# A COMPUTER BASED ERROR CONTROL AND DATA MONITORING SYSTEM FOR ANTHROPOMETRIC DATA

by

Robert Michael Leahy

B.A Simon Fraser University 1976

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE (KINESIOLOGY)

in the Department

of

Kinesiology

SIMON FRASER UNIVERSITY

September 1982

APPROVAL

Name: Robert Michael Leahy

Degree: Master of Science (Kinesiology)

Title of thesis: A Computer Based Error Control and Data Monitoring System for Anthropometric Data

Examining Committee:

Dr. T.W. Calvert
Dean, Interdisciplinary Studies
Senior Supervisor

Dr. J. Dickinson
Chairman, Kinesiology Department

Dr. W.D. Ross
Professor, Kinesiology Department

Dr N. Cercone
Chairman, Computing Science Department
Simon Fraser University
External Examiner

Date Approved: 24 September 1982

## ABSTRACT

The purpose of this thesis was to design an interactive micro computer based error detection system to monitor anthropometric data assembly. Errors which occur in anthropometric data collections are discussed and a system which provides for comparison of obtained data with values predicted from a set of predictor equations is described. In demonstration of applicability, a sample of 29 female subjects, who were measured three times on a 32 item anthropometric protocol, was used to develop multiple regression equations (with five predictor variables) and determine tolerance levels. Various tests of the system included randomly introduced errors and the use of the median values as 'true' values to illustrate the effectiveness of the system in detecting gross errors.

Versatility was built into the system to 1) provide for acceptance of virtually any predictor equation and tolerance level (referred to as a MODEL), 2) test remeasured data, 3) provide for editing, modification, storing and retrieval of both data and models and 4) provide on-line printing of results with various format options.

Anticipated use of the system is in anthropometric training, interactive quality control in field studies, and a general scrutiny of any assembly of data where known relationships can predict values within useful tolerance levels.

ACKNOWLEDGEMENT

I wish to express special gratitude and appreciation to all those who have helped prepare this thesis. My particular thanks goes to Drs. T. Calvert, J. Dickinson, and W. Ross for their guidance, support and constructive criticism. They have been enthusiastic and encouraging throughout the project and their help in oganizing this thesis has been invaluable.

I am also very grateful to Kathy Gaul who did most of the text preparation. She was always willing to help and her many comments and suggestions were always constructive and much appreciated.

TABLE OF CONTENTS

## List of Tables

## List of Figures

# I. ERRORS AND THEIR SOURCES

## 1.1. Introduction

Many scientific endeavours are completely dependent on the process of gathering and analyzing data. Whether the data is used to support a hypothesis, test a model or quantify some phenomenon, the validity of the conclusions are dependent on the accuracy of the data. Data containing errors may lead to false predictions and conclusions. The problem may be exacerbated in the application of findings based on the erroneous data. The cost of dealing with the effects of decisions which are based on such data may far outweigh the cost of striving for error free data.

An example of a problem which required considerable expense to correct occurred recently during the collection of data for the Canada Fitness Survey (Ross, 1982). The survey was designed to assess the general fitness level of members of randomly selected households across Canada and to allow access to the data by researchers making comparisons across geographical regions of Canada. A number of teams were formed and each team had responsibility for a geographical portion of Canada. This division of the data collection task presented problems of standardization and consistency. Measurement techniques needed to be standardized in order for comparisons to be made between regions.

Although efforts were made to train each of the teams in the selected measurement protocol, one team measured their entire region with a protocol which was different from the standard. Fortunately this problem was discovered in sufficient time for a number of people to be transported to Ottawa in order to reconcile the problem. The costs that this problem had caused were far greater than those of providing slightly more training for the measurers, and it would have been even higher if the problem had not been detected before the data were released for general use. A method of on-site error detection during data collection would have obviated the problem early in the collection process.

Data collectors are generally concerned with minimizing errors in their data assemblies. However, once the data has been assembled there is, on occasion, little interest in considering the data quality any further. Sterling and Weinkam (1979) pointed out that there may even be a tendancy to resist any suggestion that errors exist in the data, especially if some research and recommendations have already been based on this data.

Sterling and Weinkam (1979) found that their efforts to have corrections made to a large national (American) data base were rejected. It was discovered that up to <u>twenty-five percent</u> of the "former smoking veterans" in the Dorn study (Kahn, 1966) were misclassified. The Dorn study of smoking and mortality is one of the most important collections of data containing information on tobacco smoking and mortality. It has had considerable influence on the American government's Advisory Committee to the Surgeon General and on research concerned with the hazards of smoking. The report of these errors was not welcomed by the author of the major report (Kahn, 1966) or by the Editor of the National Cancer Institute Monograph in which the report was first published. They seemed to "..opt for an adversary rather than a cooperative mode of responding to discovery of errors.." (Sterling and Weinkam, 1979).

It is obvious then, that the discovery of errors should come as early as possible in the life of any archive of data, and certainly before any conclusions are drawn. If errors are discovered during collection it will be more likely that they can be corrected. It will also be more likely that the owners of the data will react to the discovery of errors with a cooperative rather than an adversarial attitude. It must be recognized that the collection of data will never be completely free from errors as long as a human is a direct link in the chain from initial measurement to final recording of the information.

"Human Error" is a well known phrase which implies not only how some errors originate but also that susceptibility to error is an undeniable human trait. For this reason efforts to replace the human part of the data collection chain with a much more reliable component, a machine, is an attractive possibility. This machine, or computer, will itself have the potential of introducing errors but, in contrast to human error, computer error can be eliminated once it is detected. A human may make a mistake and, although he is aware of the error, he may make the same mistake at a later time. Humans are susceptible to distractions and forgetting. If a computer program makes an error then the program can be fixed so that the error will never occur again.

It may not always be practical to remove the human from the chain since the number of important contributions that man makes will also be removed. Trade-offs between the contributions of the human versus the contributions of his replacement (usually a machine or program) need to be considered. In many cases, and perhaps most cases, it is not practical to replace a slow, error-prone, but brilliant component with one that is fast, accurate but stupid.

Three statements can be made concerning the collection of data: 1. Humans are fallible and prone to making errors. 2. It is often impractical and sometimes undesirable to curtail or eliminate the human role in data assembly and resolution. 3. Errors are therefore inevitable.

These points present an obvious problem in that we may need the human to lend some intelligence to the collection of data but that same human will very likely introduce errors of a magnitude which cannot be tolerated.

There are two general approaches to the detection and handling of errors in data. One method is to first assemble all of the data, clean it using some systematic error detection technique (Duquet, Meulenaere and Borms, 1979; Freund and Hartley, 1967; Goldstein, 1960; Leahy, Drinkwater, Marshall, Ross and Vajda, 1980) and then deal with the errors which have been detected. Duquet _et.al_(1979) reported a method in which outliers in human growth data may be identified by selecting all values that are greater than 2.575 standard deviations from the mean. These outliers are then subjected to three subsequent screenings, each of which identifies values as "non-errors", "possible errors", or "certain errors" according to a defined criterion. Goldstein (1960) described a technique for cleaning longitudinal data which involved selecting values which were more than 2.17 standard deviations from the mean and then providing substitute values for errors by fitting curves to the data. Leahy _et.al_ (1980) detected possible errors in longitudinal data by displaying all data points on a computer generated three dimensional display and visually selecting extreme values. Here too, the errors were replaced by values generated by fitting a curve to the data. Freund and Hartley (1967) suggested detecting outliers and then selecting replacements for errors by minimizing the weighted sum of squared differences between original and corrected data. All of these techniques would be enhanced if the detected errors could be replaced by valid measured data rather than statistically generated data. This is possible only if the phenomenon being observed is not affected by the passage of time. It should be noted that unless the detection system is on-line with a computer at measurement time, confirmation by re-measurement is usually not possible and adjustments have to be made by logical determination of cause.

An alternative approach to the post-hoc detection of errors, which would allow immediate correction, is one in which, during the data collection process, the investigator is notified of possible errors by a computer based error control system. This approach would involve careful monitoring of the collection process in an attempt to identify errors as they occur so that they may be corrected while the phenomenon is still available to be measured. This is an obvious advantage since the relevant data can then be re-collected rather than estimated by some statistical method. Statistical estimations are based on underlying assumptions concerning relationships between values, such as simple linear relationships or much more complex relationships that are represented in some curve fitting techniques. Remeasuring avoids the need to make such assumptions and allows the investigator to collect data that are as close as possible to the true values. A further advantage gained by the use of such a system is the elimination of one step in the transmission of information from measurement to recording. If the measured data are entered into the computer for the error scan then the computer can be instructed to make a permanent machine readable record of the data. Hand recording of data on recording sheets would no longer be necessary.

The disadvantages of this approach include problems of cost and portability. Computer hardware and software, especially in studies where data is collected simultaneously at different sites, may not be a viable option because of budget restraints or physical limitations such as the lack of electrical power.

Extensive reviews of available literature (i.e. DIALOG on-line information retrieval service) failed to locate any such system that would be applicable to the collection of anthropometric data. Although a similar approach has been taken by the designers of Data Base Management Systems, where checks are made to ensure that such things as data formats (i.e. integer, real or character) and certain logical constraints are not violated (Date, 1975), this is not applicable to the collection of large amounts of numerical data for disciplines such as anthropometry.

## 1.2. Purpose

The purpose of this project was to construct a computer based error control system which may be used to monitor the data collection process and report to the experimenter any occurrence of a value which, because of its difference from some predicted value, may be erroneous. This gives the experimenter the opportunity to replace erroneous data with measured values rather than with values derived from some statistical technique as in the post-hoc data cleaning methods (i.e. Goldstein, 1977). This system would allow a set of predictor equations and a set of tolerance levels to be defined. Any data that are collected after this definition stage may be compared to a value predicted by the equations. If the collected data differ from the predicted value by more than the given tolerance then the program will inform the experimenter of the discrepancy. The set of equations which are used to predicted the incoming data will, in the context of this thesis, be referred to as a "model".

The system was designed to run on a very small self contained computer. By putting this constraint on the design, the system will be useful in situations where access to large computing facilities is not possible. An Apple II computer with one 5 1/4 inch floppy disk drive and the UCSD PASCAL operating system was the development system. The UCSD operating system, because it is available for most micro computers, ensures reasonable portability between different computers (i.e. different from APPLE). This portability will allow users of the monitoring system to select computer hardware which will be best suited to their circumstances and budget.

In order to design such a system it is important to understand the sources of error and the types of errors common to the collection of anthropometric data.

## 1.3. The Nature of Error

The concept of "error" is fundamental to any discipline that deals with the quantification of phenomena since it represents the distinction between "true" values and "measured" or "observed" values. It is rare that these two values are exactly the same because it is often impossible to obtain the true value directly. Beers (1957) noted that "Except in a few trivial cases (such as the experimental determination of the ratio of the circumference to the diameter of a circle) the true value is unknown and the magnitude of error is hypothetical." When the true value is not directly accessible it must be inferred by some indirect technique such as a measure of central tendency for repeated measurement. The concept of a true value is important since it requires some thought concerning the magnitude of the difference between observed and true. This difference is "error". Ideally, the closer the observed values are to true values the better, and it is important to attempt to minimize the difference between these two values.

The magnitude of deviation in observed values from true values and the source of the deviation is also important when analyzing data. Discrepancies that are repeated and constant are known as systematic errors and errors which, over a number of repeated measures, cause the observed values to cluster around some central point are known as random error. Both systematic and random error may influence observed data and may be difficult to recognize. They affect the data in different ways and in addition require different approaces to interpretation. (See Ross, Hebbelinck, Van Gheluwe, and Lemmens, 1972).

Random error can have considerable influence on the data if the sample size is small. If the sample size increases then the effects of random error will tend to decrease since errors in one direction (i.e. larger than the true value) will cancel those in the other direction (i.e. smaller than the true value). The influence of systematic error on the other hand is not affected by sample size. It is always either larger or smaller than the true value and therefore will not cancel with increased sample size. If systematic error is identified, it is possible to offset its effects by an appropriate adjustment of the data. This is not true with random error. Beers (1957) and Ross, Hebbelinck, Van Gheluwe and Lemmens (1972) provided excellent discussions of systematic and random error and their influences on data.

The problem of identifying errors is somewhat recursive since it is possible to make an error during the attempt to identify an error. For example, an extreme but valid data point may be marked as a random error. On the other hand, a random error may produce a data point which is not extreme relative to the rest of the data and may be difficult to recognize as an error. If it is mandatory that all valid data points be recorded then probably some erroneous data will be classified as valid and recorded as well. If the detection technique is designed to ensure that all invalid points are rejected then it will probably be at the expense of allowing some valid points to be rejected also.

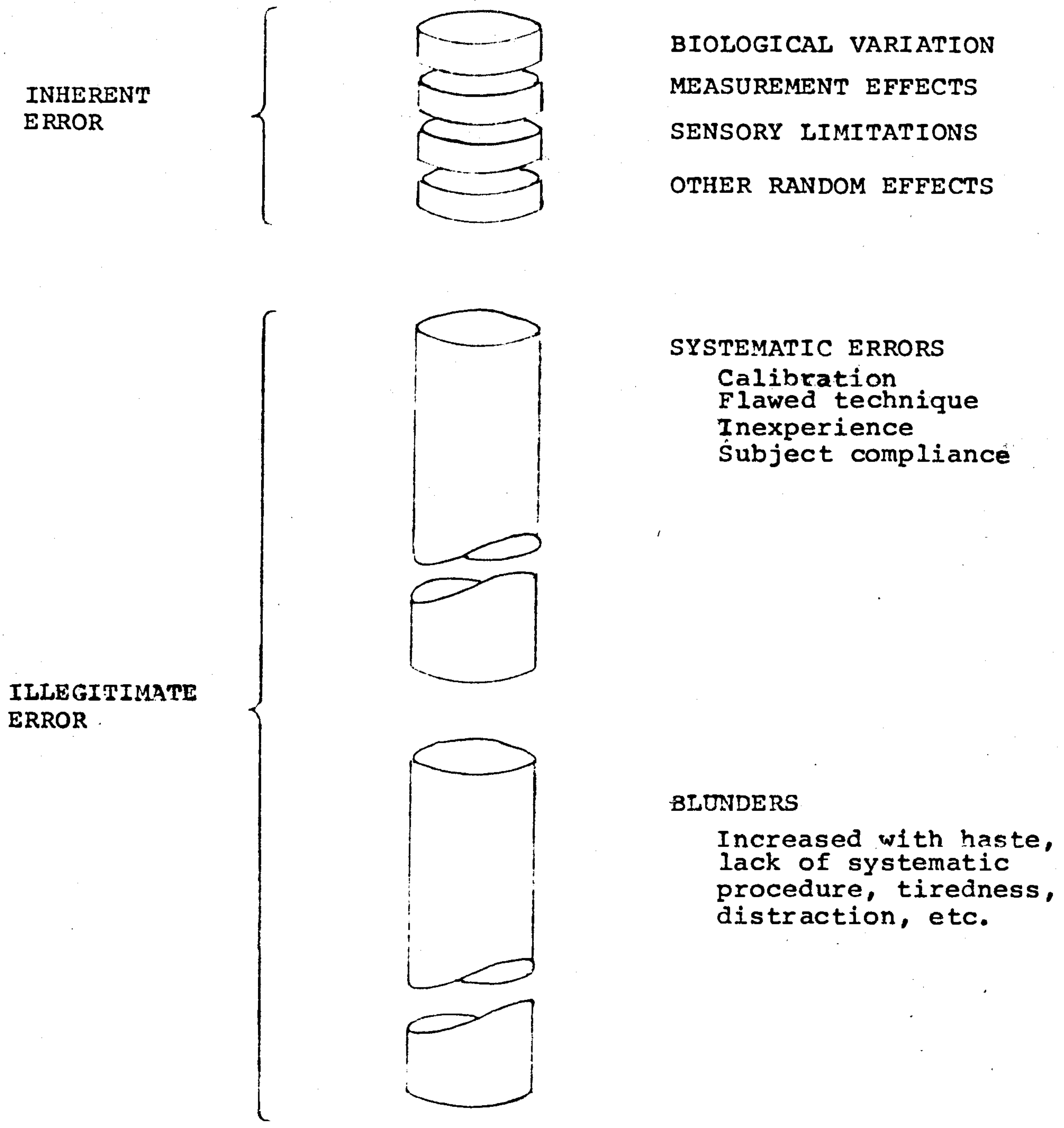
## 1.4. Sources of Error

In designing a computer based, interactive, error monitoring system of this type, it is necessary to appreciate the nature and sources of error likely to be present in the particular data assembly. If errors are to be eliminated or at least reduced during the collection of data, then it is important to know where they originate; i.e. at what point during the sequence of events, from observation to the production of the final data record, they are introduced.

There is great diversity in the mechanics of data collection, from very simple to extremely complex, from predominantly subjective to predominantly objective, from manual to automatic, as well as combinations of all of these. Different conditions will no doubt lead to different sources of error, but these sources can be classified into a number of common categories.

As illustrated in figure 1-1, errors in anthropometric data assembly may arise from many sources:

FIGURE 1-1



SOURCES OF ERROR

### 1.4.1. Inherent Error

Errors that can be identified and quantified but cannot be eliminated are classified as INHERENT error and fall under four headings.

#### Biological Variation

The first of these subgroups is concerned with the fact that humans are in dynamic equilibrium with their environment. For example, it is well known that subjects are taller and lighter in the morning than in the afternoon. In order to control for diurnal variation in these variables it is necessary to specify that the subjects be measured nude, in the morning, eight hours post absorptive after voiding. Stretch stature as specified by Weiner and Lourie(1977) and Ross and Wilson (1974) is used to overcome or minimize diurnal variation in height caused by compression of tissues due to gravitational stress throughout the day. Similar techniques of standardization of measurement ensure minimum influence of such biological variation.

#### Measurement Effects

Some errors may be introduced by the measurement process itself. An example in the anthropometric protocol is in the measurement of skinfolds where the application of the calipers may cause tissue compression which affect subsequent measures. These errors can be quantified (to some extent) and controlled but not eliminated (Ward, 1980).

## Sensory Limitations

The nature of the instruments and the physical charactersitics of the investigator are two factors which may produce errors in collected data. For example, with the best mechanical scaling and vernier calipers, the closest demarcation lines that the human eye can discriminate at 25 cm distance, under ideal lighting conditions, is about 0.022 mm (Padgham, 1965). The nature of the instrument and visual acuity and manipulative skill of the investigator are sources of inherent error which may be controlled by providing suitable working conditions with respect to lighting, noise levels, temperature and other factors that influence sensory input.

## Other Random Error

There are other sources of random error which cannot be classified under the previous three categories. Judgement errors, for example, occur when the observer is required to estimate some quantity such as time or distance. These estimates may vary from time to time. Errors arising from fluctuation in the environmental conditions, such as temperature and humidity, and small disturbing effects, such as vibrations or electro-magnetic activity are also possible. Beers (1957) also included definition errors in this list. Errors from this source may arise if the quantity being measured is not precisely defined. These are sources of random error that can be minimized but not eliminated. Beers (1957) gave an example of minimizing random error while using a steel measuring tape. "Some of the random error of this tape may be due to expansion and contraction of the tape with fluctuations of temperature. By noting the temperature at the time of each measurement and ascertaining the coefficient of linear expansion of the tape, the individual values may be compensated for this effect". Although these sources of random error cannot be eliminated their effect on subsequent analysis will decrease as the sample size increases.

### 1.4.2. Systematic Error

Statistical manipulation such as increasing sample size cannot reduce the effects of systematic error. Hence it is important to ensure that measurements are performed using well maintained instruments which have been recently calibrated. It is also important that these instruments are manipulated exactly as specified in the protocol with, in the case of anthropometry, application to designated landmarks.

When a number of the same obtained variables are outside tolerance levels, systematic error may be suspected. Common systematic errors in anthropometry include the following:

## Landmark Errors.

Landmark errors are errors which result from taking a measurement from an inappropriate location or at an inappropriate time. The phenomenon which is intended to be measured is missed and data from some other source is collected. Errors that can be termed 'landmark errors' may occur when the phenomenon being measured is not directly accessible. For example, some anthropometry requires the measurement of bone lengths, but the bones of living animals are covered in flesh and are not directly accessible. A landmark error can occur if the flesh covering the area to be measured causes errors in locating the ends of the bone.

If an error of this class is committed then the data collected are related to a phenomenon which is different from that which it was intended to observe. The effect of landmark errors can range from extremely small and relatively insignificant, such as a slight error in locating the end of a long bone, to completely erroneous measures such as measuring the wrong bone. Many of these errors can be avoided by precise techniques and thoughtful analysis of the underlying theories and assumptions.

## Errors Arising From Instrumentation

A second source of systematic error originates with the use of measuring devices such as scales, calipers, and meters. Although these devices are usually very precise and reliable, their correctness may be taken for granted. Some instruments require periodic calibration to verify accuracy and to adjust for varying environmental conditions. An instrument is calibrated by using it to measure a known quantity and if this measurement and the known value are different then an adjustment must be made. Instruments which are affected by such things as temperature changes and movement may require frequent calibration.

It may seem extreme to suggest that even simple devices such as measuring tapes need calibration checks, but a recent incident in the anthropometry laboratory at SFU illustrates the value of such checks. A measuring tape had been fastened to a wall to allow a subject's standing height to be measured. One of the researchers, during a discussion of technique, placed a hand held measuring tape next to the wall mounted one, and noticed a difference. A number of other tapes and measuring devices were then used to check the wall mounted tape, and all of them showed the same discrepancy. The wall tape was out of calibration by about 1 cm every 100 cm, which was probably a manufacturing error, and the tape had to be discarded. The calibration of this simple device was taken for granted.

Once an instrument is properly calibrated, errors may still be introduced in a number of ways. First the dial, meter or whatever output the device provides may be misread. Some common errors include reading the wrong scale when multiple scales are provided, (i.e. inches and centimeters), misreading a vernier scale and misjudging the reading in situations where there is always some fluctuation such as the oscillations of a balance beam scale.

Another source of error related to instrumentation involves technique. How the instrument is used is extremely important. Location, orientation and timing may be important considerations. A skinfold caliper that is not released completely when on the site, a weight scale which is not level or a thermometer which is not left long enough in the environment to be measured, are all examples of poor technique. Training and thorough understanding of the underlying principles of the instrument generally ensures that technique errors are minimized.

