COMPUTER GRAPHICS APPLIED TO REGRESSION ANALYSIS

Philippe C. Thibault, Dip. Eng. IEG

ABSTRACT

Computer graphics are found to be an invaluable tool for a large number of applications in which man-machine interaction is needed. This work is concerned with the writing of a software handler for a disc-oriented graphics unit and with the application of this system to interactive regression analysis.

Chapter I contains a survey of computer graphics concepts and systems. The mathematics involved in regression analysis and a review of the techniques commonly used are exposed in Chapter II. In Chapter III, the Regression Analysis Package is introduced, as well as the computing facilities for which it has been developed. Chapter IV is concerned with the details of the software graphics interpreter and discusses the usage of the multiple user tracks and of the light-pen. The last chapter contains an evaluation of the system from the designer's and from the user's viewpoints.

A DISC-ORIENTED GRAPHICS SYSTEM APPLIED TO INTERACTIVE REGRESSION ANALYSIS

by

Philippe C. Thibault, Dip. Eng. IEG

Department of Electrical Engineering

McGill University,

Montreal, Quebec.

A DISC-ORIENTED GRAPHICS SYSTEM

APPLIED TO

INTERACTIVE REGRESSION ANALYSIS

by

Philippe C. Thibault, Dip. Eng. IEG

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Engineering.

Department of Electrical Engineering,
McGill University,
Montreal, Quebec.

July, 1972.

ACKNOWLEDGEMENTS

I wish to express my sincere thanks to Dr. A. Malowany for his guidance and encouragement during this project.

I also express my gratitude to the Canada Council for the award of a scholarship and to the National Research Council and Canada Cement Lafarge, who supported this project.

Thanks are also due to

Messrs, F.Barrow and G.Taillon of Canada Cement Lafarge for gathering and evaluating the industrial data which have been used in this work;

Messrs. G.Husson and R.Mildenberger for proof-reading this thesis;

and, finally, to my wife Christine who typed it.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION: A SURVEY OF COMPUTER GRAPHICS

Twenty years have elapsed since the first attempt at using cathode ray tubes (CRT) in conjunction with computers (Massachusetts Institute of Technology 1951) (1). These twenty years have seen considerable development of computer graphics, with two main goals. The first one was to improve the transfer of information of a basically graphical nature and efforts along this line have led to three-dimensional colour display (4), computer animation, (5,9), stereoscopic (5) and even holographic displays (6).

The second goal was, more generally, to ease and speed up communication between man and machine and has been fulfilled in a vast number of applications including, for instance, computer-aided instruction (7,8,9) large-scale computer-based record systems (10) and interactive modeling facilities (31).

The field of application of graphics has also considerably widened since the early systems surveyed by Davis (1). Due to their highly strategic properties and also, perhaps, due to the costs involved, the early graphics systems were developed for a military environment. In fact, one of the first fully operational systems which was of any importance was an integral

part of the SAGE Air Defense Command equipment in 1954. Later, systems of a more general purpose such as Sketchpad (3) made their appearance. Today's systems cover a wide range of size, cost, capabilities and applications. In the following, we shall describe their basic features and quote some typical examples of actual applications.

1.1. Graphics as a visual aid

Graphics systems are particularly useful in applications where the computer output is of a graphical nature or is better expressed in graphical format. This includes all kinds of graphs, charts or more complex drawings such as readily found in applications of engineering (12,13,14, 15), management (16), cartography (17), physiology and medicine (10,19,20), mathematics (21).

Improvements in this purely visual aspect of computer graphics have been obtained through hardware technology: dot and stroke vector generation (22), hardware character generation (2,22), colour CRT's (36,37) and large screen displays; and also, much more dramatically, through software development. Efforts along this line have been primarily aimed at assisting the programmer in his dealing with graphical entities.

The first step is the building of a library of subprogrammes written in a conventional language such as FORTRAN or ALGOL and stored for further reference by application programmes. Such a library may find its place in a large general-purpose installation but, very often, it is part of a sophisticated application-oriented system and, at least in its lower level structure, it is machine-dependent. A graphic library will contain basic entities such as characters, squares, circles and special elements such as electrical component symbols for a circuit designer (12) or schematic vehicles for a traffic-control engineer (23). Also in a library may reside a wide range of preprogrammed algorithms for drawing a solid or dotted line between two points or for scaling, rotating and translating an element. An interesting class of algorithms concerns the representation in two dimensions of three-dimensional objects: computation of perspective views or of projections given the viewing angle; algorithms for concealing hidden lines or surfaces in such views (20,24), for shading (20) or colouring differently the various faces of a volume or for representing the intersection curve of two volumes (4). Beside obvious applications to the design and analysis of three-dimensional objects as are found in architecture, macromolecular chemistry (5), mechanical design and many other fields, these programmes are very useful for data analysis. Examples are given by Birnbaum et al. (25) in nuclear physics and Coulam

et al. in physiology (20).

A higher level of sophistication is the creation of exclusively graphics-oriented languages which permit a quasi-infinite graphic creativity. Using such systems, the programmer may define and manipulate graphic variables just as easily as he does numerical variables in FORTRAN. For instance, W. Newman's "display procedures" in EULER-G (27) allow the user to position, rotate and rescale, in one statement, a previously defined entity. The definition itself of complex graphical entities by the programmer is greatly facilitated by a number of features such as windowing, by which one can define a graphic variable as a subset of another by "clipping" a rectangle from this latter (for instance, some figure might be composed by the juxtaposition of several arcs clipped from a number of initial circles of various diameters). We shall only mention here the use of input devices such as the light pen for the definition of graphic variables, as they will be thoroughly discussed in the next section.

At this point, it is worth while to say a few words about the data structures used in computer graphics (11). The simplest case is in fact the total absence of structure: the display information is stored in a linear file and continuously read by the display processor. This leads to

extreme redundancy (e.g. a new group of instructions is added to the file each time the same alphanumeric character is used). Considerable improvement has been brought about with the concept of structured files in which the instructions are laid out in a tree-like structure consisting of several levels of sub-pictures. The branching through those elements of the display is not performed by the computer at the display assembling stage but by the display processor itself, by means of a set of pointers. These structures provide a saving in storage space and lower the computer time requirements. They also allow a great flexibility for the modification of displays, since this can most often be reduced to the changing of a few pointers. Last, they facilitate interactive procedures. An example of a language using this type of structure is GRIN-2, described by Christensen (26). Unfortunately, such languages are usually machine-dépendent and their adaptation to different environments is difficult. Recently, a new approach has been used, eliminating the need for such a structure. An example of this kind of organisation may be found in EULER-G (27).

It is worth noting that many of the functions described above could be adequately fulfilled by a plotter or a line-printer. Such a hard copy might even be cheaper and more convenient when speed is not of primary importance or when

the output is of the one-of-a-kind type, because the user can free the installation or the computer while he is studying the output. Other competitors to computer graphics, particularly in managerial applications, are the new micrographics systems which directly produce 35mm slides, 16mm films and even microfiches as computer output (38).

However, graphics has a definite advantage when many displays are to be viewed before a hard copy (if any) is needed. A typical example of such usefulness is the debugging of a system under development. And, of course, graphics will be irreplaceable when some sort of animation takes place (dynamic graphics) (9). A typical application is real-time process control with on-line decision-making; it is hardly surprising that the largest known computer graphics installation is part of the NASA Mission Control at Houston. Many applications of this type but of a more modest size are found in military and industrial installations. Webber (23) describes the use of graphical displays for the on-line control of London traffic. Graphics animation is also widely used in modelling and simulation. A flight simulator is described in (5) and another interesting example borrowed from aviation is the simulation of an Air Traffic Control system carried out by the U.S. Department of Transportation (28). A study of computer graphics used as a tool for simulation can be found in Bell (56).

Many graphics installations provide an alternative hard copy output. In some cases it may be a camera, controlled either manually or by software, which records the information displayed on the screen. In the most sophisticated installations, the display processor may be similarly interfaced to the graphics unit or to an x-y plotter, thus providing maximum flexibility.

1.2 Graphics as a communication aid.

While the software described in the previous section was particularly aimed at facilitating the programmer's task, the contents of this section are rather user oriented. Graphics systems greatly improve man-machine communication by providing an efficient tool for interaction.

The first level of improvement is the handling of alphanumeric characters. Both hardware and software generators have been developed and graphic consoles including a storage scope and a keyboard advantageously replaced conventional keyboard-printer units for remote terminal installations. Alphanumeric computer output is much faster and editing is easier, thanks to a number of features such as scrolling, forward and backward paging, etc. A non-negligible side-advantage of such terminals is their total absence of noise. However, they may be rather costly in a time-shared environment.

