70640

JPC 426

Report Number
TM-66-7

Computer Programming for a Digital Data Acquisition System

Technical Memorandum

by

N. J. Barsic

C. M. Ehresman

GPO PRICE $ ____________

CFSTI PRICE(S) $ ____________

Hard copy (HC) 3.00

Microfiche (MF) 2.60

ff 653 July 65

NASA Grant NsG 592

August 1966

JET PROPULSION CENTER
PURDUE UNIVERSITY

SCHOOL OF MECHANICAL ENGINEERING
LAFAYETTE, INDIANA

PURDUE UNIVERSITY
AND
PURDUE RESEARCH FOUNDATION
Lafayette, Indiana

Report No. TM-66-7

COMPUTER PROGRAMMING FOR A DIGITAL DATA ACQUISITION SYSTEM

Technical Memorandum

by

N. J. Barsic
C. M. Ehresman

NASA Grant NsG 592

Jet Propulsion Center
Purdue University

August 1966

ACKNOWLEDGEMENTS

The investigation presented herein was supported by the National Aeronautics and Space Administration under Contract Number NsG 592 (PRF 3837-52-2889).

The review was prepared under the direction of Professor C. M. Ehresman, Visiting Professor of Mechanical Engineering. The author wishes to express his gratitude to Professor Ehresman for his invaluable assistance and guidance throughout the course of this study and in the preparation of this thesis. The helpful guidance and suggestions offered by Dr. C. F. Warner are also sincerely appreciated.

The author also wishes to express his appreciation to Mrs. Sue Waszilycsak for typing the final manuscript and to Mrs. Jackie McFerran for typing the original manuscript.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The objective of this investigation is to establish the digital tape format required by a data acquisition system, develop a method for obtaining records of experimental data in engineering units, and prepare the groundwork for analyzing the data by techniques that are familiar to engineers.

The digital recording subsystem is emphasized since it provides the most accurate data; however, the analog recording components are also described.

A literature survey concerning the statistical analysis of instrumentation measurements included investigations of rectangular, triangular, and Gaussian distribution curves in addition to curve fitting techniques for calibration data, and resulted in computer programs which enable extremely accurate analysis of calibration data from sensing devices.

Obtaining data in engineering units from the digital subsystem required investigating several different computer languages. Fortran and MAP languages were selected to perform all programming tasks such as altering the format of information on the digital output tape from

LIST OF FIGURES (continued)

the recording system and converting raw data to engineering units for use in performance calculations. The manner in which computer programs are written is described in order to aid the Jet Propulsion Center personnel should the programs require alteration resulting from data system expansion or modification.

Results of several tests conducted at the Combustion Research Laboratory were recorded, the data read and then analyzed. Samples of the corresponding computer output appear in Appendix D.

It is recommended that existing computer programs be simplified as they become more familiar to the programmers. Additional programs should be written to calculate heat transfer rates and performance parameters. Transferring the computer programs from punched cards to magnetic tape should be investigated when the combined program size justifies such action.

On line processing offers many advantages which should be investigated when the computer facilities at Purdue University justify purchase of the required equipment.

1. INTRODUCTION

The data acquisition system described herein is part of the Purdue University Jet Propulsion Center, Combustion Research Laboratory which was constructed under National Aeronautics and Space Administration contracts NsG-592 and NsG(f)-21. The specific research described in this paper was conducted under NsG-592.

The data system being described is composed of an analog and a digital subsystem. The digital subsystem, which produces the most accurate data with the system, is emphasized in this paper.

The first of three major subtasks concerning the digital subsystem was to establish the digital tape coding format. An investigation of different digital tape coding formats was required both to understand the basic raw data format and to devise a format suitable for storing experimental data for easy accessibility. The process of altering the data format also required investigating various computer programming languages.

The second subtask, obtaining data in engineering units, required performing a statistical analysis investigation regarding instrumentation measurements. Several curve fitting techniques were studied before selecting the ones that are presented in later sections of this paper.

The final subtask established procedures for analyzing data and preparing the digital subsystem programmers to most effectively utilize the existing computer programs and alter the programs if necessary. The computer programs which read data tapes, alter the format, and write output tapes in preparation for data analysis are explained in detail in Sections 5 and 6.

Sample digital recording data obtained from several test firings of a rocket engine at the Combustion Research Laboratory are presented in the Appendix section. The results of these tests are discussed, and recommendations for future utilization of the data system are presented.

2. THE DATA ACQUISITION SYSTEM

The data acquisition system shown in Figure 1 was designed to record experimental data in the most accurate possible manner. Experimental data is conditioned and then recorded by either strip chart recorders, an oscillograph, an analog tape recorder, digital tape recorder or any combination of the indicated equipment. The digital subsystem is capable of recording 20 data channels and is designed to accept a maximum of 40 channels to provide for future expansion.

2.1 System Components

The block diagram (Figure 2) illustrates major components of the data acquisition system. Signals are directed from sensing devices to junction boxes (one per test cell). Data are then transmitted to the control building where the patch panel, signal conditioning equipment, and recording devices are located.

2.1.1 Sensing Devices

Data from experiments conducted at the Combustion Research Laboratory are obtained from sensing devices such as strain or

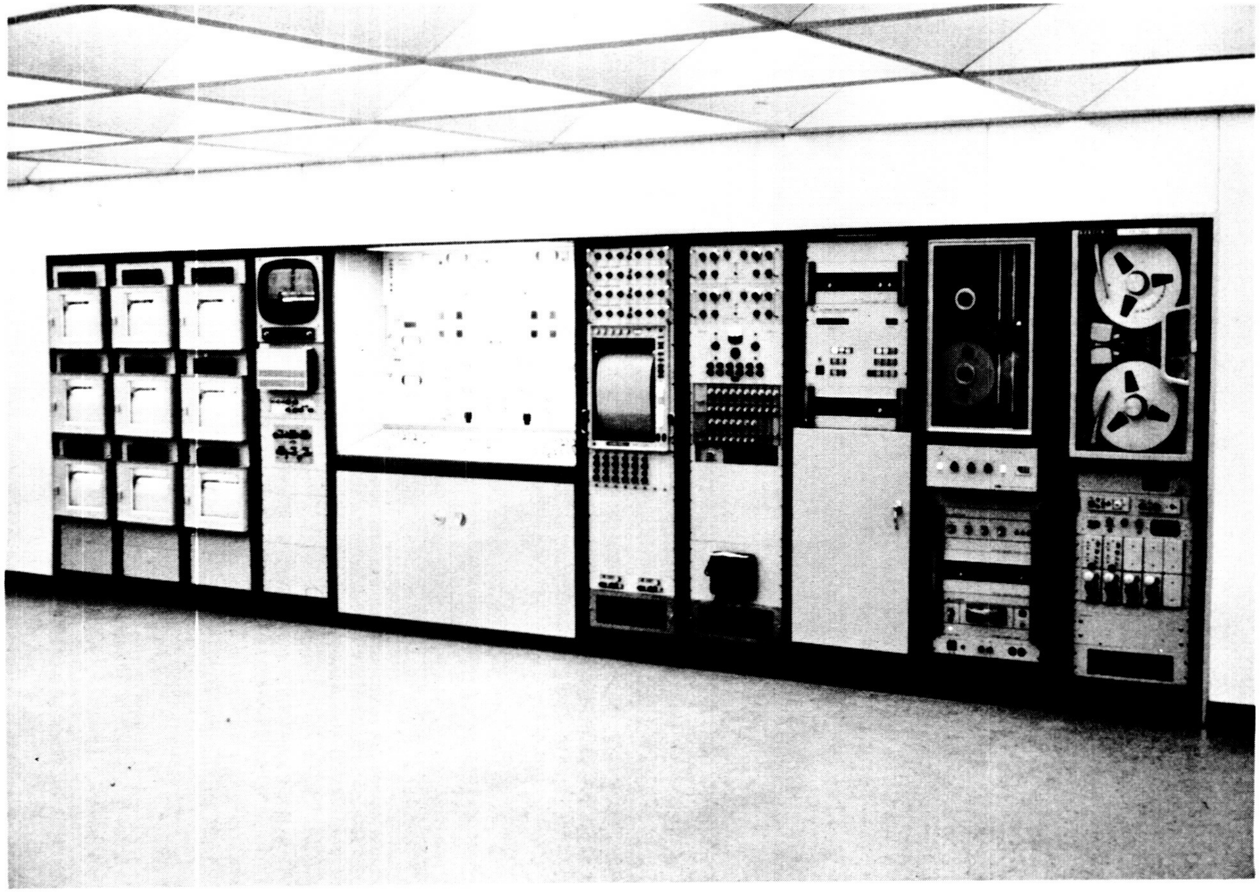


FIGURE I. THE DATA ACQUISITION SYSTEM

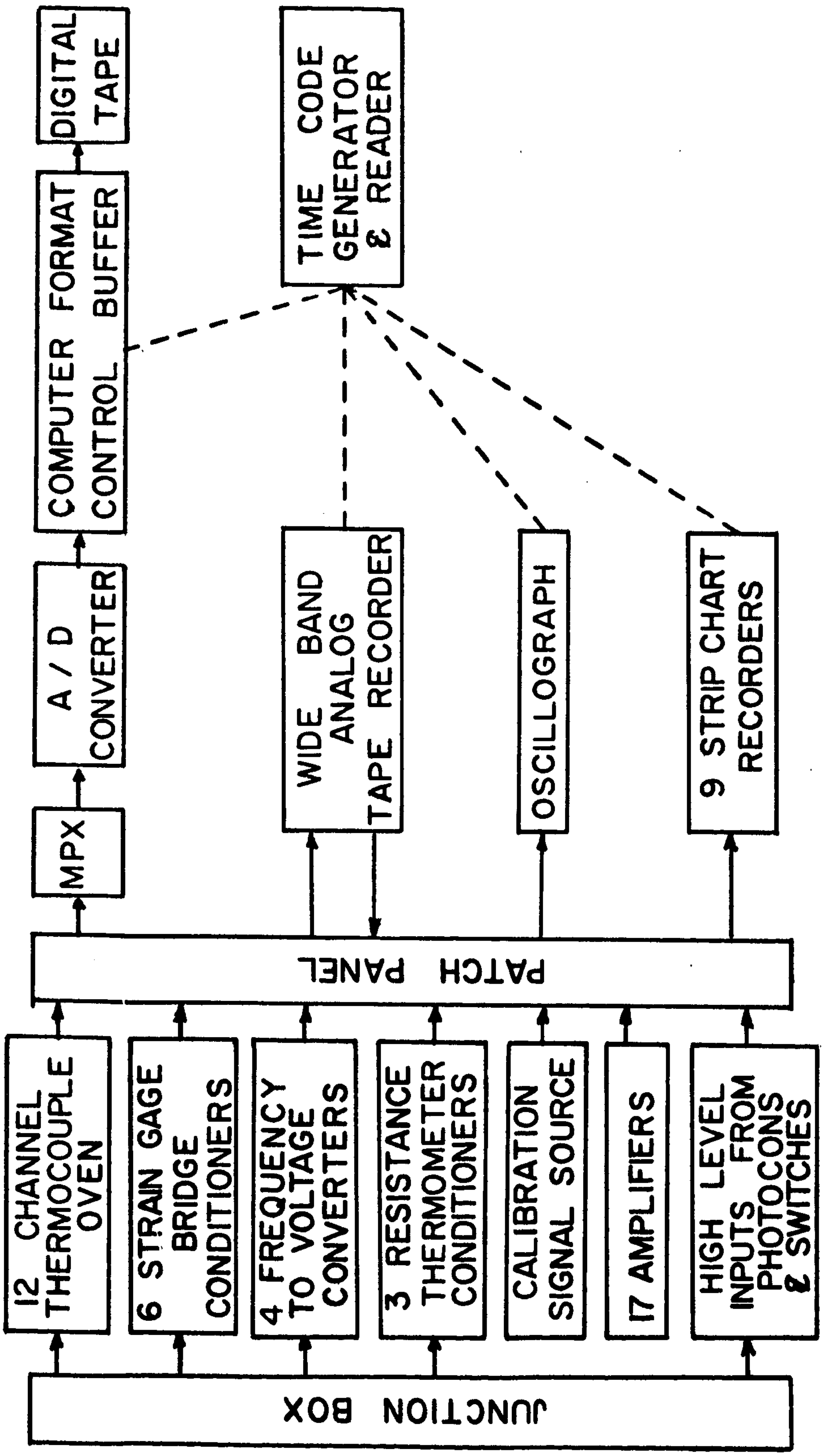


FIGURE 2. DATA ACQUISITION SYSTEM BLOCK DIAGRAM

capacitance type pressure transducers, thermocouples, flow meters, and load cells. The output from a sensing unit is either a frequency signal or an analog millivolt signal. Before recording the experimental data, it must be conditioned to provide matched input to the recording instruments.

2.1.2 Signal Conditioning Equipment

Signal conditioning equipment is necessary since recording devices require analog voltage signals in the 1/2 to 10 volt range. The components are listed below and also appear on Figure 2.

a. Pace Thermocouple Reference Junction and Control Unit, Model TC2R-12-610: The maximum capacity of the Pace unit is 12 temperature inputs.

b. Honeywell Strain Gage Unit, Model 902641 and Sorenson Power Supply, Model QB12-1: These devices convert line voltage (117 v. AC) to a precisely regulated 12 v. D.C. to power the six strain gage circuits.

c. Frequency to Voltage Converters: Four Potter Model 574R2L frequency to voltage converters are employed to convert the frequency output from Potter turbine flowmeters to a voltage signal.

d. Resistance Thermometer Bridge Conditioners: Three Rosemount Thermister bridge conditioners, Model 400A, are necessary to provide output signals from propellant and coolant tank temperature sensors.

e. Amplifiers: Twenty Beckman C-44 variable gain amplifiers are provided to amplify signals from pressure transducers, thermocouples, and other millivolt output sensing devices.

f. Calibration Signal Source: A Honeywell constant voltage supply serves as a voltage standard to check out the digital system.

g. Photocon Signal Conditioners: Power Supplies, Model PS-616, and Dynagage Units, Model DG-605D, designed for use with Photocon pressure transducers are employed to ensure the most accurate possible high frequency pressure records.

The AMP, 480 pin patch board shown in Figure 2 allows interconnection of incoming signals with the data conditioning equipment and then to the digital subsystem and/or components of the analog subsystem.

2.1.3 Analog Subsystem

The analog recording system is designed to record data at frequencies from D.C. to 20,000 cps, at signal levels varying from 1/2 volt to 10 volts.

A Wide Band Analog Tape Recorder, Honeywell Model LAR 7400 with FM electronics, will be used to record signals ranging from DC to 20,000 cps. Initially, 4 channels will be utilized; however, 10 additional channels are available for future expansion.

The Oscillograph is a Honeywell 36 channel model 1612. This unit presently includes electronics and galvonomters for recording

24 channels of data. Addition of 12 more galvonometers and galvonometer amplifiers will increase the recording capability to 36 channels.

Nine Strip Chart Recorders, Honeywell model 16K, are used to record low frequency or steady state data such as tank pressures and temperatures or flow rates.

2.1.4 Digital Subsystem

The Object of the digital subsystem is to obtain the most accurate possible record of experiments. Digitizing the experimental data is the best method of obtaining the most accurate data since measurements are machine transformed into specific numerical voltages (and later, engineering units) with an overall digital system accuracy of 0.5%, including the transducer[1]. Transducers, load cells, thermocouples, and other sensing devices yield voltage output signals which represent experimental data. Signals are then conditioned (i.e. altered from a frequency to a voltage signal) or amplified as described in the Signal Conditioning Equipment paragraph. The continuous analog voltage signals from as many as 20 channels are then sampled sequentially, digitized, and recorded on digital tape for temporary storage until processing by an IBM 7094 computer occurs.

The Digital Subsystem Manufacturer is the Electronic Engineering

---

1. The accuracy specification for the digital subsystem was more restrictive than for other recording devices.

Company of California (EECO). The system is presently capable of accepting 20 incoming data signals, however, the system can be expanded in the field to accept 40 input channels by adding circuit cards.

An Analog Multiplexer capable of accepting up to 100 input channels is the first major component of the digital subsystem. Electronics for 20 input channels are presently included. Overall scanning rates of 100, 200, 500, 1000, 2000 and 4000 samples per second are available. With the maximum input capacity of 40 channels, a sampling rate of 100 samples per second per channel is possible. A single channel mode enables sampling one channel 4000 times per second. By employing multiple patching, two channels can be sampled 2000 times per second or four 1000 times per second and so on.

An Analog to Digital Converter digitizes the data after multiplexing. The EECO model 760A A to D converter provides a nixie tube display of the contents of any desired incoming channel. The maximum number displayed is $\pm$ 9999, thus characterizing this as a 4 decimal digit system.

A Format Generator and Buffer unit follows the A to D converter to control placement of identification characters and temporary storage of data by means of a sequential interlace memory device. The memory function is necessary because gaps must be inserted between data records on the digital tape[1] as shown in Figure 3. Since the data are sampled continuously and data cannot be recorded while gaps are being generated on the magnetic tape, this data storage capability is necessary.

In order to label tapes, experimental tests, sensing devices used, and other data which serves to identify important characteristics of an experiment, the Format Generator and Buffer provides a means of manually entering identification digits (often called header information). Up to 12 decimal digits can be selected by setting thumbwheel switches on the format generator and buffer unit. If more than 12 digits are required for identification purposes, header data can be recorded in any multiples of 12, up to a maximum of approximately 1000 characters per block[2].

<u>A Time Code Generator and Reader</u> is included to correlate time between several different recording devices (i.e., oscillograph, strip chart recorders, analog tape, and digital system). An EECO 858A time code generator and reader generates time signals at one second intervals (fast code) for the digital subsystem and at either 5, 10,

---

1. The gaps are required by IBM data processing equipment (1)[3].

2. The terms record and block are synonymous. A record or block refers to the data that is stored between gaps as shown in Figure 3.

3. Numbers in parentheses indicate references listed in the Bibliography.

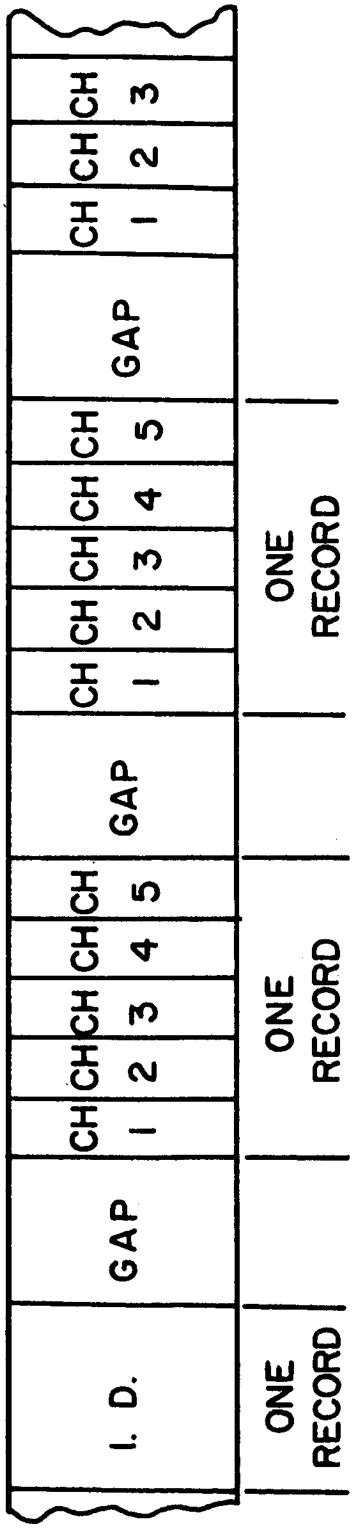


FIGURE 3. TAPE SECTION SCHEMATIC

or 60 second intervals (slow code) for the analog subsystem components.

The Digital Tape Recorder is a Potter model MT-36. Recording densities of either 556 or 800 bits per inch are possible with this 7-track digital tape recorder. The unit is compatible with IBM 729 tape readers; therefore, a digital tape recording can be placed directly onto the IBM 7094 computer for data evaluation.

2.2 Coordinating the Data System with IBM Equipment

The most critical task required for operation of the digital recording system is completing the gap that exists between the digital tape recorder and the University computer facilities. The specific requirements of the Jet Propulsion Center digital system indicated that a twofold project was necessary. First, the selected digital system must be physically compatible with the University IBM equipment. Second, the digital system output tape must be of such a configuration that only minor format modification need be necessary before processing the experimental data by a Fortran program to calculate heat transfer rates, flow characteristics and other experimental results.

Prior to requesting proposals from bidders, the capabilities and requirements of the Purdue University computer facilities were investigated. To provide for compatibility, the digital data acquisition system output must be recorded at densities of either 556 or 800 bits per inch on seven track tape, and with either BCD or binary coding; however, binary coding was preferred.

2.2.1 Digital Tape Coding Format

The digital tape code format for experimental data consists of binary code, while time code generator output and identification digits are recorded in BCD. Using both binary and BCD on the same tape is an acceptable combination; however, the complexity of the data processing increases. No serious problems arise since the computer programs which read the digital tape are also capable of handling both types of code.

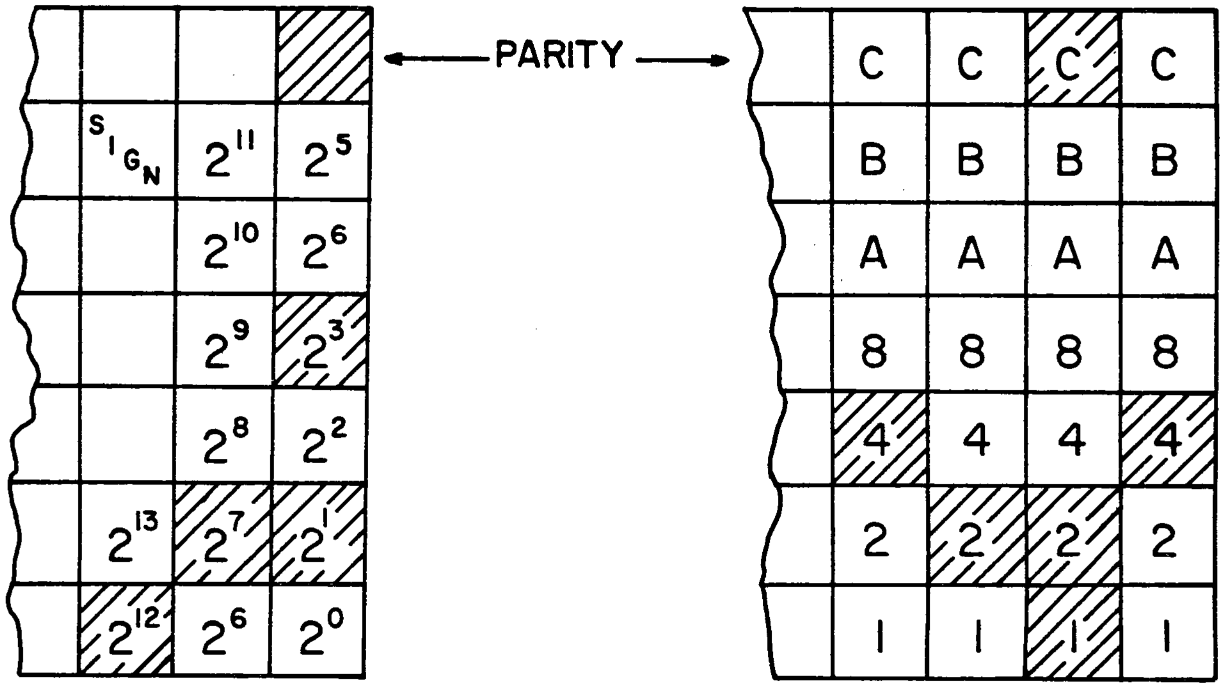
The Differences Between BCD and Binary Coding are illustrated in Figures 4a and 4b. Standard IBM tape consists of 7 vertical tracks, the top track being reserved for parity in both coding systems. Numeric data is stored in tracks 1 through 6 with binary code and tracks 1 through 4 with BCD. Tracks 5 and 6 do not concern this particular system when BCD is used.

Binary Data is represented in powers of 2 as shown in Figure 4a. The number 4234 is represented by the magnetized tape areas or bits: 2^{12}, 2^7, 2^3 and 2^1.

$$2^{12} + 2^7 + 2^3 + 2^1 = 4096 + 128 + 8 + 2 = 4234$$

Binary is the most practical code to use with an IBM 7094 computer since the machine itself uses pure binary coding.

BCD is an abbreviation for binary coded decimal. In BCD code each decimal digit is generated separately. A combination of the numbers 8, 4, 2, and 1 is employed to represent any decimal digit from 0 to 9.



$2^{12}+2^{7}+2^{3}+2^{1} = 4234$

FIGURE 4a. FIGURE 4b.

DIGITAL TAPE SCHEMATICS ILLUSTRATING

a. BINARY CODE and b. BCD

Magnetized areas in blocks 4, 2, and 1 represent the number 7. The number 4234, which was previously coded in binary, is shown in Figure 4b using BCD. Note that binary code utilizes the tape more efficiently since only 3 tape frames[1] are needed to represent 4234, while four frames are necessary with BCD.

Parity refers to the method which computers employ to check the validity of coded information (1). Lateral parity checks the validity of data contained in each tape frame. Longitudinal parity checks recorded information by generating a check character at the end of each block.

A longitudinal parity error is indicated if the character block length thumbwheel switches (see Section 4) are incorrectly set, or if entire frames of data have been lost.

Odd lateral parity, which is used with this data system, requires the total number of bits within any frame to be an odd number. If stored data requires four bits in a particular frame, then the area in track seven is magnetized to obtain an odd number of magnetized locations within the frame under discussion. If three bits had been required to represent data, then no parity bit would be generated in the seventh track. When errors occur due to bits being lost or excessive bits being generated as data is transferred from tape to memory, from memory to tape, or from different locations within a computer, the parity bits are also affected by the above errors.

---

1. A frame is one vertical column of 7 bits.

The appearance of a parity error cannot indicate the quantitative value of error that has occurred. Rather, parity serves to indicate when a bit has been lost or gained and detailed investigation of the affected area of the tape and/or system is required.

If both BCD and binary coding are employed, it is not possible to read or record a tape using two different types of lateral parity[1]; therefore, although BCD and binary code appear on the same tape, both must be in either even or odd lateral parity. Since most of the data (experimental data) for the digital subsystem are recorded in binary, odd lateral parity is used for all records. Since BCD data are recorded with odd lateral parity, this data must be processed by a special computer program before it is meaningful, while the binary data can be read directly. The program that processes BCD data will be described in Section 5.

2.2.2 Digital Tape Data Format

The digital tape format consists of two basic configurations. In the first configuration the experimental data format is recorded in pure binary code. The second configuration contains identification data, consisting of numerical information that serves to identify the experimental data. Identification data are generated in BCD by means of

---

1. Normally a magnetic tape is written with either all BCD coding or all binary coding. Binary code requires odd lateral parity and BCD requires even lateral parity.

manually operated thumbwheel switches on the Computer Format Control Buffer.

Experimental Data are recorded as a digital voltage represented by up to 4 decimal digits. In Figure 4 note that any 4 decimal digit number can be represented by two complete 6-bit tape frames and one 3-bit frame (a total of 14 bits plus sign). The maximum voltage that is anticipated for this system is $\pm$ 9.999 volts, therefore, the digital system must be capable of recording $\pm$ 9999. The series below illustrate the maximum number that can be represented by 12, 13, and 14 binary bits, and indicate the need for using 14 bits with this system.

a. 12 bits (2 complete frames): $2^0 + 2^1 + \ldots + 2^{11} = 4,095$

b. 13 bits: $2^0 + 2^1 + \ldots + 2^{12} = 8,191$

c. 14 bits: $2^0 + 2^1 + \ldots + 2^{13} = 16,383$

The 7094 computer reads 6 tape frames in succession and stores the data contained in these 6 frames into one memory location. As noted in the preceeding paragraph, one data sample is contained in 3 tape frames; therefore, two data samples are stored in one location within the computer. The computer "trade" term for storing two different numbers within the same location is called "packing". Before data can be analyzed and processed, the data must be "unpacked" or separated as shown in Figure 5. The most effective way of unpacking data is to employ a semi-machine language programming method called MAP (2,3,4). The specific manner in which a MAP program is employed to unpack data is demonstrated in Section 5.

PACKED DATA $\boxed{43757624}$

UNPACKED DATA $\boxed{4375}$ $\boxed{7624}$

FIGURE 5. PACKED AND UNPACKED DATA

Identification Data require a more complex manner of processing than experimental data since the BCD code cannot be read directly by the computer because odd rather than even lateral parity is used. As with binary data, the computer reads 6 frames of BCD information and stores the data into one location. Section 5 includes a description of BCD data processing.

2.2.3 Programming Languages

The two programming languages required for tape reading, format alteration (i.e. unpacking), and tape rewriting are Fortran (5,6) and MAP (2,3,4). A decision to employ either of these two languages for a program was based on the task to be performed. Fortran was preferred because the personnel that will be using the system are more familiar with Fortran, however, many of the program tasks require MAP language.

Fortran means formula translation, and is generally used to solve engineering problems through the use of mathematical equations or logical expressions. For example, the equation, $y = mx + b$, appears as follows in Fortran:

```
Y = M * X + B
```

A logical expression commands the computer to perform a specific task. If N equals zero, then the next statement to be executed is 101.

```
IF (N EQ. 0)    GO TO 101
```

Macro Assembly Program (MAP) Language is the most efficient method of processing coded digital data. MAP is a semi-machine language which combines the advantages of utilizing basic machine instructions and meaningful symbolic commands. Pure machine language consists of numeric codes such as 400 for addition and 200 for multiplication. MAP utilizes the symbols ADD and MPY rather than the respective numeric codes cited above. Fortran simply requires the symbols + and *, however, the flexibility of MAP is much greater as Section 5 indicates in greater detail.

