UNITED STATES DEPARTMENT OF THE INTERIOR
WILLIAM P. CLARK, Secretary

GEOLOGICAL SURVEY
Dallas L. Peck, Director

UNITED STATES GOVERNMENT PRINTING OFFICE: 1984

Free on application to:

Branch of Distribution
U. S. Geological Survey
604 S. Pickett Street
Alexandria, VA 22304

U.S. GEOLOGICAL SURVEY CIRCULAR 963

APSAS—An Automated Particle Size Analysis System

APSAS—An Automated Particle Size Analysis System

By L. J. Poppe, A. H. Eliason, and J. J. Fredericks

U.S. GEOLOGICAL SURVEY CIRCULAR 963

A computer-based system designed to rapidly and accurately perform sediment grain-size analyses and calculate statistics

DEPARTMENT OF THE INTERIOR
DONALD PAUL HODEL, Secretary

U.S. GEOLOGICAL SURVEY
Dallas L. Peck, Director

Library of Congress Cataloging in Publication Data

Poppe, Lawrence J.
APSAS, an automated particle size analysis system.
(Geological Survey circular ; 963)
Bibliography: p. 8
Supt. of Docs. no.: I. 19.4/2:963
1. Particle size determination—Data processing. I. Eliason, A. H. II. Fredericks, J. J. III. Title. IV. Series.
QE75.C5 no. 963 557.3s[620'.43'0287] 84-600396 [TA418.8]

Any trade names used in this publication are used for descriptive purposes only and do not constitute endorsement by the U.S. Geological Survey.

Free on application to Distribution Branch, Text Products Section,
U.S. Geological Survey, 604 South Pickett Street, Alexandria, VA 22304

CONTENTS

ILLUSTRATIONS

APSAS — An Automated Particle Size Analysis System

By L. J. Poppe, A. H. Eliason, *and* J. J. Fredericks

Abstract

The Automated Particle Size Analysis System integrates a settling tube and an electroresistance multichannel particle-size analyzer (Coulter Counter) with a Pro-Comp/8-8 microcomputer and a Hewlett Packard 2100 MX (HP 2100 MX) minicomputer. This system and its associated software digitize the raw sediment grain-size data, combine the coarse- and fine-fraction data into complete grain-size distributions, perform method of moments and inclusive graphics statistics, verbally classify the sediment, generate histogram and cumulative frequency plots, and transfer the results into a data-retrieval system. This system saves time and labor and affords greater reliability, resolution, and reproducibility than conventional methods do.

INTRODUCTION

The grain-size distribution of a detrital sediment is of considerable importance to sedimentologists and engineers because distribution is related to the dynamic conditions of transportation and deposition. The size distribution can also reveal valuable information about the permeability, stability, or diagenetic history of the sediments. Inasmuch as many geological observations consist of measurements made on a large number of specimens, the techniques and equipment used for particle-size analysis must be fast and yield highly reproducible results.

The Automated Particle Size Analysis System (APSAS), which we describe in this paper, digitizes the Rapid Sediment Analyzer (RSA) and Electroresistance Multichannel Particle-Size Analyzer (EMPSA) data and stores them to disk. It integrates the coarse and fine-fraction data into a complete size distribution, performs method of moments and inclusive graphics statistics, and texturally and statistically classifies the sediment with verbal equivalents. In addition, the cumulative frequency distribution, method of moments statistics, and sample identifiers are stored in a data-retrieval system that can be accessed by a large number and variety of users.

For many years, the sand and gravel fractions were determined by sieve analyses, and the silt and clay fractions were determined by pipette or hydrometer analyses. Later, the Woods Hole Rapid Sediment Analyzer (Ziegler and others, 1960; and Schlee, 1966) and electroresistance multichannel particle-size analyzers such as the Coulter Counter have removed much of the tedium from grain-size analyses. About this same time Formula Translation (FORTRAN) programs for the calculation of statistical parameters on geologic data also were developed (Kane and Hubert, 1962; Schlee and Webster, 1967). Other particle-size analysis programs have since been written in Algorithmic Language (ALGOL) (Jones and Simpkin, 1970), Beginners All-Purpose Symbolic Instruction Code (BASIC) (Sawyer, 1977), and even for use with hand-held calculators (Benson, 1981). Early attempts to integrate computers with particle-size analysis equipment began with the settling tube (Ziegler and others, 1964; Rigler and others, 1981), and hardware and software packages are now available for EMPSA units (Muerdter and others, 1981). However, each of these previously existing systems analyzes only portions of a typical grain-size distribution. The recent development and commercial availability of inexpensive microcomputers now allow sedimentologists to construct complete, computerized particle-size analysis systems.

INSTRUMENTATION

The cornerstone around which our system was developed is the Pro-Comp/8-8 (Pro-Comp Systems Inc., 1982) IEE-696 S-100 bus-based microcomputer (fig. 1). This system (Jennings and others, 1984) incorporates a 4 Mhz Z-80A master processor with 64k bytes of random access memory (RAM), a NEC uPD765A chip Direct Memory Access floppy disk controller that supports up to 1.2 megabytes of disk storage (for fast memory-to-disk storage without central processing unit (CPU) intervention), two RS-232 serial ports, and two parallel ports (fig. 2). This system also contains two slave processor cards, each of which has two additional serial ports and 64k bytes of RAM. The Pro-Comp/8-8 microcomputer also supports TurboDos version 1.22 (Software 2000 Inc., 1982): a multiuser, multiprocessing operating system. Under this operating system, each slave processor shares a common pool of mass storage and printers or other peripherals while independently accessing its own input/output ports and memory. This system allows simultaneous operation of, and data storage to, the computer from both the RSA and EMPSA units. Because this system utilizes a conventional, self-contained microcomputer, the system can be used for word processing and other computational tasks, in addition to particle-size analyses.

Although the Pro-Comp/8-8 provides support for a software clock, a battery-powered Scitronics TRC-100 real time clock and the appropriate patch software were installed. This clock allows the processors to access correct date and time information for assignment to each set of raw data even if the system has been shut down since its last use. Exact intrasample timing for the RSA data is established by generating interrupts that use an available precise time base on the assigned slave processor board; these interrupts, in turn, cause incrementation of the counter register that is accessible to the RSA software.

One serial port (A) on each slave processor board interconnects the computer with the RSA and EMPSA Kimtron ABM85 CRT data terminals. On the slave assigned to the EMPSA, the second serial port (B) is connected to an Epson RX-80 printer that is used to generate hard copies of the analyses data. The EMPSA data are transmitted directly to the EMPSA data terminal, which eliminates the need for an additional interface port and cable and simplifies the software for the data entry. The second serial port on the RSA slave is assigned to a Small System Design Inc. model ADM12S 12-bit analog-to-digital converter. This dedication is necessary because the intrasample timing is critical and must be controlled directly by the slave. The RSA sample identifiers and processed output from the RSAT program are sent to the common pool printer, which is interfaced to the masterprocessor serial port B. This is possible because timing is not critical during this portion of the program execution and the printers contain data buffers that allow simultaneous printer and computational operation. The graphics capability of the printer also permits the production of cumulative frequency plots showing the change in pressure with time. These plots allow the technician to monitor system noise, transducer operation, and sample reproducibility. The disk port on the master card controls dual Shugart 800/801 disk drives that operate with 8-inch, single-sided, double-density floppy disks.

A Hayes Smartmodem 1200, which is connected to master processor port A, allows the Pro-Comp/8-8 to communicate via telephone lines with other computers or timesharing devices. This modem operates online in full or half-duplex at a rate of 1200 or 300 bauds per second.

The fine-fraction grain-size distribution, which is composed of silt- and clay-sized material, is determined using a Coulter Counter Model TAII EMPSA equipped with a Coulter Counter Model PCA2 Population Count Accessory (PCA) and a PCA Computer Interface (fig. 3). The Coulter Counter determines the number and sizes of particulate sediment in an electrolyte solution by drawing the suspension through small apertures (Coulter Electronics Inc., 1982). Electrodes are immersed in the electrolyte on opposite sides of the aperture, and the electrical current is monitored as the particle suspension and current pass through the aperture. As a particle travels through the aperture, the particle changes the resistance between the electrodes. This produces a current pulse of short duration whose magnitude is proportional to the particle volume. The series of pulses is electronically classified by size and is counted. The PCA adds the capability of differentially counting particles in each of the 16 size-fraction channels established by the main unit. The computer interface transmits the data from the PCA to the microcomputer via the EMPSA data terminal.

The RSA provides a means for rapid size analysis of sand-sized material by settling the

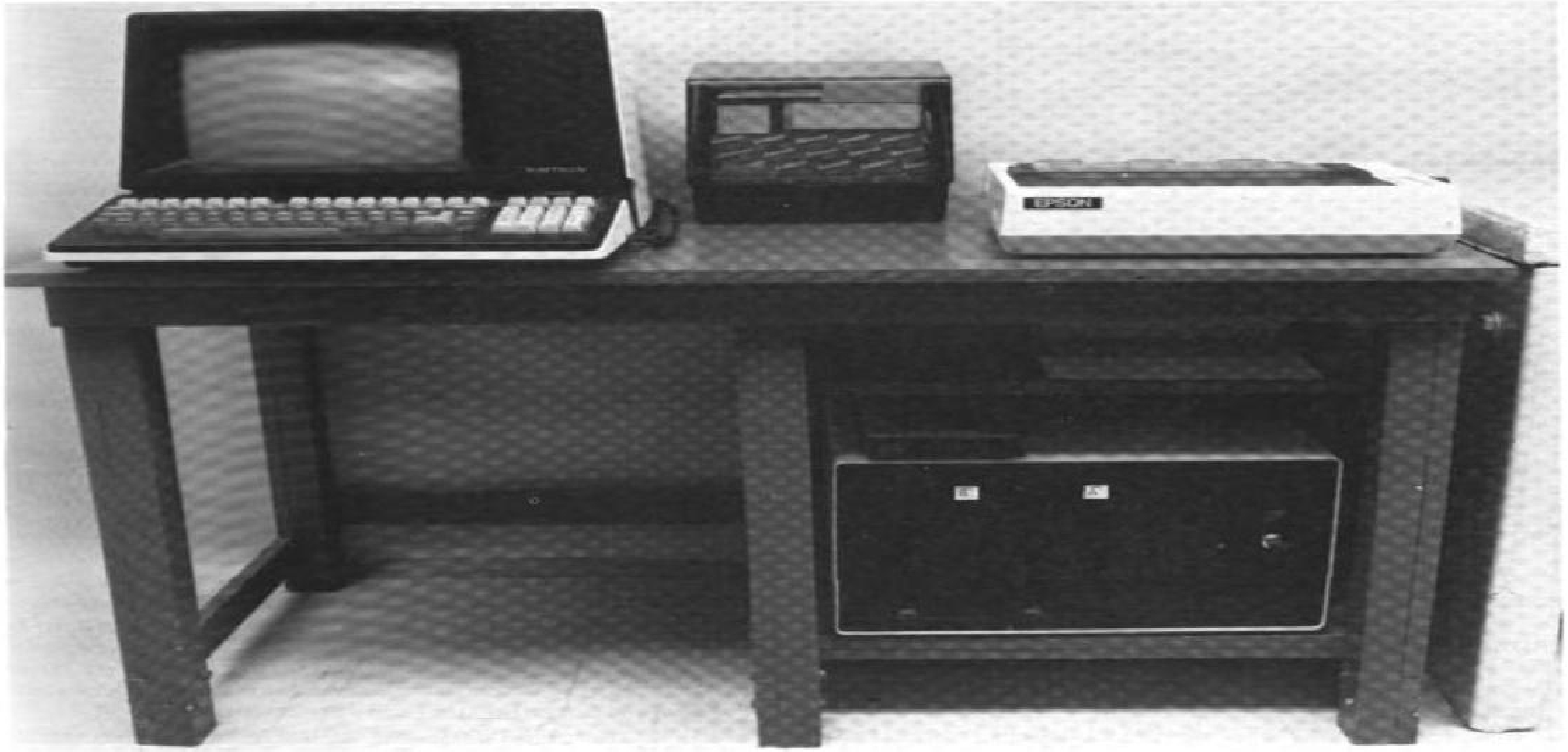


FIGURE 1.—Pro-Comp/8-8 microcomputer and associated computer hardware.

grains through a column of water (fig. 4). An ultra-low range pressure transducer measures the pressure differential between two columns of water having a common head. The change caused by the introduction of sediment within one of the columns is measured by the transducer and amplified by a signal conditioner. This amplifier/signal conditioner provides the excitation signal to the RSA pressure transducer and converts its output to a voltage analog. This voltage is fed to the Small System Design Inc. analog-to-digital converter, converted to digital output, and relayed to the microcomputer at a sampling interval of about once every 50 milliseconds. As the sediment settles past the opening to the pressure transducer, the pressure differential decreases with time. Because the sedimentation rate, in accordance with Stoke's Law, is a function of grain size, one can interpret the sand-frction grain-size distribution from the variation in pressure differential.

The coarse (sand and gravel) and fine (silt and clay) fractions are separated by wet sieving the sample through a 63-µm, number 230 U.S. Standard sieve. The gravel-fraction (>2.0 mm) distribution must also be determined by sieving. The relative percentages of the material in the gravel-fraction phi classes are keyed in along with the sample weight, coarse weight, sand weight, and sample identifiers during the RSA analyses. The computer determines the weight of the fine-fraction material by subtracting the coarse weight from the sample weight. Because the raw data from the RSA and EMPSA analyses are converted to relative

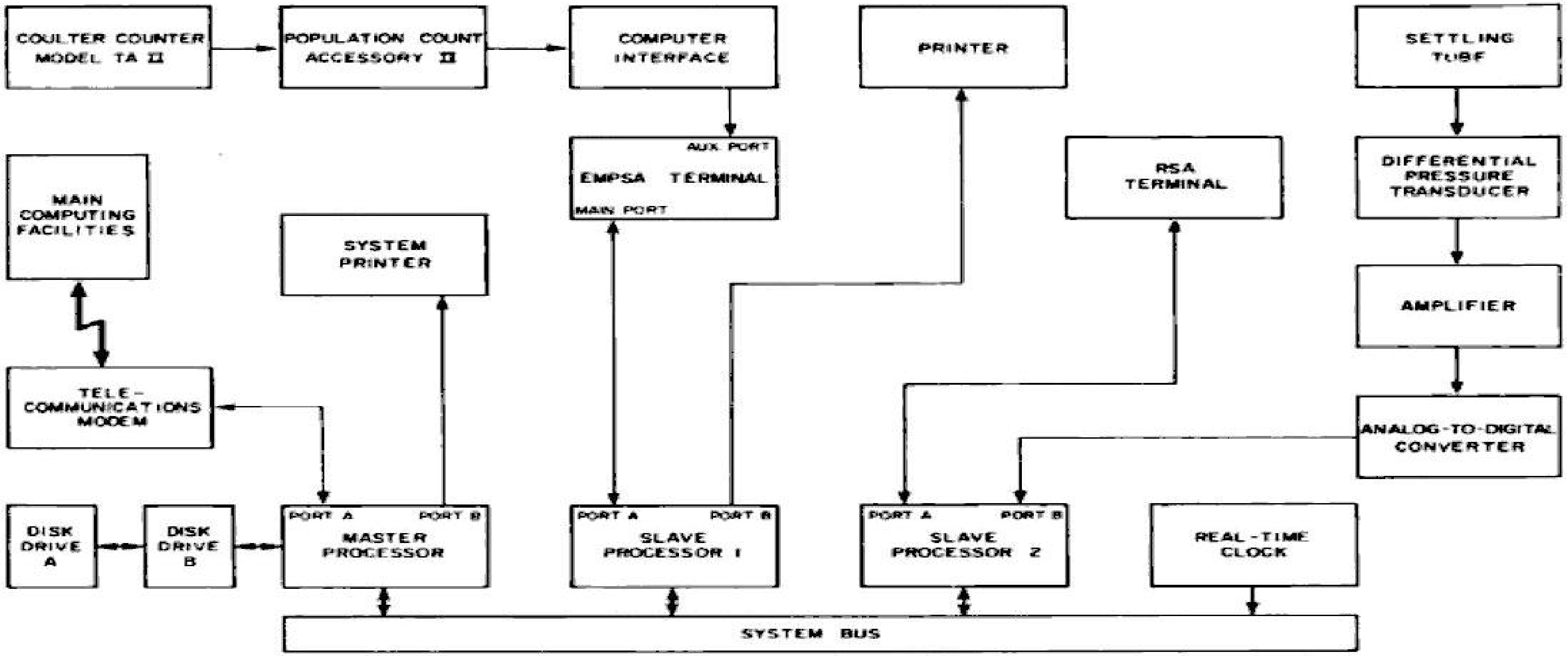


FIGURE 2.—Automated Particle Size Analysis System.

percentages, these weights can be used to calculate complete grain-size distributions.

Further processing of the data is performed on the larger, multiuser, office computer system. This system consists of three HP 2100 MX minicomputers with access to 284 megabytes of core memory and disk storage, six tape drives, and a variety of graphic-display devices. Although the software could have been designed to operate entirely on the Pro-Comp/8-8 microcomputer, the HP 2100 MX offers larger mass storage, a faster response time, and centralized access to the sediment data base.

SOFTWARE

Figure 5 outlines the software sequence of operation in a generalized flow diagram for those programs used on both the Pro-Comp/8-8 and HP 2100 MX computers. Raw data records are created on the microcomputer by using the programs RSAT and CLTRT (Appendixes A and B). The user is prompted for all the necessary identifiers, and, upon completion of a satisfactory analysis, the data and identifiers, are written to both an output file and a printer. Three raw data records are generated for each sample: an RSA record, a 200-µm aperture EMPSA record, and a 30-µm aperture EMPSA record. These records each contain a lab number, equipment type, sample identification, project identification, requestor, operator, and analysis date. The RSA data records include sample weight, coarse weight, sand weight, and the relative percentages of the -5 to -1 phi gravel and 0 to 4 phi sand fractions. The EMPSA data records include aperture diameter and tabulated micrometer diameters and the corresponding relative-frequency percentages from the 16 size-fraction channels collected by the Coulter Counter.

The raw RSA and EMPSA data are archived into master files and written to transfer disks by using the program SEDIT (Appendixes A and B). This program also contains utilities to inspect, edit, and print hard copies of the data

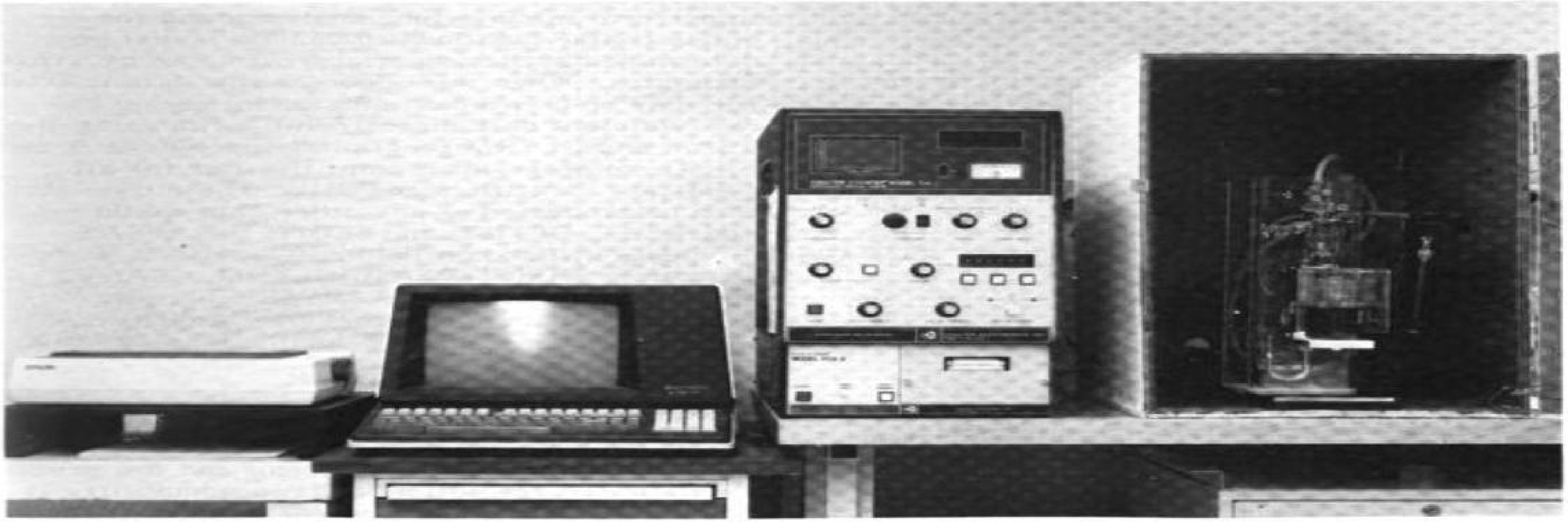


FIGURE 3.—Coulter Counter portion of APSAS.

generated by the RSAT and CLTRT programs. The master files are subsequently transferred to the main office computer system via modem and telephone lines by a commercially available telecommunications software package.

The field and navigation parameters are then entered by using the program GSANV (Appendixes A and B). These parameters and their coding are described in the Request For Analysis forms which are completed by all personnel who submit samples for particle-size analysis. The field and navigation parameters include latitude, longitude, area, sampling device, water depth, depth in section to the top of the sample, and depth in section to the bottom of the sample. The program JSORT (Appendixes A and B) is used to sort the multiline records according to lab numbers and to output the records that are complete (all EMPSA data, RSA data, and field and navigation parameters) to an output file for further processing.

The program GSTAT (Appendixes A and B) is used to retrieve data from the complete, sorted, raw data files and to compute the modified frequency percents, method of moments statistics, textural classification (Shepard, 1954), and percentages of gravel, sand, silt, and clay. The modified frequency percentages are given for size distributions up to and including 11 phi to -5 phi (< 0.0005 mm X < 640 mm). The method of moments statistics generated by GSTAT include modal classes; modal frequencies; arithmetic mean, median, and standard deviation; skewness; and kurtosis. The user may also request a histogram, a cumulative frequency plot, and inclusive graphics statistics (Folk, 1974) and verbal equivalents for standard deviation, skewness, and kurtosis. The frequency percentages for the corresponding phi classes are computed by the subroutines MPVC, SUMRY, and WTFP. The cumulative frequency-percent curve used in computing the inclusive graphics statistics is approximated by using the International Mathematical and Statistical Library Inc. (IMSL) routines IQHSCU and ICSEVU.

To correct the errors introduced by the assumption that each value within a phi class is centered at the midpoint of that class, Shepard's Correction (Kenny and Keeping, 1954) has been applied to the second and fourth moments about the mean.

When a pre-GRASP (Geologic Retrieval And Synopsis Program) database file is specified in GSTAT, the cumulative frequency percents, phi classes, method of moments statistics, and the appropriate identifiers will be written to a file in a free-field format for the Geologic Retrieval And Synopsis Program (GRASP). GRASP (Bowen and Botbol, 1975) was specifically designed and written to accommodate

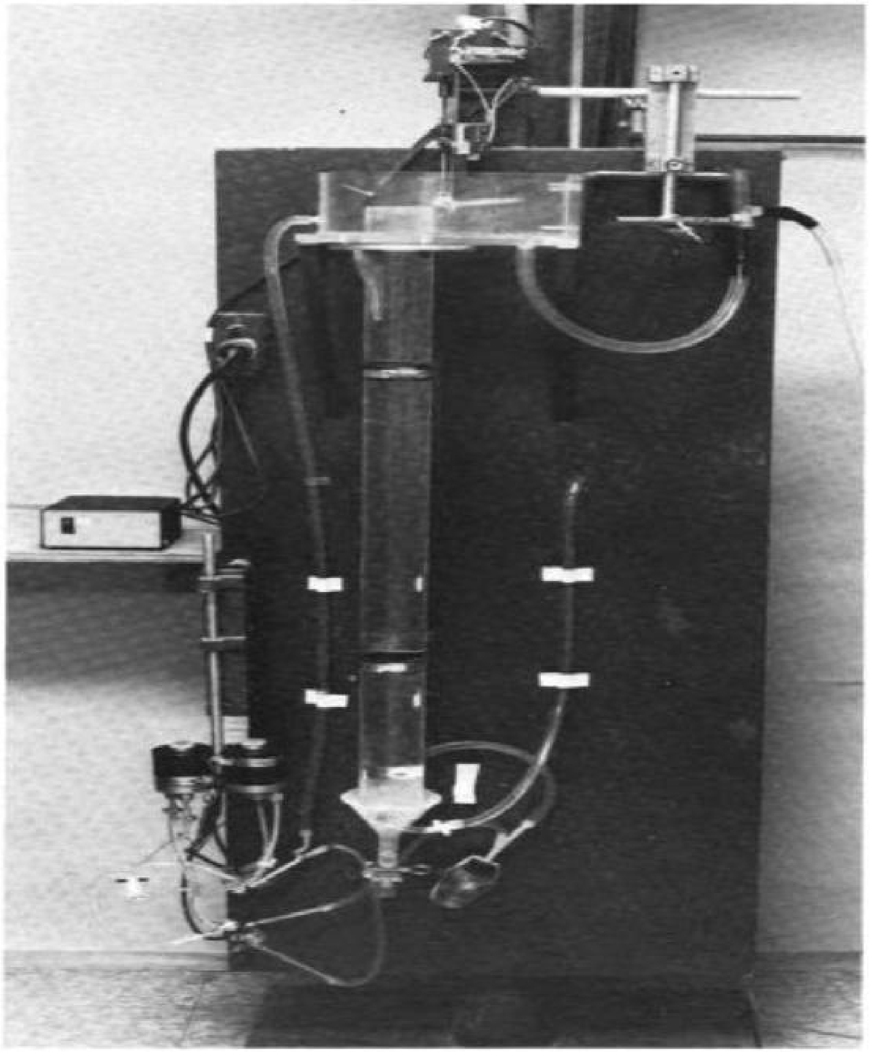


FIGURE 4.—Rapid Sediment Analyzer.

interactive access to earth-science databases and to be portable and user friendly. Once the data are in the GRASP database, multivariate analyses such as factor analysis or cluster analysis may be applied to retrieved data to help interpret complicated sedimentological phenomena.

Occasionally, samples which were not analyzed or only partially analyzed under APSAS must be included in the particle-size database or statistics must be generated for these samples. For example, many clay-rich samples do not contain enough sand (< 5 grams) to perform settling-tube analyses, and the entire coarse fraction (>0.062 mm) must be determined by sieving or approximated by visual estimates. When this occurs, a raw data master file may be created or updated by keying the raw sediment data directly into a file using the RSAM and CLTRM programs (Appendixes A and B). These programs prompt the user for all necessary identifiers and data. These data can then be processed as any other data would be.

RSAT, CLTRT, and SEDIT, the APSAS programs utilized on the Pro-Comp/8-8, were written in BASIC. RSAM, CLTRM, GSANV, JSORT, and GSTAT, the APSAS programs utilized on main office computer facilities, were written in RATFOR for the HP 2100 MX.

DISCUSSION

The major advantages derived from the system we have developed are the time- and labor-saving function it performs and the greater reliability, resolution, and reproducibility it affords. Manual calculation of a single grain-size distribution from raw RSA and EMPSA data requires about 1 hour of work for the technician and could involve mathematical errors. Correction of these errors then requires further work, and the analyses are still incomplete without the generation of useful statistics or a means of storing and retrieving the data.

Because the APSAS system processes samples in batches, an operator can assign the raw RSA and EMPSA data to master files, transfer the data to the main computer facilities via telephone lines, enter the sample parameters, sort the data, calculate the statistics, convert the data into a database, and generate a hardcopy in about 5 minutes per sample. The operation of this system is reasonably simple; because the programs are interactive and contain extensive internal error checking routines, all the programs can be operated by most laboratory technicians after minimal training.

Future modifications planned for APSAS include software changes to (1) accept pipette analyses as the fine-fraction input data and (2) output the particle-size distributions and statistics in 1/2- and 1/4-phi intervals.

Further hardware automation, such as automatic sample changing or introduction, probably is not desirable because human judgment is necessary to determine whether the analysis was run properly. Partially plugged aperture tubes on the EMPSA or density currents in the settling tube of the RSA are exam-

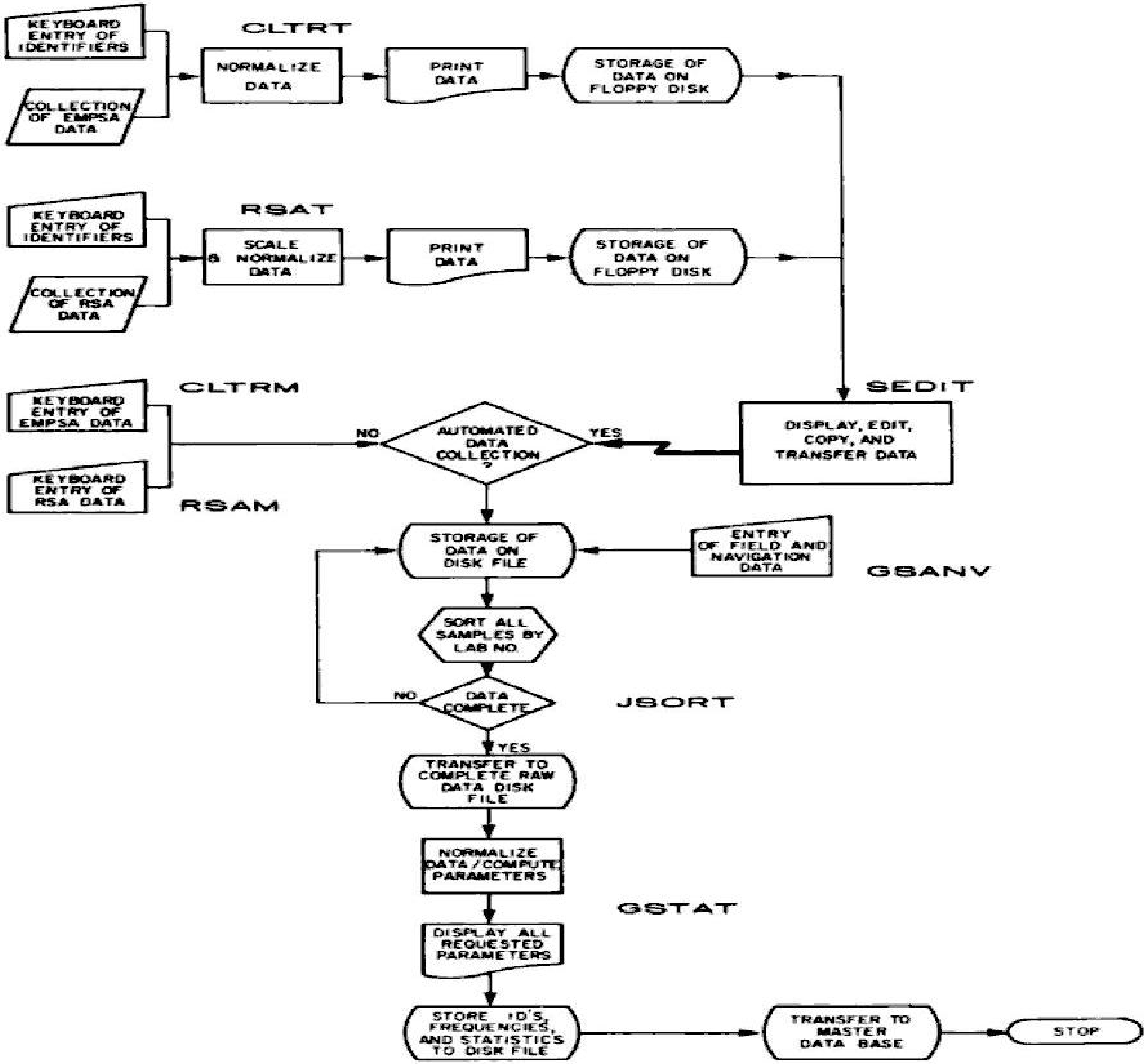

CLTRT
KEYBOARD ENTRY OF IDENTIFIERS
COLLECTION OF EMPSA DATA
NORMALIZE DATA
PRINT DATA
STORAGE OF DATA ON FLOPPY DISK
RSAT
KEYBOARD ENTRY OF IDENTIFIERS
COLLECTION OF RSA DATA
SCALE & NORMALIZE DATA
PRINT DATA
STORAGE OF DATA ON FLOPPY DISK
SEDIT
DISPLAY, EDIT, COPY, AND TRANSFER DATA
CLTRM
KEYBOARD ENTRY OF EMPSA DATA
KEYBOARD ENTRY OF RSA DATA
RSAM
NO
AUTOMATED DATA COLLECTION ?
YES
STORAGE OF DATA ON DISK FILE
ENTRY OF FIELD AND NAVIGATION DATA
GSANV
SORT ALL SAMPLES BY LAB NO.
NO
DATA COMPLETE
JSORT
YES
TRANSFER TO COMPLETE RAW DATA DISK FILE
NORMALIZE DATA / COMPUTE PARAMETERS
GSTAT
DISPLAY ALL REQUESTED PARAMETERS
STORE ID'S, FREQUENCIES, AND STATISTICS TO DISK FILE
TRANSFER TO MASTER DATA BASE
STOP

FIGURE 5.—APSAS programs used on the Pro-Comp/8-8 and Hewlett Packard 2100 MX Computers.

ples of problems that would adversely affect the accuracy of the data. Because a computer cannot detect the resultant errors, an operator must be present to visually monitor each analysis.