Yet the essential feature in interactive graphics is the graphic input ability which resides in two types of devices; the light-pen(29) consists of a photo-diode which detects the electron beam and acts by interrupting the computer; the track-ball or joy-stick simply outputs the x,y position of a point on the screen and feeds it to an analog/digital converter for use by the computer. A similar type of device is the RAND tablet (30); typical examples of its use are shown in (28) and (31).

Most of the software effort in graphics interaction has been aimed at rendering man-machine communication more and more natural. The ultimate goal is to free as much as possible the user from the programming language constraints and to prevent such time-consuming errors as mistyping, choice of illegal code or off-limit values etc.

One very convenient feature in this aspect is the light-button. When a choice, such as a branching point in a programme flow or a parameter value, is required from the user, a menu of possible choices is offered in all or part of the display and the user selects his decision by pointing at the appropriate item with the light-pen. An application where this feature is particularly useful is for the on-line input of data by users unfamiliar with data processing. Some of the advantages of light-button selection over keyboard entry are speed and accuracy,(even if the user has no particular typing skill) and relief from the necessity of coding or abbreviating entries, since one

single selection may be a long and explicit entry. This will considerably reduce the risk of time-consuming spelling errors or of wasteful misinterpretations. Uber et al. (10) describe a system used by physicians for the storage of medical records and for the preparation of lab test orders and drug orders from menus of duly classified tests, drugs, doses, modes of administration etc. A very extensive use of light-pen interaction is also illustrated in the modeling system BIOMOD set up by Rand Corporation (31).

Additional efficiency may be obtained through the concept of reactive display (32). Using this technique, items among the menu which are not acceptable in the particular context (e.g. off-limit values of parameters) are selectively disabled. Choosing such an item would not cause an interrupt and might for instance light up a warning signal on the screen. Other features in the same line include static intensification; for instance only those titles or light-buttons or parts of the display which are currently active might be intensified (by brighter illumination, blinking or use of a larger size). This allows the user's attention to be directed to current actions to be taken without depriving him of the rest of the display. Dynamic intensification will help to prevent errors of inattention such as slips of the light-pen; the item being pointed at, provided of course that it is acceptable, is temporarily intensified by one of the means mentioned so that the user can check his actual choice; no interrupt will take place

until the light-pen is moved away from the screen surface.

It should be noted that some research has been done to seek the optimization of light-button layout and of the break-up of large menus into successive selections (10,33).

1.3 Graphics system organisation and hardware.

In some systems, such as BRAD (34) or the IBM 1500 Instructional System (35), the CRT is a standard TV monitor in which the display is refreshed by a video signal. Their main advantage is the use of a standard display equipment; this is particularly economical when multiple outputs from the same display processor are needed. However, the computation required for the coding or the graphical information into a video signal as well as the storage space necessary for that digitized signal are prohibitive. Also, the synchronization of the information with the raster causes a serious problem. In a recent paper, Lovercheck (36) describes a system in which this problem has been successfully solved through servo-control of a disc-buffer and which is used in a number of installations. It may be worth noting that this system makes provision for vertical raster-scanning, very convenient for the plotting of single-valued functions. Covvey et al. (19) discuss an installation where TV signals are used for both input and output.

In fact, in many typical graphics systems, the standard TV monitor and its raster have been replaced by a display unit in

which the beam can be moved by software control to the desired position on the screen. This leads to a considerable saving in the storage required for the display files and it also makes light-pen interaction possible.

A class of systems make use of storage scopes. On such a screen, the phosphor excited by the passage of the electronic beam stays illuminated until the whole screen is erased. Therefore, the display file needs to be transferred only once to the display unit; however, severe limitations restrict the areas of application of such systems. Although portions of the display may be added to the current picture, none can be removed without erasing the whole picture and rewriting it in full, even if the change is of minor extent. This makes dynamic graphics impractical and in any case expensive. Also, the only graphical input device that can be used is an x,y detector such as a joy-stick. Since the beam scans the picture only once, the light-pen is not useable.

Apart from their simplest use as screen-keyboard units as mentioned in the beginning of section 1.2, some typical applications of storage tube graphics can be found in the hybrid computer-aided design system introduced in (15) and in many of the systems specially designed for mathematics and surveyed by Smith (21).

Other systems make use of a short persistence phosphor screen. The display needs then to be refreshed at a conven-

ient rate (30 to 40 times a second) to prevent flicker. This refreshing allows the updating of the display or of part of it as often as permitted by computation speed and by other delays such as those due to communication lines or to time-sharing. Such conditions permitting, actual animation is no problem. The computer keeps updating the file and the refreshing task is carried out on a cycle stealing basis. Some examples of computer animation are found in reference (5) and a very complete bibliography is given by Weiner (9).

Such a setup will find its limitation when this frequent cycle stealing is not desirable, for instance because the computer has a heavy non-graphic task or because it is used in a time-sharing environment. It may also happen that the transfer rate is not sufficiently high to ensure a flicker-free display (or to match the refreshing rate if this latter is fixed). This might be because of long transmission lines or, simply, because the pictures displayed are too complicated. Data compression schemes such as proposed by Pitteway (39) help solve this problem. Also, the refreshing rate may be somewhat reduced by using phosphorus of longer persistence.

Yet the most efficient solution, even after data compression, is the insertion of a refresh buffer between the computer and the display unit. This buffer may be a magnetic drum or disc, a delay line or a core bank. Once the display file has been written in the buffer a reading head continu-

ously reads it off the buffer and transmits it to the display unit; thus the computer is totally free of refreshing tasks until a change is required in the display, at which point either the total picture is reorganised and rewritten on the buffer or only part of it is patched. At the cost of a slightly slower transfer rate between core and display unit, considerable savings in computing time are obtained.

The magnetic disc buffer has a particular advantage; it gives the capability of storing several different pictures on different parallel tracks which can be read simultaneously. A typical 16-track disc can thus be used to monitor simultaneously up to 16 different terminals sharing the same display processor. This makes the disc buffer the cheapest technique when several terminals are required. It also permits the storage of colour pictures, by using three parallel tracks for each colour CRT or the use of a grey scale with conventional black and white CRT's. Other side advantages include the superimposition of two or more independent pictures on one display; for instance, one track may contain a fixed, write-protected frame inaccessible to the user (e.g. a map or an administrative form) while another, accessible to the user, is used to enter the data (36).

Refreshed displays allow the use of both light-pen and joy-stick devices and they are therefore most flexible since each of these devices has its respective advantages. Acting

by computer interrupt, the light-pen is very convenient for identifying an item on the screen, regardless of its actual scale or position. It is then widely used for light-button operation or for constructing a picture from pre-existing elements. On the other hand, it will be totally useless for pointing at a blank portion of the screen. In particular, drawing with the light-pen requires the use of a tracking pattern such as the one shown in Figure 1.1 and of an algorithm which recognizes the direction taken by the pen.

The joy-stick or the Rand tablet is less convenient for item identification, because of the necessity of mapping back the physical screen into the actual display but it performs very easily graphical input tasks such as drawing. Hand drawing is with talking the most natural way of communicating with the machine. Covvey et al. (19) use a light-pen to outline a cardiac image on a TV monitor and feed the information to a computer. Groner et al. (31) show how a BIOMOD user may enter text information by handwriting on a Rand tablet.

This brief survey has attempted to sample the numerous applications of computer graphics in order to give a general idea of what is being done in the field. Before introducing the regression analysis package which has been written for this project, we shall give in the next chapter a brief presentation of the mathematics and methods involved.

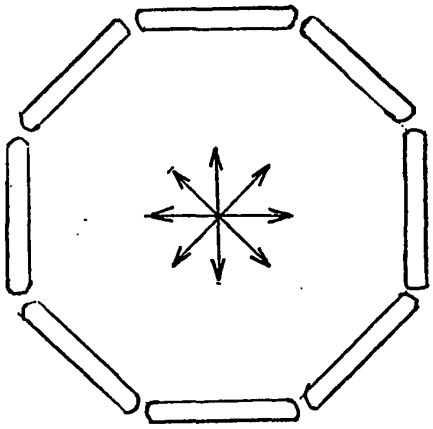


Figure 1.1 A pen-tracking pattern .

CHAPTER II

THE REGRESSION PROBLEM

In most areas of industrial activity, it is desirable to have a mathematical model of the process, in the form of a set of equations between the process variables. The model may be used to design a new installation, to simulate the process, to control it or to establish production schedules. Very often,however,pure theoretical knowledge is not sufficient to understand fully the processes involved and to describe them quantitatively. Even in the case of well known processes, there often are a number of parameters which it would be unrealistic to calculate through theory. This is the case in particular when solid raw materials are used in the process (e.g. in paper and cement making) (40).