2.2.4 Programming Tasks

The four main programming tasks required by the Combustion Research Facility digital system are tape reading, data format alteration, data writing, and engineering calculations. Each task is best accomplished by means of a particular programming language.

Reading the Digital Tape is best accomplished by means of MAP in conjunction with the IBM Library Input/Output Control System (IOCS). IOCS is a programming subsystem of the 7094 that automatically controls data transmission to and from recording devices and makes data readily available for processing (7). MAP commands are used to read a digital data tape because Library IOCS must be called by MAP statements.

Data Format Alteration is best performed in MAP since the necessary tasks readily lend themselves to MAP programming techniques. For example, to "unpack" two data samples, a shifting instruction, LGL N, can be used. LGL means "logical left shift" and effectively shifts a 36 bit data sample N places to the left[1]. LGL 18 is used to shift half of a packed word into an empty storage position which can be labeled for future reference. The remaining half of the packed word (18 additional bits) can be moved to another location for later reference. The specific MAP instructions employed in the programs written for this data system appear in Appendix A.

Writing Data for visual inspection of raw data after altering its format is accomplished by means of Fortran subroutines. It is possible to use either MAP or Fortran for writing the data, however, nearly equal programming effort is required with either language; therefore, Fortran is employed since most engineers are familiar with this language. Data will be written on paper and on another magnetic tape by using two slightly different Fortran statements. Typical "write" statements for printed output and magnetic tape output appear below in respective order.

```
100 FORMAT (2X,20I6)

    WRITE (6,100) (DATA (I), I = 1,N)

    WRITE (2) (DATA (I), I = 1,N)
```

If data are to be read from the magnetic tape record generated by the last "write" statement above, the proper "read" statement is:

---

1. Recall that two data samples will be "packed" into one 36 bit location called an IBM word.

```
READ (2)   (DATA (I), I = 1,N)
```

Note that both statements are identical except for the READ rather than WRITE command.

<u>Data Processing Programs</u> actually depend on the experiment being conducted. For this paper it is merely necessary to indicate that data can be processed by means of a Fortran program since the experimental data have been written on a magnetic tape in Fortran as described in the previous paragraph.

3. STATISTICAL ANALYSIS OF EXPERIMENTAL DATA

3.1 Introduction

The results of an experiment are generally determined by measuring characteristics of the system under consideration and recording the results with equipment similar to the Combustion Research Laboratory Data Acquisition System. Errors are always present, therefore, determining the uncertainty concerned with a particular measurement is necessary.

3.2 Analysis of Measurements

Since methods by which the uncertainty can be established are dependent on particular sources of error, investigating the error sources of measurements is beneficial in order to provide a firm background for subsequent discussions. An inherent characteristic of any measurement is the presence of errors which are difficult to isolate. In order to obtain valid measurements, the sources of error that are likely to be found in a measurement must be analyzed. One way of determining errors is to carefully analyze the instruments and assess their limitations. Another way of determining the errors of a measurement is to analyze the data obtained from an experiment.

3.3 General Classes of Error

Two important types of disturbances are random error and systematic error. If repeated measurements of the same quantity give rise to differing values, then random error is present. Systematic error refers to perturbations which influence all measurements of a particular quantity in the same manner (8). For example, a series of weight measurements would all be high if a systematic error were present in the measuring system.

The term "truly random error" refers to a situation in which a large number of small perturbing influences affects a measurement. Perturbations in instrumentation systems are usually line voltage fluctuations, vibration of instruments, RF interference, and many other occurrences which are frequently grouped under the heading of noise. Considering the combination of all error to be random is a realistic approach for measuring systems because many errors are, in fact, random and the net result of the different systematic errors is, effectively, a group of random errors.

3.4 Accuracy and Precision

Measurement analysis discussions generally always include the terms accuracy and precision; therefore, a brief explanation of these two words appears necessary.

Precision refers to the degree of mutual agreement that is characteristic of independent measurements of a single quantity yielded by repeated applications of a measuring process under specified conditions. Precision is characterized by conformity of several measurements with each other, and without regard to conformity with the true value of a measured parameter.

For example, a measuring system may contain a bias which makes all readings incorrect by 2 units. If all measurements of a given quantity made by the system of this example (such as the weight of a specific object) yield identical results, the measuring process is said to be precise; however, the results are actually incorrect by 2 units.

Accuracy concerns the degree of agreement which independent measurements of a single quantity have with the true value of the quantity concerned. Conformity with the truth characterizes the accuracy of a measurement of a quantity that is performed by the repeated application of the measuring process under identical conditions. In summary, precision concerns closeness together, and accuracy concerns closeness to the truth (9).

The two preceeding paragraphs reveal that accuracy requires precision, but precision does not necessarily imply accuracy.

3.5 Statistics for Measurement Analysis

3.5.1 Purpose of Statistics

Since there is always some magnitude of error present in a measurement, and due to the variety of forms which error assumes, properly treating measurement errors is a difficult task. When treating instrumentation systems, the general practice is to consider that all errors combine into a form which can be treated as noise. Unless gross errors are noticed (i.e. measurements appear to be incorrect by an order of magnitude) or results seem suspicious, all measurements are assumed to contain random errors which are usually called noise. The problem then becomes one of attempting to establish the magnitude of random errors or noise which is present when a reading is taken, thereby establishing the validity of a measurement.

In order to establish the validity of a measurement, the frequent practice is to take several measurements of the same quantity. Duplicating measurements is an important part of any experiment, and utilizing the large array of measurements which contain random interference effects requires analytical treatment by statistical methods (10).

3.5.2 The Object of Statistical Theory

The object of statistical theory is the study of random variables. The symbol X denotes a random variable and it represents the result of an unspecified trial such as the outcome of tossing a coin or the result of any action whose outcome is due to chance occurrences (11). For the engineer, a random variable is simply the result of an experiment that is subject to so many disturbing effects that its outcome cannot be predicted. Measurements fall into this category because, as indicated in the previous section, instrumentation is subject to so many small perturbations that obtaining the true value of a measured parameter is entirely a chance occurrence.

3.5.3 Graphical Representation of Statistical Data

A Histogram is probably the best means of describing the range of values obtained for one measurement. To construct this figure, the observed readings are divided into several groups. The number of times the value of a measurement falls within each group is termed the frequency. Figure 6 describes the number of times a measurement Q falls within each of the selected ranges, a through g.

In cases where a large number of readings are taken so that fine subdivisions are possible, the histogram can be drawn in the form of a continuous curve as shown in Figure 7. This is commonly called a distribution curve. Histograms or distribution curves serve as an excellent means of presenting tabular data since the pictorial representation provides a much clearer understanding of the experimental observations.

Basic Concepts of Distribution Curves require definition before discussing specific statistical theory concerning measurement analysis. Symmetrical and non-symmetrical distribution curves, standard deviation, variance, mean, mode, and median are basic terms that must be understood

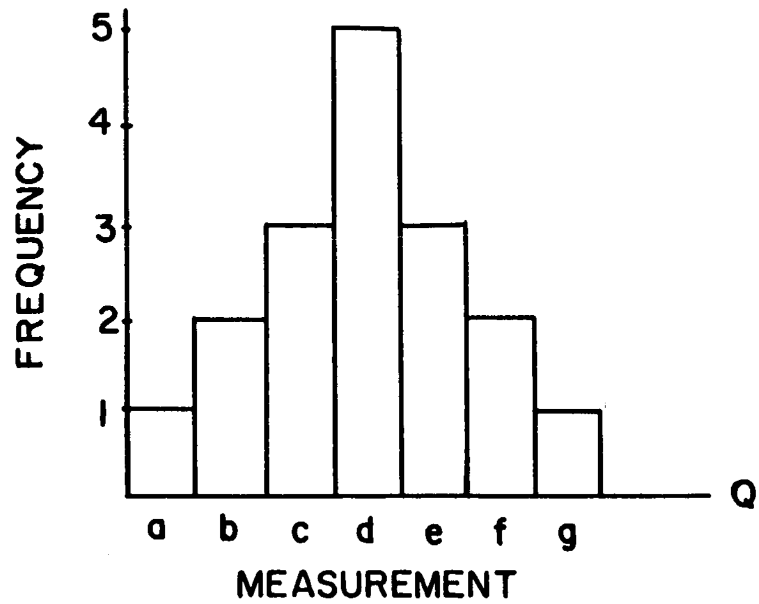


FIGURE 6. SYMMETRICAL HISTOGRAM

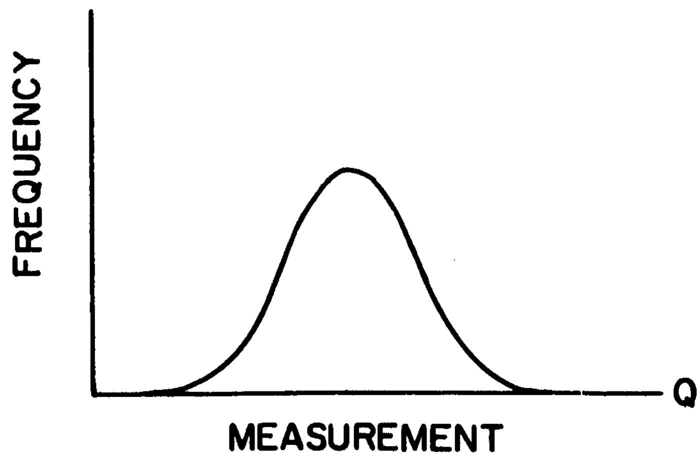


FIGURE 7. SYMMETRICAL DISTRIBUTION CURVE

before a thorough treatment of statistics is possible.

A symmetrical distribution curve is symmetrical about a vertical line drawn through the curve at its highest point. The Gaussian distribution, Figure 8, is the most common form of symmetrical distribution curve. A non-symmetrical distribution curve is shown in Figure 9. Note that the non-symmetrical distribution curve shown in Figure 9 is the sum of two symmetrical (Gaussian) distribution curves (12).

In statistical theory, the mean, $\bar{x}$, of a set of n numbers x_1, x_2, . . . x_n is defined as

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{1}$$

The mean (13) is actually the familiar arithmetic average. For symmetrical distributions the mean appears at the peak of the curve (i.e. the most frequently recorded value). For non-symmetrical distributions this is not true. The most frequently recorded value for non-symmetrical distributions is called the mode, which is quite different from the mean.

Another quantity frequently used in statistical theory is the median. This value is selected so that as many data points fall above it as fall below it. For example, 20 is the median of the numbers in Figure 10.

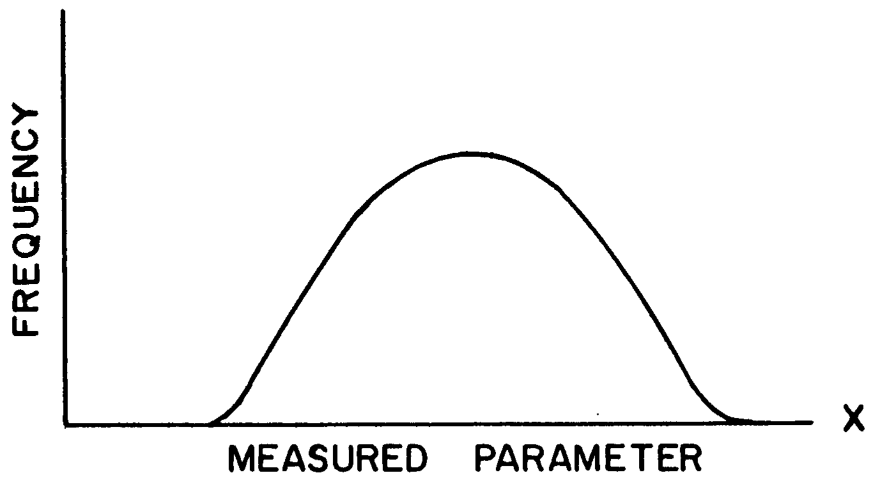


FIGURE 8. GAUSSIAN DISTRIBUTION CURVE

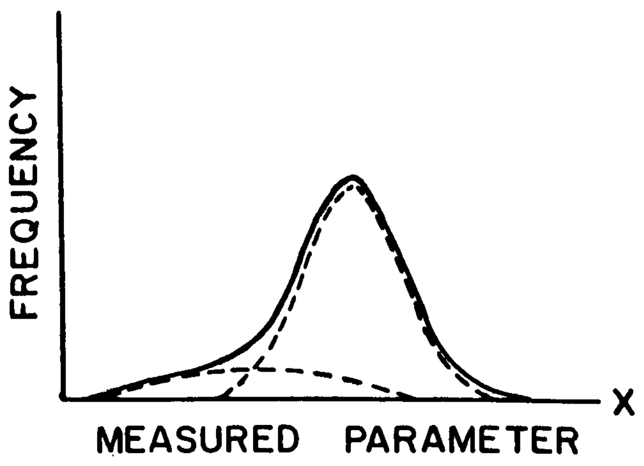


FIGURE 9.
NON-SYMMETRICAL DISTRIBUTION CURVE

10, 15, 12, 9, 11, 19, 20, 25, 28, 35, 41, 22, 30

6 values median 6 values

Figure 10. Illustration of the Median

For a symmetrical distribution, the mean, mode, and median all coincide. The mean, or arithmetic average occurs at the peak of a symmetrical distribution curve. Since the mode is the most frequently recorded value, it also occurs at the center of the abcissa. Half of the data points fall to the left and half fall to the right of the peak on a symmetrical distribution curve, therefore, the median of a symmetrical distribution curve also occurs at the center.

To describe the "uncertainty" in the "best value" for a measured parameter, a quantity connected with the actual width of the distribution curve must be defined. Obviously, such a value is associated with the deviations of the readings from some central value. A quantity often concerned with the uncertainty of a measurement is the mean deviation. Consider a set of n measurements of a quantity such as pressure (x_1, x_2, . . . x_n ; mean = $\bar{x}$). The mean deviation is defined as

$$\delta_m = \frac{1}{n} \sum |x_i - \bar{x}| \tag{2}$$

Another common quantity often used to describe the uncertainty of measurements is the variance

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2 = \int_{-\infty}^{\infty} (x_i - \bar{x})^2 f(x)\, dx \qquad (3)$$

The square root of the variance is called the standard deviation. Many other similar "uncertainty" definitions are available, however, the most useful one depends on the particular problem being considered.

The Meaning of the Term Probability becomes clearer after altering the ordinate of a distribution curve so that the total area under the curve is equal to one. The equation describing the distribution curve is then called a probability density function. The probability of a particular measurement falling into a certain abcissa interval is given by the area which lies within this interval and under the probability density curve.

If a continuous random variable X has a probability density function p(x) (Figure 11) then for any real numbers b and c the probability that X will lie between b and c is

$$P_r(b < X < c) = \int_{b}^{c} p(x)\, dx \qquad (4)$$

REFERENCE (11)

In Figure 11 the ordinate has been adjusted to yield a total area of 1 under the curve thus making this a probability density curve. A probability of 1 guarantees an experimenter that a random variable X will assume a value between the limits necessary to obtain a probability of 1.

$$P_r\,(o < x < \infty) = 1 = \int_{o}^{\infty} p(x)\;dx \tag{5}$$

When considering measurements it is not practical to demand a probability of 1 since the useful range that must be accepted is too large to be of value (the range is $-\infty$ to $+\infty$ for a normal or Gaussian distribution, and 0 to ∞ in Figure 11). The best solution is to compromise and accept a probability somewhat less than 1 thereby, limiting the spread of values of the random variable to a more reasonable range such as b to c in Figure 11.

Applying Statistical Theory to Instrumentation first requires careful analysis of instrumentation systems. Distribution curves[1] adequately describe the situation in which a large amount of data has been collected by the measurement of a specific quantity, such as taking several samples of the boiling point of a new compound. The real physical situation concerning instrumentation at best generally involves only a few measurements of one quantity and often only one measurement. For example, in sampling data from a

---

1. The terms probability density and probability distribution are interchangeable.

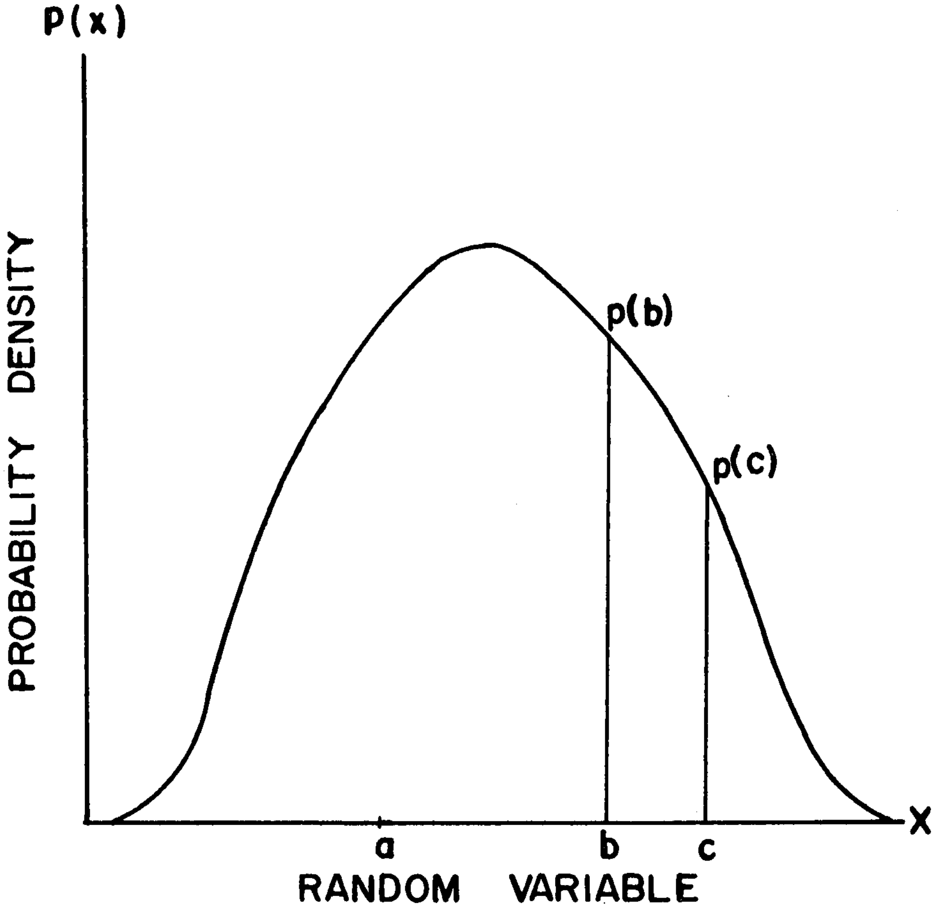


FIGURE II. PROBABILITY DENSITY CURVE

rocket engine test facility only one data sample per sensing device per unit time can be obtained. Because it is often impossible to obtain more than one data sample during an experiment, a complete statistical analysis for the purpose of evaluating instrumentation systems, including the generation of a distribution curve, cannot be readily accomplished. Analysis of instrumentation systems for the purpose of establishing statistical information concerning measurements rather than recording experimental data enables the repeated study of a few experiments so that statistical data can be obtained.

As early as the eighteenth century, scientists observed a surprising degree of regularity in errors of measurement. Experimenters found that patterns (distributions) which they observed were closely approximated by a continuous curve denoted as the "normal curve of errors" and attributed to the laws of chance. The mathematical properties of the normal distribution and its theoretical basis were first investigated by Pierre Laplace, Abraham de Moivre, and Carl Gauss (11). Since measurements follow the Gaussian distribution, the procedure used to statistically describe experimental data is to assume a Gaussian distribution and calculate quantities such as the "uncertainty in a measurement."

3.5.4 Distribution Curve Comparison

As noted in the previous section, experimental evidence indicates that instrumentation measurements are characterized by a Gaussian distribution. To further justify the use of a Gaussian distribution for instrumentation measurements, a comparison of several distribution curves will be considered.

A Rectangular Distribution Curve is obtained if six measurements have been recorded and the data are presented by the histogram in Figure 12.

Mathematical treatment of this data is begun by reconstructing the histogram, assuming it would appear as the distribution curve in Figure 13. Note that Figure 13 is a reconstruction of Figure 12 with the ordinate falling midway between the two measured values, a and b. Since the distance from 0 to a equals 0 to b, redefine the limits as shown in Figure 14. The ordinate is the frequency; however, it is also related to the probability:

Probability, Pr, equals the area under the distribution curve:

$$\int p(x)\,dx$$

The probability that a measurement will lie within $\pm\sigma$ of the central or "best value" of a recorded measurement will be calculated below. Due to the coordinate system chosen in Figure 14, $\bar{x}$, the average value of measurements, should be zero.

$$\bar{x} = \int_{-\infty}^{\infty} x\,p(x)\,dx = \int_{-\alpha}^{\alpha} \frac{x}{2\alpha}\,dx = \left.\frac{x^2}{4\alpha}\right|_{-\alpha}^{\alpha} = 0 \qquad (6)$$

The variance, σ^2, is calculated directly as shown below; however, the square root of this value is of real significance.

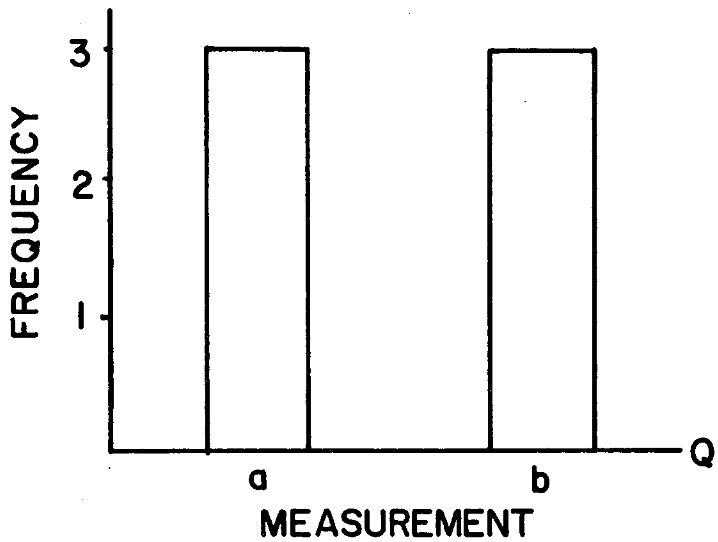


FIGURE 12. HISTOGRAM FOR SIX VALUES OF Q

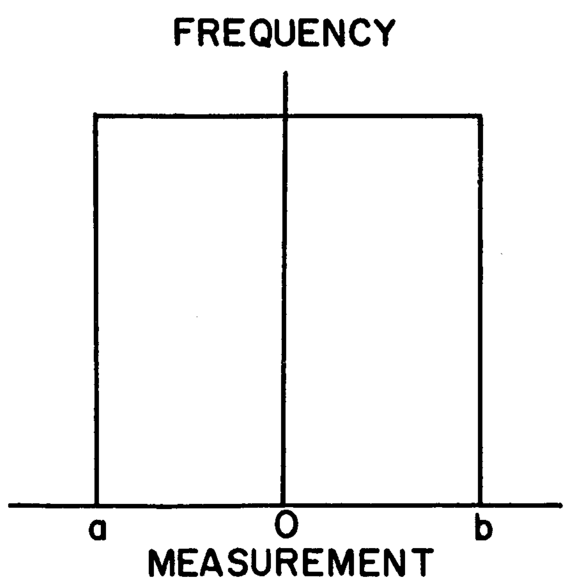


FIGURE 13. RECTANGULAR DISTRIBUTION CURVE

$$\sigma^2 = \int_{-\infty}^{\infty} (x-\bar{x})^2 \, p(x)dx = \int_{-\alpha}^{\alpha} \frac{x^2}{2\alpha} dx = \frac{x^3}{6\alpha} \Big|_{-\alpha}^{\alpha} = \frac{\alpha^2}{3} \tag{7}$$

$$\sigma = \frac{\alpha}{\sqrt{3}} \tag{8}$$

The probability of any measurement falling within $\pm\sigma$ of the mean value is equal to the area under the distribution curve between $-\sigma$ and $+\sigma$ as shown by the shaded area in Figure 15.

$$Pr = \int_{-\sigma}^{\sigma} p(x)dx = \int_{-\frac{\alpha}{\sqrt{3}}}^{\frac{\alpha}{\sqrt{3}}} \frac{dx}{2\alpha} = \frac{x}{2\alpha} \Big|_{-\frac{\alpha}{\sqrt{3}}}^{\frac{\alpha}{\sqrt{3}}} = 0.577 \tag{9}$$

A probability of 0.577 means that the probability of having a measurement fall within $\pm\sigma$ of some "best value", $\bar{x}$, is 0.577, while an exact probability (corresponding to 100% accuracy) would be 1.

A Triangular Distribution Curve results when measurement data are collected which yields the distribution curve shown in Figure 16.

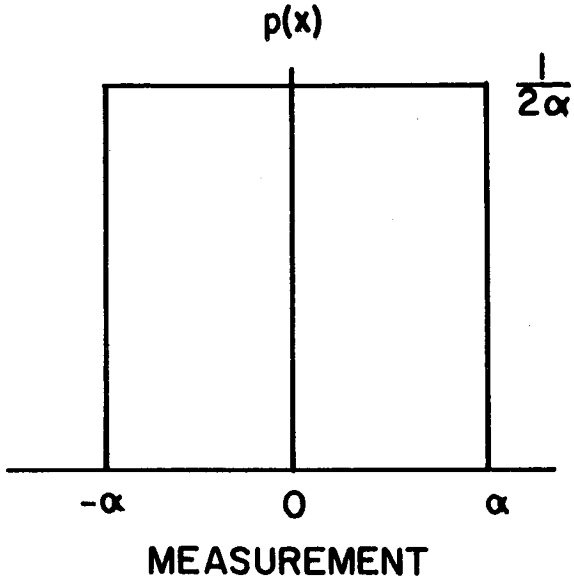


FIGURE 14.
RECTANGULAR DISTRIBUTION CURVE WITH AREA = 1

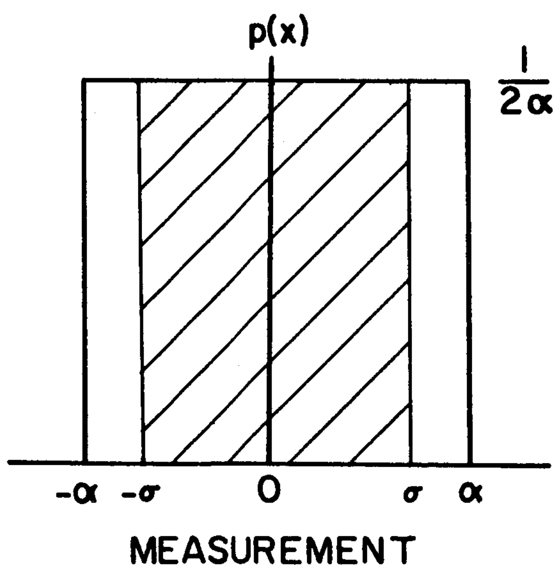


FIGURE 15.
DISTRIBUTION CURVE ILLUSTRATING THE SIGMA LIMITS

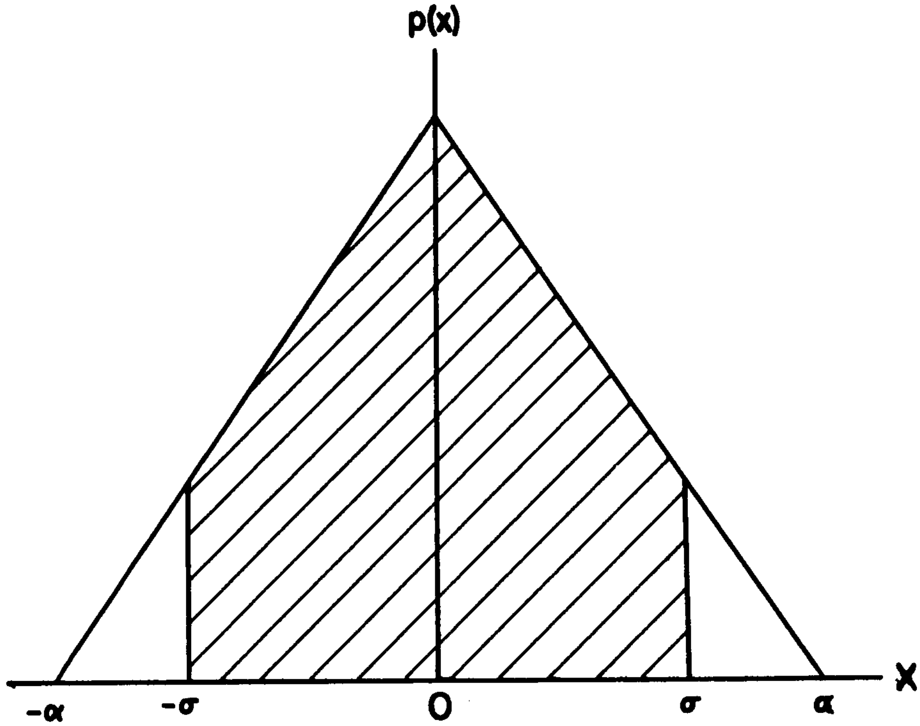


FIGURE 16.
TRIANGULAR PROBABILITY DENSITY CURVE

TABLE 1

PROBABILITY DENSITY VALUES CORRESPONDING TO FIGURE 16

$$-\infty < x < -\alpha \quad , \quad p(x) = 0$$

$$-\alpha < x < 0 \quad , \quad p(x) = \frac{x}{\alpha^2} + \frac{1}{\alpha}$$

$$0 < x < \alpha \quad , \quad p(x) = \frac{-x}{\alpha^2} + \frac{1}{\alpha}$$

$$\alpha < x < \infty \quad , \quad p(x) = 0$$

A mathematical calculation of the mean, $\bar{x}$, variance, σ^2, standard deviation, σ, and probability, Pr, follows.

$$\bar{x} = \int_{-\alpha}^{\alpha} x\, p(x)\, dx = \int_{-\alpha}^{0} x \left(\frac{x}{\alpha^2} + \frac{1}{\alpha}\right) dx + \int_{0}^{\alpha} x \left(\frac{1}{\alpha} - \frac{x}{\alpha^2}\right) dx \qquad (10a)$$

$$\bar{x} = \left(\frac{x^3}{3\alpha^2} + \frac{x^2}{2\alpha}\right)\Bigg|_{-\alpha}^{0} + \left(\frac{x^2}{2\alpha} - \frac{x^3}{3\alpha^2}\right)\Bigg|_{0}^{\alpha} \qquad (10b)$$

$$\bar{x} = 0 - \left(-\frac{\alpha^3}{3\alpha^2} + \frac{\alpha^2}{2\alpha}\right) + \left(\frac{\alpha^2}{2\alpha} - \frac{\alpha^3}{3\alpha^2}\right) = 0 \qquad (10c)$$

$$\sigma^2 = \int x^2\, p(x)\, dx = \int_{-\alpha}^{0} x^2 \left(\frac{x}{\alpha^2} + \frac{1}{\alpha}\right) dx + \int_{0}^{\alpha} x^2 \left(\frac{1}{\alpha} - \frac{x}{\alpha^2}\right) dx \qquad (11a)$$

$$\sigma^2 = \left(\frac{x^4}{4\alpha^2} + \frac{x^3}{3\alpha}\right)\Bigg|_{-\alpha}^{0} + \left(\frac{x^3}{3\alpha} - \frac{x^4}{4\alpha}\right)\Bigg|_{0}^{\alpha} \qquad (11b)$$

$$\sigma^2 = 0 - \frac{\alpha^4}{4\alpha^2} + \frac{2\alpha^3}{3\alpha} - \frac{\alpha^4}{4\alpha^2} = \frac{\alpha^2}{6} \qquad (11c)$$

$$\sigma = \frac{\alpha}{\sqrt{6}} \qquad (12)$$

$$Pr = \int_{-\sigma}^{\sigma} p(x)\, dx = \int_{-\alpha/\sqrt{6}}^{0} \left(\frac{x}{\alpha^2} + \frac{1}{\alpha}\right) dx + \int_{0}^{\alpha/\sqrt{6}} \left(\frac{1}{\alpha} - \frac{x}{\alpha^2}\right) dx \qquad (13a)$$

$$Pr = \left(\frac{x^2}{2\alpha^2} + \frac{x}{\alpha}\right)\Bigg|_{-\alpha/\sqrt{6}}^{0} + \left(\frac{x}{\alpha} - \frac{x^2}{2\alpha^2}\right)\Bigg|_{0}^{\alpha/\sqrt{6}} \qquad (13b)$$

$$Pr = 0 - \frac{1}{12} + \frac{1}{\sqrt{6}} + \frac{1}{\sqrt{6}} - \frac{1}{12} - 0 = 0.64983 \qquad (13c)$$

Consideration of a Gaussian (or Normal) Distribution Curve completes the comparison of distribution curves. The Gaussian distribution curve is symmetrical, therefore, $\bar{x}$ is zero.