Our ability to develop this useful microcomputer-based system with readily available components and relatively easy to produce software should encourage other sedimentologists and programmers in the geosciences. Copies of the RATFOR and BASIC program software, documentation, and computational methods used in this system are included in the attached appendixes. If the BASIC and RATFOR programming supplied here are used, the cost of this system, exclusive of the Coulter Counter and settling tube related hardware, is about $11,000.

ACKNOWLEDGMENTS

The sorting program used within JSORT was written by G.I. Evenden. System configuration and BASIC programming were provided under contract by Eliason Data Services, P.O. Box 3, Mashpee, Mass. Part of the financial support for the BASIC programming and for computer hardware was provided by the U.S. Minerals Management Service under Interagency Agreement AA851-1A2-18.

REFERENCES CITED

Benson, D.J., 1981, Textural analyses with Texas Instruments 59 programmable calculator: Journal Sedimentary Petrology, v. 51, no. 2, p. 641-642.

Bowen, R.W., and Botbol, J.M., 1975, The Geologic Retrieval and Synopsis Program (GRASP): Geological Survey Professional Paper 966, 87 p.

Coulter Electronics Inc., 1982, Coulter Counter Model TA II product reference manual: Coulter Counter Technical Communications Div., Hialeah, Fla.

Folk, R.L., 1974, Petrology of sedimentary rocks: Hemphill Publishing Co., Austin, Tex., 182 p.

IMSL Inc., 1982, IMSL library reference manual, 9th, IMSL Inc., Houston, Tex.

Jennings, F.M., Botbol, J.M., and Evenden, G.I., 1985, A hybrid microcomputer system for geological investigations: U.S. Geological Survey Open-File Report 85-11, 11 p.

Jones, S.B., and Simpkin, P., 1970, A computer program for the calculation of hydrometer size analyses: Marine Geology, v. 9, p. M23-M29.

Kane, W.T., and Hubert, J.F., 1962, FORTRAN program for the calculation of grain-size textural parameters on the IBM 1620 computer: Sedimentology, no. 2, p. 87-90.

Kenny, J.F., and Keeping, E.S., 1954, Mathematics of statistics, Part 1: Princeton, N.J., D. Van Nostrand Company, Inc., 348 p.

Kernighan, B.W., and Plauger, P.J., 1976, Software tools: Addison-Wesley Publishing Co., Reading, Mass.

Muerdter, D.R., Dauphin, J.P., and Steele, G., 1981, An interactive computerized system for grain-size analyses of silt using electroresistance: Journal Sedimentary Petrology, v. 51, no. 2, p. 647-650.

Pro-Comp Systems Inc., 1982, Pro-Comp/8-8 users manual: Pro-Comp Systems Inc., New York.

Rigler, J.K., Collins, M.B., and Williams, S.J., 1981, A high precision, digital-recording sedimentation tower for sands: Journal Sedimentary Petrology, v. 51, no. 2, p. 642-644.

Sawyer, M.B., 1977, Computer program for the calculation of grain-size statistics by the method of moments: U.S. Geological Survey Open-File Report 77-580, 15 p.

Schlee, John, 1966, A modified Woods Hole Rapid Sediment Analyzer: Journal Sedimentary Petrology, v. 36, no. 1, p. 403-413.

Schlee, John, and Webster, Jacqueline, 1967, A computer program for grain-size data: Sedimentology, v. 8, no. 1, p. 45-54.

Shepard, F.P., 1954, Nomenclature based on sand-silt-clay ratios: Journal Sedimentary Petrology, v. 24, p. 151-158.

Software 2000 Inc., 1982, TurboDos 1.2: Advanced Digital Corp., Huntington Beach, Calif.

Zeigler, J.M., Whitney, G.G., Jr., and Hayes, C.R., 1960, Woods Hole, Rapid Sediment Analyzer: Journal Sedimentary Petrology, v. 30, p. 490-495.

Zeigler, J.M., Hayes, C.R., and Webb, D.C., 1964, Direct readout of sediment analyses by settling tube for computer processing: Science, v. 145, no. 3627, p. 51.

APPENDIXES

APPENDIX A

Program Documentation

1. RSAT

Program Name: RSAT

Type: Main program

Purpose: To create raw RSA data records

Machine: Pro-Comp/8-8

Operating System: TurboDos version 1.22

Source Language: BASIC

Program Category: Data Processing

Input: At the beginning of each run the operator will be prompted for the plotter scale, operator, requestor, cruise id, project id, sample id, lab number, sample weight, coarse weight, sand weight, and relative percents of the phi classes from the gravel fraction.

Output: A raw data record stored on floppy disk and a hard copy (fig. A-1)

Usage: To execute program:

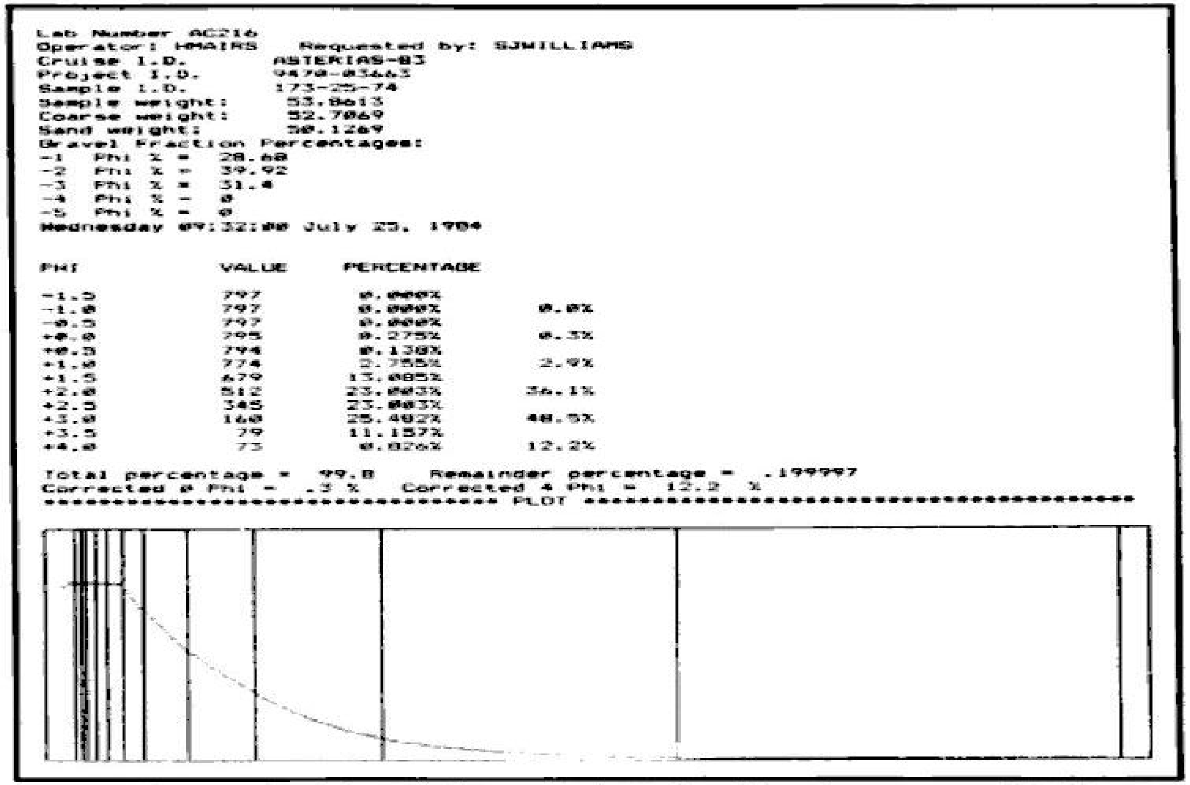
[RU] RSAT

User is prompted for all necessary identifiers and data.

For each run enter the plotter scale (program defaults to full scale).

For each analysis enter:

a) Operator
b) Requestor
c) Cruise id
d) Project id
e) Sample id
f) Lab number
g) Sample weight



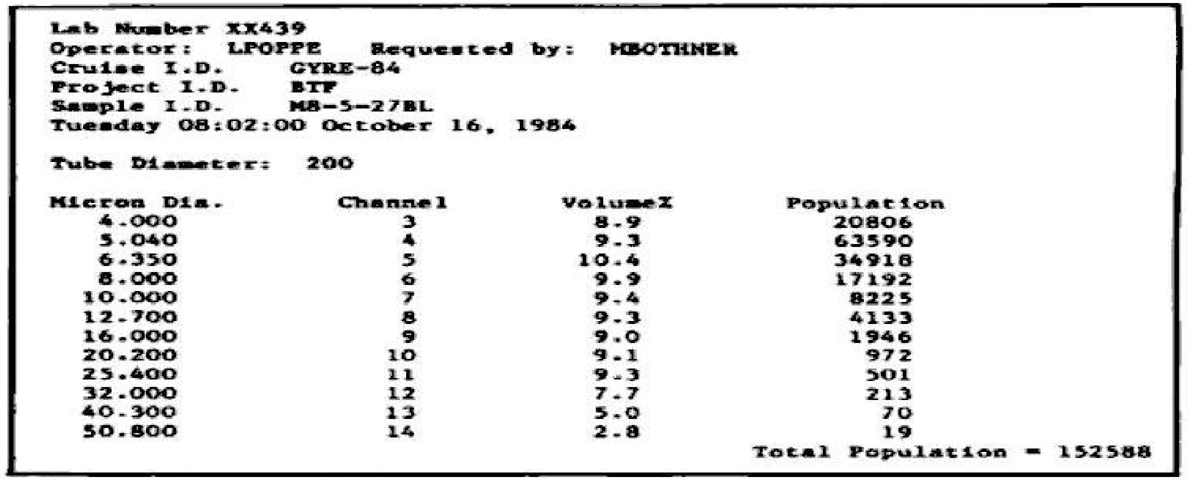
```
Lab Number AC216
Operator: HMAIRS         Requested by: SJWILLIAMS
Cruise I.D.              ASTERIAS-83
Project I.D.             9470-03663
Sample I.D.              173-25-74
Sample weight:            53.8613
Coarse weight:            52.7069
Sand weight:              50.1269
Gravel Fraction Percentages:
-1   Phi  % =   28.68
-2   Phi  % =   39.92
-3   Phi  % =   31.4
-4   Phi  % -   0
-5   Phi  % =   0
Wednesday 09:32:00 July 25, 1984

PHI              VALUE      PERCENTAGE

-1.5              797         0.000%
-1.0              797         0.000%          0.0%
-0.5              797         0.000%
+0.0              795         0.275%          0.3%
+0.5              794         0.138%
+1.0              774         2.755%          2.9%
+1.5              679        13.085%
+2.0              512        23.003%         36.1%
+2.5              345        23.003%
+3.0              160        25.482%         48.5%
+3.5               79        11.157%
+4.0               73         0.826%         12.2%

Total percentage =   99.8     Remainder percentage =   .199997
Corrected 0 Phi -   .3 %    Corrected 4 Phi =   12.2   %
*********************************** PLOT *********************************************
```

FIGURE A1.—A raw data record produced by the program RSAT.

h) Coarse weight
i) Sand weight
j) Relative percents of the phi classes from the gravel fraction

The sample is introduced into the settling tube when the "Waiting For Sample Introduction" prompt is given.
A copy of the identifiers, the data, and a cumulative frequency plot of the pressure change versus time are generated on the printer.
The operator may then save (S), reprint (R), or, if the data is no good, purge the data (E). A default saves the data. Upon responding to any of the three choices, the program is cycled back to (a) above.
A default on any of the prompted identifiers other than sample id or lab number will enter the identifier from the previous analysis.
Restrictions: Operator, requestor, cruise id, and project id may be up to 19 characters and have no imbedded commas or spaces. Speed requirements necessitate that the BASIC source language be compiled into machine language.
Subprograms Required: The SSINT and RTCSET routines must be included when TurboDos is generated to handle sample timing interrupts and to set the software clocks, respectively. BASTOD is called by the main program to access time and data information.
Storage Requirements: 11.5k bytes

2. CLTRT
Program Name: CLTRT
Type: Main program
Purpose: To create raw EMPSA data records
Machine: Pro-Comp/8-8
Operating System: TurboDos version 1.22
Source Language: BASIC
Program Category: Data Processing
Input: At the beginning of each run the user will be prompted for the operator, requestor, cruise id, project id, sample id, lab number, tube aperture diameter, initial micrometer diameter, channel number associated with that diameter, and Active Channels switch setting.
Output: A raw data record stored on floppy disk and a hard copy (fig. A-2)
Usage: To execute program:
[RU] CLTRT
User is prompted for all necessary identifiers.



```
Lab Number XX439
Operator:   LPOPPE     Requested by:   MBOTHNER
Cruise I.D.      GYRE-84
Project I.D.     BTF
Sample I.D.      M8-5-27BL
Tuesday 08:02:00 October 16, 1984

Tube Diameter:   200

Micron Dia.          Channel          Volume%          Population
   4.000                3                8.9              20806
   5.040                4                9.3              63590
   6.350                5               10.4              34918
   8.000                6                9.9              17192
  10.000                7                9.4               8225
  12.700                8                9.3               4133
  16.000                9                9.0               1946
  20.200               10                9.1                972
  25.400               11                9.3                501
  32.000               12                7.7                213
  40.300               13                5.0                 70
  50.800               14                2.8                 19
                                                  Total Population = 152588
```

FIGURE A2.—An EMPSA analysis raw data record produced by the program CLTRT.

For each analysis enter:

a) Operator
b) Requestor
c) Cruise id
d) Project id
e) Sample id
f) Lab Number
g) Tube diameter
h) Initial micrometer diameter
i) Associated channel number
j) Active Channels switch setting

Press the PRINT/PLOT button on the EMPSA after the analysis is complete and "Waiting For Sample" prompt is given.

A copy of the identifiers and the raw data are generated on the printer.

The operator may then save (S), reprint (R), or, if the data is no good, purge the data (E). Upon responding to any of the three choices, the program is cycled back to (a) above.

A default on any of the prompted identifiers other than sample id or lab number will enter the identifier from the previous analysis.

Restrictions: Operator, requestor, cruise id, and project id may be up to 19 characters and have no imbedded commas or spaces. Speed requirements necessitate that the BASIC source language be compiled into machine language.

Subprograms Required: BASTOD. The KTCSET routine, which sets the real time clock, must be included when TurboDos is generated to handle sample timing interrupts.

Storage Requirements: 9.5k bytes

Errors and Diagnostics: After transfer of accumulated data from the EMPSA to the computer, the program checks for transmission errors. If any error is detected the user is prompted to reenter the data by use of the EMPSA PRINT/PLOT button.

3. SEDIT

Program Name: SEDIT

Type: Main program

Purpose: To examine, edit, and generate additional hard copies of the raw data records. SEDIT also archives these records into master files and writes the master files to transfer disks.

Machine: Pro-Comp/8-8

Operating System: TurboDos version 1.22

Source Language: BASIC

Program Category: Data Processing

Input: RSA and EMPSA raw data records, which are stored to disk during operation of the RSAT and CLTKT programs, are input into files accessed by SEDIT.

Output: Hard copies of the RSA and EMPSA raw data records and master files which have been written to transfer disks

Usage: To execute program:

[RU] SEDIT

User is prompted for which raw data records (RSA, EMPSA, or both) are to be accessed. Operations performed by the program are listed in and selected from the Main Menu (table A-1).

When a number 1 through 7 is entered on the terminal, the operator can use that utility to perform the corresponding task.

In the Display or Print mode, the user can scroll through the list of logged samples. This list is composed of lab numbers and letters (O=open, D=deleted, or A=assigned) designating the samples present status.

When options 3 or 4 are selected, the Assign and Edit/Display Mode Menu (Table A-2) appears on the top of the screen, and the data records scroll up from the bottom.

The samples may be scrolled as in the Display or Print modes but, when the A, D, or O commands are given, the indicated sample is tagged for that respective operation.

When the Edit command (E) is given, the raw data record for the corresponding lab number appears on the terminal screen. The user is then prompted for all editing commands.

When all raw data records on the SEDIT disk have been written to transfer disks or deleted, the K key on the terminal will permanently remove these records from the disk.

Storage Requirements: 14.2k bytes

TABLE A1.—*Main menu utilized by the program SEDIT*

	MAIN MENU
1	DISPLAY STATUS OF LOGGED SAMPLES
2	PRINT STATUS OF LOGGED SAMPLES
3	ASSIGN MASTER FILE NAME TO LOGGED SAMPLES
4	EDIT/DISPLAY LOGGED SAMPLES
5	ARCHIVE ASSIGNED RECORDS TO TRANSFER DISKS
6	RESTART PROGRAM
7	EXIT TO SYSTEM

TABLE A2.—*Edit/Display mode menu utilized by the program SEDIT*

	EDIT/DISPLAY MODE MENU
?	DISPLAY THIS MENU
A	ASSIGN MASTER FILE NAME
B	BACKUP TO PREVIOUS SAMPLE
D	DELETE THIS SAMPLE
E	EDIT THIS SAMPLE
O	REOPEN TAGGED SAMPLE
Q	EXIT SESSION
RETURN KEY	SCROLLS TO NEXT SAMPLE

4. GSANV

Program Name: GSANV

Type: Main Program

Purpose: To add navigational readings and descriptive identifiers to the grain-size analysis raw data files

Machine: HP 2100 MX

Operating System: RTE-IVB

Source Language: RATFOR

Program Category: Data Processing

Input: The user is prompted to enter data via the keyboard.

Output: A file which can be read by the GSTAT and JSORT programs for inclusion in a pre-GRASP grain-size analyses file

Usage: To execute program:

:[RU,]GSANV

For each run enter:

a) Nav filename:# Enter the output file for field information. The file may be a new file or, if file named exists, GSANV will append to the existing file.

b) Are data in decimal degrees?# If answered yes (y or Y), locations are expected in decimal degrees. Otherwise, locations should be entered as follows: Degrees Minutes Seconds N or S (or E or W) or Degrees Decimal Minutes N or S (or E or W). If seconds are not entered, minutes should be entered as decimal minutes; the program converts the entry to decimal degrees. Negative quadrants are S and W.

For each sample enter:

c) Lab Number:# Enter the sample's lab number which will be used as a "key" to match field information with grain-size analysis data in GSTAT.

d) Enter latitude (DD DMIN N or S):# (eg. 45 23.7 N)
This latitude will be converted to decimal degrees before output to file. Or enter latitude in decimal degrees.

e) Enter longitude (DD DMIN E or W):# (eg. 102 23.5 W)
This longitude will be converted to decimal degrees before output to file. Or enter longitude in decimal degrees.

Null fields are allowed for the following:

f) Sampling Device (2 characters):#
g) Area (2 characters):#
h) Depth: #
i) Top-depth: #
j) Bottom-depth: #

Restrictions: Maximum number of characters for lab number is 5. If latitude/longitude is in decimal degrees, the maximum characters is 12 digits including the decimal point. Otherwise, each numeric value in the latitude or longitude entry, will be limited to 7 characters (that is 113.12345678 or 113 12 12.1234 W).

Storage Requirements: 15 pages required for program and subroutines: GNV, GSANV, GSTNV, and others from %JJLIB and %TOOLS

5. CLTRM

Program Name: CLTRM

Type: Main program

Purpose: To create or update an existing sediment raw data file with Coulter analyses from sediment laboratory, when analyses were not run on APSAS

Machine: HP 2100 MX

Operating System: RTE-IVB

Source Language: RATFOR

Program Category: DATA PROCESSING

Input: At the beginning of each run, the user will be prompted for the project id, cruise id, requestor, operator, and analysis date. These identifiers will apply to each sample within a given run and may only be changed by exiting the program. For each sample, a lab number and sample id (or field number) are requested. The lab number is the key to all further processing of sediment analyses. The analysis data are then keyed in by the user.

Output: For each sample, a header is written, including lab number, aperture diameter, sample id, project id, cruise id, requestor, operator, and analysis date. The µm diameters and corresponding EMPSA data are tabulated below the header in relative frequency percents (table A-3).

Usage: When responding to a prompt, always wait for "#". A "Control D" will abort the run in steps a-j. All entries since the last write are ignored (see step l).

To execute program:

```
:[RU,]CLTRM
```

For each run enter:

a) Raw data master file name:#
Program will respond with one of the following:
"Starting new file" or "Appending to existing file."
Any error will cause program termination.

b) Project id: # (up to 19 characters with no embedded spaces or commas)

c) Cruise id: # (up to 19 characters with no embedded spaces or commas)

d) Requestor: # (up to 19 characters with no embedded spaces or commas)

e) Operator's Name: # (up to 19 characters with no embedded spaces or commas)

f) Analysis Date: # MO/DA/YR (up to 8 characters)

For each sample analysis:

g) Lab Number: #

Options	Results
CCNNN	Lab number assigned by sediment lab manager, where C = character and N = number
. (period)	Allows user to exit CLTRM
Control D	Allows user to exit CLTRM
space	Allows user to enter next analysis of same lab number/sample id (last entered within current run). Defaults aperture diameter to 30 µm.

h) Sample id: #

Options	Results
Sample ID	As specified by requestor or chief scientist. (No embedded spaces or commas)
Control D	Immediately terminates execution. Program will not allow null sample id

i) Aperture Diameter: #

Options	Results
200	For 200 µm aperture diameter
30	For 30 µm aperture diameter
Control D	Immediate termination of CLTRM
Space	Defaults aperture diameter to 200 µm
Any other response	Invalid response, try again

j) Initial µm diameter ($\substack{30.8\\12.7}$ default): #

Options	Results
Space	If aperture diameter = 200, initial µm diameter defaults to 30.8 µm.

TABLE A3.—*Example of one record from a raw data master file created by CLTRM*

```
AA000
200
MI-19N
BTP
M2
MBOTHNER
JFREDERICKS
8/30/82
     4.000          8.00
     5.040         10.90
     6.350         12.90
     8.000         11.50
    10.080          9.70
    12.700          8.40
    16.000          7.80
    20.200          7.30
    25.400          7.30
    32.000          7.40
    40.300          5.40
    50.800          2.80
```

	Otherwise, the initial um diameter defaults to 12.7 μm
μm diameter	See Restrictions for list of μm diameters. Start at larger diameters
Control D	Immediate termination of CLTRM.
NOTE:	The program will not allow entry of Coulter data with um diameter greater than 50.8 μm

k) μm diameter: #

Options	Results
frequency %	Enter frequency %; or,
"."	allows user to exit CLTRM
Control D	Terminates entry for specified sample analysis, i.e., given lab number and aperture diameter
Space	Prompts user with same μm diameter until value entered or entry terminated as above

l) Is data OK?

Response	Results
Y	Data set is accepted and at this time written to specified raw data master file, and user is cycled back to (g) above
N	Allows user to enter μm diameter, correct frequency % until "." is entered, at which time data is listed and (l) is repeated until data is okay. If Control D entered as response to "μm diameter, correct frequency % #", data set is ignored and program returns to (g) above
Control D	Data set is ignored, returns to (g) above

All other responses are invalid.

Restrictions: Length of identifiers are specified in Usage (above). The μm-diameter values are 50.8, 40.3, 32.0, 25.4, 20.2, 16.0, 12.7, 10.08, 8.00, 6.35, 5.04, 4.00, 3.17, 2.52, 2.00, 1.59, 1.26, 1.00, .794, .630, .500, and .397.

Subprograms Required: COL (main), COULTR (main subprogram), SETUP, SHEAD, GETDAT, GETRES, SPRINT, UCHECK, LINCLR, and OREAL, and MOUT.

COMMONS: GSACOM

SYMBOL FILE: GSASYM

LIBRARIES: Above subprograms are included in GSALIB. The library XTOOLS is required, also.

Storage Requirements: The main program plus all subroutines requires 17 pages of memory.

Errors and Diagnostics: Self-explanatory

6. RSAM

Program Name: RSAM

Type: Main program

Purpose: To create or update existing sediment raw data file with RSA data, sample weight, coarse fraction weight and sand weight from sediment laboratory analyses not performed by using the Automated Particle Size Analysis System (APSAS)

Machine: HP 2100 MX

Operating System: RTE-IVB

Source Language: RATFOR

Program Category: Data Processing

Input: At the beginning of each run, the user will be prompted for the project id, cruise id, requestor, operator, and analysis date. These identifiers will apply to each sample within a given run and may only be changed by exiting the program. For each sample, a lab number and sample id (or field number) is requested. The lab number is the key to all further processing of sediment analyses.

Output: For each sample, a header is written, including lab number, "RSA", sample id, project id, cruise id, requestor, operator, and analysis date. Sample weight, coarse fraction weight, sand weight, and the phi classes and relative frequency percents for the sand and gravel fractions are tabulated below the header (table A-4).

Usage: When responding to a prompt, always wait for "#". "Control D" will allow user to abort RSAM in steps a-h.

To execute program:

```
:[RU,]RSAM
```

For each run enter:

a) Raw data master file name:#

b) Project id: # (up to 19 characters with no embedded spaces or commas)

c) Cruise id: # (up to 19 characters with no embedded spaces or commas)

d) Requestor: # (up to 19 characters with no embedded spaces or commas)

e) Operator's Name: # (up to 19 characters)

f) Analysis Date: # MO/DA/YR

For each sample analysis:

g) Lab Number: #

Options	Results
CCNNN	Lab number assigned by sediment lab manager, where C = character & N = number
. (period)	Allows user to exit RSAM
Control D	Allows user to exit RSAM

h) Sample id: #

Options	Results
Sample id	As specified by requestor or chief scientist (no embedded spaces or commas)
Control D	Immediately terminates execution

Program will not allow null sample id.

i) Net sample weight #

Net coarse weight #

Net sand weight #

Options	Results
Weight (in grams)	Weight accepted
Space	Returns user back to "net sample Weight" for reentry of weights
Control D	Same as space (above)

j) Phi class: #

Options	Results
Frequency %	The percent of the given phi class relative to either the sand weight or the gravel weight, as specified
"." or Control D	To terminate entry for specified sample analysis, i.e., given lab number
Space	Prompts user until value entered or entry terminated as above

k) Is data OK?

Response	Results
Y	Data is accepted and at this time written to specified raw data master file and user is cycled back to (g) above.
N	Allows user to enter phi class, correct frequency % until "." is entered as the first

TABLE A4.—*Example of one record from a raw data master file created by RSAM*

```
>XX576    RSA
 M11-5-118D
 BTF
 GYRE-84
 MBOTHNER
 LPOPPE
 10/16/84
      54.3434
      51.0034
      49.5682
    -2.000      26.80
    -1.000      73.20
      .000       3.00
     1.000      30.10
     2.000      36.90
     3.000      21.30
     4.000       8.70
```

character, at which time data is listed and (k) is repeated until data is okay. If Control D is entered as a response to "phi, frequency % #", the data set is ignored and the program returns to (g) above.

Control D — The data set is ignored and the program returns to (g) above.

All other responses are invalid.

Restrictions: Length of identifiers are specified in Usage (above). The percent sand (phi 4 to phi 0) must add up to 100 percent before "RSAM" will allow user to enter gravel data. The percent gravel (phi -1 to phi -5) must add up to 100 percent before RSAM will allow user to exit data entry loop.

Subprograms Required: RS (main).

COMMONS: CSACOM, RSACOM

SYMBOL FILE: GSASYM

LIBRARIES: Necessary subprograms are included in CSALIB and rsalib. ZTOOLS is required also.

Storage Requirements: The main program plus all subroutines require 17 pages of memory.

7. JSORT

Program Name: JSORT

Type: Main Program

Purpose: To sort multiline records and output records that are complete to an output file for further processing

Machine: HP 2100 MX

Operating System: RTE-IVB

Source Language: RATFOR

Program Category: Data Processing

Input: Data file with unsorted multiline records

Output: A file that can be read by CSTAT, contains sorted and complete raw data (200 μm EMPSA data, 30 μm EMPSA data, RSA data, and field and navigation id records). Data sets that are not complete remain in the input file. A list of what data sets are missing from incomplete records can be listed from file ERRNAM.

Usage: To execute program:

:[RU,]JSORT

Input file name:# file with raw data from grain-size analyses.
Control D stops run

JSORT opens input file. Then it creates scratch file (JSnnn) for sorting. It also creates a sort command scratch file (SCnnn). A header (table A-5) is output before copying each line to JSnnn:

JSORT spawns program SORT with commands stored in SCnnn. The file JSnnn is sorted by key, id, type and sn. Output is the sorted file JSnnn. SORT then prompts user: Enter filename for complete data set output: #

If a complete raw data output file was specified, JSORT then outputs data from JSnnn as follows. Complete data sets are written to the output file; incomplete data sets are written to a scratch file (TOnnn). Upon completion of this copy phase, scratch file TOnnn is copied over the original file. All scratch files are purged, and input and output files are

TABLE A5.—*Header file created by the program JSORT*

name	cols	code	description
key	(cols.1,5)	lab number	
id flag	(col.6)	0	data record
		9	header record for sort
type flag	(col.7)	0	header record
		1	typ1=200μm ERMCPSA data
		2	typ2=30μm ERMCPSA data
		3	typ3=RSA data
		4	typ4=NAV (field data)
sn	(cols.8,9)		sequence (line) number within data set

closed. A Control D will abort the program and leave the unsorted source file untouched.

Restrictions: Maximum number of characteristics for lab number is 5.

The maximum number of lines for each data set (that is 200 μm ERMCPSA for a given sample) is 99. The program is set up to read groups flagged by >> for each data set. The lab number should be in the form of AANNN where A is alphanumeric and NNN is a number.