It is then necessary to derive the model from the observed process, by some statistical method such as regression analysis (41,42,43). Sometimes this empirical model will even constitute the starting point for some theoretical research.

Pencil and paper regression analyses have been conducted for decades in a number of fields but the expanding use of computer facilities in recent years has largely increased their practicality. At the same time, data collection is easier and

easier thanks to modern recorders and it often comes as a by-product of computerized process control. In many instances one can afford to measure every single process parameter at any desired frequency and hourly, daily or weekly data logs are often available on tape or punch cards. This formidable amount of numerical information about the process makes statistical analysis more and more significant and therefore attractive.

It should be noted that the lack of theory is not the only reason for using regression analysis. One of the most common uses of mathematical models is for computerized process control. It often happens that theoretical knowledge is complete enough to provide a precise and reliable model but that this model is much too complex for routine use by the computer control system. It may then be advisable to use an empirical but more practical model.

Regression analysis may also be used for testing a model obtained through some other technique. The discrepancy between the observed dependent variable and the calculated one is used as the dependent variable to be regressed against the independent variables. This may show a trend indicating that the model should be made more complex and give a clue to the form of the additional terms to be included in the equation (40).

2.1 Generalities.

In the regression analysis of a set of data, a functional relationship between two or more variables is first assumed either by theoretical knowledge or after a preliminary examination of the data. The "model" sought by the analyst is thus an equation of the form

$$\hat{y} = f\ (x_1, x_2, x_3, \ldots).$$

This equation predicts the average value $\hat{y}$ of the "dependent" variable (or "response") y when the "independent" (or "controlled") variables x_1, x_2, x_3,... take on specific values. To "build" the model is to derive satisfactory values for the parameters of the functional 'f' from a set of observed values of the dependent and independent variables.

When a model has been found, a significance test may be used to assess its quality with respect to some given criterion. Such a test will help the analyst in deciding between several models, by comparing their performances, but it cannot decide whether one model is the best in the absolute sense (43, 44).

2.2 Properties attached to models.

2.2.1 Linearity.

A very important class of models is the linear class.

It is important to note that, in this context, "linearity" means "linearity in the parameters" and not as one might be tempted to interpret it, in the dependent variables (45). This class, in fact, covers a wide range of equations. For instance, the following models

$$\hat{y} = a_0 + a_1 x_1 + a_2 x_2^3$$

$$\hat{y} = a_0 + a_1 \sin(x) + a_2 e^x$$

are linear (in the unknown parameters a_0, a_1, a_2) and they could be sought by using a linear technique after transforming the variables; this would be done by defining the new variables $z_1 = x_2^3$; $z_2 = \sin(x)$; $z_3 = e^x$.

On the other hand, a model such as

$$\hat{y} = a_0 + x_1^b$$

is non-linear if b is an unknown parameter to be estimated.

An interesting advantage of linear models is that they can be sought by the very common least-squares technique, discussed later. In fact, most of the statistical techniques available are linear whereas nearly all physical processes are, strictly speaking, non-linear. Fortunately, it is also true that most non-linear processes can be approximated to linear ones, within reasonable boundaries to be determined in each case.

2.2.2 Bias.

Let us assume that one has obtained several estimates of the model parameters, from different sets of data taken under identical conditions (so that they actually are samples of a homogeneous population). The estimation technique used is said to be unbiased if these estimates, in the long run, average out to the exact values of the model parameters.(44,45). In the linear case where the true model is expressed in the matricial form $\hat{y} = Ax$, and where the estimates of the matrix A are matrices A_i, the unbiasedness condition is that

$$E(A_i) = A.$$

2.2.3 Goodness of fit.

The performance of the model is evaluated in terms of the overall "closeness" of the actual values y_i of the dependent variable to the corresponding predicted values $\hat{y}_i$. The discrepancy $e_i = \hat{y}_i - y_i$ is called the residual. The measure commonly used to assess the performance is the standard deviation about the regression line. From now on and for clarity, the model will be assumed to be a first order linear model $\hat{y} = a + bx$; hence the expression "regression line". The following discussion will be easily extended to the general case of a regression hyperplane.

One can show that an unbiased estimate of that variance is

$$s = \sum_{j=1}^{n} (y_j - a - bx_j)^2/(n-2)$$

where n is the number of data points. This expression is called the "standard error of estimate". If the model is correct, this estimate s equals the variance of the residuals e_i.

2.3 The least-square method.

This is the most commonly used method for finding a linear model. It is the one which will be used in this project. The exact model is assumed to be

$$\hat{y} = E(y/x) = \alpha + \beta x$$

where E(y/x) is the mathematical expectation of y when the dependent variable takes on the value x. The least-squares approach seeks for α and β estimates a and b which minimize the cost function

$$D = \sum_{j=1}^{n} (y_i - \hat{y}_i)^2$$

$$= \sum_{j=1}^{n} (y_i^2 - a^2 - b^2 x_i^2 - 2abx_i)$$

i.e. the sum of the squares of the vertical distances d_i as shown in Figure 2.1.

This method will be particularly appropriate and will

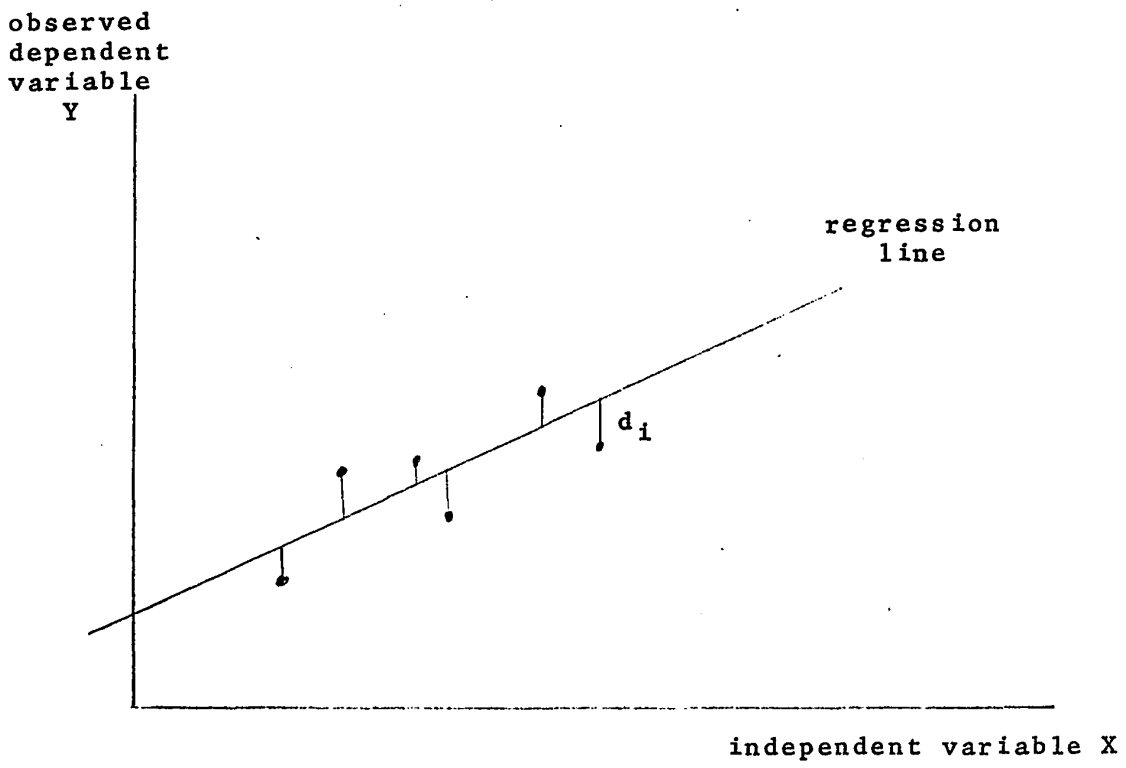


Figure 2.1 The least-squares technique.

yield best results when a number of assumptions are satisfied. The basic asssumptions are the following (44)

1. The model is linear.
2. The residuals $e_i = y_i - \hat{y}_i$ are mutually independent.
3. Their variance σ^2 does not depend on x.(this assumption is known as the homoscedasticity condition).

If these conditions are satisfied, it can be shown that the least-squares technique gives estimates a and b which are unbiased. Furthermore, among all estimators which can be obtained linearly from the data, a and b are those with the smallest variances. The linear property is very convenient but is should be noted that some non-linear estimates might be found to have smaller variances.