Since integrating the normal distribution probability function is difficult,

$$p\,(x) = \frac{e^{(-x^2)/(2\sigma_n^{\,2})}}{\sqrt{2\pi\sigma_n^{\,2}}} \tag{14}$$

values of the probability integral, Pr, have been tabulated. Mathematical handbooks (14, 15) contain tables that enable a rapid determination of the probability:

$$Pr = \int_{-\sigma}^{\sigma} p\,(x)\,dx = \int_{-\sigma}^{\sigma} \frac{e^{-\frac{x^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}}\,dx = \int_{-1}^{1} \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}}\,dx = 0.6826 \tag{15}$$

Comparison of the three probabilities just calculated, 0.577 for the square distribution, 0.64983 for the triangular distribution, and 0.6826 for the Gaussian distribution, indicates that although three quite different probability curves are investigated, the resulting probabilities are very similar. This result, combined with the discovery that instrumentation systems are best described by the Gaussian distribution, further justifies choosing the Gaussian distribution to describe the statistical characteristics of instrumentation systems.

The Gaussian Distribution is the most significant (for measurement analysis) of the many distribution curves that have been investigated by the science of statistics. The Gaussian distribution is based on the assumption that the total deviation of a measured quantity, x, from some central value, $\bar{x}$, results from a large number of small perturbations. If there are m contributions to the total deviation and each is of equal magnitude, $\pm$ a, then the maximum range of readings is $\bar{x} \pm$ ma if all deviations happen to be either positive or negative and in the same direction (8). In a random summation of positive and negative quantities, the most probable sum is zero, indicating that the most common values of x are in the vicinity of $\bar{x}$. A distribution curve that describes the above example is symmetric, peaked at the center, and declines smoothly to zero at $x = \bar{x} + ma$ and $x = \bar{x} - ma$.

A curve for the limiting case of an infinite number of infinitesimal contributions to the total deviation in shown in Figure 8. The mathematical equation describing the Gaussian distribution is

$$y = Ce^{-h^2 (x - \bar{x})^2} \tag{16}$$

C is a measure of the maximum height (note that $y = C$ when $x = \bar{x}$) and h governs the curve width. A large h results in a high, narrow curve, while a small h produces a low, broad curve (8). The standard deviation, σ, is defined by

$$\sigma = [\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2]^{1/2} \tag{17}$$

where N is the maximum number of data points. Sigma is related to h by the following equation

$$\sigma = \frac{1}{h\sqrt{2}} \tag{18}$$

All previous discussions adequately describe the theoretical nature of a Gaussian distribution curve; however, there is need for a correspondence between the normal distribution and actual observations.

Since the Gaussian distribution is a continuous curve, it can correspond only with a limiting case as the number of observations tends to infinity. This rather limiting requirement is the first point of difference between the Gaussian distribution curve and any actual set of observations. A second characteristic of the normal distribution which at first appears undesirable is that in each reading there is assumed to be an infinite number of infinitesimal perturbations, each of which is likely to be positive or negative.

To solve the two difficulties mentioned in the previous paragraph, carefully consider the nature of instrumentation systems. The discussion concerning measurements indicated that errors found in instrumentation systems were extremely numerous, originated in a random fashion, and were small in magnitude. A large number of experiments have yielded a Gaussian distribution, and all sources appear to indicate that measurements are best described by a Gaussian distribution (8, 12).

The two preceeding paragraphs indicate a very close correlation between characteristics of instrumentation measurements and the Gaussian distribution. In fact, the present knowledge of statistical theory indicates no better approximation than the Gaussian distribution for describing instrumentation data (8).

The real purpose of the statistical analysis discussion just presented was to aid in establishing the degree of confidence in measurements recorded at the Combustion Research Laboratory. The topics of "best value" and "uncertainty" of a measured quantity were introduced so that experimental test data (such as pressure P_1) could be analyzed and assigned a definite value, $P_1 \pm \varepsilon$, including a confidence level ε.

3.6 Applications of Statistical Theory To Curve Fitting Procedures

3.6.1 Introduction

Application of Statistical Theory to the subject of this investigation involves curve fitting procedures. The term "curve fitting" implies the process of representing a succession of related data points by means of a mathematical equation. An example of curve fitting is deriving an equation to describe the relationship between output voltages from a transducer and corresponding pressures applied to the transducer. In most cases, transducer manufacturers try to construct a sensing device that has a very nearly linear pressure-voltage relationship; consequently, the following discussions are concerned with linear data analysis.

The ideal approach to calibrating a transducer is to apply the same value of a parameter (e.g. pressure) to the device many different times, and record the output voltage for each pressure application. After collecting a large number of voltage output values for a given pressure input, a distribution curve could be constructed and a "best value" in addition to an "uncertainty" could be calculated as described in the statistical analysis section. A significant number of different pressures within the range in which the instrument is to be used must be analyzed statistically. Although most of the mathematical calculations involved could be handled by a digital computer, the manual labor required to collect the necessary data renders this approach impractical. The only alternative is to determine a quick, approximate method of establishing a calibration curve given only one measurement each for several different values of a parameter (pressure in the example cited above), within the desired range.

3.6.2 Establishment of Calibration Curves

Four methods of establishing a calibration curve are:

i) The Graphical Method

ii) The Method of Sequential Differences

iii) The Method of Extended Differences

iv) The Method of Least Squares

(i) Graphical Method

This technique involves plotting the collected data on graph paper and drawing what appears to be the best straight or curved line through the data points. Such a method, since it depends on human judgment, is not as desirable as other methods available. Individual prejudices affect the final curve position and, since a great deal of effort has been expended in collecting accurate data, employing such a gross approximation in the final step is hardly justified (8). Although the amount of error involved in obtaining measurements may have been carefully controlled, there is absolutely no way of determining the final error if the aforementioned method is used to draw the final calibration curve.

(ii) Method of Sequential Differences

As the name implies, this method involves consideration of data points in sequential order as a means of obtaining the final calibration curve. Here, adjacent data points are considered in turn, and the slope between each pair of points is calculated. If n measurements have been collected, then n-1 estimates of slope are possible. From the n-1 estimates ofslope, the mean slope is calculated and taken as the true slope of a straight line.

An obvious fault with the method of sequential differences is that the value of the average slope is based entirely on the first and last data points recorded. For constant independent variable intervals, Δx, the sum of the differences eliminates all values of the dependent variable, y, except the first and last values. For example, in Table 2 note that all values of $(\frac{\Delta y}{\Delta x})_n - (\frac{\Delta y}{\Delta x})_{n-1}$ except the first and last can be discarded and the results are obviously unchanged (12). Effectively, the method of sequential differences implies that the best straight line is that joining the first and last data points.

Slope calculation using sequential differences:

$$\sum \left[\left(\frac{\Delta y}{\Delta x}\right)_{n+1} - \left(\frac{\Delta y}{\Delta x}\right)_n \right] = \frac{5}{2} \qquad (19)$$

TABLE 2

SLOPE CALCULATION BY SEQUENTIAL DIFFERENCES

x	y	Δx	Δy	$\frac{Y_n - Y_{n-1}}{\Delta x}$	Δy
0	0	1	2	2-0	2
1	2	1	3	5-2	3
2	5	1	2	7-5	2
3	7	1	2	10-7	3
4	10				
		4	10	10	10
				mean: $\frac{5}{2}$	mean: $\frac{5}{2}$

Slope calculation using only endpoint values:

$$\frac{y_{final} - y_{initial}}{x_{final} - x_{initial}} = \frac{10-0}{4-0} = \frac{5}{2} \qquad (20)$$

In reality, the true slope should not be based on endpoint values since the first and last data points are very often the most questionable ones. A much more satisfactory approach is to employ the intermediate readings which are less likely to be affected by systematic errors. This is done by using the method of differences to handle non-sequential points in a systematic manner.

iii) Method of Extended Differences

For this approach, the data points are separated into two equal groups, high and low values of X, the independent variable. Corresponding points in the two groups are subtracted, as illustrated in Table 3.

Calculating the standard error of the mean slope involves subtracting all estimates of the slope (column 5 of Table 3) from the mean slope.

$$\text{Mean slope} = 11.965 \div 6 = 1.995 \qquad (21)$$

The mean value of data in column 6 is σ, or 0.022. The standard error is defined below, where N is the number of data pairs employed in the calculation (12).

TABLE 3

SLOPE CALCULATION BY METHOD OF EXTENDED DIFFERENCES

x	y	$\Delta X = X_{n+6} - X_n$	$\Delta y = y_{n+6} - y_n$	$\frac{\Delta y}{\Delta x}$	$\left\lvert \frac{\Delta y}{\Delta x} - 1.995 \right\rvert$
0	0.0	6 = 6-0	12.2=12.2-0.0	2.03	.035
1	2.0	6 = 7-1	12.0=14.0-2.0	2.00	.005
2	4.2	6 = 8-2	11.6=15.8-4.2	1.934	.061
3	6.1	6 = 9-3	11.9=18.0-6.1	1.985	.010
4	8.0	6 = 10-4	12.1=20.1-8.0	2.016	.021
5	10.0	6 = 11-5	12.0=22.0-10.0	2.000	.005
6	12.2				
7	14.0		Sum:	11.965	Sum: 0.137
8	15.8		Mean:	1.995	Mean: 0.02285
9	18.0				
10	20.1				
11	22.0				

$$\text{Standard Error} = \frac{5}{4}\ |\ \sigma\ |\ \left(\frac{1}{\sqrt{n-1}}\right) = \frac{5}{4}\ \frac{(.022)}{\sqrt{6-1}} = .0123 \tag{22}$$

The intercept of the calibration curve must be determined also. This calculation involves finding a point through which the line must pass, or will most likely pass. The centroid of all points is the best point to use for fixing the line:

$$\bar{x} = \frac{1}{n} \sum x_i = 5.45 \tag{23}$$

$$\bar{y} = \frac{1}{n} \sum y_i = 11.05 \tag{24}$$

Essentially, the method of extended differences is equivalent to connecting the centroid of the data points lying on one half of the abcissa (or ordinate) with the centroid of data points lying on the opposite half of the scale. The method of extended differences is useful when errors are likely to occur in both the dependent and independent variables. However, if all errors definitely occur in only one of the variables, then the method of least squares is preferred.

(iv) Method of Least Squares

Several methods of determining a straight line through a set of data points are available. In cases where accuracy is not an important factor, simplified methods can be employed. In some cases a very accurate method of establishing the most representative line upon which experimental points lie is necessary. The method of least squares is, by far, the most accurate approach to determining a straight line; however, the required calculations are more complex than those mentioned previously.

Consider an array of data points plotted in a two dimensional space (i.e., the common x-y plane, Figure 17)[1]. Establishing a straight line that best represents the data shown in Figure 17 is initiated by assuming that all errors occur in the variable, y[2]. The least squares method assumes that the best straight line is the one which minimizes the sum of the squares of the dependent variable errors[3].

Any straight line has the following equation

$$y = mx + b \tag{25}$$

For a particular point, i, the equation is

$$y_i = mx_i + b \tag{26}$$

If an experimental point corresponding to x_i falls off the calculated straight line by a distance ε_{y_i} (the error in point i), then:

---

1. The circled points will be discussed in Section 6.2.8.

2. All errors occur in the y-variable since x represents the reference standard such as pressure from a dead weight tester.

3. Refer to Whittaker and Robinson (16) or Baird (8) for a more thorough explanation of this assumption.

TABLE 4. CALIBRATION DATA

PRESSURE (psi)	TRANSDUCER OUTPUT (volts)
500	.100
1000	.200
1500	.320
2000	.400
2500	.500
3000	.600
3500	.715
4000	.800
4500	.900
5000	1.000

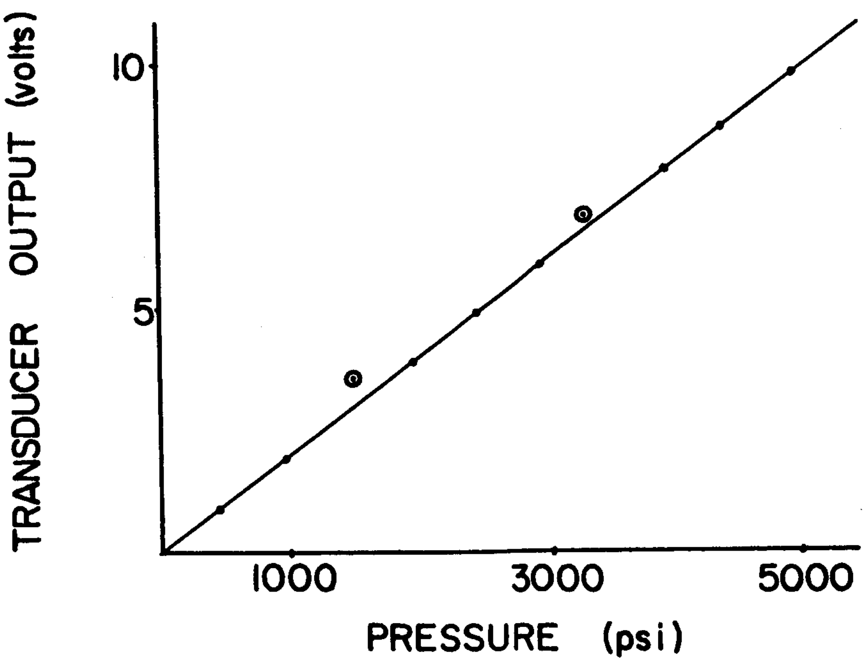


FIGURE 17. CALIBRATION BY LEAST SQUARES

$$y_i = mx_i + b + \varepsilon_{y_i} \tag{27}$$

As stated in the preceeding, the theory of least squares implies that the best straight line through a group of points is the one which minimizes the sum of the squares of the errors. The expression is stated as follows:

$$\sum (\varepsilon_{y_i})^2 = \text{minimum} \tag{28}$$

Rewriting Equation 27 and combining with Equation 28 to solve for the cumulative square error, $\sum(\varepsilon_{y_i})^2$:

$$\sum_{i=1}^{n} (\varepsilon_{y_i})^2 = \sum_{i=1}^{n} (y_i - mx_i - b)^2 \tag{29}$$

To minimize the square error, differentiate the right side of Equation 29 with respect to m and b, then equate the result to zero, giving two conditions to be specified:

$$\sum x_i(-2)\ (y_i - mx_i - b) = 0 \tag{30}$$

$$\sum (-2)\ (y_i - mx_i - b) = 0 \tag{31}$$

In order to apply the conditions of Equations 30 and 31 to an array of data, write the original n data points as n equations similar to Equation 26 (these are called equations of conition, column 3 in Table 5). Multiply each of the equations of condition by its corresponding value of x (column 4 of Table 5). Then add the equations of condition (column 3) and add the equations of condition times x, (column 4). Both summation equations (Equations 32 and 33) can be solved simultaneously for m, the slope, and b, the y - intercept.

$$5.535 = 27{,}500 + 10b \qquad (32)$$

$$19{,}330 = 96{,}250{,}000\, m + 27{,}500b \qquad (33)$$

$$m = 1.99 \times 10^{-4} \qquad (34)$$

$$b = 0.0065 \qquad (35)$$

$$y = 1.99 \times 10^{-4}\, X + 0.0065 \qquad (36)$$

In many cases the two summation equations are very similar; therefore, the roundoff error may be significant if calculations are not performed carefully. The "uncertainty" of a calculated value for y based on the equation just derived (Equation 36) is called the standard error. Table 6 illustrates standard error calculations.

$$\sigma_y = \text{Standard Error} = \sqrt{\frac{\sum (\epsilon_{y_i})^2}{n-2}} \qquad (37)$$

TABLE 5

LEAST SQUARES EQUATION CALCULATIONS

x	y	Equations of Condition	x·(Equations of Condition)
500	.100	.100 = 500 m + b	50= 250,000 m + 500b
1000	.200	.200 =1000 m + b	200=1,000,000 m + 1000b
1500	.320	.320 =1500 m + b	480=2,250,000 m + 1500b
2000	.400	.400 =2000 m + b	800=4,000,000 m + 2000b
2500	.500	.500 =2500 m + b	1250=6,250,000 m + 2500b
3000	.600	.600 =3000 m + b	1800=9,000,000 m + 3000b
3500	.715	.715 =3500 m + b	2500=12,250,000 m + 3500b
4000	.800	.800 =4000 m + b	3200=16,000,000 m + 4000b
4500	.900	.900 =4500 m + b	4050=20,250,000 m + 4500b
5000	1.000	1.000 =5000 m + b	5000=25,000,000 m + 5000b
		5.535=27,000 m + 10b	19,330=96,250,000 m + 27,500b

TABLE 6

STANDARD ERROR CALCULATIONS

x	mx	$y_{calc}=mx+b$	y_{exp}	$\lvert\varepsilon_y\rvert$	ε_y^2
500	.0995	.1060	.100	.0060	.00003600
1000	.1990	.2055	.200	.0055	.00003025
1500	.2985	.3050	.320	.0150	.00022500
2000	.3980	.4045	.400	.0045	.00002025
2500	.4975	.5040	.500	.0040	.00001600
3000	.5970	.6035	.600	.0035	.00001225
3500	.6965	.7030	.715	.0120	.00014400
4000	.7960	.8025	.800	.0025	.00006250
4500	.8955	.9020	.900	.0020	.00004000
5000	.9950	1.0015	1.000	.0015	.00002250
				Sum:	.00060875

$$\sigma_y = \sqrt{(0.00060875)/10} = 7.795 \times 10^{-3}$$

In Equation 37, n is the number of data points collected, and ε_{y_i} is the error in the dependent variable, y.

Using the straight line equation determined previously (Equation 36), values of the dependent variable are calculated and compared with the experimental data points. The difference between a calculated dependent variable, y_{calc}, and the experimental value, y_{exp}, is ε_y.

A standard error for the slope, m, and intercept, b, can also be determined. Knowledge of the slope or intercept standard error is important if one is interested in comparing these parameters for different sets of similar data[1].

$$\text{Standard Error for Slope, } \sigma_m = \sigma_y \sqrt{\frac{n}{n\sum x_i^2 - (\sum x_i)^2}} \tag{38}$$

$$\text{Standard Error for Intercept, } \sigma_b = \sigma_y \sqrt{\frac{\sum x_i^2}{n\sum xi^2 - (\sum xi)^2}} \tag{39}$$

Since experiments to be conducted at the Combustion Research Laboratory may require collecting data for a very short time period (less than 1 minute), a statistical analysis of the data is not possible. Statistical analyses concerning instrumentation must be performed by considering calibration data which are generated by

1. The method of least squares has been programmed for an IBM 7094 digital computer. An explanation of the use of this program is presented in Section 6.2.8.

simulating experimental conditions (i.e. applying pressure to a transducer by means of a dead weight tester). For the immediate use of the data acquisition system, sufficient calibration data are not available for statistical analysis; therefore, assuming Gaussian statistical data is necessary.

A continuous statistical evaluation program of the instrumentation system is recommended. A suggested approach for future use of the data system is presented in Section 8.

4. TEST PROCEDURES

Preparation procedures for conducting an experiment or calibration at the Combustion Research Laboratory are extremely important, otherwise valuable data may be lost. The digital recording system is the most complex portion of the data system; therefore; it is discussed in the greatest detail. A step by step discussion of the manual functions required to operate the digital system are presented in the system operating manual (17). This manual must be consulted and followed carefully before operating the digital system.

4.1 Identification Data

Writing a digital tape first requires recording at least one identification record on the tape itself. The identification record data areentered by means of the 12 thumbwheel switches located on the Computer Format Control Buffer. A MAP program[1], BCD, processes header[2] data consisting of either two-, four-, or six-digit groups.

---

1. To be explained in Section 5.

2. The terms header data and I.D. are synonymous.

4.1.1 Preliminary Header Information

At least one record or block of identification data must be entered each time a recording is made. This first block contains the date, number of channels in use on the digital system, and up to 196 more digit pairs. The last 196 decimal digit pairs, PRE(5) to PRE(200) inclusive, are intended to correspond to format statements in a subroutine called PRELIM, which is an abbreviation for preliminary data. The detailed operation of PRELIM is explained in Section 6. Briefly, PRELIM serves to identify non-numerical data such as the operator's name, the type of fuel, oxdizer, and cooling employed by an experiment. For example, ID(6)[1] which is entered by means of the 11th and 12th thumbwheel switches is presently capable of specifying either an uncooled, a film cooled, a transpiration cooled, or a regeneratively cooled engine through use of the following logical - IF (5,6) and format statements:

```
C......ID(6) = COOLING DESIGNATION
       IF ( ID(6) .EQ. 0 ) WRITE(6,20)
       IF ( ID(6) .EQ. 1 ) WRITE(6,21)
       IF ( ID(6) .EQ. 2 ) WRITE(6,22)
       IF ( ID(6) .EQ. 3 ) WRITE(6,23)
    20 FORMAT(5X, 10H UNCOOLED //)
    21 FORMAT(5X, 12H FILM COOLED //)
    22 FORMAT(5X, 21H TRANSPIRATION COOLING //)
    23 FORMAT(5X, 21H REGENERATIVE COOLING //)
```

---

1. The name PRE(n) designates preliminary data in the main program. Subroutine PRELIM designates ID(n) as preliminary data, therefore PRE(n) corresponds to ID(n) throughout this paper.

4.1.2 General Header Information

After recording the preliminary data (PRELIM) block, additional identification data can be entered as the digital system operator desires. Subroutine BCD processes[1] binary coded decimal data and returns the processed data to the calling (usually main) program where the data can be employed in a Fortran program to satisfy the programmer's needs. A brief explanation of one such Fortran program, PRELIM, was discussed in the last paragraph . Anticipated data acquisition system requirements indicate a need for BCD programs capable of processing I.D. in four- and six-digit groups in addition to the two-digit groups described above.

Using header information requiring processing by BCD is optional; however, entering identification data by means of thumbwheel switches is an important capability of the data system which should be employed as much as possible.[2] Two example Fortran programs have been written to demonstrate the use of four-digit and six-digit I.D. to identify either pressure transducers or thermocouples.

A subroutine entitled PTRANS presently accepts six-digit I.D.

---

1. The exact nature of processing will be described in Section 5.
2. Identification data required by PRELIM must be entered.

to identify a pressure transducer number and a calibration range. Since two six-digit numbers can be dialed in with one setting of the 12 thumbwheel switches, the first six digits represent a transducer number and the last six represent the corresponding pressure range. Entering another group of 12 digits defines a second transducer and so on. Since the number of digits that BCD will accept is variable[1], nearly any number of transducers can be specified in one record.

PTRANS is a sample of the BCD data capability available to the data acquisition system users. Since output from BCD enters a Fortran program (the calling program), any desirable Fortran program can be written that utilizes six decimal digit data.

Another subroutine, THERM, is capable of accepting four-digit data. Recall that BCD decodes binary coded decimal data. If the last argument of the "CALL BCD" statement is 4, the BCD subroutine divides the stored data into four-digit numbers; therefore, each entry with all 12 thumbwheel switches represents three numbers. Input to THERM is printed out in groups of three numbers, each containing four decimal digits. THERM presently writes the three numbers in the following respective order: thermocouple number, minimum temperature, and maximum temperature. Again, this Fortran program is only one

---

1. The digital system limit is approximately 1000 decimal digits per record because the Computer Format Control Buffer storage area has capacity for 1024 characters.

possible manner in which four-decimal-digit I.D. data are utilized.

The number of digits employed to represent one item of header data must be consistent within every record. For example, all data in a two-digit I.D. record must contain two-decimal digits per input entry. If an entry containing more than two decimal digits is desired, another record must be begun, and all entries in one optional BCD record must contain four characters if the last argument of CALL BCD is 4, or six characters if the last argument is 6.

Several sample header data records and their corresponding computer subroutines are illustrated in Appendices C and D.

After recording all desired BCD data, the digital system controls must be prepared to record experimental data from sensing devices. Experimental data are recorded in binary code, except for the time data which are BCD. Once the digital system begins recording data, header information cannot be entered until the experiment is completed. After completing the experiment, more BCD data can be entered if desired.

The very last operation after completing a recording is to write a file mark on the tape by depressing the "Write File Mark" button located on the Computer Format Control Buffer. The file mark is necessary in order to signal the computer to stop reading tape, otherwise the entire reel would be read by the 7094, which is an expensive and unnecessary operation.

4.2 Calibration

Before conducting an experiment at the Combustion Research Laboratory, each sensing device must be calibrated. Since input to the digital subsystem is an analog voltage signal, and output on the digital tape is a specific digitized voltage, a means of obtaining data in engineering units (pressure, temperature, force, flow rate) is necessary. A calibration procedure is therefore required to relate the voltage signal data to a value in engineering units.

Two means of calibrating sensing devices are end to end calibration and single point calibration. Single point calibration serves as a quick check of a sensing device's electrical characteristics just prior to conducting an experiment. End to end calibration is employed to calibrate a sensing device throughout its anticipated range of operation by applying the appropriate input to the sensing device.

4.2.1 End to End Calibration

One end to end calibration method involves performing a calibration procedure employing a standard traceable to the Bureau of Standards such as a dead weight tester. Another method involves the use of calibration data from manufacturers such as in the case of the Potter turbine flow meter.

End to end calibration of a pressure transducer first requires the transducer (or transducers if several are to be calibrated within the same pressure range) to be attached to a dead weight tester manifold, located at the experimental test site. All electrical connections that are present during an experiment should be included within the circuit in order to calibrate under conditions resembling an actual test as nearly as possible.

The Number of Desired Calibration Points are then determined, depending on the range of operation, the data already available for the transducer under consideration, and the desired accuracy. Investigations have indicated that 20 calibration points should be employed with the calibration subroutine (LINEAR, see Section 6.2.8) which was based on the statistical analysis investigation.