Storage Requirements: 16 pages required for program and subroutines: JSO and others from %JSOLB, %JJLIB, and %TOOLS

Errors and Diagnostics: When the program has aborted, the scratch files are left on the system. This allows the user to check through these files when necessary.

8. GSTAT

Program Name: GSTAT

Type: Main Program

Purpose: To retrieve sediment lab analyses and the corresponding identifiers from the raw data files and compute the modified frequency percents. The method of moments statistics and Shepard's (1954) textural classification are computed and listed on the standard output file (STDOUT).

The user may request a listing of the modified frequency percents, a histogram, and a cumulative frequency plot. These options, when requested, will be written to the STDOUT file. The inclusive graphics statistics may also be computed and listed to STDOUT. When a pre-GRASP file is specified, the cumulative frequency percents, phi classes, method of moments statistics, and the appropriate identifiers will be written to the file in a free-field GRASP format.

Machine: HP 2100 MX

Operating System: RTE-IVB

Source Language: RATFOR

Process: For each sample, the 200 μm EMPSA data, the 30 μm EMPSA data, the RSA data and the field identifiers and navigational information (NAV) are retrieved from the sorted raw data file. A message is sent to the ERROUT file noting each successful retrieval.

The EMPSA data are modified by subroutines MPVC and SUMRY to relative percent fines. Then the data is further modified by subroutines WTFP to the relative percent values of the whole sample ("Volume Percent Method" of computation found in Coulter Counter Model TA-II Product Reference Manual).

In MPVC:

kO is calculated as follows

$$k(1) = \frac{\text{fp200}(8.0\mu m)}{\text{fp30}(8.0\mu m)}$$

$$k(2) = \frac{\text{fp200}(6.35\mu m)}{\text{fp30}(6.35\mu m)}$$

$$k(3) = \frac{\text{fp200}(5.04\mu m)}{\text{fp30}(5.04\mu m)}$$

$$k(0) = \frac{\text{fp200}(8.0\mu m)+\text{fp200}(6.35\mu m)+\text{fp200}(5.04\mu m)}{\text{fp30}(8.0\mu m)+\text{fp30}(6.35\mu m)+\text{fp30}(5.04\mu m)}$$

The adjustment factor, k0, is the k value (k(1), k(2), k(3)) that is closest to k(0). The crossover point at which this adjustment factor is applied is where k(n) is closest to 1.0. For data from the $64.0\mu m > x \geq 50.8\mu m$ interval to the data value at the crossover point interval, the 200 μm aperture data are used. From that point up to last data points, the frequency percent equals fp(30) x k0. The frequency percents are then normalized to 100 percent for the fines in SUMRY, and the EMPSA data are grouped to form phi classes where:

Diameter µm	Class
$64.0>x\geq32.0$	=phi(5.0)
$32.0>x\geq16.0$	=phi(6.0)
$16.0>x\geq8.0$	=phi(7.0)
$8.0>x\geq4.0$	=phi(8.0)
$4.0>x\geq2.0$	=phi(9.0)
$2.0>x\geq1.0$	=phi(10.0)
$1.0>x\geq.50$	=phi(11.0)

At this point, the relative frequency percents for phi(11) to phi(5.0) (fines) sum to 100 percent; the sum of frequency percents from phi(4) to phi(0) (sand) equals 100 percent; and, if there is gravel in the sample, the relative frequency percents phi(-1) to phi(-5) sum to 100 percent. The subroutine WTFP calculates the relative percents for the sample as follows:

Weight fines = sample weight - coarse weight

Percent fines = weight fines/sample weight

for phi(11) - phi(5)
fp(%) = fp x percent fines

percent sand = weight sand/sample weight for

For phi(4) - phi(0)
fp(%) = fp x percent sand

weight gravel = weight coarse - weight sand

percent gravel = weight gravel/sample weight

for phi(-1) - phi(5)
fp(%) = fp x percent gravel

The RATFOR code for the computation of the method of moments statistics was converted from the Sigma 7 SDA Fortran IV code written by Jackie Webster, 1968, Woods Hole Oceanographic Institution, Woods Hole, Mass. The following documentation was extracted from the program writeup GSANAL to describe the methods of computation.

Let frequency percent be represented by F_i and phi size by Φi, i going from 1 to N, where N is the number of data points in the sample.

The modes of the sample are found by examining the first differences of the frequency percent. When the first difference, $\Delta_i = F_i - F_{i-1}$, changes sign, the center of the phi class corresponding to F_{i-1} is taken as a mode, provided the frequency percent for that class, F_{i-1}, is greater than 5 times the class interval, $\Delta\Phi$. This latter provision sets an arbitrary limit to eliminate minor modes within the distribution.

The median of the sample is found by calculating cumulative frequency percents and then interpolating linearly to find the phi value corresponding to a cumulative frequency percent of 50.

Let C_i be the center of the phi class corresponding to F_i. Then moment measures are calculated as follows:

$$n_1 = \sum_{i=1}^{N} F_i \; C_i \; /S$$

$$n_2 = \sum_{i=1}^{N} F_i \; C_i^2 \; /S$$

$$n_3 = \sum_{i=1}^{N} F_i \; C_i^3 \; /S$$

$$n_4 = \sum_{i=1}^{N} F_i \; C_i^4 \; /S$$

where:
$$S = \sum_{i=1}^{N} F_i$$

The uncorrected moments about the mean are found

$$m_2 = n_2 - n_1^{2}$$

$$m_3 = n_3 - 3n_2n_1 + 2n_1^{3}$$

$$m_4 = n_4 + n_1(-4n_3 + 6n_1n_2 - 3n_1^{3})$$

Shepard's correction (Kenney and Keeping, 1954, p. 95-96) is applied to the fourth and second moments about the mean:

$$(m_4)_{corr} = m_4 - \frac{m_2}{2}(\Delta\phi)^2 + \frac{7}{240}(\Delta\phi)^4$$
$$(m_2)_{corr} = m_2 - (\Delta\phi)^2 /12$$

where $\Delta\phi$ is the phi class interval for the sample.

The standard deviation, skewness, and kurtosis are computed as follows:

$$\sigma = (m_2)_{corr}$$

$$Sk = m_3/2\sigma(m_2)_{corr}$$

$$K = \frac{(m_4)_{corr}}{[(m_2)corr]^2} -3$$

The textural classification of sediments as described by Shepard (1954) is based on a sample distribution with no more than 10 percent gravel. If the sample contains more than 10 percent gravel, it will be classified as a GRAVEL. Otherwise, the percent gravel is added to the percent sand for the textural classification.

When a pre-GRASP file was specified, the appropriate data for the sediment master data base will be output in free-field GRASP format.

When the IG option is selected by the user on the run line, the inclusive graphics statistics are then computed and listed to STDOUT. The cumulative frequency percent curve is approximated by using the IMSL routine IQHSCU and the IMSL routine ICSEVU. The cumulative frequency percents are evaluated at this point for all phi classes between 11 and -5 at 0.01 phi intervals. The inclusive graphics statistics are then computed on the basis of methods described by Folk (1974). A linear interpolation between the two closest phi classes (at 0.01 intervals) gives a best approximation of the phi values corresponding to the cumulative frequency percents at 5, 16, 25, 50, 75, 84, and 95 percent.

Program Category: Data Processing

Input: The raw data input file contains the 200 µm EMPSA data, the 30 µm EMPSA data, the RSA data, and the appropriate nav-field identifiers for all samples. This file has been sorted by using the program JSORT to ensure that the data sets are in the appropriate order and that all sample entries are complete.

See table A-6 for an example of a complete sample from a sorted raw data master file.

Output: The program will list all samples that have been successfully retrieved from the raw data file in the ERROUT file.

The method of moment statistics and the textural classification according to Shepard (1954) will automatically be printed on the STDOUT file.

TABLE A6.—*An example of one complete sample from a sorted raw data master file for input into GSTAT*

```
>>AA562   200
  M05-01-00-BL
  BTF
  MS
  MBOTHNER
  R3
  1/25/83
        2.520          .00
        3.170          .00
        4.000         9.50
        5.040        11.30
        6.350        12.50
        8.000        11.30
       10.080         9.00
       12.700         7.00
       16.000         5.30
       20.200         4.90
       25.400         4.40
       32.000         5.00
       40.300         4.70
       50.800         4.20
>>AA562    30
  M05-01-00-BL
  BTF
  MS
  MBOTHNER
  R3
  1/25/83
         .397          .00
         .500          .00
         .630         6.60
         .794         5.70
        1.000         7.10
        1.260         7.30
        1.590         7.50
        2.000         7.90
        2.520         9.80
        3.170        10.30
        4.000        10.20
        5.040         9.40
        6.350         8.10
        8.000         5.60
       10.080         3.00
       12.700          .40
>>AA562  RSA
  M05-01-00-BL
  BTF
  MS
  MBOTHNER
  R3
  1/26/83
     24.7000
     24.6700
     24.6700
        3.000        30.00
        4.000        70.00
>>AA562,NAV,41.207889,-67.241056,VV,GB,53.000,0,2
```

Optional output includes a line-printer plot of the grain-size distributions as both a histogram and a cumulative frequency curve, a listing of the frequency percents for each phi class, and the computed inclusive graphics statistics. These options are directed to the STDOUT file.

All output to STDOUT is formatted to allow output to the line printer by using the dump command (:DU,STDOUT). A sample dump of a STDOUT file with all three options selected (PR, PL, and IG) can be found in figures A-3, A-4, and A-5.

When a pre-GRASP data file is specified, the modified frequency percents, phi classes, sample weight, method of moments statistics, and corresponding field identifiers will be output to the file named as follows:

line 1:

```
    lab_number,
    sample_id(field_number),
    project_id,
    cruise_id,
    requestor_name,
    month,
    day,
    year,
    latitude (in decimal degrees),
    longitude (in decimal degrees),
    sampling_device
    sampling_area
    depth,
    top_depth,
    bottom_depth,
    sample_weight,
    percent_sand,
    percent_gravel,
    percent_silt,
    percent_clay,
```

line 2:

```
    sediment_class_name,
    mean,
    standard_deviation,
    skewness,
    kurtosis,
    modal_class_1,
    modal_frequency_1,
    modal_class_2,
    modal_class_3,
    modal_frequency_3,
    number_of_modes,
```

line 3:

```
    phi11, cfp11, ... phi04, cfp04,
```

line 4:

```
    phi03, cfp03, ... phi-5, cfp-5,$
```

where "cfp" represents cumulative frequency percent.

The field delimiter is a comma (,) and the record delimiter is a dollar sign ($).

Usage: To execute program:

```
[RU,]CSTAT [,PR,PL,IC] >STDOUT ?ERROUT
```

Where STDOUT is the file name to which STDOUT output will be directed and ERROUT is the file name where the listing of error conditions and successful data retrievals will be stored.

The options may be selected in any combination and are described as follows:

PR to list modified frequency percentages to STDOUT;

PL to plot histogram and cumulative frequency on STDOUT;

IC to compute and list inclusive graphics statistics.

The user is then prompted to enter:

raw data master file name:#

File named is opened.
Control D (EOF) aborts run.
Null field is invalid.

Pre-GRASP file name:#

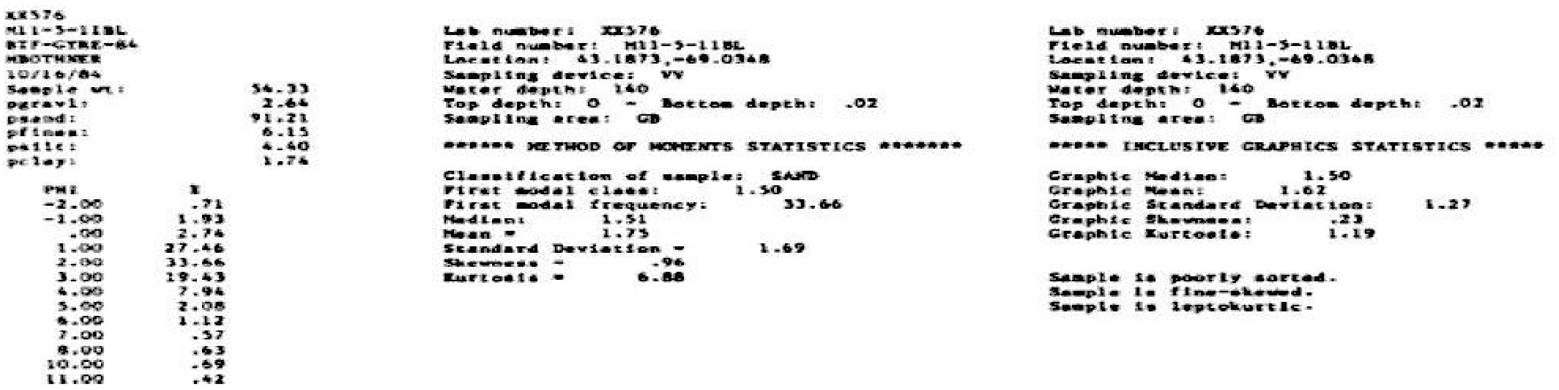
File is opened or created.
Control D aborts run.
Null field is valid and will omit pre-GRASP output during current run.

When the job or run has finished, list the error file on the terminal to check for successful completion of run: :LI,ERROUT.

Then the STDOUT file may be dumped to the line printer as follows:

```
:USL,18
:DU, STDOUT,18
:CN,18
:CN,18
:USL,18,-
```

Timing: Approximately 10 samples (all options selected) in 1 minute



```
XX576                                     Lab number:  XX576                                   Lab number:  XX576
M11-5-11BL                                Field number:  M11-5-11BL                            Field number:  M11-5-11BL
BIF-GTRE-84                               Location:  43.1873,-69.0348                          Location:  43.1873,-69.0348
MBOTHNER                                  Sampling device:  VV                                 Sampling device:  YV
10/16/84                                  Water depth:  140                                    Water depth:  140
Sample wt:                  54.33         Top depth:  0  -  Bottom depth:  .02                 Top depth:  0  -  Bottom depth:  .02
pgravl:                      2.64         Sampling area:  GB                                   Sampling area:  GB
psand:                      91.21
pfines:                      6.15         ****** METHOD OF MOMENTS STATISTICS *******          ***** INCLUSIVE GRAPHICS STATISTICS *****
psilt:                       4.40
pclay:                       1.74         Classification of sample:  SAND                      Graphic Median:        1.50
                                          First modal class:         1.50                      Graphic Mean:        1.62
     PHI          %                       First modal frequency:          33.66                Graphic Standard Deviation:      1.27
   -2.00         .71                      Median:        1.51                                  Graphic Skewness:        .23
   -1.00        1.93                      Mean =         1.73                                  Graphic Kurtosis:        1.19
     .00        2.74                      Standard Deviation =      1.69
    1.00       27.46                      Skewness =          .96                              Sample is poorly sorted.
    2.00       33.66                      Kurtosis =        6.88                               Sample is fine-skewed.
    3.00       19.43                                                                           Sample is leptokurtic-
    4.00        7.94
    5.00        2.08
    6.00        1.12
    7.00         .57
    8.00         .63
   10.00         .69
   11.00         .42
```

FIGURE A3.—Example of a printer dump of a GSTAT STDOUT file with the PR and IG options selected. Figure shows percentages gravel, sand, fines, silt, and clay; relative weight percent grain-size distribution; navigation; sample parameters; method of moment statistics; verbal classification and limits; and graphics statistics.

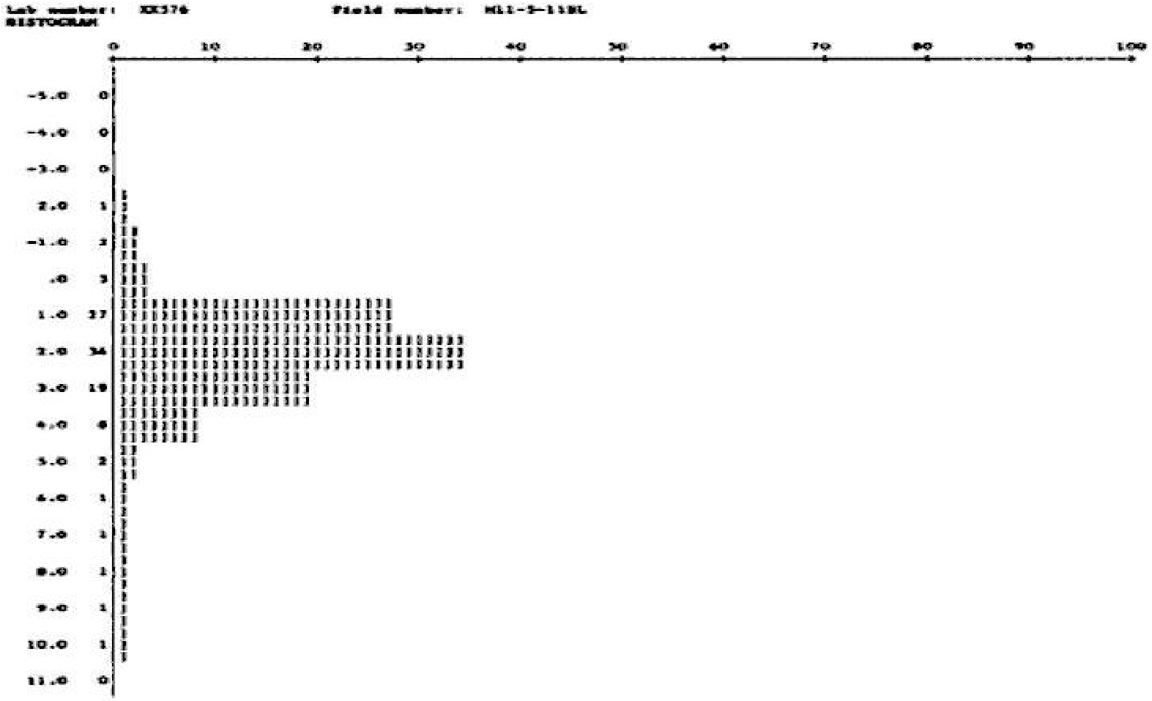


FIGURE A4.—Example of a printer dump of a GSTAT STDOUT file histogram plot.

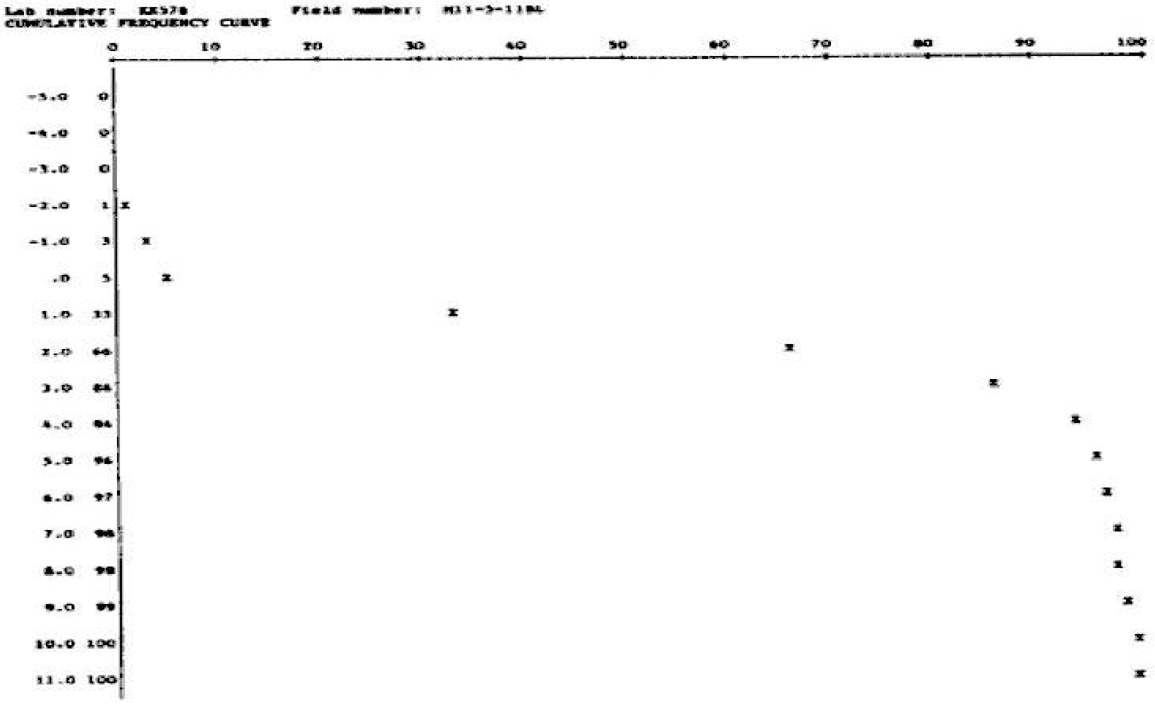


FIGURE A5.—Example of a printer dump of a GSTAT STDOUT file cumulative curve.

Restrictions: The maximum number of samples that may be read by GSTAT is 999. All other restrictions will have been delineated in RSAM, ARSA, CLTRM, ACOLTR, GSANV and JSORT.

Storage Requirements: 27 pages required to load GST, GSTAT, ICHSCU, ICSEVU, and routines from %GSTLB, %RSALB, %GSALB, %IGSLB and %TOOLS libraries

Errors and Diagnostics: A check is made to ensure that the lab number for all entries of a given sample are the same. If they do not match or unexpected EOF is encountered within the sample entry, an error message will be printed and the program will stop.

When "Coulter data not found for lab number: AANNN" is printed, the raw data file was not sorted by using JSORT. Check sediment lab log to determine status of file.

When an error is detected or the program does not run through a complete raw data set, a user can determine where the problem lies by inspecting the ERROUT file. The last successful read will be noted in ERROUT, and the last successful analysis will be listed on STDOUT.