The latter property lies on the assumption of homoscedasticity; if the residual variance is not constant, the estimates found with the least-squares technique are still unbiased but their variance is no longer minimum among linear estimates. In this case, a weighted least-squares approach would be recommended.

If the residuals are not mutually independent (failure to comply with assumption 2), which fortunately happens rarely, the validity of most statistical techniques, in particular of the F-test,will be affected. The use of these techniques also requires a fourth assumption. The residuals should have a

normal distribution. This property is most likely to be unsatisfied. However, a deviation from normality can be most of the time tolerated to some extent without leading to serious problems. It is shown that when there actually is normality, the least-squares estimates also are maximum-likelihood estimates and do have the smallest possible variance (among both linear and non-linear estimates).

In addition to those basic assumptions, required by the very nature of the method, its practical application also necessitates that the following conditions be met. The data used should be "typical", i.e. they should be taken from the very population for which the model is sought and cover the whole range of values which the model is supposed to cover. Also, there should be no other variable than those in the model which makes the relationship described by the model of little intrinsic value.

The estimates a and b are found by equating the partial derivatives $\partial A/\partial a$ and $\partial A/\partial b$ to zero. A complete derivation will be found in reference (43).

2.4 Evaluation of a model.

Once a model has been found, it is necessary to check whether the assumptions made in using the method are correct. It is important to note that only a negative answer can be decisive. If the assumptions look violated, the model is to

be rejected; however, if they do not look violated, this is no sufficient reason to accept the model. One can only conclude that there is no evidence that it is invalid.

We have seen in the previous paragraph that the residuals are assumed to be independent and normally distributed about a zero mean with a constant variance. There exists a number of statistics which permit a quantitative evaluation of the validity of these assumptions (46) but visual techniques are usually more informative. Some of them will be described in the following (40,44).

2.4.1 Histogram.

A histogram of the residuals will provide a rapid check of the normality. With a little practice, one can easily detect significant departures from a Gaussian distribution curve with zero mean. A more elaborate technique to check normality is the plotting of the residual cumulative frequencies on a normal grid. Plotted on such a grid, normal variates will fall on a straight line, the slope of which represents their variance. (Figure 2.2).

2.4.2 Time-sequence plot.

This plot helps to detect the influence of time in the model. If the model is correct, the residuals should lie in a horizontal band centred at zero (Figure 2.3.a).

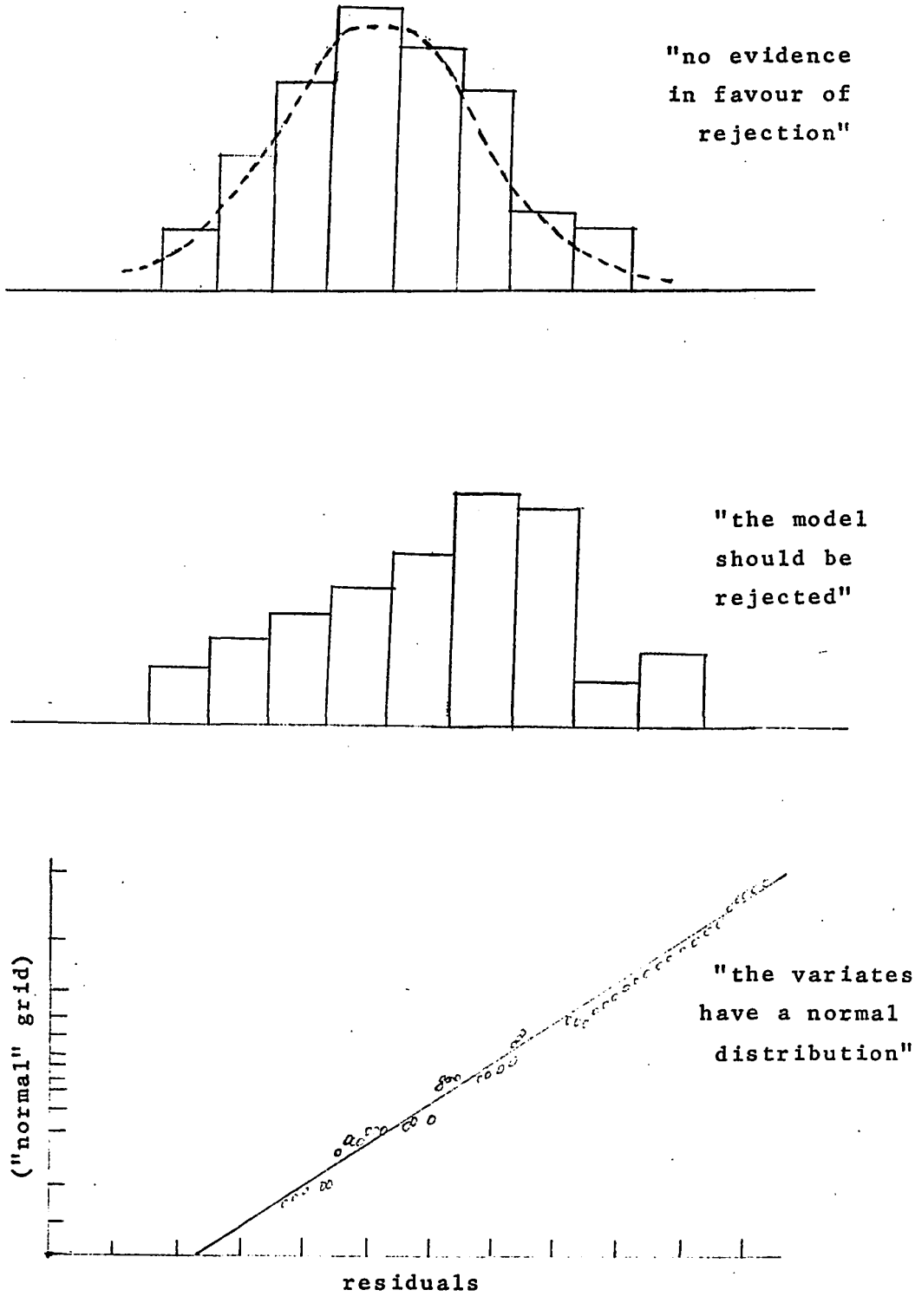


Figure 2.2 Checking the normality of the residuals.

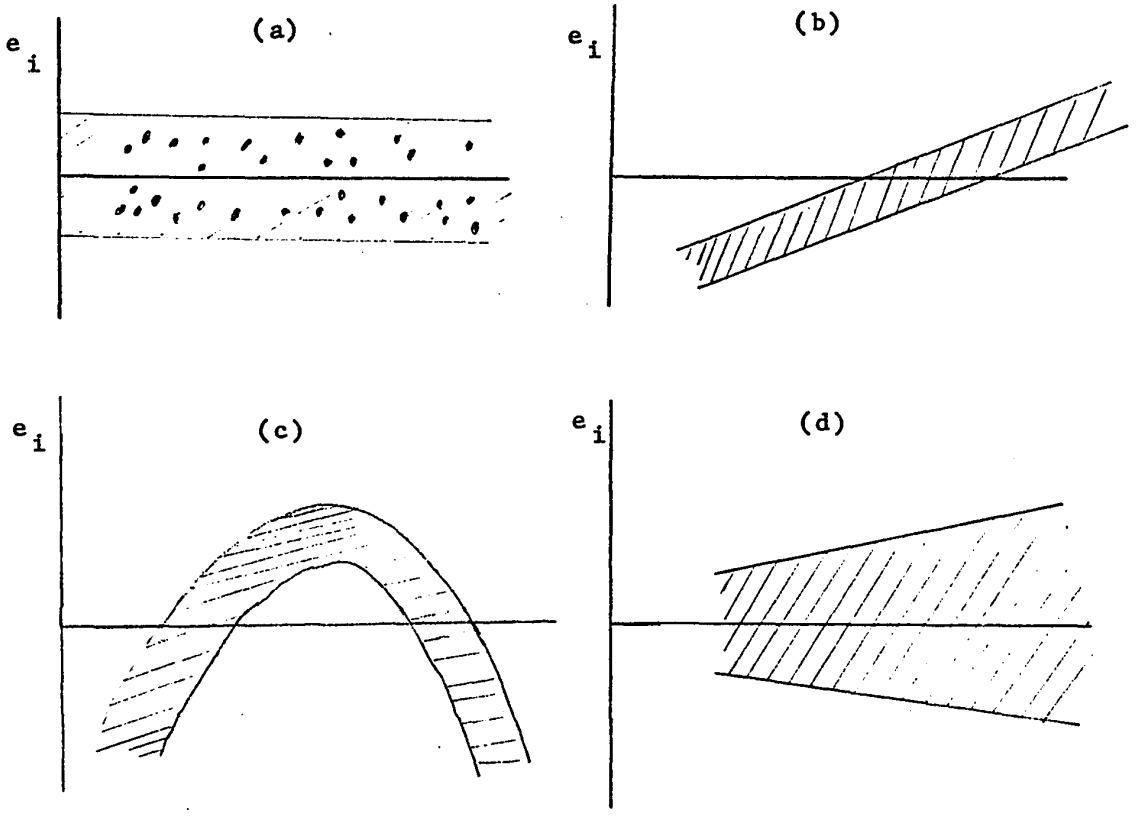


Figure 2.3 Residual plots.