Amplifier Gain is first adjusted for the range of voltages anticipated with each sensing device. For pressure transducers, after applying a pressure to the transducers with a dead weight tester, the strain gage bridges are then adjusted until the analog to digital converter nixie tubes display a number that equals the input pressure. If several different pressure transducers are employed, each having different pressure ranges, the dead weight tester must be readjusted for each transducer.

Identification Data are recorded just prior to recording end to end calibration data. Each calibration run must be preceeded by the data for PRELIM. Additional header information describing sensing devices, pressure and temperature ranges, or maximum values can be entered in either two, four or six decimal digit groups as described previously.

Calibration Data are recorded after placing all necessary identification characters onto the tape. The dead weight tester is loaded and the multiplexer completes one scan of the channels being calibrated. The dead weight tester is then loaded to the next calibration pressure value and another complete scan of the channels is recorded. This process is repeated until all calibration points have been recorded.

Another two-, four-, or six-character "Calibration Identification Data Block" (or blocks) can then be recorded for a second set of sensing devices. (These will probably have a different range of operation than the first set.)

In summary, the end to end calibration consists of the following six steps, which are illustrated in Figure 18.

a. Divide the pressure range into 10 or 20 increments.

b. Set the amplifier gain to the range of voltages anticipated.

c. Manually dial in any desired I.D. information with the thumbwheel switches located on the Computer Format Control Buffer, and record the identification data on tape.

d. Apply a pressure of established accuracy (by means of a dead weight tester).

e. Record the voltage output with the digital tape recorder.

f. Repeat steps c, d, and e.

The computer than reads the data tape, prints I.D. information, and a calibration curve is calculated for the transducers being

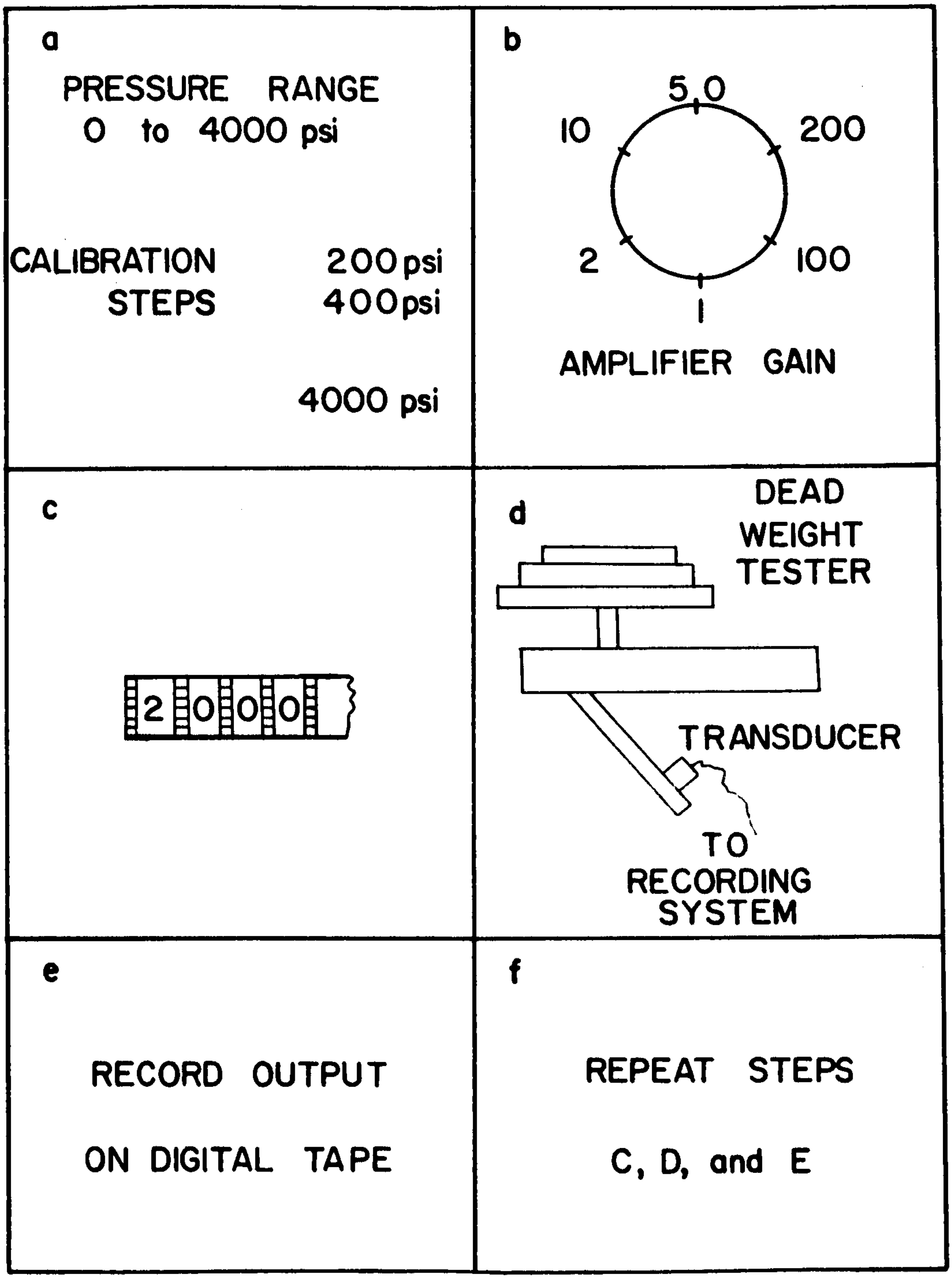


FIGURE 18. END TO END CALIBRATION STEPS

considered. Explanations concerning the calibration curve programs are presented in Section 6.

The second method of end to end calibration utilizes published data from the manufacturer. Thermocouples in general are calibrated from published data utilizing certified wire because there is no simple inexpensive wide range temperature standard of comparison as there is for pressure (i.e., a dead weight tester).

Manufacturers of thermocouple wire generally supply tables of calibration data and a certification with each roll. The tables are usually furnished in two forms, millivolts to degrees and degrees to millivolts. The tabulated data can be handled in two ways. First, use of a computer program which interpolates between data points enables one to obtain the temperature corresponding to any output voltage from the thermocouple. Tabulated data is read into the main computer program and transferred to the subroutines described in Sections 6.2.6 and 6.2.7. The temperature corresponding to any voltage can be obtained by specifying the desired voltage. Second, the tabulated data can be read into the least squares program described in Section 6.2.8.

4.2.2 Single Point Calibration

The purpose of single point calibration is to check the validity of end to end calibration data. Since an end to end calibration process is time consuming, it will be performed at intervals of one or two months, when the single point calibration results indicate the need for recalibration, or when experimental results are suspect.

Single point calibration data must be recorded before and after each experimental test and end to end calibration. Comparison of single point calibration with end to end calibration data serves to determine the bias that has arisen in the instrumentation system and whether a sensing device has become defective or inoperative. By performing an end to end calibration before conducting an experiment, the chance of losing valuable data due to an inoperative or faulty sensor is greatly reduced. The single point calibration following an experimental test is compared with the pre-test calibration as an indication of transducer drift which might indicate damage during the course of an experiment.

A single point calibration procedure is generally conducted at 40% or 50% of full scale. The exact point is arbitrary, however, for reference purposes the same point should be selected for a particular sensing device until another end to end calibration is performed.

The single point calibration process consists of replacing the transducer with a calibrated resistor. The resistor is placed in parallel with the transducer. Adjustment of the variable resistor simulates various loaded transducer conditions. All transducers to be employed in an experiment are replaced by resistors and adjusted before recording calibration data. A precision signal is applied to thermocouple circuits for single point calibration.

Simulated output is recorded by one complete multiplexer scan of all channels. The single point calibration data are then compared with end to end calibration data to determine the calibration validity. If the deviation between single point calibration and end to end calibration data is slight, the deviation can be used to adjust the end to end data for bias. If the deviation is large, another end to end calibration should be performed.

The single point calibration procedure is outlined below and illustrated in Figure 19.

a. Insert a calibrated resistor into the transducer circuit in place of the transducer (insertion is accomplished through switching.

b. Adjust the resistance to simulate a loaded transducer condition.

c. Repeat steps a and b for all transducers to be employed in the experiment.

d. Energize the transducer circuits and record the voltage output from each channel by one complete multiplexer scan.

e. Compare the calibration point with the end to end calibration data.

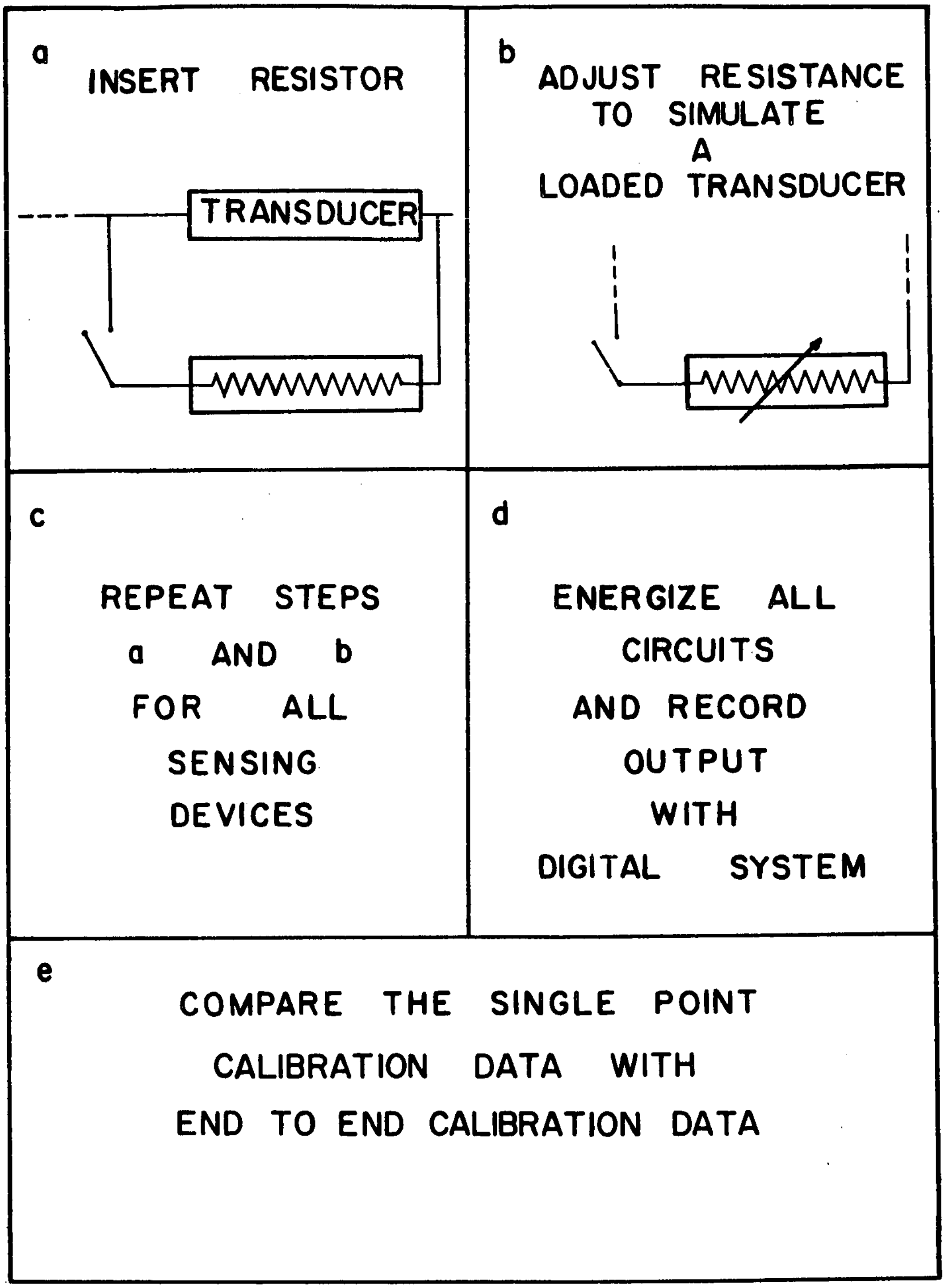


FIGURE 19. SINGLE POINT CALIBRATION STEPS

4.3 Block Length

There are several restrictions regarding the number of characters contained in one record. Since each data sample requires three tape frames (or characters), the number of channels in use on the digital system is multiplied by three to obtain the number of frames per multiplexer scan, and this is multiplied by the desired number of multiplexer scans per data record. Six must be added to the total number of characters to account for the time word.

(Number of Channels in Use)

x (3)

x (Number of Scans Per Record)

+ 6

= (Number of Characters Per Record)

At the maximum sampling rate (4000 samples per second) the maximum block length is approximately 900 characters, therefore, if 20 channels are being used, no more than 14 scans per record are possible (15 scans would exceed the 900 character limit):

$$20 \times 3 \times 14 + 6 = 846$$

The number of characters per record (846) must be dialed with the thumbwheel switches located on the upper right side of the Computer Format Control Buffer. The allowable block lengths for various sampling rates are illustrated in Figure 20. Safe operation results if the block length is confined to the values below the 800 bpi line (or 556 bpi line) and to the left of the 1024 line.

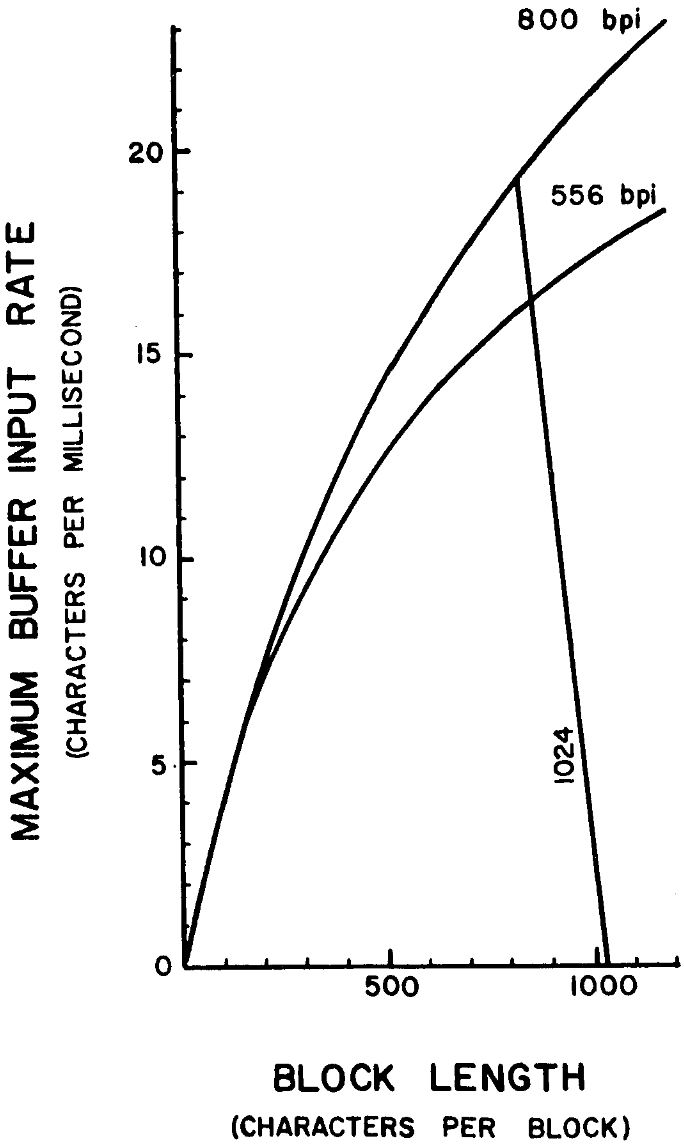


FIGURE 20. ALLOWABLE BLOCK LENGTHS

If the number of characters is not entered correctly (i.e., 845 rather than 846), a parity error indication will result. The 10CS13 subroutine[1] will detect the number of IBM words in each record and the main program will print this number. As an aid to debugging the system, recall that one IBM word contains six characters; therefore, the number of characters dialed on the CFCB should be three times greater than the number of IBM words printed by the main program[2].

For BCD data, the block length simply consists of the number of decimal digits entered with the 12 thumbwheel switches. The buffer size (1024 characters) establishes the maximum block length.

---

1. Section 5.2.3.

2. Each input IBM word generates two output data words; therefore, the factor above is three rather than six.

4.4 Recording Data

Experimental data and calibration data are recorded in the same manner (i.e. binary code); the basic procedures employed are identical. First, the PRELIM block must be recorded, next any other desired BCD records are recorded, then binary experimental or binary calibration data are recorded. Finally, a tape mark is written to signify the end of a recorded section which enables the computer to sense a desired stopping point.

Obtaining a satisfactory understanding of the data acquisition system requires detailed knowledge of the computer programs employed. Before describing the programs, basic programming concepts of the computer languages (MAP and Fortran) are discussed.

5. MAP PROGRAMMING

5.1 General Concepts

5.1.1 Introduction

MAP (Macro Assembly Program) combines the flexibility and versatility of machine language with symbolic instructions that are meaningful to a programmer. Unlike pure machine language that consists of binary code, MAP employs symbols such as ADD for addition and MPY for multiplication.

Writing a Fortran program in MAP generally requires more instructions since each mathematical operation must be programmed separately. The Fortran statement, A = B * C * D, requires a separate computer card for each multiplication when programmed with MAP. Although MAP requires more programming effort, the programs are more efficient since less translation by the computer from source program to object program[1] is required. Many operations

---

1. The source program is punched on computer cards and then translated into the object program by the computer. Actual operations within the computer are controlled by the object program.

such as shifting bits, unpacking data, and other basic[1] computer operations are available in MAP but not in Fortran.

Since most engineers are not generally familiar with the MAP language, a detailed description of all MAP subroutines employed by the data acquisition system is presented in this section. The reader should consult References 2, 3, 4, and 7 for aid in programming with MAP.

Before describing specific MAP subroutines in detail, an understanding of the basic computer operations is necessary. Since MAP is a more basic language than Fortran, a programmer must become familiar with actual operations that are being performed by the computer.

5.1.2 Mathematical Operations

Addition and Subtraction are performed in conjunction with a machine component called the accumulator. The accumulator is a storage location containing 36 bits (sign and bits 1 to 35) plus two extra bits (P & Q), which are required by certain instructions but not by the programs now being discussed. When a programmer specifies a variable in Fortran such as, N2 = 35, the computer assigns the number 35 to a location which represents N2. There are 32,768

---

1. Basic computer operations comprise the simplest set of instructions which need no further translation by the computer.

locations or IBM "words" in the 7094 and each contains 36 bits.

Addition of two numbers is accomplished by placing one number into the accumulator (by means of a CLA pseudo-operation[1]) and adding the other number to the first (using an ADD pseudo-operation). The sum is located in the accumulator. Subtraction is accomplished in a similar manner, with the SUB pseudo-operation substituted for the ADD pseudo-operation.

Multiplication and Division require using both the accumulator and a device called the MQ register (multiplier-quotient register). The MQ contains 36 bits, (sign and bits 1-35) and is similar to the accumulator because it serves as a temporary working area for data while mathematical operations are being performed.

To multiply two numbers, M and N, one number is loaded into the MQ by means of the (LDQ M) pseudo-operation. Multiplication is performed by following the LDQ with a (MPY N) instruction. The result is a 70-bit product located in both the MQ and the accumulator. The 35 most significant bits are in the accumulator and the 35 least significant bits are located in the MQ register. For most operations the product consists of less than 35 binary digits; therefore, the entire product will be located in the MQ register, and the accumulator

---

1. Pseudo-operations are operations in MAP that are not pure machine instructions, but are required by the programming needs of the language.

will contain zeros.

Since the programs to be described in this paper include multiplication which always results in a product composed of fewer than 35 bits, the STQ pseudo-operation must be employed to store data following multiplication. An XCA then exchanges the contents of the accumulator and the MQ register. After executing the XCA, a product containing fewer than 35 bits will be located in the accumulator where addition or subtraction may then take place.

The aforementioned mathematical operations are in the fixed point mode[1], as are all mathematical operations performed by the MAP programs for the Combution Research Laboratory. MAP has no restriction concerning the use of I,J,K,L,M or N to begin a fixed point variable name as is the requirement with Fortran.

Although not employed by the MAP programs described in this report, floating point arithmetic is possible. Corresponding to the following fixed point instructions (ADD, SUB, MPY) are the floating point instructions (FAD, FSB, FMP). The capability of using floating point arithmetic in MAP has only been mentioned briefly should future use of the data system require floating point operations. For more detailed explanations

---

1. Fixed point mode refers to a type of computer mathematics which does not require decimal points. A decimal point is assumed to follow the last digit of a number (6).

concerning floating point arithmetic in MAP, References 2, 3, and 4 should be consulted.

5.1.3 MAP Instruction Cards

Each MAP instruction is punched on a separate IBM card. Symbolic instructions are formed by punching alphabetic, numeric, or symbolic characters on a separate IBM card. An instruction may have from one to five parts, each part occupying one field of the card.

The first of five fields on a MAP program card is the name field which extends from column 1 through column 6. The name field may not always be filled; in fact, a more desirable program results if the name field is used only when a reference must be made to a specific card in the program or when a name is required by the instruction. The name field may contain either all alphabetic[1], alphabetic and numeric[2], or symbolic[3] information.

The Operation Field, located in columns 8 through 15, must be filled. The alphabetic symbols contained in the operation field are pseudo-operations, which have been described previously[4].

---

1. A, ALPHA, BEGIN, LOOP
2. A23, ALPHA6
3. 123., 5.
4. Section 5.1.2

A complete list of the pseudo-operations required by the computer programs for the data acquisition system is presented in Appendix A. In addition, the list contains several important pseudo-operations which are not used presently, but which may be helpful if the programs are altered. A complete list of all available pseudo-operations for use with the IBM 7094 can be found in Reference 18.

The Variable Field is separated from the operation field by at least one blank column; it may begin in column 12 but may never begin beyond column 16. In general, the variable field begins in column 16. Since the variable field cannot extend beyond column 72, extension of a variable field to another card is possible if the ETC pseudo-operation is used.

The variable field may be vacant or may contain up to 3 subfields, separated by commas. The subfields are, in respective order, the address, tag, and decrement. Appendix A of Reference 2 indicates the subfields that are required, optional, or not permitted in the variable field of all 7090/7094 pseudo-operations. Appendix A of this report, which is a list of the machine instructions required by the Data Acquisition System, indicates the subfields which are necessary for each instruction.

The variable field of a pseudo-operation may contain symbols[1],

---

1. ALPHA, A, N23, 25, * *

symbolic expressions[1], and literals[2].

<u>The Address Subfield</u> indicates the location to which control is transferred after execution of the present instruction. The address is generally the name of an array (card #2 below) or another instruction (card #3).

```
GO    CLA   *-1
      STO   DATA
      TRA   GO
       .     .
       .     .
       .     .
       .     .
DATA  BSS   35
```

A symbol such as *-1, which appears above, is very frequently employed as an address. Rather than assigning a name to the instruction preceeding GO CLA *-1, the symbol *-1 in the address field transfers control to the instruction preceeding the present one. Similarly, *+5 refers to an instruction that is 5 cards beyond the instruction containing *+5 as an address.

---

1. ALPHA+3, *-4

2. Literals provide a simple means of entering data into a program. Decimal literals will concern us here. They consist of an equal sign followed by a decimal number (3). For example: =8, =152, =31.

The Tag Subfield, when it is present, contains a number from 1 to 7, inclusive. The tag refers to one of the seven computer index registers[1]. If a tag is present, the indicated index register is modified by the address of the current instruction, or the address of the present instruction is modified by the indicated index register. To clarify the preceeding statement, consider the following two examples.

First, to set an index register (i.e., specify the contents of the index register) the AXT instruction is generally employed. If 15 is to be stored into index register 2, the following instruction is necessary:

```
AXT      15,2
```

Here the index register, 2, is modified by the address, 15, of the AXT pseudo-operation.

Second, to modify an address by means of an index register, the proper index register must be set or modified, as described in the preceeding paragraph. Given index register #2, containing 15, the instruction below will add the contents stored in location DATA to the accumulator. The address portion of the instruction has

---

1. The 7094 has 7 index registers (abbreviated XR).

been decreased by the contents of the specified index register before execution of the instruction itself.

CLA DATA +15,2

<u>The Decrement Subfield</u> is generally used with transfer instructions such as TIX, TXI, TXH, TXL, and others (2,3,4). Only the TIX and TXI pseudo-operations are required by the programs being discussed in this paper; therefore, these will be described in detail.

<u>A Transfer on Index</u> (TIX Y,T,D) pseudo-operation causes control to be transferred to the location cited in the address portion, Y, if the contents of the specified index register, T, are greater than or equal to the contents of the decrement, D. For example, the instruction (TIX LOOP,3,10) will substract 10 from index register 3 (provided XR3 contains a number equal to or greater than 10) and transfer control to location LOOP. If the contents of XR3 are less than 10, the next sequential instruction is executed.

<u>A Transfer With Index Incremented</u> (TXI Y,T,D) pseudo-operation is very similar to the TIX instruction. With a TXI instruction XRT is increased by an amount equal to the decrement of this instruction and the computer unconditionally follows the address cited, Y.

<u>A Comments Field</u> is provided for the convenience of the programmer and does not affect execution of the program. The comments field is generally used at the programmer's discretion for

explanatory remarks. A blank must separate the comments field from the variable field. If no variable field is required, the comments field may begin in column 17.

5.2 MAP Programs

The three basic types of MAP subroutines required by the data acquisition system are classified according to the function which must be performed by each. The functions are tape reading, altering BCD data format, and altering binary data format. When reading Section 5, the respective computer programs (Appendix B) must be read simultaneously. Many helpful explanations are included in the comments field of MAP subroutines to aid the programmer.

5.2.1 IOCS13

Tape reading is accomplished by means of a MAP subroutine called IOCS13. The name IOCS13, although arbitrarily chosen, is derived from the Computer Input/Output Control System, Version 13, which controls tape reading and writing operations. A complete description of all available Input/Output Control System (IOCS) programming commands appears in Reference 7.

A program that enables tape reading is written in MAP[1] and

---

1. A tape can be called for reading in Fortran; however, the Fortran call actually refers to a MAP (or other machine language) subroutine that has been stored on a processing tape or in memory.

contains relatively few IOCS instructions or instruction groups (i.e., cards which are necessary to accomplish a given task). The IOCS commands required by the IOCS13 subroutine and the operation of this subprogram are described below. For information concerning additional IOCS instructions, consult Reference 7.

To transfer data from magnetic tape into the computer, data are entered into storage devices called buffers. Generally, data are transferred from an input tape to input buffers. After rearranging the data and locating it (i.e. separating or unpacking and storing in specified locations), the data are transferred to output buffers for writing on output tape if desired.

Two types of commands used by IOCS provide a programmer with a means of finding desired locations within input or output buffers, or a means of skipping over words in a buffer. The first type is called non-transmitting commands, the second, transmitting commands.

Non-transmitting commands provide a means of finding desired locations within input or output buffers and processing input data directly within a buffer.

Since transmitting commands are generally easier for the inexperienced programmer to understand, these were employed by the

IOCS13 subroutine. Transmitting commands move data words[1] either from an input buffer (or buffers) to working storage, or from working storage to an output buffer. During processing, buffers associated with one file[2] are connected together and presented in sequence for processing; therefore, each file can be treated as a unit or continuous string of words.

A complete understanding of the IOCS13 subroutine is most effectively attained if the function of each instruction card is described. Since IOCS13 is a subroutine, the first card in this deck is the $IBMAP control card. This card is analogous to the $IBFTC card which preceeds Fortran subroutines. A deck name containing no more than 6 alphanumeric characters[3] must begin in column 8.

The card following $IBMAP IOCS13 is a FILE card which describes the nature of input data written on the tape. IN1 is the name

---

1. A data word is defined as one IBM word or 36 binary digits (12 octal digits).

2. A file is any group of data enclosed by 2 file marks (3 1/2 inch gaps) on magnetic tape.

3. "Alphanumeric character" is an IBM term that signifies either alphabetic or numeric characters. The first character of a variable name must be alphabetic:
EX: IOCS13, AREA1, N4X.

assigned to the input data file[1], FILE is a pseudo-operation, and INPUTFILE describes the tape as input information for the 7094.

A(1) specifies the tape unit on which the data reel is mounted. This particular designation is peculiar to the specific 7094 computer being employed.

INPUT designates the file as an input file, and BLK=n describes the maximum block length to be encountered on the input file. If the block length is exceeded, the data in excess of n IBM words will be lost, however; if a block contains less than n words, no data are omitted.

HIGH specifies the density option, which is 800 characters per inch. Since the data acquisition system is capable of recording at either 556 or 800 characters (or bits) per inch, the seventh option in the variable field may be either 556 or 800.

BIN indicates that a tape written in binary code is to be read. This option must always be used although I.D. is coded in BCD.

The MOUNT option results in a printed message on the on-line printer which stops the computer and tells the operator that a tape reel must be mounted.

The HOLD option signifies that the file is to be saved. A message appears on the on-line printer telling the operator to save

---

1. A file is the data that appears on tape between file marks.

the tape after the computer rewinds it upon completion of the program.

The three ENTRY cards merely define the names OPEN, READ, and EFF as entry points of the IOCS13 subroutine. An instruction that is defined as an entry point allows a call statement to transfer control to the subroutine at the specified entry point and to ignore preceeding parts of the subroutine. For example, a CALL READ statement would transfer control to the READ SXA SV1,4 instruction. Instructions from OPEN SAVE 4 to RETURN OPEN would be ignored.

