3. Load the OZUNI program using the RTE-C "LO" command.
4. If necessary, modify the Calibration Table using the procedure in Section 3.3.
5. Bring up OZ to set the pre-run parameters values.
6. Bring up OZGO for the experimental run.
7. Bring up OZPST to set the post-run parameter values.
8. If desired, bring up DADMP to examine the data.
9. Bring up SEL86 and transfer the data to the OZFIL file in the SEL.
10. Run SELOZ.
11. Bring up OZ to change any pre-run parameters that are to be changed for the next run.
12. Go to step 6 for another run.

3.2 STARTING RTE-C

1. Press Master Power button on upper right side of 9640A rack.
2. Insert Key into front panel of 21MX. Turn key counter-clockwise to R and then back to operate.
3. Turn on paper tape reader.
4. Turn on teletype (to line position).
5. Load RTE-C paper tape into paper tape reader.
6. Enter 001500_8 into switch (5) register.
7. Press IBL.

8. Press RESET.

9. Press RUN.

10. When the paper tape has been read, the light pattern on the front panel should be 100277 and this should be the contents of the S register.

11. Enter 000002_8 into P register.

12. Press Preset and RUN.

3.3 MODIFYING THE CALIBRATION TABLE

The Calibration Deck used by the SELOZ program can be changed from the LHMEL teletype using the SEL86 TEST EDITOR. Any time after RTE-C has been started, an

```
ON,SEL86
```

command will initialize communication with the SEL. The following sequence will provide access to the MØ$CALDK Edit file, which can then be changed using the EDITOR commands:

```
-SEL TERMINAL SUPPORT SYSTEM, TERMINAL TYØØ
ENTER FUNCTION CODE,?, OR CR TO TERMINATE: ED
-BEGIN TEST EDITOR.
ENTER 2-CHARACTER USER CODE OR CR TO TERMINATE: MØ
COMMAND? USE CALDK
COMMAND?
```

When the changes have been made, the following response saves the new file:

```
COMMAND? SAVE CALDK SCR
```

A response of EXIT terminates the EDITOR.

The following command terminates SEL86:

```
OF,SEL86,1.
```

APPENDIX E
FORTRAN PROGRAM SER TO CALCULATE RESPONSE FUNCTIONS

```
      PROGRAM SER(INPUT,OUTPUT,TAPE11=/910,TAPE10)
      DIMENSION TH(35), SA(35), DIDI(35), QCONV(35), QRP(35), HW(35), CM
     1T(35), GMST(35), PGPU(35), DECOM(35), TB(35), DEDT(35), QCHEMT(35)
     2, QCONDI(35), QLOSST(35), BPRM(35), BPRMG(35), CMD(35), GSMS(35),
     3BR(35), NDR(35), CPGAS(35), CH(35), MATL(12), TA(12), H(12), THK(3
     45), P1(35), P2(35), P3(35), P4(35), P5(35), P6(35), RECORD(36), CN
     51(12), RO1(12), RAD(35), DEL(12), CP1(12), DMDG(12), RR(12), PD3(3
     65), PN4(35), PN5(35), PN6(35), X(12), BFT(35), TS(35), WFT(35)
C     ******************************************************************
C     *THIS PROGRAM CALCULATES THE MATERIAL RESPONSE FUNCTION AS       *
C     *DEFINED IN AFML-TR-77-XX,USING AS INPUT, THE CALCULATIONS       *
C     *FROM A MODIFIED VERSION OF CMA FROM THE UNIFIED SERIES OF CODES.*
C     *                                                                *
C     *AUTHOR DALE FORD 11-20-76 (513)255-3778 UDRI                    *
C     *                                                                *
C     *VARIABLES NAMES AS FOUND IN DIMENSIONS ARE THOSE USED IN CMA    *
C     *THE RESPONSE FUNCTIONS ARE DEFINE IN AFML-TR-77-XX              *
C     *BACK FACE TEMP IS RZERO  RONE TO RSIX ARE P1 TO P6              *
C     *INPUT-TAPE11 OBTAINED FROM MODIFIED CMA                         *
C     *OUTPUT-1) LISTING OF TIME AND P1 TO P6                          *
C     *        2) TAPE 10 OUTPUT TO GO TO PUNCH TO BE USED IN GENERAL  *
C     *           PLOTTING PROGRAM                                     *
C     ******************************************************************
C***  QCONV,QRP,RAD,ARE REALLY QCONVT,QRPT,RADT AS FOUND IN CMA
C***  READ HEADER CARDS
      READ (11,230) (RECORD(I),I=1,36)
      PRINT 220
      PRINT 230, (RECORD(I),I=1,36)
      WRITE (10,230) (RECORD(I),I=1,36)
C***  READ SURFACE DATA
      DO 170 I=1,35
      READ (11,240) TH(I),SA(I),DIDI(I),QCONV(I),QRP(I),RAD(I),HW(I),CMT
     1(I),GMST(I),PGPU(I),DECOM(I),TB(I),DEDT(I),QCHEMT(I),QCONDT(I),QLO
     2SST(I),BPRM(I),BPRMG(I),CMD(I),GSMS(I),BR(I)
      IF (EOF(11)) 180,110
110   CONTINUE
      READ (11,250) NUMN,NDR(I),CPGAS(I),BFT(I),CH(I)
C***  CALCULATE NUMBER OF NODES
      NOLF=NUMN-NDR(I)-2
      NOLF1=NOLF
      IF (NOLF.GT.6) NOLF1=6
C***  READ MATERIAL NODAL PROPERTIES
      READ (11,250) (KK,MATL(J),TA(J),RO1(J),CN1(J),CP1(J),H(J),DMDG(J),
     1RR(J),DEL(J),PN4(J),PN5(J),PN6(J),J=1,NOLF1)
C***  PN5 IS NOT USED
      IF (NOLF.LE.6) GO TO 120
      READ (11,250) (KK,MATL(J),TA(J),RO1(J),CN1(J),CP1(J),H(J),DMDG(J),
     1RR(J),DEL(J),PN4(J),PN5(J),PN6(J),I=7,NOLF)
120   CONTINUE
      TS(I)=TA(1)
      IF (I.EQ.1) GO TO 160
      SUM=0.
      SU1=0.
      SU2=0.
      SU3=0.
      SU4=0.
      SU5=0.
      WI=0.
C***  CALCULATE THICKNESS AND NODAL POSITION
      DO 130 J=1,NOLF
      SUM=DEL(J)+SUM
      IF (J.GT.1) SU1=SU1+0.5*DEL(J-1)
      X(J)=0.5*DEL(J)+SU1
      [illegible]
```

```
130   CONTINUE
      IM1=I-1
      IM2=I-2
      THK(IM1)=SUM
C*** CALCULATE P1
      DO 140 J=1,NDLF
140   SU5=CN1(J)/RO1(J)/CP1(J)/SUM**2*DEL(J)+SU5
      P1(IM1)=SU5*TH(I)/SUM
      PD3(IM1)=QCONV(I)+QRP(I)-RAD(I)
C*** CALCULATE P3
      P3(IM1)=QLOSS1(I)/PD3(IM1)
C*** CALCULATE THE KERNALS OF P4,P5,P6
C*** INTEGRATING OVER X
      DO 150 J=1,NDLF
      SU2=PN4(J)*DEL(J)+SU2
      SU3=SU3-H(J)*SUM*DMDG(J)/RR(J)*DEL(J)
C*** -363. IS THE HEAT OF FORMATION OF THE PLASTIC (BASELINE)
      SU4=SU4+DMDG(J)/RR(J)*(-363.)*DEL(J)
      WT=WT+DEL(J)*RO1(J)*.001/12.
150   CONTINUE
      SU2=SU2/SUM
      P4(IM1)=SU2/PD3(IM1)
      SU3=SU3/SUM
      P5(IM1)=SU3/PD3(IM1)
      SU4=SU4/SUM
      P6(IM1)=SU4/PD3(IM1)
      WET(IM1)=WT
160   CONTINUE
170   CONTINUE
180   CONTINUE
      I=I-1
      IM1=I-1
      S2=S3=S4=S5=S6=0.
C*** CALCULATE THE INTEGRALS OF P4,P5,P6
C*** FOR EACH TIME INTERVAL
      PRINT*,"     TIME        P1              P2            P3          P4
     *       P5           P6            WEIGHT "
      DO 190 J=1,IM1
      J1=J+1
      DTH=(TH(J1)-TH(J))*0.5
      P2(J)=DTH*P1(J)+DTH*S2
      S2=P1(J)
      PN4(J)=DTH*(P4(J)+S4)
      S4=P4(J)
      PN5(J)=DTH*(P5(J)+S5)
      S5=P5(J)
      PN6(J)=DTH*(P6(J)+S6)
      S6=P6(J)
190   CONTINUE
C*** SUM OVER ALL INTERVALS AND OUTPUT DATA
      S2=S3=S4=S5=S6=0.
      DO 200 J=1,IM1
      J1=J+1
      P2(J)=P2(J)+S2
      S2=P2(J)
      PN4(J)=PN4(J)+S4
      S4=PN4(J)
      PN5(J)=PN5(J)+S5
      S5=PN5(J)
      PN6(J)=PN6(J)+S6
      S6=PN6(J)
      PRINT 260, TH(J1),P1(J),P2(J),P3(J),PN4(J),PN5(J),PN6(J),WET(J)
C*** WRITE OUTPUT FOR PUNCH
      WRITE (10,270) TH(J1),TS(J1),DFT(J1),THK(J),P1(J),WET(J)
200   CONTINUE
      PRINT 2[illegible]
```

```
PRINT*," TIME   FRONT-TEMP    REAR-TEMP    THICKNESS "
DO 210 J=1,IM1
J1=J+1
PRINT 260, TH(J1),TS(J1),BFT(J1),THK(J)
WRITE (10,270) TH(J1) P2(J),P3(J),PN4(J),PN5(J),PN6(J)
STOP

FORMAT (1H1)
FORMAT (6X1,A6)
FORMAT (21E13.5)
FORMAT (2I5,11E13.5)
FORMAT (1X,F8.2,7E13.5)
FORMAT (F13.4,5E14.7)
FORMAT (1H1)
END

********           DALFOOG     //// END  OF LIST ////
```