If the band, although of constant width, significantly departs from the horizontal, then time (or some other variable dependent on time) should be included in the model. This term might be linear (Figure 2.3.b) or quadratic (Figure 2.3.c).

If the spread shows a tendency to vary in time, then a lack of homoscedasticity is to be suspected and a simple unweighted least-square technique is not sufficient (Figure 2.3.d).

This examination should be carried out not only on the overall plot, but also on repetitive subsets, to check for possible short-term time effects.

2.4.3 Plots against the variables.

Residuals may be plotted against the dependent variable and against each of the independent variables. Here again, a correct model will be indicated by a horizontal band. If the band has a constant width but is not horizontal (Figure 2.3.b), it means that a constant term may have been omitted (if the plot is against the dependent variable) or that the coefficient of x_j has been wrongly calculated (if the residuals are plotted against the independent variable x_j).

A pattern such as the one shown in Figure 3.3.d indicates a need for a weighted least-squares method or else for some preliminary transformation on the dependent variable observed.

The pattern of Figure 3.3.c would call for some additional terms to be included in the model (quadratic or cross-product terms).

2.5 Analysis of variance.

In order to standardize the assessment of a model, it is desirable to double the visual inspection with a significance test. Every model found can thus be tagged with a quantitative rating. This rating can be used to compare two different models which, by simple visual inspection, appear of equal performance. It also provides a means for appreciating how reliable a given model is, by comparing it to a statistical distribution table at a specified level of significance, for the appropriate sample size and number of degrees of freedom. This assessment will be particularly necessary when the sample size is small.

The test usually carried out in such an analysis is the F-test, which is now briefly discussed.

The residual e_i can be broken up in two parts

$$e_i = y_i - \hat{y}_i = (y_i - \bar{y}_i) - (\hat{y}_i - \bar{y}_i)$$

where y_i, $\hat{y}_i$ and $\bar{y}_i$ respectively are the observed, predicted and average values of the dependent variable. If one develops the residual sum of squares already mentioned in 2.3, one

obtains

$$e_i^2 = \sum [(y_i - \bar{y}_i) - (\hat{y}_i - \bar{y}_i)]^2$$

$$= \sum (y_i - \bar{y}_i)^2 - \sum (\hat{y}_i - \bar{y}_i)^2$$

or

$$\sum (y_i - \bar{y}_i)^2 = \sum (y_i - \hat{y}_i)^2 + \sum (\hat{y}_i - \bar{y}_i)^2$$

$$SS = SS1 + SS2$$

SS is the sum of squares about the mean

SS1 is the sum of squares about the regression line

SS2 is the sum of squares due to the regression

The larger the ratio

$$R^2 = SS1 / SS ,$$

the better the model. This ratio, which ideally equals 1, is called the multiple correlation coefficient. It indicates what percentage of the total variation can be explained by the regression.

In order to be fully significant, the test has to take into account the number of degrees of freedom of each sum of squares. Let us assume that there are n data points in the sample used and p dependent variables in the model. There is one linear relationship between the n deviations about the mean (their sum equals zero); therefore, the sum of squares about the mean

SS only has (n-1) degrees of freedom, i.e. it is calculated from (n-1) linearly independent elements. Similarly, since the p coefficients of the model (used to calculate SS2) are found as p independent linear functions of the Y's, the sum of squares due to regression SS2 has p degrees of freedom. By subtraction SS1 has (n-p-1) degrees of freedom.

Thus, the mean squares are

MS = SS/(n-1) mean squares about the mean

MS1 = SS1/(n-p-1) mean squares about the regression

MS2 = SS2/p mean squares due to the regression

It is shown that, if the residuals are normally distributed, the residual sum of squares SS and the sum of squares due to regression SS2 follow two independent χ^2 distributions with respectively (n-p-1) and p degrees of freedom.

Therefore, the ratio

$$F = MS2/MS$$

should follow an F-distribution with p and (n-p-1) degrees of freedom. This can be used to assess the normality of the residuals, by comparing the calculated F ratio with the percentile point of the tabulated F-distribution having the same degrees of freedom F(p,n-p-1) at a specified level of significance.

2.6 Choice and examination of the data.

One of the most common reasons for the failure of a regression analysis is the use of a poor set of data. Whenever data are specially gathered in view of the building of a model, the collection should be carefully designed. In fact, the design of experiments constitutes an important branch of statistics; a very complete reference is the book by Cochran and Cox (47). Some common faults to be avoided in data sets are the following

- statistical dependence in the observed values of the dependent variable;
- multi-variate data which are not properly balanced;
- data which do not cover adequately and uniformly the whole range of values for which the model is sought.

Outliers need particular attention. These are points which grossly depart from the mass of the other data. They should first be detected (either by a statistical test or by visual inspection of data plots or residual plots) and then carefully analysed. An outlier may be due to an error in the recording of the data but it may also a valid data point obtained for some extreme value of one variable. In both cases, it should be removed from the data set as its presence affects the estimated model, but if it has been found to be a valid point, it will give a precious indication as to the range of validity of the model found. This important subject is

discussed by Anscombe (48) and Johnson (49).

2.7. Computer programmes for regression analysis.

When one undertakes the regression of a given data matrix the first step is to decide how many variables are to enter the model and which ones. The safest method is to try all possible combinations, derive the most satisfactory model for each of them and then select the best among these (50). With this strategy, one is certain to make the best possible use of the data given. However, as the number of variables goes up, this technique becomes extremely lengthy and its cost in computing time soon becomes prohibitive. In fact, it is wasteful in the sense that some preliminary analysis may often lead to the safe elimination of a number of variables from the original set.

Several schemes have been developed to systematize the choice of the variables entering the model. These strategies, which include forward and backward regression, stage-wise regression and step-wise regression, are thoroughly discussed by Draper and Smith (43) and their adaptation for computer programming is found in Efroymson (42). At the beginning of this research, some experience has been acquired of stepwise regression analysis, using one of the BMD programmes developed by Dixon (51). This programme gives the user great freedom in the scheduling of the regression. Although the choice of the variable to enter or leave the model at each step is made by the

programme by interpreting the F-test, the user can to some extent express his preferences by affecting a priority coefficient for every variable. He can also force a variable which he thinks important into the model (or keep an uncontrollable one out of it) and still obtain at each step the partial correlation analysis pertaining to the variable in question. All this freedom is extremely convenient, but it still has to be exerted before the regression is initiated. As the amount of output (tables, plots) is also to be pre-specified, the rich way to use this programme is to request the maximum information; this will end up in lengths of useless listing unless a significant result is obtained. If one tries to be money-conscious, by requesting the minimum at each trial, one typically ends up with a four-term equation and a promising fifth variable which has been kept out because only four steps have been asked for, or else with a satisfactory model which has to be reprocessed, the residual plots being requested this time. It did not take very long, in these conditions, to feel the need for interaction in regression analysis. The BMD programmes have been adapted for graphics displays by Dixon (52) and a number of other systems are surveyed and compared by Smith (21).

In the next chapter, we shall introduce the interactive regression analysis package which has been developed as part of this project.

CHAPTER III

AN INTERACTIVE GRAPHICS REALISATION

We have seen in the previous chapter that visual analysis plays an important role in a regression study. The object of this research was to set up a regression analysis package based as much as possible on those visual methods and making full use of the interactive features of the graphics system available in this Department.

This chapter contains a functional description, in general terms, of the programme which has been developed and a presentation of the computing facilities available. The details of the graphics software involved will be discussed in Chapter IV.