OPEN SAVE 4 causes the present contents of index register 4 to be saved, thus releasing that index register for utilization by instructions which follow. Index register 4 is the only one available for many MAP operations such as transferring data between subroutines and opening, reading, or closing files; therefore, index register 4 must be saved each time a subroutine is entered if any of the operations requiring index register 4 are being performed. The SAVE pseudo-operation automatically saves index register 4 without placing a 4 in the variable field, however, including the 4 helps a programmer remember the instruction's function. Other index registers which are used in a subroutine should also be saved with the same SAVE pseudo-operation:

```
SAVE  4,1,2,6,7
```

The TSX .OPEN,4 and PZE IN1 instructions open a file; that is, they prepare a tape for being read by the computer. The above two cards must be executed before commanding data to be read from a tape.

RETURN OPEN returns control to the subroutine that called OPEN and restores index register 4 and any other index registers that were specified in the SAVE pseudo-operation. If a subroutine or entry point begins with a SAVE pseudo-operation, the return to the calling program must be accomplished with a RETURN pseudo-operation to properly restore the index registers.

READ SXA SV1,4 is the entry point for the section of IOCS13 that reads data from the magnetic tape. The SXA pseudo-operation saves the contents of index register 4 in location SV1. The SAVE pseudo-operation is not used because index register 4 is restored immediately prior to returning control to the calling program when an index register is saved by this pseudo-operation. For the READ section, index register 4 must be restored (at location SV1) before returning control to the calling program; therefore, the SXA pseudo-operation is required.

The next 3 cards read one record from tape and place the data into input buffers[1]. IN1 refers to the FILE card (first card in the subroutine) which specifies the file definition to be

---

1. Buffers are storage areas in the 7094 which serve as transfer areas between magnetic tape and the computer working area.

associated with the present READ command. EOB is the symbolic location to which control transfers if an end of buffer condition occurs. The EOB option is used for nontransmitting[1] commands (7); therefore, a non-functional address is specified. (EOB PZE ** appears later in the program.)

EOF defines the symbolic location to which control transfers if a file mark in encountered on the tape. The result of sensing a file mark will be discussed shortly.

ERR is the symbolic location to which control transfers if a parity error, check sum error, or sequence error is encountered. (Only the parity error will concern us.)

IORT DATA1,,** stores one record into locations beginning at DATA1 and ending at DATA1+n where n+1 is the number of IBM words in the record[2]. After reading a complete record, the ** is replaced by n+1. Now one record of data is stored in locations to which a programmer can refer. For more details concerning the IORT command consult pages 30 to 32 of Reference 7.

GO CLA *-1 places the contents of the preceeding instruction (IORT DATA1,,**)[3] into the accumulator. Note that the number of IBM

---

1. Only transmitting commands are employed by IOCS13.

2. One IBM word contains 36 binary digits. An IBM word is the space allotted to data, regardless of its magnitude.

3. DATA1 contains the last data word in the tape record that has just been read and ** equals n+1. DATA1 occupies bits 21 to 35 and n+1 occupies the decrement, positions 3 to 17.

words in the present record is stored in the decrement of the accumulator.

The STZ DECR clears symbolic location DECR and,STD DECR stores the decrement portion of the accumulator (which is n+1) into the decrement portion of DECR. Now only the number of IBM words in the data record being considered is stored in location DECR. A CLA DECR first clears the accumulator and then places n+1 into the decrement field of the accumulator. Since n+1 is in locations 3 to 17 of the accumulator, an ARS 18 (accumulator right shift 18 binary locations) places n+1 into the address (bits 21 to 35) of the accumulator. The address is the normal position of data which are to be stored or placed into an index register.

The STO N instruction stores n+1 into location N for later use. Since N remains in the accumulator after execution of the STO pseudo-operation, a STA WC replaces the address portion of symbolic location WC (which is **) with N. Symbolic location WC contains an AXT pseudo-operation; therefore, index register 1 now contains N.

The AXT 1000,2 instruction places 1000 into index register 2. The reason for executing this instruction will become apparent later.

SV1 AXT **,4 restores index register 4 to its configuration prior to execution of the TSX .READ,4 instruction[1]. Now the contents of index register 4 are the same as when the call to READ was executed; therefore, index register 4 is prepared to transfer data from IOCS13 to the calling program.

CLA 3,4 obtains the location of the first argument of the calling statement and places the address of argument 1 into the accumulator. The ADD N pseudo-operation adds N, the number of IBM words in the record being considered, to the first argument. For the call statement shown below, the address portion of the accumulator would contain ARG1+N.

CALL READ (ARG1, ARG2, ARG3)

The STA OUTPUT instruction stores the address portion of the accumulator (ARG1+N) into the address of symbolic location OUTPUT.

CLA DATA1+1000,2 clears the accumulator and adds the IBM words stored in location DATA1. Recall that an address modified by an index register is decreased by the contents of the specified index register before execution of the instruction. (DATA1+1000,2 = DATA1 since XR2 contains 1000.)

OUTPUT STO **,1 now reads OUTPUT STO ARG1+N,1. (Recall

---

1. Recall that the contents of XR4 were saved in location SV1 by the SXA pseudo-operation.

the operation of STA OUTPUT which appears above.) Since index register 1 contains N, DATA1 is stored in ARG1.

TXI *+1,2-1 decreases the contents of index register 2 by 1 (due to the -1 decrement) and transfers control to the next instruction (as a result of *+1). TIX *-3,1,1 decreases the contents of index register 1 by 1 and transfers control back 3 instructions (i.e., to CLA DATA1+1000,2). Now index register 2 contains 999 and index register 1 contains N-1; therefore, the computer seeks DATA1+1 and stores it into ARG1+1. The computer continues to cycle through the four cards described in the last three paragraphs until index register 1 contains one. Then the TIX *-3,1,1 instruction is not executed and control passes to CLA 4,4.

The 1OCS13 subroutine counts the number of words in a record and stores this number into location N. It is very important to have the number of words in a record available to the calling program. Transferring N to the calling program is accomplished by the next four cards. CLA 4,4 stores the address of the call statement's second argument into the accumulator. STA COUNT stores the address of ARG2 into a location named count. CLA N clears the accumulator and adds N, the number of words in the present record. COUNT STO ** stores the value of N into the address which has replaced **. (Recall that ARG2 has replaced **, therefore, the address, ARG2, contains N.)[1]

Note the difference between the four cards required to transfer N into the calling program and the seven cards required to transfer the data record to the calling program. Index registers 1 and 2 were not required when only one number, N, is being transferred. The index registers combined with TXI and TIX pseudo-operations are necessary when arrays are being transferred.

After transferring N to the calling program, control returns to the calling program by means of: HOME TRA 1,4. The name "HOME" is not required; however, it has been included for descriptive purposes. Since the READ section of IOCS13 was entered with a SXA rather than a SAVE pseudo-operation, the TRA 1,4 returns control to the calling program.

Recall that EOF is the location to which control transfers if a file mark is read by the computer. The transfer to EOF when a file mark is encountered results in terminating further tape reading. Commanding the computer to stop before reading an entire reel of tape, most of which is blank, is an important capability of this program, since valuable computer time and project funds

---

1. This may seem confusing since it appears as if ** has been replaced twice, once by ARG2 and then by N. This is not true. ARG2 (an address or pointer to a location) replaces ** (a dummy address). The STO pseudo-operation stores the contents of the accumulator (N) into the location that is called ARG2; therefore, location ARG2 contains N.

are saved as a result.

EOF XEC SV1 causes the instruction with a name field containing SV1 to be executed at the present time. Recall from previous discussions that SV1 restores index register 4. With index register 4 restored, transferring data and commands between the subroutine and calling program is possible.

TRA 5,4 is a non-standard return which transfers control to the location cited in argument 3 of the calling statement[1]. When TRA 5,4 is executed, the message "NON STANDARD RETURN" is printed by the computer on the data output page.

The transfer to ERR occurs if a parity error is encountered. The next six cards cause the message, "WARNING", to be printed on the output page and then a return to location GO for processing the data although it may be in error. Processing in spite of the error enables the programmer to salvage some results and allows him to correct for faulty data if possible.

ERR SXA SV2,4 saves the contents of index register 4 in location SV2.

CALL .FWRD.(.UN06. 111.) calls a machine language subroutine (contained on the processor tape and available to all users)

---

1. Note that reference to argument 1 is made by means of 3,4 (recall CLA 3,4 to obtain ARG1); reference to ARG2 requires 4,4, and reference to ARG3 requires 5,4. Index register 4 must be employed, and the address portion of the pseudo-operation involved is equal to the argument number +2. This is required due to internal operation of the computer and need not concern the programmer except for following the requirement of using argument number +2.

that enables a MAP programmer to specify printed data in Hollerith fields just as with Fortran. .FWRD. means Fortran Write Decimal, .UN06. calls tape unit 6, and 111. refers to a format statement number.

TSX .FCNV.,4 commands writing the data specified by a .FWRD. instruction, and TSX .FFIL.,4 must be the last card in any sequence that specifies printed output in MAP.

SV2 AXT **,4 restores index register 4 to its value preceeding the transfer due to a parity error, and TRA GO transfers control back to the normal program routine so that data continues to be placed into the calling program.

EFF SAVE 4 is the entry point for closing the file (i.e., rewinding the data tape and releasing input buffers). TSX .CLOSE,4 and PZE IN1 actually command closing input file IN1. RETURN EFF transfers control back to the calling program.

DECR BSS 1, N BSS 1, and DATA1 BSS 200 allocate 1,1, and 200 storage locations (each containing 36 bits or one IBM word), respectively, to DECR, N, and array DATA1. The BSS pseudo-operation is analogous to a Fortran DIMENSION statement. The BSS instructions must follow all executable program statements.

111. BCI 3,(10 H WARNING) is analogous to a Fortran format statement. The characters enclosed in parentheses form a Fortran Hollerith field. The 111. is an allowable MAP format number. Note that the period would not be permitted in Fortran. BCI is a

binary coded information pseudo-operation. The 3 defines the number of 6 character groups contained within and including the parentheses. Fifteen spaces are required; therefore: $15 \div 6 = 2.5$ which becomes 3 to the nearest whole number greater than 2.5.

An END pseudo-operation must be the last card in any MAP program or subroutine deck.

The preceeding discussion has described the manner in which a data tape is read by the 7094 and how data are transferred to referenced locations in both IOCS13 and the calling program. After requesting the operations available in IOCS13, the programmer has at his disposal readily accessible, stored data. The format, or position of data words relative to each other, is identical with the input tape; therefore, further processing is necessary. Two more MAP subroutines are available to alter the data format so that data can be called by a Fortran programmer.

5.2.2 BCD

The BCD subroutine processes BCD data and separates the decimal digits into either two-digit groups, four-digit groups, or six-digit groups. The last argument of CALL BCD (NI, NO, IDIN, IDOUT, NDGTS) determines the size of an output word (i.e., either a two-, four-or six-digit number). To aid the reader during the present discussion, follow Subroutine BCD from Appendix B.

When the IBM 7094 computer reads BCD data generated by the digital system, it assumes that pure binary data have been recorded since the lateral parity is odd. Since the data system uses odd parity and BCD requires even parity, the BCD data must be decoded by computer programs.

If 497 is entered onto the digital tape by means of the thumbwheel switches, the tape areas shown in Figure 21 will be magnetized. Figure 21a illustrates 497 in BCD; however, the IBM computer, because odd parity is employed, assumes that 16,967 (Figure 21b) has been recorded. A decoding process is necessary to obtain 497 rather than 16,967.

To begin the decoding process, note that the octal equivalent of 16,967 is 041107, which very closely resembles the number that was entered with thumbwheel switches. First, separate the left most digit pair (04) from 041107 and multiply the pair by 100. Next separate the middle pair[1], $(11)_8$ or $(09)_{10}$, from 041107 and multiply this pair by 10; then add: 400 + 90 + 7 to obtain 497[2].

---

1. Subscripts refer to the numbering system employed. $11)_8$ is the octal equivalent of 9 in the decimal system $9)_{10}$. The octal numbering system employs the digits 0,1,2,3,4,5,6, and 7. One digit beyond 7 is 10, and numbers continue as in the decimal system until 17, which is followed by 20, etc.

2. Every group of three bits represents the octal numbers 1,2, and 4. Corresponding numbers are; $2^0,2^1,2^2$ and 1,2,4;...$2^{15},2^{16},2^{17}$ and 1,2,4. Bits $2^0,2^1$, and 2^2 fill the units position of the octal number and bits $2^3,2^4$, and 2^5 fill the tens place, etc.

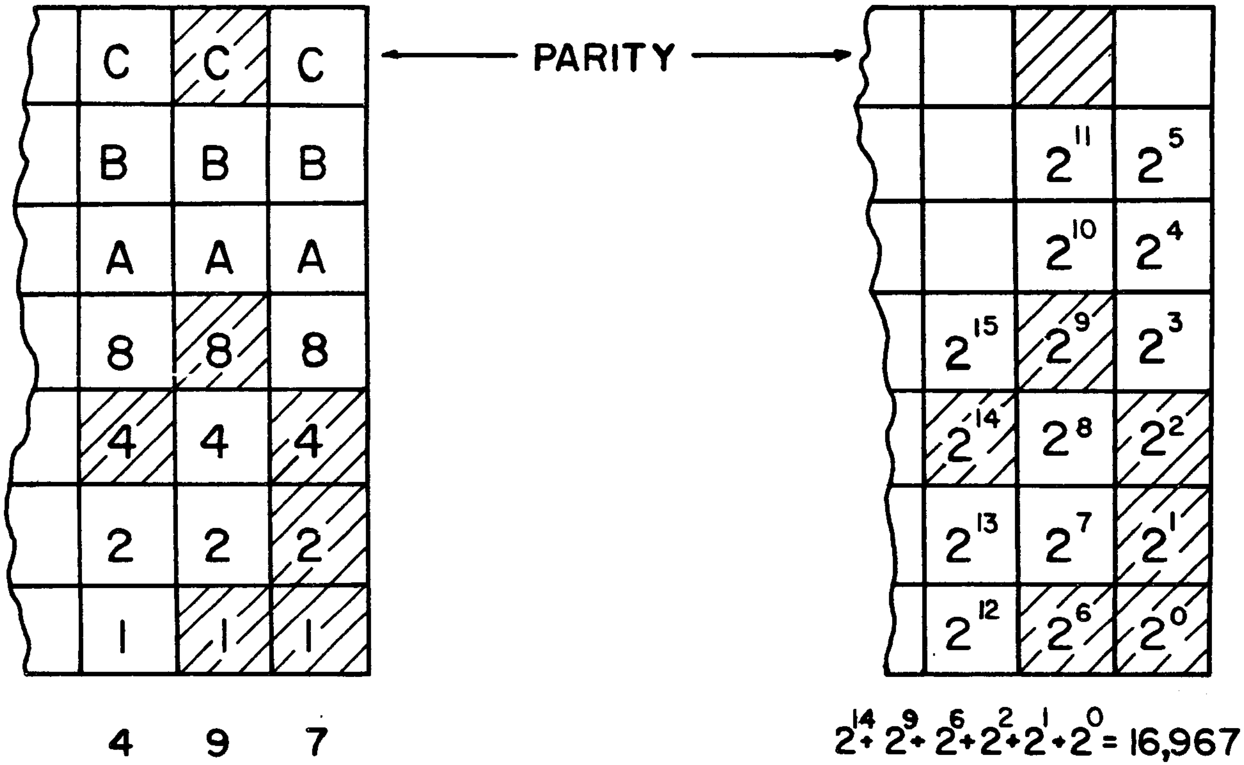


FIGURE 21a. FIGURE 21 b.

REPRESENTATION OF 497

a. WITH BCD and b. WITH BINARY INTERPRETATION OF BCD

The basic procedure described above is employed by the BCD subroutine. To aid the reader during the present discussion, follow Subroutine BCD from Appendix B at this point. Index registers 1 through 7 are saved by the first instruction. CLA* 3,4 obtains the first argument of CALL BCD and the next instruction stores the number into location LENGTH. STA *+1 and AXT **,1 set the value of XR1 equal to LENGTH, the input array length.

CLA* 4,4 places the second argument of the call statement into the accumulator. STO LOUT stores the second argument, the output array length, into LOUT. The next two instructions place LOUT into the address portions of OCOUNT (output count) and BEGIN, respectively.

AXT 200,2 sets index register 2. Note that the maximum number of input words to BCD is therefore 200. If the capacity of BCD is to be increased, every 200 that appears must be changed. Since the maximum output array is three times greater than the input array size (for the case of a two-digit output word), several instructions containing 600 appear. Each of these must also be changed if the program capacity is to be increased.

The instructions from CLA 5,4 to CLA **,1 obtain the third argument of CALL BCD, add the input data array length (LENGTH) to the address of argument three (the input array), and store this address in ** of CLA **,1.

STO AREA1+200,2 stores the first input data array word into AREA1 since XR2 contains 200.

TXI *+1,2,-1 alters XR2 and transfers control to TIX *-3,1,1, which decreases XR1 by one and transfers control back three instructions.

The last four instructions are repeated sequentially until XR1 contains one, which indicates that all input data has been transferrred to the BCD subroutine.

The following three instructions obtain the fourth argument of CALL BCD, add LOUT to the argument, and store this address, ARG4 + LOUT, into location OUT, which is the output array.

CLA* 7,4 transfers argument five, the number of digits in one output word, into the BCD subroutine, and stores the fifth argument into location DIGITS. STA START stores DIGITS into the address of START AXT **,3. Index register 3 is termed the digits counter since it controls storage of output data when the proper number of digits has been separated from an input word.

Further into the subroutine, a program section multiplies each I.D. digit by ten to a specific power. Index register 5, which controls the above-mentioned multiplication, must be set to a number (CONTROL) related to DIGITS by the following equation:

CONTRL = (DIGITS-1) *3

The seven instructions that follow generate CONTRL. CLA DIGITS clears the accumulator and places DIGITS into it. After executing SUB =1 the accumulator contains DIGITS-1. XCA places DIGITS-1 into the MQ register, and MPY =3 forms (DIGITS-1)*3, which is now located in the MQ register. STQ CONTRL stores (DIGITS-1)*3 into location CONTRL. CLA CONTRL and STA START+1 replace the address of the instruction following START with CONTROL; therefore, XR5 contains CONTRL.

Several index registers must be set before actual data processing can be accomplished. BEGIN AXT **,1 actually reads: AXT LOUT+1. AXT 600,7 sets the maximum possible output array size. (Recall that this must be changed if the program size is altered.) AXT 6,6 sets XR6 equal to 6, the number of digits in one IBM word. Later in the subroutine, XR6 will be decreased by one after each digit has been processed. After all operations on one entire IBM word are completed, control transfers to another AXT instruction which resets XR6 to six.

AXT 200,2 resets XR2 and LDQ AREA1+200,2 loads one word, AREA1, into the MQ register.

START AXT **,3 sets XR3 which counts the number of digits that are placed into one output word. The cycle of instructions that form the output number is initiated by resetting XR3.

AXT **,5 sets XR5 whose purpose will be discussed later.

STZ TEMP2 stores zero into location TEMP2. Each time a digit is processed it will be added to TEMP2 in a cumulative manner until the output number has been formed.

LOOP ZAC clears the accumulator. LGL 6 shifts each bit of the MQ register and accumulator 6 places to the left.

Recall that one IBM word, AREA1, has been loaded into the MQ register. Since the six left-most bits of AREA1 are shifted into the six right-most accumulator places, the accumulator must be cleared before shifting bits, otherwise the 30 left hand digits of the accumulator would contain irrelevant data.

STQ TEMP1 stores the 30 bits remaining in the MQ register and STO TEMP3 stores the six bits (the subject) that were shifted from the MQ register to the accumulator.

SUB =10 subtracts ten from the accumulator which still contains the six bits that were shifted from the MQ register. The purpose of SUB =10 is to test for an IBM zero. If a digit is an IBM zero (i.e., a 10 rather than a 0), then subtracting ten from the subject will leave zero in the accumulator.

If an IBM zero is encountered, the TZE ENTER instruction is executed[1]. When no IBM zero is encountered, the six bits that were placed into TEMP3 are loaded into the MQ register.

Twenty cards are then employed to multiply each of the decimal digits by its proper power of ten to form one output number.

TRA MULT+15,5 transfers control to one of the multiplication instructions, depending on the contents of XR5. For the first digit, XR5 contains 15; therefore, control transfers to MULT. MULT MPY =

---

1. The transfer to ENTER will be explained in a later section.

100000 multiplies the subject by 100,000, XCA places the product into the accumulator, and TRA ENTER transfers control to location ENTER.

At ENTER, the subject is added to TEMP2, the cumulative storage area for processed digits.

ZERO TXI *+1,5-3 is then executed and the contents of XR5 decreasedby 3. Index register 5 must be decreased in steps of three because control must transfer to MULT, MULT +3, MULT +6, MULT +9, MULT +12, or MULT +15.

As stated previously, a transfer to ZERO is made if an IBM zero is encountered. This transfer bypasses the entire MULT section, then decreases XR5 by three and the program proceeds. A zero obviously need not be added to the cumulative storage area, TEMP2, however, XR5 must be altered so that the next digit is multiplied by the next lower power of ten.

STO TEMP2 stores the data that havebeen processed and LDQ TEMP1 loads the unprocessed data into the MQ register.

TIX PASS,6,1 decreases XR6 by one and transfers control to PASS. As discussed previously, XR6 was initially set to six. When one complete IBM word(six digits) hasbeen processed, control will transfer to the next instruction (TXI *+1,2,-1) rather than to PASS.

If control transfers to PASS (i.e., all six digits have been processed), a transfer to LOOP is executed and the processing of another digit begins.

When the next sequential instruction after TIX PASS,6,1 is executed, XR2 decreases by unity and another six-digit word is loaded into the MQ register. Index register 6 is reset to six and control transfers to LOOP.

Note that XR3 has been decreased by unity again. If XR3 contains one when this instruction is encountered, control does not pass to LOOP. Each time an output word is formed, the transfer to LOOP is not taken; rather, the output word which is in the accumulator is stored into ID+300,7. Index registers 1 and 7 are then decreased by one and control passes to START, where XR3 is reset.

The transfer to START is not completed when XR1, the input words counter, contains one, which signifies that all data have been processed.

CLA ID+600,7 adds ID to the accumulator the first time this instruction is executed since XR7 contains 600. STO **,2 then stores ID into IDOUT since ** is actually IDOUT+LOUT and XR2 contains LOUT. TXI *+1,7,-1 and TIX *-3,2,1 decrease index registers 7 and 2 so that ID+1 is then stored into IDOUT+1. During the next pass through the above four instructions, ID+2 is stored into IDOUT+2. The process continues until XR2 contains one, which indicates that all data have been transferred to the calling program.

Since the BCD subroutine is quite general, (i.e., it can generate either two-, four-, or six-digit words) no changes are anticipated except for altering the maximum input array size. The necessary steps for increasing the program capacity have already been discussed.

5.2.3 UNPACK

After entering all header information, experimental data are recorded. Data records consist of a six character time word followed by packed binary data words. When reading this section, simultaneously read the UNPACK subroutine contained in Appendix B.

Subroutine UNPACK unpacks the binary data and transfers the unpacked data to the main program where another subroutine (Section 6.2.9) is called to alter the sequence of recorded data. The following statement calls UNPACK.

```
CALL  UNPACK  (AREA, TIME, MAP, N)
```

AREA is the array that stores unprocessed data, TIME refers to the time word, MAP is the array that stores processed data, and N is the number of IBM words in the record being processed.

UNPACK begins by saving the previous contents of all index registers. The number of packed words to be processed is transferred to the UNPACK subroutine by the CLA* 6,4 instruction and this number is then stored into location LENGTH.

The input data array length is also stored into XR6 by means of the STA *+1 command, followed by AXT **,6. Index register 6 must be reset later in the program; therefore, STA COUNT is necessary, where COUNT refers to COUNT AXT **,6.

AXT 1031,7 sets the index register that counts the maximum possible number of words that can be stored in one record. Actually, 1031 is beyond the limit of the present buffer; however, the excess capacity is slight and acts as a computer program storage safety factor.

CLA 3,4 obtains the address of the first argument of the call statement. The first argument passes the input data array from the main program to the subroutine. ADD LENGTH forms the address, AREA + LENGTH, which is located in the accumulator, and is the location[1] of the end of the input array. STA *+1 stores the above address into the address portion of CLA **,6. (** actually becomes AREA + LENGTH.) Since the address of CLA **,6 is modified by XR6, this instruction causes (AREA + LENGTH) minus (the contents of XR6) to be loaded into the accumulator. Initially, the contents of AREA will be placed into the accumulator.

STO AREA2 + 1031,7 stores the contents of the accumulator into

---

1. The terms address and location are synonymous. And address is a pointer to the position of a word that is stored within the computer.

AREA2 the first time this instruction is executed since XR7 contains 1031.

TXI *+1,7-1 decreases the contents of XR7 by one and transfers control to the next instruction. TIX *-3,6,1 decreases the contents of XR6 by one and transfers control back three instructions. AREA+1 is transferred from the calling program into the accumulator, and then stored into location AREA2+1.

The four instructions from CLA **,6 to TIX *-3,6,1 continue to be executed in sequential order until XR6 contains one, which indicates that all data have been transferred to the UNPACK subroutine. AXT 1031,7 COUNT AXT **,2, and AXT 2060,2 are then executed and processing the raw data begins.

The time word is processed by a method similar to the BCD subroutine, but less complex because a time word always contains six decimal digits. START AXT 6,3 sets XR3 which counts the number of digits that have been processed. AXT 15,5 sets the index register that controls multiplication of the time digits by the proper power of ten.

STZ TEMP2 clears location TEMP2 which is the cumulative storage location for digits that have been processed. LDQ AREA2+1031,7 loads the unprocessed data into the accumulator. The first word loaded will be AREA2 since XR7 contains 1031 initially.

The next 30 instructions, LOOP ZAC through LDQ TEMP1, perform the same functions as the corresponding block of instructions contained in the BCD subroutine.

TIX LOOP,3,1 decreases the contents of XR3 (the digit counter) by one and transfers control to LOOP five times. After executing the instructions from LOOP ZAC to TIX LOOP,3,1 for the sixth time, all time digits have been processed and control passes to the next sequential instruction, STO TIME.

STO TIME stores the processed time word into location TIME. Index registers 6 and 7 are then decreased by one. Recall that XR6 counts the number of IBM words that must be processed, and XR7 controls loading the proper IBM word for processing.

The computer is now perepared to process experimental data. SORT LDQ AREA2+1031,7 loads the first experimental IBM word into the MQ register. Since shifting from the MQ to the accumulator is to be performed, the accumulator must be cleared (ZAC).

The following computer instructions unpack the IBM words which contain two experimental data samples each. Complex decoding is not required here as it is for BCD data because the tape is written and read as a binary file[1]. The only required coding is to program the computer to detect an algebraic sign, and to properly recognize negative numbers.

If a data sample is negative, a mark is recorded in bit 18

---

1. Refer to the first card in IOCS13.

(bit 18 actually generates a $4)_8$, Figure 4a). No mark is recorded in bit 18 for positive numbers. Since each bit on the digital tape generates three binary numbers, the LGL 3 instruction shifts a sign indication into the accumulator. If the accumulator contains zero after execution of the LGL 3 instruction, then the first 15 bits (the first data sample) in the MQ register are positive and the TZE PLUS1 instruction transfers control to location PLUS1 for further processing.

If the accumulator contains the four bit, a negative sign is indicated and the next sequential instruction MINUS1 ZAC, is executed.

The five instructions from MINUS1 ZAC to TRA LAST are executed to process a negative number. After clearing the accumulator, LGL 15 moves the first data sample (the left half of the packed word contained in the MQ) into the accumulator. Since the digital system records the system complement of a negative number rather than its true numerical value, a brief explanation of the term system complement is necessary.

The term complement, as employed here, is concerned with modulo arithmetic, a form of mathematics that requires only a finite set of integers used repeatedly in a cyclic manner. The hour designation on a clock is an excellent example of modulo arithmetic. Time is indicated by the integers from 1 to 12. After the twelfth hour, one is repeated (4).

The complement, C, of a number, N, is the difference between the modulus (maximum number, M, in the modulo system) and the original number, N.

$$C = M-N$$

The maximum number that can be recorded by the digital system is 16,383 (14 binary bits); therefore, the system modulus is $16{,}383)_{10}$ or $37777)_{8}$. The magnitude of a negative number is obtained by subtracting the complement (value recorded for negative numbers) from the modulus. The SUB = 037777 instruction[1] forms the true value of a negative number.

STO DATA+2060,2 stores the unpacked data sample into location DATA during the first execution of the instruction since XR2 contains 2060. Subsequent executions of the STO instruction will store data into DATA+1, DATA+2, etc.

TRA LAST transfers control to LAST ZAC, which clears the accumulator in preparation for processing the last half of the packed word.

If the first half of the packed word is positive, control transfers to PLUS1 at the TZE PLUS1 instruction rather than following the next sequential instruction, MINUS1 ZAC. At PLUS1,

---

1. The =037777 literal could have been written as =16383 also; the "O" is not a zero, but the letter O to signify an octal number.

the left half of the MQ register is shifted into the accumulator (LGL 15). Since a positive data sample is already coded in proper form, the 15 binary bits located in the accumulator are stored into location DATA by means of the STO pseudo-operation.

Whether the left half of the packed IBM word is positive or negative, the LAST ZAC instruction is executed after storing the first data sample. Next, execution of a TXI *+1,2,-1 instruction results in XR2 containing 2059; therefore, the second half of the packed word is stored into location DATA+1.