APPENDIX F

FORTRAN PROGRAMS FOR THE ANALYSIS OF MICROSTRUCTURES

```
PROGRAM PARSEPI    74/74    OPT=1                          FTN 4.5+414        06/01/77  09.34.51        PAGE      1

           PROGRAM PARSEPI(INPUT,OUTPUT,TAPE3,TAPE4,TAPE5)
      C-
      C-PROGRAM PARSEPI DETERMINES THE PARTICLES IN THE THRESHOLDED IMAGE AFTER THE
      C-INTERIOR PIXEL VALUES HAVE BEEN GENERATED BY PROGRAM PARSER.  IN ITS PRESENT
      C-STATE IT DETERMINES THE PARTICLE CORES,UPPER EXTREMES,AND LOWER EXTREMES;
      C-CONVERTS THEM TO INTERVAL NOTATION,AND OUTPUTS THE PARTICLES (IN INTERVAL
      C-NOTATION) ON TAPE5.
      C-
           DIMENSION B(3),C(32),D(1500)
           DIMENSION STACK(1000,3)
           INTEGER B,C,D
           INTEGER STACK,TOP
           TOP=1
           READ 1000,NROWS,NCOLS
      1000 FORMAT(2I5)
           IFLAG=0
           IY=3
           IY1=4
           NWR=NCOLS+2
      C-
      C-INITIALIZE I POINTER ARRAY (B) AND SHIFT ARRAY (C)
      C-
           B(1)=1
           B(2)=NWR+1
           B(3)=B(2)+NWR
           C(32)=58
           C(1)=0
           KE=60
           DO 10 I=2,31
           KE=KE-2
      10   C(I)=KE
           K=1
           PRINT 5000,C
           PRINT 5000,B
      5000 FORMAT(1H1,31I4)
           K=3
           CALL FIRST(D,B,C,NWR,K,IY,I,J,II,IROW,IY1)
           CALL PARFIND(D,B,C,I,J,II,IROW,STACK,TOP,K,NROWS,IY,IY1,NWR)
           CALL STKSRT(STACK,TOP)
           L=1
           CALL INTNOTN(STACK,TOP,L,LL)
           CALL SURMO2S(D,B,C,STACK,TOP,L,NROWS,NWR,IY,IY1,IROW)
           CALL TEST(D,B,C,STACK,TOP,L,NROWS,NWR,IY,IY1,IROW)
           DO 50 I=1,25
      50   PRINT *,(STACK(I,J),J=1,3)
           STOP
      100  FORMAT(1H ,*FIRST INFO   *,3(I5,5X))
           END
      SUBROUTINE FIRST(D,B,C,NWP,K,IY,I,J,II,IROW,IY1)
      DIMENSION D(1),B(1),C(1)
      INTEGER D,B,C
C-
C-THIS SUBROUTINE RETURNS THE LOCATION (I,J,II) OF THE FIRST K-VALUED ELEMENT
C-ENCOUNTERED IN THE IMAGE.  I-ROW ADDRESS(RECORD),J-WORD ADDRESS,II-ELEMENT
C-ADDRESS.
      INTV(I1,J,II)=AND(SHIFT(D(B(I1)+J),-C(II)),3)
      M=NWP-1
      I=1
10    BUFFER IN(IY,1) (D(1),D(NWP))
      IF(UNIT(IY).EQ.0) GO TO 40
      DO 30 J=1,M
      DO 20 II=2,31
      KK=INTV(1,J,II)
      IF(KK.EQ.K) GO TO 50
20    CONTINUE
30    CONTINUE
      I=I+1
      GO TO 10
40    I=J=II=0
50    REWIND IY
      M=I-2
      DO 60 I=1,M
      BUFFER IN(IY,1) (D(1),D(NWP))
      IF(UNIT(IY).EQ.0) GO TO 90
      BUFFER OUT(IY1,1) (D(1),D(NWP))
60    L=UNIT(IY1)
      BUFFER IN(IY,1) (D(1),D(NWP))
      IF(UNIT(IY).EQ.0) GO TO 90
      BUFFER IN(IY,1) (D(B(2)),D(B(3)-1))
      IF(UNIT(IY).EQ.0) GO TO 90
      BUFFER IN(IY,1) (D(B(3)),D(B(3)+NWP-1))
      IF(UNIT(IY).EQ.0) GO TO 90
      IROW=M+1
      I=2
      RETURN
90    CONTINUE
      PRINT 110
      RETURN
110   FORMAT(1H ,*ERROR IN FIRST---EOF ENCOUNTERED*)
      END
```

```
SUBROUTINE PARFIND    74/74    OPT=1                    FTN 4.5+414        06/01/77  09.34.51        PAGE    1

              SUBROUTINE PARFIND(D,B,C,I,J,II,IROW,STACK,TOP,K,NROWS,IY,IY1,NWR)
              DIMENSION D(1),B(1),C(1),STACK(1000,3)
              INTEGER D,B,C,STACK,TOP
              INTV(I,J,II)=AND(SHIFT(D(B(I)+J),-C(II)),3)
              TOK=1
              IFLAG=0
C-
C-TEST UPPER LEFT ELEMENT
C-
   30         J1=J
              IF(C(II).EQ.58) J1=J-1
              KK=INTV(I-1,J1,II-1)
              IF(KK.NE.K) GO TO 40
              I=I-1
              J=J1
              II=II-1
              GO TO 95
C-
C-TEST UPPER ELEMENT
C-
   40         CONTINUE
              KK=INTV(I-1,J,II)
              IF(KK.NE.K) GO TO 50
              I=I-1
              GO TO 95
C-
C-TEST UPPER RIGHT ELEMENT
C-
   50         CONTINUE
              J1=J
              IF(C(II).EQ.0) J1=J+1
              KK=INTV(I-1,J1,II+1)
              IF(KK.NE.K) GO TO 70
              I=I-1
              J=J1
              II=II+1
              GO TO 95
C-
C-TEST LOWER LEFT ELEMENT
C-
   70         CONTINUE
              J1=J
              IF(C(II).EQ.58) J1=J-1
              KK=INTV(I+1,J1,II-1)
              IF(KK.NE.K) GO TO 80
              IF(IFLAG.EQ.0) CALL STORE(D,B,C,STACK,I,J,II,IROW,TOP)
              IFLAG=0
              I=I+1
              J=J1
              II=II-1
              GO TO 95
C-
C-TEST LOWER ELEMENT
C-
   80         CONTINUE
              KK=INTV(I+1,J,II)
              L2=B(I)
```

```
      L3=B(3)+NWR-1
      IF(KK.NE.K) GO TO 90
      IF(IFLAG.EQ.0) CALL STORE(D,B,C,STACK,I,J,II,IROW,TOP)
      IFLAG=0
      I=I+1
      GO TO 95
C-
C-TEST LOWER RIGHT ELEMENT
C-
90    CONTINUE
      J1=J
      IF(C(II).EQ.0)J1=J+1
      KK=INTV(I+1,J1,II+1)
      IF(KK.NE.K) GO TO 95
      IF(IFLAG.EQ.0) CALL STORE(D,B,C,STACK,I,J,II,IROW,TOP)
      IFLAG=0
      I=I+1
      J=J1
      II=II+1
      GO TO 95
C-
C-TEST RIGHT ELEMENT
C-
95    CONTINUE
      J1=J
      IF(C(II).EQ.0) J1=J+1
      KK=INTV(I,J1,II+1)
      IF(KK.NE.K) GO TO 60
      IF(IFLAG.EQ.0) CALL STORE(D,B,C,STACK,I,J,II,IROW,TOP)
      IFLAG=0
      J=J1
      II=II+1
      GO TO 95
C-
C-TEST LEFT ELEMENT
C-
60    CONTINUE
      J1=J
      IF(C(II).EQ.58)J1=J-1
      KK=INTV(I,J1,II-1)
      IF(KK.NE.K) GO TO 96
      IF(IFLAG.EQ.0) CALL STORE(D,B,C,STACK,I,J,II,IROW,TOP)
      J=J1
      II=II-1
      GO TO 60
C-
C-EXECUTE BACKSTACK SEARCH
C-
96    CONTINUE
      CALL BKSTACK(STACK,TOP,I,J,II,IFLAG,IBK,D,B,C,IROW),RETURNS(200)
C-
C-UPDATE FILES.GET PROPER RECORDS IN CORE
C-
100     CONTINUE
      CALL UPDATE(I,J,II,D,IY,IY1,NWF,NROWS,IROW,B)
1000  FORMAT(1H ,*HERE I IS BABY*)
      GO TO 30
```

```
        200   CONTINUE
              DO 210 IL=1,TOP
        210   PRINT 300,(STACK(IL,I),I=1,3)
              RETURN
        300   FORMAT(1H ,3(I5,10X))
              END
```

```
      SUBROUTINE BKSTACK(STACK,TOP,I,J,II,IFLAG,IBK,D,B,C,IROW),RETURNS
     1(A1)
      DIMENSION D(1),B(1),C(1),STACK(1000,3)
      INTEGER D,B,C,STACK,TOP
C-
C-THIS SUBROUTINE RETRIEVES THE LOCATION OF AN ELEMENT TO BE USED AS INITIAL
C-POINT OF SEARCHING
C-
      IF(IFLAG.EQ.1) GO TO 10
      IBK=TOP
      CALL STORE(D,B,C,STACK,I,J,II,IROW,TOP)
      IFLAG=1
      GO TO 20
   10 IBK=IBK-1
      IF(IBK.LE.0)RETURN A1
   20 I=STACK(IBK,1)-IROW+1
      J=STACK(IBK,2)
      II=STACK(IBK,3)
      RETURN
      END
```

```
SUBROUTINE STORE      74/74    OPT=1                           FTN 4.5+414        06/01/77  09.34.51        PAGE     1

              SUBROUTINE STORE(D,B,C,STACK,I,J,II,IROW,TOP)
              DIMENSION D(1),B(1),C(1),STACK(1000,3)
              INTEGER D,B,C,STACK,TOP
        C-
        C-THIS SUBROUTINE STORES THE ABS. ADDRESS OF AN ELEMENT ON THE STACK,AND ZEROS
        C-THAT ELEMENT OUT IN THE IMAGE
        C-
        C-
        C-STORE ADDRESS
        C-
              STACK(TOP,1)=IROW+I-1
              STACK(TOP,2)=J
              STACK(TOP,3)=II
              DO 10 L=1,TOP
        10    CONTINUE
              TOP=TOP+1
        100   FORMAT(1H1,*SUBROUTINE STORE,TOP=*,I4)
        110   FORMAT(1H ,3(I4,5X))
        C-
        C-ZERO OUT ELEMENT IN IMAGE
        C-
              D(B(I)+J)=AND(D(B(I)+J),SHIFT(COMPL(3),C(II)))
              RETURN
              END
```

```
      SUBROUTINE UPDATE(I,J,II,D,IY,IY1,NWP,NROWS,IROW,B)
      DIMENSION D(1),B(1)
      INTEGER D,B
C-
C-THIS SUBROUTINE MAINTAINS PROPER INFORMATION IN CORE,KEEPS POINTERS AT
C-CORRECT RELATIVE AND ABSOLUTE VALUES,AND MAINTAINS PROPER FILE STRUCTURE
C-
1030   FORMAT(1H ,*UPDATE----IT IS HERE*)
      IF(I.EQ.2) GO TO 70
C-
C-CLEAR CORE
C-
      BUFFER OUT(IY1,1) (D(1),D(NWP))
      L=UNIT(IY1)
      BUFFER OUT(IY1,1)(D(B(2)),D(B(3)-1))
      L=UNIT(IY1)
      BUFFER OUT(IY1,1)(D(B(3)),D(B(3)+NWP-1))
      L=UNIT(IY1)
      M=IROW+3
      DO 10 IL=M,NROWS
      BUFFER IN(IY,1)(D(1),D(NWP))
      IF(UNIT(IY).EQ.0) GO TO 15
      BUFFER OUT(IY1,1)(D(1),D(NWP))
10    L=UNIT(IY1)
15    CONTINUE
      ENDFILE IY1
      REWIND IY
      REWIND IY1
C-
C-CHANGE FILES
C-
      L=IY
      IY=IY1
      IY1=L
C-
C-DETERMINE PROPER RECORDS TO HAVE IN CORE:SKIP FORWARD
C-
      IROW=IROW+I-2
      M=IROW-1
1010   FORMAT(1H ,*UPDATE DATA      *,6(I5,5X))
      DO 20 IL=1,M
      BUFFER IN(IY,1)(D(1),D(NWP))
      IF(UNIT(IY).EQ.0) GO TO 30
      BUFFER OUT(IY1,1)(D(1),D(NWP))
20    L=UNIT(IY1)
      L=1
      BUFFER IN(IY,1)(D(1),D(NWP))
      IF(UNIT(IY).EQ.0) GO TO 80
      L=2
      BUFFER IN(IY,1)(D(B(2)),D(B(3)-1))
      IF(UNIT(IY).EQ.0) GO TO 80
      L=3
      BUFFER IN(IY,1)(D(B(3)),D(B(3)+NWP-1))
      IF(UNIT(IY).EQ.0) GO TO 80
      I=2
70    CONTINUE
      RETURN
```

```
         75     CONTINUE
                BACKSPACE IY1
                DO 76 NNR=1,4
         76     BACKSPACE IY
                CALL MVW1(D(R(2)).1.D(R(3)).1.NWR)
                CALL MVW1(D(1).1.D(R(2)).1.NWR)
                BUFFER IN(IY.1)(D(1).D(NWR))
                IF(UNIT(IY).EQ.0) GO TO 80
                I=2
                IROW=IROW-1
                RETURN
         80     PRINT 100
                STOP
         90     PRINT 110
                STOP
        100     FORMAT(1H ,*ERROR IN UPDATE WHILE INPUT TO CORE*)
        110     FORMAT(1H ,*ERROR IN UPDATE WHILE SKIPPING*)
                END
```

```
SUBROUTINE MVW1       74/74    OPT=1                              FTN 4.5+414        06/01/77  03.34.51        PAGE     1

                SUBROUTINE MVW1(A,IFROM,B,ITO,K)
          C-
          C-THIS SUBROUTINE TRANSFERS K WORDS FROM CORE,BEGINNING WITH LOCATION A(IFROM).
          C-TO CORE,BEGINNING WITH LOCATION B(ITO).
          C-
                DIMENSION A(1),B(1)
                INTEGER A,B
                I=IFROM
                J=ITO
                DO 10 L=1,K
                B(J)=A(I)
                I=I+1
          10    J=J+1
                RETURN
                END
```

```
SURROUTINE STKSRT     74/74   OPT=1                    FTN 4.5+414       06/01/77  09.34.51       PAGE     1

                 SUBROUTINE STKSRT(STACK,TOP)
                 DIMENSION STACK(1000,3)
                 INTEGER STACK,TOP
           C-
           C-THIS SUBROUTINE USES SUBROUTINES SRTP AND PRESRT TO SORT PIXEL STACK
           C-
                 TOP=TOP-1
                 M=1
                 N=1
                 IR=1
                 IP=TOP
           10    CALL SRTP(STACK,IR,IP,M)
                 IF(IP.GE.TOP.AND.M.GT.1)GO TO 30
                 IF(M.NE.1)IR=IP+1
           20    CALL PRESRT(STACK,TOP,IR,IP,N)
                 M=N+1
                 GO TO 10
           30    CONTINUE
                 IP=1
                 N=N+1
                 IF(N.LE.2) GO TO 20
                 RETURN
                 END
```

```
SUBROUTINE PRESRT      74/74    OPT=1                                   FTN 4.5+414         06/01/77  09.34.51        PAGE      1

          SUBROUTINE PRESRT(STACK,TOP,IR,IP,N)
          DIMENSION STACK(1000,7)
          INTEGER STACK,TOP
C-
C-SUBROUTINE PRESRT FINDS THE EQUAL-VALUED ELEMENTS OF THE N TH COLUMN
C-OF STACK,AND DETERMINES THE LIMITS FOR SRTP(IR,IP)
C-
          L=STACK(IR,N)
          IP=IP+1
   10     IF(STACK(IP,N).GT.L) GO TO 20
          IF(N.GT.1.AND.STACK(IP,1).GT.STACK(IP,1))GO TO 20
          IP=IP+1
          IF(IP.GT.TOP)GO TO 20
          GO TO 10
   20     CONTINUE
          IP=IP-1
          RETURN
          END
```

```
      SUBROUTINE SRTR(STACK,IR,IP,N)
      DIMENSION STACK(1000,3)
      INTEGER STACK
C-THIS SUBROUTINE SORTS THE STACK OF PIXELS:SORTING ON THE N TH COLUMN
C-OF STACK,BEGINNING WITH THE IR TH ELEMENT AND ENDING  ON THE IPTH ELEM
C-
      IF(IR.EQ.IP) RETURN
      M=IP-1
      DO 20 I=IR,M
      L=I+1
      DO 10 J=L,IP
      IF(STACK(I,N).LE.STACK(J,N)) GO TO 10
      DO 5 I1=1,3
      MM=STACK(I,I1)
      STACK(I,I1)=STACK(J,I1)
5     STACK(J,I1)=MM
10    CONTINUE
20    CONTINUE
      RETURN
      END
      SUBROUTINE INTNOTN(STACK.TOP.L.LL)
      DIMENSION STACK(1000.3)
      INTEGER STACK.TOP.R.EC.BC
C-
C-THIS SUBROUTINE PUTS THE ELEMENTS OF THE STACK INTO INTERVAL NOTATION