3.1 RAP: the Regression Analysis Package

3.1.1 Generalities

This programme has been designed in a modular way to allow maximum flexibility in the conduct of the analysis. Most user actions are to be performed with the light-pen, except for a few numerical inputs from the teletype keyboard. All outputs of a graphical nature are displayed on the CRT and may be put on hard copy by photographing the screen. Whenever numerical results are available, they may also be output on the line-printer at the user's option. To assist the user in the book-

keeping of his hard copies, a numerical index is automatically updated every time a new model is tried and the same index is displayed along with all plots, histograms and numerical results of any kind pertaining to that equation. Every set of data is stored on disc under a data name which is displayed along with every output.

The programme is initiated by typing in the data name. A basic "menu" (which will be called the "monitor" in the following) is then proposed for light-pen selection (Figure 3.2). This monitor is the main cross-road of the flow-chart and the programme may be restarted at this point at virtually any stage of the analysis. This allows the user to abort any manoeuver which he might have initiated and does not wish to complete.

Of the sixteen tracks (1-16) of the graphics disc, all are accessible to the user except track no. 16 which is reserved for system displays and messages and in particular all light-buttons. At many points in the programme, the user is requested to decide on which track he desires the subsequent display to be stored (Figure 3.5). Should he select track no. 16, his decision would be ignored. If the track selected is acceptable, the display is stored in the chosen track and the display processor is automatically switched, by software, to this track, so that the user always sees the latest display, without having to set the track selector. However, he has

the possibility of manually switching to different tracks if he wishes to examine or compare previous displays.

The functional flow-chart of the programme is shown in Figure 3.1 and will be documented in the following sections.

3.1.2 Preliminary examination of the data.

Before undertaking any regression on the data, it usually is valuable to gather their elementary statistics. These are provided by two of the modules proposed by the monitor.

By selecting the module MEAN SD, the user obtains the following information: mean value, range, variance and standard deviation. The module CORR provides the simple correlation coefficients between any two variables. Both modules first give the user the choice shown in Figure 3.3.a. If the option "TABLE" is selected, the total information concerning all the variables of the set is printed on hard copy. Control is then returned to the monitor. If "PARTICULAR" is selected, the user is invited to pick the variable(s) in which he is interested from the list shown in Figure 3.4. When it is called by the module CORR, this frame is displayed twice, so that one can select both variables involved. The requested information concerning the variable (or the pair) chosen is then displayed on the screen and, upon a light-pen hit, control is returned to the beginning of the module (Figure 3.3.b. The user can call the monitor by selecting "RETURN"

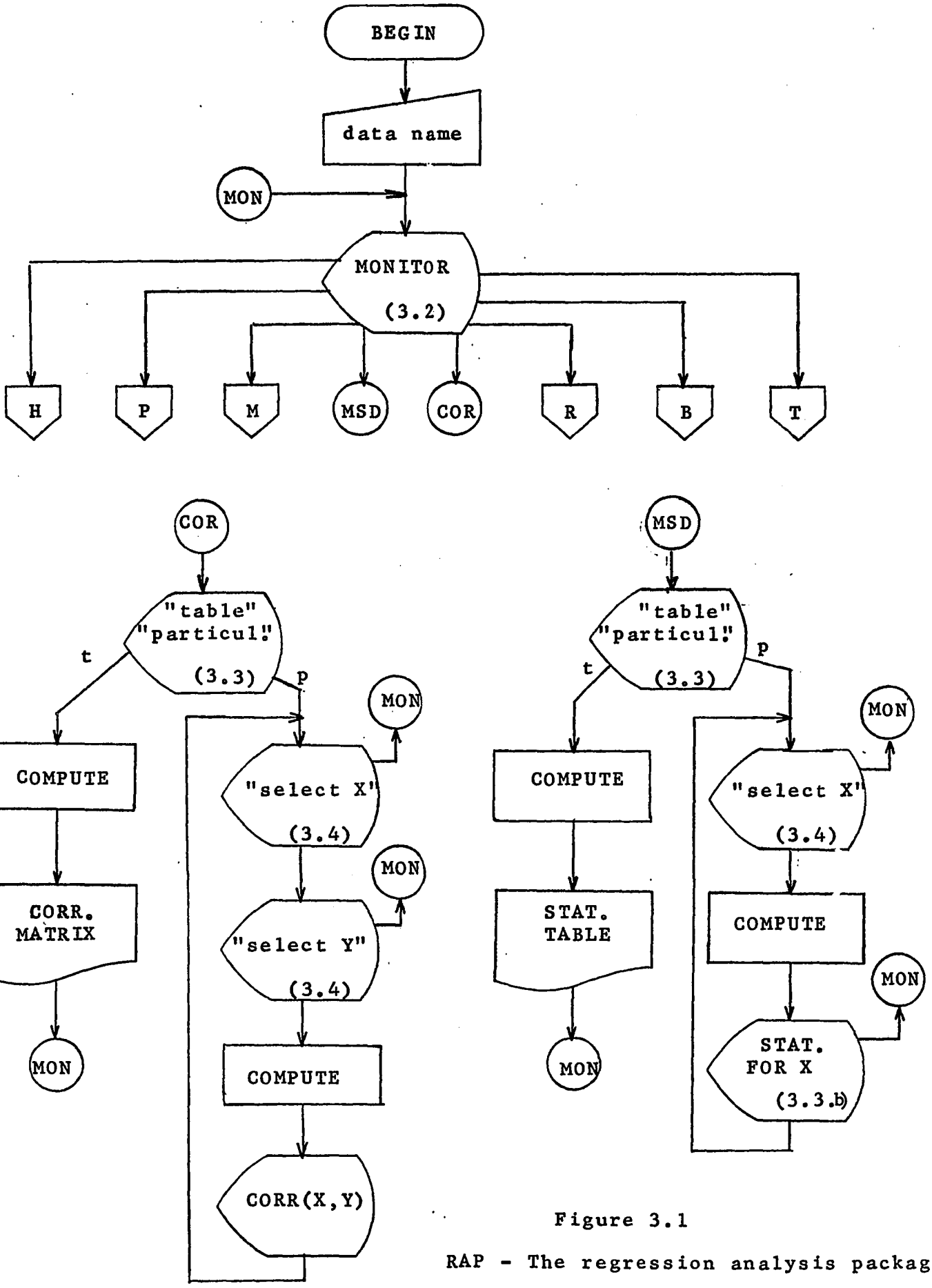
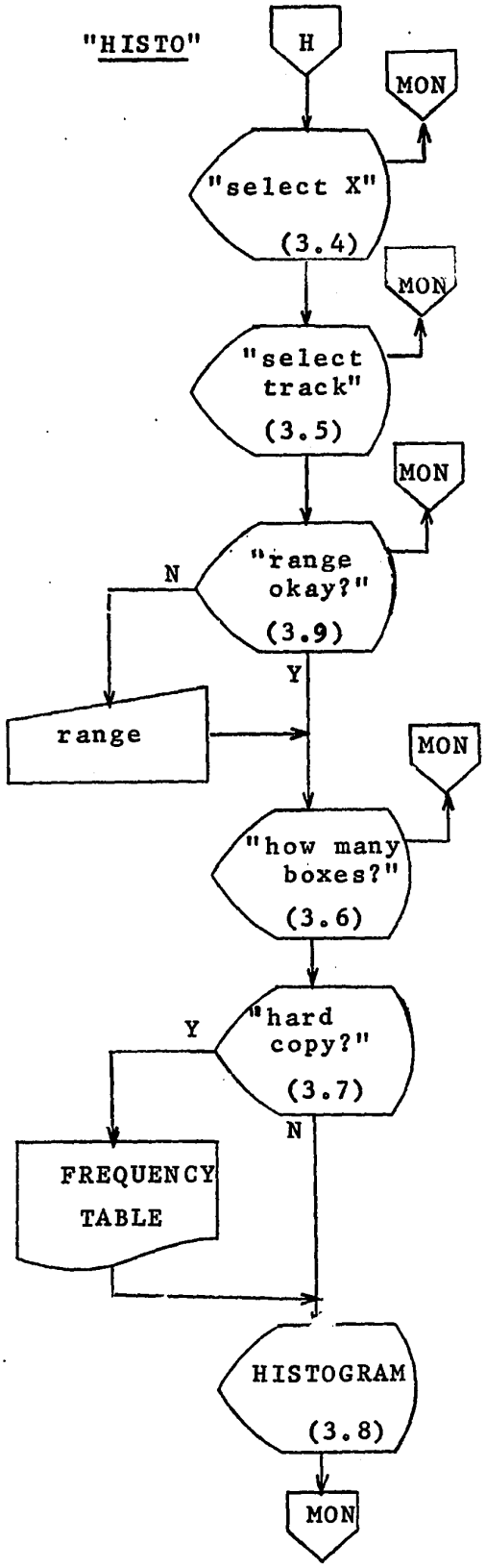


Figure 3.1

RAP - The regression analysis package.