## 1.4.3. Blunders

Blunders "... are errors caused by outright mistakes in reading instruments, adjusting conditions of the experiment, or performing calculations" (Beers, 1957). They may also be introduced during the process of recording the data. Personal communication with a number of expert anthropometrists (Drinkwater, 1982; Ross, 1982; Ward, 1982b) revealed that blunders occur during the data collection process, but the frequency of occurrence is not known. These anthropometrists agree, however, that the blunders occur too frequently and steps should be taken to minimize their effect on data assembly. A literature search (i.e. DIALOG on-line information retrieval service) failed to locate any reports of research in this area. Some sources of blunders are discussed below.

## Transmission Errors

The step between measurement and recording is very dependent on human procedures and therefore has the potential to introduce blunders. Typically, at this step, the instrument is read and the reading is then recorded by the operator. It is this reading - recording process which may be marred by human error. Although, in some cases, recent technologies have been able to provide a direct electronic interface between measurement instrument and recording medium (Jones and Marshall, 1979), it is still more commonly a human interface.

The steps involved in the measurement and transmission of anthropometric data are as follows.

1. Measurer reads instrument.
2. Measurer vocalizes reading for recorder.
3. Recorder listens to measurer.
4. Recorder writes information.

In situations where the measurer is also the recorder then only steps one and four are carried out. Each of these steps will be discussed in terms of possibilities for error.

## Measurer Reads Instrument

The problems of reading instruments was briefly mentioned earlier as a source of random error but, it may also be the case that an error is introduced after a correct reading. One such error is to misplace a decimal point, as in reading the value 15.9 from a scale and assuming it is 1.59. A second common error is 'digit reversal' which seems to occur suprisingly often. An example of digit reversal is reading 15.9 and recording 19.5.

## Measurer Vocalizes Reading for Recorder.

It is possible that misplaced decimal points and digit reversals will happen during this step. Any unfamiliarity with language can also lead to errors by using wrong words or bad pronunciation.

## Recorder Listens to Measurer

Language may also introduce errors at this step. The recorder must correctly translate the vocalized information into written information. Familiarity with language and pronunciation will affect this translation. A good example of confusion arising from pronunciation is in the transmission of numbers such as thirteen and thirty. Each of these could easily be confused for the other. Another factor which influences the precision of this step is the ambient noise level. Any attempt to pass information by voice is obviously affected by interference from noisy surroundings.

Recorder Writes Information

At this step the digit reversal may be a problem. Writing data in the wrong location on a form, (or in the correct location but on the wrong form) and bad character formation (i.e. 'ones' that look like 'sevens') are other examples of error at this step.

## 1.5. Errors Detectable by the Data Monitoring System

The Data Monitoring System has been designed to detect errors which are categorized as ILLEGITIMATE errors. These errors can be of any size as the broken cylinders in Figure 1-1 suggest. They should never be recorded in the final data assembly. Illegitimate errors include most systematic errors such as calibration errors, and all blunders.

Some INHERENT errors may also be detectable by the monitoring system, if they are large enough in relation to the data. For example, diurnal variation in height may be detectable if the variation is large and the predictions are based on a sample that has been carefully controlled for effects of daily height variation. The ability of the monitoring system to identify either INHERENT or ILLEGITIMATE errors will depend on the model which is used to predict the measured values.

# II. PREDICTION MODELS AND TOLERANCES

## 2.1. Prediction

In order to make a prediction of any phenomenon, it is necessary that some facts, information or even assumptions be known about this phenomenon. Humans are able to make accurate predictions based on procedures which are not entirely clear. Intuition, "gut feeling" and other explanations are often used to describe the basis for these predictions. Since this system's software is not able to operate in this mode, it requires that information and relationships be described in a clear, unambiguous, highly structured form that has a consistent set of rules. The language of mathematics fits this description and is used by this system to describe the data which is to be collected.

The model definition phase of the Data Monitoring System is based on the assumption that a mathematical model which is highly related to the phenomenon being measured, can be constructed. If this is true, the predictive value of this model will be high, and it will probably be able to distinguish between "good" and "bad" data with an acceptable degree of reliability. (See Goldstein, 1977 for a definition of 'good' and 'bad' data.) The real power of the Data Monitoring System is then dependent on the development of this model.

Before developing a model something must be known about the population which the model is intended to describe. In most cases this information can be gained by collecting an initiator sample which may act as a representative subset of the entire population to be modelled. Through regression analysis of the data from the initiator sample, a number of predictor equations can be developed to form the model. This model can then be expressed in terms of ordinary mathematical expression (including multi-character variable names). Figure 2-1 shows the syntax rules for expressions.

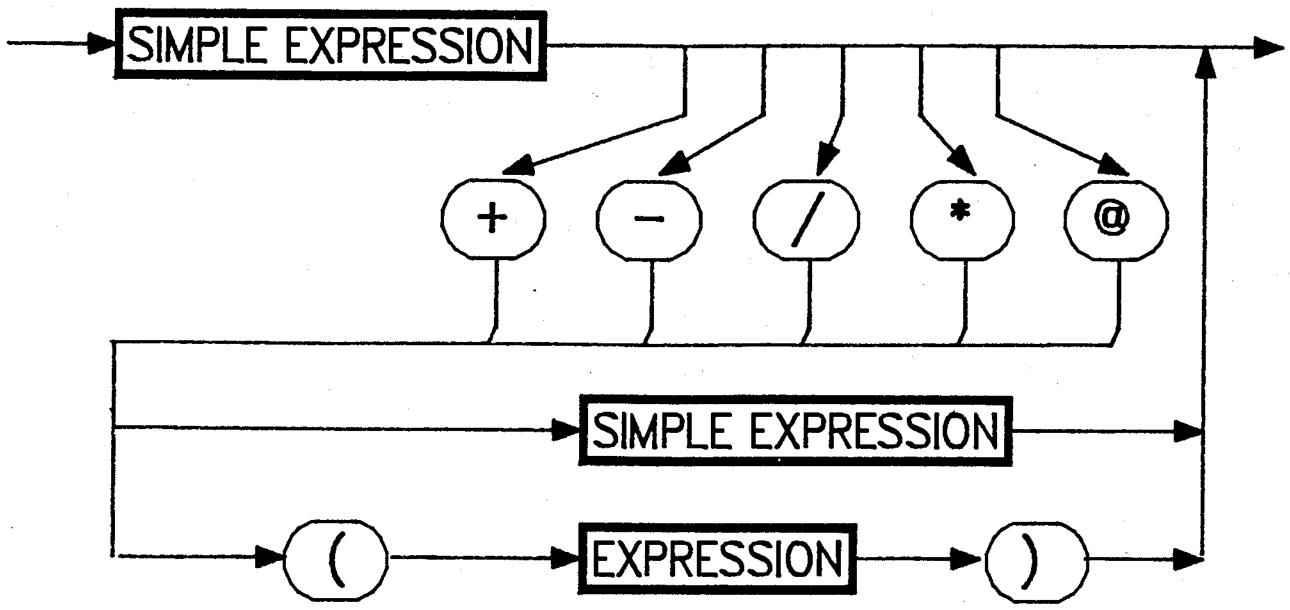
To read the syntax diagram start on the left side of the page and follow one of the horizontal lines from left to right. When rectangular shapes (non terminal symbols) are encountered they may be replaced with their definitions which appear elsewhere in the figure. A circular or round edged shape (terminal symbols) contains symbols which must appear in the expression exactly as they appear in the figure. By following the lines and arrows and substituting the definitions for the non terminal symbols, all possible legal syntactic combinations can be constructed. Any expression which cannot be constructed by using these rules is illegal.

The following example illustrates the definition of the predictive model:

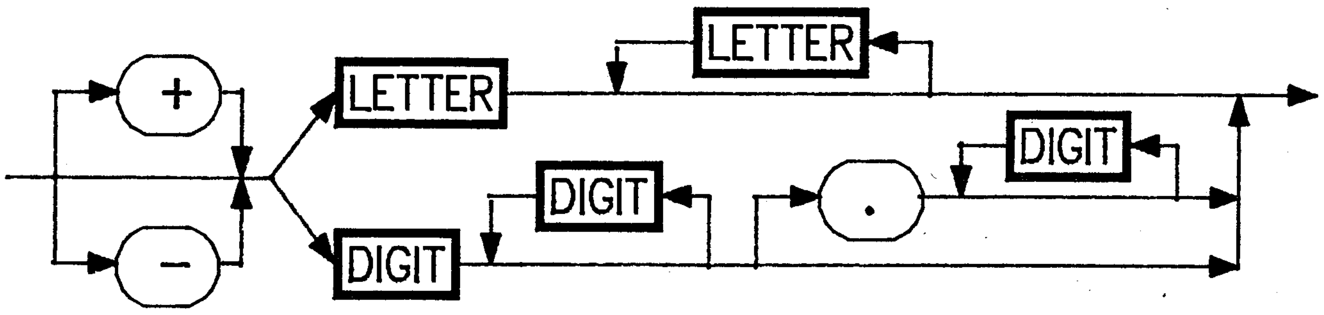
If it is known that the height (in centimeters) of a certain population of people is always twice their weight (in kilograms) then the following expressions can be written to

# MODEL DEFINITION SYNTAX

## EXPRESSION



## SIMPLE EXPRESSION



## VARIABLE ASSIGNMENT

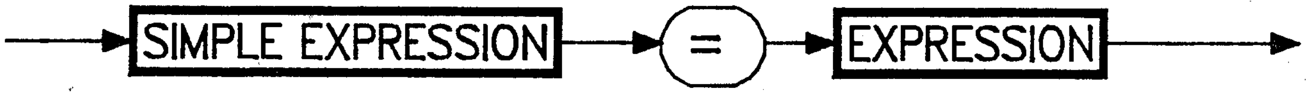


FIGURE 2–1

represent this known relationship:

WT = HT/2.0 (equation 1)

HT = WT*2.0 (equation 2)

where:

WT is a variable which represents WEIGHT.

HT is a variable which represents HEIGHT.

* is a diadic operator for multiplication.

/ is a diadic operator for division.

These relationships constitute a model of the population (at least with respect to these two variables) and such a model can be used to predict the measurement of any individual member of this population. If subject A is measured as 150 cm tall and 75 kg in weight then by substituting 150 for HT in equation 1 and 75 for WT in equation 2 and evaluating the expressions, the resulting predicted values are 75 and 150 for WT and HT respectively. Now, comparing the predicted values for HT (150) and WT (75) with actual values for HT (150) and WT (75) we find no difference. Had the collected data values indeed been 75 and 150 for WT and HT respectively, but an erroneous values of 57 (digit reversal) recorded for WT then the predicted value would be 75 (150/2.0) for WT and 114 (57*2.0) for HT. The predicted values for WT differs from the actual value which shows that the new data is not conforming to the relationship described in the model. This is an indication to the investigator that something is wrong and steps may be taken to correct or at least rationalize the discrepancy.

There is a possibility of recording a second erroneous value which will compensate for the first, and cause what will appear to be conformity with the model. For example, if the true values were 75 for WT and 150 for HT but as a result of errors WT was recorded as 57 and HT as 114 then no error would be detected. This is an unlikely coincidence and probability decreases as the model becomes more complex.

It is unlikey that any model describing human data will ever behave as precisely as the example given above. If a model is not perfect then predictions derived from it will not be precise. Therefore slight deviations of these predictions from the recorded values should not be considered as error. The size of the acceptable deviation is called the tolerance. A high tolerance means that the model is very tolerant of deviations from its predicted values and therefore describes a large range over which the recorded values may differ from the predicted value without being classified as an error. As confidence in the predictive power of the model increases the tolerance may be decreased. As the tolerance approaches zero the model becomes more and more sensitive to error. In our example above, the tolerance was (assumed to be) zero which means the model was very sensitive to error in that any deviation from the predicted value would be classified as an error.

A more realistic example then would be to add a tolerance to the prediction since it is unlikely that the HT and WT ratio in every subject is exactly 2:1. For example, by setting the tolerance for HT prediction at 2.0, a recorded value for HT that is within 2cm on either side of the predicted value for HT will not be identified as a possible error. All larger discrepancies will be so identified.

The tolerance, then, can be set extremely high which will increase the possibility of allowing errors (large discrepancies) to go undetected or very low which means that a larger percentage of accurate data may be identified as an error. Below are some guidelines for determining the level of tolerance for anthropometric variables. When a recorded value is equal to the predicted value  the tolerance then the recorded value is classified as correct.

## 2.2. Measurement Tolerance

Measurement tolerance is the range which one establishes for accepting measures without further scrutiny. The limits are arbitrarily determined usually with some justification based on an understanding of the variation in the data. For example, the tolerance must exceed the limits imposed by biological variation, measurement effects, sensory limitations, and other random events encountered by investigators under ideal conditions. The magnitude of these inherent errors for anthropometric items in this thesis have been derived by Borms, Hebbelinck, Carter, Ross, and Larivière 1979 as training standards used in the Montreal Olympic Games Anthropological

Project (MOGAP) as summarized in table 2-1.

In addition to inherent error which is present in replicated measures under ideal conditions, random error may also be introduced by imperfect regressions. Unless the sample used to derive the regression equations shows perfect relationships, there will be some variation in the predictions from these equations. This variation is quantified by the Standard Error of Estimate which is calculated by the regression analysis procedure. If the predictions from the regression equations are not perfect then some consideration of this must be included in the setting of the tolerances. As a rule of thumb, values outside the inherent error limit plus two standard errors of estimate (from the regression equation) may be used as a screening level for monitoring errors during data collection. Figure 2-2 shows all of the previously discussed sources of error with PREDICTION ERROR added as the new category of random error.

Illegitimate errors, either systematic or blunders, can be of any magnitude. Only when they exceed tolerance will they be flagged for subsequent re-measurement.

TABLE 2-1

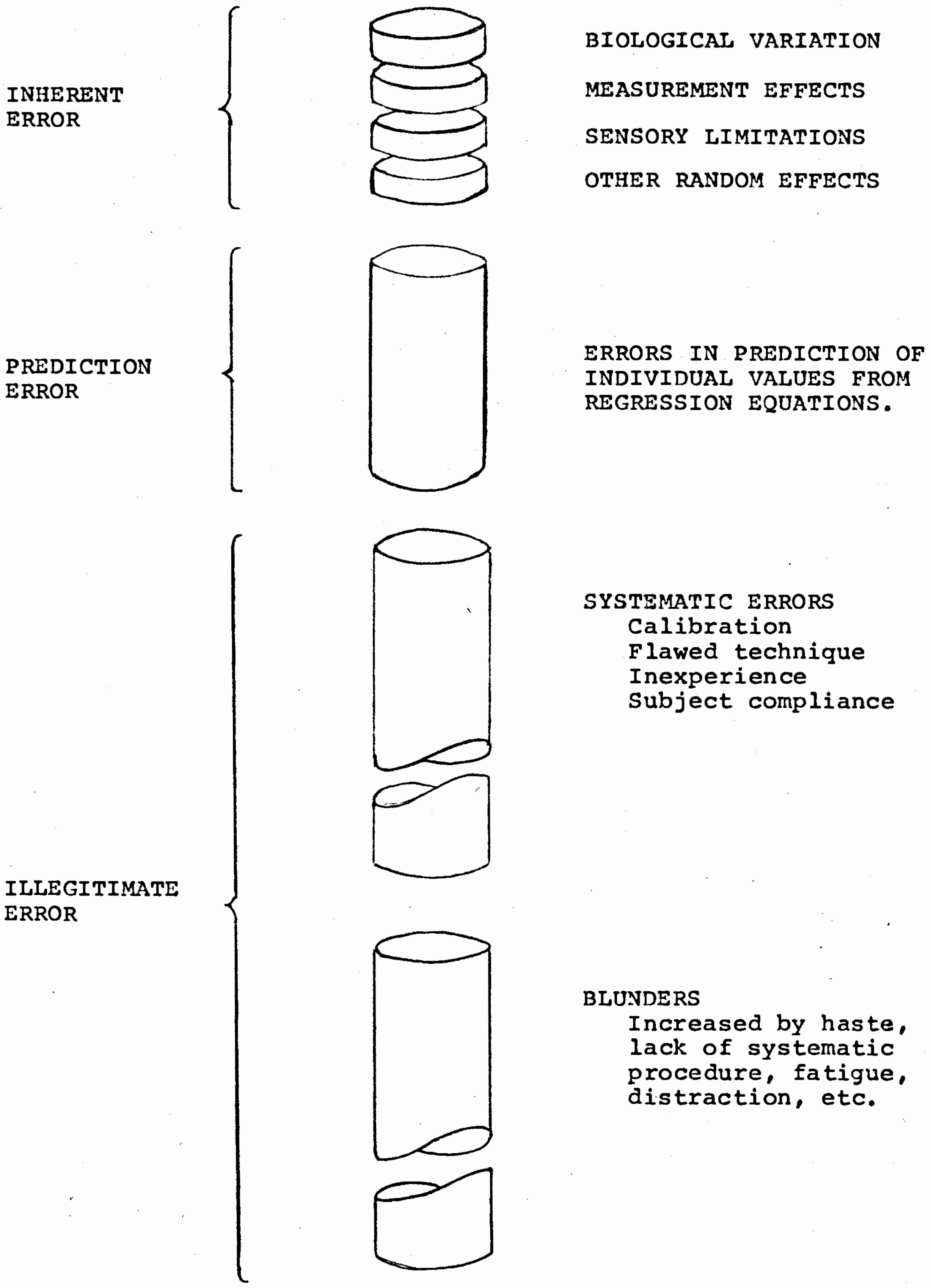
ANTHROPOMETRIC MEASUREMENT TOLERANCES

| | |
|---|---|
| Body Weight | .5 KG |
| Height | 3 mm |
| Acromial Height | 2 mm |
| Radial Height | 2 mm |
| Stylion Height | 2 mm |
| Dactylion Height | 2 mm |
| Trochanter Height | 2 mm |
| Spinal Height | 2 mm |
| Malleolar Height | 2 mm |
| Sitting Height | 2 mm |
| Tibial Height | 1-2 mm |
| Bicondylar Humerus Width | 1 mm |
| Bicondylar Femur Width | 1 mm |
| Biacromial Diameter | 1-2 mm |
| Thorax Diameter Transv. | 2-3 mm |
| Thorax Diameter Sagit. | 1-2 mm |
| Bi-iliocristal Diameter | 1-2 mm |
| Wrist Width | 1-2 mm |
| Neck Girth | 2 mm |
| Upper Arm Extended | 2 mm |
| Upper Arm Flexed | 2 mm |
| Chest Girth Inspiration | 1-2 % |
| + Chest Girth Expiration | 1-2 % |
| Waist Girth | 2-3 % |
| Abdomen Girth | 1 mm |
| Thigh Girth | 1 mm |
| Calf Girth | 1 mm |
| + Skinfolds | 5 % |
| * Forarm Girth | 2 mm |
| * Wrist Girth | 1 mm |
| * Ankle Girth | 1 mm |
| * Foot Girth | 1 mm |
| * Head Girth | 1 mm |

Adapted from Borms et.al, (1977).

+ Percentage of the sample mean was used.
* Included by author. (not published by Borms et.al)

FIGURE 2-2



Sources of error with prediction error

## III. THE ERROR CONTROL SYSTEM: A System overview

The Data Monitoring System is an attempt to provide a solution to the problem of encountering errors when it is too late to correct them by re-measurement. This problem is particularly evident with the collection of human data since the passage of only a few hours may result in changes which would invalidate any re-collected data. If the need for re-measurement is known within minutes rather than hours or days then data may be re-collected without the risk of the phenomenon being seriously changed during the time between measurements. The Data Monitoring System scans data as it is collected (or very soon after) and predicts each data value according to some previously defined prediction scheme or model. If the collected data do not agree with the predicted data then the experimenter may be notified by the system. Having been informed of the discrepancy he may either re-enter the relevant value or values following re-measurement or force the systen to accept the original value. Since only out of range values, which are not necessarily errors, are noticed by this data monitor, it is up to the investigator to decide if the indicated values are erroneous or are legitimate extreme values. The system acts as a very diligent and critical observer.

The Data Monitoring System has been designed to provide three basic phases of operation: The DEFINITION phase where each of the variables to be measured and their tolerances are described to the system; the COLLECTION phase where data are collected and entered into the system; and, the VERIFICATION phase where collected data from each of the variables are compared to the values predicted by the rules established in the definition part. The investigator is notified of discrepancies which are outside the previously established tolerance ranges, giving an opportunity to correct errors immediately.

The definition process begins by collecting the initiator sample. To reduce the random error, repeated measures (three or more) of each variable should be taken and then the set of median values used to develop the model of the population using multiple regression analysis. The model consists of a number of regression formulae, one for each variable, which show how any given item varies in relation to one or more of the others. These regression lines make up the prediction formulae. Figure 3-1 illustrates the conceptual organization of the system.