C-
      LL=1
      IP=1
10    CALL PRESRT(STACK.TOP.IR.IP.1)
      R=STACK(IR.1)
      EC=BC=STACK(IR.2)*30+STACK(IR.3)
      I=IP+1
20    IF(STACK(I.2).GT.STACK(IR.2)) GO TO 30
      IF(STACK(I.3)-STACK(I-1.3).NE.1) GO TO 40
      M=STACK(I.3)-STACK(I-1.3)
25    EC=EC+1
      I=I+1
      IF(I.GT.TOP) GO TO 40
      IF(I.GT.IP) GO TO 40
      GO TO 20
30    IF(STACK(I.3).NE.2.OR.STACK(I-1.3).NE.31) GO TO 40
      GO TO 25
40    CONTINUE
      IF(L.GT.I) PRINT*." UNDERFLOW IN INTNOTN  ".L.I
      STACK(L.1)=R
      STACK(L.2)=BC
      STACK(L.3)=EC
      IF(I.GT.TOP) GO TO 60
      IF(I.GT.IP) GO TO 50
      EC=BC=STACK(I.2)*30+STACK(I.3)
      L=L+1
      LL=LL+1
      IR=I
      I=I+1
      GO TO 20
50    CONTINUE
      IP=IP+1
      L=L+1
      LL=LL+1
      GO TO 10
60    CONTINUE
      RETURN
      END
      SUBROUTINE SURM02S(D,B,C,STACK,TOP,L,NROWS,NWR,IY,IY1,IROW)
      DIMENSION D(1),B(1),C(1),STACK(1000,3)
      INTEGER D,B,C,STACK,TOP,POINT2
      INTEGER POINT1
      INTV(I,J,II)=AND(SHIFT(D(B(I)+J),-C(II)),3)
C-    ASSUMED LAST INTERVAL OF PARTICLE TO BE IN 2ND RECORD LOCATION IN CORE(D)
C-
C-GENERATE WORKING STACK:POINT1 IS POINTER TO TOP OF WORKING STACK:POINT2 IS
C-POINTER TO TOP OF STORAGE STACK .
C-
      PRINT *,"SURM02S"
      PRINT *,"1 L= ",L
      TOP=L
      I=STACK(L,1)-IROW+1
      CALL UPDATE(I,J,II,D,IY,IY1,NWR,NROWS,IROW,B)
      POINT1=L
      M=STACK(L,2)
      N=STACK(L,3)
      DO 10 I1=M,N
      POINT1=POINT1 + 1
      STACK(POINT1,1)=IROW+1
      STACK(POINT1,2)=I1/30
      STACK(POINT1,3)=MOD(I1,30)+1
10    CONTINUE
      POINT2=POINT1+1
C-
C-EXECUTE LOWER RECORD SEARCH:PLACING ELEMENTS ON STORAGE STACK
C-
      I=3
      M=L+1
      L=POINT1
15    DO 30 I1=M,POINT1
C-
C-SET POINTERS TO WORD AND SHIFT ARRAYS
C-
      J=STACK(I1,2)
      II=STACK(I1,3)
C-
C-TEST LOWER LEFT ELEMENT
C-
      J1=J
      IF(C(II).EQ.58) J1=J-1
      KK=INTV(I,J1,II-1)
      IF(KK.NE.2)GO TO 30
      J=J1
      II=II-1
      CALL STORE(D,B,C,STACK,I,J,II,IROW,POINT2)
      GO TO 50
30    CONTINUE
C-
C-TEST LOWER ELEMENT
C-    KK=INTV(I,J,II)
      IF(KK.NE.2) GO TO 40
      CALL STORE(D,B,C,STACK,I,J,II,IROW,POINT2)
      GO TO 50
40    CONTINUE
C-
```

```
      C-TEST LOWER RIGHT ELEMENT
      C-
            J1=J
            IF(C(II).EQ.0)J1=J+1
            KK=INTV(I,J1,II+1)
            IF(KK.NE.2)GO TO 80
            J=J1
            II=II+1
            CALL STORE(D,B,C,STACK,I,J,II,IROW,POINT2)
      50    CONTINUE
      C-
      C-SEARCH LEFT
      C-
            J1=J
            IF(C(II).EQ.58)J1=J-1
            KK=INTV(I,J1,II-1)
            IF(KK.NE.2)GO TO 60
      C-ABOVE MIGHT HAVE TO BE CORRECTED IN I,J,II
            J=J1
            II=II-1
            CALL STORE(D,B,C,STACK,I,J,II,IROW,POINT2)
            GO TO 50
      60    CONTINUE
      C-
      C-SEARCH RIGHT
      C-
            J1=J
            IF(C(II).EQ.0) J1=J1+1
            KK=INTV(I,J1,II+1)
            IF(KK.NE.2)GO TO 80
            J=J1
            II=II+1
            CALL STORE(D,B,C,STACK,I,J,II,IROW,POINT2)
            GO TO 60
      80    CONTINUE
      C-
      C-TEST FOR EMPTY STORAGE STACK
      C-
            IF(POINT2.EQ.POINT1+1) GO TO 99
      C-SHIFT WORKING STACK:REPEAT SEARCH
            M=POINT1+1
            POINT1=POINT2-1
      C-
      C-SHIFT RECORDS IN CORE:INPUT NEXT RECORD
      C-
            BUFFER OUT(IY1,1)(D(1),D(NWR))
            NC=UNIT(IY1)
            CALL MVW1(D(B(2)),1,D(1),1,NWR)
            CALL MVW1(D(B(3)),1,D(B(2)),1,NWR)
            BUFFER IN(IY,1) (D(B(3)),D(B(3)+NWR-1))
            IF(UNIT(IY))85,97,95
      95    IROW=IROW+1
            GO TO 15
      97    CONTINUE
            PRINT *,"ERROR IN SUBMO2S"
            STOP
      99    CONTINUE
```

```
SUBROUTINE SURMGPS    74/74    OPT=1                         FTN 4.5+414         06/01/77  09.34.51        PAGE     7

              MM=POINT1-L+1
              PRINT *,"2 L= ",L
              CALL STKSRT(STACK(L+1,1),MM,M)
              M=TOP-L-1
              CALL INTNGIN(STACK(L+1,1),MM,M,LL)
              L=TOP+LL
              TOP=L
              RETURN
              END
```

```
      SUBROUTINE TEST(D,B,C,STACK,TOP,L,NROWS,NWR,IY,IY1,IROW)
      DIMENSION D(1),B(1),C(1),STACK(1000,3)
      INTEGER D,B,C,STACK,TOP,POINT2
      INTEGER POINT1
      INTV(I,J,II)=AND(SHIFT(D(B(II)+J),-C(II)),3)
C-    ASSUMED LAST INTERVAL OF PARTICLE TO BE IN 2ND RECORD LOCATION IN CORE(D)
C-
C-GENERATE WORKING STACK:POINT1 IS POINTER TO TOP OF WORKING STACK:POINT2 IS
C-POINTER TO TOP OF STORAGE STACK
C-
      TOP=L
      I=STACK(1,1)-IROW+1
      CALL UPDATE(I,J,II,D,IY,IY1,NWR,NROWS,IROW,B)
      POINT1=L
      M=STACK(1,2)
      N=STACK(1,3)
      DO 10 I1=M,N
      POINT1=POINT1 + 1
      STACK(POINT1,1)=IROW
      STACK(POINT1,2)=I1/30
      STACK(POINT1,3)=MOD(I1,30)+1
   10 CONTINUE
      POINT2=POINT1+1
C-
C-EXECUTE LOWER RECORD SEARCH:PLACING ELEMENTS ON STORAGE STACK
C-
      I=1
      M=L+1
      L=POINT1
   15 DO 80 I1=M,POINT1
C-
C-SET POINTERS TO WORD AND SHIFT ARRAYS
C-
      J=STACK(I1,2)
      II=STACK(I1,3)
C-
C-TEST LOWER LEFT ELEMENT
C-
      J1=J
      IF(C(II).EQ.58) J1=J-1
      KK=INTV(I,J1,II-1)
      IF(KK.NE.2)GO TO 30
      J=J1
      II=II-1
      CALL STORE(D,B,C,STACK,I,J,II,IROW,POINT2)
      GO TO 50
   30 CONTINUE
C-
C-TEST LOWER ELEMENT
C-    KK=INTV(I,J,II)
      IF(KK.NE.2) GO TO 40
      CALL STORE(D,B,C,STACK,I,J,II,IROW,POINT2)
      GO TO 50
   40 CONTINUE
C-
C-TEST LOWER RIGHT ELEMENT
C-
```

```
          J1=J
          IF(C(II).EQ.0)J1=J+1
          KK=INTV(I,J1,II+1)
          IF(KK.NE.2)GO TO 30
          J=J1
          II=II+1
          CALL STORE(D,B,C,STACK,I,J,II,IROW,POINT2)
   50     CONTINUE
C-
C-SEARCH LEFT
C-
          J1=J
          IF(C(II).EQ.59)J1=J-1
          KK=INTV(I,J1,II-1)
          IF(KK.NE.2)GO TO 60
C-ABOVE MIGHT HAVE TO BE CORRECTED IN I,J,II
          J=J1
          II=II-1
          CALL STORE(D,B,C,STACK,I,J,II,IROW,POINT2)
          GO TO 50
   60     CONTINUE
C-
C-SEARCH RIGHT
C-
          J1=J
          IF(C(II).EQ.0) J1=J1+1
          KK=INTV(I,J1,II+1)
          IF(KK.NE.2)GO TO 80
          J=J1
          II=II+1
          CALL STORE(D,B,C,STACK,I,J,II,IROW,POINT2)
          GO TO 60
   80     CONTINUE
C-
C-TEST FOR EMPTY STORAGE STACK
C-
          IF(POINT2.EQ.POINT1+1) GO TO 90
C-SHIFT WORKING STACK;REPEAT SEARCH
          M=POINT1+1
          POINT1=POINT2-1
C-
C-SHIFT RECORDS IN CORE;INPUT NEXT RECORD
C-
          CALL UPDATE(I,J,II,D,IY,IY1,MWR,NROWS,IROW,B)
   85     IROW=IROW+1
          GO TO 15
   87     CONTINUE
          PRINT *,"ERROR IN SUBND2S"
          STOP
   90     CONTINUE
          MM=POINT1-L+1
          CALL STKSRT(STACK(L+1,1),MM,M)
          M=TOP-L-1
          CALL INTNOTN(STACK(L+1,1),MM,M,LL)
          L=M
          TOP=M+L+1
          RETURN
```

```
SUBROUTINE TEST      74/74    OPT=1                    FTN 4.5+414      06/01/77  09.34.51      PAGE     3

 115            END
```

```
PROGRAM IMGPRNT     74/74    OPT=1                        FTN 4.5+414         06/01/77  09.37.42        PAGE      1

      PROGRAM IMGPRNT(INPUT,OUTPUT,TAPE1)
      DIMENSION IN(200),LINE(132),ICHAR(64)
      NAMELIST/SIZE/NROWS,NCOLS,IROW,ICOL
C-
C-THIS PROGRAM PRINTS THE THRESHOLDED IMAGE OF 6-BIT PIXELS ON THE LINE
C-PRINTER ASSIGNING EACH INTENSITY VALUE A CHARACTER,DETERMINED FROM INPUT CARD.
C-
C-INPUT PARAMETERS
C-NROWS---LAST RECORD OF IMAGE TO BE PRINTED(ASSUMES 500)
C-NCOLS---LAST SAMPLE(PIXEL) OF IMAGE TO BE PRINTED (ASSUMES 500)
C-IROW---STARTING RECORD OF IMAGE TO BE PRINTED (ASSUMES 1)
C-ICOL---STARTING SAMPLE(PIXEL) OF EACH RECORD TO BE PRINTED(ASSUMES1)
C-F
C-ICHAR---ARRAY OF 64 CHARACTERS TO REPRESENT INTENSITIES 0-63 RESPECTIVELY ON
C-        PRINTED IMAGE
C-
C-ASSIGN DEFAULT VALUES TO $SIZE$ PARAMETERS
C-
      NROWS=NCOLS=500
      IROW=ICOL=1
C-
C-CARD INPUT PHASE:PRINT INPUT PARAMETERS
C-
      PRINT 100
      READ SIZE
      PRINT SIZE
      READ 110,(ICHAR(I),I=1,64)
      PRINT 120
      DO 10 I=1,8
      PRINT 130,I-1,ICHAR(I),I+7,ICHAR(I+8),I+15,ICHAR(I+16),I+23,
     1ICHAR(I+24),I+31,ICHAR(I+32),I+39,ICHAR(I+40),I+47,ICHAR(I+48),
     2I+55,ICHAR(I+56)
10    CONTINUE
C-
C-DETERMINING RECORD LENGTH IN WORDS AND START PAGE LOOP
C-
      NWR=NCOLS/10
      IF(NWR*10.LT.NCOLS)NWR=NRW+1
      PRINT 140
      N=130
      DO 60 II=ICOL,NCOLS,130
C-
C-SKIP FORWARD TO FIRST RECORD
C-
      IF(IROW.EQ.1) GO TO 20
      L=IROW-1
      DO 15 I=1,L
      BUFFER IN(1,1)(IN(1),IN(2))
      IF(UNIT(1))15,999,15
15    CONTINUE
20    CONTINUE
      PRINT 190
      DO 55 I=IROW,NROWS
      BUFFER IN(1,1)(IN(1),IN(NWR))
      IF(UNIT(1))30,999,60
30    CONTINUE
      IF(II+N.GT.NCOLS)N=NCOLS-II+1
```

```
             CALL UN6PCK(IN,LINE,II,N)
             DO 40 J=1,N
       40    LINE(J)=ICHAR(LINE(J)+1)
             PRINT 150,(LINE(J),J=1,N)
       55    CONTINUE
              REWIND 1
       60    CONTINUE
             STOP
      999    CONTINUE
             PRINT *," EOF ON TAPE1,RECORD =  ",I
             STOP
      100    FORMAT(1H1)
      110    FORMAT(64A1)
      120    FORMAT(1H1,*TABLE OF CHARACTERS USED FOR INTENSITIES  *////)
      130    FORMAT(1H0,I2,* - *,A1,7(8X,I2,* - *,A1))
      140    FORMAT(1HT)
      150    FORMAT(1H ,130A1)
             END
```

```
PROGRAM PARSER     74/74    OPT=1                              FTN 4.5+414        06/01/77  09.31.42        PAGE     1

            PROGRAM PARSER(INPUT,OUTPUT,TAPE3,TAPE4,TAPE5)
      C-
      C-PROGRAM PARSER GENERATES THE INTERIOR PARTICLE VALUES IN THE THRESHOLDED IMAGE
      C-
      C-
      C-NROWS-- THE NUMBER OF ROWS IN INPUT IMAGE
      C-NCOLS-- THE NUMBER OF COLUMNS PER ROW IN THE INPUT IMAGE
      C-TAPE3-- CONTAINS THE THRESHOLDED IMAGE (INPUT).
      C-TAPE4-- USED AS SCRATCH FILE
      C-TAPE5--CONTAINS THE OUTPUT IMAGES WITH INTERIOR VALUES GENERATED.
      C-
            DIMENSION B(3),C(32),D(1500)
            INTEGER B,C,D
            READ 1000,NROWS,NCOLS
      1000  FORMAT(2I5)
            IFLAG=0
            IY=3
            IY1=4
            NWR=NCOLS/2
      C-
      C-INITIALIZE I POINTER ARRAY (B) AND SHIFT ARRAY (C)
      C-
            B(1)=1
            B(2)=NWR+1
            B(3)=B(2)+NWR
            C(32)=58
            C(1)=0
            KE=60
            DO 10 I=2,31
            KE=KE-2
      10    C(I)=KE
            K=1
            PRINT 5000,C
            PRINT 5000,B
      5000  FORMAT(1H1,31I4)
      C-
      C-    INITIALIZE FIRST "RECORD" OF D TO 0'S
      C-
            D(1)=0
            CALL MVW1(D,1,D,2,NWR-1)
      C-
      15    CONTINUE
      C-
      C-START FIRST PASS--FIND AND CONVERT PROPER 1-VALUED PIXELS TO 2-VALUED PIXELS
      C-
      C-    BUFFER IN 1ST 2 INPUT RECORDS  INTO 2ND, 3RD RECORD POSITIONS OF D
            BUFFER IN (IY,1) (D(B(2)+1),D(B(3)-2))
            IWAIT=UNIT(IY)
            BUFFER IN(IY,1) (D(B(3)+1),D(B(3)+NCOLS))
            IWAIT=UNIT(IY)
            IF(IY.EQ.3) GO TO 25
            D(B(2))=0
            D(NWR*2)=0
      20    IF(IY.EQ.3) GO TO 25
            D(B(3))=0
            D(NWR*3)=0
      C-
      C-USE INT123 TO SEARCH FOR   2'S AND 3'S (CONVERSION)
      C-
      25    CONTINUE
            CALL INT123(D,B,C,K,NWR)
      C-
      C-WRITE OUT FIRST "RECORD" FROM D
            BUFFER OUT(IY1,1) (D(1),D(NWR))
            IWAIT=UNIT(IY1)
            IF(IFLAG.EQ.1) GO TO 30
      C-
      C-SHIFT D  TACK UP 1 "RECORD",AND BUFFER IN NEXT RECORD
      C-
            CALL MVW1(D,B(2),D,1,NWR)
            CALL MVW1(D,B(3),D,B(2),NWR)
            BUFFER IN (IY,1) (D(B(3)+1),D(B(3)+NCOLS))
            IF(UNIT(IY).NE.0) GO TO 20
            IFLAG=1
            D(B(3))=0
            CALL MVW1(D,B(3),D,B(3)+1,NWR-1)
            GO TO 20
      30    BUFFER OUT(IY1,1) (D(B(2)),D(B(3)-1))
            IWAIT=UNIT(IY1)
            BUFFER OUT(IY1,1) (D(B(3)),D(B(3)+NWR-1))
            IWAIT=UNIT(IY1)
            ENDFILE IY1