APPENDIX B

Program Software

```
1.   RSAT
1000 DEFINT I-M
1010 ICNT-4800
1020 IPLOT-7
1030 TFAC-.0491803:REM FOR TURBODOS 122hz TIMER ie -(1/122)*6
1040 FFAC-10: REM TIME DELAY FACTOR
1050 WIDTH LPRINT 255
1060 MINC-INT (50/TFAC+15):MAXC-INT(180/TFAC+15):REM MIN AND MAX SAMPLE COUNT
1070 LN1$-"PLEASE ":LN2$-"ENTER"
1080      DIM TIME (13),W$(7),MO$(12),IV(4800),IVP(10),GF(5),PHIT(13),PHIV(12)
1090      DIM PHIP(12),PHIR(6),QP$(9)
1100      DIM MO(12)
1110      DATA "Sunday","Monday","Tuesday","Wednesday","Thursday","Friday"
1120      DATA "Saturday"
1130      DATA "January","February","March","April"
1140      DATA "May","June","July","August","September","October","November"
1150      DATA "December"
1160      DATA 31,28,31,30,31,30,31,31,30,31,30,31
1170      DATA 4.2,5.1,6.0,7.1,9,11.3,14.5,21,31,50,94,160,2000
1180      FOR I=0 TO 6
1190      READ W$(I)
1200      NEXT I
1210      FOR I=1 TO 12
1220      READ MO$(I)
1230      NEXT I
1240      FOR I= 1 TO 12: READ MO(I): NEXT I: 'DAYS PER MONTH
1250      FOR I=1 TO 13:READ X:PHIT(I)=X+.5:NEXT I:REM  ADD .5 SEC PRECOUNT TIME
1260        GOSUB 3380
1270 REM *********** SET UP SLAVE PORT B FOR ADM12S A/D CONVERTER **********
1280 OUT 3,24               ' DISABLE PORT
1290 OUT 3,3                ' SELECT WR3
1300 OUT 3,193              ' 8 BITS/CHAR & RX ENABLE
1310 OUT 3,4                ' SELECT WR4
1320 OUT 3,76               ' X16 BAUD & TWO STOP BITS
1330 OUT 3,5                ' SELECT WR5
1340 OUT 3,234              ' DTR,  8 BITS/CHAR, & TX ENABLE
1350 OUT 3,0                ' LEAVE POINTER AT RO
1360 OUT 16,238             ' SET BAUD RATE PORT FOR 9600 ON A & B (HEX EE)
1370 REM ********** END PORT SET UP ******************************************

1380 REM


*************** INTRODUCTION ***************
1390 PRINT"  RSA Data Logging Program        TURBODOS Version"
1400 PRINT"  Elisson Data Services           10/25/84"
1410 PRINT
1420 PRINT"Please answer query questions with 'Y' for Yes, 'N' for No, or the
     information asked for.  You may also just hit the 'RETURN' key to accept
     the default reply which is contained within square brackets, i.e. [
     BLA ].
1430 PRINT:INPUT"Do you want the data to be plotted on the printer? [Y] ";A$
1440 PLT=-1:IF A$-"N"OR A$-"n" THEN PLT-0
1450 IF NOT PLT THEN 1490
1460 PRINT:PRINT"Enter plot scale factor (1 to 10) [";IPLOT;:INPUT"] ",A$
1470 IF A$<>"" THEN IPLOT-INT(VAL(A$))
1480 IF IPLOT<1 OR IPLOT>10 THEN 1460
1490 OLD=0
1500 PRINT:PRINT"Please set printer to proper 'Top Of Form' then turn it ON."
1510 INPUT"When done, please hit 'RETURN' ";A$
1520 LPRINT CHR$(27)"C"CHR$(0)CHR$(11);: REM SET EPSON FOR 11 IN. FORM LENGTH
1530 REM


************* START THE SAMPLE SEQUENCE ***********

1540 VER =0:SID$-"PLEASE ENTER":IGF=-1
1550 GOSUB 3380: REM READ CLOCK
1560 PRINT:PRINT "Operator Name: [";OPN$;:INPUT"] ",A$
1570 IF A$-"" THEN 1600
```

```
1580  OPN$=A$:PRINT"   Hello,  ";OPN$;".  Please  run  one  or  two  test  samples
      before you start"
1590 PRINT  "  the real ones."
1600 PRINT:PRINT "Requested by:  [";SN$;:INPUT"] ",A$
1610 IF A$="" THEN 1630
1620 SN$=A$
1630 PRINT:PRINT"Cruise I.D.:   [";CID$;:INPUT"] ",A$
1640 IF A$="" THEN 1660
1650 CID$=A$
1660 PRINT:PRINT"Project I.D.:  [";PID$;:INPUT"] ",A$
1670 IF A$="" THEN 1690
1680 PID$=A$
1690 PRINT:PRINT"Sample I.D.:   [";SID$;:INPUT"] ",A$
1700 IF A$="" AND VER=0 THEN 1690
1710 IF A$="" AND VER>0 THEN 1730
1720 SID$=A$
1730 PRINT:PRINT"Lab Number: (format 'AAnnn') [";LN1$;LN2$;:INPUT"] ",A$
1740 IF A$="" THEN 1780
1750 IF LEN(A$)<3 OR LEN (A$)>5 THEN 1730
1760 LN1$=LEFT$(A$,2)
1770 LN2$=MID$(A$,3,3)
1780 IF LEN(LN1$)<> 2 OR LEN (LN2$) >3 OR VAL(LN2$)=0 THEN 1730
1790  PRINT:PRINT"Time  delay  factor?  Positive  integer  only.  1  unit  =  .05
      seconds [";FFAC;:INPUT"] ",A$:IF A$="" THEN 1820
1800 FFAC=INT(VAL(A$))
1810 IF FFAC<0 THEN PRINT "Positive Value Only!":GOTO 1790
1820 PRINT:PRINT"Sample Weight? [";SW;:INPUT "] ",A$
1830 IF A$="" THEN 1850
1840 SW=VAL(A$)
1850 PRINT "Coarse weight? [";CW;:INPUT"] ",A$:IF A$="" THEN 1870
1860 CW=VAL(A$)
1870 PRINT "Sand Weight? [";SNDW;:INPUT"] ",A$:IF A$="" THEN 1890
1880 SNDW=VAL(A$)
1890 PRINT"Is there a gravel fraction? [";:IF IGF THEN PRINT "Y";
1900 IF NOT IGF THEN PRINT "N";
1910 INPUT "] ",A$
1920 IF A$="Y" OR A$="y" THEN IGF=-1
1930 IF A$="N" OR A$="n" THEN IGF=0
1940 REM ELSE REMAINS THE SAME
1950 IF NOT IGF THEN 2050
1960 PRINT"Enter Gravel Fraction percentage for each PHI Class."
1970 TGF=0
1980 FOR I=1 TO 5
1990 PRINT -1*I;" PHI % = [";GF(I);:INPUT"] ",A$
2000 IF A$<>"" THEN GF(I)=VAL(A$)
2010 TGF=TGF+GF(I):REM TOTAL GRAVEL FRACTION
2020 NEXT I
2030 IF TGF<100.001 AND TGF>99.999 THEN TGF=100!:GOTO 2050
2040 PRINT"WARNING! Gravel fraction must sum to 100."
2050 IF VER>0 THEN 2070
2060 VER=VER+1:PRINT CHR$(27);"*";"PLEASE VERIFY THE DATA":PRINT:GOTO 1560
2070 PRINT:INPUT"All entries correct? [Y] ",A$
2080 IF A$="" OR A$="y" OR A$="Y" THEN VER=0:GOTO 2100
2090 GOTO 2060
2100 GOSUB 3700:REM PRINT HEADER
2110 REM

*************** GET THE SAMPLES ***************************

2120 IVPC=0
2130 GOSUB 3230: REM READ A/D CONVERTER
2140 PRINT:PRINT"Waiting for sample introduction ";OPN$;" ..."
2150 OLDERIGR=OLDIGR:OLDIGR=IGR
2160 GOSUB 3230          ' A/D CONVERTER READ
2170 IVPC=IVPC+1:IF IVPC>10 THEN IVPC=1
2180 IVP(IVPC)=IGR: REM SAVE THE PRE-COUNT DATA
2190 IF IGR>OLDIGR+20 AND IGR>OLDERIGR+20 THEN 2240:REM TRY TO AVOID GLITCHES
2200 IF PEEK (80) MOD 6<5 THEN 2200: 'WAIT .05 SECONDS ### TURBODOS
2210 IF PEEK (80) MOD 6=5 THEN 2210: 'FINISH WAITING
2220 POKE 80,0
2230 GOTO 2150
2240 REM

 *********** SAVE THE DATA AS INTEGER ARRAY ************

2250 FOR I=1 TO 10          ' ADD THE PRE-COUNT DATA TO THE ARRAY
2260     IVPC=IVPC+1: IF IVPC>10 THEN IVPC=1
2270     IV(I)=IVP(IVPC)
2280 NEXT I
```

```
2290 I=11
2300 GOSUB 3230          ' A/D CONVERTER READ
2310 PRINT IGR,
2320 IV(I)=IGR
2330 IF I<MINC THEN 2360
2340 IF I>MAXC THEN COUNT=I:GOTO 2400:REM TIMEOUT
2350 IF IGR<(IV(1)+1) AND IGR=LASTIGR THEN COUNT=I:GOTO 2400
2360 IF PEEK(80) MOD 6<5 THEN 2360: ' TURBODOS .049 SEC TIMER
2370 IF PEEK(80) MOD 6=5 THEN 2370: ' FINISH WAIT
2380 POKE 80,0: ' RESET TIMER - PREVENTS ERROR AT HIGH COUNT
2390 I=I+1:LASTIGR=IGR:GOTO 2300
2400 PRINT"Count = ";COUNT
2410 PRINT
2420 REM  ENTRY POINT FOR REPRINT OF DATA
2430 REM


********************** SCALE DATA *******************

2440 PRINT"SCALING DATA"
2450 X=0
2460 FOR I= COUNT-9 TO COUNT
2470    X=X+IV(I)
2480 NEXT I
2490 BL=X/10
2500 PRINT "BASELINE = ";BL
2510 MAX=0:MIN=1000
2520 FOR I=1 TO COUNT
2530 IF IV(I)> MAX THEN MAX=IV(I)
2540  IF  IV(I)<BL-2  THEN  PRINT  "DATA  POINT  ";I;"  =  ";IV(I);"  REPLACED  WITH
     ";BL:IV(I)=BL ' WIPE OUT NEGATIVE GLITCHES
2550 IF IV(I)< MIN THEN MIN=IV(I)
2560 NEXT I
2570 RANGE= MAX-MIN
2580 RINC=RANGE/200:REM Y MAX= 200
2590 OFFSET = 0-MIN
2600 GOSUB 3870:REM COMPUTE PHI
2610 REM
2620 IF NOT PLT THEN 2890
2630                LPRINT"**********************************                     PLOT
**************************************"
2640 PRINT"PLOTTING"
2650 LPRINT CHR$(27)"A"CHR$(8):REM 8/72 SPACING FOR PLOT
2660 LPRINT CHR$(27)"K"CHR$(225)CHR$(1);:REM 480 COLUMNS
2670 FOR I=1 TO 480
2680 LPRINT CHR$(1);:NEXT I:LPRINT
2690 FOR J=200 TO 1 STEP-8:REM 25 LINES
2700 PRINT J
2710 HILIM=J:LOLIM=J-7:IPC=1
2720 LPRINT CHR$(27)"K"CHR$(225)CHR$(1);:REM 480 COLUMNS
2730    FOR I=1 TO 480
2740    IF I=1 OR I=480 THEN LPRINT CHR$(255);:GOTO 2830
2750    IP=I*IPLOT:IF IP>COUNT THEN 2820
2760    IF (IP*TFAC)>=PHIT(IPC) THEN LPRINT CHR$(255);:IPC=IPC+1:GOTO 2830
2770    IVAL=INT((IV(IP)+OFFSET)/RINC)
2780    IF IVAL<LOLIM THEN 2820
2790    IF IVAL>HILIM THEN 2820
2800    LPRINT CHR$(2 (IVAL-LOLIM));
2810    GOTO 2830
2820    LPRINT CHR$(0);
2830    NEXT I
2840 LPRINT
2850 NEXT J
2860 LPRINT CHR$(27)"K"CHR$(225)CHR$(1);
2870 FOR 1=1 TO 480:LPRINT CHR$(128);:NEXT I:LPRINT
2880 LPRINT CHR$(27)"2";
2890 LPRINT CHR$(12)CHR$(27)"@";:REM TOF AND RESTORE PRINTER
2900 REM

************* SAVE TO DISC *************

2910 PRINT CHR$(27);"*";:PRINT"S - Save Data To Disk   R - Reprint   E - Exit
     (Data no good)   [S] ";
2920 INPUT A$: IF A$="" OR A$="S" OR A$="s" ve to disc
2930 IF A$="R" OR A$="r" THEN GOSUB 3700:GOTO 2430 :REM  Reprint header then
     recalculate and print data
2940 IF A$="e" OR A$="E" THEN 1530 ELSE 2910
2950 PRINT"SAVING DATA"
2960 OPEN "R",#1,"RSA.NDX",6
2970 FIELD #1,5 AS DISC$,1 AS STAT$
2980 GET#1,1: REM READ REC #1
```

```
2990 NFILES=VAL(DISC$)
3000 LSET DISC$=STR$(NFILES+1)
3010 PUT #1,1
3020 LSET DISC$=LN1$+LN2$
3030 LSET STAT$ = "O"
3040 PUT #1,NFILES+2
3050 CLOSE
3060 A$=LN1$+LN2$
3070 OPEN "R",#1,"RSA.DAT",181
3080 FIELD #1,6 AS QM$,5 AS QL$,3 AS QT$,16 AS QS$,16 AS QP$,16 AS QC$,16 AS
     QR$,16 AS QO$,8 AS QD$,8 AS QH$
3090 FIELD #1,110 AS DUMMY$,7 AS QSW$,7 AS QCW$,7 AS QSN$,5 AS QP$(0),5 AS
     QP$(1),5 AS QP$(2),5 AS QP$(3),5 AS QP$(4),5 AS QP$(5),5 AS QP$(6),5 AS
     QP$(7),5 AS QP$(8),5 AS QP$(9)
3100 LSET QM$=" ":LSET QL$=A$:LSET QT$="RSA":LSET QS$=SID$:LSET QP$=PID$:LSET
     QC$=CID$:LSET QR$=SN$:LSET QO$=OPN$
3110 LSET QD$=RIGHT$(STR$(MO),2)+"/"+D$+"/"+Y$:LSET QH$=H$+":"+MI$+":"+S$
3120      LSET      QSW$=RIGHT$(STR$(SW),7):LSET      QCW$=RIGHT$(STR$(CW),7):LSET
     QSN$=RIGHT$(STR$(SNDW),7)
3130 FOR I=0 TO 4:LSET QP$(I)=RIGHT$(STR$(CF(5-I)),5) :NEXT I
3140 FOR I=5 TO 9:LSET QP$(I)=RIGHT$(STR$(PHIR(I-3)),5):NEXT I
3150 PUT #1,NFILES+1
3160 CLOSE
3170 REM ************ CLEAN UP AND START NEXT ******************
3180 LN1$="PLEASE ":LN2$="ENTER"
3190 SW=0:CW=0:SNDW=0
3200 FOR I=1 TO 5:CF(I)=0:NEXT I:REM ZERO CF
3210 GOTO 1530
3220 REM

************ A/D CONVERTER ROUTINE ************

3230 REM ******** FOR ADM12S CONVERTER ******
3240 OUT 2,192              ' ASK FOR A READING
3250 STAT=INP(3)            ' READ STATUS WORD
3260 IF (STAT AND 1)=0 THEN 3250        ' MASK & TEST FOR NOT READY
3270 LOW = INP(2)           ' READ LOW BYTE
3280 STAT=INP(3)            ' READ STATUS AGAIN
3290 IF (STAT AND 1)=0 THEN 3280        ' NOT READY
3300 HI = INP(2)            ' READ HIGH BYTE
3310 LOW=LOW AND 63         ' MASK BITS 6 AND 7
3320 HI =HI  AND 63         ' MASK BITS 6 AND 7
3330 X= HI*64+LOW           ' CONSTRUCT 12 BIT WORD
3340 IF X AND 2048 THEN 3350 ELSE 3360  ' CHECK BIT ELEVEN (SIGN)
3350 X= X OR -4096          ' CONSTRUCT 2'S COMPLEMENT NEGATIVE REPRESENTATION
3360 IGR = X                ' TEST THIS WAY FIRST
3370 RETURN
3380 REM

******************** TURBODOS READ TIME ROUTINE ****************

3390     PRINT CHR$(26): REM CLEAR SCREEN
3400     REM  THIS READS THE DATA
3410     A$="1234" ' ALLOCATE STRING AREA FOR A$
3420     ONE=1 ' GET AROUND ARGUMENT BUG IN BASCOM
3430     CALL BASTOD(ONE,A$)
3440     DAYS=(256*ASC(MID$(A$,2,1))+ASC(LEFT$(A$,1)))
3450     H$=HEX$(ASC(MID$(A$,3,1))) ' HOURS
3460     IF LEN(H$)=1 THEN H$="0"+H$
3470     MI$=HEX$(ASC(MID$(A$,4,1))) ' MINUTES
3480     IF LEN(MI$)=1 THEN MI$="0"+MI$
3490     S$="00"
3500     DAYS=DAYS-1096 ' GETS US TO DAYS PAST DEC.31,1980
3510     W=(DAYS+3) MOD 7 ' JAN 1,1981 WAS A THURSDAY
3520     I=0
3530     I=I+1
3540     Y=365 : IF I MOD 4 = 0 THEN Y=366 ' DAYS PER YEAR
3550     IF DAYS>Y THEN DAYS=DAYS-Y:GOTO 3530
3560     IF I MOD 4 =0 THEN MO(2)=29 ' LEAP YEAR
3570     Y$=RIGHT$(STR$(80+I),2) ' YEAR
3580     I=0
3590     I=I+1
3600     IF DAYS>MO(I) THEN DAYS=DAYS-MO(I):GOTO 3590
3610     MO=I
3620     IF       DAYS<10       THEN       D$="0"+RIGHT$(STR$(DAYS),1)       ELSE
     D$=RIGHT$(STR$(DAYS),2)
3630     REM
3640     PRINT CHR$(30);:REM  CURSOR HOME
3650     PRINT W$(W);" ";
3660     PRINT H$;":";MI$;":";S$;" ";
3670     PRINT MO$(MO);" ";D$;", 19";Y$
```

```
3680     PRINT CHR$(30)
3690     RETURN
3700 REM

*************** PRINT HEADER INFORMATION ***************

3710 LPRINT "Lab Number ";LN1$;LN2$
3720 LPRINT "Operator: ";OPN$;"    Requested by: ";SN$
3730 LPRINT "Cruise I.D.          ";CID$
3740 LPRINT "Project I.D.         ";PID$
3750 LPRINT "Sample I.D.          ";SID$
3760 LPRINT "Sample weight:       ";SW
3770 LPRINT "Coarse weight:       ";CW
3780 LPRINT "Sand weight:         ";SNDW
3790 IF NOT IGF THEN 3840
3800 LPRINT"Gravel Fraction Percentages:"
3810 FOR I=1 TO 5
3820 LPRINT -1*I;" Phi % = ";GF(I)
3830 NEXT I
3840 LPRINT W$(W);" ";H$;":";MI$;":";S$;" ";MO$(MO);" ";D$;", 19";Y$
3850 LPRINT
3860 RETURN
3870 REM

**************** COMPUTE PHI % **************

3880 REM****** FFAC = TIME DELAY FACTOR. 1 UNIT = TFAC  SEC
3890 REM****** TFAC = A/D CONVERT TIME BETWEEN SAMPLES [NOW .05]
3900 REM****** PHIT(N) =  TIME COUNT WHEN AT .5 PHI CROSSOVER POINT
3910 REM****** PHIV(N) =  PHI RAW VALUES AT CROSSOVER POINTS
3920 REM****** PHIP(N) =  PHI PERCENTAGES AT .5 PHI POINTS
3930 REM****** PHIR(N) =  PHI % ROUNDED TO .1% AT 1 PHI INTERVALS
3940 FOR I=1 TO 12
3950 IP=INT(PHIT(I)/TFAC):REM PHI DATA POINT
3960 IF IP+FFAC > COUNT THEN PHIV(I)=MIN:GOTO 3980
3970 PHIV(I)=IV(IP+FFAC )
3980 REM MAY WANT TO DO SOME AVERAGEING HERE
3990 IF PHIV(I)>PHIV(1) THEN PHIV(1)=PHIV(I):REM  SET ALL VALUES LESS THAN
     INITIAL TO INITIAL VALUE
4000 NEXT I
4010 FOR I=1 TO 12
4020 FOR J=I TO 12
4030             IF PHIV(J)>PHIV(I) THEN PHIV(I)=PHIV(J)
4040     NEXT J
4050 NEXT I
4060 RANGE = PHIV(1)-MIN:REM MIN FROM SCALING ROUTINE
4070 FOR I=1 TO 11
4080     PHIP(I+1)=(PHIV(I)-PHIV(I+1))/(PHIV(1)+OFFSET)*100
4090 NEXT I
4100 LPRINT:LPRINT"PHI              VALUE     PERCENTAGE":LPRINT
4110 FOR I=1 TO 12
4120     IF I MOD 2 = 1 THEN 4140
4130     PHIR(I/2)=(INT(((PHIP(I)+PHIP(I-1))*10)+.5))/10
4140 NEXT I
4150 X=0
4160 FOR I=1 TO 6
4170     X=X+PHIR(I)
4180 NEXT I
4190 IF 100-X<.001 THEN X=100
4200 PHIR(6)=PHIR(6)+(100-X)
4210 PHIR(6)=(INT((PHIR(6)*10)+.5))/10
4220 PHI=-2!
4230 FOR I=1 TO 12
4240     PHI=PHI+.5
4250     LPRINT USING"+#.#";PHI,
4260     LPRINT USING"#############";PHIV(I),
4270     LPRINT USING"########.###_%";PHIP(I),
4280     IF I MOD 2 = 1 THEN LPRINT:GOTO 4300
4290     LPRINT USING"########.#_%";PHIR(I/2)
4300 NEXT I
4310 LPRINT :LPRINT"Total percentage = ";X;"   Remainder percentage = ";100-X
4320 PHIR(2)=PHIR(2)+PHIR(1):REM ADD -1 PHI TO 0 PHI
4330 LPRINT "Corrected 0 Phi = ";PHIR(2);"%   Corrected 4 Phi = ";PHIR(6);" %"
4340 RETURN
4350 END:REM

******************************** END OF PROGRAM ********************************
```