The Data Monitoring System is interactive and command driven from a hierarchy of command levels. This means that the experimenter is able to maintain an interactive dialogue with the system during the data monitoring process. This dialogue permits changing the model or any of its parameters, loading different models into the system, using the built-in calculator, modifying the set of collected data values, and a number of other operations provided by the system (see appendix C for a complete list of the available commands). Having control of the system at any time allows the experimenter to react to

THE DATA MONITORING SYSTEM ORGANIZATION

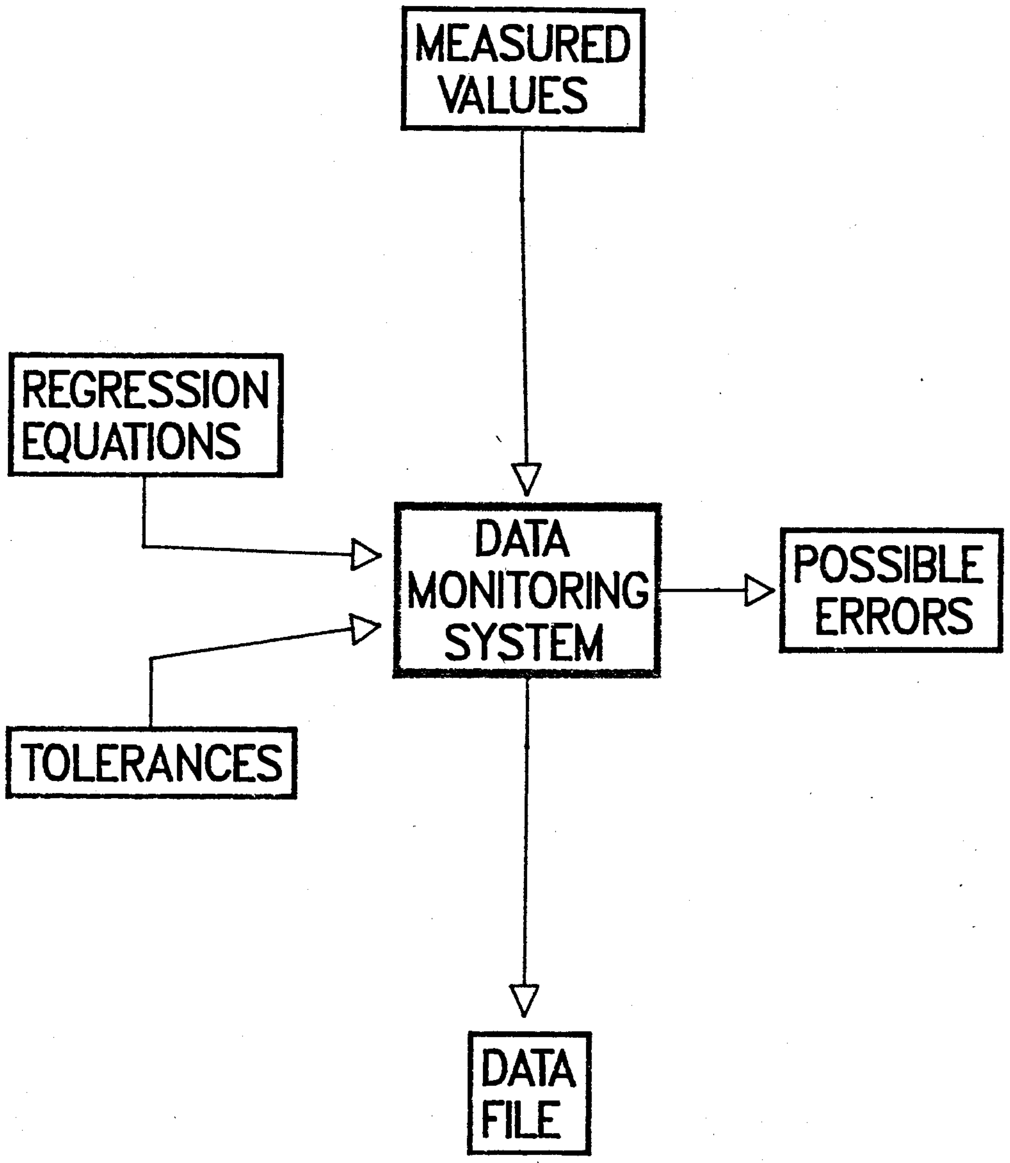


FIGURE 3–1

situations as they arise in the data collection process.

## 3.1. Physical Constraints on the System

It is not unusual that data are collected in locations specifically and temporarily set up for the collection process. These locations rarely offer the facilities of a well equipped laboratory. If a computer based error control system is to be useful in the field, it is necessary that it be self contained and not require the computer facilities of a laboratory or research institution. Both the hardware and the software must be reasonably portable. The hardware often may be carried from place to place and the software may be required to operate on a number of different processors.

In light of these constraints, a small widely available micro-computer was chosen as the development processor and one of the two most common micro-computer operating systems was chosen as the development system.

## 3.2. Hardware and software requirements

1. The minimum requirements of the Data Monitoring System:
   a. UCSD pascal - the operating system.
   b. A computer with sufficient memory to run UCSD pascal (eg. APPLE II with 64k bytes memory).
   c. External storage device (eg. 5 1/4" floppy disk drive).

d. Keyboard.

e. CRT display with a minimum of 80 characters per line.

2. Optional devices:

a. Printer

b. Second external storage device (disk drive).

## 3.3. Screen Format

The display screen is divided into three fields; the MESSAGE FIELD, the PROMPT FIELD and the DATA DISPLAY FIELD. (see figure 3-2) The message field is the top line of the screen. The PROMPT FIELD is the second and third lines of the screen and the DATA DISPLAY FIELD makes up the remainder of the screen.

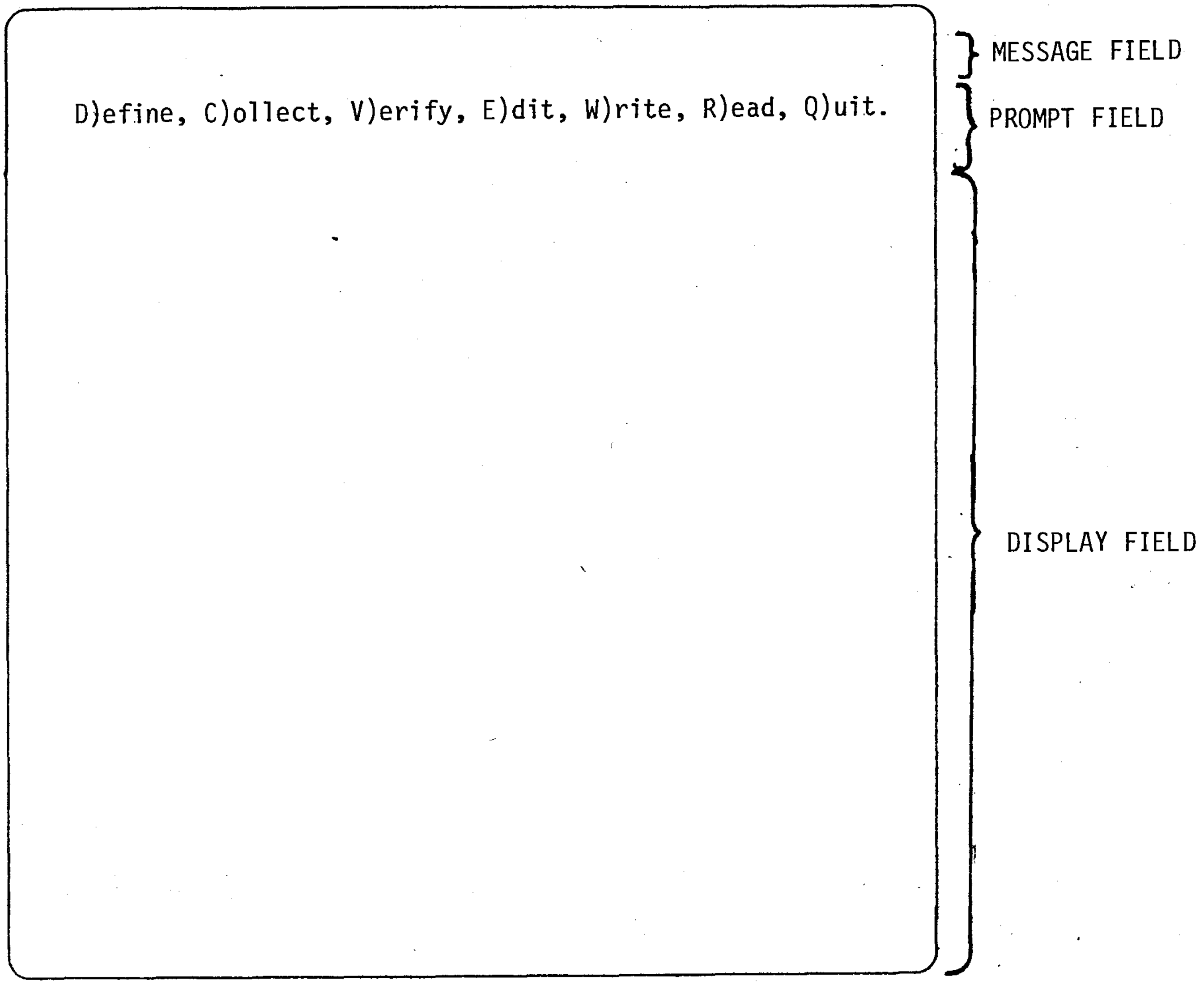
The screen is divided in this way to provide some consistency between levels of the system. Whenever a choice of commands is available the user is able to see the possibilities in the Prompt field which is always at the same location on the screen. Since this is true for any level of the system, the user may quickly become accustomed to looking at specific parts of the screen for needed information.

1. The Message field.

When it is necessary to provide the user with information concerning the current state of the system, a message is written on the top line of the screen. All error

FIGURE 3-2

SCREEN FORMAT FOR THE DATA MONITORING SYSTEM



messages from the system are written in this message field.

2. The Prompt field.

At any given time the system is ready to respond to one of a limited set of commands. Only those commands which are listed on the prompt field (with a few exceptions) are recognized. The contents of the prompt field, and therefore the commands that are available, depend on which portion of the system is currently in control (see figure 3-3).

The available commands are listed in the Prompt field and separated by commas. Figure 3-2 shows an example of a command prompt line in the Prompt field. The letters of the command to the left of the parenthesis appear on one of the keyboard keys and it is that key which must be struck in order to select the command. Striking keys which do not correspond to commands has no effect.

3. Data display field.

This field constitutes the entire screen except for the Message line (line 1) and the Prompt line (line 2). All data which the system is asked to write on the screen is displayed here. As the data display field fills the screen is scrolled. When the display is finished the message field and the prompt line are re-written (the message field is cleared if there is no message and the list of available commands is written on the prompt line).

The data display field is also used to collect information from the user. The model and collected data are

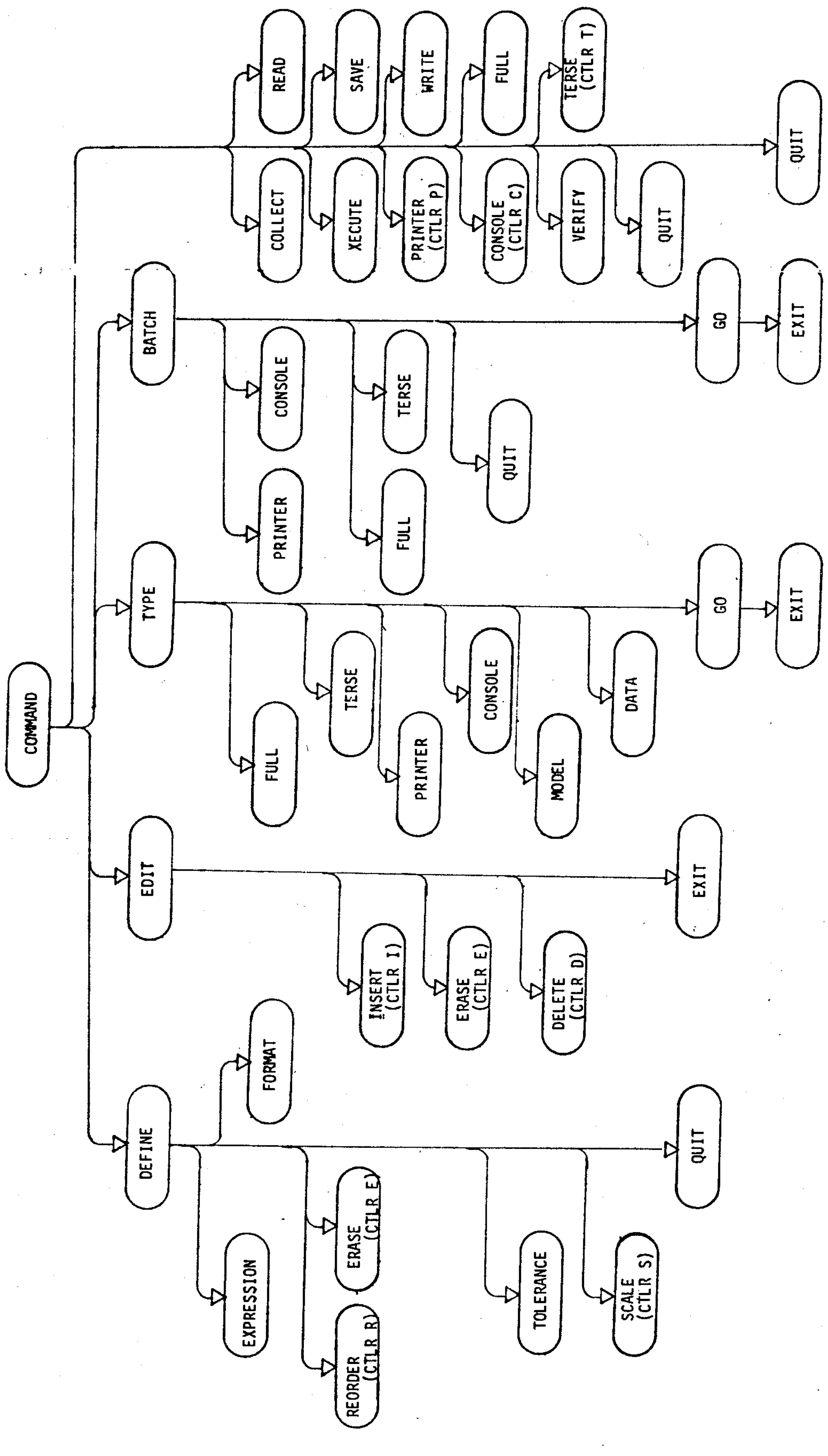


FIGURE 3-3

COMMAND TREE

entered in this area and the editor uses it to allow visual editing of the model.

## 3.3. System Functions

Figure 3-3 illustrates the system hierarchy and organization. Appendix C lists all of the possible commands, the level at which they are recognized and what function they perform. This section gives a general description of the functions of each command at the top level of the system.

### 3.3.1. Commands available from the top level prompt line.

DEFINE.

By striking the "D" key the system becomes ready to perform functions related to the definition of the predictive model used for error detection. The system responds with a prompt which gives the choice of entering expressions, tolerances or formats. The expressions are the predictor equations for each variable; the tolerances indicate how sensitive each prediction is to error; and the formats indicate where each variable can be found on a data record when this record is stored on disk.

COLLECT.

The collect command causes the system to step through each of the variables defined in the model and prompts (by displaying the variable name followed by a '?') for an observed value to be entered. In this way, a set of data values for an individual is gathered and stored in the system. When each of the variables has received a value the top level prompt is displayed.

The COLLECT routine prompts for data in the same order as the variables are currently defined in the model (the order may be changed by the DEFINE process). After each prompt, the measured value for that variable is entered by typing the number and then the return key. If the return key is hit without first typing the number then no value for that variable is entered. The old value, if there is one, is retained, and the next prompt is displayed.

VERIFY.

By typing "V", for Verify, the system begins to scrutinize the data and look for possible errors. The value of each variable is predicted by evaluating its predictor expression. This expression was previously given to the system during the DEFINE process and is part of the model. The predicted value is then compared to the actual value given during the COLLECT process, and the actual value, predicted value and tolerance are printed. If the actual value differs from the predicted value by more than the tolerance then the value of

$$(\text{Predicted} - \text{actual})/ \text{ tolerance}$$

is printed at the end of the line. This value shows not only that there is a discrepancy but also the size of the discrepancy and its direction.

When each of the variables has been scanned for errors, control is returned to the top level so that any of the top level commands are again accessible.

<u>EDIT.</u>

The EDIT command allows editing of the symbolic expressions in the model. Immediately after EDIT has been selected the system prompts for the number of the item to be edited. When a number is given (if there is an expression in the model definition which corresponds to that number) then that expression is shown in the data display area and is ready for editing.

The editor is a single line visual editor. The cursor can be positioned and any character can be replaced, inserted or deleted. When the return key is struck the contents of the screen are saved as part of the model. If the variable name on the left side of the equal sign is changed then the old variable and its predictor expression are deleted and the new one is inserted as the last equation in the model.

<u>WRITE.</u>

Once the data for a subject have been collected and verified and are deemed by the investigator to be error-free then they can be saved on disk using the WRITE command . The current value of all variables is saved according to the format given in the DEFINE process. For each variable that is "out of range" a counter is incremented and that variable is considered a "Force in" value.

<u>SAVE.</u>

The current state of the system may be saved on disk. This means that all model information as well as the current values for the variables can be stored for use at a later time. The SAVE command is useful for storing models which have been "fine tuned" for the collection of data from different samples. These models do not have to be re-defined. They can be read directly from disk using the READ command.

<u>READ.</u>

The READ command permits a file containing a complete model definition and values for variables to be read from the disk. Once a file name has been given to the system, the model is read from disk and the top level command prompt is displayed. Only files which have been written using the SAVE command can be read with the READ command.

<u>QUIT.</u>

This command causes the Data Monitoring System to be unloaded and control is returned to the UCSD operating system.

### 3.3.2. Hidden Commands at the Top Level

There are a number of commands available at the top level which are not shown on the prompt line. These commands are not often used but are made available to increase the versatility of the system.

<u>BATCH.</u>

This command provides a facility for entering data from a disk file rather than from the keyboard. The system prompts for the name of a disk file which contains the data and then this file is read according to the format specifications from the DEFINE procedure. After a complete set of data has been read (ie. one value for each of the defined variables) the VERIFY command is automatically initiated. The output from this VERIFY can be directed to the printer or the video display and can be set to FULL or TERSE format. In TERSE format only the "out of range" variables and their associated values is printed.

The BATCH process continues to run until END-OF-FILE on the data file or until it is interrupted by striking the escape key.

<u>EXECUTE.</u>

This command is invoked by striking the 'X' key. With the EXECUTE command an expression containing variables, constants and operators can be entered. As soon as the expression is entered it is evaluated and the result is printed on the screen. The EXECUTE command can be used as a calculator which accepts not only numbers but variable names in a complex expression. Each variable used in the expression must be defined in the current model and must have a current value.

<u>GET.</u>

Striking 'G' permits reading a set of equations and tolerances from disk. Each input line contains a complete equation (see Figure 2-1 for Syntax Rules) followed by a semicolon, followed by the tolerance value and finally another semicolon. The GET command is useful for introducing a set of equations and tolerances that have been generated by a computer as a result of some statistical analysis (ie. equation generated from a regression analysis).

TYPE

It is often desirable to examine the current content of the data or model stored in the system. By typing 'T' the investigator is given the opportunity to list the DATA or MODEL on the printer or screen.

## 3.4. Typical Examples

The following section briefly outlines the steps necessary to use the system. The first example shows the hand entry facilities while the second example shows automatic entries. The third is an example of how the data and model can be changed.

### 3.4.1. Entering model and data by hand.

Once the regression equations are constructed they are entered into the Data Monitoring System. The following is a step by step example of monitoring the data collection. First the model is definded, then the data for one subject is monitored.

When the top level prompt is displayed, "D" for DEFINE is typed. The system responds with the Definer prompt

"E)expression, F)ormat, T)olerance, Q)uit."

Since the expressions are to be entered, "E" is typed and

"Enter expression"

is displayed.

A complete equation is entered and the return key is hit. At this point, the system processes the equation and the variable name on the left side of the equal sign is entered into the system symbol table. The independent variables, which are mentioned on the right side of the equations, are assumed to be already defined or about to be defined. Step 2 is repeated once for each prediction equation up to the limit of the system. A carriage return immediately after the ENTER EXPRESSION prompt causes the system to return to the Definer Prompt.

The tolerance for each variable is entered in step 3. After typing 'T' the system displays the first variable name defined in the previous step and waits for the tolerance for that variable to be entered. When the tolerance value and a "RETURN" are typed, the name of the next variable is displayed. This process continues until the last variable receives a value for the tolerance. The Definer Prompt is then displayed.

The last step for defining a model is to specify the format for writing data on disk. After typing 'F' for Format in response to the Definer Prompt the system behaves in a way which is similar to Tolerance in step 3. After each variable is displayed the system waits for input. In this case the system expects 3 numbers, separated by blanks, before the return key is hit. The 3 numbers represent the format and have the following meaning:

1. The column on the record which is the beginning of the field for this variable

2. The length of the field

3. The number of columns for the fractional part of the number.

For example, the format "12 5 3" indicates that the field for this data begins on column 12, has a length of 5 columns and three places are used for the fractions part. The largest number that can be written according to such a format is 99999 and the smallest is -9999 (notice that the minus sign takes one column).

When the last variable has received a format specification the Define Prompt is displayed again.

Once the equations, tolerances and formats have been entered the Define process is complete. By striking the 'Q' key the top level prompt returns. It is a good idea at this time to SAVE the model. To do this, an 'S' is typed at the top level and the system prompts for a file name. After the name is typed, the model is saved and the top level prompt is displayed again.

The next step is to collect data from a subject who is part of the population which the model was designed to predict. Typing 'C' prepares the system to accept observed data by printing the name of the first variable identified in the DEFINE process. As each item is measured and the observed value entered, the system prompts for the next. When all items have been entered the control is passed back to the top level.

The system now contains a complete definition of the model and a complete set of data for a newly measured subject. The next step is to type 'V' and begin the VERIFICATION process which predicts each of the variables by evaluating the expressions associated with that variable. As each prediction is made the observed, predicted and tolerance values are displayed and then an analysis is made of the agreement between predicted and observed. If the disagreement is greater than the tolerance the final value printed on the line is the difference divided by the tolerance; that is the number of tolerances by which they disagree (see VERIFY).

## 3.4.2. Entering Model and Data Automatically

This example shows how some of the automatic features may be used. The set of equations and the data from measured individuals can both be entered automatically rather than by hand. The steps, which are similar to the steps in the previous example, are as follows:

a. Develop the regression equation for the model and store on some machine readable medium (i.e. disk). To make the next step useful the equation should be generated automatically by a computer program which has as its input the raw data from which the regression equations are to be derived.

b. Type 'G' for 'GET EQUATIONS' when the top level prompt is on the screen. The system then asks for the name of the file containing the equations. These equations are read and stored as the current model. The top level prompt appears again when this process has finished.

c. If Formats are stored in the system before the 'G' command they remain unchanged after the equations have been read. If they are still appropriate for the new set of equations then no further information is needed for the next step. If Formats are not appropriate or if they have not been defined then they must be entered before proceeding. Type 'D' for DEFINE and then 'F' for FORMATS and then enter the format. Type 'Q' for 'QUIT' after the last format has been entered.

d. In response to the top level prompt, typing a 'B' initiates the BATCH process. Data that is to be verified must be on disk and in the format as specified in the Define phase. The Batch procedure asks for a file name and for the destination and type of output.

## 3.4.3. Changing the Model

This example shows how the model can be manipulated by the functions provided by the Define process. The model can be re-ordered, a specific prediction equation can be changed or the tolerances can be manipulated.

a. Reordering the Equation

The model can be re-ordered by typing 'D' to get into the defining procedure and 'E' to begin definition of the expressions. When the

"enter expressions"

prompt appears a "CTRL R" (which is typed by holding down the CTRL key and then typing 'R') enables the experimenter to move one of the equations to another location in the model. After typing CTRL R the system prompts with:

"Enter source <SPACE> destination."

By typing '8 2' and then hitting the return key, equation number 8 is moved to location 2. Equation 2 through 7 are moved to 3 through 8 respectively. The model is then reordered. Any subsequent command that requires stepping through the variables, such as COLLECT, uses this new order.

b. Scaling the Tolerances

If a "CTRL S" is typed as a response to the prompt for entering a TOLERANCE, the system allows global SCALING of the tolerances. After CTRL S the prompt

"Enter operator (+,-,*,/) followed by constant"

is displayed.