            REWIND IY
            REWIND IY1
      C-
      C-REPEAT ABOVE PROCEDURE FOR SECOND PASS: FIND AND CONVERT PROPER 2-VALUED PIXEL
      C- TO 3-VALUED PIXELS
      C-
            IY1=3
            IY=4
            K=2
            BUFFER IN(IY,1) (D(1),D(NWR))
            IWAIT=UNIT(IY)
            IFLAG=0
            K=2
            BUFFER IN(IY,1) (D(B(2)),D(B(3)-1))
            IWAIT=UNIT(IY)
            BUFFER IN(IY,1) (D(B(3)),D(B(3)+NWR-1))
            IWAIT=UNIT(IY)
      40    CALL INT123(D,B,C,K,NWR)
            BUFFER OUT(IY1,1) (D(1),D(NWR))
            IWAIT=UNIT(IY1)
            IF(IFLAG.EQ.1)GO TO 50
            CALL MVW1(D,B(2),D,1,NWR)
            CALL MVW1(D,B(3),D,B(2),NWR)
            BUFFER IN(IY,1) (D(B(3)),D(B(3)+NWR-1))
            IF(UNIT(IY).NE.0)GO TO 40
            IFLAG=1
            D(B(3))=0
            CALL MVW1(D,B(3),D,B(3)+1,NWR-1)
            GO TO 40
      50    CONTINUE
            BUFFER OUT(IY1,1) (D(B(2)),D(B(3)-1))
            IWAIT=UNIT(IY1)
      ENDFILE IY1
      REWIND IY
      REWIND IY1
      IY1=4
      IY=3
      STOP
      END
      SUBROUTINE INT123(A,B,C,N,M)
C-
C-THIS SUBROUTINE GENERATES THE INTERIOR VALUES FOR THE PARTICLES. ON THE FIRST
C-THROUGH THE IMAGE. IT CONVERTS 1-VALUED PIXELS TO 2-VALUED PIXELS WHERE THE
C-PROPER CONDITIONS ARE PRESENT; ON THE SECOND PASS THROUGH THE IMAGE, 2-VALUED
C-PIXELS ARE CONVERTED TO 3-VALUED PIXELS.WHERE THE PROPER CONDITIONS ARE
C-PRESENT.
C-
      DIMENSION A(1),B(1),C(1)
      INTEGER A,B,C
      INTV(I,J,II)=AND(SHIFT(A(B(I)+J),-C(II)),3)
      MSK1=3
      MSK3=COMPL(MSK1)
      MM=M-1
      I=2
      K=N
      DO 10 J=1,MM
      DO 10 II=2,31
      KK=INTV(I,J,II)
      IF(KK.LT.K)GO TO 10
      IF(K.EQ.1)GO TO 5
C-
C-TEST UPPER LEFT ELEMENT
C-
      IF(C(II).NE.58)KK=INTV(I-1,J,II-1)
      IF(C(II).EQ.58)KK=INTV(I-1,J-1,II-1)
      IF(KK.LT.K)GO TO 10
C-
C-TEST UPPER RIGHT ELEMENT
C-
      IF(C(II).NE.0)KK=INTV(I-1,J,II+1)
      IF(C(II).EQ.0)KK=INTV(I-1,J+1,II+1)
      IF(KK.LT.K) GO TO 10
C-
C-    TEST LOWER LEFT ELEMENT
C-
      IF(C(II).NE.58)KK=INTV(I+1,J,II-1)
      IF(C(II).EQ.58) KK=INTV(I+1,J-1,II-1)
      IF(KK.LT.K)GO TO 10
C-
C-TEST LOWER RIGHT ELEMENT
C-
      IF(C(II).NE.0) KK=INTV(I+1,J,II+1)
      IF(C(II).EQ.0) KK=INTV(I+1,J+1,II+1)
      IF(KK.LT.K)GO TO 10
C-
C-TEST UPPER ELEMENT
C-
5     KK=INTV(I-1,J,II)
      IF(KK.LT.K)GO TO 10
C-
C-TEST RIGHT ELEMENT
C-
      IF(C(II).NE.0)KK=INTV(I,J,II+1)
      IF(C(II).EQ.0)KK=INTV(I,J+1,II+1)
      IF(KK.LT.K)GO TO 10
C-
```

```
      C-TEST LEFT ELEMENT
      C-
            IF(C(II).NE.58)KK=INTV(I,J,II-1)
            IF(C(II).EQ.58)KK=INTV(I.J-1.II-1)
            IF(KK.LT.K)GO TO 10
      C-
      C-TEST LOWER ELEMENT
      C-
            KK=INTV(I+1.J.II)
            IF(KK.LT.K)GO TO 10
            KT=B(I)+J
            LT=C(II)
      C-
      C-LOAD THE NEW VALUE INTO ITS POSITION IN CORE
      C-
            A(KT)=OR(SHIFT(K+1.LT).AND(A(KT).SHIFT(MSK3.LT)))
      10    CONTINUE
            RETURN
            END
```

```
              SUBROUTINE MVW1(A,IFROM,B,ITO,K)
        C-
        C-THIS SUBROUTINE TRANSFERS K WORDS FROM CORE.BEGINNING WITH LOCATION A(IFROM).
        C-TO CORE.BEGINNING WITH LOCATION B(ITO).
        C-
              DIMENSION A(1).B(1)
              INTEGER A,B
        100   FORMAT(* INSIDE MVW1   *,3(5X,I10))
              I=IFROM
              J=ITO
              DO 10 L=1,K
              B(J)=A(I)
              I=I+1
        10    J=J+1
              RETURN
              END
```

```
PROGRAM PARPLOT     74/74    OPT=1                          FTN 4.5+414        06/01/77  09.37.37        PAGE     1

              PROGRAM PARPLOT(INPUT,OUTPUT,PLOT,TAPE1)

        C-
        C-THIS PROGRAM DRAWS THE IMAGE PARTICLES ON THE PLOTTER AS ELLIPSES: USING AS
        C-INPUT THE OUTPUT TAPE OF PROGRAM GENPRO,WHICH CONTAINS PARTICLE GEOMETRIC DATA
        C-
              REAL MAX,MAXEX
        C-
        C-SKIP DOWN TO DATA
        C-
              DO 10 I=1,58
        10    READ(1,100) A
        100   FORMAT(A10)
        C-
        C-READ MAX FOR SCALING
        C-
              READ(1,110) XMAX,YMAX
        110   FORMAT(10X,2F10.2)
              MAX=XMAX
              IF(YMAX.GT.MAX)MAX=YMAX
        C-
        C-COMPUTE SCALING FACTOR
        C-
              SCAL=9.0/MAX
        C-
        C-INITIALIZE PLOT
        C-
              CALL PLOT(0.,-3.,-3)
              CALL PLOT(0.,1.5,-3)
        C-
        C-READ DATA RECORD
        C-
        20    READ(1,120) NUM,XCEN,YCEN,AREA,ASPAT,ANGLE,MAXEX
              IF(EOF(1)) 999,30
        120   FORMAT(I10,3F10.2,30X,3F10.2)
        C-
        C-COMPUTE ELLIPSE PARAMETERS
        C-
        30    CONTINUE
              IF(AREA.LT.1000) GO TO 20
              ANGLE=ANGLE+90.0
              RMAJ=(MAXEX/2.0)*SCAL
              RMIN=RMAJ/ASPAT
              X=(RMAJ*COS(ANGLE))+(XCEN*SCAL)
              Y=(RMAJ*SIN(ANGLE))+(YCEN*SCAL)
        C-
        C-DRAW ELLIPSE
        C-
              CALL ELIPS(X,Y,RMAJ,RMIN,ANGLE,0.0,360.0,3)
              XCEN=XCEN*SCAL
              YCEN=YCEN*SCAL
              Z=FLOAT(NUM)
              CALL NUMBER (XCEN,YCEN,0.2,Z,ANGLE,0)
              GO TO 20
        999   CALL PLOTE
              STOP
              END
```

```
PROGRAM THRSPRN     74/74    OPT=1                          FTN 4.5+414         06/01/77  09.38.14        PAGE     1

                PROGRAM THRSPRN(INPUT,OUTPUT,TAPE1)
                DIMENSION IN(200),IOUT(120)
          C-
          C-THIS PROGRAM PRINTS THE THRESHOLDED IMAGE ON THE LINE PRINTER
          C-NROWS---LAST RECORD OR
          C-NROWS---LAST RECORD OF IMAGE TO BE PRINTED(ASSUMES 500)
          C-NCOLS---LAST SAMPLE OF IMAGE TO BE PRINTED(ASSUMES 500)
          C-IROW---STARTING RECORD OF IMAGE TO BE PRINTED(ASSUMES 1)
          C-ICOL---STARTING SAMPLE OF IMAGE TO BE PRINTED(ASSUMES 1)
          C-
                NAMELIST/SIZE/NROWS,NCOLS,IROW,ICOL
          C-
          C-ASSIGN DEFAULT VALUES TO INPUT PARAMETERS
          C-
                NROWS=NCOLS=500
                IROW=ICOL=1
                PRINT 110
                READ SIZE
                PRINT SIZE
                NWP=(NCOLS*2)/60
                IF(((NWR*60)/2).LT.NCOLS)NWP=NWP+1
                M=4
                DO 60 II=ICOL,NCOLS,120
                PRINT 120
          C-
          C-SKIP FORWARD TO STARTING RECORD
          C-
                IF(IROW.EQ.1) GO TO 20
                L=IROW-1
                DO 10 I=1,L
                BUFFER IN(1,1) (IN(1),IN(2))
                IF(UNIT(1)) 10,5,10
          10    CONTINUE
          20    CONTINUE
                IB=(((II-1)*2)/60)+1
                IF(IB+3.GT.NWP)M=NWR-IB+1
                DO 55 I=IROW,NROWS
                BUFFER IN(1,1) (IN(1),IN(NWP))
                IF(UNIT(1))30,5,60
          30    CONTINUE
                DO 50 J=1,M
                L=(J-1)*30+1
                LL=J*30
                KE=58
                JJ=IB+J-1
                DO 40 K=L,LL
                IOUT(K)=AND(SHIFT(IN(JJ),-KE),3)
                KE=KE-2
          40    CONTINUE
          50    CONTINUE
                MM=(M*60)/2
                PRINT 100,(IOUT(K),K=1,MM)
          55    CONTINUE
                REWIND 1
          60    CONTINUE
                STOP
          5     CONTINUE
          PRINT *,"EOF ON TAPE1,RECORD=  ",I
          STOP
  100     FORMAT(1H ,130I1)
  110     FORMAT(1H0)
  120     FORMAT(1H1)
          END
```

```
      PROGRAM PICTURE(INPUT,OUTPUT,TAPE1)
C-
C-PROGRAM PICTURE USES THE OUTPUT OF PROGRAM PRIM TO PRODUCE A PICTURE OF THE
C-RAW IMAGE. 6 LINES OF OVERPRINT PER PRINTED LINE ARE USED TO PRODUCE 16
C-DIFFERENT GREY LEVELS ON THE OUTPUT.
C-
      DIMENSION LINE(6,36),IN(136),LINE1(136),LINE2(136),LINE3(136),LINE
     14(136),LINE5(136),LINE6(136)
C-
C-INITIALIZE GREY LEVEL TABLE
C-
      DATA (LINE(1,I1),I1=1,36)/1HM,1HM,1HM,1HM,1HM,1HM,1HM,1HM,1HH,1HH,
     11HH,1HH,1HH,1HX,1HH,1HY,1HO,1HZ,1HW,1HM,1HN,1HH,1HJ,1HS,1H=,1HI,1H
     2*,1H+,1H+,1H=,1H:,1H-,1H.,1H-,1H',1H /
      DATA (LINE(2,I1),I1=1,35)/1HW,1HW,1HW,1HW,1HW,1HW,1HW,1HW,1H#,1H#,
     11H#,1H*,1H+,1H+,1H-,1H-,1H-,1H-,1H ,1H ,1H ,1H ,1H ,1H ,1H=,1H ,1H
     2 ,1H-,1H ,1H ,1H ,1H-,1H ,1H ,1H ,1H /
      DATA (LINE(3,I1),I1=1,10)/1H#,1H#,1H#,1H#,1H#,1H#,1HJ,1HO,1H+,1H-/
      DATA(LINE(4,I1),I1=1,5)/1HO,1HO,1HO,1HJ,1HO/
      DATA(LINE(5,I1),I1=1,3)/1HO,1HP,1H#/
      DATA LINE(6,1)/1H+/
C-
C-FINISH CHARACTER SET INITIALIZATION
C-
      DO 10 I1=2,36
      LINE(6,I1)=1H
      IF(I1.GT.3)LINE(5,I1)=1H
      IF(I1.GT.5)LINE(4,I1)=1H
   10 IF(I1.GT.10)LINE(3,I1)=1H
      PRINT 110
      READ *,NP
      PRINT 140
   15 READ(1,100)(IN(I1),I1=1,NP)
      IF(EOF(1)) 999,17
   17 DO 20 I1=1,NP
      IF(IN(I1).EQ.1H*) IN(I1)=1HA
      IF(IN(I1).EQ.1H ) IN(I1)=1HA
      IN(I1)=SHIFT(IN(I1),6)
      MSK=77B
      IN(I1)=AND(IN(I1),MSK)
      LINE1(I1)=LINE(1,IN(I1))
      LINE2(I1)=LINE(2,IN(I1))
      LINE3(I1)=LINE(3,IN(I1))
      LINE4(I1)=LINE(4,IN(I1))
      LINE5(I1)=LINE(5,IN(I1))
   20 LINE6(I1)=LINE(6,IN(I1))
      PRINT 120,(LINE1(I1),I1=1,NP)
      PRINT 130,(LINE2(I1),I1=1,NP)
      PRINT 130,(LINE3(I1),I1=1,NP)
      PRINT 130,(LINE4(I1),I1=1,NP)
      PRINT 130,(LINE5(I1),I1=1,NP)
      PRINT 130,(LINE6(I1),I1=1,NP)
      GO TO 15
  999 STOP
  100 FORMAT(136A1)
  110 FORMAT(1H ,*ENTER WIDTH OF LINE(CHAR)  *)
  120 FORMAT(1H ,136A1)
```

```
PROGRAM PICTURE    74/74    OPT=1                    FTN 4.5+414        06/01/77  09.37.51        PAGE     2

          130    FORMAT(1H+,136A1)
          140    FORMAT(1H /1H1)
                 END
```

```
PROGRAM FREQPLT    74/74   OPT=1                         FTN 4.5+414       06/01/77  09.39.14       PAGE     1

             PROGRAM FREQPLT(INPUT,OUTPUT,TAPE2,PLFILE=0)
      C-
      C-THIS PROGRAM USES THE OUTPUT OF PROGRAM HISTCOC AS INPUT (TAPE2) TO PRODUCE
      C-PLOTS OF FREQUENCY VS. DENSITY FOR RAW IMAGE PIXELS
      C-
             DIMENSION X(260),Y(260)
             READ *,NTIMES
             CALL COMPRS
             DO 70 K=1,NTIMES
             I=0
       20    READ(2,100) IX,IY
             IF(EOF(2).NE.0) GO TO 30
             I=I+1
             X(I)=FLOAT(IX)
       10    Y(I)=FLOAT(IY)
             GO TO 20
       30    CONTINUE
             CALL RGNPL(J)
             CALL TITLE(17HFREQ. VS. DENSITY,17,7HDENSITY,7,9HFREQUENCY,9,8.0,8
            1.0,9.0)
             CALL GRAF(0.,30.0,255.0,0.0,0.31.0,250.00)
             CALL CURVE(X,Y,I,0)
       70    CALL ENDPL(J)
             STOP
      100    FORMAT(I8,I11)
             END
      PROGRAM SAMPLOT(INPUT,OUTPUT,TAPE5,PLOT)
C-
C-THIS PROGRAM PRODUCES PLOTS OF THE LOG OF THE AREA VS. FREQUENCY OF PARTICLES,
C-USING OUTPUT FROM PROGRAM GENTRO (TAPE5) AS INPUT
C-
      DIMENSION X(258),Y(258),FRAME(5)
      DIMENSION A(5)
5      CONTINUE
C-
C-READ TITLE OF FRAME TO BE PLOTTED FROM DATA CARD
C-
      READ 100,(A(I),I=1,5)
      IF(EOF(5LINPUT))999,6
6     CONTINUE
C-
C-READ TITLE OF NEXT FRAME ON TAPE5
C-
      READ (5,100) FRAME
      PRINT 110,(FRAME(I),I=1,5)
C-
C-INPUT DATA FROM TAPE5
C-
      DO 10 I=1,256
      READ(5,120) X(I),Y(I)
      IF(X(I).EQ.0.0) X(I)=1.0
10    CONTINUE
C-
C-TEST FOR PROPER FRAME
C-
      IF(FRAME(1).NE.A(1)) GO TO 6
C-
C-SET UP PLOT AXES AND TITLES
C-
      CALL PLOT(0.,0.,-3)
      CALL PLOT(0.0,-3.3,-3)
      CALL PLOT(0.,1.5,-3)
      CALL LGSCAL(X,20.0,256)
      YMIN=0.0
      YDEL=2.5
      CALL LGAXIS(0.,0.,4HAREA,-4,20.0,0.,X(257),X(258))
      CALL AXIS(0.,0.,4HFREQ,4,8.0,90.0,YMIN,YDEL)
      CALL SYMBOL(1.0,8.5,0.35,FRAME,0.0,20)
      Y(257)=YMIN
      Y(258)=YDEL
C-
C-PLOT LINE
C-
      CALL LGLINE(X,Y,256,5,1,-1)
      CALL PLOT(0.,0.,-3)
      CALL PLOT(25.0,0.,-3)
C-BRANCH BACK FOR NEXT PLOT
      GO TO 5
999   CONTINUE
      CALL PLOTE
      STOP
100   FORMAT(5A10)
110   FORMAT(1H ,5A10)
```

```
      120    FORMAT(F8.2,2X,F6.2)
      130    FORMAT(1H ,F8.2,5X,F6.2)
              END
```