2. CLTRT

```
1000 DEFINT I-N
1010 WIDTH LPRINT 255
1020 ID$="@ABCDEFGHIJKLMNOPQRSTUVWXYZ[_] _"
1030 LN1$="PLEASE ":LN2$="ENTER"
1040 TD=200
1050 DIM TIME(13),W$(7),MO$(12),MO(12)
1060 DIM DIA(38),VP(16),PC(16)
1070 DIM DD(2,16): REM DISK DATA ARRAY
1080 DIM QMD$(16),QVP$(16)
1090     DATA "Sunday","Monday","Tuesday","Wednesday","Thursday","Friday"
1100     DATA "Saturday"
1110     DATA "January","February","March","April"
1120     DATA "May","June","July","August","September","October","November"
1130     DATA "December"
1140     DATA .198,.250,.315,.397,.500,.630,.794,1.00,1.26,1.59,2.00,2.52
1150     DATA 3.17,4.0,5.04,6.35,8.00,10.08,12.7,16.0,20.2,25.4,32.0,40.3
1160     DATA 50.8,64.0,80.6,101.6,128,161,203,256,322,406,512,645,812,1024
1170     DATA 31,28,31,30,31,30,31,31,30,31,30,31
1180     FOR I=0 TO 6
1190     READ W$(I)
1200     NEXT I
1210     FOR I=1 TO 12
1220     READ MO$(I)
1230     NEXT I
1240     FOR I=1 TO 38:READ DIA(I):NEXT I
1250     FOR I=1 TO 12:READ MO(I):NEXT I
1260       GOSUB 2480
1270     FOR I=0 TO 32000!:NEXT I
1280 REM

*************** INTRODUCTION ***************

1290 PRINT CHR$(27);"*";
1300 PRINT"  Coulter Counter Data Logging Program  Turbodos Version"
1310 PRINT"  Eliason Data Services  (617) 477-3155    10/25/84"
1320 PRINT
1330 PRINT"Please answer query questions with 'Y' for Yes, 'N' for No, or the
     information  asked for. You may also just hit the 'RETURN' key to accept
     the default reply   which is contained within square brackets, i.e. [ BLA
     ]."
1340 PRINT:PRINT"Please set printer to proper 'Top Of Form' then turn it ON."
1350 PRINT "Set the Coulter Counter PRINT/PLOT switch to INTERFACE."
1360 INPUT"When done, please hit 'RETURN' ";A$
1370 REM Line removed to test relationship to turbo slave 2 crash problem.
1380 REM

************** START THE SAMPLE SEQUENCE ***********

1390 VER =0:SID$="PLEASE ENTER":IGF=-1
1400 GOSUB 2480: REM READ CLOCK
1410 PRINT:PRINT "Operator Name: [";OPN$;:INPUT"] ",A$
1420 IF A$="" THEN 1440
1430 OPN$=A$:PRINT"  Hello, ";OPN$;".  Nice to see you today."
1440 PRINT:PRINT "Requested by:  [";SN$;:INPUT"] ",A$
1450 IF A$="" THEN 1470
1460 SN$=A$
1470 PRINT:PRINT"Cruise I.D.:   [";CID$;:INPUT"] ",A$
1480 IF A$="" THEN 1500
1490 CID$=A$
1500 PRINT:PRINT"Project I.D.:  [";PID$;:INPUT"] ",A$
1510 IF A$="" THEN 1530
1520 PID$=A$
1530 PRINT:PRINT"Sample I.D.:   [";SID$;:INPUT"] ",A$
1540 IF A$="" AND VER=0 THEN 1530
1550 IF A$="" AND VER>0 THEN 1570
1560 SID$=A$
1570 PRINT:PRINT"Lab Number: (format 'AAnnn') [";LN1$;LN2$;:INPUT"] ",A$
1580 IF A$="" THEN 1620
1590 IF LEN(A$)<3 OR LEN (A$)>5 THEN 1570
1600 LN1$=LEFT$(A$,2)
1610 LN2$=MID$(A$,3,3)
1620 IF LEN(LN1$)<> 2 OR LEN (LN2$) >3 OR VAL(LN2$)=0 THEN 1570
1630 PRINT:PRINT "Tube Diameter [";TD;:INPUT "] ",A$:IF A$="" THEN 1660
1640 IF VAL(A$)<>30! AND VAL(A$)<>200! THEN PRINT"Must be 30 or 200":GOTO 1630
1650 TD=VAL(A$)
1660 PRINT:PRINT"Initial Micron Diameter [";DID;:INPUT "] ",A$
1670 IF A$="" AND DID=0 THEN PRINT "Must be entered":GOTO 1660
1680 IF A$="" THEN 1700
1690 DID=VAL(A$)
```

```
1700 IC=0 :OK=0:FOR I=1 TO 38:IF DID=DIA(I) THEN OK=-1:IC=I
1710 NEXT I
1720 IF OK THEN 1760
1730 FOR I=1 TO 38:PRINT USING "####.###  ";DIA(I);:IF I MOD 6=0 THEN PRINT
1740 NEXT I
1750 PRINT:PRINT"Must be one of the above.":GOTO 1660
1760 PRINT:PRINT"Associated Channel Number [";CN;:INPUT"] ",A$
1770 IF A$="" AND CN=0 THEN PRINT "Must be entered":GOTO 1760
1780 IF A$="" THEN 1820
1790 CN=VAL(A$):OK=0 :FOR I=1 TO 16:IF CN=I THEN OK=-1
1800 NEXT I
1810 IF NOT OK THEN PRINT"Must be 1 to 16":GOTO 1760
1820  PRINT:PRINT"Active  Channel  Switch  Setting   example  14-3     [";AC1;"-
     ";AC2;:
1830 INPUT "] ",A$:IF A$="" THEN 1880
1840 B$=MID$(A$,3,1):IF LEN(A$)<4 THEN 1870
1850 IF B$<>" " AND B$<>"-" THEN 1870
1860 AC1=VAL(LEFT$(A$,2)):AC2=VAL(RIGHT$(A$,1)):GOTO 1880
1870 PRINT"Must be entered as NN-N or NN N . Please re-enter":GOTO 1820
1880 OK=0:FOR I=9 TO 16:IF AC1=I AND AC2=17-I THEN OK=-1
1890 NEXT I
1900  IF  NOT  OK  THEN  PRINT"Entry  ";AC1;"-";AC2;"  is  not  valid.  Please  re-
     enter":GOTO 1820
1910  IF  IC<AC1  THEN  PRINT"Initial  diameter  too  small  for  number  of  channels
     selected.":GOTO 1660
1920 REM
DID= Initial Micron Dia.
IC = Pointer to value DID in array DIA
CN = Largest channel to be saved as data
AC1= Number of channels
AC2= Smallest chan to be saved as data
OS = Offset IC-CN
1930  OS=IC-CN:PRINT:PRINT"Data  will  be  recorded  from  channels  ";AC2;"
     to   ";CN
1940   PRINT                                              "corresponding   to
     ";DIA(AC2+OS);"  to  ";DIA(CN+OS);" microns."
1950 IF VER>0 THEN 1970
1960 VER=VER+1:PRINT CHR$(27);"*";"PLEASE VERIFY THE DATA":PRINT:GOTO 1410
1970 PRINT:INPUT"All entries correct? [Y] ",A$
1980 IF A$="" OR A$="y" OR A$="Y" THEN VER=0:GOTO 2000
1990 GOTO 1960
2000 GOSUB 2800:REM PRINT HEADER
2010 REM

*************** GET THE SAMPLES ***************************

2020 IVPC=0
2030  PRINT:PRINT"Waiting  for  sample,  ";OPN$;",  please  press  the  'PRINT/PLOT'
     button when ready."
2040 PRINT CHR$(18);:REM CTRL R FOR BIDIRECTIONAL PRINTER
2050 A$=INPUT$(144):B$=INPUT$(144)
2060 OK=-1:FOR I=1 TO 16
2070     IF MID$(A$,(I-1)*9+3,1)<>MID$(ID$,I,1) THEN OK=0
2080     VP(I)=VAL(MID$(A$,(I-1)*9+4,7))
2090 NEXT I: TPOP=0
2100 FOR I=1 TO 16
2110     IF MID$(B$,(I-1)*9+3,1)<>MID$(ID$,I+16,1) THEN OK=0
2120     PC(I)=VAL(MID$(B$,(I-1)*9+4,7)): TPOP=TPOP+PC(I)
2130 NEXT I
2140 IF OK THEN 2180
2150 PRINT"Data input error. Press the  ESC' key to recover."
2160 A$=INKEY$:IF A$="" THEN 2160
2170 IF ASC(A$)=27 THEN 2030 ELSE 2150
2180 REM

********************** NORMALIZE DATA ***********************

2190 TVP=0
2200 FOR I=AC2 TO CN: TVP=TVP+VP(I): NEXT I
2210 FOR I=AC2 TO CN: VP(I)=(VP(I)/TVP)*100: NEXT I
2220 REM

********************* PRINT THE RESULTS **********************

2230 PRINT:PRINT "Tube Diameter :    ";TD
2240 LPRINT:LPRINT "Tube Diameter :    ";TD
2250 PRINT:PRINT "Micron Dia. - Channel - Volume % - Population
2260 LPRINT:LPRINT "Micron Dia. - Channel - Volume % - Population
2270 LPRINT:PRINT
2280 FOR I=1 TO 16
2290     DD(1,I)=0: DD(2,I)=0: REM ZERO DISK DATA ARRAY
```

```
2300 NEXT I
2310 J=1
2320 FOR I=AC2 TO CN
2330     DD(1,J)=DIA(I+OS): DD(2,J)=VP(I): J=J+1
2340         PRINT  USING  "  ####.###    ";DIA(I+OS);:PRINT  USING  "          ##
     ";I;:PRINT USING "   ####.#    ";VP(I);:PRINT USING "  #######";PC(I)
2350        LPRINT  USING  "  ####.###    ";DIA(I+OS);:LPRINT  USING  "          ##
     ";I;:LPRINT USING "   ####.#    ";VP(I);:LPRINT USING "  #######";PC(I)
2360 NEXT I
2370 PRINT :PRINT SPC(40);"Total Population = ";TPOP
2380 LPRINT:LPRINT SPC(40);"Total Population = ";TPOP
2390 LPRINT CHR$(12);
2400 PRINT:PRINT"S - Save the data    R - Reprint   E - Exit (Data no good)
     [S] ";
2410 INPUT A$:IF A$="" OR A$="s" OR A$ ="S" THEN 2890:REM SAVE TO DISK
2420 IF A$="R" OR A$="r" THEN GOSUB 2800:GOTO 2230
2430 IF A$="e" OR A$="E" THEN 2440 ELSE 2400
2440 REM

 ************ CLEAN UP AND START NEXT *****************

2450 PRINT CHR$(27);"*";
2460 LN1$="PLEASE ":LN2$="ENTER"
2470 GOTO 1390
2480 REM

****************** TURBODOS READ TIME ROUTINE ****************

2490     PRINT CHR$(26): REM CLEAR SCREEN
2500     REM  THIS READS THE DATA
2510     A$="1234" ' ALLOCATE STRING AREA FOR A$
2520     ONE=1 ' GET AROUND ARGUMENT BUG IN BASCOM
2530     CALL BASTOD(ONE,A$)
2540     DAYS=(256*ASC(MID$(A$,2,1))+ASC(LEFT$(A$,1)))
2550     H$=HEX$(ASC(MID$(A$,3,1))) ' HOURS
2560     IF LEN(H$)=1 THEN H$="0"+H$
2570     MI$=HEX$(ASC(MID$(A$,4,1))) ' MINUTES
2580     IF LEN(MI$)=1 THEN MI$="0"+MI$
2590     S$="00"
2600     DAYS=DAYS-1096 ' CETS US TO DAYS PAST DEC.31,1980
2610     W=(DAYS+3) MOD 7 ' JAN 1,1981 WAS A THURSDAY
2620     I=0
2630     I=I+1
2640     Y=365 : IF I MOD 4 = 0 THEN Y=366 ' DAYS PER YEAR
2650     IF DAYS>Y THEN DAYS=DAYS-Y:GOTO 2630
2660     IF I MOD 4 =0 THEN MO(2)=29 ' LEAP YEAR
2670     Y$=RIGHT$(STR$(80+I),2) ' YEAR
2680     I=0
2690     I=I+1
2700     IF DAYS>MO(I) THEN DAYS=DAYS-MO(I):GOTO 2690
2710     MO=I
2720 IF DAYS<10 THEN D$="0"+RIGHT$(STR$(DAYS),1) ER$(DAYS),2)
2730     REM
2740     PRINT CHR$(30);:REM  CURSOR HOME
2750     PRINT W$(W);" ";
2760     PRINT H$;":";MI$;":";S$;" ";
2770     PRINT MO$(MO);" ";D$;", 19";Y$
2780     PRINT CHR$(30)
2790     RETURN
2800 REM

*************** PRINT HEADER INFORMATION ***************

2810 LPRINT "Lab Number ";LN1$;LN2$
2820 LPRINT "Operator: ";OPN$;"   Requested by: ";SN$
2830 LPRINT "Cruise I.D.        ";CI$
2840 LPRINT "Project I.D.       ";PI$
2850 LPRINT "Sample I.D.        ";SI$
2860 LPRINT W$(W);" ";H$;":";MI$;":";S$;" ";MO$(MO);" ";D$;", 19";Y$
2870 LPRINT
2880 RETURN
2890 REM

********************** LOG TO DISK *******************

2900 PRINT"SAVING DATA"
2910 OPEN "R",#1,"CLTR.NDX",6
2920 FIELD #1,5 AS DISC$,1 AS STAT$
2930 GET#1,1: REM READ REC #1
2940 NFILES=VAL(DISC$)
2950 LSET DISC$=STR$(NFILES+1)
```

```
2960 PUT #1,1
2970 LSET DISC$=LN1$+LN2$
2980 LSET STAT$ = "0"
2990 PUT #1,NFILES+2
3000 CLOSE
3010 A$=LN1$+LN2$
3020 OPEN "R",#1,"CLTR.DAT",254
3030 FIELD #1,6 AS QM$,5 AS QL$,3 AS QT$,16 AS QS$,16 AS QP$,16 AS QC$,16 AS
     QR$,16 AS QO$,8 AS QD$,8 AS QH$
3040 FOR I=0 TO 15
3050    FIELD #1,(I*9+110) AS DUMMY$,5 AS QMD$(I),4 AS QVP$(I)
3060 NEXT I
3070 LSET   QM$="     ":LSET   QL$=LN1$+LN2$:LSET   QT$=RIGHT$(STR$(TD),3):LSET
     QS$=SID$:LSET QP$=PID$:LSET QC$=CID$:LSET QR$=SN$:LSET QO$=OPN$
3080 LSET QD$=RIGHT$(STR$(MO),2)+"/"+D$+"/"+Y$:LSET QH$=H$+":"+MI$+":"+S$
3090 FOR I= 1 TO 16
3100    LSET      QMD$(I-1)=RIGHT$(STR$(DD(1,I)),5):             LSET      QVP$(I-
     1)=MID$(STR$(DD(2,I)),2,4)
3110 NEXT I
3120 PUT #1,NFILES+1
3130 CLOSE
3140 REM GO BACK FOR NEXT
3150 GOTO 2440
3160 END:REM


********************************** END OF PROGRAM ********************************

3.  SEDIT

1010 DEFINT I-M
1020 WIDTH LPRINT 255
1030 DIM RLN$(600),CLN$(600)
1040 DIM QMD$(16),QVP$(16),QP$(10),ED$(42)
1050 DIM SL(18)
1060 DATA 6,5,3,16,16,16,16,16,8,8,7,7,7,5,5,5,5,5:REM EDIT MAX $ LENGTHS
1070 FOR I=1 TO 18:READ SL(I):NEXT I

1080 REM ************************ INTRODUCTION ************************************

1090 PRINT CHR$(27);"*";:REM CLEAR + HOME CURSOR
1100 PRINT "  U.S.G.S. Sed Lab  RSA and COULTER data editor                Version
     05/02/84"
1110    PRINT    "           Eliason     Data     Services     (617)     477-3155
                                    01/09/84

1120 PRINT
1130 PRINT "  This program consists of utilities to inspect, edit, and archive
     data which"
1140 PRINT "  have been generated by the RSA and COULTER programs. In most
     cases, you will"
1150 PRINT"  need only to enter the letter or number requested by the program
     queries to"
1160 PRINT "  perform the desired action.  It will NOT be necessary to follow
     these entries";:PRINT "  with the  RETURN' key.  The  RETURN' key is used
     to scroll to the next"
1170 PRINT "  item of a list or to terminate a correction in the EDIT mode."
1180 PRINT :PRINT
1190 PRINT "Do you wish to display and/or edit: RSA Data          (R)"
1200 PRINT "                                    COULTER Data      (C)"
1210 PRINT "                                    Both              (B)"
1220 PRINT "?";
1230 A$=INKEY$:IF A$="" THEN 1230
1240 PRINT A$
1250 RSA=0:CLTR=0:ALL=0
1260 IF A$="r" OR A$="R" THEN RSA=-1:GOTO 1300
1270 IF A$="c" OR A$="C" THEN CLTR=-1:GOTO 1300
1280 IF A$="b" OR A$="B" THEN RSA=-1:CLTR=-1:ALL=-1:GOTO 1300
1290 GOTO 1220:REM invalid entry
1300 PRINT:GOSUB 3110: REM READ INDEX FILES INTO ARRAY
1310 REM

************************** Main Menu ********************************************

1320 PRINT CHR$(27);"*"
1330 PRN=0
1340 PRINT "Display / Edit Menu for ";
1350 IF RSA THEN PRINT "RSA ";
1360 IF ALL THEN PRINT "and ";
1370 IF CLTR THEN PRINT "COULTER ";
1380 PRINT "Data"
```

```
1390 PRINT:PRINT:PRINT
1400 PRINT"      1.        Display Status of logged samples"
1410 PRINT"      2.        Print Status of logged samples"
1420 PRINT"      3.        Assign Master File Number to logged samples"
1430 PRINT"      4.        Edit / Display logged sample data"
1440 PRINT"      5.        Archive Assigned samples to transfer disk"
1450 PRINT"      6.        Begin (Restart program)"
1460 PRINT"      7.        Exit to System (Terminate program)"
1470 PRINT:PRINT"Which number ?";
1480 A$=INKEY$:IF A$="" THEN 1480
1490 PRINT A$
1500 N=VAL(A$):IF N<1 OR N>7 THEN 1470
1510 IF N=1 THEN 1580
1520 IF N=2 THEN PRN=-1:GOTO 1580
1530 IF N=3 THEN 1800:REM  ASSIGN
1540 IF N=4 THEN 1860:REM  EDIT
1550 IF N=5 THEN 4340: REM ARCHIVE
1560 IF N=6 THEN 1080: REM RESTART
1570 IF N=7 THEN CLOSE:SYSTEM:REM RETURN TO SYSTEM
1580 REM

************* DISPLAY OR PRINT STATUS OF LOGGED SAMPLES *****************

1590 FLN$="CLTR":IF RSA THEN FLN$="RSA"
1600 K=1
1610 PRINT CHR$(27);"*"
1620 PRINT "                          STATUS OF LOGGED SAMPLES"
1630 PRINT :PRINT"Type                    Lab Number          Status":PRINT
1640 IF PRN THEN LPRINT "                                             STATUS OF LOGGED SAMPLES"
1650 IF PRN THEN LPRINT:LPRINT"Type                          Lab Number          Status":LPRINT
1660 N=CFILES:IF FLN$="RSA" THEN N=RFILES
1670 IF FLN$="RSA" THEN GOSUB 3290: REM OPEN RSA
1680 IF FLN$="CLTR" THEN GOSUB 3350: REM OPEN CLTR.DAT
1690 FOR I=1 TO N
1700 K=K+1
1710 GOSUB 3430:REM DISPLAY/PRINT Ith record
1720 PRINT
1730  IF NOT PRN AND K MOD 22=0 THEN INPUT"Hit  RETURN' for more or  Q' to
     Quit.",B$
1732 IF B$<>"Q" AND B$<>"q" THEN 1740
1734 B$="": GOTO 1790: REM Quit
1740 NEXT I
1750 CLOSE
1760 IF ALL AND FLN$="RSA" THEN FLN$="CLTR":GOTO 1660
1770 IF PRN THEN LPRINT CHR$(12);
1780 INPUT"End of data. Hit  RETURN' to return to menu. ",B$
1790 GOTO 1320:REM MAIN MENU
1800 REM

*************************** EDIT / ASSIGN ***********************************

1810 REM ***** ASSIGN ENTRY POINT ****
1820 PRINT CHR$(27);"*";
1830 INPUT "Enter new Master File name to be assigned ",MFN$
1840 IF LEN(MFN$)>6 THEN PRINT"NAME TOO LONG. 6 CHARACTERS PLEASE":GOTO 1830
1850 GOTO 1890
1860 REM ***** EDIT ENTRY POINT ******
1870 MFN$="??????"
1880 PRINT CHR$(27);"*";
1890 REM ***** BEGIN *******
1900 PRINT "Edit / Examine Mode                                      Master File Name:
     ";MFN$
1910 PRINT
1920 GOSUB 3580:REM PRINT MENU
1930 PRINT
1940 IF NOT RSA THEN 1980
1950 K=1:N=RFILES:FLN$="RSA":IC=0
1960 GOSUB 3290: REM OPEN RSA.DAT
1970  GOTO 2000
1980 K=1:N=CFILES:FLN$="CLTR":IC=-1
1990 GOSUB 3350: REM OPEN CLTR.DAT
2000 I=K: REM ***** RE-ENTRY POINT *****
2010 PRINT K;".";CHR$(9);:
2020 GOSUB 3430:REM DISPLAY STATUS
2030 PRINT "  ";
2040 A$=INKEY$:IF A$="" THEN 2040
2050 PRINT A$
2060 IF A$="/" OR A$="?" OR A$="H" OR A$="h" THEN PRINT:GOSUB 3580:GOTO 2000
2070 IF A$<>"a" AND A$<>"A" THEN 2140
2080 IF MFN$="??????" THEN PRINT "No Master File name assigned.":GOTO 1830
2090 GOSUB 2280:REM TEST FOR PREVIOUS ASSIGNMENT ETC
```

```
2100 IF NG THEN 2000
2110 IF NOT IC THEN RLN$(K)=LEFT$(RLN$(K),5)+"*"
2120 IF      IC THEN CLN$(K)=LEFT$(CLN$(K),5)+"*"
2130 GOTO 2000
2140 IF A$<>"b" AND A$<>"B" THEN 2170
2150 IF K=1 THEN K=N:GOTO 2000
2160 K=K-1 :GOTO 2000
2170 IF A$<>"d" AND A$<>"D" THEN 2230
2180 GOSUB 2280
2190 IF NG THEN 2000
2200 IF NOT IC THEN RLN$(K)=LEFT$(RLN$(K),5)+"_"
2210 IF      IC THEN CLN$(K)=LEFT$(CLN$(K),5)+"_"
2220 GOTO 2000
2230 IF A$="e" OR A$="E" THEN GOTO 3690
2240 IF A$<>"o" AND A$<>"O" THEN 2410
2250 GOSUB 2280
2260 IF NG THEN 2000
2270 GOTO 2380

2280 REM****************** TEST SUBROUTINE ************************************

2290 IF NOT IC THEN B$=RIGHT$(RLN$(K),1)
2300 IF      IC THEN B$=RIGHT$(CLN$(K),1)
2310 NG=0
2320 IF B$="*" OR B$="_" OR B$="O" THEN 2360
2330 PRINT"WARNING! You are attempting to open previously assigned or deleted
     record."
2340 INPUT "If that's ok, type  YES' ",A$
2350 IF A$<> "YES" AND A$<>"yes" THEN NG=-1
2360 RETURN

2370 REM******************** END SUBROUTINE **********************************


2380 IF NOT IC THEN RLN$(K)=LEFT$(RLN$(K),5)+" "
2390 IF      IC THEN CLN$(K)=LEFT$(CLN$(K),5)+" "
2400 GOTO 2000
2410 IF A$<>CHR$(13) THEN 2470
2420 K=K+1
2430 IF K>N AND ALL AND NOT IC THEN 1980
2440 IF K>N AND ALL AND      IC THEN 1950
2450 IF K>N THEN K=1:GOTO 2000
2460 GOTO 2000
2470 IF A$<>"q" AND A$<>"Q" THEN PRINT " What?":GOTO 2000
2480 PRINT"DOING ASSIGNMENTS..."
2490 REM ***** DO THE ASSIGNMENTS AND DELETIONS *****
2500 CLOSE
2510 IF NOT RSA THEN 2680
2520 GOSUB 3290:REM OPEN RSA.DAT
2530 OPEN "R",#2,"RSA.NDX",6
2540 FIELD #2, 6 AS DISC$
2550 FOR K=1 TO RFILES
2560 A$=LEFT$(RLN$(K),5):B$=RIGHT$(RLN$(K),1)
2570 IF B$="_" THEN RLN$(K)=A$+"D":LSET DISC$=A$+"D":PUT #2,K+1
2580 IF B$="‾" THEN RLN$(K)=A$+"O":LSET DISC$=A$+"O":PUT #2,K+1
2590 IF B$<>"*" THEN 2660
2600 GET #1,K
2610 RLN$(K)=A$+"A"
2620 LSET QM$=MFN$
2630 LSET DISC$=A$+"A"
2640 PUT #1,K
2650 PUT #2,K+1
2660 NEXT K
2670 CLOSE
2680 IF NOT CLTR THEN 1310
2690 GOSUB 3350:REM OPEN CLTR.DAT
2700 OPEN "R",#2,"CLTR.NDX",6
2710 FIELD #2, 6 AS DISC$
2720 FOR K= 1 TO CFILES
2730 A$=LEFT$(CLN$(K),5):B$=RIGHT$(CLN$(K),1)
2740 IF B$="_" THEN CLN$(K)=A$+"D":LSET DISC$=A$+"D":PUT #2,K+1
2750  IF B$="‾" THEN CLN$(K)=A$+"O":LSET DISC$=A$+"O":PUT #2,K+1
2760 IF B$<>"*" THEN 2830
2770 GET #1,K
2780 CLN$(K)=A$+"A"
2790 LSET QM$=MFN$
2800 LSET DISC$=A$+"A"
2810 PUT #1,K
2820 PUT #2,K+1
2830 NEXT K
2840 CLOSE
```

```
2850 GOTO 1310
2860 END
2870 REM

********************* LSET RSA.DAT ******************************************

2880  LSET  QM$=ED$(1):LSET  QL$=ED$(2):LSET  QT$=ED$(3):LSET  QS$=ED$(4):LSET
     QP$=ED$(5):LSET QC$=ED$(6):LSET QR$=ED$(7):LSET QO$=ED$(8)
2890 LSET QD$=ED$(9):LSET QH$=ED$(10)
2900 LSET QSW$=ED$(11):LSET QCW$=ED$(12):LSET QSN$=ED$(13)
2910 FOR I=0 TO 9:LSET QP$(I)=ED$(14+I):NEXT I
2920 RETURN
2930 REM

********************* LSET CLTR.DAT *****************************************

2940  LSET  QM$=ED$(1):LSET  QL$=ED$(2):LSET  QT$=ED$(3):LSET  QS$=ED$(4):LSET
     QP$=ED$(5):LSET QC$=ED$(6):LSET QR$=ED$(7):LSET QO$=ED$(8)
2950 LSET QD$=ED$(9):LSET QH$=ED$(10)
2960 FOR I=0 TO 15
2970      LSET QMD$(I)=ED$(11+I):LSET QVP$(I)=ED$(27+I)
2980 NEXT I
2990 RETURN
3000 REM

******************* FILL ED BUFFER FROM RSA.DAT ***************************

3010 ED$(1)=Q$(3)=QT$:ED$(4)=QS$:ED$(5)=QP$:ED$(6)=QC$:ED$(7)=QR$:ED$(8)=QO$:E
     D$(9)=QD$:ED$(10)=QH$
3020 ED$(11)=QSW$:ED$(12)=QCW$:ED$(13)=QSN$
3030 FOR I=0 TO 9: ED$(14+I)=QP$(I):NEXT I
3040 RETURN
3050 REM

******************** FILL ED BUFFER FROM CLTR.DAT *************************

3060 ED$(1)=QM$:ED$(2)=QL$:ED$(3)=QT$:ED$(4)=QS$:ED$(5)=QP$:ED$(6)=QC$:ED$(7)=
     QR$:ED$(8)=QO$:ED$(9)=QD$:ED$(10)=QH$
3070 FOR I= 0 TO 15
3080      ED$(11+I)=QMD$(I):ED$(27+I)=QVP$(I)
3090 NEXT I
3100 RETURN
3110 REM

********** Open index files and read Lab#'s and Status ********************

3120 PRINT"Reading status files.... Please wait"
3130 FLN$="CLTR":IF RSA THEN FLN$="RSA"
3140 OPEN "R",#1,FLN$+".NDX",6
3150 FIELD #1, 6 AS DISC$
3160 GET #1,1
3170 NFILES=VAL (DISC$)
3180 PRINT NFILES;" ";FLN$;" records found"
3190 PRINT "Reading..."
3200 IF FLN$="RSA" THEN RFILES=NFILES
3210 IF FLN$="CLTR" THEN CFILES=NFILES
3220 FOR I= 2 TO NFILES+1
3230     GET #1,I
3240     IF FLN$="RSA" THEN RLN$(I-1)=DISC$ ELSE CLN$(I-1)=DISC$
3250 NEXT I
3260 CLOSE
3270 IF ALL AND FLN$="RSA" THEN FLN$="CLTR":GOTO 3140
3280 RETURN
3290 REM

********************* OPEN AND FIELD RSA.DAT ********************************

3300 CLOSE#1
3310 OPEN "R",#1,"RSA.DAT",181
3320  FIELD #1,6 AS QM$,5 AS QL$,3 AS QT$,16 AS QS$,16 AS QP$,16 AS QC$,16 AS
     QR$,16 AS QO$,8 AS QD$,8 AS QH$
3330  FIELD #1,110 AS DUMMY$,7 AS QSW$,7 AS QCW$,7 AS QSN$,5 AS QP$(0),5 AS
     QP$(1),5 AS QP$(2),5 AS QP$(3),5 AS QP$(4),5 AS QP$(5),5 AS QP$(6),5 AS
     QP$(7),5 AS QP$(8),5 AS QP$(9)
3340 RETURN
3350 REM

********************* OPEN AND FIELD CLTR.DAT *******************************

3360 CLOSE#1
3370 OPEN "R",#1,"CLTR.DAT",254
```

```
3380 FIELD #1,6 AS QM$,5 AS QL$,3 AS QT$,16 AS QS$,16 AS QP$,16 AS QC$,16 AS
     QR$,16 AS QO$,8 AS QD$,8 AS QH$
3390 FOR I=0 TO 15
3400     FIELD #1,(I*9+110) AS DUMMY$,5 AS QMD$(I),4 AS QVP$(I)
3410 NEXT I
3420 RETURN
3430 REM

*************************DISPLAY/PRINT STATUS ********************************

3440 IF FLN$="RSA" THEN PRINT"RSA                        ";LEFT$(RLN$(I),5);:IF PRN
     THEN LPRINT "RSA                ";LEFT$(RLN$(I),5);
3450 REM ENTER WITH FLN$, I, PRN AND PROPER .DAT OPEN
3460  IF FLN$="CLTR"THEN PRINT"COULTER                   ";LEFT$(CLN$(I),5);:IF PRN
     THEN LPRINT "COULTER         ";LEFT$(CLN$(I),5);
3470 IF FLN$ ="RSA" THEN ST$=RIGHT$(RLN$(I),1)
3480 IF FLN$="CLTR" THEN ST$=RIGHT$(CLN$(I),1)
3490  IF  ST$="O"  THEN  PRINT"                           OPEN";:IF  PRN  THEN  LPRINT
     "             OPEN"
3500  IF  ST$="D"  THEN  PRINT"                        DELETED";:IF  PRN  THEN  LPRINT
     "          DELETED"
3510  IF  ST$="F"  THEN  PRINT"                          FILED";:IF  PRN  THEN  LPRINT
     "             FILED"
3520  IF  ST$="A"  THEN  PRINT"                  ASSIGNED TO ";:IF PRN THEN LPRINT
     "             ASSIGNED TO ";
3530 IF ST$="*" THEN PRINT CHR$(9);CHR$(9);"Tagged for assignment to ";MFN$;
3540 IF ST$="_" THEN PRINT CHR$(9);CHR$(9);"Tagged for deletion";
3550 IF ST$="¯" THEN PRINT "             Tagged for re-openning";
3560 IF ST$="A" THEN GET #1,I:A$=QM$:PRINT A$;:IF PRN THEN LPRINT A$
3570 RETURN
3580 REM

**************************** PRINT EDIT MENU ********************************

3590 PRINT "Commands:     ?          Help              Display this menu"
3600 PRINT "              A          Assign            Assign Master File Name"
3610 PRINT "              B          Back              Backup to previous sample"
3620 PRINT "              D          Delete            Delete this sample"
3630 PRINT "               E          Edit              Edit or Examine DATA for this
     sample
3640 PRINT "              O          Open              Reopen TAGGED sample"
3650 PRINT "              Q          Quit              MUST be done to exit session"
3660 PRINT "              RETURN key                   Moves to next sample"
3670 PRINT
3680 RETURN
3690 REM

********************** EDIT, I SEDIT... **************************************

3700 GET #1,K
3710 IF NOT IC THEN GOSUB 3000: REM FILL ED BUFFER RSA
3720 IF     IC THEN GOSUB 3050: REM                        CLTR
3730 IF NOT IC THEN M=23:KE=18:REM BUFFER SIZE & ED LIMIT RSA
3740 IF     IC THEN M=42:KE=10:REM                             CLTR
3750 PRINT CHR$(27);"*";:REM HOME
3760 PRINT "************ SED LAB EDIT - (the benevolent redeemer) ************"
3770 FOR I=1 TO KE
3780     PRINT I;"        ";ED$(I)
3790 NEXT I
3800 PRINT
3810 PRINT"ENTER:   Line # to edit   P to Print   V to View data   Q to Quit";
3820 INPUT A$:IF A$="" THEN 3810
3840 IF A$<>"P" AND A$<>"p" THEN 3900
3850 LPRINT"SEDIT LISTING      ";FLN$;".DAT    RECORD # ";K
3860 LPRINT
3870 FOR I=1 TO M:LPRINT I;"    ";ED$(I):NEXT I
3880 LPRINT CHR$(12);
3890 GOTO 3750
3900 IF A$<>"V" AND A$<>"v" THEN 4050
3910 PRINT CHR$(27);"*";
3920 PRINT "Data from ";ED$(1);"  ";ED$(2);"  type: ";ED$(3)
3930 PRINT "     ";ED$(4);"       ";ED$(5);"      ";ED$(6)
3940 PRINT "     ";ED$(7);"       ";ED$(8)
3950 PRINT "     ";ED$(9);"       ";ED$(10):PRINT
3960 IK = 23: IF IC THEN IK=26
3970 FOR I=11 TO IK
3980 PRINT"     ";ED$(I);
3990 IF IC THEN PRINT " ";ED$(I+16);
4000 PRINT
4010 NEXT I
4020 PRINT
```

```
4030 INPUT "Hit  RETURN' to return to EDIT mode. ",A$
4040 GOTO 3750
4050 IF A$<>"Q" AND A$<>"q" THEN 4090
4060 PRINT CHR$(27);"*Edit / Examine Mode                    Master File
     Name: ";MFN$:PRINT
4070 GOSUB 3580:REM EDIT MENU
4080 GOTO 2000: REM RE-ENTER EDIT/ASSIGN
4090 IK=VAL (A$)
4100 IF IK<1 OR IK>KE THEN PRINT "What???":GOTO 3810
4110 IF IK=1 AND ED$(1)="          " THEN PRINT "Master File name not assigned.
     Use ASSIGN function.":GOTO 3810
4120 IF IK=3 AND NOT IC THEN PRINT"RSA can not be wrong!":GOTO 3810
4130 PRINT "Change ";ED$(IK);" to what? ";
4140 INPUT A$
4150 IF IK=3 AND A$<>"30" AND A$<>"200" THEN PRINT "Only 30 or 200 acceptable
     here!":GOTO 4130
4160  IF LEN(A$) > SL(IK) THEN PRINT"Entry too long. Must be ";SL(IK);"
     characters maximum.":GOTO 4130
4170 IF IK=2 AND LEN(A$)<5 THEN PRINT "Lab Number must be 5 characters e.g.
     AE345": GOTO 4130
4180 ED$(IK)=A$
4190 IF NOT IC THEN GOSUB 2870: REM SET RSA.DAT
4200 IF     IC THEN GOSUB 2930: REM     SET CLTR>DAT
4210 PUT #1,K
4220 IF IK<>2 THEN GOTO 3750: REM REFRESH SCREEN
4230 REM **** SPECIAL CASE - MUST UPDATE .NDX FILE ****
4240 OPEN "R",#2,FLN$+".NDX",6
4250 FIELD #2, 6 AS DISC$
4260 GET #2,K+1
4270 STI$=RIGHT$(DISC$,1)
4280 LSET DISC$=A$+ST1$
4290 PUT #2,K+1
4300 CLOSE #2
4310 IF NOT IC THEN RLN$(K)=A$+RIGHT$(RLN$(K),1)
4320 IF     IC THEN CLN$(K)=A$+RIGHT$(CLN$(K),1)
4330 GOTO 3750
4340 REM

****************** ARCHIVE (Write assigned data to transfer disk) ************

4350 PRINT CHR$(27);"*";:REM HOME
4360 PRINT"*********************************** ARCHIVE ************************"
4370 PRINT:PRINT"This program segment is used to write ASSIGNED data records
     to the Transfer"
4380  PRINT "disk. This routine must be used with great care due to the
     following "
4390 PRINT "considerations:"
4400 PRINT:PRINT"1.     The old transfer file will be erased, NOT appended, so
     be sure that it"
4410 PRINT"      has been put in the HP computer system."
4420 PRINT"2.    When ARCHIVE finds that all data have been filed or deleted it
     will"
4430 PRINT"      erase the data files on the RSA/COULTER disk."
4440 PRINT"3.     ARCHIVE requires two disks, the RSA/COULTER disk now in use,
     and the"
4450 PRINT"        TRANSFER disk.  DO NOT run this routine if the other disk
     drive is"
4460 PRINT"      in use by another user/program."
4470 PRINT"4.    Please note that ARCHIVE will file all assigned data from BOTH
     the"
4480 PRINT"      RSA and COULTER files."
4490 PRINT:PRINT"If you wish to archive now, place the TRANSFER DISK in the
     available drive and"
4500 PRINT"enter the letter (A or B) which designates the DRIVE YOU PLACED IT
     IN."
4510 PRINT"Hit  RETURN' to start Archive or to exit."
4520 INPUT "Transfer Drive (A or B) ";A$
4530 IF A$="" THEN 1310:REM MAIN MENU
4540 IF A$<>"A" AND A$<>"a" AND A$<>"B" AND A$<>"b" THEN 4520
4550 TDV$=A$
4560 B$="":REM OLD MFN
4570 RSA=-1:CLTR=-1:ALL=-1: REM SETUP FOR BOTH
4580 GOSUB 3120: REM READ LN# & STATUS
4590 OPEN "O",#2,TDV$+":SEDXFER.DAT"
4600 IF RFILES=0 THEN 4830
4610 FOR I=1 TO RFILES
4620 RLN$(I)=RIGHT$(RLN$(I),1)
4630 NEXT I
4640 PRINT "ARCHIVING RSA DATA"
4650 GOSUB 3290:REM OPEN RSA
4660 FOR J=1 TO RFILES
```

```
4670 IF RLN$(J)<>"A" THEN 4810
4680 GET#1,J
4690 GOSUB 3000:REM FILL ED BUFFER
4700 A$=ED$(1)
4710 IF A$<>B$ THEN PRINT#2,"**"+A$:B$=A$:REM PRINT NEW MFN
4720 PRINT#2,">>"+ED$(2)+"   "+ED$(3)
4730 PRINT ED$(2),ED$(3)
4740 FOR I=4 TO 13
4750 IF I=10 THEN I=11:REM SKIP HOUR
4760 PRINT#2,ED$(I)
4770 NEXT I
4780 FOR I=14 TO 21-19;"   "+ED$(I)
4800 NEXT I
4810 NEXT J
4820 CLOSE#1
4830 FOR I=1 TO CFILES
4840 CLN$(I)=RIGHT$(CLN$(I),1)
4850 NEXT I
4860 PRINT "ARCHIVING CLTR DATA"
4870 GOSUB 3350:REM OPEN CLTR
4880 FOR J=1 TO CFILES
4890 IF CLN$(J)<>"A" THEN 5030
4900 GET#1,J
4910 GOSUB 3050: REM FILL ED BUFFER
4920 A$=ED$(1)
4930 IF A$<>B$ THEN PRINT#2,"**"+A$:B$=A$:REM PRINT NEW MFN
4940 PRINT#2,">>"+ED$(2)+"   "+ED$(3)
4950 PRINT ED$(2),ED$(3)
4960 FOR I=4 TO 9
4970 PRINT#2,ED$(I)
4980 NEXT I
4990 FOR I=0 TO 15
5000 IF VAL(ED$(I+11))=0 THEN 5020
5010 PRINT#2,ED$(11+I)+"   "+ED$(27+I)
5020 NEXT I
5030 NEXT J
5040 CLOSE
5050 OPEN "R",#1,"RSA.NDX",6
5060 FIELD #1,5 AS DUMMY$,1 AS DISC$
5070 OPN=0
5080 FOR I=1 TO RFILES
5090 IF RLN$(I)="A" THEN GET#1,I+1:LSET DISC$="F":PUT#1,I+1
5100 IF RLN$(I)="O" THEN OPN=OPN+1
5110 NEXT I
5120 CLOSE
5130 OPEN "R",#2,"CLTR.NDX",6
5140 FIELD #2,5 AS DUMMY$,1 AS DISC$
5150 FOR I=1 TO CFILES
5160 IF CLN$(I)="A" THEN GET#2,I+1:LSET DISC$="F":PUT#2,I+1
5170 IF CLN$(I)="O" THEN OPN=OPN+1
5180 NEXT I
5190 CLOSE
5200 IF OPN>0 THEN 5340
5210 PRINT"All data are now FILED or DELETED. Enter 'K' to kill the temporary"
5220  PRINT"files on the RSA/COULTER disk. Any other key will leave them
     intact."
5230 A$=INKEY$:IF A$="" THEN 5230
5240 IF A$<>"k" AND A$<>"K" THEN 5340
5250 OPEN "R",#1,"RSA.NDX",5
5260 FIELD #1,5 AS DISC$
5270 LSET DISC$="00000":PUT#1,1
5280 CLOSE
5290 OPEN "R",#1,"CLTR.NDX",5
5300 FIELD #1,5 AS DISC$
5310 LSET DISC$="00000":PUT#1,1
5320 CLOSE
5330 OPN=0
5340 PRINT "All assigned samples have been written to the TRANSFER DISK."
5350 PRINT OPN;" samples remain open."
5360 INPUT"Hit 'RETURN' to return to main menu. ",B$
5370 A$="B":GOTO 1280
5380 REM

************************** END OF PROGRAM ***************************************
```

4. Symbol files for grain size analysis program

a. GSASYM file

```
# symbols for Coulter Counter input program
```

```
define(MAXPROJID,   20)
define(MAXOPRNAM,   20)
define(MAXANADAT,   10)
define(MAXSID,      20)
define(MAXDEVICE,    3)
define(MAXAREA,      3)
define(MAXDEPTH,     8)
define(MAXASIZE,    4)
define(MAXDLIST,    38)
define(MAXANS,      80)
define(MAXCVAL,     20)
define(UD200,       14)
define(UD30,        20)
define(PDEFAULT,    8)
define(MAXPRMPT,    40)
define(SUMBASE,      1)
define(SUMMOD,       3)
define(MAXPHI,       7)
define(MAXPHILET,    3)
define(MAXLOCLET,   8)
define(MAXDIRLET,   2)
define(MAXPLIST,   17)
define(MAXSTATS,   999)
define(MAXPHIINT, 34)
define(MAXSEDNAM, 14)
define(MAXCORDLET, 13)
define(NLINE,       11)
define(FIELDLINE,    6)
```

b. JSOSYM file

```
#symbol file for jsort
define(MAXSYSFILES,   7)
define (MAXKEYLET, 6)
define (MAXTYPLET, 4)
define (MAXTYPES,  4)
```

5. Common files for grain size analysis programs

```
#gsacom - common area for id segment of grainsize analysis system
  common /gsacom/ crusid(MAXCRUSID), reqnam(MAXREQNAM), oprnam(MAXOPRNAM),
                  anadat(MAXANADAT), sid(MAXSID), labnum(MAXLABNUM),
                  asize(MAXASIZE), projid(MAXPROJID)
  character crusid # cruise id for sample set
  character reqnam # requestor name for sample set
  character oprnam # operator name for sample set
  character anadat # analysis date for sample set
  character sid #    sample-id or field no. of sample
  character labnum # lab no. assigned to sample by sed lab
  character asize  # aperture diameter or "RSA" to identify analysis subset
  character projid # project id for sample set

    #rsacom - common for rsa which includes the weights & percentage wts of
sample
  common /rsacom/ sampl, coars, sand, fines, gravl,
                  psand, pfines, pgravl, psilt, pclay
     real sampl, coars, sand, fines, gravl,
          psand, pfines, pgravl, psilt, pclay

 #gstcom - common for grain size analysis
common /gstcom/ %
      labnv(MAXLABNUM),
      latdd(MAXCORDLET), londd(MAXCORDLET),
      device(MAXDEVICE), depth(MAXDEPTH),
      dtop(MAXDEPTH), dbotm(MAXDEPTH),
      area(MAXAREA),
      nstats
  character labnv # lab number in nav file
  character latdd, londd # decimal degrees lat, lon
  character device # sampling device
  character depth # water depth at which sample was taken
  character dtop  # depth of sample at top of layer
  character area # area which sample was taken
  character dbotm # depth of sample at bottom of layer
  integer nstats #number of stations read in field-nav file
# chpblk - block data for HP control
   block data chpblk
   include chpdcb
   data maxsys / MAXSYSFILES /
   end

#  io data control block common for HP system.
```

```
#
#   note: since various tools required different no. of input
#   files this segment should be specified for each tool's
#   block data area.
    common /chpdcb/  maxsys, namea

    integer namea (155,MAXSYSFILES)
#                                      HP internal form of file name
#                                        wrd 11 contains access
#                                        wrd 12 on contains DCB
    integer maxsys # maximum no. of files that may be open
```