If '*2.0' is entered all tolerances are multiplied by 2.0, '-5.0' subtracts 5.0 from each and /3.0 divides each one by 3.0.

c. Editing an Individual Equation

The third method of changing the model is the editing of a predictor equation. This example shows how the editor which was described earlier, can be used to change an equation.

'E', for EDIT, is typed as a top level command, the system responds with

"Enter item number?"

"8" is entered and the system then displays the model's eighth equation:

```
DAHT=STHT*1.009-FOOT*0.856+HUM*1.666-TRCH*0.244+3.220
```

and the cursor is positioned at the first character after the equal sign.

The equation can now be changed by typing over what appears on the screen, deleting the character under the cursor by typing CNTR D (hold the CTRL key down and then strike D) or inserting a character by typing CTRL I. Moving the cursor by striking the right or left arrows has no effect on the equations. A CTRL E deletes all characters to the right of the cursor.

The return key causes the system to process the changed equation, store it as part of the model and return to the Definer prompt.

## IV. SYSTEM PERFORMANCE AND EVALUATION

This section is a demonstration of the Data Monitoring System's performance when it was applied in the collection of anthropometric data. The demonstration consisted of assembling data, building a model and using the Data Monitoring System, with this prediction model, to verify data for individual subjects.

The data used in this demonstration were collected during two sessions of an anthropometry course at Simon Fraser University. The subjects (students in the course) were each measured three times by an expert anthropometrist.

## 4.1. Building the Model and Defining it in the Monitoring System

A set of regression equations and their respective tolerances were assembled and used as a prediction model. The regression equation parameters were computed by the multiple regression analysis computer program provided by the Statistical Package for the Social Sciences (SPSS). All of the median values of the female data except those for four randomly selected subjects were used as raw data for the regression analysis. These data constitute the data for the 'initiator sample' as mentioned during the discussion of 'prediction' in chapter 2. The four randomly chosen subjects were used to simulate the measurement-recording process.

The regression analysis program was instructed to find the five best independent (or predictor) variables for each of the items in the data. (In this context "best" means that the first independent variable is the best single predictor, the second is the best of the remaining variables, etc. Appendix C lists all of the items and their corresponding variable names.) The output from this program lists each of the five variables along with their unstandardized regression coefficient, a constant value (the Y intercept of the regression line) and the standard error of estimate for the prediction equation. The equations shown in Table 4-1 were constructed from this information and have the following general format:

DV = IV1*B1 + IV2*B2 + IV3*B3 ...IVn*Bn + CONSTANT.

where

DV is the dependent variable

IVn is the nth independent variable

Bn is the B value for the nth independent variable.

These equations were then entered into the Data Monitoring System by using the DEFINE command. Next, the tolerances were established and entered. The tolerance for each variable was the value from Borms et al. (1977) as shown in Table 2-1 plus twice the standard error of estimate for the regression equations for that variable. The next part of the DEFINE process would normally be the entry of FORMAT specifications, but since the collection process was being simulated and there was no need to store the data, the formats are irrelevant and are not given.

## 4.2. Simulating a Measurement Session

For the purposes of this demonstration the median values were assumed to be the best representatives of the true values. It was also assumed that if the measurer was notified of a possible error and a re-measurement made, the extra care and attention would result in a value that was close to the true value. With these assumptions the re-measurement process was simulated by using the median value for the re-measured value. The medians for the four subjects are listed in Appendix D.

Data for the subjects that were randomly selected out of the sample were entered as if they were actually being measured. As a result of the repeated measures protocol, information on each of these four subjects consisted of three complete sets of measures. Each set was entered as if it were just being collected, giving twelve sets of data. The data were then scanned for discrepancies by using the VERIFY command. Once this process identified variables that may have contained errors, the median values for these variables were entered and the VERIFY command given again.

To be consistent during this demonstration the VERIFY process was initiated only twice, at most, for each set of data. Between these scans the value for each deviant variable was replaced by the median. In other situations it may be desirable to VERIFY more often or to enter the median values of some, but not all, the deviant variables. Some of these situations will be noted in the description of the measurement simulation.

The VERIFY command gives an option of FULL or TERSE output listing. All output shown in this chapter is in the TERSE format; which means that only the subject number and any "out of range" variables are listed.

## 4.3. Detection of Systematic Errors

Data for the first measurement occasion for the first subject is shown in Table 4-2 and the results of a series of VERIFY commands are shown in TABLE 4-3.

Table 4-3(a) lists the results of the VERIFY process for the first measurement occasion of the first randomly selected subject (#7). The variables CHG, WAG and HUM were found to deviate from the predicted value by more than the tolerance. WAG and HUM both deviated by only .013 of a tolerance while CHG showed larger discrepancy. Thus the COLLECT command was given permitting new values to be entered for some or all of the variables. The median values of 82.2, 68.3 and 5.93 for CHG, WAG and HUM respectively were entered and the VERIFY re-done. In this case since WAG and HUM were just slightly out of range it was sufficient to re-measure only CHG. This occurs if WAG and HUM have CHG as an independent variable in their predictor equation.

The results of the second VERIFY are shown in Table 4-3(b). Only CHG remained as an "out of range" value. Since its value was the median and assumed to be the "true" value it was accepted. The WRITE command was given allowing the current values to be stored on disk. CHG was considered to be a "FORCE-IN" value.

TABLE 4-1

| | TOLERANCE | EQUATION | |
|---|---|---|---|
| 1 | 0.000 | SUBJ | =SUBJ |
| 2 | 0.000 | SEX | =SEX |
| 3 | 3.582 | WT | =CHG*0.604+THG*0.738+HT*0.488+CAG*0.612-STHT*0.346-109.479 |
| 4 | 2.245 | HT | =ACHT*0.388+SPHT*0.775+SITHT*0.567-HUM*0.805+CAG*0.358+9.573 |
| 5 | 4.946 | TPSF | =THSF*0.330+ILSF*0.433+FEM*1.593-SPHT*0.188+MCSF*0.160+2.822 |
| 6 | 6.235 | SSSF | =WAG*0.716-STHT*0.450+MCSF*0.517-TRCH*0.894-CAG*0.402+28.397 |
| 7 | 4.818 | ILSF | =ABSF*0.404+TPSF*0.340-BIIL*0.783+FAG*1.078-WRG*1.664+18.942 |
| 8 | 7.838 | ABSF | =ILSF*1.415+BIIL*1.983-HUM*9.341-TIHT*1.847+ACHT*0.631+0.162 |
| 9 | 9.948 | THSF | =TPSF*1.376+DAHT*0.690-FAG*0.449-ILSF*0.273-SITHT*0.414+11.496 |
| 10 | 5.518 | MCSF | =BIIL*0.681+SSSF*0.210+TIHT*0.910+TPSF*0.331-HDG*0.543-22.056 |
| 11 | 2.691 | ACHT | =RAHT*0.842+SPHT*0.272+HUM*2.153-APCH*0.254+TRCH*0.134+10.039 |
| 12 | 1.724 | RAHT | =ACHT*0.311+STHT*0.585-THG*0.149+WRG*0.276+TIHT**0.269-0.085 |
| 13 | 1.609 | STHT | =DAHT*0.965+FOOT*0.887+TRCH*0.251-HUM*1.653+HDG*0.106-3.182 |
| 14 | 1.814 | TIHT | =SPHT*0.418-APCH*0.121+BIIL*0.409-CHG*0.059-FOOT*0.288+7.289 |
| 15 | 1.631 | DAHT | =STHT*0.928-FOOT*1.015=HUM*1.814-TRCH*0.215+HT*0.064-4.220 |
| 16 | 2.129 | SPHT | =HT*0.786-SITHT*0.380+APCH*0.354-ANG*1.038+FAG*0.613-4.303 |
| 17 | 1.410 | AGR | =AGF*0.986+WAG*0.035-BIAC*0.153-ANG*0.205+WT*0.043+4.242 |
| 18 | 1.290 | AGF | =AGR*0.881+ANG*0.161+BIAC*0.167-BIIL*0.186+HDG*0.143-7.928 |
| 19 | 1.344 | FAG | =AGF*0.332+HUM*2.496+FOOT*0.269+ANG*0.253-FEM*0.725-5.717 |
| 20 | 0.889 | WRG | =WT*0.010+ANG*0.251+FOOT*0.096+APCH*0.185+RAHT*0.031+1.215 |
| 21 | 2.814 | CHG | =WAG*0.292+WT*0.293+TRCH*0.846+AGP*0.466-CAG*0.493+30.971 |
| 22 | 5.128 | WAG | =CHG*1.330-STHT*0.272+CAG*0.464-AGF*0.683-BIAC*0.506-2.532 |
| 23 | 3.954 | THG | =WT*0.718-SPHT*0.268-WAG*0.239+HDG*0.490-SITHT*0.249+50.722 |
| 24 | 2.674 | CAG | =WT*0.284+ANG*0.696+FEM*0.861-SITHT*0.194-CHG*0.175+28.233 |

(TABLE 4-1 CONT.)

| | | |
|---|---|---|
| 25 | 1.414 | ANG =CAG*0.154+WRG*0.571-SPHT*0.160+SITHT*0.178+FAG*0.250-0.028 |
| 26 | 2.489 | BIAC =TRCH*0.330+FOOT*0.785-WRG*1.033+DAHT*0.062+CHG*0.084+13.664 |
| 27 | 2.020 | BIIL =FEM*1.315+TIHT*0.453+CHG*0.104-DAHT*0.136+HDG*0.282-19.447 |
| 28 | 1.806 | TRCH =CHG*0.426-APCH*0.165-ARG*0.297-SPHT*0.208+ACHT*0.141+0.035 |
| 29 | 1.156 | FOOT =HT*0.170+WRG*0.826-DAHT*0.199+BIAC*0.255-WAG*0.051-10.173 |
| 30 | 0.428 | HUM =FAG*0.130+ACHT*0.016+FEM*0.283-ANG*0.070-FOOT*0.057+1.115 |
| 31 | 0.679 | FEM =CAG*0.032+BIIL*0.140+HUM*0.723-SPHT*0.043+SITHT*0.041-0.108 |
| 32 | 2.430 | SITHT=STHT*0.239+ANG*0.384+HT*0.493-TIHT*0.963+HDG*0.287+5.133 |
| 33 | 2.631 | APCH =AGF*0.208+WRG*1.093-STHT*0.296+SPHT*0.344-TIHT*0.349+2.237 |
| 34 | 2.434 | HDG =BIIL*0.505-CHG*0.150+THG*0.276-FAG*0.462+SITHT*0.130+37.698 |

TABLE 4-2

|  | VARIABLE | VALUE |  | VARIABLE | VALUE |
|---|---|---|---|---|---|
| 1 | SUBJ | 7.000 | 18 | AGF | 27.400 |
| 2 | SEX | 2.000 | 19 | FAG | 23.900 |
| 3 | WT | 56.300 | 20 | WRG | 14.400 |
| 4 | HT | 167.100 | 21 | CHG | 82.100 |
| 5 | TPSF | 19.400 | 22 | WAG | 68.800 |
| 6 | SSSF | 10.900 | 23 | THG | 56.400 |
| 7 | ILSF | 14.400 | 24 | CAG | 32.400 |
| 8 | ABSF | 30.000 | 25 | ANG | 18.600 |
| 9 | THSF | 35.200 | 26 | BIAC | 36.500 |
| 10 | MCSF | 19.200 | 27 | BIIL | 28.500 |
| 11 | ACHT | 137.400 | 28 | TRCH | 26.300 |
| 12 | RAHT | 105.600 | 29 | FOOT | 23.300 |
| 13 | STHT | 81.900 | 30 | HUM | 5.890 |
| 14 | DAHT | 63.400 | 31 | FEM | 8.950 |
| 15 | TIHT | 45.200 | 32 | SITHT | 87.400 |
| 16 | SPHT | 93.100 | 33 | APCH | 16.600 |
| 17 | AGR | 26.100 | 34 | HDG | 55.200 |

DATA FOR SUBJECT #7, MEASUREMENT OCCASION #1.

TABLE 4-3
Subject #7, Measurement occasion #1

| variable | actual | predicted | tolerance | diff/tol |
|---|---|---|---|---|
| CHG | 82.100 | 86.074 | 2.814 | -1.412 |
| WAG | 68.800 | 63.605 | 5.128 | 1.013 |
| HUM | 5.890 | 6.323 | 0.428 | -1.013 |

| variable | actual | predicted | tolerance | diff/tol |
|---|---|---|---|---|
| CHG | 82.200 | 85.928 | 2.814 | -1.325 |

TABLE 4-4
Subject #7, Measurement occasion #2

| variable | actual | predicted | tolerance | diff/tol |
|---|---|---|---|---|
| SPHT | 93.000 | 95.778 | 2.129 | -1.305 |
| FAG | 24.100 | 22.725 | 1.344 | 1.023 |
| CHG | 82.200 | 85.222 | 2.814 | -1.074 |

TABLE 4-5
Subject #7, Measurement occasion #3

| variable | actual | predicted | tolerance | diff/tol |
|---|---|---|---|---|
| ILSF | 11.200 | 16.763 | 4.818 | -1.155 |
| SPHT | 92.200 | 95.264 | 2.129 | -1.439 |
| FAG | 24.100 | 22.495 | 1.344 | 1.194 |
| CHG | 82.200 | 85.928 | 2.814 | -1.325 |
| ANG | 18.200 | 19.985 | 1.414 | -1.263 |
| HUM | 5.930 | 6.374 | 0.428 | -1.038 |

| variable | actual | predicted | tolerance | diff/tol |
|---|---|---|---|---|
| FAG | 24.100 | 22.546 | 1.344 | 1.157 |
| CHG | 82.200 | 85.928 | 2.814 | -1.325 |
| ANG | 18.400 | 19.857 | 1.414 | -1.031 |
| HUM | 5.930 | 6.360 | 0.428 | -1.005 |

TABLE 4-6

Subject #8, Measurement occasion #1

| variable | actual | predicted | tolerance | diff/tol |
|---|---|---|---|---|
| MCSF | 23.800 | 15.408 | 5.518 | 1.521 |
| SITHT | 90.400 | 87.564 | 2.430 | 1.167 |

| variable | actual | predicted | tolerance | diff/tol |
|---|---|---|---|---|
| MCSF | 25.600 | 15.408 | 5.518 | 1.847 |
| SITHT | 90.200 | 87.564 | 2.430 | 1.085 |

TABLE 4-7

Subject #8, Measurement occasion #2

| variable | actual | predicted | tolerance | diff/tol |
|---|---|---|---|---|
| SSSF | 11.000 | 17.404 | 6.235 | -1.027 |
| ILSF | 13.000 | 18.389 | 4.818 | -1.119 |
| ABSF | 31.100 | 21.696 | 7.838 | 1.200 |
| MCSF | 26.400 | 14.078 | 5.518 | 2.233 |
| CHG | 85.800 | 81.688 | 2.814 | 1.461 |
| WAG | 66.300 | 71.759 | 5.128 | -1.065 |

| variable | actual | predicted | tolerance | diff/tol |
|---|---|---|---|---|
| MCSF | 25.600 | 14.078 | 5.518 | 2.088 |
| CHG | 84.700 | 81.513 | 2.814 | 1.133 |

TABLE 4-8

Subject #8, Measurement occasion #3

| variable | actual | predicted | tolerance | diff/tol |
|---|---|---|---|---|
| THSF | 37.600 | 25.438 | 9.498 | 1.281 |
| MCSF | 25.600 | 13.986 | 5.518 | 2.105 |
| CHG | 84.700 | 81.766 | 2.814 | 1.043 |

| variable | actual | predicted | tolerance | diff/tol |
|---|---|---|---|---|
| THSF | 36.000 | 25.438 | 9.498 | 1.112 |
| MCSF | 25.600 | 13.986 | 5.518 | 2.105 |
| CHG | 84.700 | 81.766 | 2.814 | 1.043 |

TABLE 4-9
Subject #13, Measurement occasion #1

| variable | actual | predicted | tolerance | diff/tol |
|---|---|---|---|---|
| SPHT | 92.400 | 94.685 | 2.129 | -1.074 |
| FOOT | 23.900 | 25.071 | 1.156 | -1.012 |

| variable | actual | predicted | tolerance | diff/tol |
|---|---|---|---|---|
| SPHT | 91.700 | 94.685 | 2.129 | -1.402 |

TABLE 4-10
Subject #13, Measurement occasion #2

| variable | actual | predicted | tolerance | diff/tol |
|---|---|---|---|---|
| DAHT | 64.100 | 65.799 | 1.631 | -1.042 |
| SPHT | 91.700 | 94.854 | 2.129 | -1.482 |
| CAG | 38.800 | 36.038 | 2.674 | 1.033 |
| FOOT | 24.000 | 25.193 | 1.156 | -1.031 |

TABLE 4-11
Subject #13, Measurement occasion #3

| variable | actual | predicted | tolerance | diff/tol |
|---|---|---|---|---|
| SPHT | 91.600 | 94.591 | 2.129 | -1.405 |
| CHG | 86.100 | 83.183 | 2.814 | 1.037 |
| CAG | 38.800 | 35.857 | 2.674 | 1.101 |

| variable | actual | predicted | tolerance | diff/tol |
|---|---|---|---|---|
| SPHT | 91.700 | 94.591 | 2.129 | -1.358 |
| CAG | 38.800 | 35.997 | 2.674 | 1.048 |

TABLE 4-12
Subject #57, Measurement occasion #1

| variable | actual | predicted | tolerance | diff/tol |
|---|---|---|---|---|
| STHT | 84.600 | 82.909 | 1.609 | 1.051 |
| DAHT | 66.600 | 68.415 | 1.631 | -1.112 |
| TIHT | 43.000 | 45.260 | 1.814 | -1.246 |
| BIAC | 38.000 | 35.429 | 2.489 | 1.033 |
| FOOT | 23.300 | 25.198 | 1.156 | -1.641 |
| SITHT | 91.000 | 93.827 | 2.430 | -1.163 |

| variable | actual | predicted | tolerance | diff/tol |
|---|---|---|---|---|
| MCSF | 6.700 | 12.492 | 5.518 | -1.050 |
| FOOT | 23.600 | 25.357 | 1.156 | -1.519 |
| HUM | 6.700 | 6.272 | 0.428 | 1.001 |

TABLE 4-13
Subject #57, Measurement occasion #2

| variable | actual | predicted | tolerance | diff/tol |
|---|---|---|---|---|
| MCSF | 6.700 | 12.860 | 5.518 | -1.116 |
| FOOT | 23.600 | 25.141 | 1.156 | -1.333 |
| HUM | 6.700 | 6.263 | 0.428 | 1.023 |

TABLE 4-14
Subject #57, Measurement occasion #3

| variable | actual | predicted | tolerance | diff/tol |
|---|---|---|---|---|
| MCSF | 6.300 | 12.709 | 5.518 | -1.162 |
| FOOT | 23.600 | 25.095 | 1.156 | -1.293 |
| HUM | 6.700 | 6.268 | 0.428 | 1.011 |
| FEM | 8.870 | 9.578 | 0.679 | -1.043 |

| variable | actual | predicted | tolerance | diff/tol |
|---|---|---|---|---|
| MCSF | 6.700 | 12.709 | 5.518 | -1.089 |
| FOOT | 23.600 | 25.095 | 1.156 | -1.293 |

Table 4-4 shows the results using the data from the second measurement occasion of subject #7. After the first VERIFY three measures were identified as deviant. Since the values for all three were in fact the median values (ie. re-measurement of these variables produced exactly the same values) the WRITE command was given and all three were "forced" in. Some of the predictor variables for each of these "forced" values might have been considerably different from the median but just within tolerance. The influence of these independent variables on the predicted values may have been the cause of the discrepancies. But to be consistent for this demonstration these values were treated as "Force-ins".

The third measurement occasion for subject #7 (Table 4-5) showed six deviant values reduced to four after entering median values (Three of the six values were already median). The four remaining deviant values were "Forced-in".

For subject #8, the first measurement occasion produces two out of range values (Table 4-6). After entering the median values they were found to be still out of range, (by an even larger discrepancy for MCSF) and were thus "forced" in. Occasion two (Table 4-7) showed that six out of range values were reduced to two when the simulated re-measurement changed five of the six original values. Only SSSF was originally the median.. Occasion three (Table 4-8) required only 1 change which did not reduce the number of out of range values.

Similar results for subjects #13 and #57 are shown in Tables 4-9 through 4-14.

Of the twelve sets of data monitored in this demonstration MCSF was forced in six times, CHG five times and FOOT four times. This indicates that one or more of the following influences was present:

1. Systematic error in measuring these subjects (ie. instrument calibration or technique).

2. Systematic error while collecting the sample used to generate the regressions.

3. Biological variation between subjects.

It is apparent that the measurement and prediction of these variables required attention.

## 4.4. The Detection of Blunders

To evaluate the system's performance with respect to the detection of blunders, the same twelve sets of values that were used in the previous demonstration were seeded with a number of blunders. The seeded blunders are all familiar to anthropometrists and probably to most data collectors. They are therefore assumed to be representative of the blunders that might be encountered during a real measurement situation.

Five categories were selected and for each of these categories, 10 blunders were randomly introduced into the data.

The system's performance in detecting these blunders is shown in Table 4-15.

The categories of blunders that were selected are:

1. Digit reversals between the 10's position and 1's position; or between the 1's and tenth's positions.

2. Decimal error: shift decimal right.

3. Decimal error: Shift decimal left.

4. Legibility errors as follows:

| | | |
|---|---|---|
| zero | replaced with | six |
| one | replaced with | seven |
| two | replaced with | three |
| three | replaced with | two |
| four | replaced with | nine |
| five | replaced with | two |
| six | replaced with | zero |
| seven | replaced with | one |
| eight | replaced with | nine |
| nine | replaced with | seven |

5. Wrong position on form.

A computer program which generated pseudo random numbers (Appendix F) was written to select each subject, measurement occasion and variable which was to receive the introduced error. Overall, the system was able to detect 94% of the introduced blunders.

TABLE 4-15

BLUNDER DETECTION

| TYPE OF BLUNDER | NO.DETECTED |
|---|---|
| Digit Reversal | 8/10 |
| Decimal Error (Right) | 10/10 |
| Decimal Error (Left) | 10/10 |
| Legibility Error | 10/10 |
| Form Location Error | 9/10 |
| Total Errors Detected | 47/50 |

# V. EVALUATION

## 5.1. Evaluation of the Detection of Systematic Errors

The preceding chapter is a demonstration of how the Data Monitoring System can provide the investigator with information concerning the possibility of errors. However, the demonstration was limited since the process for reconciling the deviations was simulated rather than a real re-measurement situation. (i.e. The median values rather than re-measured values were used). Some of the median values which were entered as re-measured values may have also been erroneous but, even when they too were rejected by the error scan, we accepted them as correct and they were "forced in".