APPENDIX G
DIGIPLT USER'S GUIDE

I. INTRODUCTION

GENERAL DESCRIPTION

DIGIPLT is an interactive data curve processor; a collection of programs that enables its user to enter, display, edit, and operate on data curves. It is an easy-to-use software package, and does not require that the user have extensive knowledge of computer processing.

While designed primarily for use on the TEKTRONIX 4014 graphics terminal, the user may take advantage of many DIGIPLT capabilities using any interactive computer terminal on-line with the ASD CDC-6600 computer in INTERCOM mode.

The user should note that a brief synopsis of this guide may be obtained at the initial stage of DIGIPLT execution by responding "YES" when asked by DIGIPLT if help is desired.

DIGIPLT is, basically, a collection of programs that are executed upon user request. The request to execute a particular program is made when the user specifies one of the five-character codes on the DIGIPLT option menu. Execution of each of the programs on the menu is completely independent of the rest; thus, the user may select menu options in any order desired.

RESPONSE TO DIGIPLT REQUESTS/QUESTIONS

All of the keyboard entries the user is required to make during DIGIPLT execution are preceded by a printed message (output by DIGIPLT) on the terminal screen specifying the input expected. The response to DIGIPLT requests must be an integer, or a "YES" or "NO," depending upon the nature of the request. There are three cases, however, when they must be real numbers.

1. When entering axis information in DGTZR.
2. When entering new X and Y values when adding or replacing points in EDTAL.
3. When entering X and Y values when creating a data file in CRTFL.

If the request is for a "YES" or "NO" response, entering only a "Y" will not suffice for a "YES," as will an "N" not suffice for a "NO." The user must enter "YES" or "NO." In most cases, any other input will be treated as a "NO" response.

II. DIGIPLT DATA FILE STRUCTURE

The purpose of this section is to describe the data files used by DIGIPLT. This information whould be of use to all DIGIPLT users, as it is essential to be familiar with the data files used by DIGIPLT, since all data processed by DIGIPLT must exist on a DIGIPLT data file in the correct format, and almost all processing is done in terms of these files.

DIGIPLT DATA FILES

The user has available five separate files on which to store data. Data is placed on these files by one of two methods: 1) the user has, prior to execution, created a data file which is compatible with the DIGIPLT data file structure, or 2) the user creates a data file through the use of the DIGIPLT options DGTZR and/or CRTFL. In the latter case, the data file is automatically formated in DIGIPLT data file structure. The user directs DIGIPLT to use a specific data file by entering its "file number" ("data file number") when requested to do so by DIGIPLT. At all times during execution of DIGIPLT, data files are referred to by their file numbers. These file numbers (integers) correspond to actual mass storage files as follows:

DIGIPLT FILE NO.	ACTUAL MASS STORAGE FILE NAME
1	Tape 1
2	Tape 2
3	Tape 3
4	Tape 4
5	Tape 5

Only those files listed above are vaiid DIGIPLT data files. Any attempt to process other files will result in an error. Data files created prior to DIGIPLT execution should be on a valid mass storate file, and be referred to during DIGIPLT execution by its corresponding file number. Similarly, files created by DIGIPLT should be referred to by their actual mass storage file name when used after completion of DIGIPLT execution.

RECORD FORMAT

Data exists on DIGIPLT data files as two real numbers per record, in free-format (separated by a blank or a comma). Each record corresponds to one data point; the first number on the record being the abscissa value, and the second number the ordinate value of that point (X, Y).

SPECIAL RECORDS (FLAGS)

There are three types of special records on DIGIPLT data files: 1) end-of-block (EOB) flag, also called end-of-curve flags, 2) end-of-data (EOD) flag, also called end-of-file (EOF) flag, and 3) header block records.

1. END-OF-BLOCK (EOB) Flag - This is a record having values of "9999.0" for both its X and Y positions. It is used to indicate

the end of a block of data, where a block of data is treated by DIGIPLT as one data (exception - see header block description) curve.

2. END-OF-DATA Flag - The last record in the data file, which has both an X and Y value of "8888.0." It should be noted that for the last block on the file there will be no EOB flag; instead, the EOD flag (EOF flag) is located in the last record. There must be only one EOD flag on a file; any data after an EOD flag on a DIGIPLT data file will be ignored and subsequently destroyed. If the EOD flag is not present, an error will occur when attempting to process the data file.

3. HEADER BLOCK RECORDS - The first block on all DIGIPLT data files contains information on the contents of the file and is used by DIGIPLT for scaling purposes. This block is three records (points) in length, and contains the following:

REC	X POSITION	Y POSITION
1	XMAX	YMAX
2	XMIN	YMIN
3	EOB	

where: XMAX and YMAX are the maximum X and Y values, respectively, of the data points on the file; XMIN and YMIN are the minimum X and Y values, respectively, of the data points on the file.

It should be noted that, due to the header block, the first actual data curve on a file is actually the second data block. The user should keep this in mind when DIGIPLT asks for "BLOCK NO." and "CURVE NO." from the user - a data curve's "BLOCK NO." will always be one greater that its "CURVE NO."

PERMANENT FILES

If the user desires to use data located on a permanent file cataloged prior to DIGIPLT execution, he must copy the file to one of the DIGIPLT data files before execution of DIGIPLT. If a permanent file is attached as one of the DIGIPLT data files, an unrecoverable error will occur in DIGIPLT execution due to the fact that DIGIPLT writes on the data files it processes, which is an illegal operation on a permanent file. The user may, however, request that a DIGIPLT data file be assigned to a permanent file device and catalog that file after DIGIPLT execution.

III. MENU OPTIONS

The following section elaborates on each of the user-selectable programs of DIGIPLT (menu options). A menu option is executed when the user enters its 5-character option code in response to the DIGIPLT query "ENTER 5-CHAR. OPTION CODE."

ALPHA-NUMERIC DATA CURVE DISPLAY (DSPAL)

The function of DSPAL is to produce a listing of the abscissa and ordinate values of points on specific data blocks. This display will indicate the block number, point number, and X and Y values for all blocks specified to be displayed on a given data file.

Since DSPAL displays are in terms of blocks, the user may display the header block of each file. Therefore, the first actual data curve on the file will be block no. 2. When asked to input the block numbers to be displayed, an entry of "0" for the first block number will cause all blocks on the specified data file to be displayed. If the user chooses to display only certain blocks, they should be entered in ascending order by their block numbers. A "0" must be entered as the last block to be displayed.

```
DSPAL
 ENTER NO. OF FILE CONTAINING DATA
  1
DIGIPLT 1.0           DSPAL        LEVEL

ENTER BLOCKS TO BE DISPLAYED IN
INCREASING ORDER. ENTER NO. '0'
FOR LAST BLOCK
BLOCK NO. 1 = 0


 BLOCK    PT. NO.         X           Y

   1          1        10.00       10.00
   1          2         0.00        0.00
   1          3      9999.00     9999.00


 BLOCK    PT. NO.         X           Y

   2          1         0.00        0.00
   2          2         1.00        1.00
   2          3         2.00        2.00
   2          4         3.00        3.00
   2          5         4.00        4.00
   2          6         5.00        5.00
   2          7        10.00       10.00
   2          8      9999.00     9999.00


 BLOCK    PT. NO.         X           Y

   3          1         0.00        0.00
   3          2        10.00        0.00
   3          3        10.00       10.00
   3          4         0.00       10.00
   3          5         0.00        0.00
   3          6      8882.00     8888.00
TYPE IN 5-CHAR. FUNCTION NAME
```

EXAMPLE OF DSPAL EXECUTION.

HERE, A "O" WAS ENTERED AS THE FIRST BLOCK TO BE DISPLAYED, RESULTING IN THE ENTIRE FILE CONTENTS BEING DISPLAYED.

```
DSPAL
ENTER NO. OF FILE CONTAINING DATA
 1
DISPLAY 1.0         DSPAL       LEVEL
ENTER BLOCKS TO BE DISPLAYED IN
ASCENDING ORDER. ENTER NO. '0'
FOR LAST BLOCK
BLOCK NO. 1 = 2
BLOCK NO. 2 = 3
BLOCK NO. 3 = 0


BLOCK   PT. NO.         X          Y

  2        1          0.00       0.00
  2        2          1.00       1.00
  2        3          2.00       2.00
  2        4          3.00       3.00
  2        5          4.00       4.00
  2        6          5.00       5.00
  2        7         10.00      10.00
  2        8       9999.00    9999.00


BLOCK   PT. NO.         X          Y

  3        1          0.00       0.00
  3        2         10.00       0.00
  3        3         10.00      10.00
  3        4          0.00      10.00
  3        5          0.00       0.00
  3        6       8888.00    8888.00
TYPE IN 5-CHAR. FUNCTION NAME
```

IN THIS EXAMPLE OF DSPAL EXECUTION, ONLY THE 2nd AND 3rd BLOCKS OF THE DATA FILE WERE DISPLAYED.

GRAPHIC DATA FILE DISPLAY (DSPGR)

The function of the DSPGR option is to produce a plot on the terminal screen of specified data curves on a specific data file. The plot produced is a simply X-Y graph of the data curves, and enables the user to examine the data in graphic form.

Since DSPAL displays are in terms of data curves, a "1," for example, entered as the data curve to be displayed will result in a plot of the first actual data curve on the specified file. As in DSPAL, a "0" entered for the first data curve to be displayed will result in the display of all data curves on the file. A "0" must be entered as the last curve to be displayed.

The criteria for scaling of the displayed graph is determined by the method in which the user chooses to display the data curves. If he elects to display all curves on a file by entering a "0" as the first curve to be displayed, scaling of the graph will be in terms of the information in the header block of the data file. If a specific curve(s) to be displayed is entered, scaling of the graph will take place in relation to the minimum and maximum abscissa and ordinate values of the data to be displayed.

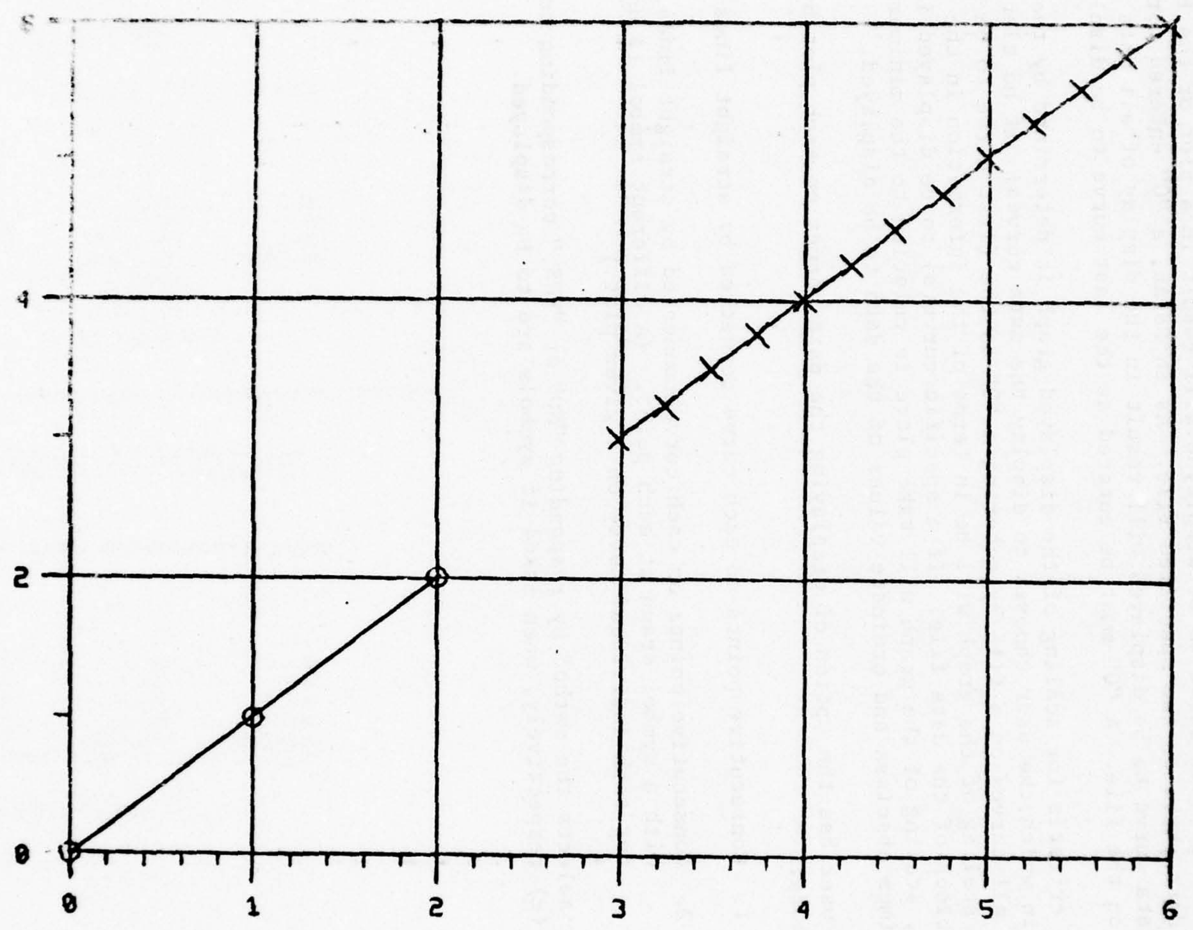
The user has the option of displaying the data curves on each plot by two methods:

1. Consecutive points on each curve connected by straight lines.

2. Consecutive points on each curve connected by straight lines, with a symbol drawn at each point. (A different symbol is used for each individual curve on a given plot.)