LGL 3 again shifts the sign bit into the accumulator, and a transfer to PLUS2 is made if the second half of the packed word is positive. If the data is negative, control continues to the next sequential instruction, MINUS2 ZAC. The next three instructions are identical to those following the MINUS1 ZAC command.

TRA REPEAT transfers control to a group of instructions that decrease the contents of index registers 7,2, and 6 by one. Unless all input data are already processed, control transfers to SORT where another unprocessed word is loaded into the MQ register.

If the data sample is positive, PLUS2 LGL 15 is executed and STO DATA+2060,2 stores the positive data into location DATA+1.

REPEAT TXI *+1,7,-1 and TXI *+1,2,-1 are executed after the second half of the packed word is stored. Index register 6 counts the number of input words that must be processed; therefore, a transfer to SORT is made unless XR6 contains one, which indicates that all data words are unpacked and stored.

CLA 4,4 obtains the address of the time word, and STA OUT stores the time word address into the address of location OUT.

CLA LENGTH places the total input array size (data and time word) into the accumulator. The next few instructions form the output data array size, which is twice the input array size. (The input array consists of packed words, except for the time code generator output, which requires a complete IBM word.) Since the time word is treated separately, one is subtracted from the total input array size,(SUB =1). The accumulator now contains LENGTH-1, which is equal to the input data array length.

XCA transfers LENGTH-1 into the MQ register and MPY=2 forms the output data array size. STQ stores the array length into location LGTH.

To set an index register that controls the amount of output data transferred back to the main program, STA COUNT2 is executed.

CLA 5,4 obtains the third argument of the calling statement and add LGTH forms the address, MAP+LGTH, which is stored into the address of OUT1 by means of STA OUT1.

CLA TIME adds the processed time word to the accumulator and STO ** stores the data into argument two of the calling statement.

Index registers 2 and 3 must be reset to transfer data from the UNPACK subroutine to the calling program. The address of COUNT2 AXT **,3 actually contains LGTH rather than **.

CLA DATA+2060,2 adds the contents of location DATA to the accumulator and OUT1 STO **,3 stores this data into location MAP.

TXI *+1,2,-1 decreases the contents of XR2 by one and TIX *-3,3,1 decreases the contents of XR3 by one before transferring control back to the CLA instruction. The second time CLA DATA+2060,2 and OUT1 STO **,3 are executed, DATA+1 is stored into MAP+1. When XR3, the output words counter, contains one, all data have been transferred and control returns to the calling program (RETURN UNPACK).

The next nine instructions allocate storage for variables or arrays required by this subroutine, and an END pseudo-operation terminates the UNPACK subroutine.

The most probable changes anticipated for the data acquisition system which will affect the UNPACK subroutine are an increase in the memory capacity and the time code generator output resolution.

If the memory capacity is increased, each 1031 that appears in UNPACK must be changed to the maximum character block length available with the new memory. Every 2060 must be at least twice as large as the new maximum block length.

A change in the time code generator would probably be made to obtain millisecond readout resolution. For example, instead of transmitting 20 hours, 15 minutes, and 31 seconds, the improved system would generate: 20 hours, 15 minutes, 31 seconds, and 001 milliseconds. The new time word requires nine digits; therefore, two IBM words must be processed. The START AXT 6,3 should be changed to START AXT 9,3, and AXT 15,5 should be: AXT 24,5.

The MULTPY TRA MULT+15,5 instruction should read: MULTPY TRA MULT+24,5. Finally, MULT MPY = 100000 must be replaced by the following group of instructions:

```
MULT    MPY    = 100000000
        XCA
        TRA    ENTER
        MPY    = 10000000
        XCA
        TRA    ENTER
        MPY    = 1000000
        XCA
        TRA    ENTER
        MPY    = 100000
```

To add the second IBM word after processing the first one, include AXT 6,1 after START AXT 9,3. Also include the following instructions after LDQ TEMP1, and remove TIX LOOP,3,1.

```
        TIX    NEXT,1,1
        AXT    6,1
        TXI    *+1,7,-1
        LDQ    AREA2+1031,7
        LGL    15
NEXT    TIX    LOOP,3,1
```

6. FORTRAN PROGRAMMING

6.1 Introduction

Fortran, (Mathematical Formulation Translation System) is the computer language most frequently employed by engineers. Programs written in Fortran closely resemble ordinary algebra. Pure machine level programs that interpret algebraic (Fortran) statements need not concern the engineer; therefore, he can devote full attention to solving analytical problems.

The use of a logical analytical approach is extremely important if a problem is to be programmed for a computer. The sequential order of calculation within a program must be meticulously planned.

Since most engineers are familiar with the Fortran language, a detailed description of the commands and logic will not be presented here. For aid in using the version of Fortran employed by the Purdue University IBM 7094, the reader should consult References 5 and 6. The non-standard Fortran techniques employed by the data acquisition system are described as they appear in the subroutines.

While reading the remainder of this section, follow the respective computer programs in Appendix C simultaneously.

6.2 Fortran Programs

6.2.1 COPY1

Subroutine COPY1 writes processed BCD data onto a magnetic tape in Fortran so that the data can be called by a Fortran statement. The two arguments of COPY1 are the BCD data array, L, and the number of words in the array, N. If more than 80 input words are desired in one BCD record, the dimension statement must be changed.

WRITE (n) (L(J),J=1,N) writes the BCD array with tape unit n. Two tapes must be placed onto the computer; one tape contains the raw data generated at the Combustion Research Laboratory and the other tape (on tape unit n) is initially blank. To read the record that has been written above, simply rewind tape unit n and execute the following statement:

```
READ (n)  (L(J),J=1,N)
```

Two Fortran statements are included to write the data on paper for immediate observation.The printout at this point is quite useful for debugging purposes also.

6.2.2 COPY2

COPY2 is identical to COPY1 except for the substitution of READ (n) for WRITE (n). To read a record of header data that has been written by COPY1, call COPY2, using the same arguments for calling both COPY1 and COPY2. The two cards that cause data to be printed on paper are required in COPY2 since the Fortran tape contains processed BCD data written in Fortran records for printing information that identifies subsequent data on the same tape.

6.2.3 PRELIM

Subroutine PRELIM writes alphabetic information that corresponds to the coded digits which are contained in record one. The PRELIM subroutine is self-explanatory when investigated along with the check list for header record number one, located in Appendix E. Table D1 is a sample computer output page for PRELIM data.

PRELIM contains expansion capability since each present category can be expanded to a total of 100 possibilities. For example, the ID(6) category, which describes the type of cooling employed, presently contains four cooling designations. The two-digit ID number can equal anything from zero to 99; therefore, 100 format statements and consequently 100 cooling designations are possible. More categories can be added in a manner that is analogous to the existing five groups; however, if more than 200 groups are required, the dimension statement capacity must be increased.

6.2.4 THERM

Subroutine THERM is presently designed to write four-digit I. D. data to describe thermocouples and their temperature ranges. Argument one is the input data array and the second argument equals the number of output words in the BCD record. THERM first writes the number of four-digit words contained in the I.D. record being processed. Several pairs of WRITE statements cause the following printed output:

THERMOCOUPLE n

TEMPERATURE RANGE a to b

Each 12-digit header entry contains three four-digit words presently programmed to describe thermocouples. ID(1), ID(2), and ID(3) describe the first thermocouple, while ID(4), ID(5), and ID(6) describe the second thermocouple, etc.

In the future, format statements in THERM can be added or removed to accommodate the number of thermocouples employed by an experiment. Additional subroutines can be written to fulfill the needs for writing four decimal digit I. D. data. A sample computer output page appears in Appendix D (Table D2).

6.2.5 PTRANS

Subroutine PTRANS performs a task similar to that performed by THERM. Argument one is the input array, and argument two equals the number of words in the array. PTRANS first prints the number of output words contained in the BCD record.

Preliminary studies indicate that six decimal digits are required to describe a pressure transducer; therefore, PTRANS presently accepts six-digit header words and writes the following, where 661121 is the manufacturer's serial number:

TRANSDUCER 661121

MAX. PRESSURE 10000 PSI.

PTRANS can be very easily expanded to accept more input words by adding additional format statements. The subroutine can be altered to describe other types of sensing devices by merely changing the format statements (numbers 10 and 11), or another similar subroutine can be written. Table D3 illustrates the output obtained from subroutine PTRANS.

6.2.6 PIF3

PIF3 is a function subprogram (5, 6) that accepts a two-dimensional input array and interpolates between data points by means of a third order curve fit procedure. Two other similar subroutines, PIF1 and PIF2,are available for interpolation by assuming a linear and a second-order curve, respectively. For thermocouple data, investigations indicated that either PIF1, PIF2, or PIF3 yielded equally satisfactory results if no more than five interpolations were requested between input data points.

The input data table is read by the main program and transferred to the subroutine that calls PIF3. The first data card must define the number of data points in the reference table. Data must be entered in ascending independent variable order.

To demonstrate the use of PIF3, consider a temperature-voltage array. After reading temperatures TCA(I) and voltages VCA(I) into computer memory, any voltage value can be specified and a corresponding temperature obtained by execution of the following statement:

TEMP(J) = PIF3 (A, VCA, N, TCA)

VCA and TCA are, respectively, the independent and dependent variables of the input array. N is the number of data points in the reference table, A is any desired independent variable value, and TEMP(J) is the corresponding dependent variable.

6.2.7 TABLE

TABLE prepares nonlinear experimental data in voltage units for being converted to engineering units by PIF3. Subroutine TABLE is called by:

CALL TABLE (V,NP, EU, NROW, FIRST,LAST,OUT, 20,30,TEMP,20,30)

V refers to the voltage (independent variable) in the calibration data array, NP equals the number of data points in the array and EU refers to the engineering units (dependent variable) in the array. NROW equals the number of data samples per channel, FIRST is the first channel to be converted, and LAST is the last channel to be converted from voltage to engineering units. OUT refers to the array of experimental data in voltage units which is converted to engineering units. Corresponding arguments in the TABLE subroutine are: (VOLT, NPTS, T, NROW, FIRST, LAST, OUT, L, M, TEMP,LL,MM).

OUT, 20,30 corresponds to OUT, L,M; therefore, to alter the dimensions of the OUT array, its dimension need only be changed in the main program and the calling card. The TEMP array is treated in a similar manner.

FIRST is ordinarily a floating point variable; therefore, it must be defined as an integer. Completing the DO 1 J=FIRST, LAST loop obtains the engineering units for channel numbers from FIRST through LAST. The DO 1 K=1,NROW loop obtains the engineering units for each data point of the channels from FIRST TO LAST. A=OUT(J,K) redefines the data points as unsubscripted variables for use as an argument of PIF3. The correct engineering units for all data in the OUT array is now contained in the TEMP array, which is written on magnetic tape in Fortran. The DO 11 loop writes one record beginning with all data from channel FIRST and ending with all data from channel LAST.

6.2.8 LINEAR

Subroutine LINEAR accepts an array of data pairs and calculates a linear equation to represent the input data. The method of least squares, described in Section 3, is programmed in LINEAR.

The first READ statement (number 44) reads two integers, the number of data pairs (N) and a channel identification number(NBR).

Statement number 2 reads N data cards containing the independent variable in columns one to ten and the dependent variable in columns 11 through 20. Immediately after reading all data, the next five cards call for printout.

The statement, 444 READ (5,59) TEST, reads a test value that is the criterion for eliminating erroneous data points. Later in the program when dependent variables are calculated and compared with experimental dependent variables, data points are cast aside if the deviation between y-calculated and y-experimental exceeds the current value of TEST. The two circled points in Figure 17 would be cast aside when TEST is one percent of full scale.

Any number of TEST values can be read for each separate linear equation calculation; however, the TEST values must decrease in magnitude. Investigation of data from experimental calibrations conducted at the Combustion Research Laboratory at Purdue University has demonstrated that the most accurate linear equation is obtained if a large number of TEST values are employed in decreasing order. The last TEST value must be zero, which signifies that least square calculations are completed.

The READ statement (number 4400) reads a single point calibration data pair. The independent variable and the linear equation are used to calculate the corresponding dependent variable. Calculated and calibration dependent variables are then subtracted to determine the deviation that is present.

LINEAR can be used as a separate program with data punched on cards and read in the order described above. If a large amount of calibration data are collected, the data can be recorded by the digital system, and LINEAR can be employed as a subroutine of the main tape processing program to be described soon.

Employing LINEAR as a subroutine in conjunction with the main tape reading program requires that the following statements be removed:

```
      DO 2  I = 1,N
2     READ (5,3) X (I), Y(I)
```

The cards indicated below must be included when the above cards are removed:

```
      SUBROUTINE LINEAR (X,N)
      Y(1) = 0.
      DELTY = a
      DO 2  I = 2,N
      J = I-1
2     Y(I) = Y(J)+DELTY
```

If an end to end calibration must be performed immediately prior to conducting an experiment and the time required to obtain a computer printout of calibration data is not available, LINEAR can be employed as a subroutine of the tape processing program. So that calibration data can be obtained prior to processing raw experimental data, LINEAR must be called before the UNPACK subroutine. The slope and intercept of transducer curves are then calculated for each transducer that is calibrated. Two arguments, the slope and y-intercept, must be added to the SUBROUTINE LINEAR card and call statement. These are illustrated below.

```
CALL LINEAR (EECAL, NE, SLOPE1, INT1)
SUBROUTINE LINEAR (X, N, AM, B)
```

Several other calibration programs have been written for second-order and other[1] non-linear equations.

6.2.9 PSIG

Subroutine PSIG converts experimental data in voltage units to data in engineering units if calibration data indicates that the sensing device is linear. The call statement is:

```
CALL PSIG (NROW, FIRST, LAST, AN, BN, SN, DN, TIME, OUT,
20,30, PSI,20,30)
```

NROW, FIRST, LAST, TIME, and OUT, 20, 30 are the same as the variable names defined in TABLE. AN is the slope and BN is the y-intercept of the linear equation that relates engineering units to voltage. SN is the standard deviation obtained from least squares consideration and DN is the maximum deviation to

---

1. $y = Ax^2 + Bx + C$
 $y = A\sqrt{x} + B$
 $y = Ax + B\sqrt{x} + C$

be expected. PSI, 20, 30 is the output data array from PSIG. Corresponding arguments in subroutine PSIG are:

(NROW, FIRST, LAST, SLOPE, YINT, SIGMA, MAXDEV, TIME, OUT, L,M, PSI,LL, MM)

MAXDEV is redefined as a floating point number, while FIRST and TIME are redefined as fixed point variables. The DO 1 J = FIRST, LAST loop obtains the engineering units for each channel, and DO 1 K=1, NROW obtains the engineering units for each data point of the channels from FIRST TO LAST. The data are then written in engineering units on magnetic tape just as was done in TABLE. If desired, the standard deviation and maximum deviation could also be written on tape by inserting the following statement within the DO loop:

```
WRITE (2)  SIGMA, MAXDEV
```

6.2.10 WREU

Subroutine WREU writes the experimental data in engineering units. The main program reads the magnetic tape that has been written by the Fortran WRITE statements contained in TABLE and PSIG. Data are then transferred from the main program to WREU by means of the following call statement:

```
CALL WREU (EUNITS, 20, 30, NCIU, N,TIME)
```

EUNITS is an array that contains the data from one record in engineering units. NCIU, the number of channels in use, is obtained from record one, a header record that has been dialed in with thumbwheel switches. N equals the number of packed words in one data record and is obtained by the IOCS13 subroutine as the input tape is read. TIME transfers the time code generator output to subroutine WREU.

Corresponding arguments in the WREU subroutine are:

OUT, L, M, NCIU, N, TIME

The second and third arguments are included to enable altering the OUT array dimensions without changing cards in the subroutine.

Since the variable TIME is ordinarily a floating point number (due to the first letter T), it must be redefined as an integer.

Format statements 1966 and 1967 cause the experimental data printout to begin on a new page, five lines from the top.

NN equals the number of experimental data points contained in the present record. Recall that N equals the number of packed words in the record, one of which is the time word. N-1 is the number of packed data words; therefore (N-1)*2 equals the number of words (unpacked) per record and NROW is the number of data samples per channel per record.

The next card defines a variable for reasons which will soon become apparent. Since the data from 20 channels cannot be printed across one page, two arrays are written with ten channels per array. If the number of channels in use exceeds ten, the logical-IF statement is executed.

Consider the case in which data have been recorded on more than ten channels, NHALF, therefore, equals ten; rather than NCIU as defined by the statement preceeding IF (NCIU .GT. 10) NHALF=10. The time word is then written and the next logical-IF statement is not executed since NHALF equals ten. Format number 34, the header label for channels one through ten, is printed. The next two statements (DO 35 and statement number 35) write all data in the present record for channels one through ten. The next logical-IF statement is not executed; however, 5 more spaces are skipped (WRITE(6,1967)) and the header label for channels 11 through 20 is printed. The DO 38 loop writes the remaining data from the present record and returns control to the main program.

Now consider the case in which from six to ten channels have been used. NHALF, therefore, equals NCIU[1] and all statements described above are executed up to:

```
IF (NCIU .LE. 10) GO TO 53
```

---

1. See the first logical-IF and the statement preceeding it.

Since NCIU equals ten or less, control transfers to statement number 53 which causes a return to the main program.

If five channels or less are employed, NHALF equals NCIU since the following statement is not executed:

```
IF(NCIU .GT. 10) NHALF = 10
```

The next two statements are executed; then control transfers to statement number 43 since NHALF is less than or equal to five. At statement 43 a header label is written for channels one through five,and then all data from the present record is written under the proper channel column. Control then transfers to the main program.

6.2.11 MAIN

The MAIN program calls all the subroutines that have been described here. MAIN serves as the major controlling program for the data acquisition system. Most of the subroutines need never be altered except for PTRANS, PRELIM, and THERM, as described in their respective sections. The main program is at the experimenter's disposal to alter in order to best suit his needs.

In the present form, the first six cards of MAIN assign storage for all subscripted variables in the main program and any arguments that are passed from a subroutine to MAIN. The

INTEGER and REAL "type statements"[1] are employed rather than a DIMENSION "type statement" so the programmer can use more meaningful array names rather than restricting himself to beginning integer (fixed point) variable names with I,J,K,L,M, or N and real (floating point) variable names with the remaining letters of the alphabet.

TIME and FIRST are defined as unsubscripted integers since only one storage location is required.

REWIND 2 prepares a blank tape for recording data in engineering units. (Recall that subroutines PSIG and TABLE write data in engineering units.)

CALL OPEN transfers control to the OPEN entry point of IOCS13 which prepares an unprocessed tape for being read by the computer.

CALL READ (PRE, NPRE, $60) transfers control to the READ entry point of subroutine IOCS13 which then reads the first record of data from the input tape. Subroutine IOCS13 transfers one record of data from the input tape to an array PRE and the number of IBM words in the record is stored into location NPRE.

---

1. "Type statements" DIMENSION, REAL, and INTEGER define a single variable contrary to its type, designated by the first letter of the variable name, and/or define storage arrays.

If a file mark is encountered after reading a record, IOCS13 first transfers the PRE array to the main program. IOCS13 instruction EOF is then executed, which causes a return (termed a non-standard return) to the main program location cited in argument three of the call statement. The statement number located in argument three must be preceeded by a $. CALL READ (PRE, NPRE, $60) returns control to statement number 60 if an end of file mark is encountered on the input tape.

The non-standard return is a powerful programming aid. The application of a non-standard return is to signal the computer to stop reading a tape when the programmer desires. If a tape is not entirely filled with data, a file mark should be written after recording data on the digital tape recorder. The non-standard return capability will prevent the computer from searching blank sections of tape for data if a file mark is recorded at the end of a complete experimental record. If tape marks (or file marks) are written between experimental test files[1], data can be processed until a file mark is encountered. When the file mark is sensed,

---

1. Record marks or tape gaps are generated several times per second by the Computer Format Control Buffer; these are 0.75 inches long. File marks (3 1/2 inches long) must be generated manually by depressing the "MANUAL RESET" and "WRITE FILE MARK" buttons, respectively, on the CFCB. A file is the data recorded between the file marks which should be generated after conducting each experiment.

the non-standard return transfers control to another part of the main program which is suited for processing the second file of data.

The $60 non-standard return transfers control to statement number 60 which causes the data tape to be rewound.

WRITE (6,1) causes the next printed output to begin on a new page.

The first record contains data that the programmer wishes to separate into two-digit groups; therefore, the number of IBM words that IOCS13 has read (NPRE words) is multipled by three to obtain the number of two-digit words.

CALL BCD (NPRE, NPRE3, PRE, IDPRE,2) separates NPRE raw BCD data words (the PRE array) into NPRE3 two-digit words which are then placed into the IDPRE array. The last argument defines the number of digits comprising each output word (from BCD, in the IDPRE array).

CALL COPY1 (IDPRE, NPRE3) causes data to be written onto the Fortran tape as described in Section 6.2.1.

CALL PRELIM (IDPRE, NPRE) writes the alphabetic data that is coded in PRE. Refer to Section 6.2.3 for an explanation of PRELIM.

The number of channels in use (NCIU) equals the fourth digit pair entered by the Computer Format Control Buffer thumbwheel switches.

The seven statements that follow are comment cards which are ignored by the computer, but which aid the programmer.

CALL READ (LABEL6, N6, $60) reads the second record from the input tape and stores the N6 IBM words into the LABEL6 array.

CALL BCD (N6, N6, LABEL6, ID6, 6) processes the N6 IBM words contained in the LABEL6 array and stores the N6 output words into the ID6 array. Six equals the number of digits per output word.

CALL COPY1 (ID6, N6) writes record two onto the Fortran tape, and CALL PTRANS (ID6, N6) writes data to describe sensing devices.

The five cards from CALL READ (LABEL4, ...) to CALL THERM (ID4, N4X) perform functions on record three that are analogous to the operations described in the preceeding three paragraphs. In the program being considered, record three describes thermocouples employed by the experiment. The output array length from BCD is now one and one-half times larger than the input array since a 12-digit number is separated into three four-digit numbers. Each IBM word (6 digits) becomes one and one-half output words; therefore, the N4X = N4 * 3/2 equation is necessary.

CALL READ (IDSC, NS, $60) reads a record of I.D. data containing the slope, intercept, standard deviation, and maximum deviation obtained from pressure transducer calibration data.

DO 2 I=1, NS and 2 SCF(I) = SC(I) change the fixed point SC array to a floating point array called SCF.

Statements A(1) = SCF(1)/100000. through D(5) = SCF(20)/1000. place the decimal point of all data contained in the SCF array into the correct position[1].

The format statement 500 and 510 are self-explanatory for anyone familiar with Fortran.

READ (5,500) NICP reads an integer that defines the number of iron-constantan thermocouple data pairs to be read by the next statement.

DO 555 I=1, NICP and 555 VIC(I) = VIC(I) *200 scale the voltages from the Pace thermocouple voltage-temperature data table to conform with the digital system gain of 200 for thermocouple data.

NDR equals the number of experimental data records that have been read by IOCS13. The next two format statements and the WRITE statement are self-explanatory.[2]

1000 NDR = NDR + 1 increases the number of data records prior to reading the current record by means of: CALL READ (AREA, N,$60). N equals the number of experimental data IBM words sensed by IOCS13 and AREA is the array assigned to data from the present record.

---

1. I.D. data consists of integers since decimal points cannot be entered by thumbwheel switches on the Computer Format Control Buffer.

2. FORMAT (1H1) ejects a page so that the next printed data begins on a new page.

CALL UNPACK (AREA, TIME, MAP, N) unpacks the raw experimental data AREA, returns the time word TIME, and returns the processed experimental data to an array called MAP. Refer to Section 5.2.3 for an explanation of the UNPACK subroutine.

NROW = N*2/NCIU calculates the number of rows contained in the printed data array for the records[1] containing NCIU channels (or columns of data).

NN equals the number of unpacked data words in the present record. N-1 equals the total number of packed words minus the time word.[2] Since two data samples are packed into one IBM word, (N-1) is multiplied by two.

The output from UNPACK consists of the array, MAP (1) through MAP(NN). Consider the case in which 20 channels have been employed for recording data. MAP(1) equals the first sample of channel one data, and MAP(21) equals the second sample of data from channel one.

The nine cards, M1=0 through 112 CONTINUE separate data into a double subscripted array according to channel number (first subscript) and row number (second subscript). Consider the present

---

1. All experimental records contain the same number of words for any one recording.

2. The time code generator output requires one complete IBM word.

record to consist of a table with 20 columns and NROW rows, where an arbitrary element is denoted by OUT(i,i). MAP(1) equals OUT(1,1), MAP(20) equals OUT(20,1), MAP(21) equals OUT (1,2), and MAP(40) equals OUT (20,2).

After executing the first four cards in this nine card group, M1 equals one, I equals one, and K equals one. The DO 66 loop is executed with M1 and I equal to one, while K advances (in increments of 1) from 1 to NN/(NCIU-1), and II advances from 1 to NN(in increments of NCIU). This DO loop collects all the data from channel one, OUT (1,1), OUT(1,2), OUT(1,3), ... Note that OUT(1,1) = MAP(1), OUT (1,2) = MAP(21), and OUT(1,3) = MAP(41). OUT (I,K) = OUT (I,K)/1000. places the decimal point into the proper position since it is located at the extreme right for integers[1].

After completing the DO 66 loop once, I equals two, M1 equals two, and K again equals one. The DO 66 loop is again entered and the following definitions are made: OUT(2,1) = MAP(2), OUT (2,2) = MAP (22), OUT (2,3) = MAP (42), OUT (2,11) = MAP (202), etc. The above procedure is repeated until all data are transferred into the OUT array.

WRITE (2) TIME writes the time word associated with the present record onto the Fortran tape. The group of instructions from

---

1. All data consists of four decimal digits since the data acquisition equipment is a four decimal digit system. Actual voltages range from .001 to 9.999; therefore, the division by 1000 is necessary.

FIRST = 1 to statement number 60 call subroutines that transform the voltage units of the OUT array into engineering units for channels from FIRST to LAST.

GO TO 1000 transfers to a statement that increases the number of data records by one and the program then continues to read the next record. The statements from 1000 to GO TO 1000 are executed until a non-standard return transfers control to statement number 60 (when a file mark is encountered).

Format statement 61 and its associated WRITE statement print a message when the non-standard return has been executed.

The remainder of the main program prints the data that has been written on the Fortran tape. REWIND 2 rewinds the Fortran tape (output tape) in preparation for tape reading.

The four CALL COPY2 statements read the four ID records and print the data on paper. NR equals the number of data records that have been read from the output tape. Data records are read until NR equals NDR, the last data record in the present file.

The three WRITE statements following instruction number 65 are self-explanatory. The TIME word is then read, and the first record of data, now in engineering units, is read from the output tape and stored into the EUNITS array by means of the following DO-loop:

```
      DO 11  J = 1, NCIU
11    READ(2)  (EUNITS  (J,K),  K = 1, NROW)
```

CALL WREU (EUNITS,NCIU , N, TIME) writes one record of data, the EUNITS array, as described in Section 6.2.10. Statements from number 65 to IF (NR .LT. NDR) GO TO 65 are executed until NR equals NDR, which indicates that all data have been read and printed. A file mark is then written onto the output tape, and it is rewound for removal from the IBM tape reader.

The main program described above was employed for several experimental tests at the Combustion Research Laboratory. Several minor alterations were made on the main program to meet the requirements of particular tests; however, no changes in the subroutines were necessary. Altering the main program appears to be the only revision that is necessary when conducting a series of similar experiments.

6.2.12 Program Control

Of the first five cards in the main program, only the $$$TAPE card is unfamiliar to most Fortran programmers. The other four cards are standard control statements. The contents of the $$TAPE card are printed for the 7094 operator before execution of the program. One option on this card defines the identification label that appears on the tape reel (JPC EECO TAPE NUMBER 1) for the 7094 operator since he must mount the tape onto the computer. The RING OUT option tells the operator to remove the plastic ring, thereby preventing data from being written onto the tape and destroying the present contents. HIGH specifies the packing density employed and SAVE indicates that the reel is to be returned to the programmer. A computer program block diagram illustrating the calling order for subroutines appears in Figure 22.

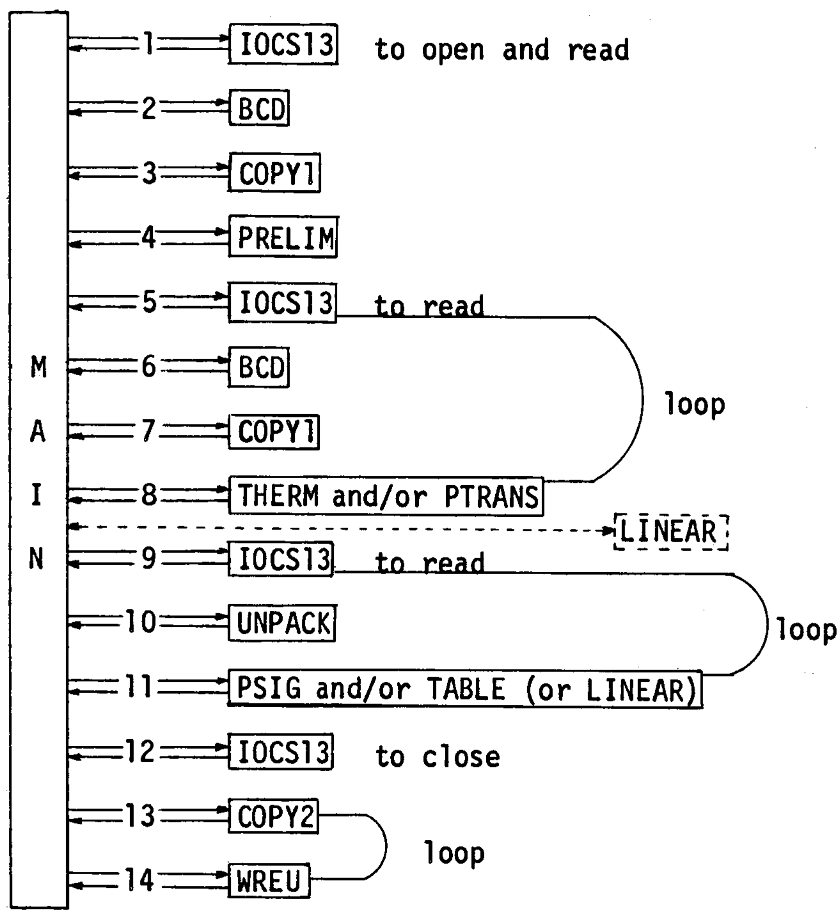


Figure 22. Computer Program Block Diagram

7. DISCUSSION OF EXPERIMENTS

Five test firings of a rocket engine were conducted at the Combustion Research Laboratory. All instrumentation devices were used to record data from the experiments and to test the operation of recording devices. The computer output sheets in Appendix D are samples of the identification and experimental data that are recorded.