These median values may have been errors because the triple measurement protocol used during the original collection of the data provided the measurer with no feedback concerning discrepancies from previous measures. He was however informed of blunders whenever they were noticed by the recorder. Without information from previous measurements the measurer was not aware of discrepancies and did not treat any of the measures with special care. Had this information been available, the median values might have been different. In other words, if the data collection process was a real event rather than a simulated one, some of the "forced in" data might not have been out of range after re-measurement. It is expected that the Data

Monitoring System would perform slightly better under real measurement conditions than under simulated ones.

The performance, even under simulated conditions, was exceptionally good. The variables which were most often identified as possible errors are, in fact, items which are difficult to measure repeatedly and obtain consistent results. The medial calf skinfold measure, which was the most often forced in value, is, like all other skinfold measures, extremely variable between subjects as well as between repeated measures on the same subject. The repeated measures are affected by the pressure of the caliper as well as other random and systematic sources of error. Chest girth is also difficult to measure accurately since posture and breathing can have a considerable influence. Both of these measures are also more variable for female subjects than male.

The Data Monitoring System therefore has consistently identified two variable which are known to anthropometrists as particularly troublesome.

## 5.2. Evaluation of the detection of blunders

Since the data that the Data Monitoring System scans has already been checked (to some degree) by the recorder, the blunders that it detects are only those which have been missed by the investigator and recorder. Presumably any noticed blunders will be corrected before entering the data. Without the Data Monitoring System the unnoticed blunders would be recorded in the data assembly. Therefore the detection of a very small percentage of unnoticed blunders would justify the use of the Data Monitoring System. If the system performs as well under real measurement conditions as it did in the detection of randomly introduced blunders, then it will be extremely useful in preventing a large percentage of blunders from reaching the final data record.

# VI. DISCUSSION

It is essential with any data assembly that the information contained therein is correct. Efforts to deal with this problem have been extensive as evidenced by the work done in areas such as the detection of 'outliers' and data cleaning. These efforts are aimed at detecting and dealing with incorrect information after it has been assembled.

The approach taken here is to attempt to identify incorrect information BEFORE it is recorded in the data assembly. This is achieved by providing an interface between the data collector and the data assembly; an interface which 'knows' something about the data and can inform the data collector if an unexpected value is encountered.

The Data Monitoring System presented in this thesis allows the description of incoming data in terms of complex mathematical relationships. The relationships may include variables which are given values during the monitoring process so that any one prediction may be influenced by the incoming data itself. The system has been designed specifically for use during the collection of anthropometric data, but it may also be used in other areas. Whenever the collected data can be predicted from some mathematical expression the Data Monitoring System could be used. Biomechanics and Physiology are two areas where the phenomena being studied (i.e. measured) behave according to basic physical laws. If prediction techniques can be influenced by some understanding of these laws then a model can be defined and the subsequent data collection process can be monitored.

Questionnaire data may also be collected and monitored with the Data Monitoring System if some of the variables are predictable from others. For example:

> if SEX is entered as '1' for female and '0' for male, then the number of pregnancies could be predicted by multiplying the response for pregnancies by sex and setting a tolerance of zero.
>
> PREG=PREG*SEX
>
> This equation will always predict zero pregnancies for a male. A non-zero response for the number of pregnancies (PREG) by a male respondent will be recognized by the Data Monitoring System as a possible error.

In this way a questionnaire could be designed in which the data could be collected and checked by the Data Monitoring System.

It is important to note at this point that a system such as this can be misused. The old adage of "garbage in, garbage out" holds true for this system just as it does for all other data processing systems. Before the user can rely on the system to produce valid output in the form of detected errors, he must be confident that the input, the model, is valid. But, too much confidence may result in giving the model more credit than it deserves. After it has been developed and defined in the system, it might be easy to forget all of the assumptions on which the predictions are based and how "good" the model is at predicting. The tendancy then may be to accept the predicted value as the "true" value and bias subsequent re-measurements so that they conform to the predicted values. Extreme values in the population must be measured objectively and if repeated measures consistently show deviations from the predicted value then the prediction model may be at fault and should be reviewed.

The Data Monitoring System acts as an informed observer of the collection process if the model is well defined. It may also serve to enhance training by making a new measurer aware of errors that result from bad technique. In fact, it may be used specifically as a training device by defining the model as simply the set of values obtained by an expert measurer. The trainee would then be required to make the same measurements and the obtained values would be compared to the predicted values (the expert measurer's values). The out of range values would indicate to the trainee where extra attention is needed.

The system, as it now stands, performs quite well and should prove useful to data collectors. There are however some features that could be included in the system to make it more versatile.

An obvious enhancement would be the inclusion of logical operators and conditional expressions in the model. Expressions such as

IF (SEX = 1) AND (AGE > 22) THEN HT=WT/2.0

would provide opportunities for developing a much more discriminating model.

Automatic adjustments to the model as new data are collected, would also be useful. If the model has been developed from a sample drawn from a different (although similar) population, then the system should adjust its predictions to conform more closely to the new sample. This could be done by applying to the predicted values, some transformation which is dependent on the new data and how much it deviates from the predicted. The more data that is collected the better the predictions would be.

A more "intelligent" algorithm for detecting the most likely cause of the out of range values would also be useful. Currently, because of the possible interdependancies of the regression equations and the variables, it is difficult to determine which is the cause of a discrepancy and which is an effect. An algorithm which could take these relationships into account could provide more meaningful information to the data collector.

The Data Monitoring System is not a substitute for a thoroughly trained measurement and recording team but it is a complementary and versatile tool. A tool which will not only provide for machine recording of data but will be capable of providing the data collectors with a tireless performance of the tedious task of reviewing every piece of data that is recorded; a review which will be done according to criteria established by the data collectors. The speed, accuracy and diligence of the computer will, to some extent, compensate for the slow and error prone attributes of the human data collector.