6. General libraries for grain size analysis programs

a.) GSALIB

```
    #-h- COULTR 1923 asc THU.,  2  SEPT, 1982 14:30:18.17
    # coultr - control for Coulter Counter input
    subroutine coultr
    character ans(MAXANS)
    real dlist(MAXDLIST), slist(MAXDLIST)
    include rsacom
    include gsacom
    integer master, dinit, dend, shead, dcheck
    integer getres, stat
    string ic "Is data OK? "
    string labl "   Dia (u)      %\n \n"
    string labl2 "Enter diameter (u)"
    data dlist / %
        1024.000,  812.000,  645.000,  512.000,  406.000,
         322.000,  256.000,  303.000,  161.000,  128.000,
         101.600,   80.600,   64.000,   50.800,   40.300,
          32.000,   25.400,   20.200,   16.000,   12.700,
          10.080,    8.000,    6.350,    5.040,    4.000,
           3.170,    2.520,    2.000,    1.590,    1.260,
           1.000,     .794,     .630,     .500,     .397,
            .315,     .250,     .198/

        call setup(master)

        while (shead(dinit, dlist) ^ = EOF) {
            call getdat(dinit, dend, dlist, slist,MAXDLIST)
            if (dinit < dend) {
                call sprint(dlist, slist, dinit, dend, STDOUT, labl)
                for (stat = getres(ic,ans,MAXANS,null) ; stat ^ = EOF ;
                        stat = getres(ic,ans,MAXANS,null)) {
                       if (null == YES)
                           next
                       call fold (ans)
                       if (ans(1) == LETY)
                           break
                       if (ans(1) ^ = LETN) {
                           call remark ("Respond with Y or N.")
                           next
                           }
                       if(dcheck (slist, dlist,dinit, dend, labl2) == EOF)
                            break
                       call sprint(dlist,slist,dinit,dend, STDOUT, labl)
                       }
                if (stat == EOF)
                    next
                    call sprint(dlist, slist, dinit, dend, master, labl)
                    }
                else
                    call remark("No sample values!  Sample entry ignored.")
                }

            call close(master)
            call remark("Coultr sample input done.")

        return
        end
    #-h- SETUP 1701 asc WED., 18  AUG., 1982 10:47:14.76
    # setup - initialize Coulter input routine
        subroutine setup(master)
        character file(FILENAMESIZE)
        integer getres, null, open, create, master
        include gsacom

            repeat { # get master file name
                if (getres("raw data master file name:.", file, FILENAMESIZE,
            null) == EOF)
            call error("EOF for file name.")
         if (null == YES)
            call remark("null file name invalid.")
         else {
            master = open(file, APPEND)
            if (master == ERR) {
               master = create(file, WRITE)
               if (master == ERR)
                  call cant(file)
               call remark("starting new file.")
               }
            else
               call remark("appending to existing file.")
            }
         }
      until (null ^ = YES)

      if (getres ("Project Id:  .", projid, MAXPROJID, null) == EOF)
         call error ("EOF entering Project Id.")
         if (getres("Cruise Id:  .", crusid,MAXCRUSID, null) == EOF)
            call error("EOF on Cruise or Project ID specification.")
      repeat  # get requestor´s name
         if (getres("Requestor:  .", reqnam, MAXREQNAM,
             null) ==EOF)
            call error("EOF on requestor´s name request.")
      until (null ^ = YES)

      repeat  # get operator´s name
         if (getres("operator´s name:.", oprnam, MAXOPRNAM,
            null) == EOF)
            call error("EOF on operator´s name.")
      until (null ^ = YES)

      repeat  # get analysis date
         if (getres("analysis date (mo/da/yr): .", anadat, MAXANADAT,
            null) == EOF)
            call error("EOF for analysis date.")
      until (null ^ = YES)
   return
   end
#-h- GETDAT 2773 asc TUE., 14  SEPT, 1982 11:7:19.44
# getdat - get sample data
   subroutine getdat(dinit, dend, dlist  slist,maxlst)
   character prmpt(MAXPRMPT), cval(MAXCVAL)
   integer dinit, dend, len, dtofc, dtofc, getres
   integer maxlst
   real slist(maxlst), dlist(maxlst), sum
   double precision dble, ctodp
   include rsacom
   string colon " : "

      for (i = 1 ; i <= maxlst ; i = i + 1)
         slist(i) = 0.0
      sum = 0.0
      for (dend = dinit; dend <= maxlst; dend = dend + 1) {
         if (dend == 8)
           call remark ("   Relative %´s SAND .")
         else if (dend == 13 & sum ^ = 0.0) {
            if (sum <= 99.9 | sum >= 100.1) {
               call remark ("Sum of relative %´s for sand must = 100.")
               dend = dinit - 1
               sum = 0.0
               next
               }
            sum = 0.0
            call remark ("   Relative %´s GRAVEL .")
            }
         len = dtofc(dble(dlist(dend)), LETF, 10, 3, prmpt) + 1
         call scopy(colon, 1, prmpt, 11)
         if (getres(prmpt, cval, MAXCVAL, null) == EOF)
            break
         if (null == YES) {
            dend = dend - 1
            next
            }
         if (cval(1) == PERIOD & cval(2) == EOS ) {
            if (dinit >= 14)  #coulter data input (no check for 100%)
               break
            if (sum >= 99.9 & sum <= 100.1) {
               gravl = coars - sand
               if (gravl < 0.009)
```

```
                  break
               if (dend > 13)
                  break
               sum = 0.0
               dend = 12
               call remark ("    Relative % Gravel. ")
               next
               }
            if (dend >= 13) {
               call remark ("Relative %'s for Gravel must sum to 100.")
               dend = 12
               sum = 0.0
               next
               }
            else {
               call remark ("Relative %'s for Sand must sum to 100.")
               dend = dinit - 1
               sum = 0.0
               next
               }
            }
         i = 1
         v = ctodp(cval, i)
         if (i == 1) {
            call remark("invalid entry.")
            dend = dend - 1
            }
         else {
            slist(dend) = v
            sum = sum + slist(dend)
            if (dend == maxlst) {
               if (dinit >= 14) {
                  call remark ("Maximum u diameter reached.")
                  dend = dend + 1
                  break
                  }
               if (sum >= 99.9 & sum <= 100.1) {
                  dend = dend + 1
                  break
                  }
               call remark ("Max phi range reached before sum %'s = 100.")
               dend = 12
               sum = 0.0
               }
            }
      }
      dend = dend - 1

   return
   end
#-h- SPRINT 2205 asc TUE., 30  NOV., 1982 12:21:18.29
# sprint - print Coulter summary
   subroutine sprint(dlist, slist, dinit, dend, out, labl)
   character line(MAXLINE)
   real dlist(ARB), slist(ARB)
   include rsacom
   include gsacom
   integer i, dinit, dend, ip, dtofc, tty, ptty, out, ndp
   integer putlin, putch
   double precision dble
        string blanks "  ", tag ">>",
           head "   Dia (u)    %\n \n"

           ptty = tty(out)
           if (out == STDOUT)
              ptty = YES
           if (ptty == YES) {
              call putlin(" Data Summary.", out)
              call putch(NEWLINE, out)
              call putch(NEWLINE, out)
              call putlin("Lab Number:   .", out)
              }
           else
              call putlin(tag, out)

           call putlin(labnum, out)
            call putlin(blanks, out)
           call putlin(asize, out)
           call putch(NEWLINE, out)
           if (ptty == NO) {
```

```
         call putlin(blanks, out)
         call putlin(sid, out)
         call putch (NEWLINE, out)
         call putlin (blanks, out)
         call putlin (projid, out)
         call putch (NEWLINE, out)
         call putlin (blanks, out)
         call putlin (crusid, out)
         call putch (NEWLINE, out)
         call putlin(blanks, out)
         call putlin(reqnam, out)
         call putch(NEWLINE, out)
         call putlin(blanks, out)
         call putlin(oprnam, out)
         call putch(NEWLINE, out)
         call putlin(blanks, out)
         call putlin(anadat, out)
         call putch(NEWLINE, out)
         }

      if (asize(1) == BIGR) {
          ndp = 4
          if (ptty == YES)
           call putlin ("Net sample wt:   .",out)
           call mout (sampl, line, out, ndp)
          if (ptty == YES)
           call putlin ("Net Coarse: .",out)
           call mout (coars, line, out, ndp)
          if (ptty == YES)
           call putlin ("Net Sand:    .",out)
           call mout (sand, line, out, ndp)

           }
      if (ptty == YES)
          call putlin (labl, out)
      for (i = dend; i >= dinit; i = i - 1) {
         call dtofc(dble(dlist(i)), LETF, 10, 3, line)
         call dtofc(dble(slist(i)), LETF,  8, 2, line(11))
         ip = 19
         line(ip) = NEWLINE
         line(ip+1) = EOS
         call putlin(line, out)
         }

   return
   end
#-h- DCHECK 1193 asc THU., 16  SEPT, 1982 13:32:37.20
#dcheck - to enter values to correct input to coultr master file
   integer function dcheck(slist, dlist, dinit, dend, labl2)
   character ans(MAXLINE)
   real dlist(ARB), slist(ARB), dval, sval, sngl
   integer dl, getlin, prompt, sl, iptr
   integer dinit, dend
   integer null
   double precision dreal

    for (sl = 1 ; sl < MAXDLIST ; sl = sl + 1) {
       call prompt (labl2,ERROUT)
       call prompt (", new value: .",ERROUT)
       if(getlin (ans,STDIN) == EOF) {
          dcheck = EOF
           break
           }

       if (ans(1) == PERIOD & ans(2) == NEWLINE)
           brea

       iptr = 1
       dval = sngl(dreal(ans,iptr,UD200,null))
       if (null == YES)
           break
       for (dl = 1 ; dl <= MAXDLIST ; dl = dl + 1) {
          if (dval == dlist(dl))
             break
          }
       if (dval ~ = dlist(dl)) {
          call remark("diameter specified not valid.")
          next
          }
       if (dl < dinit)
          dinit = dl
```

```
              if (dl > dend)
                 dend = dl
              sval = sngl(dreal(ans,iptr,0.0,null))
              if (null == YES)
                 call remark("invalid entry .")
               else
                  slist(dl) = sval

       }
   return
   end
#-h- LINCLR 188 asc THU., 10  JUNE, 1982 13:1:53.37
#clear - to clear line array
  subroutine clear (line, ichar)
  character line(ARB)
  integer ichar, i

  for (i = 1 ; i <= ichar; i = i + 1) {
     line(i) = BLANK
     }

  return
  end
#-h- MOUT 361 asc MON., 16  AUG., 1982 11:7:54.47
 #mout - to copy percent of gsa for sprint
  subroutine mout (value, line, out, ndp)
   character line(ARB)
   integer out, dtofc, putlin, putch, ndp
   real value
   double precision dble
   string blanks "  "
   call putlin (blanks, out)
   call dtofc (dble(value), LETF, 10,ndp, line)
   call putlin (line, out)
   call putch (NEWLINE, out)
   return
   end

   b.) RSALIB

   #-h- RSA 1908 asc TUE., 30  NOV., 1982 12:3:38.35
   # rsa - control for coarse grain data input
   subroutine rsa
   character ans(MAXANS), rsize(MAXASIZE)
   real plist(MAXPLIST), slist(MAXPLIST), blist(MAXDLIST), clist(MAXDLIST)
   include rsacom
   include gsacom
   integer master, pinit, pend, chead, dcheck, dinit, dend, ifs, null
   integer getres, stat, pinit, pend, err, gdats, getdat, iptr
   string ic "Is data OK? "
   string labl "       PHI          Z\n \n"
   string labl2 "Enter Phi"
   data plist / Z
        11.0, 10.0, 9.0, 8.0, 7.0, 6.0, 5.0, 4.0, 3.0, 2.0,
       1.0, 0.0, -1.0, -2.0, -3.0, -4.0, -5.0 /

   data rsize(1) /BIGR/,
        rsize(2) /BIGS/,
        rsize(3) /BIGA/,
        rsize(4) /EOS /

      iptr = 1

   call scopy (rsize,iptr, asize,iptr)
   call setup(master)
   pinit = 8
   while (chead(ans) ^ = EOF) {
      call getwt (sampl, coars, sand)
      call getdat(pinit, pend, plist, slist, MAXPLIST)
      if (pinit <= pend) {
         call sprint(plist, slist, pinit, pend, ERROUT, labl)
         for (stat = getres(ic,ans,MAXANS,null) ; stat  = EOF ;
               stat = getres(ic,ans,MAXANS,null)) {
              if (null == YES)
                 next
              call fold (ans)
              if (ans(1) == LETY)
                 break
              if (ans(1) ^ = LETN) {
                 call remark ("Respond with Y or N.")
                 next
                 }
```

```
                  if(dcheck (slist, plist,pinit, pend, labl2) == EOF)
                      break
                 call sprint(plist,slist,pinit,pend, ERROUT, labl)
                  }
            if (stat == EOF)
               next
            call sprint(plist, slist, pinit, pend, master, labl)
            }
         else
            call remark("No sample values!  Sample entry ignored.")
         }

      call close(master)
      call remark("RSA sample input done.")

   return
   end
#-h- GETV 910 asc TUE., 15  JUNE, 1982 16:20:45.25
 #getv - retrieves diameter apert, %vol stores in proper array
  subroutine getv (ans, alist)
  character ans(ARB)
  real alist(ARB), dlist(MAXDLIST), adlist, sreal
  integer iptr, i

   data dlist / %
      1024.000,  812.000,  645.000,  512.000,  406.000,
       322.000,  256.000,  303.000,  161.000,  128.000,
       101.600,   80.600,   64.000,   50.800,   40.300,
        32.000,   25.400,   20.200,   16.000,   12.700,
        10.080,    8.000,    6.350,    5.040,    4.000,
         3.170,    2.520,    2.000,    1.590,    1.260,
         1.000,     .794,     .630,     .500,     .397,
          .315,     .250,     .198/

  iptr = 1

  adlist = sreal (ans, iptr, 0.0, null)
  if (null == YES) {
     call remark ("Null field .")
       return
     }
  for (i = 1 ; i <= MAXDLIST ; i = i + 1 ) {
     if (adlist == dlist(i))
       break
     }
  alist(i) = sreal (ans, iptr, 0.0, null)
  return
  end
#-h- MSETUP 822 asc TUE., 10  AUG., 1982 12:53:56.62
 #msetup - to open or create master file for grain size analysis
   integer function msetup(amstr, mstr)
  character file(FILENAMESIZE), line(MAXLINE)
  integer amstr, mstr #amstr = flag & mstr = fid
  integer getres, null, open, create, iptr

  if (getres("Temp output filename:  .", file, FILENAMESIZE, null) == EOF) {
     msetup = ERR
     return
     }
  if (file(1) == PERIOD) {
     msetup = ERR
     return
     }
  else if (null == YES) {
     amstr = NO
     return
     }
  amstr = YES
  mstr  = open(file,APPEND)
  if (mstr == ERR) {
     mstr = create (file, WRITE)
     if (mstr == ERR) {
        call cant(file)
        msetup = ERR
        return
        }
     call remark ("Starting New Temp out File.")
     }
  else
     call remark("Adding to existing temp out file.")
  msetup = GOOD
```

```
    return
    end
#-h- GETWT 1349 asc TUE., 17  AUG., 1982 12:1:56.85
#getwt - input routine for rsa sample weights
     subroutine getwt (sampl, coars, sand)
     character ans(MAXANS)
     real sampl, coars, sand, v, sngl
     integer getres, i, null
   double precision dble, ctodp

   repeat {
   if (getres("Net sample weight. ", ans, BLANK , null) == EOF |
       null == YES) {
           call remark ("No sample weight entered.")
           null = YES
           next
           }
   i = 1
   v = sngl(ctodp(ans,i))
   if (i == 1) {
       call remark ("Invalid entry.")
       null = YES
       next
       }
   else
       sampl = v

  # get net coarse weight of sample
   if (getres("Net coarse weight. ", ans, BLANK, null) == EOF |
       null == YES) {
           call remark ("No coarse weight entered for sample.")
           null = YES
           next
           }
   i = 1
   v = sngl(ctodp(ans,i))
   if (i == 1) {
       call remark ("Invalid entry.")
       null = YES
       next
       }
   else
       coars = v
  # enter net sand weight of sample
   if (getres("Net sand weight .", ans, BLANK, null) == EOF |
       null == YES) {
       call remark ("No sand weight entered for sample.")
       null = YES
       next
       }
   i = 1
   v = sngl(ctodp(ans,i))
   if (i == 1) {
       call remark ("Invalid entry.")
       null = YES
       next
       }
   else
       sand = v
   }
   until (null ^ = YES)
   return
   end

#-h- CHEAD 1103 asc WED.,  2  FEB., 1983 9:35:6.59
 # chead - select station id for input to capr
  integer function chead(ans)
   character ans(ARB)
   integer err, null
   include gaacom
   integer getres
   repeat {
      chead = getres("Lab Number: .", ans, MAXLABNUM,null)
      if (chead == EOF)
          break
      if (ans(1) == PERIOD) {
          chead = EOF
          break
          }
      if (null == YES)
          call remark ("No Lab number entered - Enter PERIOD to quit.")
      else {
          iptr = 1
          call scopy (ans, iptr, labnum, iptr)
          repeat {
             chead = getres("Sample Id:  .", ans, MAXSID, null)
             if (chead == EOF)
                break 2
             if (ans(1) == PERIOD & ans (2) == EOS) {
                chead = EOF
                break 2
                }
             if (null == YES)
                call remark ("A sample id or field number must be entered.")
             else {
                iptr = 1
                call scopy (ans, iptr, sid, iptr)
                }
             }
          until (null ^ = YES)
          }
      }
   until (null ^ = YES)

   return
   end
```