The first identification data record is written by the PRELIM subroutine. I.D. records 2 and 3 are written by THERM and PTRANS, respectively.

Experimental data consisted of temperatures in degrees Fahrenheit (channels 1 through 7) sensed by iron-constantan thermocouples which were placed at 7 positions about the circumference of the rocket engine nozzle throat and at varying depths. The differences in the recorded temperatures result from the junction location.

Channels 8, 12, and 13 are vacant on the digital subsystem; therefore, zero is indicated. Channels 9 through 11 recorded chamber pressure data in psig.

Channels 14 through 16 are records of ambient temperatures at various positions in the test cell (^{o}F) which have been measured by Rosemount temperature sensors.

Channels 17 through 20 are outputs from Potter flowmeter equipment. Since the flowmeters were not required for the five experiments conducted, a constant frequency signal of 250 cps was employed as input.

Data from all recording devices correlated satisfactorily. The Digital subsystem printout contains data from one record (about 0.065 seconds duration). The sample record selected occurred when the engine reached a steady burning rate.

8. CONCLUSIONS AND RECOMMENDATIONS

8.1 Conclusions

The data acquisition system has been utilized during several experimental rocket engine test firings. The data obtained satisfactorily described the experiments being conducted and yielded complete and accurate records. In addition to the digital system output, which is presented in Appendix D, oscillograph, strip chart and analog tape records are also available. These test firings demonstrated the great potential which is characteristic of this data acquisition system and indicated that more sophisticated experiments can be undertaken at the Jet Propulsion Center, partially as a result of the instrumentation facilities and programming flexibility described herein.

The curve fitting computer programs have been tested with experimental calibration data to determine the most effective manner of utilizing these programs. The major requirement of calibration results was to obtain better than 0.5% overall accuracy. Several different calibration procedures were tested; however, those presented in this paper were the most effective.

8.2 Recommendations

A continuing program to assess the long-term accuracy of the data system is recommended. In addition to the previous discussions concerning calibration, a suggested approach to accomplish the accuracy assessment program is to perform an end to end calibration and use this data, along with single point calibration data, to evaluate experimental results until another end to end calibration is performed.

After performing the second end to end calibration, consider all calibration data in the latest calibration curve determination, with the most recent end to end data unadjusted and the least recent adjusted for bias by means of single point calibration data. When enough end to end calibration datahave been collected (approximately 20 data samples per calibration point), a true statistical evaluation of the calibration data could be performed. After determining the type of statistical model that describes the data (Poisson, Binominal, Gaussian, or other), the Gaussian standard deviation can be replaced by another approximation based on the latest analysis.

Computer programs were tested with both single and double precision[1]. Results indicated that single precisionwas generally

---

1. Double precision retains 17 significant figures during calculations rather than the usual nine figures. If calculations such as the difference between 100,000,025 and 100,000,020 occur during a program, then double precision should be employed.

satisfactory; however, if data contains extremely large and small numbers[1], then double precision should be employed.

At present, the data from an experiment is written, in engineering units, onto paper and onto magnetic tape. The existing computer programs for the data acquisition system are in the form of punched computer cards. In the future, as additional computer programs are written for performing calculations with the recorded data, the input card deck may become quite large. When the input program becomes large enough to justify transferring it onto magnetic tape, this should be done unless frequent program changes are anticipated.

As the personnel at the Jet Propulsion Center become better acquainted with the data processing approaches described in this paper, program simplifications may be possible. The author is presently investigating the possibility of using Fortran to accomplish some of the programming tasks that are now written in MAP. With the experience gained by the detailed understanding of computer operations that are required by MAP programming, writing

---

1. Extremely large or small refers to cases in which the difference between two numbers is greater than 9 significant figures. If all data from one variable are the same order of magnitude (i.e., $X = n*10^2$, $Y = n*10^{-7}$; $n = 0,1,2,3,\ldots,9$), a better approach is to scale the data by multiplying all values of Y by 10^7 rather than employing double precision since more computer time and programming effort is required by double precision.

some MAP subroutines in Fortran may be possible, which would greatly aid the future utilization of this system.

Computer program alterations are necessary if on line processing capability is added to the data acquisition system. On line processing offers many advantages which justifies purchase of the required equipment when the proper University computer facilities become available. To accomplish the on line processing, a computer input station must be installed at the Jet Propulsion Center which includes an input typewriter, tape reader, and transmission line between the computer center and the Combustion Research Facility.

BIBLIOGRAPHY

BIBLIOGRAPHY

1. IBM General Information Manual, Introduction to IBM Data Processing Systems.

2. IBM Systems Reference Library, IBM 7090/7094 Programming Systems: Macro Assembly Program (MAP) Language.

3. Sherman, Philip M., Programming and Coding Digital Computers, John Wiley and Sons, Inc., New York, 1963.

4. Sherman, Philip M., Programming and Coding the IBM 709-7090-7094 Computer, John Wiley and Sons, Inc., New York, 1963.

5. IBM Systems Reference Library, IBM 7090/7094 Programming Systems, Fortran IV Language, File No. 7090-25, Form C28-6274-3.

6. IBM Systems Reference Library, IBM 7090/7094 Programming Systems, Fortran IV Language, Version 13, File No. 7090-25, Form C28-6390-2.

7. IBM Systems Reference Library, IBM 7090/7094 IBSYS Operating System, Version 13, Input/Output Control System, File Number 7090-30, Form C28-6345-4.

8. Baird, D. C., Experimentation, An Introduction to Measurement Theory and Experiment Design, Prentice-Hall, Englewood Cliffs, N. J., 1962.

9. "Realistic Evaluation of Precision and Accuracy of Instrument Calibration Systems", National Bureau of Standards Journal of Research, Part C - Engineering and Instrumentation, V67c, Number 2, April-June, 1963.

10. Laning, J. H., Ph.D. and Battin, R. H., Ph.D., Random Processes in Automatic Control, Mc-Graw Hill, New York, 1956.

11. Freund, John E., Mathematical Statistics, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1962.

12. Cook, N. H., ScD., and Rabinowicz, E., Ph.D., Physical Measurement and Analysis, Addison-Wesley, 1963.

13. Wolf, Frank L., Elements of Probability and Statistics McGraw-Hill, New York, 1962.

14. Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, U. S. Dept. of Commerce, National Bureau of Standards, Applied Math. Series, 1962.

15. C. R. C. Standard Mathematical Tables, Charles D. Hodgman, Editor, Samuel M. Shelby, Ph.D., Robert C. Weast, Ph.D., Associate Editors, Chemical Rubber Publishing Co., Cleveland, Ohio, 1959.

16. Whittaker, E., and Robinson, G., The Calculus of Observations, Blackie, London, 1946.

17. Honeywell Operating Manual for the Combustion Research Laboratory Data Acquisition System.

18. IBM Systems Reference Library, IBM 7094 Principles of Operation, File No. 7094-01, Form A22-6703-1.

APPENDIX A

GLOSSARY OF PSEUDO-OPERATIONS

APPENDIX A

GLOSSARY OF PSEUDO-OPERATIONS

OPERATION	VARIABLE FIELD	
ENTRY	( )	This defines an entry point of a MAP subroutine. This is used when a subroutine can be entered at several locations, depending on the calling sequence used in the calling program.
RETURN	( )	Commands the computer to return control from the subroutine (or entry point of the subroutine) defined by the name contained in the variable field of this card.
END		This defines the end of a map program or subroutine. It must be the last card in the program deck, unless data is ready by the program, in which case the $DATA card and data follow the END card.
CALL	( )	CALL requests the computer to transfer control to a subroutine defined by the name written in the variable field.
SXA	Y,T	Store Index in Address, stores the contents of index register T into the address portion of location Y. Positions S, 1-20 of Y are unchanged.

OPERATION	VARIABLE FIELD	
AXT	Y,T	Address to Index True, stores the contents of Y in index register T. If desired, the Y may be a variable, a number or **. The ** is replaced by the address portion of the following command; which must preceed this AXT statement: (STA NAME). NAME is written in the name field of the AXT instruction.
SAVE	A,B,C,...	The SAVE instruction saves the contents of index registers A,B,C,... upon entering a subroutine. SAVE is generally the first instruction of a MAP program; therefore, use of index registers A,B,C,... by the program will not destroy index register contents in the previous subroutine.
CLA	Y,T	Clear and Add. Clear the entire accumulator, then add the contents of Y (bits S, 1-35) to the accumulator. Modification by an index register is optional[1].
STZ	Y,T	Store Zero. This instruction stores zero bits into location Y; i.e., it effectively clears locations 1-35 of Y and sets the sign to +.
STD	Y,T	Store Decrement stores the decrement portion (binary positions 3-17) of the accumulator into the decrement of location Y.

1. If T follows Y in the variable field, this indicates that modification by an index register is possible.

OPERATION	VARIABLE FIELD	
STO	Y,T	Store. This instruction stores the contents of the accumulator (positions S, 1-35) into respective positions of location Y.
STQ	Y,T	Store MQ. This instruction places the contents of the MQ register into location Y.
STA	Y,T	Store Address stores the address portion (positions 21-35) of the accumulator into respective locations of Y. Other positions of Y and all positions of the accumulator are unchanged.
SLW	Y,T	Store Logical Word stores the contents of the accumulator (positions p, 1-35) into location Y. The contents of the AC and position S of Y are unchanged.
ARS	N	Accumulator Right Shift shifts the contents of the accumulator to the right by N binary locations. Bits shifted past 35 are lost and bits vacated are filled with zeros. The sign position is unchanged and bits shifted from Q enter P, and bits from P enter position 1.
ALS	N	Accumulator Left Shift: Analogous to ARS.
LGL	N	Logical Left Shift treats the contents of the accumulator (positions Q, P, 1-35) and the contents of the MQ register (positions S, 1-35) as one register. The contents of the AC and MQ are shifted left by N binary locations, and the sign of the AC is unchanged. Bits move from the MQ to AC.
LGR	N	Logical Right Shift. Similar to LGL except the contents are shifted to the right. Bits move from AC to MQ.

OPERATION	VARIABLE FIELD	
RQL	N	Rotate MQ Left. The contents of the MQ are shifted left by N binary positions. Position S enters position 35; therefore, the register becomes a circular one from which no bits are lost.
ZAC		Zero the Accumulator. This command completely clears the accumulator. The variable field must be vacant.
XCA		Exchange Accumulator and MQ. The contents of the accumulator and the MQ register are exchanged bit for bit (bits S,1-35).
TRA	Y	Transfer. Causes an unconditional transfer to the location cited in the variable field.
TZE	Y	Transfer on Zero. If the Accumulator contains zero, the computer takes its next instruction from Y. Otherwise, the next sequential instruction is executed.
TNZ	Y	Transfer on No Zero. If the contents of the accumulator (Q, P, 1-35) are not zero, the computer takes its next instruction from location Y. Otherwise, the next sequential instruction is executed.
TIX	Y,T,D,	Transfer on Index. If the contents of index register T are greater than D, then D is subtracted from the index register and the computer takes its next instruction from location Y. If the contents of the specified index register are less than or equal to D, the computer executes the next sequential instruction.

OPERATION	VARIABLE FIELD	
TXI	Y,T,D	Transfer With Index Incremented. The decrement is added to the contents of the specified index register, and the computer executes instruction Y.
TSX	( )	Transfer and Set Index is used in conjunction with variable field information to transfer control to another location or to a basic machine subroutine[1].
PZE	( )	Plus Zero has no use by itself. It is used with tape reading commands such as IORT which will be discussed shortly.
ADD	N	Add N to the contents of the accumulator. The sum is stored in the accumulator.
SUB	N	Subtract N from the contents of the accumulator. The result is stored in the accumulator.
LDQ	Y	Load MQ. This instruction places the contents of Y into the MQ. The contents of Y are unchanged.
BSS	N	Block Starting with Symbol. This instruction provides locations for an array of N variables. A name must be assigned, beginning in column 1. This is analogous to a "dimension" statement in Fortran.

---

1. A basic machine subroutine is one that has been written to perform some function that is frequently requested by programmers, such as a read or write routine. The average programmer has no reason to concern himself with them except to know the available subroutines and request their use.

OPERATION	VARIABLE FIELD	
BCI	N,( )	Binary Code Information. This MAP pseudo-operation provides the programmer with the capability of writing alphanumeric information in Fortran. A name must be assigned. Any Hollerith field data may be written within the parentheses. N equals one sixth of the total number of characters and spaces following the comma.
FILE	( )	The FILE pseudo-operation enables a programmer to specify input and output file requirements for tape reading and writing. There are approximately 30 options that may be written within the parentheses. The options employed by the Combustion Research Facility programs are described in Section 5.2.1. For a detailed description of all available options see References 2 and 7.
IORT	N,,**	IORT is an IOCS (Input/Output Control System) transmitting command that follows a TSX, PZE, PZE 3-card sequence which reads one record of a magnetic tape. The tape data is stored in the locations N through N + n where n+1 is the number of words read. After reading the record, the ** (the decrement position, bits 3 to 17) is replaced by n+1.
CLA*	N,4	Clear and Add Indirectly. The * indicates indirect addressing[1].

1. Indirect addressing means that a command deals with the contents of the contents of a location. Where CLA N indicates clear and add the contents of location N, CLA* N means clear and add the contents of the contents of location N. This process is used for transferring data into and from a MAP program.

OPERATION	VARIABLE FIELD	
CLA* (continued)		In the programs that concern this report, the instruction is used to transfer data from a calling program argument to a MAP subroutine. Index register 4 is always used for the transfer process. N equals the calling statement argument number plus 2. For the second argument, N would equal 4. This command would transfer the value of the second argument of the CALL statement to the subroutine containing CLA* N,4.
STO	N,4	Store, Indirectly. This is similar to STO*. The contents of the accumulator are transferred to the calling program in this case. Index register 4 is again used and the manner of selecting N is identical with the above.

APPENDIX B

MAP SUBROUTINES

APPENDIX B

MAP SUBROUTINES

IOCS13

```
$IBMAP  IOCS13
IN1     FILE    INPUTFILE,A(1),INPUT,BLK=200,HIGH,BIN,800,MOUNT,
        ETC     HOLD
        ENTRY   OPEN
        ENTRY   READ
        ENTRY   EFF
OPEN    SAVE    4
        TSX     .OPEN,4
        PZE     IN1
        RETURN  OPEN
READ    SXA     SV1,4
        TSX     .READ,4
        PZE     IN1,,EOB
        PZE     EOF,,ERR
        IORT    DATA1,,**
GO      CLA     *-1
        STZ     DECR
        STD     DECR
        CLA     DECR
        ARS     18
        STO     N
        STA     WC
WC      AXT     **,1
        AXT     1000,2
SV1     AXT     **,4
        CLA     3,4
        ADD     N
        STA     OUTPUT
        CLA     DATA1+1000,2
OUTPUT  STO     **,1
        TXI     *+1,2,-1
        TIX     *-3,1,1
        CLA     4,4
        STA     COUNT
        CLA     N
```

```
COUNT     STO       **
HOME      TRA       1,4
EOB       PZE       **
EOF       XEC       SV1       EXECUTE THE SV1 CARD
          TRA       5,4       TRANSFER TO LOCATION CITED IN 3RD,
*                             ARG.OF CALL
ERR       SXA       SV2,4
          CALL      .FWRD.(.UN06.,111.)
          TSX       .FCNV.,4
          TSX       .FFIL.,4
SV2       AXT       **,4
          TRA       GO
EFF       SAVE      4
          TSX       .CLOSE,4
          PZE       IN1
          RETURN    EFF
DECR      BSS       1
N         BSS       1
DATA1     BSS       200
111.      BCI       3,(10H WARNING  )
          END
```

NOTE

A * in the first column of a MAP instruction specifies a comment card. All information contained on this card is ignored by the computer.

BCD

```
$IBMAP    BCD
BCD       SAVE    1,2,3,4,5,6,7
          CLA*    3,4       GET 1ST, ARGUMENT VALUE
          STO     LENGTH    INPUT ARRAY LENGTH
          STA     *+1
          AXT     **,1      INPUT ARRAY SIZE COUNTER
          CLA*    4,4       GET VALUE OF 2ND, ARGUMENT
          STO     LOUT      STORE OUTPUT ARRAY LENGTH
          STA     OCOUNT    SET OUTPUT ARRAY SIZE COUNTER,XR2
          STA     BEGIN
          AXT     200,2     FIRST USE XR2 AS MAX. INPUT ARRAY CTR.
          CLA     5,4       GET ADDRESS(LOCATION) OF ARG. 3
          ADD     LENGTH    INCREASE SIZE OF ARG. 1
          STA     *+1       STORE ADDRESS OF ARG. 1
          CLA     **,1      ADD FIRST ARG. TO AC
          STO     AREA1+200,2   STORE FIRST ARGUMENT
          TXI     *+1,2,-1
          TIX     *-3,1,1   GET  ANOTHER INPUT VALUE
          CLA     6,4       GET ADDRESS OF ARG.4
          ADD     LOUT      FORM ADDRESS OF OUTPUT DATA
          STA     OUT       STORE ADDRESS OF OUTPUT DATA
          CLA*    7,4       GET NUMBER OF DIGITS IN EACH OUTPUT
*                           WORD
          STO     DIGITS
          STA     START
          CLA     DIGITS
          SUB     =1
          XCA
          MPY     =3
          STQ     CONTRL    THIS CONTROLS THE MULTIPLICATION SECTION
          CLA     CONTRL
          STA     START+1
BEGIN     AXT     **,1
          AXT     600,7     MAX. OUTPUT ARRAY COUNT
          AXT     6,6       PREVENTS LOADING OF 2ND. INPUT
 *                          WORD TOO SOON
          AXT     200,2     MAX. INPUT ARRAY COUNT
          LDQ     AREA1+200,2   LOAD ONE INPUT WORD
START     AXT     **,3      DIGITS COUNTER
          AXT     **,5      CONTROLS MULT SECTION
          STZ     TEMP2
LOOP      ZAC
          LGL     6
          STQ     TEMP1
```

```
        STO    TEMP3
        SUB    =10
        TZE    ENTER
        LDQ    TEMP3     LOAD SUBJECT INTO MQ
MULTPY  TRA    MULT+15,5 MULTIPLY SUBJECT BY 100000
MULT    MPY    =100000   OR 10000, OR 1000, OR 100,
        XCA              OR 10, OR 1.
        TRA    ENTER
        MPY    =10000
        XCA              EXCHANGE C(MQ) AND C(AC)WWITH XCA
        TRA    ENTER     COMMAND SINCE THE PRODUCT OF MULT
        MPY    =1000     WILL BE IN THE MQ.  WE WANT TO ADD
        XCA              THE PREVIOUSLY PROCESSED DIGITS,
        TRA    ENTER     C(TEMP2), TO THE PRESENT SUBJECT;
        MPY    =100      THEREFORE, PLACE THE SUBJECT INTO
        XCA              THE ACCUMULATOR BY EXECUTING THE
        TRA    ENTER     XCA COMMAND (EXCHANGE C(MQ) AND
        MPY    =10       C(MQ).
        XCA
        TRA    ENTER
        MPY    =1
        XCA
ENTER   ADD    TEMP2     ADD PREVIOUS PROCESSED DATA TO SUBJECT
ZERO    TXI    *+1,5,-3  DECREASE XR5, MULT COUNTER, BY 1
        STO    TEMP2     STORE PROCESSED DATA TEMPORARILY
        LDQ    TEMP1     LOAD UNPROCESSED DATA
        TIX    PASS,6,1
        TXI    *+1,2,-1
        LDQ    AREA1+200,2
        AXT    6,6
PASS    TIX    LOOP,3,1  TAXI TO LOOP; DECREASE XR3 BY 1
        STO    ID+600,7  STORE PROCESSED DATA
        TXI    *+1,7,-1
        TIX    START,1,1  GET ANOTHER WORD  FOR PROCESSING
OCOUNT  AXT    **,2
        AXT    600,7      TRANSFER PROCESSED DATA FROM
        CLA    ID+600,7   BCD SUBROUTINE TO MAIN PROGRAM
OUT     STO    **,2
        TXI    *+1,7,-1
        TIX    *-3,2,1
        RETURN BCD
```

```
AREA1     BSS     300
ID        BSS     600
TEMP1     BSS     1
TEMP2     BSS     1
TEMP3     BSS     1
LENGTH    BSS     1
LOUT      BSS     1
DIGITS    BSS     1
CONTRL    BSS     1
          END
```

UNPACK

```
$IBMAP MAP3
UNPACK SAVE     1,3,5,7,2,6
       CLA*     6,4  GET VALUE OF LAST ARGUMENT
       STO      LENGTH
       STA      *+1  STORE VALUE OF LAST ARG. INTO ** BELOW
       AXT      **,6
       STA      COUNT
       AXT      1031,7
       CLA      3,4  GET 1ST. ARG. OF CALL STATEMENT
       ADD      LENGTH
       STA      *+1  STORE ADDRESS INTO NEXT INSTRUCTION
       CLA      **,6  ** IS REPLACED BY ADDRESS OF ABOVE INSTR.
       STO      AREA2+1031,7
       TXI      *+1,7,-1
       TIX      *-3,6,1  DECREASE C(XR6) BY 1 & GO BACK 3 CARDS
       AXT      1031,7
       AXT      2060,2
*      SPLIT OUT ZERO'S AND PROCESS HEADER ID DATA
COUNT  AXT      **,6
START  AXT      6,3    INITIALIZE
       AXT      15,5   INITIALIZE
       STZ      TEMP2  TEMP2 = STORAGE FOR PROCESSED DATA
       LDQ      AREA2+1031,7   LOAD 1 IBM WORD
LOOP   ZAC                     CLEAR ACCUMULATOR
       LGL      6   SHIFT 1ST. 2 OCTAL DIGITS (THE SUBJECT) INTO AC.
       STQ     TEMP1    STORE REMAINDER OF MQ DATA
       STO     TEMP3    STORE THE SUBJECT
       SUB     =10      TEST FOR AN IBM ZERO
       TZE     ENTER    IF SUBJECT IS AN IBM 0, BYPASS MULTPY
       LDQ     TEMP3    LOAD SUBJECT INTO MQ
MULTPY TRA     MULT+15,5  MULTIPLY SUBJECT BY 100,000, OR 10,000
MULT   MPY     =100000    OR 1,000, OR 100, OR 10, OR 1
       XCA
       TRA     ENTER
       MPY     =10000
       XCA                EXCHANGE C(MO) AND C(AC) SINCE THE
       TRA     ENTER      PRODUCT OF MULT WILL BE IN THE MQ.
       MPY     =1000      WE WANT TO ADD THE PREVIOUSLY
       XCA                PROCESSED DIGITS, C(TEMP2), TO THE SUBJECT,
       TRA     ENTER      THEREFORE PLACE SUBJECT INTO AC. BY
       MPY     =100       EXECUTING XCA, EXCHANGE C(MQ) AND C(AC).
       XCA
       TRA     ENTER
       MPY     =10
       XCA
```

```
        TRA     ENTER
        MPY     =1
        XCA
        TRA     ENTER
ENTER   ADD     TEMP2  ADD PREVIOUSLY PROCESSED DATA TO
*                      PRESENT SUBJECT
ZERO    TXI     *+1,5,-3    DECREASE XR5, MULT COUNTER, BY 1
        STO     TEMP2       STORE PROCESSED DATA TEMPORARILY
        LDQ     TEMP1       LOAD UNPROCESSED DATA
        TIX     LOOP,3,1    TAXI TO LOOP, DECREASE XR3 BY 1
        STO     TIME        STORE FINAL PROCESSED WORD
        TXI     *+1,6,-1
        TXI     *+1,7,-1
SORT    LDQ     AREA2+1031,7
        ZAC
        LGL     3     SHIFT SIGN INDICATION INTO AC
        TZE     PLUS1  IF DATA IS +, C(AC)=0, GO TO PLUS1
MINUS1  ZAC            IF DATA IS -, THIS INSTR. IS FOLLOWED
        LGL     15     SHIFT MAGNITUDE OF DATA INTO AC.
        SUB     =037777
        STO     DATA+2060,2
        TRA     LAST   UNPACK LAST HALF OF DATA WORD
PLUS1   LGL     15     SHIFT MAGNITUDE OF DATA INTO AC.
        STO     DATA+2060,2
LAST    ZAC            CLEAR THE ACCUMULATOR
        TXI     *+1,2,-1
        LGL     3      SHIFT SIGN INDICATOR INTO ACCUMULATOR
        TZE     PLUS2  IF DATA IS +, C(AC)=0, GO TO PLUS2
MINUS2  ZAC            IF DATA IS -, THIS WILL BE FOLLOWED
        LGL     15     SHIFT MAGNITUDE OF DATA INTO AC.
        SUB     =037777
        STO     DATA+2060,2
        TRA     REPEAT ALTER XR'S AND UNPACK ANOTHER NUMBER
PLUS2   LGL     15     SHIFT MAGNITUDE OF DATA INTO AC.
        STO     DATA+2060,2
```

```
REPEAT    TXI      *+1,7,-1
          TXI      *+1,2,-1
          TIX      SORT,6,1       DECREASE C(XR6) BY 1 AND GO TO SORT
          CLA      4,4
          STA      OUT
          CLA      LENGTH
          SUB      =1
          XCA                     PUT SUBJECT INTO MQ
          ZAC                     CLEAR AC
          MPY      =2             MULTIPLY BY 2
          XCA
          STO      LGTH
          STA      COUNT2
          CLA      5,4
          ADD      LGTH
          STA      OUT1
          CLA      TIME
OUT       STO      **
COUNT2    AXT      **,3
          AXT      2060,2
          CLA      DATA+2060,2
OUT1      STO      **,3
          TXI      *+1,2,-1
          TIX      *-3,3,1
          RETURN   UNPACK
TIME      BSS      2
TEMP1     BSS      1
TEMP2     BSS      1
TEMP3     BSS      1
ID        BSS      6
AREA2     BSS      1031
DATA      BSS      2060
LENGTH    BSS      1
LGTH      BSS      1
          END
```

APPENDIX C

FORTRAN SUBROUTINES

APPENDIX C

FORTRAN SUBROUTINES

COPY1

```
$IBFTC     SUB1
           SUBROUTINE COPY1 (L,N)
           DIMENSION L (80)
  101      FORMAT(1H1)
   12      FORMAT (2X,6I8)
           WRITE( 6,12) (L(J), J = 1,N)
           WRITE(2) (L( J), J = 1,N)
           RETURN
           END
```

COPY2

```
$IBFTC      SUB2
            SUBROUTINE COPY2 (L,N)
            DIMENSION L (80)
    1       FORMAT(1H1)
   11       FORMAT (2X, 6I8)
            READ (2)  (L(J), J = 1,N)
            WRITE (6,11)  (L(J) , J = 1,N)
            RETURN
            END
```

PRELIM

```
$IBFTC        SUB3
              SUBROUTINE PRELIM (ID,N)
              INTEGER ID(200)
 9999         FORMAT (1H1)
 1000         FORMAT (5X,6HDATE   ,I2,1H/,I2,1H/,I2,//)
              WRITE(6,1000) ID(1),  ID(2),  ID(3)
C
C
C
C.....ID(4) = NUMBER OF CHANNELS
C
C
C
C.....ID(5) = RUN TYPE, EXPERIMENTAL OR CALIBRATION
              IF (  ID(5)  .EQ. 0 ) WRITE (6,10)
              IF (  ID(5)  .EQ. 1 ) WRITE (6,11)
          10  FORMAT(5X, 16HEXPERIMENTAL RUN//)
          11  FORMAT(5X,16H CALIBRATION RUN//)
C
C
C
C.....ID(6) = COOLING DESIGNATION
              IF ( ID(6) .EQ. 0 ) WRITE(6,20)
              IF ( ID(6) .EQ. 1 ) WRITE(6,21)
              IF ( ID(6) .EQ. 2 ) WRITE(6,22)
              IF ( ID(6) .EQ. 3 ) WRITE(6,23)
          20  FORMAT( 5X, 10H UNCOOLED //)
          21  FORMAT(5X, 12H FILM COOLED //)
          22  FORMAT(5X, 21HTRANSPIRATION COOLING //)
          23  FORMAT(5X, 21H REGENERATIVE COOLING //)
C
C
C
C.....ID(7) = FUEL TYPE
              IF ( ID(7) .EQ. 0) WRITE(6,30)
              IF ( ID(7) .EQ. 1) WRITE(6,31)
              IF ( ID(7) .EQ. 2) WRITE(6,32)
              IF ( ID(7) .EQ. 3) WRITE(6,33)
          30  FORMAT (5X, 5H UDMH //)
          31  FORMAT(5X, 10H HYDRAZINE //)
          32  FORMAT(5X,4H LH2 //)
          33  FORMAT( 5X, 5H A-50 /)
```

```
C
C
C
C.....ID(8) = OXIDIZER TYPE
      IF ( ID(8) .EQ. 0 )  WRITE(6,40)
      IF ( ID(8) .EQ. 1 )  WRITE(6,41)
      IF ( ID(8) .EQ. 2 )  WRITE(6,42)
   40 FORMAT (5X, 19H NITROGEN TETROXIDE  //)
   41 FORMAT (5X,  4H LOX  //)
   42 FORMAT (5X,  4H AIR  //)
C
C
C
C.....ID(9) = OPERATOR
      IF ( ID(9) .EQ. 1 )  WRITE(6,1)
      IF ( ID(9) .EQ. 2 )  WRITE(6,2)
      IF ( ID(9) .EQ. 3 )  WRITE(6,3)
      IF ( ID(9) .EQ. 4 )  WRITE(6,4)
    1 FORMAT (5X, 15HR. L. STRICKLER ///)
    2 FORMAT (5X, 15HT. W. CARPENTER ///)
    3 FORMAT (5X, 15H C. M. EHRESMAN ///)
    4 FORMAT (5X, 12H C. A. BRYCE ///)
C
C
C
      RETURN
      END
```

THERM

```
$IBFTC SUB4
      SUBROUTINE THERM (ID, N)
      INTEGER ID (60)
    1 FORMAT (5X, I4,2X,20HFOUR CHARACTER WORDS //)
    2 FORMAT (5X, 12HTHERMOCOUPLE ,2X,I8)
    3 FORMAT (5X, 18HTEMPERATURE RANGE ,2X,I4,3H
     1  TO ,2X,I4//)
    4 FORMAT (1H1)
      WRITE (6,1) N
      WRITE (6,2) ID(1)
      WRITE (6,3) ID(2), ID(3)
      WRITE (6,2) ID(4)
      WRITE (6,3) ID(5), ID(6)
      WRITE (6,2) ID(7)
      WRITE (6,3) ID(8), ID(9)
      RETURN
      END
```

(A sample computer output page, Table D2, appears is Appendix D.)