## APPENDIX A

The following is the compiler listing of the DATAMONITOR program. The line number and the library segment numbers from the left side of the listing have been removed.

```
 1:D   1 {$L DATA:MAIN.LST}
 1:D   1 PROGRAM DATAMONITOR;
 1:D   3 {$  }
 1:D   3
 2:D   3   FUNCTION SIN(X:REAL):REAL;
 3:D   3   FUNCTION COS(X:REAL):REAL;
 4:D   3   FUNCTION EXP(X:REAL):REAL;
 5:D   3   FUNCTION ATAN(X:REAL):REAL;
 6:D   3   FUNCTION LN(X:REAL):REAL;
 7:D   3   FUNCTION LOG(X:REAL):REAL;
 8:D   3   FUNCTION SQRT(X:REAL):REAL;
 8:D   5
 1:D   5 IMPLEMENTATION
 1:D   1     (* 6502 compatible *)
 1:D   1
 1:D   1
 1:D   3  USES TRANSCEND;
 2:D   1  procedure poke(addr,data:integer);
 3:D   3  function peek(addr:integer):integer;
 4:D   3  function keybuff:char;
 5:D   3  function priority(ITEM:string;TYP:integer):integer;
 6:D   1  procedure getaline(var line:string;echo:boolean);
 7:D   3  function pwr(x,y:real):real;
 8:D   3  function ctb(STR:string):real;
 9:D   1  procedure get_item(EXPR:string;var INDX:integer;
 9:D   3                    DLMTR:string;
 9:D   4                    ITEMLEN:integer; var ITEM:string);
10:D   1  procedure message(MSG:string);
11:D   1  procedure pause;
 1:D   1
 1:D   3 USES TRANSCEND,UTILS;
 1:D   1  const
 1:D   1    stringsize =80;
 1:D   1    table_size =50;
 1:D   1  type
 1:D   1    symb_rec   = record
 1:D   1                  symb: string;
 1:D   1                  predictor:string;
 1:D   1                  expr   :string;
 1:D   1                  actual:real;
 1:D   1                  tol    :real;
 1:D   1                  fmt    :ARRAY[1..3] of integer;

 1:D   1                  flag   :ARRAY[1..2] of boolean;
 1:D   1                end;
 1:D   1    symb_table=array[1..table_size] of symb_rec;
 1:D    1 var
 1:D    1     TABLE:symb_table ;
 1:D  601     TEMPSTR,delim:string;
 1:D  683     namelen:integer;
 2:D    1   procedure init;
 3:D    1   procedure tablein(TARG,EXPR,POLEXPR:string);
 4:D    3   function valof(ITEM:string;var ERR:boolean):real;
 5:D    1   procedure intopost(EXPR:string;var INDX:integer;
 5:D    3               DLMT:string;VNAMLEN:integer;
 5:D    5               var ITEM1:string;var POLSTR:string);
 6:D    1   procedure evaluate(POLSTR:string;DLMT:string;
 6:D    3                         VNAMLEN:integer;VAR VAL:real;
 6:D    5                         VAR ERR:integer);
 6:D   88
 7:D    1   procedure parse(EXPR:string);
 8:D    1   procedure editor;
 9:D    1   procedure definer;
10:D    1   procedure collector;
 1:D    1  uses transcend,utils,polish;
 1:D    3  const
 1:D    3    linesize=80;
 1:D    3    minforpred=40;
 1:D    3    ESC=27;
 1:D    3  type
 1:D    3    linetype=array[1..linesize] of char;
 1:D    3    fmt=array[1..3] of integer;
 1:D    3  var
 1:D    3    device   :string;
 1:D   44    EXPR,STR,STR2,infname:string;
 1:D  208    keyboard,listfile:interactive;
 1:D  810    DATAFILE:text;
 1:D  111    ERRCODE,I,K:integer;
 1:D  114    X,Y:real;
 1:D  118    firstout,OPEN,DONE,QUIT,vfmt,batchflag:boolean;
 1:D  124    INFILE:FILE OF CHAR;
 1:D  425    FYLEA : file of symb_rec;
 1:D  857    LINE1,LINE2:linetype;
 1:D  017    F:fmt;
 1:D  020    GUESS:array[1..table_size] of real;
 1:D  120
 2:D    1  procedure scanner(terse:boolean);
 2:D    2  var
 2:D    2    I,J,K,count,indx,err  :integer;
 2:D    8    value,temp,newval  :real;
 2:D   14    msg    :string;
 2:D   55
```

```
3:D    3 function culprit:integer;

3:0    0 begin
3:1    0   count:=0;INDX:=0;
3:1    8   writeln(
3:1    8    'Now looking for most likely cause of errors....');
3:1   75   for I:=1 to table_size do begin
3:3   89     if (TABLE[I].symb <> '') and (TABLE[I].flag[1])
3:3  133     then begin
3:5  137       temp:=TABLE[I].actual;
3:5  160       TABLE[I].actual:=GUESS[I];
3:5  193       k:=0;
3:5  197       for j:=1 to table_size do begin
3:7  211         if TABLE[J].flag[1] then
3:8  239         if pos(TABLE[I].symb,TABLE[J].expr)<>0
3:8  274         then begin
3:0  278           write('.');
3:0  288           evaluate(TABLE[J].predictor,DELIM,NAMELEN,
3:0  312                    value,err);
3:0  321           if (TABLE[J].actual >= value-TABLE[J].tol)
3:0  365                and
3:0  365              (TABLE[J].actual <= value+TABLE[J].tol)
3:0  409           then K:=K+1;
3:9  420         end; {end if}
3:7  420         if ord(keybuff)=ESC
3:7  425         then J:=table_size;  { leave loop }
3:6  433       end;   {end for}
3:5  443       if k > count then begin indx:=I; count:=K; end;
3:5  464       TABLE[I].actual := temp;
3:4  487     end; {end if}
3:3  487     if ord(keybuff)=ESC
3:3  492     then I:=table_size;  { leave loop }
3:2  500   end; {end for}
3:1  510   culprit:=indx;
3:1  515   writeln;
3:0  523 end; {culprit}
3:0  550
2:0    0 begin
2:1    0   count:=0;
2:1    3   writeln(listfile,
2:1    3       'variable':14,'actual':11,'predicted':14,
2:1   62       'tolerance':12,'diff/tol':9);
2:1  111   for I:=1 to table_size do begin
2:3  123     if TABLE[I].symb <> '' then begin
2:5  142       err:=0; TABLE[I].flag[1]:=false;
2:5  170       evaluate(TABLE[I].predictor,DELIM,NAMELEN,
2:5  192                value,err);
2:5  199       GUESS[I]:=value;
2:5  216       if err=0
2:5  217       then if (TABLE[I].actual < GUESS[I]-TABLE[I].tol)
```

```
2:6 269                 or
2:6 269            (TABLE[I].actual > GUESS[I]+TABLE[I].tol)

2:6 317               then err:=5;
2:5 323          case err of
2:5 326           0:msg:=' ';
2:5 333           1:msg:='Stack underflow';
2:5 357           2:msg:='Stack overflow';
2:5 380           3:msg:='Undefined variable';
2:5 407           4:msg:='Divide by zero ';
2:5 431          end;
2:5 448          if (I=1) or (err <>0) or (not terse) then begin
2:7 460            write(listfile,
2:7 460             TABLE[I].symb:10,'   ',TABLE[I].actual:10:3,
2:7 522             '      ',value:10:3,'   ',TABLE[I].tol:10:3);
2:7 593            if err=5
2:7 594            then begin
2:9 598              writeln(listfile,
2:9 598                     (TABLE[I].actual-
2:9 617                     GUESS[I])/TABLE[I].tol:9:3);
2:9 663              count:=count+1;
2:9 668              TABLE[I].flag[1]:=true;
2:8 693            end
2:7 693            else writeln(listfile,'   ',msg);
2:6 728          end;
2:4 728        end;
2:3 728        if ord(keybuff)=ESC
2:3 733        then I:=table_size;  { leave loop }
2:2 740      end;  { for }
2:1 747      if (count > 1) and (ord(keybuff) <> ESC)
2:1 757      then begin      {look for most likely bad variable}
2:3 760        indx:=culprit;
2:3 766        if indx > 0 then
2:4 771        writeln(listfile,
2:4 771           concat('More than one error may be caused by ',
2:4 825                  TABLE[INDX].symb));
2:2 857      end;
2:1 857      if(not batchflag)and(ord(keybuff)<>ESC) then pause;
2:1 874      if ord(keybuff) = ESC then get(keyboard);
2:1 891      message(' ');
2:0 895  end;   (* end scanner *)
2:0 916
4:D   1  procedure dataout;
4:0   0  begin
4:1   0    K:=0;
4:1   4    if (not firstout) then begin
4:3  10      rewrite(datafile,'DATA:DATAOUT'); K:=ioresult;
4:3  38      if K <> 0 then reset(datafile,'DATAOUT');
4:2  65    end;
4:1  65    for I:=1 to table_size do
```

```
4:2  79      if TABLE[I].symb <> '' then
                     write(datafile,TABLE[I].actual:
4:3 146                                           TABLE[I].fmt[3]);
4:1 186    writeln(datafile);

4:1 194    firstout:=true;
4:0 198 end;
4:0 212
4:0 212
5:D   1 procedure iosub(INPUT:boolean);
5:D   2 var
5:D   2    name : string;
5:D  43    I,IOR: integer;
5:D  45    trying:boolean;
5:D  46
5:0   0 begin
5:1   0    trying:=true;
5:1   3    writeln;
5:1  11    if INPUT
5:1  11    then write(' Read what file? ')
5:1  43    else write(' Save what file? ');
5:1  74    getaline(name,true);
5:1  80    if (length(name)=0) or (ord(name[1])=ESC)
5:1  93    then exit(iosub);
5:1 100    (*$I-*) reset(FYLEA,name); (*$I+*)
5:1 110    IOR:=ioresult;
5:1 114    while trying do begin
5:3 118      case IOR of
5:3 122        0: begin
5:5 122             seek(FYLEA,0);
5:5 131             for I:=1 to table_size do begin
5:7 144               if INPUT then begin
5:9 147                 (*$I-*)get(FYLEA);IOR:=ioresult;(*$I+*)
5:9 157                 if not EOF(FYLEA) then begin
5:1 168                   TABLE[I]:=FYLEA^;
5:0 187                 end
5:9 187                 else TABLE[I].symb:='';
5:8 207               end
5:7 207               else begin
5:9 209                 if TABLE[I].symb <> '' then begin
5:1 229                   FYLEA^:=TABLE[I];
5:1 248                   (*$I-*)put(FYLEA);IOR:=ioresult;
5:1 258                   (*$I+*)
5:0 258                 end;
5:8 258               end;
5:6 258             end; {for statement}
5:5 266             if IOR=0 then trying:=false;
5:5 275             close(FYLEA,crunch);
5:4 284           end;
5:3 286        7: begin
5:5 286             writeln (' illegal filename  re-enter');
5:5 333             iosub(input);exit(iosub);
5:4 340           end;
```

```
5:3 342        8: begin
5:5 342             writeln (' no room on disk.');

5:5 379             trying:=false;
5:4 382           end;
5:3 384        9: begin
5:5 384             writeln;
5:5 392             writeln('Put proper diskette in drive...');
5:5 443             pause;
5:5 446             if ord(keybuff)=ESC then exit(iosub);
5:5 459             (*$I-*) reset(FYLEA,name); (*$I+*)
5:5 469             IOR:=ioresult;
5:4 473           end;
5:3 475       10: begin
5:5 475             if input then begin
5:7 478               trying:=false;
5:7 481               writeln(' no such file!');
5:7 515               PAUSE;
5:7 518               if ord(keybuff)=ESC then exit(iosub);
5:6 531             end
5:5 531             else begin
5:7 533               close(FYLEA);
5:7 542               (*$I-*) rewrite(FYLEA,name); (*$I+*)
5:7 552               IOR:=ioresult;
5:6 556             end;
5:4 556           end;
5:3 558      1,2,3,4,5,6,11,12,13,14,15,16:
5:4 558           begin (*otherwise statement*)
5:5 558             writeln(' I/O ERROR #',IOR);PAUSE;
5:5 604             trying:=false;
5:4 607           end;
5:3 609      end; (* end case *)
5:2 650    end; (*end while*)
5:1 652    write(chr(12)); {clear screen}
5:0 662 end;  (* end of iosub *)
5:0 684
6:D   1 procedure readarec(recno:integer;var buffer:linetype;
6:D   3                     len:integer;var err:boolean);
6:D   5 var
6:D   5   I,J,K,IOR:integer;
6:0   0 BEGIN
6:1   0   err:=false;
6:1   3   for I:=1 to linesize do buffer[I]:=' ';
6:1  32   if recno < 1 then K:=1 else K:=recno;
6:1  45   IOR:=0;
6:1  48   for J:=1 to K do
6:2  59     for I:=1 to len do begin
6:4  70       buffer[I]:=INFILE^;
```

```
6:4  85        (*$I-*) get(INFILE); (*$I+*)
6:4  91        IOR:=ioresult;
6:4  95        if IOR<>0
6:4  96        then begin err:=true; exit(readarec); end;
6:3 107      end;

6:0 121 end;
6:0 140
7:D   1 procedure extract(F:fmt;line:linetype;var val:real);
7:D  87 var
7:D  87   I,J,K:integer;
7:D  90   TXT  :string;
7:D 131   err  :boolean;
7:0   0 begin
7:1   0   TXT:='';
7:1  17   J:=F[2]+F[1]-1;
7:1  44   for I:=F[1] to J do begin
7:3  70     TXT:=concat(TXT,' ');
7:3 101     if line[I]=' '
7:3 113     then line[I]:=chr(ord('0'));
7:3 130     TXT[length(TXT)]:=line[I];
7:2 150   end;
7:1 158   if POS('.',TXT)=0
7:1 169   then insert('.',TXT,length(TXT)-f[3]+1);
7:1 201   VAL:=valof(TXT,err);
7:0 214 end;
7:0 228
8:D   1   procedure getequations;
8:D   1   var
8:D   1     I,J:integer;
8:D   3     buff:string;
8:0   0   begin
8:1   0     write(
8:1   0       'Equations are in what file?');
8:1  39       getaline(infname,true);
8:1  45     if ord(infname[1])=ESC
8:1  50     then exit(getequations);
8:1  58     writeln;
8:1  66     reset(infile,infname);
8:1  78     for I:=1 to table_size do TABLE[I].symb:='';
8:1 114     I:=1;
8:1 117     while not eof(infile) do begin
8:3 128       buff:='';
8:3 135       J:=1;
8:3 138       repeat
8:4 138         buff:=concat(buff,' ');
8:4 165         buff[J]:=infile^;
8:4 175         J:=J+1;
8:4 180         get(infile);
8:3 188       until INFILE^=';';
```

```
8:3 197          TEMPSTR:='';
8:3 206          J:=1;
8:3 209          get(infile);
8:3 217          repeat
8:4 217            TEMPSTR:=concat(TEMPSTR,' ');
8:4 248            TEMPSTR[J]:=infile^;

8:4 260            J:=J+1;
8:4 265            get(infile);
8:3 273          until INFILE^=';';
8:3 282          TABLE[I].tol:=ctb(TEMPSTR);
8:3 307          repeat
8:4 307            get(infile);
8:3 315          until (infile^ <> ' ') or eof(infile);
8:3 333          writeln(buff);
8:3 352          parse(buff);
8:3 357          I:=I+1;
8:2 362        end;
8:0 364      end;
8:0 388
9:D   1  procedure batch;
9:D   1  var
9:D   1    count,I:integer;
9:D   3    F:fmt;
9:D   6    terse:boolean;
9:0   0  begin
9:1   0    terse:=false;
9:1   3    batchflag:=true;
9:1   7    write('Data is in what file?');
9:1  40    getaline(infname,true);
9:1  46    if (length(infname)=0) or (ord(infname[1])=ESC)
9:1  59    then exit(batch);
9:1  66    close(infile);
9:1  75    {$i-}reset(infile,infname);{$i+}
9:1  87    if ioresult <> 0
9:1  89    then begin
9:3  93      message('I/O ERR ');
9:3 107      pause;
9:3 110      exit(batch);
9:2 114    end;
9:1 114    readarec(1,line1,78,done);
9:1 124    readarec(1,line2,78,done);
9:1 134    if done then begin
9:3 139      close(infile);
9:3 148      message('I/O ERR ');
9:3 162      pause;
9:3 165      exit(batch);
9:2 169    end;
9:1 169    count:=0;
9:1 172    writeln(chr(12));
```

```
9:1  190   repeat
9:2  190     gotoxy(1,2);
9:2  195     write(
9:2  195     'P)rinter, C)onsole, F)ull, T)erse, G)o, Q)uit');
9:2  252     gotoxy(1,3);
9:2  257     if terse
9:2  257     then write('"TERSE"') else write('"FULL"');

9:2  299     write(' list on ');
9:2  320     write(device,' ');
9:2  341     gotoxy(47,2);
9:2  346     get(keyboard);
9:2  354     case keybuff of
9:2  361      'p','P':device:='PRINTER:';
9:2  378      'q','Q':exit(batch);
9:2  384      'c','C':device:='CONSOLE:';
9:2  401      'r','R':device:='REMOUT: ';
9:2  418      'f','F':terse:=false;
9:2  423      't','T':terse:=true ;
9:2  428     end;
9:2  536     close(listfile);
9:2  545     rewrite(listfile,device);
9:1  557   until (keybuff = 'g') or (keybuff = 'G');
9:1  574   writeln(chr(12));
9:1  592   repeat
9:2  592     for I:=1 to table_size do
9:2  603       {get an array full of data }
9:3  603       if TABLE[I].symb <> '' then begin
9:5  622         count:=count+1;
9:5  627         F:=TABLE[I].fmt;
9:5  645         if F[1] <= 78
9:5  656         then extract(F,line1,TABLE[I].actual)
9:5  679         else begin
9:7  683           F[1]:=F[1]-78;
9:7  707           extract(F,line2,TABLE[I].actual);
9:6  728         end;
9:5  728         if ord(keybuff)=ESC
9:5  733         then I:=table_size; {leave loop}
9:4  740       end;
9:4  747     { now scan for errors }
9:2  747     if count=0 then begin
9:4  752       message('No predictor equations and formats!');
9:4  793       close(infile);
9:4  802       exit(batch);
9:3  806     end;
9:2  806     SCANNER(terse);
9:2  809     WRITELN(LISTFILE);
9:2  817     readarec(-1,line1,78,done);
9:2  828     readarec(-1,line2,78,done);
9:1  839   until done or (ord(keybuff)=ESC) or EOF(INFILE);
```

```
 9:1 861    close(infile);close(listfile);
 9:1 879    reset(listfile,'CONSOLE:');
 9:1 900    if ord(keybuff)=ESC then get(keyboard);
 9:0 917  end;
 9:0 940
10:D   1  procedure modellist;
10:D   1  VAR
10:D   1    I,J,K,L:integer;

10:0   0 begin
10:1   0    L:=1;
10:1   3    writeln(listfile);writeln(listfile);
10:1  19    writeln(listfile,
10:1  19               '          FORMAT    TOLERANCE      EQUATION');
10:1  77    for I:=1 to table_size do begin
10:3  88      if TABLE[I].symb <> '' then begin
10:5 107        write(listfile,I:3,
10:5 117              TABLE[I].fmt[1]:6,TABLE[I].fmt[2]:3,
10:5 181              TABLE[I].fmt[3]:3,
10:5 213              TABLE[I].tol:10:3,'      ');
10:5 256        J:=pos('=',TABLE[I].expr);
10:5 281        write(listfile,
10:5 281            copy(TABLE[I].expr,1,J-1),' ':namelen-J+1);
10:5 332        K:=length(TABLE[I].expr)-J+1;
10:5 354        if K > 40 THEN K:=40;
10:5 362        repeat
10:6 362          write(listfile,copy(TABLE[I].expr,J,K));
10:6 394          writeln(listfile);
10:6 402          J:=J+K;K:=length(TABLE[I].expr)-J+1;
10:6 429          if K>0 then write(listfile,' ':36);
10:5 444        until K <= 0;
10:5 449        L:=L+1;
10:5 454        if ((L mod 5)=0) and (device='REMOUT: ')
10:5 474        then pause;
10:4 480      end;{begin}
10:3 480      if ord(keybuff)=ESC then exit(modellist);
10:2 493    end; {for}
10:1 500    pause;
10:1 503    gotoxy(0,0); write(chr(11));
10:0 518  end;
10:0 538
11:D   1  procedure datalist;
11:D   1  VAR
11:D   1    I,K:integer;
11:0   0 begin
11:1   0    writeln(listfile);writeln(listfile);
11:1  16    writeln(listfile,
11:1  16    'VARIABLE':19,'VALUE':15,'VARIABLE':22,'VALUE':15);
11:1  98    for I:=1 to table_size do if TABLE[I].symb <> ''
11:2 121                                then K:=I;
```

```
11:1 138      if odd(K) then K:=K+1;
11:1 146      for I:=1 to (K div 2) do begin
11:3 159        if TABLE[I].symb <> ''
11:3 171        then write(listfile,I:9,
11:4 188                   '  ',TABLE[I].symb:namelen+3,
11:4 228                   TABLE[I].actual:16:3,' ':8)
11:3 264        else write(' ':namelen+38);
11:3 281        if TABLE[I+(K div 2)].symb <> ''
11:3 297        then writeln(listfile,I+(K div 2):3,

11:4 318                   '  ',TABLE[I+(K div 2)].symb:namelen+3,
11:4 362                      TABLE[I+(K div 2)].actual:16:3)
11:3 400        else writeln;
11:3 410        if ord(keybuff)=ESC then exit(datalist);
11:3 423        if ((I mod 5)=0) and (device='REMOUT: ')
11:3 443        then pause;
11:2 449      end;
11:1 456    pause;
11:1 459    gotoxy(0,0); write(chr(11));
11:0 474 end;
11:0 492
12:D   1 procedure execute;
12:0   0 begin
12:1   0    writeln;writeln;
12:1  16    write(' ENTER EXPRESSION=>');
12:1  47    getaline(expr,true);
12:1  54    str:='(';str2:='';
12:1  66    K:=1;
12:1  70    intopost(expr,K,'!',5,str,str2);
12:1  85    evaluate(str2,'!',5,X,K);
12:1  98    IF K<>0 THEN WRITELN(' ERROR') ELSE
12:2 133    WRITELN('RESULT=',X);
12:1 175    PAUSE;
12:0 178 end;
12:0 190
13:D   1 procedure dump;
13:D   1 var print,mdl:boolean;
13:0   0 begin
13:1   0    print:=false; {default to console}
13:1   3    device:='CONSOLE:';
13:1  18    mdl  :=true ; {default to model  }
13:1  21    repeat
13:2  21      gotoxy(0,1);
13:2  26      writeln(
13:2  26        'M(odel, D(ata, P(rinter, C(onsole, G)o, Q)uit');
13:2  91      if mdl then write('Model') else write(' Data');
13:2 130      write(' output will be sent to ',device);
13:2 177      gotoxy(47,1);
13:2 182      get(keyboard);
13:2 190      case keybuff of
```

```
13:2 197        'q','Q':exit(dump);
13:2 203        'm','M':mdl:=true;
13:2 208        'd','D':mdl:=false;
13:2 213        'r','R':begin
13:4 213                  print:=true;
13:4 216                  device:='REMOUT: ';
13:3 231                end;
13:2 233        'p','P':begin
13:4 233                  print:=true;
13:4 236                  device:='PRINTER:';

13:3 251                end;
13:2 253        'c','C':begin
13:4 253                  print:=false;
13:4 256                  device:='CONSOLE:';
13:3 271                end;
13:2 273      end;
13:2 376      close(listfile);
13:2 385      rewrite(listfile,device);
13:1 397    until (keybuff='g') or (keybuff='G');
13:1 414    if mdl then modellist else datalist;
13:0 423  end;  {dump}
13:0 440
 1:0    0 begin {main program}
 1:1    0   init;
 1:1   63   device:='CONSOLE:';
 1:1   78   reset(keyboard,'CONSOLE:');
 1:1   99   firstout:=false;QUIT:=false;OPEN:=false;vfmt:=false;
 1:1  115   repeat
 1:2  115     close(listfile);
 1:2  124     reset(listfile,device);
 1:2  136     batchflag:=false;
 1:2  140     gotoxy(0,1);
 1:2  145     writeln(
 1:2  145       ' D(efine, C(ollect, V(erify, E(dit, W(rite,',
 1:2  200       'R(ead, S(ave, Q(uit.',' ':15);
 1:2  250     write(' Output device = "',device,'"',
 1:2  301             'Output format = ':30);
 1:2  329     if vfmt
 1:2  329     then writeln('"TERSE".')
 1:2  362     else writeln('"FULL". ');
 1:2  392     gotoxy(64,1);
 1:2  397     EXPR:=' ';
 1:2  403     get(keyboard); EXPR[1]:=keyboard^;
 1:2  422     write(chr(12));
 1:2  432     case EXPR[1] of
 1:2  440       'b','B': batch;        {data from a file }
 1:2  444       'c','C': COLLECTOR;    {data from kbrd}
 1:2  449       'd','D': DEFINER;      {input equations}
 1:2  454       'e','E': EDITOR;       {edit equation}
```

```
1:2 459        'g','G': GETEQUATIONS;{get eq's from disk}
1:2 463        'q','Q': QUIT:=true;  {back to system}
1:2 469        'r','R': iosub(true );{read model from file}
1:2 474        's','S': iosub(false);{save model in a file}
1:2 479        't','T': dump;        {write data and equations}
1:2 483        'v','V': SCANNER(vfmt);{evaluate expressions}
1:2 490        'w','W': DATAOUT;     {write data to disk}
1:2 494        'x','X': execute;     {execute expression}
1:2 498      end;
1:2 616      case ord(keybuff) of
1:2 623         3: device:='CONSOLE:';

1:2 640         4: device:='DATA:OUTPUT';
1:2 660         6: vfmt:=false;
1:2 666        16: device:='PRINTER:';
1:2 683        18: device:='REMOUT: ';
1:2 700        20: vfmt:=true;
1:2 706      end;
1:2 750      if EXPR='' then QUIT:=true;
1:1 764    until QUIT;
1:1 769    close(datafile,lock);
1:0 778 end.
```

```
 1:D    1  {$L DATA:UTILS.LST}
 1:D    1  (*$S+*)
 1:D    1 UNIT UTILS; INTRINSIC CODE 25 DATA 26;
 1:D    1 INTERFACE
 1:D    1 {$  }
 1:D    1
 2:D    3   FUNCTION SIN(X:REAL):REAL;
 3:D    3   FUNCTION COS(X:REAL):REAL;
 4:D    3   FUNCTION EXP(X:REAL):REAL;
 5:D    3   FUNCTION ATAN(X:REAL):REAL;
 6:D    3   FUNCTION LN(X:REAL):REAL;
 7:D    3   FUNCTION LOG(X:REAL):REAL;
 8:D    3   FUNCTION SQRT(X:REAL):REAL;
 8:D    5
 1:D    5 IMPLEMENTATION
 1:D    1     (* 6502 compatible *)
 1:D    1
 1:D    1  USES TRANSCEND;
 2:D    1  procedure poke(addr,data:integer);
 3:D    3  function peek(addr:integer):integer;
 4:D    3  function keybuff:char;
 5:D    3  function priority(ITEM:string;TYP:integer):integer;
 6:D    1  procedure getaline(var line:string;echo:boolean);
 7:D    3  function pwr(x,y:real):real;
 8:D    3  function ctb(STR:string):real;
 9:D    1  procedure get_item(EXPR:string;var INDX:integer;
 9:D    3                        DLMTR:string;
 9:D    4                        ITEMLEN:integer; var ITEM:string);
10:D    1  procedure message(MSG:string);
11:D    1  procedure pause;
 1:D    1  IMPLEMENTATION
 1:D    1
 1:D    1  type
 1:D    1    byte  = packed array[0..1] of 0..256;
 1:D    1    dirty= record
 1:D    1             case boolean of
 1:D    1               true:(int:integer);
 1:D    1               false:(ptr:^byte);
 1:D    1             end;
 1:D    1  var
 1:D    1    trick : dirty;
12:D    1    procedure check(var data:integer);
12:D    2    forward;
12:D    2
 2:D    1    procedure poke{(addr,data:integer)};
 2:0    0    begin
 2:1    0      check(data);
 2:1    4      trick.int:=addr;
 2:1    8      trick.ptr^[0]:=data;
 2:0   24    end;
 2:0   36
 3:D    3     function peek{(addr:integer):integer};
 3:0    0     begin
 3:1    0       trick.int:=addr;
 3:1    4       peek:=trick.ptr^[0];
 3:0   16     end;
 3:0   28
12:D    1     procedure check{(var data:integer)};
12:0    0     begin
12:1    0       data:=abs(data mod 256);
12:0    9     end;
12:0   22
 4:D    3     function keybuff{:char};
 4:D    3     var I:integer;
 4:0    0     begin
 4:1    0       I:=trunc(peek(-16384)/256);
 4:1   18       if I < 0 then I:=127+I;
 4:1   28       keybuff:=chr(I);
 4:0   31     end;
 4:0   44
 5:D    3   function priority(*(ITEM:string;TYP:integer):integer*);
 5:0    0   begin
 5:1    0     if TYP=1 then begin
 5:3   10       if length(ITEM)=0
 5:3   14       then begin priority:=-1;exit(priority); end;
 5:3   26       if (ITEM='+')or(ITEM='-') then priority:=1
 5:3   39       else if (ITEM='*')or(ITEM='/') then priority:=3
 5:4   57       else if (ITEM='@')              then priority:=6
 5:5   69       else if (ITEM='(')              then priority:=9
 5:6   81       else if (ITEM=')')              then priority:=0
 5:7   93       else                                 priority:=7;
 5:2  101     end;
 5:1  101     if TYP=2 then begin
 5:3  106       if length(ITEM)=0
 5:3  110       then begin priority:=-1;exit(priority); end;
 5:3  122       if (ITEM='+')or(ITEM='-') then priority:=2
 5:3  135       else if (ITEM='*')or(ITEM='/') then priority:=4
 5:4  153       else if (ITEM='@')              then priority:=5
 5:5  165       else if (ITEM='(')              then priority:=0
 5:6  177       else                                 priority:=8;
 5:2  185     end;
 5:0  185   end;
 5:0  198
 5:0  198
```

```
6:D    1 procedure getaline{(var line:string;echo:boolean};var
6:D    3   ch:string;
6:0    0 begin
6:1    0   ch:=' ';line:='';
6:1   11   repeat
6:2   11     get(keyboard);ch[1]:=keyboard^;
6:2   30     if echo then write(ch[1]);
6:2   47     if ord(ch[1])=27
6:2   52     then begin line:=ch; exit(getaline); end;
6:2   65     if ord(ch[1])<>32 then begin
6:4   74       if ord(ch[1]) = 8 then
6:5   83         if length(line) > 0 then begin
6:7   90           if not echo
6:7   90           then write(ch[1]);write(' ',ch[1]);
6:7  132           delete(line,length(line),1)
6:6  140         end else
6:4  142       else if ch[1]='?'
6:5  149       then begin line:='?';ch[1]:=chr(32);end
6:5  163         else if length(line)=0 then line:=ch
6:6  173              else insert(ch,line,length(line)+1);
6:3  191     end;
6:1  191   until ord(ch[1])=32;
6:0  200 end;
6:0  214
7:D    3  function pwr(*(x,y:real):real*);
7:D    7  (* raise x to the power of y *)
7:0    0  begin
7:1    0    pwr:=exp(ln(x)*y);