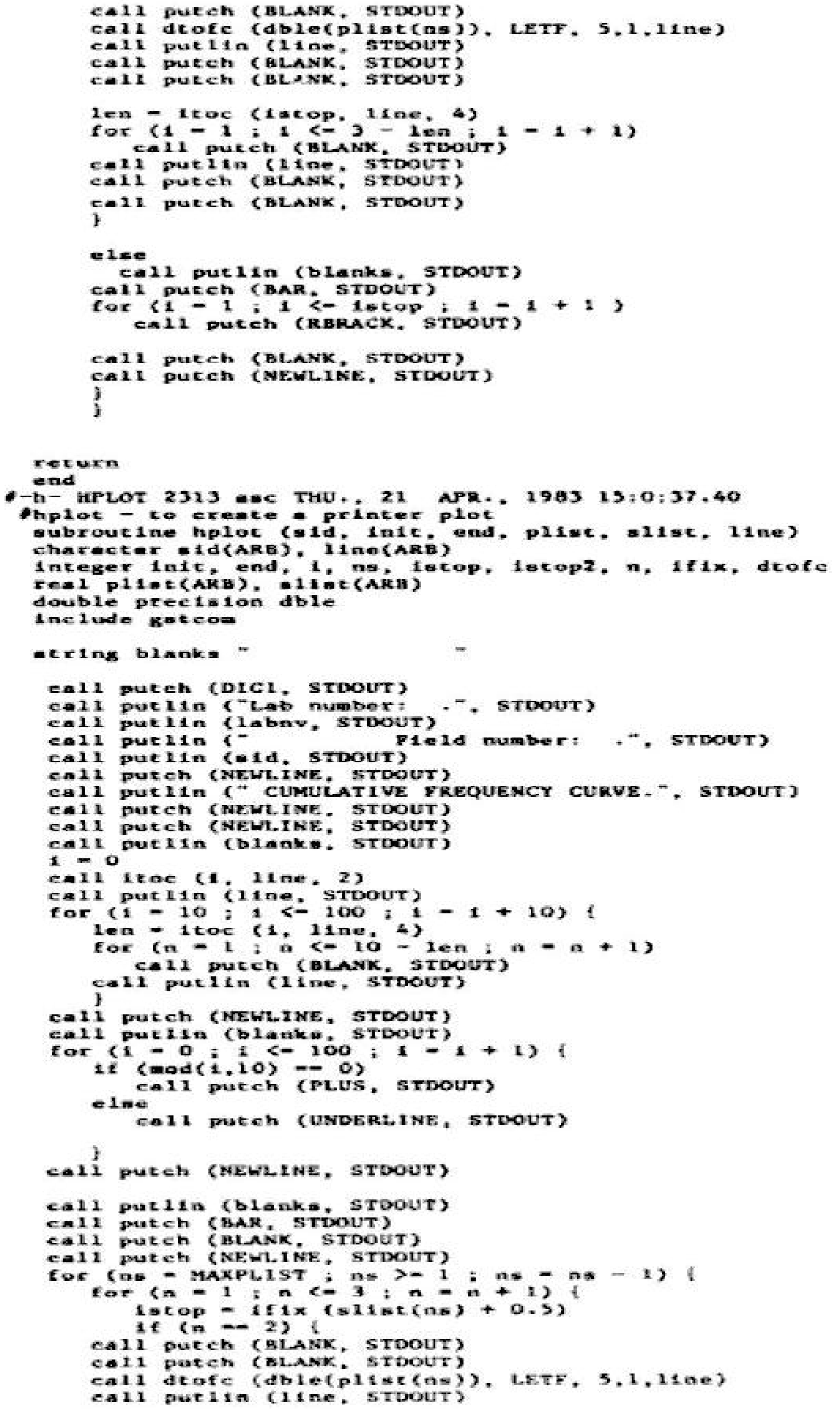
c.) GSTLIB

```
#-h- RANK 495 asc MON., 2 AUG., 1982 22:28:50.41
  subroutine rank (ival, out)
  integer ival, out
  string first "First",
         second "Second",
         third "Third"
         fourth "Fourth ",
         fifth "Fifth "
   switchto ival {
      call putlin (first, out)
      call putlin (second, out)
      call putlin (third, out)
      call putlin (fourth, out)
      call putlin (fifth, out)
      }
   else {
     call remark ("Value out of range for ranking routine.")
     call putlin ("            .", out)
     }
   return
   end
#-h- STCHK 418 asc WED., 10  NOV., 1982 11:37:12.72
#stchk - checks if station for gsa is in stlist of processed data
   integer function stchk (sid, stlist, is)
   character sid(ARB), stlist(ARB,ARB), ifsid(MAXLABNUM)
   integer is, i, equal
   for (i = 1 ; i <= is ; i = i + 1) {
      iptr = (i-1)*MAXLABNUM + 1
      jptr = 1
      call scopy (stlist, iptr, ifsid, jptr)
      stchk = equal(sid,ifsid)
      if (stchk == YES)
         break
      }
   return
   end
#-h- SEDCLS 2439 asc MON.,  8  AUG., 1983 14:51:33.86
  #sedcls - determines class for gsa according to Shepard's classification
  subroutine sedcls (name)
  character name (ARB)
  real sansil, clysnd, silcly
  integer iptr, jptr, scopy
  include rsacom

  string gravel "GRAVEL > 10%",
         ssand   "SAND          ",
         silt   "SILT          ",
         clay   "CLAY          ",
         saclay "SANDY CLAY    ",
         siclay "SILTY CLAY    ",
         clsilt "CLAYEY SILT   ",
         sasilt "SANDY SILT    "
```

```
          sisand "SILTY SAND  ",
          clsand "CLAYEY SAND ",
          ssclay "SAN SIL CLAY"
   iptr = 1
   jptr = 1

   if (pgravl > 10.0) #Shepard's classification doesn't apply
      call scopy (gravel, iptr, name, jptr)
    else {
       if (psand == 0.0)
          psand = 0.001
       if (psilt == 0.0)
          psilt = 0.001
       if (pclay == 0.0)
          pclay = 0.001
      sand = psand + pgravl
       if (sand >= 75.0) #sample is sand
           call scopy (ssand,jptr,name,iptr)
       else if (psilt >= 75.0) # sample is silt
           call scopy (silt,jptr,name,iptr)
       else if (pclay >= 75.0) # sample is clay
           call scopy (clay,jptr,name,iptr)
       else    # sample is combination of sand, silt and/or clay
           sansil = sand/psilt
           clysnd = pclay/sand
           silcly = psilt/pclay
           if (sand <=20.0) {
              if (sansil > 1.0) # sample is sandy clay
                 call scopy (saclay,jptr,name,iptr)
              else if (silcly < 1.0) # sample is silty sand
                 call scopy (siclay,jptr,name,iptr)
              else if (clysnd > 1.0) # sample is clayey silt
                 call scopy (clsilt,jptr,name,iptr)
              else # sample is sandy silt
                 call scopy (sasilt,jptr,name,iptr)
              }
           else if (pclay <= 20.0) {
              if (sansil < 1.0) # sample is sandy silt
                 call scopy (sasilt,jptr,name,iptr)
              else if (silcly > 1.0) # sample is silty sand
                  call scopy (sisand,jptr,name,iptr)
              else # sample is clayey sand
                  call scopy (clsand,jptr,name,iptr)
              }
           else if (psilt <= 20.0) {
              if (clysnd <= 1.0) # sample is a clayey sand
                  call scopy (clsand,jptr,name,iptr)
              else # sample is a sandy clay
                  call scopy (saclay,jptr,name,iptr)
              }
           else # sample is a sandy silty clay
              call scopy (ssclay,jptr,name,iptr)
           }
      }
    return
    end
#-h- MOMNTS 831 asc MON.,  2  AUG., 1982 22:33:17.09
     #calculate moments about the mean
     subroutine momnts (ip, empl, dphi, f, en1, zm2, zm3, zm4)
     real f(ARB), empl, dphi, en1, en2, en3, en4, xi, ct
     real sum, sum1, sum2, sum3, sum4, zm2, zm3, zm4
     integer ip, i
     sum = 0.0
     sum1 = 0.0
     sum2 = 0.0
     sum3 = 0.0
     sum4 = 0.0
     for (i = 1 ; i <= ip ; i = i + 1) {
        xi = i - 1
        ct = empl + xi*dphi
        sum = sum + f(i)
        sum1 = sum1 + f(i)*ct
        sum2 = sum2 + f(i)*ct**2
        sum3 = sum3 + f(i)*ct**3
        sum4 = sum4 + f(i)*ct**4
        }
     en1 = sum1/sum
     en2 = sum2/sum
     en3 = sum3/sum
     en4 = sum4/sum
```

```
      zm2 = en2 - en1**2
      zm3 = en3 - 3.0*en2*en1 + 2*en1**3
      zm4 = en4 + en1*(-4.0*en3 + 6.0*en1*en2 - 3.0*en1**3)
      dphi2 = dphi*dphi
      zm4 = zm4 - 0.5*dphi2*zm2 + 0.02916667*dphi2*dphi2
      zm2 = zm2 - dphi2/12.0
      return
      end
#-h- MEDIAN 368 asc MON.,  2  AUG., 1982 22:33:24.45
         #calculate median
    subroutine median (f, ip, dphi, empl, ctmed)
    real f(ARB), ctmed, empl, dphi
    real sum, xi, ct
    integer ip, i

      sum = 0.0
      for  (i = 1 ; i <= ip ; i = i + 1) {
          sum = f(i) + sum
          if (sum >= 50.0)
              break
          }
      xi = i - 1
      ct = empl + xi*dphi
      ctmed = ct - (sum - 50.0)*dphi/f(i)  + 0.5*dphi

      return
      end
#-h- GPRINT 3382 asc MON., 22  NOV., 1982 13:49:23.20
    #gprint - to list gsadata for pre-grasp file
    subroutine gprint (fidg, plist, elist, line,
                 name, median, mean, stdev, skew, kurt, o, os, nmodes,inv)
    character junk(MAXSID), line(ARB), name(ARB)
    real plist(ARB), elist(ARB), o(ARB), os(ARB)
    real median, mean, stdev, skew, kurt
    integer fidg, ptr, i, putlin, putch, dtofc, nmodes, index, itoc, len,
            n, sepr, inv
     double precision dble
    include rsacom
    include gsacom
    include gstcom


    call putlin (labnum, fidg)
    call putch (COMMA, fidg)
    call putlin (sid, fidg)
    call putch (COMMA, fidg)
    call putlin (projid,fidg)
    call putch (COMMA,fidg)
    call putlin (crusid, fidg)
    call putch (COMMA, fidg)
    call putlin (reqnam, fidg)
    call putch (COMMA, fidg)

    sepr = SLASH
    n = 0
    for (i = 1 ; i <= 2 ; i = i + 1) {
        ptr = index (anadat, sepr)
        if (ptr <= 0) {
            n = n + 1
            if (n >= 2) {
                call remark ("invalid date in sample.")
                break
                }
            sepr = DASH
            i = 0
            next
            }
        anadat(ptr) = COMMA
        }
    call putlin (anadat, fidg)
    call putch (COMMA, fidg)
    call putlin (latdd, fidg)
    call putch (COMMA, fidg)
    call putlin (londd, fidg)
    call putch (COMMA, fidg)
    call putlin (device, fidg)
    call putch (COMMA, fidg)
    call putlin (area, fidg)
    call putch (COMMA, fidg)
    call putlin (depth, fidg)
    call putch (COMMA, fidg)
call putlin (dtop, fidg)
call putch (COMMA, fidg)
call putlin (dbotm, fidg)
call putch (COMMA, fidg)
call dtofc (dble(sampl), LETF, 8, 2, line)
call putlin (line, fidg)
call putch (COMMA, fidg)
call dtofc (dble(psand), LETF, 8, 2, line)
call putlin (line, fidg)
call putch (COMMA, fidg)
call dtofc (dble(pgravl), LETF, 8, 2, line)
call putlin (line, fidg)
call putch (COMMA, fidg)
call dtofc (dble(psilt), LETF, 8, 2, line)
call putlin (line, fidg)
call putch (COMMA, fidg)
call dtofc (dble(pclay), LETF, 8, 2, line)
call putlin (line, fidg)
call putch (COMMA, fidg)
call putch (NEWLINE, fidg)
call putlin (name, fidg)
call putch (COMMA, fidg)
call dtofc (dble(median), LETF, 8, 2, line)
call putlin (line, fidg)
call putch (COMMA, fidg)

call dtofc (dble(mean), LETF, 8, 2, line)
call putlin (line, fidg)
call putch (COMMA, fidg)
call dtofc (dble(stdev), LETF, 8, 2, line)
call putlin (line, fidg)
call putch (COMMA, fidg)
call dtofc (dble(skew), LETF, 8, 2, line)
call putlin (line, fidg)
call putch (COMMA, fidg)
call dtofc (dble(kurt), LETF, 8, 2, line)
call putlin  (line, fidg)
call putch (COMMA, fidg)
for (i = 1 ; i <= 3 ; i = i + 1) { #output modal class1-3
    call dtofc (dble(o(i)), LETF, 8, 2, line)
    call putlin (line, fidg)
    call putch (COMMA, fidg)
    call dtofc (dble(os(i)), LETF, 8, 2, line)
    call putlin (line, fidg)
    call putch (COMMA, fidg)
    }

len = itoc (nmodes, line, 2)
call putlin (line, fidg)
call putch (COMMA, fidg)
call putch (NEWLINE, fidg)

for (i = 1 ; i <= MAXPLIST ; i = i + 1) {
    call dtofc (dble(plist(i)), LETF, 5, 2, line)
    call putlin (line, fidg)
    call putch (COMMA, fidg)
    call dtofc (dble(slist(i)), LETF, 6, 2, line)
    call putlin (line, fidg)
    call putch (COMMA, fidg)
    if (i == 8 )
        call putch (NEWLINE, fidg)
    }
   call putch (DOLLAR, fidg)
   call putch (NEWLINE, fidg)

   return
   end
 #-h- MODE 1100 asc MON.,  2  AUG., 1982 22:35:39.89
    #calculate mode of sample
    subroutine mode (os, o, f, empl, dphi, ip)
    real os(ARB), o(ARB), f(ARB)
    real dphi, del1, del2, xi, xem, empl, xf, xo
    integer ip, i, jih, jh, mono, lim, ih
    for (ih = 1 ; ih <= 5 ; ih = ih + 1) {
       o(ih) = 0.0
       os(ih) = 0.0
       }
    ih = 1
    del1 = 0.10
    for (i = 2 ; i <= ip ; i = i + 1) {
       del2 = f(i) - f(i-1)
```

```
     if ((del2*del1) < 0.0 & del2 < 0.0 ) {
        if ( (f(i-1) - 5.0*dphi) > 0.0) {
           xi = i - 2
           xem = empl + xi*dphi
           xf = f(i-1)
           i = i + 1
           o(ih) = xem
           os(ih) = xf
           ih = ih + 1
           }
        }
     del1 = del2
     }
     #sort modes in order of decreasing strength
     mono = ih - 1
     lim = mono - 1
     for (ih = 1 ; ih <= lim ; ih = ih + 1) {
        jih = mono - ih
        for (jh = 1 ; jh <= jih ; jh = jh + 1) {
           if ( (os(jh) - os(jih+1)) < 0.0) {
              xo = os(jih+1)
              os(jih+1) = os(jh)
              os(jh) = xo
              xo = o[jih+1]
              o[jih+1] = o[jh]
              o[jh] = xo
              }
           }
        }

    return
    end
#-h- PPLOT 2127 asc THU., 21  APR., 1983 15:0:11.95
 #pplot - to create a printer plot
  subroutine pplot (sid, init, end, plist, slist, line)
  character sid(ARB), line(ARB)
  integer init, end, i, ns, istop, istop2, n, ifix, dtofc
  real plist(ARB), slist(ARB)
  double precision dble
  include gstcom

  string blanks "                              "

  call putch (DIG1, STDOUT)
  call putlin ("Lab number:   .", STDOUT)
  call putlin (labnv, STDOUT)
  call putlin ("            Field number:   .", STDOUT)
  call putlin (sid, STDOUT)
  call putch (NEWLINE, STDOUT)
   call putlin (" HISTOGRAM.", STDOUT)
   call putch (NEWLINE, STDOUT)
   call putch (NEWLINE, STDOUT)
   call putlin (blanks, STDOUT)
   i = 0
   call itoc (i, line, 2)
   call putlin (line, STDOUT)
   for (i = 10 ; i <= 100 ; i = i + 10) {
      len = itoc (i, line, 4)
      for (n = 1 ; n <= 10 - len ; n = n + 1)
         call putch (BLANK, STDOUT)
      call putlin (line, STDOUT)
      }
   call putch (NEWLINE, STDOUT)
   call putlin (blanks, STDOUT)
   for (i = 0 ; i <= 100 ; i = i + 1) {
      if (mod(i,10) == 0)
         call putch (PLUS, STDOUT)
      else
         call putch (UNDERLINE, STDOUT)
      }
   call putch (NEWLINE, STDOUT)

   call putlin (blanks, STDOUT)
   call putch (BAR, STDOUT)
   call putch (BLANK, STDOUT)
   call putch (NEWLINE, STDOUT)
   for (ns = MAXPLIST : ns >= 1 ; ns = ns - 1) {
      for (n = 1 ; n <= 3 ; n = n + 1) {
         istop = ifix (slist(ns) + 0.5)
         if (n == 2) {
      call putch (BLANK, STDOUT)
```



```
      call putch (BLANK, STDOUT)
      call dtofc (dble(plist(ns)), LETF, 5,1,line)
      call putlin (line, STDOUT)
      call putch (BLANK, STDOUT)
      call putch (BLANK, STDOUT)

      len = itoc (istop, line, 4)
      for (i = 1 ; i <= 3 - len ; i = i + 1)
         call putch (BLANK, STDOUT)
      call putlin (line, STDOUT)
      call putch (BLANK, STDOUT)
      call putch (BLANK, STDOUT)
      }

      else
        call putlin (blanks, STDOUT)
      call putch (BAR, STDOUT)
      for (i = 1 ; i <= istop ; i = i + 1 )
         call putch (RBRACK, STDOUT)

      call putch (BLANK, STDOUT)
      call putch (NEWLINE, STDOUT)
      }
      }


  return
  end
#-h- HPLOT 2313 asc THU., 21  APR., 1983 15:0:37.40
 #hplot - to create a printer plot
  subroutine hplot (sid, init, end, plist, slist, line)
  character sid(ARB), line(ARB)
  integer init, end, i, ns, istop, istop2, n, ifix, dtofc
  real plist(ARB), slist(ARB)
  double precision dble
  include gatcom

  string blanks "                    "

   call putch (DICI, STDOUT)
   call putlin ("Lab number:   .", STDOUT)
   call putlin (labnv, STDOUT)
   call putlin ("              Field number:   .", STDOUT)
   call putlin (sid, STDOUT)
   call putch (NEWLINE, STDOUT)
   call putlin (" CUMULATIVE FREQUENCY CURVE.", STDOUT)
   call putch (NEWLINE, STDOUT)
   call putch (NEWLINE, STDOUT)
   call putlin (blanks, STDOUT)
   i = 0
   call itoc (i, line, 2)
   call putlin (line, STDOUT)
   for (i = 10 ; i <= 100 ; i = i + 10) {
      len = itoc (i, line, 4)
      for (n = 1 ; n <= 10 - len ; n = n + 1)
         call putch (BLANK, STDOUT)
      call putlin (line, STDOUT)
      }
   call putch (NEWLINE, STDOUT)
   call putlin (blanks, STDOUT)
   for (i = 0 ; i <= 100 ; i = i + 1) {
      if (mod(i,10) == 0)
         call putch (PLUS, STDOUT)
      else
         call putch (UNDERLINE, STDOUT)

      }
   call putch (NEWLINE, STDOUT)

   call putlin (blanks, STDOUT)
   call putch (BAR, STDOUT)
   call putch (BLANK, STDOUT)
   call putch (NEWLINE, STDOUT)
   for (ns = MAXPLIST ; ns >= 1 ; ns = ns - 1) {
      for (n = 1 ; n <= 3 ; n = n + 1) {
         istop = ifix (slist(ns) + 0.5)
         if (n == 2) {
      call putch (BLANK, STDOUT)
      call putch (BLANK, STDOUT)
      call dtofc (dble(plist(ns)), LETF, 5,1,line)
      call putlin (line, STDOUT)
```

```
       call putch (BLANK, STDOUT)
       call putch (BLANK, STDOUT)

       len = itoc (istop, line, 4)
       for (i = 1 ; i <= 3 - len ; i = i + 1)
          call putch (BLANK, STDOUT)
       call putlin (line, STDOUT)
       call putch (BLANK, STDOUT)
       call putch (BLANK, STDOUT)
       call putch (BAR, STDOUT)
       if (istop ^ = 0) {
          for (i = 1 ; i < istop ; i = i + 1)
             call putch (BLANK, STDOUT)
          call putch (BIGX, STDOUT)
          }
       call putch (NEWLINE, STDOUT)
       next
       }

       else {
         call putlin (blanks, STDOUT)
         call putch (BAR, STDOUT)
         }
       call putch (BLANK, STDOUT)
       call putch (NEWLINE, STDOUT)
       }
       }


   return
   end
 #-h- GETWDL 397 asc MON., 30  AUG., 1982 15:11:51.25
   #getwdl - retrieves word from line checking for length & nulls
   integer function getwdl (line, ptr, varabl, maxsiz)
   character line(ARB), varabl(ARB)
   integer getwdq, maxsiz, size, ptr
   size = getwdq(line, ptr, varabl,maxsiz)
   if (size <= 0)
      getwdl = YES
  else if (size > maxsiz) {
     call remark ("Field too long.")
     getwdl = YES
     }
  else
     getwdl = NO

  return
  end
#-h- GOPEN 1415 asc MON., 22  NOV., 1982 13:26:2.37
# gopen - initialize GSTAT files & parameters
   integer function gopen (fidr, fidg, delphi, line)
   character file(FILENAMESIZE), line(ARB)
   integer getres, null, open, create, fidr, fidg, getf
   real delphi

      gopen = OK
      repeat    # get master file name
         if (getres("raw data master file name:.", file, FILENAMESIZE,
            null) == EOF)
            call error("EOF for file name.")
         if (null == YES)
            call remark("null file name invalid.")
         else {
            fidr = open(file, READ)
            if (fidr == ERR) {
                call cant (file)
               break
               }
            }
         }
      until (null ^ = YES)

               # get pre-grasp file name
         if (getres("Pre-grasp file name:.", file, FILENAMESIZE,
            null) == EOF)
            call error("EOF for file name.")
         if (null == YES) {
            call remark("null file name .")
            fidg = EOF
             }
         else {
```

```
       fidg = open(file, APPEND)
       if (fidg == ERR) {
          fidg = create(file, WRITE)
          if (fidg == ERR) {
             call cant(file)
             gopen = ERR
             }
          call remark("starting new file.")
          }
       else
          call remark("appending to existing file.")
       }
   delphi = 0.0
   return
   end
#-h- GDATR 4343 asc TUE., 23  NOV., 1982 13:3:25.05
 # gdatr - retrieves data from masterfile - sample id specified
  integer function gdatr (master, plist, slist, err, line, pend)
   character iflab(MAXLABNUM),
             line(ARB), blabl(MAXASIZE)
   real plist(ARB), slist(ARB)
   real sreal, aval
   integer master, pinit, pend, stat, getlin, getwdq,index,i, j, null
   integer len
   integer iflag, equal, iptr, ptr, err
   include rsacom
   include gsacom
   string tag ">>"
   string stars "**"
   string arsa "RSA"
   # retrieve rsa data for specified sid

     for (nf = 1; nf <= MAXPLIST ; nf = nf + 1) {
        slist(nf) = 0.0
        )
            iptr = 3
            call getwdq(line,iptr,iflab,MAXLABNUM)
            if (equal(labnum,iflab) == NO) {
               call remark ("RSA data out of order in input file.")
               err = ERR
               return
               }
         call getwdq(line, iptr,blabl,MAXASIZE)
         if(equal (blabl,arsa) ^ = YES) {
            call remark ("RSA data out of sequence in master file.")
            err = ERR
            return
            }

         if (getlin(line,master) == EOF) {
            err = ERR
            return
            }
         iptr = 1
         call getwdq (line, iptr, sid, MAXSID)

         if (getlin(line,master) == EOF) {
            err = ERR
            return
            }
         iptr = 1
         call getwdq (line, iptr, projid, MAXPROJID)

         if (getlin(line,master) == EOF) {
            err = ERR
            return
            }
         iptr = 1
         call getwdq (line, iptr, crusid, MAXCRUSID)
         if (getlin(line,master) == EOF) {
            err = ERR
            return
            }
         iptr = 1
         call getwdq (line,iptr,reqnam, MAXREQNAM)
         if (getlin(line,master) == EOF) { #skip oprnam
            call remark ("Raw data master file EOF - err!.")
            err = ERR
            return
            }
         if (getlin (line,master) == EOF) {
           call remark ("EOF retrieving anadat.")
           err = ERR
           return
           }
        iptr = 1
        call getwdq (line,iptr, anadat, MAXANADAT)
        if (getlin(line,master) == EOF) { #get sampl
           call remark ("Error retrieving sample wt.")
           err = ERR
           return
           }
        iptr = 1
        sampl = sreal (line, iptr, 0.0, null)
           if (null == YES)
               call remark ("Null field for Sample wt.")

        if (getlin(line,master) == EOF)    # get coars
            call remark ("EOF retrieving coars data.")
            err = ERR
            return
            }
        iptr = 1
        coars = sreal (line, iptr, 0.0, null)
        if (null == YES)
          call remark ("Null field for coars data.")

        if (getlin(line,master) == EOF)    #get sand
           call remark ("EOF retrieving sand wt.")
           return
           }
       iptr = 1
       sand = sreal(line, iptr, 0.0, null)
       if (null == YES)
          call remark ("Null field retrieving sand wt.")

        for (i = 8 ; i <= MAXPLIST ; i = i + 1) {
           #get relative percents for sand & gravel
           if (getlin(line,master) == EOF)
               break
            if (line == tag | line == stars)
               break
            iptr = 1
            aval = sreal(line, iptr, 0.0, null)
            for (j = MAXPLIST ; j >= 8 ; j = j - 1) {
               if (aval == plist(j))
                  break
               }
            if (j < 8) [ #sample phi value not found in plist8-
MAXPLIST
               call remark ("Phi value on line not in RSA phi range.")
               err = ERR
               return
               }
            alist(j) = sreal (line, iptr, 0.0, null)
            if (null == YES)
               call remark ("Null field in RSA data.")
            if (i >= MAXPLIST) {
               i = i + 1
               break
               }
            }
   if (line ~ = tag) {
      if (getlin(line, master) == EOF) {
         gdatr = ERR
         call remark ("No nav found for sample.")
         return
         }
      }
   pend = i - 1
   if (len == EOF) {
       call putlin ("Rsa data not found for labnum:   .", ERROUT)
       call putlin (labnum, ERROUT)
       call putch (NEWLINE, ERROUT)
       gdatr = EOF
       }
   else
      gdatr = OK
   return
   end
#-h- GDATS 3006 asc MON., 22  NOV., 1982 14:22:14.03
 # gdats - retrieves data from masterfile - sample id specified
```

```
integer function gdata (master, ylist, slist, err, line)
 character iflabn(MAXLABNUM),
           line(ARB), blabl(MAXASIZE),
           a200(4), a30(3)
 real ylist(ARB), slist(ARB)
 integer master, dinit, dend, stat, getlin, getwdq,index,i,getv, len
 include gsacom
 integer iflag, equal, iptr, ptr, err
 string tag ">>"
 string stars "**"
 data a200(1)/DIG2/, a200(2)/DIG0/, a200(3)/DIG0/,a200(4)/EOS/,
      a30(1)/DIG3/, a30(2)/DIG0/, a30(3)/EOS/
 # retrieve data for a200 and then a30

   for (nf = 1; nf <= MAXDLIST ; nf = nf + 1) {
      slist(nf) = 0.0
      ylist(nf) = 0.0
      }
 iflag = 0
 if (getlin(line, master) == EOF) {
    gdata = EOF
    return
    }
 repeat {
          if (line ^ = tag ) {
             call remark ("Input file out of order looking for COULTER.")
             gdata = EOF
             return
             }
          iptr = 3
          call getwdq(line,iptr,iflabn)
          if (iflag == 0) {
              ptr = 1
             call scopy (iflabn, 1, labnum, ptr)
             iflag = 1
             }
          else if (equal(labnum,iflabn) == NO) {
             call remark ("Input file out of order.")
             err = ERR
             break
             }
          else
             iflag = iflag + 1
       call getwdq(line, iptr,blabl)
       if(equal (blabl,a200) == YES)
          index = 1
       else if (equal(blabl,a30) == YES)
          index = 2
       else {
         call remark ("Invalid aperture diameter in master file.")
         err = ERR
         return
         }
       if (getlin(line,master) == EOF) { #skip sid
          err = ERR
          return
          }
       if (getlin(line,master) == EOF) { #skip projid
          err = ERR
          return
          }
       if (getlin(line,master) == EOF) { #skip crusid
          err = ERR
          return
          }
       if (getlin(line,master) == EOF) { #skip reqnam
          err = ERR
          return
          }
       if (getlin(line,master) == EOF) { #skip oprnam
          err = ERR
          return
          }
       if (getlin(line,master) == EOF) { #skip anadat
          err = ERR
          return
          }
       # now get apert.diam, %volume for specified apert. diam
       for (i = 1 ; i <= MAXDLIST ; i = i + 1) {
          if (getlin(line,master) == EOF)
             break
```

```
            if (line == tag | line == stars )
               break
            if (index == 1)
               call getv (line,slist)
            else
               call getv (line,ylist)
            }
   }
   until (iflag == 2)

   if (iflag < 2) {
       call putlin ("Coulter Data not found for lab number:  .", ERROUT)
       call putlin (labnum, ERROUT)
       call putch (NEWLINE, ERROUT)
       gdata = EOF
       }
   else
      gdata = OK
   return
   end
#-h- GDATN 1238 asc MON., 11  APR., 1983 13:36:28.35
#gdatn - reads  field-nav identifiers for gstat
integer function gdatn (fidr, line)
character line(ARB), iflabn(4)
integer fidr, i, getwdq, equal
include gstcom
include gsacom
include rsacom
string nav "NAV"
   iptr = 3
   if (getwdq(line, iptr, labnv) == EOF)
      call error ("EOF reading Labno for field rec.")
   if (equal(labnv,labnum) ~= YES)
      call error ("Nav id out of order in input file.")
   if (getwdq(line, iptr, iflabn) == EOF)
      call error ("NAV not present in input file.")
   if (equal(iflabn, nav) ~= YES)
      call error ("NAV not present in the input file.")
   if (getwdq(line, iptr, latdd, MAXCORDLET) == EOF) {
      call remark ("No latitude for sample in field-nav file.")
      gdatn = ERR
      return
      }
   if (getwdq(line, iptr, londd, MAXCORDLET) == EOF) {
      call remark ("No longitude for sample in field-nav file.")
      gdatn = ERR
      return
      }
   if (getwdq(line, iptr, device, MAXDEVICE) == EOF)
      return
   if (getwdq(line, iptr, area, MAXAREA) == EOF)
      return
   if (getwdq(line, iptr, depth, MAXDEPTH) == EOF)
      return
   if (getwdq(line, iptr, dtop, MAXDEPTH) == EOF)
      return
   if (getwdq (line, iptr, dbotm, MAXDEPTH) == EOF)
      return
return
end
#-h- MPVC 816 asc THU.,  9  DEC., 1982 15:20:5.05
#mpvc - calculates modified percent volume
  subroutine mpvc (slist,ylist,dinit,dend)

  real slist(ARB), ylist(ARB), k(3), k0
  real dif(3), ymin, amin1, abs
  integer i, cro, crover, dend, dinit
   k0  =  (slist(22)  +  slist(23)  +  slist(24))  /  (ylist(22)  +  ylist(23)  +
ylist(24))
  for (i = 1 ; i <= 3 ; i = i + 1) {
     k(i) = (slist(21+i))/ylist(21+i)
     dif(i) = abs(k0 - k(i))
     }
  ymin = amin1 (dif(1), dif(2))
  ymin = amin1 (dif(3),ymin)
  for (i = 1 ; i <= 3 ; i = i + 1) {
     if (dif(i) == ymin)
        break
     }
  k0 = k(i)
  # determine cross-over channel
```

```
 cro = 3
 if (abs(k(1) - 1.0) < abs(k(cro) - 1.0))
     cro = 1
 if (abs(k(2) - 1.0) < abs(k(cro) - 1.0))
     cro = 2
 crover = 21 + cro
 for (i = crover ; i <= dend ; i = i + 1) {
     slist(i) = ylist(i)*k0
     }

 return
  end
#-h- SUMRY 527 asc THU.,  9  DEC., 1982 15:20:7.91
# sumry - compute results of Coulter input
   subroutine sumry(slist, psum, dinit, dend)
   real slist(MAXDLIST),  psum(MAXPHI  ), sum
   integer dinit, dend, i, ip

   for (i = dend ; i >= dinit ; i = i - 1)
       psum(i) = 0.0
   sum = 0.0

       ip = 14
       for (i = dend; i >= dinit ; i = i - 1) {
           psum(i) = slist(ip) + slist(ip+1) + slist(ip+2)
           ip = ip + 3
           sum = sum + psum(i)
           }

       for (i = dend ; i >= dinit ; i = i - 1)
           psum(i) = psum(i) * 100. / sum

   return
   end
#-h- WTFP 1001 asc THU.,  9  DEC., 1982 15:20:10.84
 #wtfp - calculate weighted frequency percentages
   subroutine wtfp (plist, slist)
   real plist(ARB), slist(ARB)
   real sum

   integer i, iflag
  include reacom

   fines = sampl - coars
   pfines = fines/sampl
   for (i = 1 ; i <= 7 ; i = i + 1) {  # calculate weighted coulter data phi11
- phi5
       slist(i) = pfines*slist(i)
       }
   psand = sand/sampl
  for (i = 8 ; i <= 12 ; i = i + 1) { # calculate weighted rsa sand values
phi4 - phi0
       slist(i) = psand*slist(i)
       }
    gravl = coars - sand
    pgravl = gravl/sampl
   for (i = 13 ; i <= MAXPLIST ; i = i + 1) { # calculate weighted gravel phi-
1 - phi-MAXPLIST
       slist(i) = pgravl*slist(i)
       }

  #calculate total analysis of weight of sample
    sum = 0.0
   for (i = 4 ; i <= 7 ; i = i + 1)
       sum = sum + slist(i)
   psilt = sum
   sum = 0.0
   for (i = 1 ; i <= 3 ; i = i + 1)
       sum = sum + slist(i)
   pclay = sum
   pgravl = pgravl*100.0
   psand = psand*100.0
   pfines = pfines*100.0

   return
   end

#-h- MPRINT 2131 asc THU.,  9  DEC., 1982 15:20:16.90
# mprint - copy grain size data to master file
   subroutine mprint(dlist, slist, dinit, dend, out, labl)
   character line(MAXLINE)
```

```
    real dlist(ARB), slist(ARB)
    integer i, dinit, dend, ip, dtofc, out
    double precision dble
    include rsacom
    include gsacom
    string blanks "  ", tag ">>"

        call putch (DIG1, STDOUT)
        call putch (NEWLINE,STDOUT)
        call putch (NEWLINE,STDOUT)
        call putch (NEWLINE,STDOUT)
        call putlin(blanks, out)
        call putlin(labnum, out)
        call putch (NEWLINE, out)
        call putlin (blanks, out)
        call putlin (sid, out)
        call putch(NEWLINE, out)
        call putlin(blanks, out)
        call putlin(projid, out)
        call putch (DASH, out)
        call putlin(crusid, out)
        call putch(NEWLINE, out)
        call putlin(blanks, out)
        call putlin (reqnam, out)
        call putch (NEWLINE, out)
        call putlin (blanks, out)
        call putlin (anadat, out)
        call putch (NEWLINE, out)

        ndp = 2
        call putlin (blanks, out)
        call putlin ("Sample wt:   .", out)
        call mout (sampl, line, out, ndp)
        call putlin (blanks, out)
        call putlin ("pgravl:     .", out)
        call mout (pgravl, line, out, ndp)
        call putlin (blanks, out)
        call putlin ("psand:      .",out)
        call mout (psand, line, out, ndp)
        call putlin (blanks, out)
        call putlin ("pfines:     .",out)
        call mout (pfines, line, out, ndp)
        call putlin (blanks, out)
        call putlin ("psilt:      .",out)
        call mout (psilt, line, out, ndp)
        call putlin (blanks, out)
        call putlin ("pclay:      .", out)
        call mout (pclay, line, out, ndp)
        call putch (NEWLINE,  out)

        call putlin (blanks, out)
        call putlin (labl, out)
        for (i = dend; i >= dinit; i = i - 1) {
            call putlin(blanks, out)
            call dtofc(dble(dlist(i)), LETF, 10, 2, line)
            call dtofc(dble(slist(i)), LETF,  8, 2, line(11))
            ip = 19
            line(ip) = NEWLINE
            line(ip+1) = EOS
            call putlin(line, out)
            if (i == dinit)
                call putch (NEWLINE, out)
            }

    return
    end
#-h- HEADR 1275 asc WED.,  1  JUNE, 1983 15:32:42.78
#headr - to output header for gstat routine
subroutine headr
include gstcom
include gsacom
            call putch (DIG1, STDOUT)
            call putlin ("  Lab number:   .",  STDOUT)
            call putlin (labnv, STDOUT)
            call putch (NEWLINE, STDOUT)
            call putlin ("  Field number:   .", STDOUT)
            call putlin (sid, STDOUT)
            call putch (NEWLINE, STDOUT)
            call putlin ("  Location:   .", STDOUT)
            call putlin (latdd, STDOUT)
            call putch (COMMA, STDOUT)
```

```
          call putlin (blanks, STDOUT)
          call putlin (londd, STDOUT)
          call putch (NEWLINE,STDOUT)
          call putlin ("  Sampling device:   .", STDOUT)
          call putlin (device, STDOUT)
          call putch (NEWLINE, STDOUT)
          call putlin ("  Water depth:   .", STDOUT)
          call putlin (depth, STDOUT)
          call putch (NEWLINE, STDOUT)
          call putlin ("  Top depth:   .", STDOUT)
          call putlin (dtop, STDOUT)
          call putlin ("    -    Bottom depth:   .", STDOUT)
          call putlin (dbotm, STDOUT)
          call putch (NEWLINE, STDOUT)
          call putlin ("  Sampling area:   .", STDOUT)
          call putlin (area, STDOUT)
          call putch (NEWLINE, STDOUT)
          call putch (NEWLINE, STDOUT)
          call putch (NEWLINE, STDOUT)
  return
  end
```

d.) IGSLIB (library for inclusive graphics computations in GSTAT)

```
#-h- IGST 3766 asc WED.,  1  JUNE, 1983 13:56:1.22
#igst - inclusive graphic statistics
 subroutine igst (plist, clist, maxlis, line)
 character line(ARB)
 integer maxlis, iqhscu, i, j, ier, icsevu, k, ncfp, max, ist
 real clist(17), plist(17), coef(17,3), psel(100), csel(100),
     mean, stdev, median, skew, kurt, cspln, aval, bval, cfp(7),
     pcal(7), delta, x(17), y(17)

 #for aplication with gsa plist = phi values : clist = cum.freq.%s
 data cfp(1) /5.0/, cfp(2) /16./, cfp(3) /25./, cfp(4) /50./,
        cfp(5) /75./, cfp(6)/84./,cfp(7)/95./

  #invert data so x(phi) is increasing
  j = maxlis
  for (i= 1; i <= 17 ; i = i + 1) {
      x(i) = plist(j)
      y(i) = clist(j)
      j = j - 1
      }

 #iqhscu - = iqhscu from IMSL Library see vol.II June, 1981
 #             used to compute coefficients to approximate cfp,phi curve
  max = 17
  maxlis = 17
 call iqhscu (x, y, maxlis, coef, max, ier)
 if (ier == 129) {
     call putlin ("Error in computing IGSTATS:   .", ERROUT)
     call error ("row dimension is less than maxlis - 1.")
     }
 if (ier == 130)
    call error ("Error in computing igstats:  maxcol is < 4.")
 if (ier == 131) {
    call putlin ("Error in computing IGSTATS:   .", ERROUT)
    call error ("input abcissa are not in ascending order.")
    }

  #approximate the phi values for the required cfp's
  ncfp = 1
  ist = 1
 repeat {

 for (i=ist;i<=17;i=i+1) {
     if (y(i) >= cfp(ncfp))
         break
     }
 if (y(i) == cfp(ncfp)) {
     pcal(ncfp) = x(i)
     ist = i - 1
     ncfp = ncfp + 1
     next
     }
  j = i - 1
 psel(1) = x(j)
 for (i=2;i<=100;i=i+1)
     psel(i) = psel(i-1) + 0.01
  call icsevu (x, y, maxlis, coef, max, psel, csel, 100, ier)
  if (ier == 33) {
     call monitr ("X(1).",x(1), 3, 1)
    call monitr ("psel(1).",psel(1), 3, 1)
     call error ("Lower limit in search is less than lowest phi.")
    }
   if (ier == 34) {
     call monitr ("x(maxlis).",x(maxlis), 3, 1)
     call monitr ("psel(100).", psel(100), 3, 1)
     call error ("Psel(100) is greater than last phi value.")
     }
  #determine where desired cfp is on graph
  for (i=1;i<=100;i=i+1) {
     if (cfp(ncfp) <= csel(i))
        break
     }
  if (i > 100) { #pcal is between .99 and 1.00 or 1.99 & 2.00 or etc.
     delta = (cfp(ncfp)-csel(100))*0.01/(y(j+1)-csel(100)) + psel(100)
     pcal(ncfp) = delta
     ncfp = ncfp + 1
     ist = j
     }
  else {
     #interpolate between hundredths of a phi
     delta =  (cfp(ncfp) - csel(i-1))*0.01/(csel(i) - csel(i-1))
     pcal(ncfp) = delta + psel(i-1)
     ncfp = ncfp + 1
     ist = j - 1
     }
 }  until (ncfp > 7)
#statistics using inclusive graphics method
#  reference:  See FOLK
#now cfp (1-7) = 5,16,25,50,75,84,95
#     pcal(1-7) = corresponding phi values
#therefore,:
call headr
 call putlin ("******  INCLUSIVE GRAPHICS STATISTICS  ******.",
    STDOUT)
 call putch (NEWLINE, STDOUT)
 call putch (NEWLINE, STDOUT)
 median = pcal(4)
call putlin("  Graphic Median:  .", STDOUT)
call mout (median, line, STDOUT, 2)
mean = (pcal(2) + pcal(4) + pcal(6))/3.0
call putlin ("  Graphic Mean:  .", STDOUT)
call mout (mean, line, STDOUT, 2)
stdev = (pcal(6)-pcal(2))/4.0 + (pcal(7)-pcal(1))/6.6
call putlin ("  Graphic Standard Deviation:  .", STDOUT)
call mout (stdev, line, STDOUT, 2)
skew = (pcal(2)+pcal(6)-2*pcal(4))/(2*(pcal(6)-pcal(2)))
skew = skew + (pcal(1)+pcal(7)-2*pcal(4))/(2*(pcal(7)-pcal(1)))
call putlin ("  Graphic Skewness:  .", STDOUT)
call mout (skew, line, STDOUT, 2)

kurt = (pcal(7)-pcal(1))/(2.44*(pcal(5)-pcal(3)))
call putlin ("  Graphic Kurtosis:  .", STDOUT)
call mout (kurt, line, STDOUT, 2)
call putch (NEWLINE, STDOUT)
call putch (NEWLINE, STDOUT)
call putch (NEWLINE, STDOUT)
call vclass (stdev, line)
call vlimt (skew, line)
call kvlmt (kurt, line)
return
end
#-h- VCLAS 806 asc WED.,  1  JUNE, 1983 14:22:18.03
#vclas-verbal classification of sample using standard deviation
subroutine vclas (stdev, line)
character line(ARB)
real stdev
string vws "very well sorted", ws "well sorted",
       mws "moderately well sorted", ms "moderately sorted",
       ps "poorly sorted", vps "very poorly sorted",
       eps "extremely poorly sorted"
  call putlin ("  Sample is .", STDOUT)
  if (stdev < 0.35)
     call putlin (vws, STDOUT)
  else if (stdev < 0.50)
     call putlin (ws, STDOUT)
  else if (stdev < 0.71)
     call putlin (mws, STDOUT)
 else if (stdev < 1.0)
     call putlin (ms, STDOUT)
 else if (stdev < 2.0)
     call putlin (ps, STDOUT)
 else if (stdev <= 4.0)
     call putlin (vps, STDOUT)
 else
     call putlin (eps, STDOUT)
 call putch (PERIOD, STDOUT)
 call putch (NEWLINE, STDOUT)
  return
  end
#-h- VLIMT 738 asc TUE., 31  MAY , 1983 10:2:40.73
#vlimt-verbal limit of sample using skewness
subroutine vlimt (skew, line)
character line(ARB)
real skew
string sfs "strongly fine-skewed", fs "fine-skewed",
       ns "nearly-symmetrical", cs "coarse-skewed",
       scs "strongly coarse-skewed"

  call putlin ("  Sample is .", STDOUT)

  if (skew > 1.0)
     call putlin ("(skew > 1.", STDOUT)
  else if (skew > 0.3)
     call putlin (sfs, STDOUT)
  else if (skew > 0.1)
     call putlin (fs, STDOUT)
  else if (skew > -0.1)
     call putlin (ns, STDOUT)
  else if (skew > -0.3)
     call putlin (cs, STDOUT)
  else if (skew >= -1.0)
     call putlin (scs, STDOUT)
  else
     call putlin ("(skew < -1.", STDOUT)
  call putch (PERIOD,  STDOUT)
  call putch (NEWLINE, STDOUT)
return
end
#-h- KVLMT 625 asc TUE., 31  MAY , 1983 10:2:45.16
#kvlmt-verbal limit of sample using kurtosis
subroutine kvlmt (kurt, line)
character line(ARB)
real kurt

string vp "very platykurtic", p "platykurtic",
       m "mesokurtic", l "leptokurtic",
      vl "very leptokurtic", el "extremely leptokurtic"
call putlin ("  Sample is .", STDOUT)
if (kurt < 0.67)
  call putlin (vp, STDOUT)
else if (kurt < 0.90)
  call putlin (p, STDOUT)
else if (kurt < 1.11)
  call putlin (m, STDOUT)
else if (kurt < 1.5)
  call putlin (l, STDOUT)
else if (kurt <= 3.0)
  call putlin (vl, STDOUT)
else
   call putlin (el, STDOUT)
call putch (PERIOD, STDOUT)
call putch (NEWLINE, STDOUT)

return
end
```

e.) JJLIB - general utilities written for grain size analysis programs

```
#-h- GETRES 457 asc TUE., 14  SEPT, 1982 11:17:10.62
 # getres - prompt and get response word
  integer function getres(prmpt, str, maxstr, null)
  character str(maxstr), line(MAXLINE)
  integer maxstr, null, getlin, getwdq, i, equal

  string  dash "-"
      call prompt (prmpt, STDIN)
      getres = getlin(line, STDIN)
      if (getres ~ = EOF) {
        i = 1
        if (getwdq(line, i, str, maxstr) == EOF)
           null = YES
        else
           null = equal(str, dash)
        }
     return
     end
#-h- SREAL 244 asc TUE., 14  SEPT, 1982 11:17:18.21
# sreal - single precision input signed number
   real function sreal(line, i, defalt, null)
   character line(ARB)
   real defalt
   double precision dreal
   integer i, null

      sreal = dreal(line, i, dble(defalt), null)

   return
   end
#-h- DREAL 936 asc TUE., 14  SEPT, 1982 11:17:26.66
# dreal - input signed number
   double precision function dreal(line, i, defalt, null)
   define(MAXCNUM, 20) # maximum input character string length
   character line(ARB), cnum(MAXCNUM)
   double precision defalt, v, ctodp
   integer getwdq, i, null, ip, in

      dreal = defalt
      null = NO
      if (getwdq(line, i, cnum, MAXCNUM) == EOF ||
            cnum(1) == DASH & cnum(2) == EOS ||
            cnum(1) == PLUS & cnum(2) == EOS )
         null = YES
      else {
         if (cnum(1) == DASH) {
            v = -1.
            ip = 2
            }
         else {
            v = 1.
            if (cnum(1) == PLUS)
               ip = 2
            else
               ip = 1
            }
         in = ip
         v = v * ctodp(cnum, in)
         if (ip == in) {
            call remark("invalid numeric response.")
            null = ERR
            }
         else
            dreal = v
         }
   return
   end
#-h- GROUT 182 asc FRI.,  5  NOV., 1982 13:57:18.77
#grout - puts out a character variable followed by COMMA
subroutine grout(carray, out)
character carray(ARB)
integer out
call putlin (carray, out)
call putch (COMMA, out)
return
end
#-h- INOUT 304 asc THU.,  9  DEC., 1982 11:32:43.41
#subroutine to in-out data until flag or eof
integer function inout (line, input, output)
  character line(ARB)
  integer input, output, getlin
   string flag ">>"

  if (getlin(line, input) == EOF ) {
     inout = EOF
     return
     }
  if (line ~ = flag)
     call putlin (line,output)
  return
  end
```

```
 #-h- SCDRT 843 asc WED., 20  APR., 1983 10:41:44.47
 #scdrt - to direct scratch file to appropriate output device
 subroutine scdrt (scname, file, line)
 character scname(ARB), file(ARB), line(ARB), filen(FILENAMESIZE)
 integer open, create, fid
 if (getres("Direct output (Control D to abort): .", filen, FILENAMESIZE,
   null) == EOF)
   call error ("Program aborted -- purge scratch file.")
 if (null == YES) { #file will be copied over input file
    call amove (scname, file)
    call remove (scname)
    }
else { #copy scratch file onto named file
  fid = open (filen, APPEND)
  if (fid == ERR) {
      fid = create (filen, WRITE)
      if (fid == ERR)
          call cant(filen)
      else
          call remark ("created new file.")
      }
  else
      call remark ("Output will be appended to existing file named.")
  call close (fid)
  call amove (scname, filen)
  call remove (scname)
  }

 return
 end
#-h- OPENJ 1038 asc WED., 29  JUNE, 1983 14:48:21.13
#openj - just opens an input file and an output file &nothingmore.!
integer function openj (fidin, fidout)
  character file (FILENAMESIZE)
  integer getres, null, open, create, fidin, fidout

  openj = YES
  call getres ("INPUT FILENAME: .", file, FILENAMESIZE,null)
  if (null == YES)
      call error ("Program aborted.")
  fidin = open(file, READ)
  if (fidin == ERR) {
      call cant(file)
      }
  else
      call remark ("Input file has been opened.")
  if (getres ("OUTPUT FILENAME: .", file, FILENAMESIZE, null) == EOF)
      call error ("EOF for filename, program aborted.")
  if (null ~ = YES) {
      fidout = open (file, APPEND)
      if (fidout == ERR) {
          fidout = create (file, WRITE)
          if (fidout == ERR)
              call cant (file)
           else
            call remark  ("Created new output file.")
          }
      else
          call remark ("Appending to existing file.")
      }
  else {
      openj = NO
      call remark ("No output specified, output will go to STDOUT.")
      fidout = STDOUT
      }

  return
  end
```

f.) Other utilites can be found in Tools library (K & R Software Tools).