PTRANS

```
$IBFTC SUB5
      SUBROUTINE PTRANS (ID,N)
      INTEGER ID(40)
    4 FORMAT (1H1)
    1 FORMAT (5X,I4,20H SIX CHARACTER WORDS  //)
      WRITE (6,1) N
   10 FORMAT (5X,10HTRANSDUCER  ,2X,I8  )
   11 FORMAT (5X,14HMAX. PRESSURE  ,2X,I8//)
      WRITE (6,10) ID(1)
      WRITE (6,11) ID(2)
      WRITE (6,10) ID(3)
      WRITE (6,11) ID(4)
      WRITE (6,10) ID(5)
      WRITE (6,11) ID(6)
      RETURN
      END
```

(A sample computer output page, Table D3, appears in Appendix D.)

PIF3

```
$IBFTC    SUB6
CPIF3
          FUNCTION PIF3 (X,XLIST,N,FLIST)
C
          DIMENSION XLIST(100), FLIST(100)
C         DIMENSION XLIST(100), FLIST(100)
          BLI(Q000FL,Q001FL,Q002FL,Q003FL,Q004FL) =
         1((Q001FL-Q000FL)* Q003FL-
         2Q004FL)/(Q002FL-Q001FL)+Q003FL)
          IF(X-XLIST(N))  2,1,1
        1 I=N-1
          K=1
          GO TO 30
        2 IF (X-XLIST(1)) 4,4,3
        4 I=1
          K=1
          GO TO 30
        3 DO 6 I=1,N
          IF (X-XLIST(I))7,7,6
        6 CONTINUE
          I=N
        7 I=I-1
          IF((I+1)-N) 11,5,11
       11 IF (I-1) 30,5,24
        5 K = 2
          GO TO 30
       24 K=3
       30 BLIF1 = BLI(X,XLIST(I),XLIST(I+1),FLIST(I),FLIST(I+1))
       10 IF(K-1) 23,23,12
       23 PIF3 =BLIF1
          RETURN
       12 IF((I+2)-N) 13,13,16
       13 IF((I-1)-1) 14,15,15
       14 L=I+2
          GO TO 17
       15 L=I+2
          GO TO 17
       16 L=I-1
       17 BLIF2 = BLI(X,XLIST(I),XLIST(L),FLIST(I),FLIST(L))
          BLIF3 = BLI(X,XLIST(I+1),XLIST(L),BLIF1,BLIF2)
          IF (K-2) 19,19,27
       19 PIF3 = BLIF3
          RETURN
```

```
27 L=I-1                                                    PIF3042
37 BLIF4 =BLI(X,XLIST(I),XLIST(L),FLIST(I),FLIST(L))        PIF3043
   BLIF5=BLI(X,XLIST(I+1)XLIST(L),BLIF1,BLIF4)              PIF3044
21 PIF3 =BLI(X,XLIST(I+2),XLIST(L),BLIF3,BLIF5)             PIF3045
   RETURN                                                   PIF3046
   END                                                      PIF3047
```

TABLE

```
$IBFTC SUB7
      SUBROUTINE TABLE (VOLT,NPTS,T,NROW,FIRST,LAST,
     1OUT,L,M,TEMP,LL,MM)
      REAL OUT(L,M), TEMP(LL,MM),VOLT(300),  T(300)
      INTEGER  FIRST
      DO 1 J = FIRST, LAST
      DO 1 K = 1, NROW
      A = OUT(J,K)
    1 TEMP (J,K) = PIF3 (A, VOLT, NPTS,  T)
      DO 11 J = FIRST,LAST
   11 WRITE(2)  (TEMP(J,K), K = 1,NROW)
      RETURN
      END
```

LINEAR

```
$IBFTC SUB8
C      *
C      *
C      Y = A*X + B
C      *
C      *
          DIMENSION X(200), Y(200), XX(200), YY(200)
        1 FORMAT (I5, I9)
       44 READ (5,1) N , NBR
          DO 2 I = 1,N
        2 READ (5,3) X(I), Y(I)
        3 FORMAT (2F10.0)
          WRITE (6,99)
       99 FORMAT (1H1,//10X,4HDATA, // 8X,2HX , 8X, 2HY //)
          DO 89 I = 1, N
       89 WRITE (6,79) X(I), Y(I)
       79 FORMAT (3X, 2F10.4)
      444 READ (5,59) TEST
       59 FORMAT (F10.0)
          TEST1 = 100. * TEST
          NTEST = TEST1
C        IF NTEST = 0. THE PROGRAM WILL LOOK FOR MORE
         EXPERIMENTAL DATA
C        THEREFORE A DATA CARD WITH TEST = 0. MUST
C        PRECEED THE SECOND SET OF EXPERIMENTAL DATA
          IF(NTEST .EQ. 0 ) GO TO 4400
        8 NN = N
          YSUM = 0.
          DO 4 I = 1,N
        4 YSUM = YSUM + Y(I)
          XSUM = 0.
          DO 5 I = 1,N
        5 XSUM = XSUM + X(I)
C        SINCE N, THE NUMBER OF DATA POINTS, WILL BE USED FREQUENTLY AS
C        A FLOATING POINT NUMBER, REDEFINE IT AS     AN = N
          AN = N
          WRITE(6,100)
      100 FORMAT (1H1)
       69 FORMAT (8X,57HSOLVE THE NEXT 2 EQUATIONS FOR SLOPE, M, AND
         1INTERCEPT, B   )

          WRITE (6,69)
      101 FORMAT(6X,F10.4,4H = ,F8.2,5HM  +  ,F8.2,1HB)
          WRITE(6,101) YSUM, XSUM, AN
          XYSUM = 0.
          DO 6 I = 1,N
          XY = X(I) * Y(I)
```

```
    6 XYSUM = XYSUM + XY
      XXSUM = 0.
      DO 7 I = 1,N
      XX = X(I)*X(I)
    7 XXSUM = XXSUM + XX
      AA = XSUM / FLOAT(N)
C     AM = SLOPE
C     B = INTERCEPT
      AM = (XYSUM-AA*YSUM)/(XXSUM-AA*XSUM)
      B = (YSUM - XSUM*AM) / FLOAT(N)
  102 FORMAT (6X,F12.4,2H = ,F12.2,5HM +  ,F12.4,1HB////)
      WRITE (6,102) XYSUM,XXSUM,XSUM
  103 FORMAT (6X,4HY = ,F12.4,5HX +  ,F12.4//)
      WRITE (6,103) AM, B
      EYSQ = 0.
      DO 9 I = 1, N
      YCALC = AM * X(I) + B
      EY = ABS (YCALC - Y(I))
C     SUM THE SQUARES OF EPSILON-Y
    9 EYSQ = EYSQ + EY*EY
      SIGMAY = SQRT (EYSQ/(AN - 2.0))
C     SIGMA  = STANDARD ERROR FOR SLOPE, M
C     SIGMAY = STANDARD ERROR FOR DEPENDENT VARIABLE, Y
      SIGMA = SQRT ((AN*EYSQ)/((AN - 2.)*(XXSUM*AN - XSUM*XSUM)))
  104 FORMAT (9HSIGMAY = F10.6,5X,10HDIFF  .LT.  F10.6///)
 1104 FORMAT (9HSIGMA  = F10.6,5X,10HDIFF  .LT.  F10.6///)
      WRITE (6,1104) SIGMA, TEST
      WRITE (6,104 ) SIGMAY,TEST
  144 FORMAT (8X, 5HYDATA, 8X,5HYCALC, 7X, 4HDIFF//)
      WRITE (6,144)
      I = 1
      XX(I) = X(I)
      YY(I) = Y(I)
      J = I
   11 IF ( I .GT. NN )  GO TO 33
   12 YCALC = AM * XX(J) + B
      DIFF = ABS (YCALC - YY(J))
      WRITE (6,300)  YY(J), YCALC, DIFF
  300 FORMAT (3X, 3F12.5)
      I = I + 1
      IF (DIFF - TEST) 10,10,20
   10 J = J + 1
      XX(J) = X(I)
      YY(J) = Y(I)
      GO TO 11
C     THROW OUT CURRENT DATA POINT
C     DECREASE NUMBER PF DATA POINTS BY 1
```

```
   20 N = N - 1
      IF ( I .GT. NN ) GO TO 33
      XX(J) = X(I)
      YY(J) = Y(I)
      GO TO 12
C     STORE X(I)
   33 DO 13 I = 1,N
      Y(I) = YY(I)
   13 X(I) = XX(I)
 1000 IF (N .EQ. NN) GO TO 444
 1001 GO TO 8
 4400 READ (5,2000) XPB, YPB
 2000 FORMAT ( 2F10.0)
      YCOMP = AM * XPB + B
      PBDIFF = YPB - YCOMP
      WRITE (6,2001) NBR, AM, B
 2001 FORMAT(3X,10HTRANSDUCER,I8,2X,5H Y = ,F12.4,6H X + ,F12.4/)
      WRITE (6,2002) XPB, YPB, YCOMP, PBDIFF
 2002 FORMAT(8X,4HX = ,F9.4//,8X,4HY = F9.4//, 8X, 13HY,
     1COMPUTED = , F9.4//, 8X, 13HDIFFERENCE = ,F9.4///)
      GO TO 44
      STOP
      END
```

PSIG

```
$IBFTC     SUB9
          SUBROUTINE PSIG (NROW, FIRST, LAST, SLOPE, YINT, SIGMA, MAX
         1 DEV , TIME , OUT,L,M, PSI, LL, MM)
           REAL OUT(L,M), PSI(LL,MM)
           REAL MAXDEV
           INTEGER FIRST , TIME
           DO 1  J = FIRST, LAST
           DO 1  K = 1, NROW
         1 PSI(J,K) = OUT(J,K) * SLOPE + YINT
         4 FORMAT (1H1)
           DO 11 J = FIRST, LAST
        11 WRITE (2)   (PSI(J,K) , K = 1,NROW)
           RETURN
           END
```

WREU

```
$IBFTC SUB10
      SUBROUTINE WREU (OUT,L,M,NCIU,TIME)
      INTEGER  TIME
      DIMENSION  OUT(L,M)
 1966 FORMAT (1H1)
 1967 FORMAT (/////)
      WRITE (6,1966)
      WRITE (6,1967)
      NN = (N-1)*2
      NROW = N*2/NCIU
      NHALF = NCIU
      IF (NCIU .GT. 10) NHALF = 10
C     SUBSCRIPT 1 = CHANNEL NUMBER
C     SUBSCRIPT 2 = ROW NUMBER OR DATA POINT NUMBER
      WRITE (6,33) TIME
   33 FORMAT (5X, 4HTIME, I8//)
      IF ( NHALF .LE. 5 ) GO TO 43
   34 FORMAT (2X,7HCHANNEL,/ 2X,6HNUMBER,2X,2H1  ,8X,3H 2
     1,8X,3H 3 ,8X,3H 4 ,8X,3H 5 ,8X,3H 6 ,8X,3H 7
     2,8X,3H 8 ,8X,3H 9 ,8X,3H10 /)
      WRITE (6,34)
      DO 35 K = 1,NROW
   35 WRITE (6,36)  (OUT(I,K), I = 1,NHALF)
   36 FORMAT (3X, 10F11.3)
      IF ( NCIU .LE. 10 )  GO TO 53
      WRITE (6,1967)
   37 FORMAT (2X,7HCHANNEL,/ 2X,6HNUMBER,2X,3H11 ,8X,3H12
     1,8X,3H13 ,8X,3H14 ,8X,3H15 ,8X,3H16 ,8X,3H17 ,8X,3H18
     2,8X,3H19 ,8X,3H20 /)
      WRITE (6,37)
      DO 38 K = 1,NROW
      NHALF1= NHALF + 1
   38 WRITE (6,36)  (OUT(I,K), I = NHALF1, NCIU)
      GO TO 53
   43 WRITE (6,44)
   44 FORMAT (5X,7HCHANNEL,/ 5X,6HNUMBER,2X,3H 1 ,6X,3H 2 ,6X,
     1 3H 3 ,6X,3H 4 ,6X,3H 5 )
      DO 45 I = 1,NROW
   45 WRITE (6,46)  (OUT(I,K), K = 1,NCIU)
   46 FORMAT (7X, 5F12.3)
   53 RETURN
      END
```

(A sample computer output page, Table D4, appears in Appendix D.)

MAIN

```
$ID    3107*10*100**BARSIC*COMBUSTION RESEARCH*
$$$TAPE INPUTFILE        JPC EECO TAPE NUMBER 1     RING OUT  HIGH SAVE
$EXECUTE        IBJOB
$IBJOB          MAP
$IBFTC MAIN     LIST
      INTEGER PRE(20), IDPRE(60), IDID(60), LABEL6(40), ID6(40), ID66(40
     1), LABEL4(40), ID4(60), ID44(60), AREA(141), MAP(282), ISC(80),
     2IDSC(80), SC(80)
      INTEGER TIME , FIRST
      REAL OUT(20,30 ) , SCF(200),PSI(20,30) , VIC(300), TIC(300), VCA
     1 (300), TCA(300), EUNITS(20,30) , VP(300), WT(300) , TEMP(20,30)
      REWIND 2
      CALL OPEN
      CALL READ (PRE, NPRE, $60 )
      WRITE(6,1)
  130 FORMAT (2X, I12/)
      NPRE3 = NPRE * 3
      CALL BCD ( NPRE, NPRE3, PRE, IDPRE, 2)
      CALL COPY1 ( IDPRE, NPRE3)
      CALL PRELIM ( IDPRE, NPRE3 )
      NCIU = IDPRE(4)
C     NCIU = NUMBER OF CHANNELS IN USE
C     PRE = PRELIMINARY DATA       UN-PROCESSED BY BCD
C     NPRE = NUMBER OF IBM WORDS IN PRELIMINARY RECORD
C     $60 TRANSFERS TO END OF PROGRAM IF A TAPE MARK IS ENCOUNTERED
C     $60 IS A NON-STANDARD RETURN
C     IDPRE = PRELIMINARY DATA ALREADY PROCESSED BY BCD
C     NPRE3 = NUMBER OF OUTPUT DIGIT PAIRS FROM BCD
      CALL READ( LABEL6, N6, $60)
      WRITE(6,1)
      CALL BCD (N6, N6, LABEL6, ID6, 6)
      CALL COPY1 (ID6, N6)
      CALL PTRANS(ID6, N6)
      CALL READ (LABEL4, N4, $60)
      WRITE(6,1)
      N4X   =  N4  *  3/2
      CALL BCD   ( N4, N4X, LABEL4, ID4, 4)
      CALL COPY1 (ID4, N4X)
      CALL THERM (ID4, N4X)
      CALL READ(IDSC, NS, $60)
      WRITE(6,1)
      CALL BCD (NS, NS, IDSC, SC, 6)
      CALL COPY1 (SC, NS)
      DO 2 I = 1, NS
    2 SCF (I) = SC (I)
```

```
      A1 = SCF(1)/100000.
      B1 = SCF(2)/100000.
      S1 = SCF(3)/100000.
      D1 = SCF(4)/10000.
      A2 = SCF(5)/10000.
      B2 = SCF(6)/1000.
      S2 = SCF(7)/100000.
      D2 = SCF(8)/100000.
      A3 = SCF(9)/10000.
      B3 = SCF(10)/10000.
      S3 = SCF(11)/10000.
      D3 = SCF(12)/100000.
      A4 = SCF(13)/10000.
      B4 = SCF(14)/10000.
      S4 = SCF(15)/100000.
      D4 = SCF(16)/100000.
      A5 = SCF(17)/10000.
      B5 = SCF(18)/100000.
      S5 = SCF(19)/10000.
      D5 = SCF(20)/1000.
  500 FORMAT (I3)
  510 FORMAT (2F10.0)
      READ (5,500) NICP
      READ (5,510)  (VIC(I), TIC(I), I = 1,NICP)
      DO 555 I = 1, NICP
  555 VIC(I) = VIC(I) * 200.
C     NDR = NUMBER OF DATA RECORDS
      NDR = 0
 1001 FORMAT (5X, 18HDATA RECORD NUMBER ,3X, I8///)
    1 FORMAT (1H1)
      WRITE (6,1)
 1000 NDR = NDR + 1
      CALL READ (AREA, N, $60)
      CALL UNPACK (AREA, TIME, MAP, N)
 1002 FORMAT (5X, 4HTIME,3X, I8//)
      NROW = N*2/NCIU
      NN = ( N-1) * 2
      M1 = 0
      DO 112 I = 1,NCIU
      M1 = M1 + 1
      K = 1
      DO 66 II = M1, NN, NCIU
      OUT(I,K) = MAP(II)
      OUT(I,K) = OUT(I,K)/1000.
   66 K = K + 1
  112 CONTINUE
C     SUBSCRIPT 1 = CHANNEL NUMBER
C     SUBSCRIPT 2 = ROW NUMBER OR DATA POINT NUMBER
```

```
2121   WRITE(2) TIME
       FIRST = 1
       LAST = 7
       CALL TABLE (VIC, NICP, TIC, NROW, FIRST, LAST, OUT, 20, 30,
      1TEMP, 20,30)

       FIRST = LAST + 1
       LAST = 8
       CALL PSIG (   NROW, FIRST, LAST, A1, B1, S1, D1, TIME , OUT,
      120,30,PSI, 20,30)
       FIRST = LAST + 1
       LAST = 9
       CALL PSIG (   NROW, FIRST, LAST, A2, B2, S2, D2, TIME , OUT,
      120,30, PSI, 20,30)
       FIRST = LAST + 1
       LAST = 10
       CALL PSIG (   NROW, FIRST, LAST, A3, B3, S3, D3, TIME , OUT,
      120,30, PSI, 20,30)
       FIRST = LAST + 1
       LAST = 13
       CALL PSIG (   NROW, FIRST, LAST, A4, B4, S4, D4, TIME , OUT,
      120,30, PSI, 20,30)
       FIRST = LAST + 1
       LAST = 16
C      THIS IS ROSEMOUNT DATA
       CALL PSIG (   NROW, FIRST, LAST, A5, B5, S5, D5, TIME , OUT,
      120,30, PSI, 20,30)
       FIRST = LAST + 1
       LAST = 20
C      POTTER FLOWMETER
       CALL TABLE (VP, NP, WT, NROW, FIRST, LAST, OUT, 20, 30, TEMP,
      1  20,30)
       GO TO 1000
   60  CALL EFF
   61  FORMAT ( 5X,32H NON-STD. RETURN HAS BEEN TAKEN//)
       WRITE (6,61)
       REWIND 2
       CALL COPY 2 (IDID, NPRE3)
       CALL COPY 2 (ID66, N6)
       CALL COPY 2 (ID44, N4X)
       CALL COPY 2 (ISC, NS)
       NR = 0
   65  NR = NR + 1
       WRITE(6,1)
       WRITE(6,1001) NR
       WRITE(6,130) NN
```

```
      READ (2) TIME
      DO 11 J = 1,NCIU
   11 READ (2) (EUNITS (J, K ) , K = 1, NROW)
      CALL WREU ( EUNITS, 20, 30, NCIU, N, TIME)
      IF (NR .LT. NDR) GO TO 65
      WRITE FILE 2
      REWIND 2
      STOP
      END
```

APPENDIX D

SAMPLE COMPUTER OUTPUT

APPENDIX D

SAMPLE COMPUTER OUTPUT

TABLE D1

<u>Sample Data From Prelim</u>

DATE 5/23/66

EXPERIMENTAL RUN

FILM COOLED

HYDRAZINE

NITROGEN TETROXIDE

R. L. STRICKLER

TABLE D2

I. D. Data From THERM

21 FOUR CHARACTER WORDS

THERMOCOUPLE 1
TEMPERATURE RANGE 1 TO 3500

THERMOCOUPLE 2
TEMPERATURE RANGE 1 TO 1500

THERMOCOUPLE 3
TEMPERATURE RANGE 250 TO 2500

TABLE D3

I. D. Data From PTRANS

8 SIX CHARACTER WORDS

TRANSDUCER	10275
MAX. PRESSURE	8000 PSIG.
TRANSDUCER	661120
MAX. PRESSURE	10000 PSIG.
TRANSDUCER	661121
MAX. PRESSURE	10000 PSIG.
TRANSDUCER	661122
MAX. PRESSURE	6000 PSIG.

TABLE D4. COMPUTER OUTPUT FOR EXPERIMENTAL DATA

DATA RECORD NUMBER 98
WORDS PER RECORD, 260
TIME 82906

CHANNEL NUMBER 1	2	3	4	5	6	7	8	9	10
429.240	306.358	133.802	576.920	137.432	148.515	171.621	0.	898.037	915.000
428.920	306.523	131.904	575.640	134.856	148.679	171.950	0.	909.032	915.000
429.720	307.019	133.450	576.760	136.254	149.504	171.968	0.	905.034	904.679
429.560	307.515	133.802	576.920	135.382	148.350	172.773	0.	906.034	919.850
431.000	307.681	133.802	576.120	136.767	149.504	172.115	0.	917.029	919.850
432.120	308.177	133.802	576.600	137.100	150.000	171.621	0.	902.035	914.683
431.640	308.671	134.329	577.560	136.254	149.338	172.773	0.	908.033	913.600
432.600	308.836	134.681	577.240	135.906	150.000	172.279	0.	913.000	915.000
433.560	309.000	135.207	577.240	136.600	150.000	173.104	0.	906.030	926.688
431.800	309.000	134.681	577.400	136.600	150.000	173.104	0.	907.000	913.083
431.480	308.836	134.153	577.720	136.254	149.834	173.434	0.	913.670	922.000
433.880	310.286	136.427	577.880	137.927	150.000	173.434	0.	901.060	906.680
434.840	310.926	135.382	578.840	137.100	150.000	173.434	0.	905.034	917.000

CHANNEL NUMBER 11	12	13	14	15	16	17	18	19	20
921.714	0.	0.	78.661	79.704	78.884	-5002.000	-5012.000	-5005.000	-5005.000
912.720	0.	0.	78.643	79.740	78.884	-5002.000	-4999.000	-5000.000	-5010.000
917.800	0.	0.	78.652	79.720	78.802	-5003.000	-5009.000	-5001.000	-5008.000
921.140	0.	0.	78.680	79.705	78.864	-5007.000	-5002.000	-5002.000	-5006.000
913.720	0.	0.	78.680	79.810	78.824	-5002.000	-5002.000	-5001.000	-5003.000
925.700	0.	0.	78.629	79.700	78.824	-5010.000	-5011.000	-5003.000	-5005.000
917.717	0.	0.	78.590	79.723	78.840	-4991.000	-5006.000	-5005.000	-5008.000
912.790	0.	0.	78.630	79.804	78.960	-5004.000	-5013.000	-5002.000	-5003.000
908.000	0.	0.	78.665	79.764	78.884	-5002.000	-4999.000	-5002.000	-5011.000
912.840	0.	0.	78.640	79.700	78.801	-5002.000	-5011.000	-5005.000	-5007.000
918.160	0.	0.	78.642	79.700	78.801	-5005.000	-5003.000	-5000.000	-5006.000
917.000	0.	0.	78.671	79.728	78.860	-5002.000	-5009.000	-5001.000	-5006.000
904.820	0.	0.	78.710	79.640	78.824	-5010.000	-5009.000	-5003.000	-5005.000

APPENDIX E

PRE-RUN I. D. CHECK OUT LIST

APPENDIX E

PRE-RUN I. D. CHECK OUT LIST

IDENTIFICATION DATA

I. D. Record Number 1

Purpose of Record: To record data such as:

a. Characters 1-6, Date

b. Characters 7 & 8, Number of channels in use.

c. Characters 9 & 10, Run type:

0 = EXPERIMENTAL RUN

1 = CALIBRATION RUN

d. Characters 11 & 12, Cooling Designation:

0 = UNCOOLED

1 = FILM

2 = TRANSPIRATION

3 = REGENERATIVE

e. Characters 13 & 14, Fuel type:

0 = UDMH

1 = HYDRAZINE

2 = LH2

3 = A-50

f. Characters 15 & 16, Oxidizer type:

0 = NITROGEN TETROXIDE

1 = LOX

2 = AIR

g. Characters 17 & 18, Experiment Designer:

1 = R. L. Strickler

2 = T. W. Carpenter

3 = C. M. Ehresman

4 = C. A. Bryce

Computer Subroutines Required: READ,BCD,COPY1, PRELIM

Block Length: ________ Characters

Output Word Length: 2 Characters

Thumbwheel Switch Number.....	1	2	3	4	5	6	7	8	9	10	11	12	Check When Entered Into CFCB
													___
													___
													___
													___

I. D. Record Number: ________

Purpose of Record:

Computer Subroutines Required: READ, BCD, COPY 1, _____

Block Length: ______ Characters

Output Word Length: _____ Characters

Thumbwheel Switch

Number:...	1	2	3	4	5	6	7	8	9	10	11	12	Checked When Entered Into CFCB
													___
													___
													___
													___

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Jet Propulsion Center School of Mechanical Engineering Purdue University, Lafayette, Indiana	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP N/A

3. REPORT TITLE

COMPUTER PROGRAMMING FOR A DIGITAL DATA ACQUISITION SYSTEM

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

N/A

5. AUTHOR(S) *(Last name, first name, initial)*

Barsic, Nicholas J.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
August, 1966	190	18

8a. CONTRACT OR GRANT NO. NsG 592 b. PROJECT NO. c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) TM-66-7 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* JPC-426

10. AVAILABILITY/LIMITATION NOTICES

No Restrictions

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
N/A	National Aeronautics and Space Administration

13. ABSTRACT

The objective of this investigation is to establish the digital tape format required by a data acquisition system, develop a method for obtaining records of experimental data in engineering units, and prepare the groundwork for analyzing the data by techniques that are familiar to engineers.

The digital recording subsystem is emphasized since it provides the most accurate data; however, the analog recording components are also described.

A literature survey concerning the statistical analysis of instrumentation measurements included investigations of rectangular, triangular, and Gaussian distribution curves in addition to curve fitting techniques for calibration data, and resulted in computer programs which enable extremely accurate analysis of calibration data from sensing devices.

Obtaining data in engineering units from the digital subsystem required investigating several different computer languages. Fortran and MAP languages were selected to perform all programming tasks such as altering the format of information on the digital output tape from the recording system and converting raw data to engineering units for use in performance calculations. The manner in which computer programs are written is described in order to aid the Jet Propulsion Center personnel should the programs require alteration resulting from data system expansion or modification.

Results of several tests conducted at the Combustion Research Laboratory were recorded, the data read and then analyzed. Samples of the corresponding

computer output appear in Appendix D.

It is recommended that existing computer programs be simplified as they become more familiar to the programmers. Additional programs should be written to calculate heat transfer rates and performance parameters. Transferring the computer programs from punched cards to magnetic tape should be investigated when the combined program size justifies such action.

On line processing offers many advantages which should be investigated when the computer facilities at Purdue University justify purchase of the required equipment.

UNCLASSIFIED
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Data Acquisition Data Recording: Binary, BCD, Digital, Analog Computer Programming, Format Data Sampling, Conversion, Correlation						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2a. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8a. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through
_______________________________."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through
_______________________________."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through
_______________________________."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS)*, *(S)*, *(C)*, or *(U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, roles, and weights is optional.