7:0   23  end;
7:0   36
8:D    3  function ctb(*(STR:string):real*);
8:D   45  var
8:D   45    E,I,J,K,SIGN:integer;
8:D   50    SUM  :real;
8:0    0  begin
8:1    0    SUM:=0; I:=0; J:=0; E:=0;K:=1;SIGN:=1;
8:1   26    if STR[1]='-' then begin SIGN:=-1; K:=2; end;
8:1   42    if STR[1]='+' then K:=2;
8:1   54    for I:=length(STR) downto K do begin
8:3   71      if STR[I]='.' then E:=length(STR)-I
8:3   85      else begin
8:5   92        SUM:=SUM+(ord(STR[I])-48)*pwroften(J);
8:5  115        J:=J+1;
8:4  121      end;
8:2  121    end;
8:1  129    ctb:=SIGN*SUM/pwroften(E);
8:0  146  end;
8:0  160
```

```
 1:D   1  procedure get_item(*(EXPR:string;var INDX:integer;
 1:D  88                      DLMTR:string;ITEMLEN:integer;
 9:D  88                      var ITEM:string)*);
 9:D  88  var
 9:D  88    TEMP1:string;
 9:D 129    cnst:boolean;
 9:0   0  begin
 9:1   0    ITEM:='';TEMP1:='';
 9:1  23    if EXPR[INDX]=DLMTR[1]
 9:1  33    then begin INDX:=INDX+1; exit(get_item); end;
 9:1  47    if INDX <= length(EXPR) then begin
 9:3  56      TEMP1:=copy(EXPR,INDX,1);
 9:3  74      INDX:=INDX+1;
 9:3  80      ITEM:=concat(ITEM,TEMP1);
 9:3 109      cnst:=true;
 9:3 113      while (priority(TEMP1,1) = 7) and
 9:3 122            (INDX <= length(EXPR)) do begin
 9:5 132        TEMP1:=copy(EXPR,INDX,1);
 9:5 150        if (priority(TEMP1,1) =7) then begin
 9:7 161          if (TEMP1 <> DLMTR) then begin
 9:9 169            if not(TEMP1[1] in
 9:9 174            ['0','1','2','3','4','5',
 9:9 174            '6','7','8','9','.'] )
 9:9 186            then cnst:=false;
 9:9 193            ITEM:=concat(ITEM,TEMP1)
 9:8 220          end
 9:7 222          else TEMP1:='';
 9:7 231          INDX:=INDX+1;
 9:6 237        end;
 9:4 237      end;
 9:3 239      if not cnst then
 9:4 245      if length(ITEM) > ITEMLEN
 9:4 248      then ITEM:=copy(ITEM,1,ITEMLEN);
 9:2 267    end;
 9:0 267  end;
 9:0 284
10:D   1  procedure message(*(MSG:string)*);
10:0   0  begin
10:1   0    gotoxy(0,0);write(chr(11),MSG);
10:0  31  end;
10:0  44
11:D   1 procedure pause;
11:0   0 begin
11:1   0   write(CHR(25),' Hit any key to continue.....');
11:1  51   get(keyboard);
11:1  59   keyboard^:=chr(0);
11:1  66   write(chr(12));
11:0  76 end;
11:0  88
 1:0   0  BEGIN end.
 1:D      1 {$L DATA:POLISH.LST}
 1:D      1 (*$S+*)
 1:D      1 UNIT POLISH; INTRINSIC CODE 23 DATA 24;
 1:D      1 INTERFACE
 1:D      1 {$  }
 1:D      1
 2:D      3   FUNCTION SIN(X:REAL):REAL;
 3:D      3   FUNCTION COS(X:REAL):REAL;
 4:D      3   FUNCTION EXP(X:REAL):REAL;
 5:D      3   FUNCTION ATAN(X:REAL):REAL;
 6:D      3   FUNCTION LN(X:REAL):REAL;
 7:D      3   FUNCTION LOG(X:REAL):REAL;
 8:D      3   FUNCTION SQRT(X:REAL):REAL;
 8:D      5
 1:D      5 IMPLEMENTATION
 1:D      1     (* 6502 compatible *)
 1:D      1
 1:D      1
 1:D      1  USES TRANSCEND;
 2:D      1  procedure poke(addr,data:integer);
 3:D      3  function peek(addr:integer):integer;
 4:D      3  function keybuff:char;
 5:D      3  function priority(ITEM:string;TYP:integer):integer;
 6:D      1  procedure getaline(var line:string;echo:boolean);
 7:D      3  function pwr(x,y:real):real;
 8:D      3  function ctb(STR:string):real;
 9:D      1  procedure get_item(EXPR:string;var INDX:integer; DLMTR:
 9:D      4                      ITEMLEN:integer; var ITEM:string);
10:D      1  procedure message(MSG:string);
11:D      1  procedure pause;
 1:D      1 USES TRANSCEND,UTILS;
 1:D      1  const
 1:D      1    stringsize =80;
 1:D      1    table_size =50;
 1:D      1  type
 1:D      1    symb_rec  = record
 1:D      1                  symb: string;
 1:D      1                  predictor:string;
 1:D      1                  expr  :string;
 1:D      1                  actual:real;
 1:D      1                  tol   :real;
 1:D      1                  fmt   :ARRAY[1..3] of integer;
 1:D      1                  flag  :ARRAY[1..2] of boolean;
 1:D      1                end;
 1:D      1    symb_table=array[1..table_size] of symb_rec;
 1:D      1 var
 1:D      1    TABLE:symb_table ;
 1:D   6601    TEMPSTR,delim:string;
 1:D   6683    namelen:integer;
 2:D      1  procedure init;
 3:D      1  procedure tablein(TARG,EXPR,POLEXPR:string);
 4:D      3  function valof(ITEM:string;var ERR:boolean):real;
 5:D      1  procedure intopost(EXPR:string;var INDX:integer;
 5:D      3              DLMT:string;VNAMLEN:integer;
 5:D      5              var ITEM1:string;var POLSTR:string);
 6:D      1  procedure evaluate(POLSTR:string;DLMT:string;
 6:D      3                     VNAMLEN:integer;VAR VAL:real;
 6:D      5                     VAR ERR:integer);
 6:D     88
 7:D      1  procedure parse(EXPR:string);
 8:D      1  procedure editor;
 9:D      1  procedure definer;
10:D      1  procedure collector;
 1:D      1  IMPLEMENTATION
 1:D   6684
 4:D      3  function valof(*(ITEM:string;var ERR:boolean):real*);
 4:D     46  var
 4:D     46    OKNUMCHARS:set of char;
 4:D     62    I:integer;
 4:0      0  begin
 4:1      0    OKNUMCHARS:=
 4:1      7        ['1','2','3','4','5','6','7','8','9','0','.'];
 4:1     23    valof:=0;
 4:1     29    ERR:=false;
 4:1     32    for i:=1 to length(ITEM) do
 4:2     48    if not(ITEM[I] in OKNUMCHARS) then ERR:=true;
 4:1     74    if not ERR
 4:1     74    then begin valof:=ctb(ITEM); exit(valof); end;
 4:1     94    I:=1;
 4:1     97    while (I<=table_size) do begin
 4:3    103      if (TABLE[I].symb=ITEM) then  begin
 4:5    122        ERR:=false;
 4:5    125        valof:=TABLE[I].actual;
 4:5    146        exit(valof);
 4:4    150      end;
 4:3    150      I:=I+1;
 4:2    156    end;
 4:0    158  end;
 4:0    174
 1:D      1  procedure intopost(*(EXPR:string;var INDX:integer;
 1:D     89              DLMT:string;VNAMLEN:integer;
 5:D     89              var ITEM1:string;var POLSTR:string)*);
 5:D     89  const
 5:D     89    prio_alpha=7;
 5:D     89  var
 5:D     89    ITEM2:string;
 5:D    130    PRIO2:integer;
 5:D    131
```

```
11:D     1 procedure build_polish(ITEM:string; var POLSTR:string);
11:0     0  begin
11:1     0    if (length(POLSTR)+length(ITEM))>stringsize
11:1    13    then message(' Expression too long ')
11:1    41    else POLSTR:=concat(POLSTR,ITEM);
11:1    71    if (length(POLSTR) < stringsize) and
11:1    76       (priority(ITEM,1)=prio_alpha)
11:1    86    then POLSTR:=concat(POLSTR,DLMT)
11:1   113    else if length(POLSTR)>=stringsize
11:2   120    then message(' Expression too long.');
11:0   151  end;
11:0   164
 5:0     0  begin  {into post}
 5:1     0    if (INDX>0)and(INDX<=length(EXPR))
 5:1    21    then get_item(EXPR,INDX,DLMT,VNAMLEN,ITEM2)
 5:1    32    else ITEM2:=')';
 5:1    42    while priority(ITEM2,1)>priority(ITEM1,2) do
 5:2    60         intopost(EXPR,INDX,DLMT,VNAMLEN,ITEM2,POLSTR);
 5:1    73    if priority(ITEM2,1)<priority(ITEM1,2)then begin
 5:3    91      build_polish(ITEM1,POLSTR);
 5:3    95      ITEM1:=ITEM2;
 5:2   100    end
 5:1   100    else begin {right bracket discovered}
 5:3   102      if indx<=length(EXPR)
 5:3   108      then get_item(EXPR,INDX,DLMT,VNAMLEN,ITEM2);
 5:3   122      ITEM1:=ITEM2;
 5:2   127    end;
 5:0   127  end;
 5:0   142
 1:D     1  procedure evaluate(*(POLSTR:string;DLMT:string;
 1:D    88                     VNAMLEN:integer;VAR VAL:real;
 6:D    88                     VAR ERR:integer)*);
 6:D    88  const
 6:D    88    stack_size=  20;
 6:D    88    add   ='+';
 6:D    88    sub   ='-';
 6:D    88    mul   ='*';
 6:D    88    dyv   ='/';
 6:D    88    xpo   ='@';
 6:D    88  var
 6:D    88    STAK       : array[ 1 .. stack_size ] of real;
 6:D   128    ITEM       : string ;
 6:D   169    DONE,ERROR: boolean;
 6:D   171    INDX,J     : integer;
 6:D   173    OPER       : char;
 6:D   174
 6:0     0  begin   (* code for procedure 'evaluate' *)
 6:1     0    DONE:=false; INDX:=1;
 6:1    18    J:=0;ERR:=0;
 6:1    25    while (DONE=false) and (INDX <= length(POLSTR))
```

```
6:1    39    do begin
6:3    42      get_item(POLSTR,INDX,DLMT,VNAMLEN,ITEM);
6:3    56      if (priority(ITEM,1)>0) and
6:3    67         (priority(ITEM,1) < 7)
6:3    78      then begin
6:5    81        if J < 2 then begin
6:7    88          ERR:=1;  {Stack underflow}
6:7    91          DONE:=true;
6:6    95        end
6:5    95        else begin
6:7    97          OPER:=ITEM[1];
6:7   106          case OPER of
6:7   111            add:STAK[J-1]:=STAK[J-1]+STAK[J];
6:7   160            sub:STAK[J-1]:=STAK[J-1]-STAK[J];
6:7   209            mul:STAK[J-1]:=STAK[J-1]*STAK[J];
6:7   258            dyv:if STAK[J]<>0
6:8   272                then STAK[J-1]:=STAK[J-1]/STAK[J]
6:8   320                else ERR:=4;
6:7   332            xpo:STAK[J-1]:=pwr(STAK[J-1],STAK[J]);
6:7   385          end;
6:7   438          J:=J-1;
6:6   446        end;
6:4   446      end
6:3   446      else if (priority(ITEM,1)=7) and
6:4   459              (ITEM <> DLMT) then begin
6:6   469        J:=J+1;
6:6   477        if J > stack_size then ERR:=2  {Stack overflow}
6:6   485        else begin
6:8   489          STAK[J]:=valof(ITEM,ERROR);
6:8   513          if ERROR then ERR:=3;  { UNDEFINED }
6:7   521        end;
6:5   521      end;
6:2   521    end; { while loop }
6:1   523    if ERROR then VAL:=0.0 else VAL:=STAK[1];
6:0   555  end;
6:0   580
2:D     1 procedure init;
2:D     1 var
2:D     1   i:integer;
2:0     0 begin
2:1     0   for I:=1 to table_size do begin
2:3    11     TABLE[I].symb:='';
2:3    28     TABLE[I].actual:=0;
2:3    46     TABLE[I].tol    :=0;
2:3    64     TABLE[I].predictor:='';
2:3    83     TABLE[I].expr   :='';
2:3   102     TABLE[I].flag[1]   :=false;
2:3   127     TABLE[I].flag[2]   :=false;
2:2   152   end;
2:1   159   DELIM:='!';
2:1   166   NAMELEN:=5;
2:0   171 end;  (* end init *)
2:0   188
```

```
3:D     1 procedure tablein(*(TARG,EXPR,POLEXPR:string)*);
3:D   127 var
3:D   127   I,FREE:integer;
3:D   129   done:boolean;
3:0     0 begin
3:1     0   DONE:=false;
3:1    19   I:=1;
3:1    23   FREE:=0;
3:1    26   repeat
3:2    26     if (TABLE[I].symb='') and (FREE=0) then FREE:=I
3:2    52     else if TABLE[I].symb=TARG  then begin
3:5    79       TABLE[I].expr:=EXPR;
3:5    99       TABLE[I].predictor:=POLEXPR;
3:5   119       DONE:=true;
3:5   123       message(concat(' Definition for "',
3:5   157                             TARG,'" has been replaced.'));
3:4   199     end;
3:2   199     I:=I+1;
3:1   207   until DONE or (I>table_size);
3:1   218   if (not DONE) then begin
3:3   224     if (FREE<>0) then begin
3:5   230       TABLE[FREE].symb:=TARG;
3:5   247       TABLE[FREE].expr:=EXPR;
3:5   266       TABLE[FREE].predictor:=POLEXPR;
3:4   285     end
3:3   285     else message(' Symbol table full!');
3:2   312   end
3:0   312 end;  (* end of tablein *)
3:0   328
7:D     1 procedure parse{(EXPR:string)};
7:D    43 var
7:D    43   INDX    :integer;
7:D    44   TARG,POLSTR:string;
7:0     0 begin
7:1     0   INDX:=pos('=',EXPR);
7:1    18   if INDX >0 then begin
7:3    24     if INDX>NAMELEN then TARG:=copy(EXPR,1,NAMELEN)
7:3    49                     else TARG:=copy(EXPR,1,INDX-1);
7:3    71     POLSTR:='';
7:3    78     TEMPSTR:=')';
7:3    85     INDX:=INDX+1;{skip '=' }
7:3    91     EXPR:=concat(EXPR,TEMPSTR);
7:3   120     TEMPSTR:='(';
7:3   127     intopost(EXPR,INDX,DELIM,NAMELEN,TEMPSTR,POLSTR);
7:3   147     EXPR:=copy(EXPR,1,length(EXPR)-1);{remove ')' again }
7:3   167     tablein(TARG,EXPR,POLSTR);
7:2   175   end
7:1   175   else begin
7:3   177     gotoxy(0,0);
7:3   182     write(chr(11),
7:3   192     ' Bad expression     "=" missing.',chr(7));
7:2   246   end;
7:0   246 end;
7:0   260
```

```
 8:D     1 procedure editor;
 8:D     1 const
 8:D     1   xpos=2;
 8:D     1 var
 8:D     1  I,J    :integer;
 8:D     3  INSRT :boolean;
 8:D     4
12:D     1 procedure help(selector:integer);
12:0     0 begin
12:1     0   write(chr(12));gotoxy(0,1);
12:1    15   case selector of
12:1    18   1:writeln(
12:2    18     'Enter the NUMBER of the EXPRESSION to be edited.');
12:1    88   2:begin
12:3    88     writeln('The following characters ');
12:3   168     writeln('may not be enterred as text:');
12:3   216     writeln;
12:3   224     writeln('    KEY                      FUNCTION');
12:3   278     writeln;
12:3   286     writeln('    ESC               leave the EDITOR');
12:3   341     writeln(' CTRL D               delete the character');
12:3   419     writeln(' CTRL E               del. to end of line');
12:3   491     writeln(' CTRL I               insert toggle');
12:3   563     writeln(' (left arrow)         backspace');
12:2   611     end;
12:1   613   end;{case}
12:1   624   pause;
12:0   627 end;
12:0   644
13:D     1 procedure prompt;
13:0     0 begin
13:1     0   repeat
13:2     0     gotoxy(0,1);
13:2     5     write(chr(12),'Which item (enter item #) ?');
13:2    54     getaline(TEMPSTR,true);
13:2    62     if TEMPSTR='' then exit(editor);
13:2    77     if ord(TEMPSTR[1]) = 27 then exit(editor);
13:2    92     if TEMPSTR='?' then help(1)
13:2   102     else J:=trunc(ctb(TEMPSTR));
13:1   120   until TEMPSTR <> '?';
13:0   129 end;
13:0   144
14:D     1 procedure delchar;
14:0     0 begin
14:1     0   TABLE[J].expr:=concat(copy(TABLE[J].expr,1,I-1),
14:1    58   copy(TABLE[J].expr,I+1,length(TABLE[J].expr)-I));
14:1   120   gotoxy(xpos+I,3);
14:1   129   write(chr(11),
14:1   139     copy(TABLE[J].expr,I,length(TABLE[J].expr)-I+1));
14:0   198 end;
14:0   210
```

```
8:0     0 begin
8:1     0   prompt; {for item to edit}
8:1     2   if J <= table_size then begin
8:3     7     I:=length(TABLE[J].symb)+2;
8:3    25     if length(TABLE[J].symb) <> 0 then begin
8:5    43       gotoxy(xpos+1,3);write(chr(11),TABLE[J].expr);
8:5    83       INSRT:=false;
8:5    86       repeat
8:6    86         gotoxy(1,19);
8:6    91         if INSRT
8:6    91         then write('insert mode') else write(chr(11));
8:6   129         gotoxy(xpos+I,3);
8:6   136         get(keyboard);
8:6   144         if ord(keyboard^)=27 then exit(editor)
8:6   158         else if keyboard^='?' then help(2)
8:7   171         else if ord(keyboard^) =21 then I:=I+1
8:8   186         else if ord(keyboard^)=8
8:9   198              then begin I:=I-1; if I<1 then I:=1; end
8:9   215         else if ord(keyboard^)=4 then delchar
8:0   227         else if ord(keyboard^)=5 then begin
8:3   241              if I = 1 then TABLE[J].expr:='' else
8:4   267              TABLE[J].expr:=copy(TABLE[J].expr,1,I-1);
8:3   308              gotoxy(xpos+I,3);write(chr(11));
8:2   325            end
8:1   325         else if ord(keyboard^)=9
8:2   333              then INSRT:=(not INSRT)     { ctrl I }
8:2   339         else if ord(keyboard^) <> 32 then  begin
8:5   353           if INSRT or (I > length(TABLE[J].expr))
8:5   372           then begin
8:7   375             if I > length(TABLE[J].expr) then begin
8:9   395               I:=length(TABLE[J].expr);
8:9   413               TABLE[J].expr:=
8:9   427                    concat(copy(TABLE[J].expr,1,I),
8:9   463                           copy(TABLE[J].expr,1,1));
8:9   498               I:=I+1;
8:8   503             end
8:7   503             else begin
8:9   505               TABLE[J].expr:=
8:9   519                       concat(copy(TABLE[J].expr,1,I),
8:9   555                       copy(TABLE[J].expr,I,
8:9   573                       length(TABLE[J].expr)-I+1));
8:8   609             end;
8:7   609             TABLE[J].expr[I]:=keyboard^;
8:7   632             gotoxy(xpos+I,3);
8:7   639             write(copy(TABLE[J].expr,I,
8:7   659                   length(TABLE[J].expr)-I+1));
8:6   690           end
```

```
 8:5   690              else begin
 8:7   692                TABLE[J].expr[I]:=keyboard^;
 8:7   715                gotoxy(xpos+I,3);put(keyboard);
 8:6   730              end;
 8:5   730              I:=I+1;
 8:4   735            end;
 8:6   735            if I < 1 then I:=1;
 8:5   743          until (ord(keyboard^) = 32);
 8:5   753          if length(TABLE[J].expr)>0 then
 8:6   773          if TABLE[J].expr[length(TABLE[J].expr)] =' '
 8:6   805          then TABLE[J].expr:=
 8:7   823             copy(TABLE[J].expr,1,length(TABLE[J].expr)-1);
 8:5   865          parse(TABLE[J].expr);
 8:5   881          if pos(concat(TABLE[J].symb,'='),TABLE[J].expr)=0
 8:5   933          then TABLE[J].symb:='';  { item has been deleted}
 8:4   954        end
 8:3   954        else begin gotoxy(0,0);
 8:5   961               write(chr(11),'item ',j,' is empty!');end;
 8:2  1020      end
 8:1  1020      else begin
 8:3  1022        gotoxy(0,0);
 8:3  1027        write(chr(11),
 8:3  1037             'Item # must be between 1 and ',table_size);
 8:2  1088      end;
 8:0  1088  end;
 8:0  1124
 9:D     1  procedure definer;
 9:D     1  var
 9:D     1    QUIT:boolean;
 9:D     2    EXPR:string;
 9:D    43    I,J,INDX:integer;
 9:D    46
15:D     1  procedure help1;
15:0     0  begin
15:1     0    write(chr(12));gotoxy(0,1);
15:1    15    writeln('The system is expecting a PREDICTOR');
15:1    89    writeln('equation of the following form:');
15:1   148    writeln;
15:1   156    writeln('VARIABLE NAME followed by "=", followed by');
15:1   229    writeln('a fully parenthesized expression including');
15:1   307    writeln('OPERATORS (+,-,*,/,@), and CONSTANTS.');
15:1   364    pause;
15:0   367  end;
15:0   380
16:D     1  procedur help2;
16:0     0  begin
16:1     0    write(chr(12));gotoxy(0,1);
16:1    15    writeln('The MEASURED value is considered correct when');
16:1    92    writeln('within  PREDICTED VALUE + or - TOLERANCE');
16:1   172    pause;
16:0   175  end;
16:0   188
```

```
17:D     1 procedure help3;
17:0     0 begin
17:1     0   write(chr(12));gotoxy(0,1);
17:1    15   writeln('The Format for each VARIABLE.
17:1    93   writeln('is a set of three numbers:');
17:1   127   writeln('    1: Column at which the variable begins');
17:1   195   writeln('    2: The total length of this variables');
17:1   273   writeln('    3: The number of decimal digits.');
17:1   355   writeln('       overridden by a "." in the data');
17:1   430   writeln;
17:1   438   writeln(' Enter three numbers, separated by commas');
17:1   517   pause;
17:0   520 end;
17:0   532
18:D     1 procedure help4;
18:0     0 begin
18:1     0   write(chr(12));gotoxy(0,1);
18:1    15   writeln(' To REORDER two expressions type the item no.');
18:1    95   writeln(' that is to be moved, comma,');
18:1   173   writeln(' the number of the destination location .');
18:1   251   writeln(' EXAMPLE: "8,4" moves item #8 to position 4.');
18:1   332   writeln(' Items 4-7 are moved to locations 5-8.');
18:1   413 pause;
18:0   416 end;
18:0   428
19:D     1 procedure reorder;
19:D     1 var
19:D     1   S,D,INC,J:integer;
19:D     5   BUFF :symb_rec;
19:0     0 begin
19:1     0   write(chr(12)); gotoxy(0,1);
19:1    15   write('ENTER.. SOURCE INDEX,DESTINATION INDEX ?');
19:1    67   S:=0; D:=0;
19:1    73   getaline(TEMPSTR,true);
19:1    81   if TEMPSTR='?' then help4
19:1    90   else if length(TEMPSTR)>0 then
19:3   104        if (ord(TEMPSTR[1])=27) then exit(reorder)
19:3   119        else begin
19:5   121          if pos(',',TEMPSTR) <> 0 then begin
19:7   138            INC:=pos(',',TEMPSTR);
19:7   153            S:=trunc(ctb(copy(TEMPSTR,1,INC-1)));
19:7   179            INC:=INC+1;  {skip ','}
19:7   184            D:=trunc(ctb(copy(TEMPSTR,INC,
19:7   192                 length(TEMPSTR)-INC+1)));
19:6   217          end
19:5   217          else help4;
19:4   221        end;
```

```
19:1   221    if (S=D)or(S<1)or(S>table_size)or(D<1)or
19:1   236       (D>table_size)
19:1   239    then exit(reorder);
19:1   246    if S < D then INC:=1 else  INC:=-1;
19:1   260    BUFF:=TABLE[S];
19:1   277    repeat TABLE[S]:=TABLE[S+INC]; S:=S+INC; until S=D;
19:1   316    TABLE[D]:=BUFF;
19:0   333  end;
19:0   348
20:D     1  procedure scale;
20:D     1  var
20:D     1    K:integer;
20:D     2    x:real;
20:D     4    ERR:boolean;
20:0     0  begin
20:1     0    ERR:=false;
20:1     3    write(chr(12)); gotoxy(0,1);
20:1    18    write(
20:1    18      'Enter OPERATOR (+,-,*,/) followed by a constant.');
20:1    78    TEMPSTR:='';
20:1    87    getaline(TEMPSTR,true);
20:1    95    if length(TEMPSTR) > 1 then begin
20:3   105      x:=valof(copy(TEMPSTR,2,length(TEMPSTR)-1),ERR);
20:3   135      if ERR then writeln(' ERROR') else
20:4   166      for K:=1 to table_size do if TABLE[K].symb <> ''
20:5   189                   then case TEMPSTR[1] of
20:6   205                        '+':TABLE[K].tol:=TABLE[K].tol + x;
20:6   244                        '-':TABLE[K].tol:=TABLE[K].tol - x;
20:6   283                        '*':TABLE[K].tol:=TABLE[K].tol * x;
20:6   322                        '/':TABLE[K].tol:=TABLE[K].tol / x;
20:6   361                      end;
20:6   387
20:2   387    end;
20:0   387  end;
20:0   414
20:0   414
 9:0     0  begin
 9:1     0    QUIT:=false;
 9:1     3    TEMPSTR:=' ';
 9:1    10    write(chr(12)); gotoxy(0,1);
 9:1    25    while not QUIT do begin
 9:3    29      TEMPSTR:=' ';
 9:3    36      gotoxy(0,1);
 9:3    41      write(chr(11),
 9:3    51        'E(xpressions, T(olerances, F(ormats  or Q(uit');
 9:3   108      get(keyboard); TEMPSTR[1]:=keyboard^;
 9:3   129      writeln;
```

```
9:3   137      case TEMPSTR[1] of
9:3   146       'e','E': begin
9:5   146         I:=1;
9:5   149         while not QUIT do begin
9:7   153           gotoxy(0,I);
9:7   159           write(chr(11),'Enter expression =>');
9:7   200           EXPR:=''; TEMPSTR:=' ';
9:7   214           get(keyboard); TEMPSTR[1]:=keyboard^;
9:7   235           write(TEMPSTR[1]);
9:7   251           if TEMPSTR[1] = '?' then help1
9:7   262           else if TEMPSTR[1] = chr(18) then reorder
9:8   277           else while TEMPSTR[1] <> chr(32) do begin
9:1   292             if TEMPSTR[1] = chr(5)
9:1   300             then begin
9:3   303               gotoxy(19,I);write(chr(11));EXPR:='';
9:2   326             end
9:1   326             else if TEMPSTR[1] = chr(8) {backspace}
9:2   336                  then begin
9:4   339                    EXPR:=copy(EXPR,1,length(EXPR)-1);
9:4   359                    write(' ',chr(8));
9:3   379                  end
9:2   379             else if ord(TEMPSTR[1]) > 31
9:3   388                  then EXPR:=concat(EXPR,TEMPSTR);
9:1   421             get(keyboard);
9:1   429             TEMPSTR[1]:=keyboard^;write(TEMPSTR[1]);
9:0   458           end;  {end while loop }
9:7   460           if length(EXPR) = 0 then QUIT:=true
9:7   468           else begin
9:9   473             parse(EXPR);
9:9   477             I:=I+1;
9:8   483           end;
9:6   483         end; {end while loop}
9:5   485         QUIT:=false;
9:4   488       end;
9:3   490       't','T':
9:4   490         for I:=1 to table_size do begin
9:6   503           if TABLE[I].symb <> '' then begin
9:8   523             repeat
9:9   523               write('Tolerance for ',TABLE[I].symb,' =');
9:9   585               getaline(TEMPSTR,true);
9:9   593               if TEMPSTR='?' then help2
9:9   602               else if ord(TEMPSTR[1])=19
9:0   613                    then begin scale; exit(definer); end
9:0   623               else if length(TEMPSTR)>0 then
9:2   635                    if (ord(TEMPSTR[1])=27)
9:2   644                    then exit(definer)
9:2   650                    else TABLE[I].tol:=ctb(TEMPSTR);
9:8   678             until TEMPSTR <> '?';
9:7   687           end;
9:6   687           writeln;
9:5   695         end;
```

```
 9:3    705        'f','F':
 9:4    705          for I:=1 to table_size do begin
 9:6    718            if TABLE[I].symb <> '' then begin
 9:8    738              repeat
 9:9    738                write('Format for ',TABLE[I].symb,' =');
 9:9    797                getaline(TEMPSTR,true);
 9:9    805                if TEMPSTR='?' then help3
 9:9    814                else if length(TEMPSTR)>0 then
 9:1    828                     if (ord(TEMPSTR[1])=27)
 9:1    837                     then exit(definer)
 9:1    843                     else begin
 9:3    845                       J:=1; TEMPSTR:=concat(TEMPSTR,',');
 9:3    879                       while(pos(',',TEMPSTR) <>0) and
 9:3    894                             (J < 4) do begin
 9:5    901                         INDX:=pos(',',TEMPSTR);
 9:5    916                         TABLE[I].fmt[J]:=
 9:5    940                         trunc(ctb(copy(TEMPSTR,1,INDX-1)));
 9:5    964                         TEMPSTR:=copy(TEMPSTR,INDX+1,
 9:5    978                                 length(TEMPSTR)-INDX);
 9:5    994                         J:=J+1;
 9:4   1000                       end;
 9:2   1002                     end;
 9:8   1002              until TEMPSTR <> '?';
 9:7   1011            end;
 9:6   1011            writeln;
 9:5   1019          end;
 9:3   1029        'q','Q':QUIT:=true;
 9:3   1034        end; {case}
 9:2   1138        end;(* end while loop *)
 9:0   1140      end;  (* end definer   *)
 9:0   1196
10:D      1 procedure collector;
10:D      1 var
10:D      1   I:integer;
10:D      2
21:D      1 procedure help;
21:0      0 begin
21:1      0   write(chr(12));gotoxy(0,1);
21:1     15   writeln(
21:1     15     'Enter the OBSERVED value for the given variable.');
21:1     83   pause;
21:0     86 end;
21:0     98
```

```
10:0      0 begin
10:1      0   gotoxy(0,1);
10:1      5   for I:=1 to table_size do begin
10:3     16     if TABLE[I].symb <> '' then begin
10:5     35       repeat
10:6     35         write(chr(11),TABLE[I].symb:-10,' ?');
10:6     81         getaline(TEMPSTR,true);
10:6     89         if TEMPSTR='?' then help
10:6     98         else if length(TEMPSTR)>0 then
10:8    112              if ord(TEMPSTR[1]) = 27
10:8    119              then exit(collector)
10:8    127              else TABLE[I].actual:=ctb(TEMPSTR);
10:5    154       until TEMPSTR <> '?';
10:4    163     end;
10:3    163     writeln;
10:2    171   end;
10:0    178 end;   (* end collector *)
 1:0      0 begin end.
```