The user selects the method by responding "NO" or "YES," corresponding to (1) and (2) respectively, when asked if symbols are to be displayed.

```
DO YOU WANT SYMBOLS DISPLAYED ON CURVES?
YES
```



EXAMPLE OF DSPGR OUTPUT. DATA PLOTTED HERE IS DATA USED IN OSPAL EXAMPLE IN THIS GUIDE.

ALPHA-NUMERIC DATA FILE EDITOR (EDTAL)

The purpose of EDTAL is to provide the user with the capability of editing data blocks. As in DSPAL, EDTAL refers to the data file in terms of blocks, thus allowing the editing of the header block on a file.

The user has three editing functions available: 1) addition of new points to, 2) deletion of points from, and 3) replacing of points with new ones on a data block. These functions are effected by entering a one-letter edit command when asked by DSPAL to do so. Valid edit commands are as follows:

1-Letter Command	Function
A	Add point(s) to specified block
D	Delete point(s) from specified block
R	Replace point(s) on a specified block

The user is asked to enter the point number to be processed by EDTAL. This point number entered here should be the point number on the display corresponding to the position the user wishes to begin editing. In the case of deleting or replacing points, the point specified will actually be affected (example: point no. specified is 5, edit command was D, point no. 5 will be deleted). In the addition of point(s), the new point(s) will be inserted in the block immediately after the point number specified.

Both before and after executing an edit command, the user has the option of displaying the block he wishes to edit. All "point numbers" referred to in this discussion are those which appear on this display. (This display is of the same nature as that produced by DSPAL.)

When adding or replacing points, the user must enter the new X and Y values for each point. They must be entered as real numbers, separated by a blank or a comma.

The user, when specifying a deletion, has the option of deleting the entire block. If a "YES" response is given when the user asks if he wishes this done, the entire block will disappear from the data file. Note that if the last block of the file is deleted, the last record (end-of-block flag) of the preceding block must be changed from "9999.0 . 9999.0" to "8888.0 . 8888.0" to assure normal processing of that file and prevent the occurance of errors. In the same respect, the user may add an entire block to the file by adding points after the end of block or end of data (file) flag, specifying as the last X-Y pair the end-of-block or end-of-data flag accordingly. In determining proper use of the above flags, it is important to remember that any data appearing after an end-of-data flag ("8888.0 . 8888.0") will be ignored and consequently deleted from the data file.

```
TER NO. OF FILE CONTAINING DATA
2
DIGIPLT 1.0          EDTAL       LEVEL
ENTER BLOCK NO. TO BE EDITED
3
DO YOU WANT THE BLOCK CONTENTS DISPLAYED?
YES
BLOCK    PT. NO.          X              Y

  3           1          3.00           3.00
  3           2          4.00           4.00
  3           3          5.00           5.00
  3           4       8888.00        8888.00
ENTER A 1 LETTER EDIT COMMAND
A
ENTER RECORD NO. TO BE PROCESSED
3
HOW MANY POINTS DO YOU WISH TO ADD/REPLACE/DELETE?
1
ENTER X AND Y VALUES,SEPARATED BY A BLANK
OR A COMMA, FOR THE POINT NO.S SPECIFIED
4= 6. 6.
DO YOU WANT THE BLOCK CONTENTS DISPLAYED?
YES
BLOCK    PT. NO.          X              Y

  3           1          3.00           3.00
  3           2          4.00           4.00
  3           3          5.00           5.00
  3           4          6.00           6.00
  3           5       8888.00        8888.00
DO YOU WANT TO EDIT MORE IN CURRENT BLOCK?
NO
DO YOU WANT TO EDIT ANOTHER BLOCK?
NO
```

EXAMPLE OF EDTAL USE. HERE A POINT WITH COORDINATES (6.0, 6.0) WAS INSERTED AFTER THE 3rd POINT IN THE 3rd BLOCK BY USING THE ADD (A) COMMAND.

```
ENTER NO. OF FILE CONTAINING DATA
 2
DIGIPLT 1.0        EDTAL      LEVEL
 ENTER BLOCK NO. TO BE EDITED
 3
 DO YOU WANT THE BLOCK CONTENTS DISPLAYED?
 YES
 BLOCK   PT. NO.         X            Y

    3         1         3.00         3.00
    3         2         4.00         4.00
    3         3         5.00         5.00
    3         4      8888.00      8888.00
 ENTER A 1 LETTER EDIT COMMAND
 D
DO YOU WANT TO DELETE THE ENTIRE BLOCK?
 NO
 ENTER RECORD NO. TO BE PROCESSED
 3
HOW MANY POINTS DO YOU WISH TO ADD/REPLACE/DELETE?
 1
 DO YOU WANT THE BLOCK CONTENTS DISPLAYED?
  YES
 BLOCK   PT. NO.         X            Y

    3         1         3.00         3.00
    3         2         4.00         4.00
    3         3      8888.00      8888.00
 DO YOU WANT TO EDIT MORE IN CURRENT BLOCK?
 YES
 ENTER A 1 LETTER EDIT COMMAND
 A
 ENTER RECORD NO. TO BE PROCESSED
 2
HOW MANY POINTS DO YOU WISH TO ADD/REPLACE/DELETE?
 1
ENTER X AND Y VALUES,SEPARATED BY A BLANK
OR A COMMA, FOR THE POINT NO.S SPECIFIED
3. 5. 5.
 DO YOU WANT THE BLOCK CONTENTS DISPLAYED?
```

```
 YES
 BLOCK   PT. NO.         X            Y

    3         1         3.00         3.00
    3         2         4.00         4.00
    3         3         5.00         5.00
    3         4      8888.00      8888.00
DO YOU WANT TO EDIT MORE IN CURRENT BLOCK?
NO
DO YOU WANT TO EDIT ANOTHER BLOCK?
NO
TYPE IN 5-CHAR. FUNCTION NAME
```

IN THIS EXAMPLE OF EDTAL USE, THE 3rd POINT OF THE 3rd BLOCK WAS REMOVED USING THE DELETE (D) COMMAND; AND THEN PUT BACK USING THE ADD (A) COMMAND.

```
ENTER NO. OF FILE CONTAINING DATA
 2
DIGIPLT 1.0          EDTAL       LEVEL

ENTER BLOCK NO. TO BE EDITED
2
DO YOU WANT THE BLOCK CONTENTS DISPLAYED?
YES
BLOCK    PT. NO.            X              Y

  2          1            0.00           0.00
  2          2            1.00           1.00
  2          3            2.00           2.00
  2          4         8888.00        8888.00
ENTER A 1 LETTER EDIT COMMAND
R
ENTER RECORD NO. TO BE PROCESSED
4
HOW MANY POINTS DO YOU WISH TO ADD/REPLACE/DELETE?
1
ENTER X AND Y VALUES,SEPARATED BY A BLANK
OR A COMMA, FOR THE POINT NO.S SPECIFIED
4- 9999. 9999.
DO YOU WANT THE BLOCK CONTENTS DISPLAYED?
 YES
BLOCK    PT. NO.            X              Y

  2          1            0.00           0.00
  2          2            1.00           1.00
  2          3            2.00           2.00
  2          4         9999.00        9999.00
DO YOU WANT TO EDIT MORE IN CURRENT BLOCK?
YES
ENTER A 1 LETTER EDIT COMMAND
A
ENTER RECORD NO. TO BE PROCESSED
4
HOW MANY POINTS DO YOU WISH TO ADD/REPLACE/DELETE?
4
ENTER X AND Y VALUES,SEPARATED BY A BLANK
OR A COMMA, FOR THE POINT NO.S SPECIFIED
5- 3. 3.
6- 4. 4.
7- 5. 5.
8- 8888. 8888.
 DO YOU WANT THE BLOCK CONTENTS DISPLAYED?
  YES
 BLOCK    PT. NO.            X              Y

   2          1            0.00           0.00
   2          2            1.00           1.00
   2          3            2.00           2.00
   2          4         9999.00        9999.00
   2          5            3.00           3.00
   2          6            4.00           4.00
   2          7            5.00           5.00
   2          8         8888.00        8888.00
 DO YOU WANT TO EDIT MORE IN CURRENT BLOCK?
 NO
 DO YOU WANT TO EDIT ANOTHER BLOCK?
 NO
TYPE IN 5-CHAR. FUNCTION NAME
```

IN THIS EXAMPLE, AN ADDITIONAL BLOCK (CURVE) WAS ADDED TO THE END OF DATA FILE BY 1) REPLACING THE EOD FLAG (8888. 8888.) BY AN EOB FLAG (9999. 9999.) AND ADDING THE POINTS OF THE NEW CURVE, WITH AN EOD FLAG AS THE LAST POINT.

DATA CURVE INTERPOLATION (INTRP)

The function of the INTRP option is to interpolate points between the existing points on a specified data curve(s). As in DSPAL and DSPGR, a "0" entered for the first curve to be processed will result in interpolation on all curves on the specified data file; if specific curve(s) are to be processed, a "0" must be entered as the last curve to be interpolated.

The user is asked to enter the number of intervals to be interpolated between the original points on the data curve(s). In order for interpolation to occur, this number must be in the range of 2 - 10 inclusive. INTRP will interpolate and insert between the original points of the data curve(s) the number of points equal to one less than the number of intervals specified. (Thus, if an interval of "1" was specified, no points would be interpolated.)

Due to limited storage space in the computer, individual data curves are limited to having a maximum of 900 points each. Should the number of intervals specified result in a curve of length greater than 900 points, the largest interval which will contain the curve to the maximum 900 points is automatically used by INTRP.

In the actual interpolation, INTRP uses a 3d degree, piece-wise fitting of the original data points to determine the value of interpolated points. The abscissa values of the interpolated points are determined by the number of intervals specified and the abscissa values of the original points; their ordinate values are determined by the abscissa values and the fit. It is mandatory that any curve being interpolated by continuous, and the abscissa values be increasing as their point numbers increase.

```
DSPAL

 ENTER NO. OF FILE CONTAINING DATA
  2

DIGIPLT 1.0             DSPAL       LEVEL

ENTER BLOCKS TO BE DISPLAYED IN
INCREASING ORDER. ENTER NO. '0'
FOR LAST BLOCK
BLOCK NO. 1 = 2

BLOCK NO. 2 = 0


 BLOCK    PT. NO.          X           Y

               1         3.00        3.00
               2         4.00        4.00
               3         5.00        5.00
               4         6.00        6.00
               5      8888.00     8888.00
TYPE IN 5-CHAR. FUNCTION NAME

 ENTER NO. OF FILE CONTAINING DATA
  2

ENTER CURVES TO BE PROCESSED
IN INCREASING ORDER.ENTER NO. 0
FOR LAST CURVE
 CURVE NO. 1  2

 CURVE NO. 2  0

DIGIPLT 1.0             INTRP       LEVEL


ENTER NO. OF INTERVALS TO BE INTERPOLATED
BETWEEN ORIGINAL DATA CURVE POINTS(MAX. OF 10)
 4

END INTRP-----NORMAL TERMINATION
TYPE IN 5-CHAR. FUNCTION NAME
 DSPAL

 ENTER NO. OF FILE CONTAINING DATA
  2

DIGIPLT 1.0             DSPAL       LEVEL

ENTER BLOCKS TO BE DISPLAYED IN
INCREASING ORDER. ENTER NO. '0'
FOR LAST BLOCK
BLOCK NO. 1 = 3
```

```
BLOCK NO. 2 = 0


 BLOCK    PT. NO.          X           Y

   3         1          3.00        3.00
   3         2          3.25        3.25
   3         3          3.50        3.50
   3         4          3.75        3.75
   3         5          4.00        4.00
   3         6          4.25        4.25
   3         7          4.50        4.50
   3         8          4.75        4.75
   3         9          5.00        5.00
   3        10          5.25        5.25
   3        11          5.50        5.50
   3        12          5.75        5.75
   3        13          6.00        6.00
   3        14       8888.00     8888.00
TYPE IN 5-CHAR. FUNCTION NAME
```

INTERPOLATED POINTS

THIS IS AN EXAMPLE OF INTRP USE. THE DSPAL MENU OPTION WAS USED TO DISPLAY THE CURVE (BLOCK) BEFORE AND AFTER INTERPOLATION.

DATA FILE STATISTICS AND COPYING (FILES)

The function of the FILES option is to provide the user with the status of each of the five data files at his disposal, and enable the copying of one data file to another.

The number of data curves, the number of points on each curve, and the total number of points contained in the file are given for each file. If a file contains no data, the message "FILE EMPTY" is given for that file.

In copying files, any valid data file may be copied to any other valid data file. If the user attempts any copy involving non-existent files, an error message is printed and no copying occurs. If a file that is specified to be copied to contains data prior to the copy, a warning is given; and if the user elects to proceed with the copy, any data previously on the file will be destroyed.

```
FILES

DIGIPLT 1.0          FILES      LEVEL

FILE TAPE1 STATISTICS

CURVE NO. 1 ----- 2 DATA POINTS
CURVE NO. 2 ----- 3 DATA POINTS

CURVE NO. 3 ----- 13 DATA POINTS

3 CURVES; 18 TOTAL POINTS

FILE TAPE2 STATISTICS

CURVE NO. 1 ----- 2 DATA POINTS
CURVE NO. 2 ----- 3 DATA POINTS

CURVE NO. 3 ----- 13 DATA POINTS

3 CURVES; 18 TOTAL POINTS

FILE TAPE3 STATISTICS

FILE EMPTY

FILE TAPE4 STATISTICS

FILE EMPTY

FILE TAPE5 STATISTICS

FILE EMPTY


ALL DATA FILES HAVE BEEN REWOUND
IF YOU WISH TO COPY ONE FILE TO ANOTHER,TYPE 'COPY'.
OTHERWISE,TYPE ANYTHING
 COPY

ENTER NO. OF FILE TO BE COPIED
 1

ENTER NO. OF FILE TAPE1 IS TO BE COPIED TO
 3

COPY COMPLETED

ALL DATA FILES HAVE BEEN REWOUND
IF YOU WISH TO COPY ONE FILE TO ANOTHER,TYPE 'COPY'.
OTHERWISE,TYPE ANYTHING
 U

TYPE IN 5-CHAR. FUNCTION NAME
```

EXAMPLE OF FILES EXECUTION.
HERE DATA FILE #1 WAS COPIED TO
DATA FILE #3.

DATA FILE CREATION FROM KEYBOARD ENTRY (CRTFL)

The purpose of the CRTFL option is to provide the user with the capability of creating data curve(s) by entering X-Y values from the terminal keyboard. Data curves may be created on any valid DIGIPLT data file. (A warning will be issued if an attempt is made to create data curves on a file currently containing data.)

The process of the creation is a point-by-point entry of data curves. An end of block flag entered (9999.0 . 9999.0) as the X and Y values results in the beginning of a new curve on the file. The last point entered must be an end-of-data flag ("8888.0 . 8888.0"); this results in the completion of the data file creation. CRTFL will continue to ask for X-Y values until the end-of-data flag is entered. An end-of-block flag must not be entered on the last block to be created. Each point must be entered as an X and a Y value, separated by a blank or comma, on the same line.