7. GSANV loader file and software (See also common files and libraries.)

```
LL,~GSANV
RE,%GNV
RE,%GSANV
RE,%GSTNV
SEA,%JJLIB
SEA,%GSALB
SEA,%TOOLS
END
```

```
# gnv - driver for grain size master file creation
#        to compile data from raw data file
      program gnv

                    call initr4

                    call gsanv

                    call endr4

      end
      block data
      include chpdcb
      data maxsys / 4/
    end

#subroutine to control gsanv
subroutine gsanv
character file(FILENAMESIZE), line(MAXLINE)
integer fidout, open, create, getres, gstnv, null, close
  if (getres ("Nav filename:  .", file, FILENAMESIZE, null) == EOF)
     return
  fidout = open(file,APPEND)
  if (fidout == ERR) {
     fidout = create (file, WRITE)
     if (fidout == ERR) {
        call cant (file)
        return
        }
     call remark ("Starting new nav file.")
     }
  else
     call remark ("Appending to existing raw data file.")
  if (gstnv(line,fidout) == EOF)
     call putch (NEWLINE, fidout)
call close(fidout)
return
end

#gstnv - input routine for retrieving sample specs for gstat
  integer function gstnv (line, fidout)
  character line(ARB), labnum(MAXLABNUM), latdd(MAXCORDLET),
            londd(MAXCORDLET)
  character    latd(MAXCORDLET), latm(MAXLOCLET), nors(MAXDIRLET),
               sec(MAXLOCLET),
      lond(MAXCORDLET), lonm(MAXLOCLET), eorw(MAXDIRLET),
      device(MAXDEVICE), depth(MAXDEPTH), dtop(MAXDEPTH), dbotm(MAXDEPTH),
      area(MAXAREA)
  integer iptr, getres, null, getwdq, putlin, putch, stat,fidout,getlin,
         equal, monitr, bell, ddflag, alldig

  real aval, areal
  double precision dble
  string south "S", west "W"
  string tag ">>", nav "NAV"
  data bell/03400b/

 gstnv = getres ("Are locations in decimal degs?  .", line, MAXLINE, null)
 if (gstnv == EOF)
    return
 if (line(1) == LETY | line(1) == BIGY)
    ddflag = YES
 else
    ddflag = NO

 for (stat = 1 ; stat <= MAXSTATS ; stat = stat + 1) {
  call prompt ("Input lab number:  .", ERROUT)
  if (getlin(line, STDIN) == EOF)
     return
  if (line(1) == PERIOD)
     return
  iptr = 1
  null = getwdq (line, iptr, labnum, MAXLABNUM)
  if (null == EOF) {
     stat = stat - 1
     next
     }
  repeat { # get latitude for sample
     null = NO
     if (ddflag == NO)
        call putlin ("Enter latitude (DD DMIN NorS):  .", ERROUT)
    else
       call putlin ("Enter latitude in dec degs:   .", ERROUT)
    gstnv = getlin(line, STDIN)
    if (gstnv == EOF) {
       call remark ("EOF retrieving latitude.")
       stat = stat - 1
       return
       }
    iptr = 1
    null = getwdq (line, iptr, latd, MAXCORDLET)
    if (null == EOF) {
       call bufout (1, bell, 2)
       next
       }
    if (ddflag == NO) {
    null = getwdq(line, iptr, latm, MAXLOCLET)
    if (null == EOF) {
       call bufout(1, bell, 2)
       next
       }
    null = getwdq (line, iptr, sec, MAXLOCLET)
    if (null == EOF) {
       call bufout(1, bell, 2)
       next
       }
    if (alldig(sec) == YES) {
       jptr = 1
       aval = sreal(sec, jptr, 0.0, null)/3600.0
       null = getwdq (line, iptr, nors, MAXDIRLET)
       if (null == EOF) {
          call bufout (1, bell, 2)
          next
          }
       }
    else {
       iptr = 1
       jptr = 1
       call scopy (sec, jptr, nors, iptr)
       aval = 0.000
       }
    iptr = 1
    aval = sreal(latd, iptr, 0.0, null) + aval
    iptr = 1
    aval = aval + sreal(latm, iptr, 0.0, null)/60.0
    if (equal(nors,south) == YES)
       aval = -aval
    call dtofc(dble(aval), LETF, MAXCORDLET - 1, 4, latdd)
    next
    }
    else { #loc in dec degs
    iptr = 1
    jptr = 1
    call scopy (latd, iptr, latdd, jptr)
    }
    }
    until (null ^ = EOF)

repeat {
    null = NO
    if (ddflag == YES)
       call prompt ("Enter longitude in dec degs:   .", ERROUT)
    else
       call prompt ("Enter longitude (DD DMIN EorW):   .", ERROUT)
     gstnv = getlin (line, STDIN)
     if (gstnv == EOF) {
        call remark ("EOF retrieving longitude.")
        return
        }
     iptr = 1
     null = getwdq (line, iptr, lond, MAXCORDLET)
     if (null == EOF) {
        call bufout (1, bell, 2)
        next
        }
     if (ddflag == NO) {
     null = getwdq (line, iptr, lonm, MAXLOCLET)
     if (null == EOF) {
        call bufout (1, bell, 2)
        next
        }
     null = getwdq (line, iptr, sec, MAXLOCLET)
```

```
        if (null == EOF) {
           call bufout (1, bell, 2)
           next
           }
        if (alldig(sec) == YES) {
           jptr = 1
           aval = sreal (sec, jptr, 0.0, null)/3600.00
           null = getwdq (line, iptr, eorw, MAXDIRLET)
           if (null == EOF) {
              call bufout (1, bell, 2)
              next
              }
           }
        else {
           jptr = 1
           iptr = 1
           call scopy (sec, jptr, eorw, iptr)
           aval = 0.000
           }
        iptr = 1
        aval = sreal(lond, iptr, 0.0, null) + aval
        iptr = 1
        aval = aval + sreal(lonm, iptr, 0.0, null)/60.0
        if (equal(eorw,west) == YES)
           aval = -aval
        call dtofc(dble(aval), LETF, MAXCORDLET - 1, 4, londd)
        next
        }
        else { #data in dec degs
        iptr = 1
        jptr = 1
        call scopy (lond, iptr, londd, jptr)
        }
        }
        until (null ^ = EOF)
    gstnv = getres ("Sampling Device (2 Chars):   .", device, MAXDEVICE, null)
    if (gstnv == EOF)
       return
    gstnv = getres ("Area (2 chars):   .", area, MAXAREA, null)
    if (gstnv == EOF)
       return
    gstnv = getres ("Depth:   .", depth, MAXDEPTH, null)
    if (gstnv == EOF)
       return
    gstnv = getres ("Top-Depth:   .", dtop, MAXDEPTH, null)
    if (gstnv == EOF)
       return
    gstnv = getres ("Bottom-Depth:   .", dbotm, MAXDEPTH, null)
    if (gstnv == EOF)
       return
    call putlin (tag, fidout)
    call  grout (labnum, fidout)
    call grout (nav, fidout)
    call grout (latdd, fidout)
    call grout (londd, fidout)
    call grout (device, fidout)
    call grout (area, fidout)
    call grout (depth, fidout)
    call grout (dtop, fidout)
    call putlin (dbotm, fidout)
    call putch (NEWLINE, fidout)
    }
    return
    end
```

8. CLTRM loader file and software. (See also common files and libraries.)

```
LL,'COL
RE,%COL::222
RE,%SHEAD::222
SEA,%GSALB::222
LI,%TOOLS::99
END
```

```
# col - program for coulter counter data input
      program col
                     call initr4

                     call coultr

                     call endr4
```

```
    end
 block data
 include chpdcb
 include rsacom
 include gsacom

      data maxsys/ 4 /

   end
# shead - get sample header data
   integer function shead(dinit, dlist)
   character ans(MAXANS), a200(4), a30(3)
   integer getres, null, i, dinit, getres, iptr, equal, dtofc
   real dlist(MAXDLIST), v, vl, vh
   include gsacom
   double precision ctodp, dble
   string p1 "initial u diameter (",
          P2 " default): "
   data a200(1) /DIG2/,a200(2) /DIG0/,a200(3) /DIG0/, a200(4) /EOS/,
        a30(1)/DIG3/, a30(2)/DIG0/, a30(3)/EOS/

      shead = getres("Lab Number:.", ans, MAXLABNUM, null)
      if (shead == EOF)
         return
      if (null == YES) {
         #sid = same sid
         iptr = 1
         call scopy (a30, iptr, asize, iptr)
         }
      if (ans(1) == PERIOD & ans(2) == EOS) {
          shead = EOF
          return
          }
      iptr = 1
      if (null == NO)
         call scopy (ans, iptr, labnum, iptr)


    if (null == NO)   {
      repeat [ # get sample id or field no. & aperture diameter
         if (getres("Sample ID:  .", ans, MAXSID, null) == EOF)
            call error("EOF entering sample id.")

          if (null == YES)
             next
      iptr = 1
      call scopy (ans, iptr, sid, iptr)
      # get normal aperture diameter

         if (getres("Aperture Diameter: .",ans, MAXANS,null) == EOF)
            call error ("EOF on Apert Diameter.")
         if (null == YES) {
            call scopy (a200,iptr,asize,iptr)
            null = NO
            }
         else if (equal(ans,a200) == YES)
            call scopy (a200, iptr, asize, iptr)
         else if (equal(ans,a30) == YES)
            call scopy (a30, iptr, asize, iptr)
         else {
          call remark ("Invalid entry.")
          null = YES
           }
      ]  until (null == NO)
    }

    repeat { # get starting point of size
       if (asize == a200)
          v = dlist(UD200)
       else
          v = dlist(UD30)
       i = dtofc(dble(v), LETF,8,3,ans)
       call putlin (p1, ERROUT)
       call putlin (ans, ERROUT)
       if (getres(p2, ans, MAXANS, null) == EOF)
          call error("EOF on u diameter.")
       if (null == YES) {
          if (equal(asize,a200) == YES)
             dinit = UD200
          else
             dinit = UD30
            break
            }
         else {
            i =1
            v = ctodp(ans, i)
            if (i == 1)
               call remark("invalid entry.")
            else {
               vl = v * .99999
               vh = v * 1.00001
               for (dinit = 1; dinit <= MAXDLIST; dinit = dinit + 1)
                  if (dlist(dinit) >= vl & dlist(dinit) <= vh)
                     break 2
               call remark("invalid u diameter.")
               }
            }
         }
   if (dinit < 14) {
       call putlin ("Initial u diameter must be <= to .", ERROUT)
       i = dtofc (dble(dlist(14)), LETF, 5, 1, ans)
       call putlin (ans, ERROUT)
       call putch (NEWLINE, ERROUT)
       dinit = 14
       }
   return
   end
```

9. RSAM loader file and software. (See also common files and libraries.)

```
LL,^RSA
OP,LB
RE,%RS::222
SEA,%RSALB::222
SEA,%GSALB::222
LI,%TOOLS::99
END

#rs - driver for Coarse Grain Size data input (keyed input)
      program rs
            call initr4
            call rsa
            call endr4
      end

      block data
      include chpdcb
      include rsacom
      include gsacom

      data maxsys /5/

      end
```

10. JSORT loader file and software.

```
LL,^JSORT
RE,%JSO
SEA,%JSOLB
SEA,%JJLIB
SEA,%TOOLS
END
#symbol file for jsort
define(MAXSYSFILES,  7)
define (MAXKEYLET, 6)
define (MAXTYPLET, 4)
define (MAXTYPES,  4)
# jsort - driver for jsort sorting program
      program jso

                    call initr4

                    call jsort

                    call endr4

      end
      block data
      include chpdcb
      data maxsys /MAXSYSFILES/
    end
#-h- JSORT 3518 asc TUE., 31  MAY , 1983 14:22:39.32
```

```
#jsort - to sort multi-record groups of data & determine status of each
#         sample as to n types of groups.
#
  subroutine jsort
  character line (MAXLINE), key(MAXKEYLET), headr(MAXLINE),
            name(FILENAMESIZE), rline(81), scnam(FILENAMESIZE),
            jsnam(FILENAMESIZE), sortn(41),
            typ(MAXTYPLET, MAXTYPES), temp(MAXTYPLET)
  integer input, sct, getlin, putlin, open, create, itoc, getres,
          head, type, seqno, equal, null, stat, heado
  integer iptr, jptr, i, spawn
 string  sort1 "..K:1:A:A:1:5,K:2:A:D:6:6,K:3:A:A:7:7,K:4:A:A:8:9,L:30,/E"
  string seed "js",
         seed2 "sc",
         sort "sort",
         tag ">>"
  data typ(1,1) /DIG2/, typ(2,1) /DIGO/, typ(3,1) /DIGO/, typ(4,1) /EOS/,
       typ(1,2) /DIG3/, typ(2,2) /DIGO/, typ(3,2) /EOS/,
       typ(1,3) /BIGR/, typ(2,3) /BIGS/, typ(3,3) /BIGA/, typ(4,3) /EOS/,
       typ(1,4) /BIGN/, typ(2,4) /BIGA/, typ(3,4) /BIGV/, typ(4,4) /EOS/
  #open input file
  if (getres("Input file name:  -", name, FILENAMESIZE, null) == EOF)
     return
  input =  open (name, READ)
  if (input == ERR)
     call cant (name)
  #create scratch file
  call scratf (seed, jsnam) #create scratch file for job sorting
  call scratf (seed2, scnam) #create scratch file for sort control file
  sct = open(scnam, WRITE)
  if (sct == ERR)
     call cant(scnam)
  call putlin (jsnam, sct)
  call putch (NEWLINE, sct)
  call putlin (sort1, sct)
  call putch(NEWLINE, sct)
  call close (sct)
  sct = open (jsnam, WRITE)
  if (sct == ERR) {
     call remove (scnam)
     call cant (jsnam)
     }
  #read input file & output to scratch file with following key fields
  #       key, head, type, seqno  where
  #       key == sample id
  #       head == 0 for data record or > 0 for type only record
  #       type == 0 if head > 0
  #               1 if type is typ1
  #               2 if type is typ2
  #               3 if type is typ3
  #               4 if type is typ4
  #       seqno == no. of line within group
  repeat {
     stat = getlin (line, input)
     if (stat == EOF)
        break
     else if (stat == ERR) {
        call remark ("Error reading input file.")
        break
        }
     if (line == tag) {
        seqno = 1
        iptr = 3
        call getwdq (line, iptr, key, MAXKEYLET)
        call getwdq (line, iptr, temp, MAXTYPLET)
        for (i = 1 ; i <= MAXTYPES ; i = i + 1) {
           if (equal(temp, typ(1,i)) == YES)
              break
           if (i == MAXTYPES) {
              call putlin ("Type of sample for key .", ERROUT)
              call putlin (key, ERROUT)
              call error  (" not in type list.")
              }
           }
        type = i
        head = 9
        if  (heado (key, head, type, seqno, headr, sct) == ERR)
           break
        call putch (NEWLINE, sct)
        head = 0
        }
        if  (heado (key, head, type, seqno, headr, sct) == ERR)
            break
        call putlin (line, sct)
        seqno = seqno + 1
     }  until (stat == EOF)
     if (stat == ERR)
        return
     #close input file, scratch file
     call close (input)
     call close (sct)
     #sort scratch file in order above
     stat = spawn (sort, scnam, sortn, WAIT)
     call kill (sortn)
     call remove (scnam)
     if (stat == ERR) {
        call remark ("Error in sort, sort err = .")
        call itoc (stat, line, 6)
        call putlin (line, ERROUT)
        return
        }
     #open output file(s)
     #output all samples to appropriate file(s)
     call gsout (jsnam, name, line)
     #close output file(s) & purge scratch file
     return
     end
#-h- HEADO 911 asc WED.,  6  APR., 1983 14:49:21.63
#subroutine heado - writes out header for sorting used by jsort
integer function heado (key, head, type, seqno,temp, out)
character key(ARB), temp(ARB), junk(4)
integer head, type, len, seqno, out, itoc, putlin, putch, iptr, length
     iptr = 1
     if (length(key)  = MAXKEYLET - 1)
        call error ("Number of chars in KEY field not = MAXKEYLET.")
     call stcopy (key, 1, temp, iptr)
     len = itoc(head, junk, 2)
     call stcopy (junk, 1, temp, iptr)
     len = itoc (type, junk, MAXLINE)
     if (len > 1) {
        call remark ("type > 1 char ... abort.")
        heado = ERR
        return
        }
     call stcopy (junk, 1, temp, iptr)
     len = itoc (seqno, junk, MAXLINE)
     if (len > 2) {
        call remark ("Greater than 99 lines per group .. abort.")
        heado = ERR
        return
        }
     if (len == 1) {
        temp(iptr) = DIGO
        iptr = iptr + 1
        }
     call scopy (junk, 1, temp, iptr)
     call putlin (temp, out)
     return
     end
#-h- GSOUT 2350 asc FRI., 19  NOV., 1982 11:59:48.64
#gsout - treat sorted file from jsort and output acc. to gss format.
subroutine gsout (jsname, name, line)
character jsname(ARB), name(ARB), line(ARB), scnam(FILENAMESIZE),
          newn(FILENAMESIZE)
integer open, create, getwdq, getlin, getres, ctoi
integer sct, sct2, fid, len, stat, iptr, type, fidout
string seed "to"
     sct = open(jsname, READ)
     if (sct == ERR)
        call cant (jsname)
        call scratf (seed, scnam)
     sct2 = open(scnam, WRITE)
     if (sct2 == ERR)
           call cant (scnam)
        icond = 15     #all groups must be present to go to newfile
                    #else will go to scratch & then rn to rdmaster(name)
        #where should groups go if icond is met??
     if (getres("Enter filename for complete data set output:  .",
           newn, FILENAMESIZE, null) == EOF)
              return
        fid = open(newn, APPEND)
        if (fid == ERR) {
```

```
       fid = create(newn,WRITE)
          if (fid == ERR)
             call cant(newn)
          else
             call remark ("Starting new output file.")
       }
      else
         call remark ("Appending to existing complete dataset file.")
      #read through sorted file and output data appropriately
       stat = getlin(line,sct)
   repeat {
      iflag = 0
      for ( ; line(6) ~ = DIGO ;
           stat = getlin(line,sct)) {
         if (stat == EOF | stat == ERR)
            break 2
         line(8) = BIGA
         iptr = 7
         type = ctoi(line,iptr)
         switchto type {
            iflag = ior(iflag,1B) #set bit 0 on
            iflag = ior(iflag,2B) #set bit 1 on
            iflag = ior(iflag,4B) #set bit 2 on
            iflag = ior(iflag,10B) #set bit 3 on
            }
         else {
            call error ("Data type in sort file > 4 or < 1.")
            }
         }
         if (iflag == icond)
            fidout = fid
         else {
            fidout = sct2
            call icout (line, iflag)
            }
      call putlin(line(10),fidout)

      for (stat = getlin(line,sct) ; line(6) == DIGO ;
           stat = getlin(line,sct)) {
         if (stat == EOF | stat == ERR)
            break 2
         call putlin(line(10), fidout)
         }

   }  until(stat == EOF)
   call close (fid)
   call close (sct)
   call close (sct2)
   call remove (jename)
   call amove (scnam,name)
   call remove (scnam)

   return
   end
#-h- ICOUT 921 asc THU., 25  AUG., 1983 15:45:24.03
#icout-tells user which group(s) not present for incomplete samples for jsort
subroutine icout (line, iflag)
character line(ARB), key(MAXKEYLET)
integer iflag, i, iand, ifrst
  ifrst = 0
if (iand(iflag,1B) ~ = 1) {
   call putlin ("200 u Coulter.", STDOUT)
   ifrst = ifrst + 1
   }
if (iand(iflag,2B) ~ = 2) {
   if (ifrst > 0)
      call putch (COMMA, STDOUT)
   call putlin (" 30 u Coulter.", STDOUT)
   ifrst = ifrst + 1
   }
if (iand(iflag,4B) ~ = 4) {
   if (ifrst > 0)
      call putch (COMMA, STDOUT)
   call putlin (" RSA.",STDOUT)
   ifrst = ifrst + 1
   }
if (iand(iflag,10B)  = 8) {
   if (ifrst > 0)
      call putch (COMMA, STDOUT)
   call putlin (" NAV.", STDOUT)
   }
   call putlin (" not present for .", STDOUT)
   for (i= 1 ; i < MAXKEYLET ; i = i + 1)
      key(i) = line(i)
   key(i) = EOS
   call putlin (key, STDOUT)
   call putch (NEWLINE, STDOUT)
   return
   end
```

11. GSTAT loader file and software. (See also common files and libraries.)

```
OP,LB
LL,^GSTAT::222
RE,%CST
RE,%IGST
RE,%IQHSU
RE,%ICSEU
LI,%GSTLB
LI,%IGSLB
LI,%RSALB
LI,%CSALB
LI,%TOOLS::99
END

# gs - driver for grain size master file creation
#          to compile data from raw data file
      program gst

                         call initr4

                         call gstat

                         call endr4

      end
    block data
    include chpdcb
    include rsacom
    include gstcom
    include gsacom

       data maxsys/ 5 /

    end

#gstat - subroutine to calculate statistics for gsa data & output for grasp
    subroutine gstat
    character line(MAXLINE),
              name(MAXSEDNAM), ans(MAXANS)
    integer i, j, ip, ih, k, nsam, ns, is, irec, iptr, jptr, ifprnt, ifplot,
          ifigs, fidr, fidg, ifs, pinit, pend, spend, istop, gopen, mout,
            getlin, getwdq, getarg, putlin, putch, gdatr, gdats, gdatn,
            istats, ghead, ists
#istats = number of stations for which field ids found
     integer monitr
    real plist(MAXPLIST), slist(MAXPLIST), f(MAXPHIINT),
         clist(MAXDLIST), blist(MAXDLIST),
         delphi, dphi, mode1, mode2, median, mean, stdev, skew, kurt,
         cmp1, sqrt, sigma, en1, dp, zm2, zm3, zm4, o(5), os(5), aval, sum

    include rsacom
    include gstcom
    include gsacom
    string tag ">>", blanks "  "
    string blank "      "
    string labl "     PHI        % \n \n"
    data plist / 11.0, 10.0, 9.0, 8.0, 7.0, 6.0, 5.0, 4.0, 3.0,
                 2.0, 1.0, 0.0, -1.0, -2.0, -3.0, -4.0, -5.0/
    iptr = 1
    call scopy (blank,iptr, oprnam, iptr)
    iptr = 1
    call scopy (blank,iptr, asize, iptr)
    dp = 1.0

    ifprnt = NO
    ifplot = NO
    for (iptr = 1 ; iptr <= 3 ; iptr = iptr + 1) {
       if (getarg(iptr, line, MAXLINE) == EOF)
          break
       else if (line(1) == BIGP & line(2) == BIGR)
          ifprnt = YES
       else if (line(1) == BIGP & line(2) == BIGL)
```

```
         ifplot = YES
      else if (line(1) == BIGI & line(2) == BIGG)
         ifigs = YES
      else

      call remark ("Undefined command on run line.")
   }
istop = NO
  istats = 0
repeat {
istop = YES
   if (gopen(fidr, fidg, delphi, line) == ERR)
      break

   irec = 1
   ists = 1
   for (is = 1 ; is <= 999 ; is = is + 1) {
      ifs = NO
      if (gdats(fidr, blist, clist, err, line) == EOF)
         break 2
      if (err == ERR) {
         call remark ("ERR retrieving coulter data.")
         break 2
         }
       #get rsa data
      if (gdatr(fidr, plist, slist, err, line, pend) == EOF) {
         break 2
         }
      if (err == ERR) {
         call remark ("ERR retrieving rsa data.")
         break 2
         }
      if (gdatn(fidr, line) == ERR)
         break 2
      call putlin ("Data retrieved for lab number:   .", ERROUT)
      call putlin (labnum, ERROUT)
      call putch (NEWLINE, ERROUT)
      pinit = 14
      spend = 34
      iptr = 1
      jptr = 1
      call scopy (labnum, iptr, ans, jptr)
      call mpvc (clist, blist, pinit, spend)
      pinit = 1
      spend = 7
      call sumry (clist, slist, pinit, spend)
      call wtfp (plist, slist)

      pinit = 1
         clist(MAXPLIST) = alist(MAXPLIST)
         for (i = MAXPLIST -1 ; i >= 1 ; i = i - 1)
            clist(i) = alist(i) + clist(i+1)
         if (clist(1) < 99.9 | clist(1) > 100.1) {
            call putlin ("Sum of relative frequencies not 100 +- 1% for:",
                      ERROUT)
            call putlin (ans, ERROUT)
            call putlin ("   ; Sample ignored.", ERROUT)
            call putch (NEWLINE, ERROUT)
            next
               }

         if (ifplot == YES) {
            iptr = 1
            jptr = 1
            call scopy (sid, iptr, ans, jptr)
            call pplot (ans, pinit, pend, plist, slist, line)
            call hplot (ans, pinit, pend, plist, clist, line)
            }
         if (ifprnt == YES) {
            call mprint(plist, slist, pinit, pend, STDOUT, labl)
            }

         call headr          #to output to STDOUT id-header
         #no need to check for even phi intervals

            for (i = 1 ; i <= pend ; i = i + 1 ) {
               if (slist(i)  = 0.0)
                  break
               }
            pinit = i
            empl = plist(pend) - 0.5*dp
```

```
    nsam = pend - pinit + 1
    ip = nsam
    j = 1
 if (delphi == 0.0) {
#       call remark ("No interpolation done on samples.")
    dphi = dp
    j = 1
    for ( i = pend ; i >= pinit ; i = i - 1) {
       f(j) = alist(i)
       j = j + 1
       }
    }
 call putlin ("  ******  METHOD OF MOMENTS STATISTICS  ******.",
          STDOUT)
 call putch (NEWLINE, STDOUT)
 call putch (NEWLINE, STDOUT)

 #determine classification according to Shepard
 call sedcls (name)
 call putlin (blanks, STDOUT)
 call putlin ("Classification of sample:   .", STDOUT)
 call putlin (name, STDOUT)
 call putch (NEWLINE, STDOUT)
 call mode (os, o, f, empl, dphi, ip)
 for (i = 1 ; i <= 5 ; i = i + 1) {
    aval = o(i)
    if (aval == 0.0)
       break
    call putlin (blanks, STDOUT)
    call rank (i, STDOUT)
    call putlin ("modal class:   .", STDOUT)
    ndp = 2

    call mout (aval, line, STDOUT, ndp)
    call putlin (blanks, STDOUT)
    call rank (i, line, STDOUT)
    call putlin ("modal frequency:   .", STDOUT)
    aval = os(i)
    call mout (aval, line, STDOUT, ndp)
    }
 call median (f, ip, dphi, empl, ctmed)
 call putlin (blanks, STDOUT)
 call putlin ("Median:   .", STDOUT)
 call mout (ctmed, line, STDOUT, ndp)

if (ip > 1) {
 call momnts (ip, empl, dphi, f, enl, zm2, zm3, zm4)
 #calculate standard deviation, skewness, and kurtosis
 sigma = sqrt(zm2)
 skew = 0.5*zm3/(sigma*zm2)
 kurt = zm4/zm2**2 - 3.0
 }
else { # all sample in one phi class
 enl = empl
 sigma = 0.0
 kurt = 0.0
 skew = 0.0
 }
 call putlin (blanks, STDOUT)
 call putlin ("Mean = .", STDOUT)
 call mout (enl, line, STDOUT, ndp)
 call putlin (blanks, STDOUT)
 call putlin ("Standard Deviation = .", STDOUT)
 call mout (sigma, line, STDOUT, ndp)
 call putlin (blanks, STDOUT)
 call putlin ("Skewness = .", STDOUT)
 call mout (skew, line, STDOUT, ndp)
 call putlin (blanks, STDOUT)
 call putlin ("Kurtosis = .", STDOUT)
 call mout (kurt, line, STDOUT, ndp)
 call putch (NEWLINE, STDOUT)
 call putch (NEWLINE, STDOUT)

 #output data to pre-grasp file
 if (fidg ~ = EOF) {
     i = i - 1  #no. modal classes in sample
       call gprint (fidg, plist, clist, line,
      name, ctmed, enl, sigma, skew, kurt, o, os, i,inv)
    }
 if (ifigs == YES) {
    maxlis = MAXPLIST
    call igst (plist, clist, maxlis, line)
```

```
            }
         }
      } until (istop == YES)

   call close (fidr)

if (fidg ^ = EOF)
   call close (fidg)
return
end
```

POPPE AND OTHERS—APSAS—AUTOMATED PARTICLE SIZE SYSTEM—U.S. GEOLOGICAL SURVEY CIRCULAR 963