(Part 1)



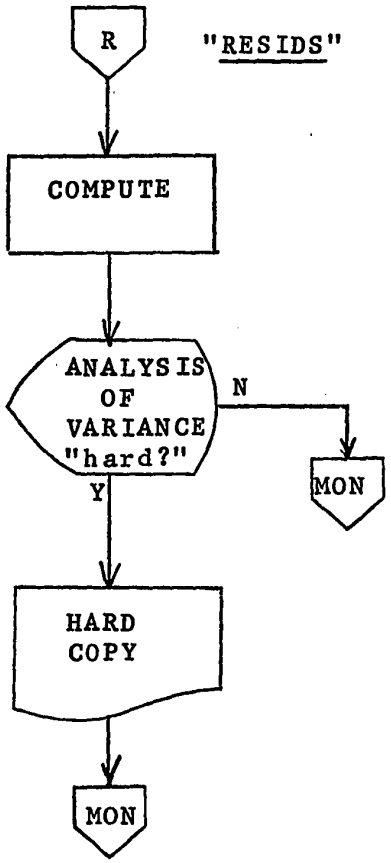


Figure 3.1

RAP - The Regression Analysis Package.

(Part 2)

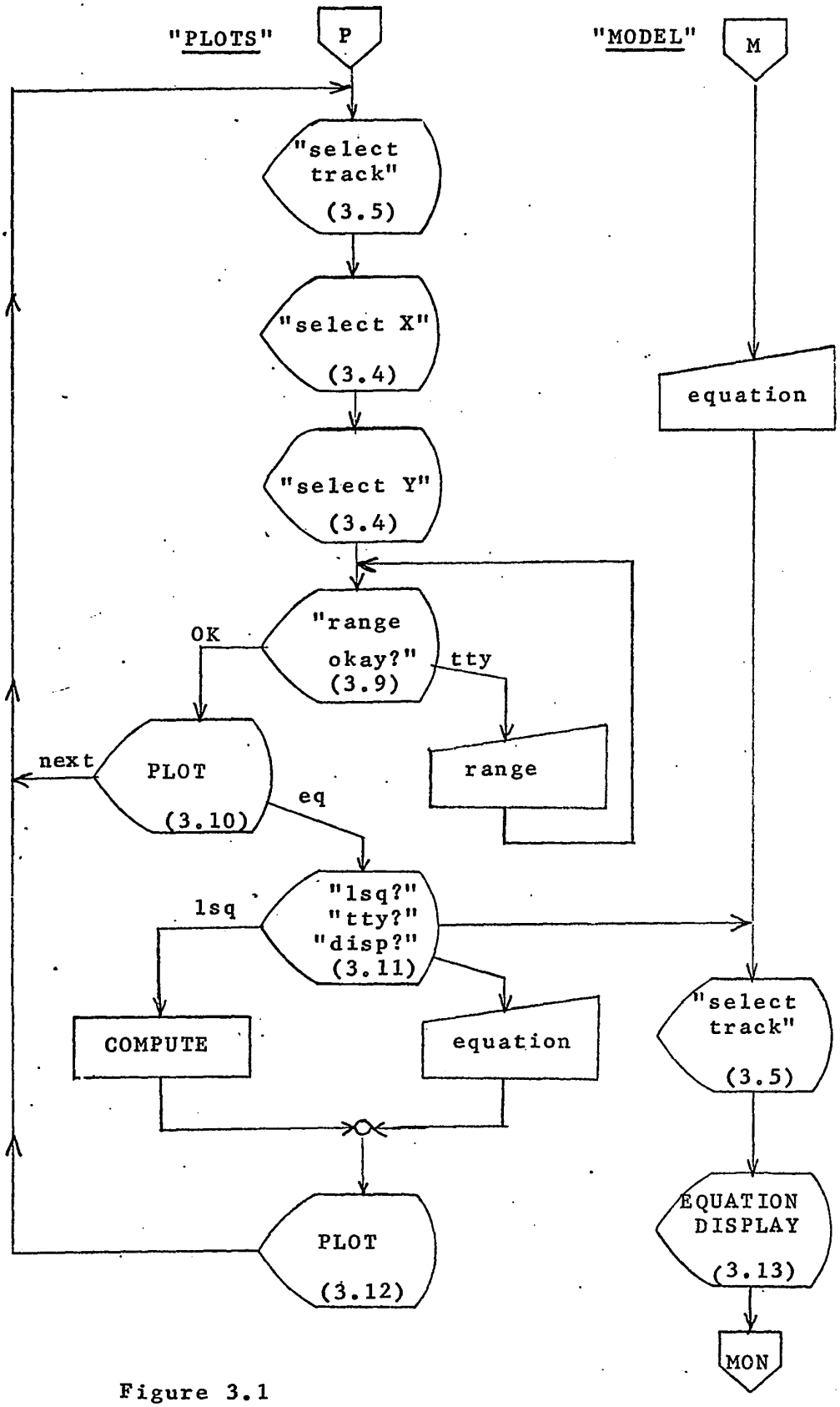


Figure 3.1

RAP - The Regression Analysis Package (Part 3)

in frame 3.4.

It is worth noting that this statistical information may be obtained about the residuals, once a model has been proposed, and that the correlation of any variable, or of the residuals, with time is also available.

Although the basic information provided by the two modules previously discussed is most useful, the user often needs to have some visual insight into the distribution of the variables. This may be provided by the histogram generator HISTO and by the plotting module PLOTS. After requesting HISTO, the user is invited to select the variable which he wishes to examine (Figure 3.4) and the track on which he wishes the histogram to be drawn (Figure 3.5). He must then decide the number of columns the histogram should include, up to a maximum of 16 (Figure 3.6). In each of these frames, the user has the option of restarting the monitor by selecting "RETURN". The frame shown* in Figure 3.7 is then displayed and the user may request that the frequency table be printed while the histogram is displayed (Figure 3.8). Hitting any part of the display with the light-pen has the effect of returning control to the stage of Figure 3.6 so that the user may request another histogram for the same variable.

The module PLOTS is a general plotting routine which allows the user to plot any two variables against each other. Time sequence plots are also available, as well as all the

* Photographs for page 47 could not be reproduced successfully.

residual plots mentioned in Chapter II. As in HISTO, the user is invited to select a track (Figure 3.5) and then the "Y" and"X" variables to be plotted (Figure 3.4). Both ranges are calculated and displayed (Figure 3.9). If the user selects "ACCEPT", the ranges of both axes are the total ranges of the variables plotted. By selecting "TTY INPUT", the user may choose his own scaling and type it on the keyboard. In this case the corrected frame of Figure 3.9 is displayed for a check; the plot will not be displayed until an "ACCEPT" choice is made. The plot, as shown in Figure 3.10, includes two options; "NEXT" restarts the module at the stage of Figure 3.5; "EQ" will be discussed later (Section 3.1.3).

3.1.3 Choosing a model.

The user has three options for the definition of a model. The first one is to request the module MODEL from the monitor. This module prints a format guide on the teletype, so that the user can easily type in a linear equation of his choice. The equation is then displayed for a check (Figure 3.13).

The second way to define a model is to select "EQ" after a plot of Y versus X has been displayed (Figure 3.10). The frame shown in Figure 3.11 is then displayed. The option "TTY INPUT" enables one to type in the coefficients a and b of a tentative model $y = ax + b$, in much the same way as described above. By selecting "LSQ", the user requests a univariate least square fitting of Y against X. In both cases, the

plot of Y versus X is restored and the model line is superimposed (Figure 3.12). The user may then restart the module PLOTS by selecting "NEXT", in order to initiate another plot; else, he may try another equation for the same plot by selecting "EQ" again. The third option in Figure 3.11, "DISPLAY" is used when one wishes to display the latest equation chosen (or computed by the least square routine). This display is stored on the track chosen by the user and it includes the model's identification number.

The third way of defining a model is by calling the module BIVAR. Once the user has selected the dependent and both independent variables, a bivariate least squares fitting is performed and the resulting equation is displayed on the track chosen by the user.

3.1.4 Evaluation of the model.

The quantitative evaluation of the model is performed by the module RESIDS. The residuals are computed and stored in an additional column of the data matrix. The module also carries out an analysis of variance and its results are displayed on the screen, along with the equation's identification number. The display includes a "HARD COPY" option; if this option is ignored, control is returned to the monitor; otherwise, all numerical results (sums of squares, degrees of freedom and F-value) are first output on the line-printer.