The header block required on all DIGIPLT data files is automatically created and placed in the correct location on the file created.

```
DIGIPLT 1.0            CONTR       LEVEL

DO YOU NEED HELP?
 NO

DIGIPLT 1.0            TRCKR       LEVEL

TYPE IN 5-CHAR. FUNCTION NAME
 CRTFL

DIGIPLT 1.0            CRTFL       LEVEL

ENTER NO. OF FILE TO BE CREATED
 1

ENTER AN X VALUE AND A Y VALUE(MUST BE READ NO.S)
ON EACH LINE,SEPARATED BY A BLANK OR A COMMA
THE X AND Y VALUES OF THE LAST POINT ON THE FILE
MUST BE '8888.0'. ENTER X AND Y VALUES OF '9999.0'
TO INDICATE THE BEGINNING OF A NEW DATA CURVE
DATA CURVE NO. 1
POINT NO.     X  Y
1=0. 0.

2=1. 1.

3=2. 2.

4=3. 3.

5=4. 4.

6=5. 5.

7=10. 10.

8=9999. 9999.

DATA CURVE NO. 2
POINT NO.     X  Y
1=0. 0.

2=10. 0.

3=10. 10.

4=0. 10.

5=0. 0.

6=8888. 8888.

DATA FILE TAPE1
NO. OF CURVES = 2 TOTAL NO. OF POINTS = 12
TYPE IN 5-CHAR. FUNCTION NAME
```

IN THIS EXAMPLE OF CRTFL USE,
DATA FILE #1 WAS CREATED FROM
KEYBOARD ENTRY OF POINTS.

DIGITIZER INPUT OPTION (DGTZR)

This option should be executed only if using a TEKTRONIX 4014 graphics terminal, with a TEKTRONIX 4954 digitizing tablet. Any mention of the "digitizer" in the following discussion refer to the 4954 tablet and controller.

TABLET OPERATION

This option provides the user with the capability of entering data curves by digitizing graphs. If it is intended to use this option, it is advisable to turn on the tablet controller before beginning execution of DIGIPLT.

Secure the graph to be digitized to the tablet with four small pieces of adhexive tape. It is advisable to place the graph near to the center of the tablet, as the usable area of the tablet does not necessarily extend to its edges. DIGIPLT automatically corrects for skew of the graph with respect to the tablet. While digitizing do not lift the mouse from the tablet. If this happens while the digitizer is armed (ready light on controller is on), the ready status of the digitizer will be lost. In this case, return the mouse to the tablet and move it around until the ready status returns. Do not key the mouse unless the tablet is armed. To digitize a point, place the crosshairs of the mouse over the point, and key the mouse once. Do not move the mouse while it is keyed. DIGIPLT automatically arms the tablet when it expects input from the digitizer.

GRAPH FORMAT

Any graph that is to be digitized must have an X (abscissa) axis, and a Y (ordinate) axis perpendicular with respect to the X axis, associated with the graph (i.e. must have a cartesian coordinate system).

It is also required that there be associated with these axes:

1. an abscissa and ordinate value at the origin.
2. an abscissa value at the maximum extent (pos. sense) of the X axis.
3. an ordinate value at the maximum extent (pos. sense) of the Y axis.

These values may be arbitrary. The points digitized will have coordinates valued with respect to the above. To establish the relation between the values of the points above to the coordinate values output by the tablet, DGTZR will ask the user to type in the values, (1, 2, and 3 above) and then digitize the corresponding points on the graph.

At this point, DGTZR will ask the user to input (digitize) the curve with the mouse, entering a "D" from the keyboard when finished digitizing that curve. When entering a curve, move the mouse along the desired curve, keying it at the locations where actual points are to be taken. Begin at the left-most point on the curve and proceed to its end. The output of the digitizer is echoed on the terminal screen (appears as "strange" character strings). If many points are being digitized, be careful that the terminal screen does not become "full," causing the terminal to stop receiving data from the digitizer. When the curve has been digitized, type a "D" and a carriage return on the keyboard. If at this time there is another curve to be digitized, respond "YES" when asked by DGTZR if another curve is to be input. A "NO" response here will result in final processing of the data entered and exit from DGTZR.

EXIT FROM MENU OPTION SELECTION MODE (FINSH)

The purpose of the FINSH option is to exit from the user-selectable option mode. This option is executed when the user has completed processing all data. Upon exiting, DIGIPLT will ask the user if he is finished. A "YES" reply given at this point will result in the termination of DIGIPLT execution. All data will remain on the DIGIPLT data files, for the user to do with as he wishes (example: catalog as permanent file, dispose to a batch terminal, or use as input to another program). A "NO" response will result in the return of the user-selectable option mode, and the user may continue processing.

IV. USE OF DIGIPLT FROM TERMINAL OTHER THAN TEKTRONIX 4014

DIGIPLT, while primarily designed to be executed using a TEKTRONIX 4014 terminal, can be executed using any terminal on-line with the CDC-6600 (INTERCOM). Two restrictions are observed:

1. DO NOT SELECT DGTZR OPTION.
2. DO NOT SELECT DSPGR OPTION.

The procedures for use of all DIGIPLT options other than those above from a terminal other than the TEKTRONIX 4014 are exactly as outlined in this guide.

V. GENERAL INFORMATION

EXECUTION OF DIGIPLT

The DIGIPLT program is maintained in absolute binary form on the AFML program library. To execute DIGIPLT, the user must login to INTERCOM on the CDC-6600, attach DIGIPLT from the library, and execute. Since DIGIPLT is in absolute form it requires no additional supporting software.

DIGIPLT is attached and executed by the following commands:

```
ATTACH,DIGIPLT,DIGIPLT,ID=M754321.
DIGIPLT.
```

These commands must be given exactly as shown. Any variations will result in execution errors.

The user should become, if not already, familiar with the use of CDC-6600 INTERCOM. Information on its use is available from AFML/DOC.

IF DIGIPLT BOMBS

Should the user cause DIGIPLT to bomb, all data files are preserved in the state they were in immediately before the termination of execution. INTERCOM commands may be used to examine data (ex.: COPYSBF, TAPE1, OUTPUT) attempt to ascertain the cause of the error, if not known already. The user may elect to use DIGIPLT to examine the data. In this case, it is strongly advisable to select the FILES option before any other, as this will give the user the most information on the data files.

REPORTING BUGS AND ERRORS

Should the user discover a bug or error in DIGIPLT, he is asked to document and report the problem to AFML/DOC. The documentation need only be the terminal output sufficient to show the existence of the bug or error, or the equivalent explanation.

ADDITIONAL INFORMATION

Manuals on the use of INTERCOM, the TEKTRONIX 4014 terminal, and TEKTRONIX 4954 digitizing tablet are available at AFML/DOC, should the user desire more information on these.

VI. SPECIFICATIONS

EXECUTION REQUIREMENTS

Central Memory	60000_8 words (60-bit)
Time	Dependent upon length of "session" of use
Mode	Interactive. Must use CDC-6600 INTERCOM
File Requirements	Uses Mass Storage files:
	Tape 1
	Tape 2
	Tape 3
	Tape 4
	Tape 5
	Tape 99

PROCESSING LIMITATIONS

Maximum No. of Data Curves/File	50
Maximum No. of Data Points/Curve	900

APPENDIX H
GENERAL USER PROGRAMS FOR THE CDC 6600

```
PROGRAM CONVRT      74/74    OPT=1                        FTN 4.5+414        06/01/77  09.31.23        PAGE     1

      PROGRAM CONVRT(INPUT,OUTPUT,PUNCH,TAPE1=PUNCH,TAPE2)
C-
C-PROGRAM WHICH PROCESSES ELECTRON MICROPROBE DATA: RE-FORMATTING AND
C-CALCULATING THE MEAN AND STANDARD DEVIATION.
C-
C-INPUT
C-         INPUT DATA IS IN THE FORM OF PUNCHED CARDS.  THESE CARDS ARE GENERATED
C-         BY CONVERTING THE PAPER TAPE OUTPUT OF THE MICROPROBE TO PUNCHED CARDS.
C-         THE LAST CARD IN EACH DATA SET MUST HAVE AN 'E' IN THE 2ND COLUMN
C-         (MULTIPLE DATA SETS MAY BE PROCESSED DURING 1 RUN).
C-
C-OUTPUT
C-         OUTPUT IS THE RE-FORMATTED DATA (F7.0) ON MASS STORAGE FILE TAPE2 AND
C-         LINE PRINTER: AND PRINTED MEAN AND STANDARD DEVIATION.
C-
      DIMENSION X(1000)
5     CONTINUE
      I=1
C-
C-READ DATA CARDS:TEST FOR VALIDITY(IFLAG=1H2),END-OF-DATA SET(IFLAG = 1HE),AND
C-END OF FILE:PRINT DATA VALUES
C-
10    CONTINUE
      READ 100,IFLAG,X(I)
      IF(EOF(5LINPUT))50,20
20    CONTINUE
      IF(IFLAG.EQ.1H2) GO TO 30
      IF(IFLAG.EQ.1HE) GO TO 60
C-
C-PRINT ERROR MESSAGE FOR INVALID CARD
C-
      PRINT 110,I
      GO TO 10
30    PRINT 120,I,X(I)
      I=I+1
      GO TO 10
50    CONTINUE
      IF(I.NE.1) GO TO 60
      PRINT 130
      STOP
60    CONTINUE
      I=I-1
C-
C-WRITE DATA ON FILE TAPE2
C-
      DO 70 J=1,I
70    WRITE(2,140) X(J)
      ENDFILE 2
      REWIND 2
C-
C-READ DATA IN FLOATING POINT FORMAT FROM TAPE2 FOR CALCULATIONS
C-
      DO 90 J=1,I
      READ (2,150) X(J)
      IF(EOF(2))85,90
90    CONTINUE
85    CONTINUE
```

```
      C-
      C-SUBROUTINE STAT RETURNS MEAN AND STANDARD DEVIATION OF DATA
      C-
            CALL STAT(X,I,XM,S)
      C-
      C-PRINT MEAN AND STANDARD DEVIATION
      C-
            PRINT 160,XM,S
            REWIND 2
      C-
      C-BRANCH BACK TO BEGINNING FOR NEW DATA SET
            GO TO 5
      100   FORMAT(1X,A1,14X,A7)
      110   FORMAT(1H0,*ERROR IN DATA CARD NO.*,1X,I5)
      120   FORMAT(1H ,I5,5X,A7)
      130   FORMAT(1H0,*ERROR--EMPTY DATA SET*)
      140   FORMAT(A7)
      150   FORMAT(F7.0)
      160   FORMAT(1H0,/* MEAN FOR DATA IS *,F11.4,*, STANDARD DEVIATION IS  *
           1,F11.6)
      170   FORMAT(*END*)
            END
```

```
SUBROUTINE STAT       74/74   OPT=1                              FTN 4.5+414        06/01/77  09.31.23        PAGE     1

                  SUBROUTINE STAT(X,N,XM,S)
                  DIMENSION X(1)
            C-
            C-THIS SUBROUTINE CALCULATES THE MEAN (XM) AND STANDARD DEVIATION (S) FOR A
            C-STRING OF N NUMBERS IN THE ARRAY X.
            C-
                  XM=0.0
                  XSQ=0.0
            C-
            C-CALCULATE THE MEAN
            C-
                  DO 10 I=1,N
            10    XM=XM+X(I)
                  XM=XM/FLOAT(N)
            C-
            C-CALCULATE THE STANDARD DEVIATION
            C-
                  DO 20 I=1,N
            20    XSQ=XSQ+((X(I)-XM)**2)
                  S=SQRT(XSQ/FLOAT(N-1))
                  RETURN
                  END
```

```
PROGRAM WALT        74/74    OPT=1                      FTN 4.5+414         06/01/77  09.38.24         PAGE      1

            PROGRAM WALT(INPUT,OUTPUT,PLFILE=0)
            DIMENSION X(9,7)
            CALL COMPRS
            N=1
      C-
      C-THE " I " LOOP IS A COUNTER FOR THE 9-CARD DATA SETS
      C-
            DO 20 I=1,4
      C-
      C-READ IN DATA FOR 3 PLOTS(6 CURVES)
      C-
            DO 10 J=1,9
            READ 100,(X(J,L),L=1,7)
      10     CONTINUE
      C-
      C-SET UP AND PLOT 3 CURVES
      C-
            DO 5 M=1,3
            CALL BGNPL(N)
            CALL PAGE(10.0,11.0)
            CALL PHYSOR(1.0,2.0)
            CALL TITLE(18HHARDNESS VS. TIME$,100,11HTIME , HRS$,100,9HHARDNESS
           1$,100,8.0,7.0)
            CALL XLOG(0.1,2.0,30.0,10.0)
            CALL CURVE(X(1,1),X(1,2*M),9,-1)
            CALL CURVE(X(1,1),X(1,(2*M)+1),9,-1)
            CALL BLNK1(0.0,8.0,0.0,7.0,3)
            CALL ENDPL(N)
            N=N+1
      5      CONTINUE
      20     CONTINUE
            STOP
      100   FORMAT(7F6.2)
            END
```

```
PROGRAM DATPLOT    74/74   OPT=1                     FTN 4.5+414        06/01/77  09.37.59       PAGE     1

          PROGRAM DATPLOT(INPUT,OUTPUT,PLFILE=0)
      C-
      C-DATPLOT PRODUCES PLOTS OF STRESS INTENSITY VS. CRACK GROWTH RATE FOR TITANIUM
      C-ALLOY DATA.  THERE ARE 2 SETS OF LOG-LOG AXES ON EACH PLOT:ONE IN ENGLISH
      C-SYSTEM UNITS AND ONE IN METRIC SYSTEM UNITS.
      C-
      C-INPUT DATA
      C-        INPUT DATA IS IN PUNCHED CARD FORM.  THE DATA IS FREE-FORMAT:WITH ONE X
      C-        AND ONE Y VALUE PER CARD.  THERE ARE TWO SETS OF THESE CARDS: ONE FOR
      C-        EACH OF THE TWO CURVES TO BE PLOTTED.  THE FIRST CARD IN THE DATA DECK
      C-        CONTAINS TWO INTEGERS IN FREE FORMAT: THE NUMBER OF DATA POINTS IN THE
      C-        FIRST AND SECOND CURVES, RESPECTIVELY. THE 2ND THRU 4TH CARDS CONTAIN
      C-        THE 1ST THRU 3RD LINES OF THE PLOT LEGEND. EACH MUST BE NO MORE THAN
      C-        30 CHARACTERS IN LENGTH, AND ARE INPUT IN ALPHA-NUMERIC FORMAT.
      C-
      C-OUTPUT
      C-        OUTPUT CONSISTS OF THE PLOT DESCRIBED ABOVE.
      C-
            DIMENSION LEG1(3),LEG2(3),LEG3(3),LEG4(3),LEG5(2)
            DIMENSION X(40),Y(40)
            DATA LEG4/10H*TRIANGULA,10HR WAVE FOR,1HM/
            DATA LEG5/10H*5 MIN. HO,10HLD-TIME)  /
      C-
      C-READ DATA CARD: NPC1 = NO. OF DATA POINTS FOR CURVE NO. 1: NPC2 = NO. OF
      C-POINTS FOR CURVE NO. 2.
      C-
            READ *,NPC1,NPC2
      C-
      C-READ LEGEND TITLES
      C-
            READ 100,LEG1
            READ 100,LEG2
            READ 100,LEG3
      100   FORMAT(3A10)
      C-
      C-INITIALIZE PLOTTING: SET CHARACTER FONTS TO BE USED.
      C-
            CALL COMPRS
            CALL PGNPL(1)
            CALL SCMPLX
            CALL BASALF("L/CSTD")
            CALL MIXALF("INSTRUCTION")
            CALL MX3ALF("STAND","*")
            CALL MX4ALF("MATHE","+")
      C-
      C-SET UP FIRST SET OF AXES
      C-
            CALL TITLE(1H ,1,"KST +%)IN+$(EH0.5)1/2(EXHX)$",100,"DA/DN - *CRAC
           1K GROWTH RATE +%)IN/CYCLE+$)$",100,5.0,7.0)
            CALL LOGLOG(10.0,5.0,1.0E-6,2.75)
            N=NPC1
      C-
      C-LOOP 2 TIMES: READ DATA POINTS (CARD INPUT) AND PLOT 2 CURVES
      C-
            DO 20 I=1,2
      C-
      C-LOOP----READ DATA POINTS FOR CURVE
```

```
C-
      DO 10 J=1,N
10    READ *,X(J),Y(J)
      IF(I.EQ.1)CALL MARKER(2)
      IF(I.EQ.2) CALL MARKER(0)
      CALL CURVE(X,Y,N,-1)
      N=NPC2
20    CONTINUE
      CALL MARKER(0)
      CALL CURVE(3.3E1,1.3E-6,1,-1)
      CALL MARKER(2)
      CALL CURVE(3.3E1,1.6E-6,1,-1)
C-
C-SET UP SECONDARY AXES
C-
      CALL YLGAXS(3.0E-5,2.75,6.75,"MM/CYCLE$",-100,5.0,0.25)
      CALL XLGAXS(11.0,5.0,5.15,"*MP)A+%)M+&(EH0.5)1/2(EXHX)$",-100,-.15
     1,7.0)
C-
C-SET UP LEGEND
C-
      CALL MX5ALF("GREEK","#")
      CALL HEIGHT(0.1)
      CALL MESSAG(LEG5,20,2.8,0.25)
      CALL MESSAG(LEG4,21,2.8,0.55)
      CALL MESSAG(LEG3,30,2.5,0.85)
      CALL MESSAG(LEG2,30,2.5,1.15)
      CALL MESSAG(LEG1,30,2.5,1.45)
      CALL HEIGHT(0.15)
      CALL MESSAG("#D*K-STRESS INTENSITY RANGE$",100,0.6,-1.0)
C-
C-DRAW BOXES AROUND LEGEND AND ENTIRE PLOT
C-
      CALL BLNK1(2.375,5.0,0.0,1.688,1)
      CALL RESET("BLNKS")
      CALL BLNK1(0.0,5.0,0.0,7.0,1)
      CALL RESET("BLNKS")
C-
C-TERMINATE PLOTTING
C-
      CALL ENDPL(2)
      CALL DONEPL
      STOP
      END
```