## APPENDIX B.

### The Data Monitor System Software

This system has been designed to be run on small computers with limited memory and mass storage. It is written in Pascal for the University of California San Diego (UCSD) pascal system and has been developed using an APPLE II computer with 64K bytes of Ramdom Access Memory (RAM) and 1 diskII 5 1/4 inch floppy disk drive. Implementation dependencies were avoided whenever possible but the few that were necessary to include are pointed out in this discussion.

A hierarchy of sub systems, each invoked by a command selected from the level above it, constitutes the structure of the monitoring system. Zero or more commands are available at each level which allows further descent into the hierarchy, the changing of relevant parameters at the current level, or ascending the hierarchy back to the level which initiated the current invocation. If zero commands are available then that sub section of the system is not interactive. Its invocation is current until its task is finished, when it immediately returns to the level which called it. The system hierarchy is shown in Figure 3-1.

Any command at any level is selected by typing only one character on the keyboard. Once the key has been struck the command begins execution immediately, without the need for a carriage return. If the key that is struck does not correspond to one of the available commands then it is ignored. None of the commands accept or require parameters. If a command needs information before it begins processing (such as the need for a file name with the READ, SAVE and WRITE commands) then the user is prompted for this information immediately after the command has been issued.

The only command that is recognizable while the system is busy is the ESCape command. Issuing this command causes the system to interrupt what it is doing and return to the previous command level. The system is not interrupt driven so the recognition of a keystrike during execution of any piece of code must be done by polling. (A routine which looks directly at the keyboard buffer is included in the UTILS unit.) Most processes in the Data Monitoring System poll the keyboard periodically and respond to the ESCape key.

Much of the code for the system resides in the modified system library which must be present when compiling or running the system. All routines which do not require input or output with devices other than the console are placed in this library. Details for constructing such libraries can be found in the APPLE PASCAL USER's GUIDE (APPLE, 1978) and Tonkens (1982).

The main program and all procedures that access the disk are compiled and stored in a code file for execution via the UCSD system's RUN command. Therefore the files needed to run the system (besides the pascal system files) are the modified library file and the main code file.

## The Library

The code which is stored in the library is organized in the form of two UNITS, named POLISH and UTILS. The UTILS unit contains the utility routines which are used extensively throughout the system, by both POLISH and the main program (called DATAMONITOR). The keyboard handling routines, data conversion routines (such as CTB Which converts an ASCII string to binary) and other often used routines are compiled as a unit and stored in the library under the name UTILS. Since both DATAMONITOR and POLISH require these utility routines the UTILS unit must be compiled, stored in a library and made available to the other segments before they are compiled.

The POLISH unit contains procedures that are directly related to the main functions of the system. The first major routine is INTOPOST which converts expressions which are in INFIX notation to POSTFIX (or reverse POLISH) notation. This optimizes the subsequent evaluation of the expression by the other routines in the system. INTOPOST is a recursive expression parser that uses the recursion stack to construct a postfix expression which is equivalent to the INFIX expression input. During the conversion of an unparenthesized expression, the operator with the highest priority is evaluated before the others, and the left most operator before the others in the case of equal priorities. The inclusion of parentheses causes the evaluation of the expression within the parentheses to occur first. Tremblay and Sorenson (1976) provide an excellent discussion of INFIX to POSTFIX expression conversion.

The routine EVALUATE evaluates a POSTFIX expression in one pass where it would require many passes for all but the most simple INFIX expression.

The EDITOR is also contained in the POLISH unit. This editor is a visual editor which is restricted to operation on one line of the screen. It provides for cursor positioning, inserts, deletes and replacement of the characters which are contained in a line image buffer. The editor is used to modify expressions (in INFIX notation) and then replace them in the data structure in postfix notation. The main program requires the routines in the POLISH unit and therefore this unit must be compiled and stored in the library for use during compilation of the main program.

## The MAIN Program

The main program is compiled under the name DATAMONITOR. The codefile produced after it has successfully compiled may be executed by issuing the UCSD RUN command while the codefile is saved under the name SYSTEM.WRK.CODE or by the EXECUTE command when it is saved under another name. In order to compile it successfully the compiler must be informed that the library units POLISH and UTILS are required. This is done by the USES statement at the beginning of the program.

## Major Data Structure

The most important data structure in the system is TABLE which is an array of PASCAL records of type "SYMB-REC" (see program listing for details of SYMB-REC). The size of this array is the limiting factor for the number of predictor equations that can be defined in a model. The limit is currently set at fifty. Each of the elements in TABLE contains the dependent variable name, the complete predictor expression, the current value of the dependent variable (the measured value, not the predicted value) and the format specifications. Whenever a dependent variable name string is empty, that entire record is considered to be unused.

The SAVE procedure simply writes, onto the disk, each element in TABLE that is not empty. The READ procedure inputs the data that was stored by the SAVE command, placing the records in successive locations of TABLE. The READ command destroys the current contents of TABLE.

10 TABLE is a global variable and is used extensively by routines in POLISH and DATAMONITOR. It is necessary, therefore, that it be declared in POLISH as part of the UNIT INTERFACE so that it will also be known to DATAMONITOR.

## APPENDIX C.

### Alphabetic List of Commands

| COMMAND | KEY | LEVEL | ACTION |
|---|---|---|---|
| BATCH | B | Top | Begin batch processor. |
| COLLECT | C | Top | Begin Collecting data. |
| CONSOLE | C | TYPE, BATCH | Output to console. |
| CONSOLE | CTRL C | Top | Output to console. |
| DATA | D | TYPE | List current values. |
| DEFINE | D | Top | Begin model definition. |
| DELETE | D | EDIT | Delete char. at cursor. |
| EDIT | E | Top | Begin editing. |
| ERASE | CTRL E | EXPRESSION, EDIT | Erase to end of line. |
| EXPRESSION | E | DEFINE | Begin expression input. |
| FORMAT | F | DEFINE | Begin format input. |
| FULL | F | TYPE, BATCH | Full output listings. |
| FULL | CTRL F | Top | Full output listings. |
| GO | G | TYPE, BATCH | Begin processing. |
| INSERT | I | EDIT | Insert char at cursor. |
| MODEL | M | TYPE | Type out current model. |
| PRINTER | P | TYPE, BATCH | Output to printer. |
| PRINTER | CTRL P | Top | Output to printer. |
| QUIT | Q | Top, TYPE, BATCH | Back to previous level. |
| READ | R | Top | Input model from disk. |
| REORDER | R | EXPRESSION | Change expr. order. |
| SAVE | S | Top | Model to disk. |
| SCALE | S | TOLERANCE | Scale tol. by constant |
| TERSE | T | TYPE, BATCH | Abbrev. output listings. |
| TERSE | CTRL T | Top | Abbrev. output listings. |
| TOLERANCE | T | DEFINE | Begin tolerance input. |
| TYPE | T | Top | Print out Model or Data. |
| VERIFY | V | Top | Begin error scan. |
| WRITE | W | Top | Current data to disk. |
| XECUTE | X | Top | Evaluate any expression. |

## APPENDIX D. MEDIAN VALUES FOR DEMONSTRATION SUBJECTS.

| | VARIABLE | VALUE | | VARIABLE | VALUE |
|---|---|---|---|---|---|
| 1 | SUBJ | 7.000 | 18 | AGF | 27.200 |
| 2 | SEX | 2.000 | 19 | FAG | 24.100 |
| 3 | WT | 56.300 | 20 | WRG | 14.100 |
| 4 | HT | 167.100 | 21 | CHG | 82.200 |
| 5 | TPSF | 18.600 | 22 | WAG | 68.300 |
| 6 | SSSF | 10.900 | 23 | THG | 56.300 |
| 7 | ILSF | 14.000 | 24 | CAG | 32.400 |
| 8 | ABSF | 30.000 | 25 | ANG | 18.400 |
| 9 | THSF | 35.400 | 26 | BIAC | 36.800 |
| 10 | MCSF | 19.000 | 27 | BIIL | 28.900 |
| 11 | ACHT | 137.400 | 28 | TRCH | 26.300 |
| 12 | RAHT | 105.600 | 29 | FOOT | 23.300 |
| 13 | STHT | 81.200 | 30 | HUM | 5.930 |
| 14 | DAHT | 63.300 | 31 | FEM | 8.880 |
| 15 | TIHT | 44.500 | 32 | SITHT | 87.400 |
| 16 | SPHT | 93.000 | 33 | APCH | 17.000 |
| 17 | AGR | 26.100 | 34 | HDG | 55.600 |

MEDIAN VALUES FOR SUBJECT NUMBER 7

| | VARIABLE | VALUE | | VARIABLE | VALUE |
|---|---|---|---|---|---|
| 1 | SUBJ | 8.000 | 18 | AGF | 26.400 |
| 2 | SEX | 2.000 | 19 | FAG | 24.400 |
| 3 | WT | 57.800 | 20 | WRG | 14.700 |
| 4 | HT | 165.900 | 21 | CHG | 84.700 |
| 5 | TPSF | 18.500 | 22 | WAG | 65.700 |
| 6 | SSSF | 11.000 | 23 | THG | 56.400 |
| 7 | ILSF | 14.600 | 24 | CAG | 36.800 |
| 8 | ABSF | 30.000 | 25 | ANG | 21.000 |
| 9 | THSF | 36.000 | 26 | BIAC | 36.900 |
| 10 | MCSF | 25.600 | 27 | BIIL | 27.500 |
| 11 | ACHT | 133.300 | 28 | TRCH | 24.600 |
| 12 | RAHT | 102.700 | 29 | FOOT | 24.300 |
| 13 | STHT | 79.100 | 30 | HUM | 6.240 |
| 14 | DAHT | 60.300 | 31 | FEM | 8.870 |
| 15 | TIHT | 41.800 | 32 | SITHT | 90.200 |
| 16 | SPHT | 90.900 | 33 | APCH | 17.300 |
| 17 | AGR | 25.600 | 34 | HDG | 53.800 |

MEDIAN VALUES FOR SUBJECT NUMBER 8

|  | VARIABLE | VALUE |  | VARIABLE | VALUE |
|---|---|---|---|---|---|
| 1 | SUBJ | 13.000 | 18 | AGF | 28.600 |
| 2 | SEX | 2.000 | 19 | FAG | 24.800 |
| 3 | WT | 61.400 | 20 | WRG | 15.400 |
| 4 | HT | 171.800 | 21 | CHG | 85.300 |
| 5 | TPSF | 14.800 | 22 | WAG | 66.300 |
| 6 | SSSF | 7.300 | 23 | THG | 58.400 |
| 7 | ILSF | 8.500 | 24 | CAG | 38.800 |
| 8 | ABSF | 20.300 | 25 | ANG | 21.800 |
| 9 | THSF | 25.700 | 26 | BIAC | 36.900 |
| 10 | MCSF | 12.600 | 27 | BIIL | 29.800 |
| 11 | ACHT | 136.600 | 28 | TRCH | 24.600 |
| 12 | RAHT | 106.400 | 29 | FOOT | 24.000 |
| 13 | STHT | 83.000 | 30 | HUM | 6.190 |
| 14 | DAHT | 64.100 | 31 | FEM | 9.390 |
| 15 | TIHT | 43.000 | 32 | SITHT | 92.700 |
| 16 | SPHT | 91.700 | 33 | APCH | 18.500 |
| 17 | AGR | 28.100 | 34 | HDG | 57.000 |

MEDIAN VALUES FOR SUBJECT NUMBER 13

|  | VARIABLE | VALUE |  | VARIABLE | VALUE |
|---|---|---|---|---|---|
| 1 | SUBJ | 57.000 | 18 | AGF | 28.000 |
| 2 | SEX | 2.000 | 19 | FAG | 25.300 |
| 3 | WT | 62.600 | 20 | WRG | 15.600 |
| 4 | HT | 172.600 | 21 | CHG | 85.500 |
| 5 | TPSF | 10.500 | 22 | WAG | 68.200 |
| 6 | SSSF | 8.500 | 23 | THG | 55.500 |
| 7 | ILSF | 7.100 | 24 | CAG | 37.200 |
| 8 | ABSF | 17.800 | 25 | ANG | 22.300 |
| 9 | THSF | 15.300 | 26 | BIAC | 38.000 |
| 10 | MCSF | 6.700 | 27 | BIIL | 28.600 |
| 11 | ACHT | 141.000 | 28 | TRCH | 25.600 |
| 12 | RAHT | 107.200 | 29 | FOOT | 23.600 |
| 13 | STHT | 82.700 | 30 | HUM | 6.700 |
| 14 | DAHT | 65.800 | 31 | FEM | 8.910 |
| 15 | TIHT | 43.600 | 32 | SITHT | 91.000 |
| 16 | SPHT | 95.900 | 33 | APCH | 17.200 |
| 17 | AGR | 26.300 | 34 | HDG | 54.700 |

MEDIAN VALUES FOR SUBJECT NUMBER 57

## APPENDIX E.

## Variable Names and Meanings

| | |
|---|---|
| SUBJ | Subject |
| SEX | Sex |
| WT | Weight |
| HT | Height |
| TPSF | Triceps Skinfold |
| SSSF | Subscapular Skinfold |
| ILSF | Suprailiac Skinfold |
| ABSF | Abdominal Skinfold |
| THSF | Front Thigh Skinfold |
| MCSF | Medial Calf Skinfold |
| ACHT | Acromial Height |
| RAHT | Radial Height |
| STHT | Stylion Height |
| DAHT | Dactylion Height |
| TIHT | Tibial Height (Lateral) |
| SPHT | Spinale Height (Anterior-Superior Supine) |
| AGR | Arm Girth (Relaxed) |
| AGF | Arm Girth (Flexed) |
| FAG | Forearm Girth (Relaxed) |
| WRG | Wrist Girth (Proximal Styloid Process) |
| CHG | Chest Girth (Mesosternal) |
| WAG | Waist Girth (Min Circ.) |
| THG | Thigh Girth (1 cm. Distal Gluteal Line) |
| CAG | Calf Girth (Max Circ.) |
| ANG | Ankle Girth |
| BIAC | Bi-Acromial Breadth |
| BIIL | Bi-Ilochristal Breadth |
| TRCH | Transverse Chest |
| FOOT | Foot Lenght (Akropodion Pternion) |
| HUM | Bi-Epicondylar Humerus Width |
| FEM | Bi-Epicondylar Femur Width |
| SITHT | Sitting Height |
| APCH | Anterior-Posterior Chest Depth |
| HDG | Head Girth |

## APPENDIX F.

### Basic Program for Randomly Selecting Subjects and Variables

```
00010 rem     a random number generator to select subject
00020 rem     and variables for introducing error.
00030 print
00040 print
00050 for j=1 to 4
00060  print 'SUBJ #          VARIABLE'
00070 mx1=11
00080 mx2=31
00090 for k=1 to 10
00100 print mod(int(rnd*10000),mx1)+1,mod(int(rnd*10000),mx2)+1
00110 next k
00120 print
00130 print
00140 next j
00150 print
00160 print
00170 print 'SUBJ #          DESTINATION          SOURCE'
00180 print '                VARIABLE            VARIABLE'
00190 print
00200 for j=1 to 10
00210 l=mod(int(rnd*10000),mx2)+1
00220 print mod(int(rnd*10000),mx1)+1,l,
            mod(int(rnd*10000),mx2)+1
00230 next j
00240 print
00250 print
00260 end
```

## REFERENCES

Apple Computers Inc. Apple Pascal Reference Manual, Apple Computers Inc., Cupertino California, 1979.

Beers, Yardley. "Introduction to the Theory of Error", Addison-Wesley, Reading Mass., USA, 1957.

Borms,J.; Hebbelinck, M.; Carter, J.E.L.; Ross, W.D.; Larivière, G. "Standardization of Basic Anthopometry in Olympic Athletes - The MOGAP Procedure" Methods of Functional Anthropometry , Novotny and Titlbachova eds., pp 31-39, Universitas Carolina Pragensis, Praha, 1979.

Burkinshaw, L.; P.R. Jones; D.W. Krupowicz. "Observer Error in Skinfold Thickness Measurements." Human Biology. 45:273-279, 1973.

Date, C.J. An Introduction to Database Systems, Addison-Wesley, Reading, Massechusetts, 1975.

Drinkwater, D. Personal communication, 1982.

Duquet. W.; F.De Meulenaere; J. Borms. "A Method For Detecting Errors in Data of Growth Studies." Annals of Human Biol. 6:431-441, 1979.

Freund,R.J. and H.O. Hartley. "A Procedure for Automatic Data Editing." Am.Stat.Ass.J. June: 341-352, 1967.

Gaito, J. and E.C. Gifford. "Components of Variance in Anthropometry." Human Biology 30:102-127, 1958.

Garn, S.M. "Some Pitfalls in the Quantification of Body Composition." Annals N.Y. Acad.Science. 110:171-174, 1963.

Goldstein, H. "Data Processing for Longitudinal Studies." J. Royal Statistical Soc. Series C19:145-151, 1977.

Greenway, R.M.; A.S. Littell; H.B. Houser. "An Evaluation of the Variability in Measurement of Some Body Composition Parameters." Soc.Exp.Biol.Medicine. 120:487-492, 1965.

Harrington, G. "The Separation of Technical Errors and Biological Variation, and Other Statistical Problems Arising in Body Composition Studies." Annals N.Y. Acad.Science. 110:642-653, 1963.

Healy, M.J.R. "Notes on the Statistics of Growth Standards." Annals of Human Biol. 1:41-46, 1974.

Jensen, K.; Niklaus Wirth. Pascal User Manual and Report, Springer-Verlag, New York, 1978.

Jamison, P.L. and S.L. Zegura. "A Univariate and Multivariate Examination of Measurement Error in Anthropometry." Am.J.Physical.Anthrop. 40:197-204,1974.

Jones, P.R.M.; W.A. Marshall. "Harpenden Electric Read-out (HERO) Skinfold Calipers", Annals of Human Biology, 1979, vol 6, no.2, 159-162.

Kahn, H.A. "The Dorn Study of Smoking and Mortality Among U.S. Veterans: Report on 8-1/2 Years of Observations." Epidemiological Approaches to the Study of Cancer and Other Chronic Diseases, Haenszel, W. ed., National Cancer Istitute Monograph No. 19, National Cancer Institute, Bethesda, Maryland, 1966.

Largo, R.H.; T. Gasser; A. Prader; W. Stuetzle; P.J. Huber. "Analysis of the Adolescent Growth Spurt Using Smoothing Spline Functions." Annals of Human Biology. 5:421-434, 1978.

Leahy, R.M.; D.T. Drinkwater, G.W. Marshall, W.D. Ross, A. Vajda. "Computer Solutions For Longitudinal Data: Tri Dimensional Computer Graphics in the Resolution of Growth Curves", Kinanthropometry II Beunen and Simons, eds, Park University Press, 1980.

Padgham, C.A. "Subjective Limitation on Physical Measurements", Chapman and Hall, London, 1965.

Ross, W.D.; M. Hebbelinck; B. Van Gheluwe; M.L. Lemmens. "Kinanthropometrie et l'appreciation de l'error de mesure." Kinanthropologie, 4:23-34, 1972.

Ross, W.D., and N.C. Wilson. A stratagem for proportional growth assessment. In: J. Borms and M. Hebbelinck (Eds.), CHILDREN AND EXERCISE, IV International Symposium on Pediatric Work Physiology (Den Haan, Belgium). ACTA PAEDIAT. BELG. 28., suppl. 169-172, 1974.

Ross, W.D. Personal communication, 1982.

Sterling, T.D.; James J. Weinkam. "What Happens When Major Errors are Discovered Long after an Important Report Has Been Published". An invited talk to the session on "Policy Issues in the Use of Third Party Data Bases" of the American Statistical Association annual meeting, August, 1979.

Tonkens, Ross M. "A Guided Tour of Apple Pascal Units and Libraries", Byte 225-243, Feb. 1982.

Tremblay, J.P.; P.G. Sorenson. An Introduction to Data Structures with Applications. Mcgraw-Hill, New York, 1976.

van der Linden, F.P.G.M. "The Interpretation of Incremental Data and Velocity Growth Curves." Growth. 34:221-224, 1970a.

van der Linden, F.P.G.M.; W.J. Hirschfeld; R.L. Miller. "On the Analysis and Presentation of Longitudinally Collected Growth Data." Growth. 34:385-400, 1970.

Walpole, Ronald, E.. Introduction to Statistics The Macmillan Co., New York, 1968.

Ward, R., The Proposal, Validation, and Application of a Caliper Technique for the Estimation of Uncompressed Skinfold Thickness, Msc. Thesis, Simon Fraser University Kinesiology Department, 1980.

Ward, R. Personal communication, 1982.

Weiner, J.S. and J.A. Lourie. "A Guide to Field Methods" IBP Handbook No. 9., Blackwell Scientific Publications, 1969.