APPENDIX I
LSI-11 PERIPHERAL DRIVER PROGRAMS

```
D
          .TITLE  CLOCK
          .GLOBL  CLOCK
          .MCALL  .REGDEF
          .REGDEF
          .ASECT
          .=100
INIVEC:   START
NPSW:     0
          .CSECT
CLOCK:    ADD     #2,R5
          TST     (R5)+
          BEQ     READ
          MOV     (R5)+,NHR
          MOV     (R5)+,NMIN
          MOV     (R5)+,NSEC
          MOV     (R5)+,NMSEC
          BR      RET
READ:     MOV     NHR,(R5)+
          MOV     NMIN,(R5)+
          MOV     NSEC,(R5)+
          MOV     NMSEC,(R5)+
RET:      RTS     PC
START:    INC     NMSEC
          CMP     #1000.,NMSEC
          BGT     DONE
          CLR     NMSEC
          INC     NSEC
          CMP     #60.,NSEC
          BGT     DONE
          CLR     NSEC
          INC     NMIN
          CMP     #60.,NMIN
          BGT     DONE
          CLR     NMIN
          INC     NHR
          CMP     #24.,NHR
          BGT     DONE
          CLR     NHR
DONE:     RTI
NHR:      .WORD   0
NMIN:     .WORD   0
NSEC:     .WORD   0
NMSEC:    .WORD   0
          .END
```

```
DSS
        .TITLE ADAC1
        .GLOBL ADC
        .CSECT
        R0=%0
        R1=%1
        R5=%5
        PC=%7
ADC:    MOV     (R5)+,R0
        MOV     #10,R1
        MOV     @(R5)+,R0
        ASH     R1,R0
        CLR     R1
        MOV     R0,@#176770
LOOP:   MOV     @#176770,R0
        INC     R1
        CMP     #20,R1
        BNE     LOOP
        MOV     @#176772,@(R5)
        BIC     #170000,@(R5)+
        MOV     R1,@(R5)+
        RTS     PC
        .END
```

```
00
         .TITLE ADAC2
         .GLOBL  DAC
         R0=%0
         R1=%1
         R5=%5
         PC=%7
DAC:     MOV     (R5)+,R0
         MOV     (R5)+,R0
         MOV     (R5)+,R1
         CMP     #1,R0
         BEQ     CH1
         MOV     R1,@#176762
         RTS     PC
CH1:     MOV     R1,@#176760
         RTS     PC
         .END
```

APPENDIX J

PLM SUPPORT PROGRAMS FOR THE CONTROL LOGIC MICPROCESSORS

```
04MAY77      15:51:00     PTCARD2W              SYSTEMS REAL-TIME MONITOR-5.0                        PAGE    1

      STRING M,[illegible]
      INTEGER B
      FILELIST [illegible]
      DATA N1,N2/"[illegible]OR "/
      DATA N3,N4,N5/"[illegible]",";0 PL";"N2 V"/
      DATA N6/0[illegible]000[illegible]0/
      DATA N7/0[illegible]056[illegible]5[illegible]/
      DATA N8,N9,N10,N11/07[illegible]7000000,0777000,0777,0777000000000/
      DIMENSION M(2000),J(20)
      DIMENSION K(4)
      PRINT 1
    1 FORMAT(" INPUT FILE NAME: ")
      READ 2,A
    2 FORMAT(V)
      IFLG=0
    3 DO 10 II=1,2000
      READ(1,2,END=999)M(II)
      ENCODE(J,2)M(II)
      IF(J(3).NE.N1)GO TO 7
      IF(J(4).NE.N2)GO TO 7
      IF(IFLG.NE.0)GO TO [illegible]
      KK=II-2
      IF(KK.LT.1)KK=KK+2000
      PRINT 2,M(KK)
      PRINT 2,M(II)
      IFLG=1
      GO TO 10
    9 PRINT 2,M(II)
      GO TO 10
    7 IFLG=0
      IF(J(1).NE.N3)GO TO 10
      IF(J(2).NE.N4)GO TO 10
      IF(J(3).EQ.N5)GO TO 11
   10 CONTINUE
      GO TO 3
   11 PRINT 2,M(II-2)
      PRINT 12
   12 FORMAT(" SYMBOL TABLE")
      READ 13,B
   13 FORMAT(A1)
      IF(B.NE.N6)GO TO 999
      PRINT 14
   14 FORMAT("-")
   20 READ(1,2,END=999)MM
      ENCODE(J,2)MM
      IF(J(8).NE.N7)GO TO 20
   15 K(1)=AND(J(9),N8)
      K(1)=SHIFTR(K(1),18)
      K(2)=AND(J(9),N9)
      K(2)=SHIFTR(K(2),9)
      K(3)=AND(J(9),N10)
      K(4)=AND(J(10),N11)
      K(4)=SHIFTR(K(4),27)
      DO 21 I=1,4
```

```
04MAY77     15:51:04     PICARD2@            SYSTEMS REAL-TIME MONITOR-5.0                   PAGE   2

640         IF(K(I).LT.48)GO TO 22
650         IF(K(I).GT.57)GO TO 22
660         K(I)=K(I)-48
670         GO TO 21
680      22 IF(K(I).LT.65)GO TO 999
690         IF(K(I).GT.70)GO TO 999
700         K(I)=K(I)-55
710      21 CONTINUE
720         L1=K(1)*16+K(2)
730         L2=K(3)*16+K(4)
740         PRINT 25,(J(I),I=1,8),L1,L2
750      25 FORMAT(8A4,O3,"/",O3)
760         READ(1,2,END=999)MM
770         ENCODE(J,2)MM
780         IF(J(8).NE.N7)GO TO 99
790         GO TO 15
800      99 PRINT 14
810     999 STOP;END

READY
```

```
DDP
      STRING M
      FILELIST A
      COMMON L(2-[illegible]),[illegible],NWC,N27,N18,N9
      DIMENSION J(13),K(52),ITEMP(42)
      DATA I1,I2,I3,I4/0777000000000,0777000000,0777000,0777/
      DATA I5,I6,I7/052052052052,017,0360/
      IFLG=0
      NWC=1
      IFLG=0
      DO 90 II=1,50
   95 J(I)=0
      N27=2**27
      N18=2**18
      N9=2**9
      PRINT 50
   50 FORMAT(" INPUT FILE NAME: ")
      READ 2,A
    1 READ(1,2,END=999)M
    2 FORMAT(V)
      DECODE(J,2)M
      DO 3 I=2,6
      IF(J(I).NE.I5)GO TO 1
    3 CONTINUE
    4 READ(1,2,END=999)M
      DECODE(J,2)M
      NT=J(1)
      K(1)=0
      K(2)=AND(NT,I3)
      K(2)=K(2)/N9
      K(3)=AND(NT,I4)
      DO 5 I=4,51,4
      NT=J(I/4+1)
      K(I)=AND(NT,I1)
      K(I)=K(I)/N27
      K(I+1)=AND(NT,I2)
      K(I+1)=K(I+1)/N18
      K(I+2)=AND(NT,I3)
      K(I+2)=K(I+2)/N9
      K(I+3)=AND(NT,I4)
    5 CONTINUE
      DO 8 I=1,52
      IF(K(I).EQ.0)GO TO 8
      IF(K(I).LT.48)GO TO 7
      IF(K(I).GT.57)GO TO 7
      K(I)=K(I)-48
      GO TO 8
    7 IF(K(I).LT.65)GO TO 999
      IF(K(I).GT.70)GO TO 999
      K(I)=K(I)-55
    8 CONTINUE
      DO 9 I=1,42
    9 ITEMP(I)=0
      ITEMP(1)=15
      ITEMP(2)=15
      ITEMP(4)=K(7)
      ITEMP(3)=K(6)
      ITEMP(6)=K(5)
      ITEMP(5)=K(4)
      ITEMP(8)=K(3)
      ITEMP(7)=K(2)
      N=K(2)*16+K(3)
      IF(N.NE.16)IFLG=1
      ISUM=0
      NN=N*2+7
      DO 10 I=9,NN,2
      ITEMP(I)=K(I+1)
      ITEMP(I+1)=K(I+2)
      ISUM=ISUM+ITEMP(I)*16+ITEMP(I+1)
   10 CONTINUE
      ITEMP(NN+3)=AND(ISUM,I6)
      ITEMP(NN+2)=AND(ISUM,I7)
      ITEMP(NN+2)=ITEMP(NN+2)/16
      NN=NN+2
      DO 11 I=1,NN,2
   11 CALL STACK(ITEMP(I),ITEMP(I+1),NFLG)
      IF(IFLG.EQ.0)GO TO 4
      CALL STACK(11,15,1)
      II1=II+1
      II2=II+51
      DO 12 I=II1,II2
   12 L(I)=0
      WRITE("BINFIL")L
  999 STOP;END
      SUBROUTINE STACK(N1,N2,NFLG)
      COMMON L(2500),II,NWC,N27,N18,N9
      M=N1*16+N2
      IF(NWC.NE.1)GO TO 1
      M=SHIFTL(M,27)
      NNN=OR(NNN,M)
      GO TO 4
    1 IF(NWC.NE.2)GO TO 2
      M=SHIFTL(M,18)
      NNN=OR(NNN,M)
      GO TO 4
    2 IF(NWC.NE.3)GO TO 3
      M=SHIFTL(M,9)
      NNN=OR(NNN,M)
      GO TO 4
    3 NNN=OR(NNN,M)
    4 NWC=NWC+1
      IF(NWC.NE.5)GO TO 5
      L(II)=NNN
   10 FORMAT(1X,O12)
      NNN=0
      II=II+1
      NWC=1
    5 IF(NFLG.EQ.1)L(II)=NNN
      RETURN;END
```

REFERENCES

1. Crist, J., Drzal, L.T., and Wisnosky, D., "An Automated Liquid Adsorption System for Surface Energetic Measurements", Air Force Materials Laboratory Technical Report AFML-TR-76-   , Submitted.

2. Macha, D.E., Sharpe, Jr., W.N., Grandt, Jr., A.F., Submitted to ASTM for presentation and publication at the Ninth National Symposium on Fracture Mechanics, University of Pittsburgh, Pittsburgh, PA, 25-27 August 1975.

3. Tong, H., Bonnett, W.S., Copper, L., Moyer, C.B., Neuner, G.J., "Radiant Heating of Aerospace Materials", Air Force Materials Laboratory Technical Report AFML-TR-73-190.

4. Suchsland, K.E., and Maurer, C.E., "A Unified Ablative Material Thermal Response Analysis Procedure", Air Force Materials Laboratory Technical Report AFML-TR-76-46.

5. Rondeau, R.E., and Ford, D.L., "Use of the APPLE Code in Analyzing the Temperature Response of Aluminum to Nuclear Thermal Radiation", Air Force Materials Laboratory Technical Memo AFML-TM-LP-76-3.

6. Rondeau, R.E., and Ford, D.L., "Coating Analysis with the APPLE Code, Part II: Flight Simulation and Thermal Conductivity Effects", Air Force Materials Laboratory Technical Memo AFML-TM-LP-76-6.

7. Rondeau, R.E., and Ford, D.L., "Thermal and Stress Analyses of the AF Weapons Laboratory Zinc Selenide Laser Window", Air Force Materials Laboratory Technical Memo AFML-TM-LP-76-4.

8. Rondeau, R.E. and Ford, D.L., "Computed Temperature Response of the AFML Flat-Top Laser Window", Air Force Materials Laboratory Technical Memo AFML-TM-LP-76-12.

9. Rondeau, R.E., and Ford, D.L., "Analytical Comparisons of KCl and ZnSe for Laser Window Applications", Air Force Materials Laboratory Technical Memo AFML-TM-LP-76-14.

10. Rondeau, R.E., and Ford, D.L., "Development of AFML Prediction Capability for Laser Effects", AFML-TR-76-243, Submitted.

11. Carlson, R.L., "A 15 Kilowatt CW CO_2 Coaxial Electric Discharge Laser", Air Force Weapons Laboratory Technical Report AFWL-TR-76-115.

12. Kessler, W.C., private communication.

13. Dekaney, A., and Guthrie, L.E., "Determination of the Feasibility of Applying Computer Analysis of Titanium Alloy Microstructures to Quality Control. Air Force Materials Laboratory Technical Report AFML-TR-73-123.

14. Feng, F., Forsen, G., Gillotte, M., "Image Processing Experiments Titanium Alloy Photomicrographs", Pattern Analysis and Recognition Corporation Report No. 76-10.

15. Leet, D.G., "OZ: A Data Acquisition and Analysis System for the Air Force Laser Hardened Materials Evaluation Laboratory Facility", UDRI-TR-77-21, University of Dayton Research Institute, Dayton, OH, (1977).