Visual examination of the model is possible thanks to the two modules previously discussed, HISTO and PLOTS, which can be applied to the residuals. Significant departure from normality may be detected on a residual histogram; a particular trend in time or the need to include a new variable or a nonlinear term in the model will be shown in plotting the residuals against time and against all independent variables, as discussed in Chapter II.

3.1.5 Transformation in the variables.

By selecting TRANSFORM in the monitor, the user can create an additional variable Z as a simple arithmetic function (fun) of two pre-existing variables X1 and X2. He first chooses the number of the new variable Z (Figure 3.4); then the same frame is produced twice more, for the selection of X1 and X2; the function is selected from another frame. The new variable is then computed as

$$Z = X1 \text{ (fun) } X2 \ ,$$

where (fun) represents one of the operators: add, subtract, multiply, divide. This is not limitative and other functions such as sin, cos, log and exp might be easily added to the system when needed. A few extra columns should be reserved when the data matrix is built up, in order to provide storage space for the transformed variables.

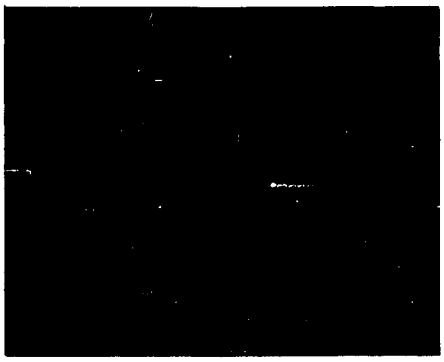


Figure 3.2

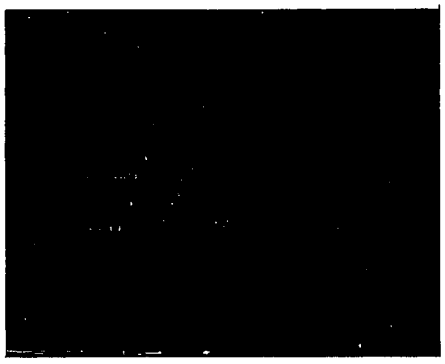


Figure 3.3b

Figure 3.4

Figure 3.5

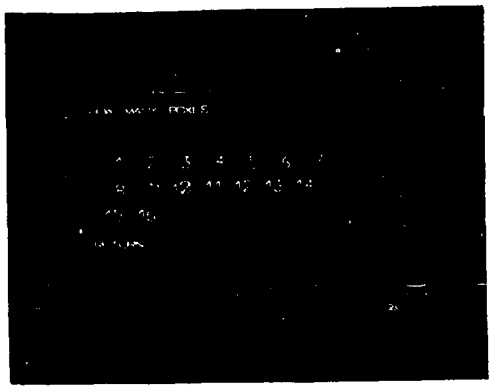


Figure 3.6

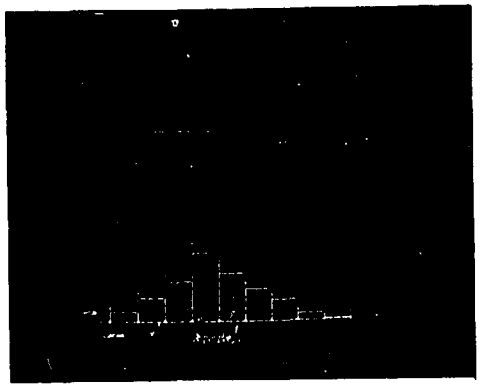


Figure 3.8

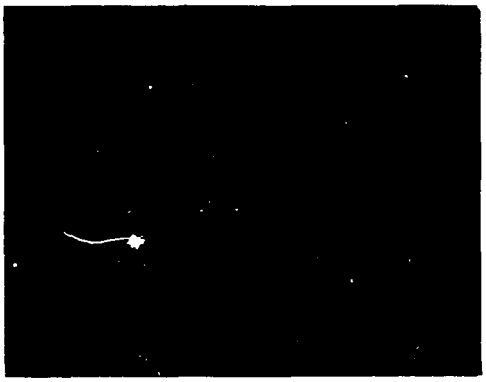


Figure 3.9

Figure 3.10

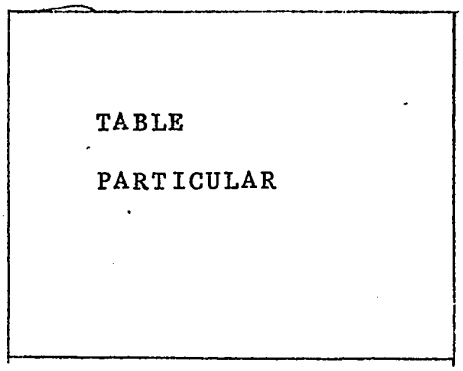


Figure 3.3.a

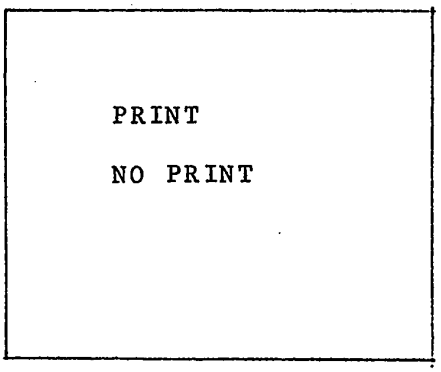


Figure 3.7

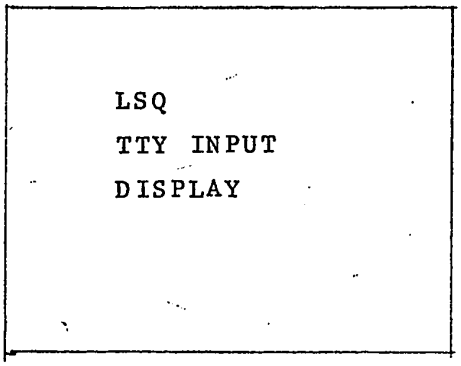


Figure 3.11

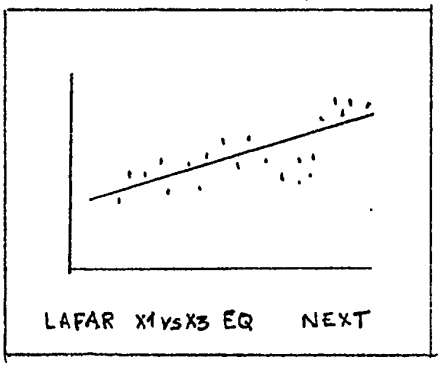


Figure 3.12

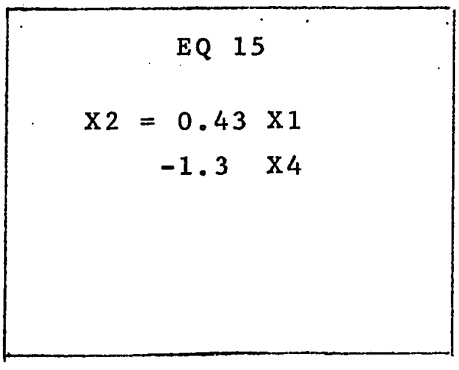


Figure 3.13

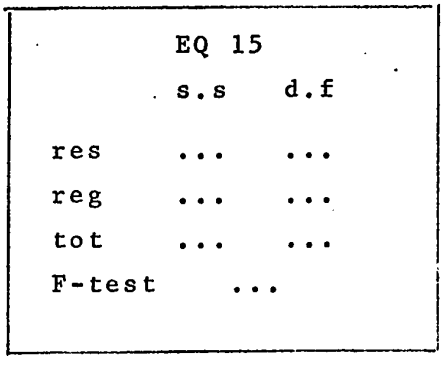


Figure 3.14

3.2 The equipment used.

The graphics unit used for this research is of the disc-refreshed type and was designed in the Department of Electrical Engineering at McGill University (53,54). The lay-out of the computer facilities available is shown in the block diagram of Figure 3.15. The main computing tasks, including the assembly of the display files, are performed by the PDP-15/20. The assembled instructions are transmitted, via the Data Buffer, to the PDP-8. The latter writes the instructions onto the graphics disc buffer, through the Disc Interface. The Display Processing Unit (referred to in the following as the DPU) decodes the information contained on the disc and monitors the electron beam of the CRT. Graphical input functions are performed by the light-pen and the joystick and their respective processors. In the following, each of the components mentioned above will be discussed in detail.

3.2.1 The computers and their peripherals.

3.2.1.1 The PDP 15/20.

The memory of this 18-bit machine has been extended to 24K in the course of this project. A 512K disc storage has also been adjoined to the two existing tape drives. This disc is used as a working storage only, as the various users

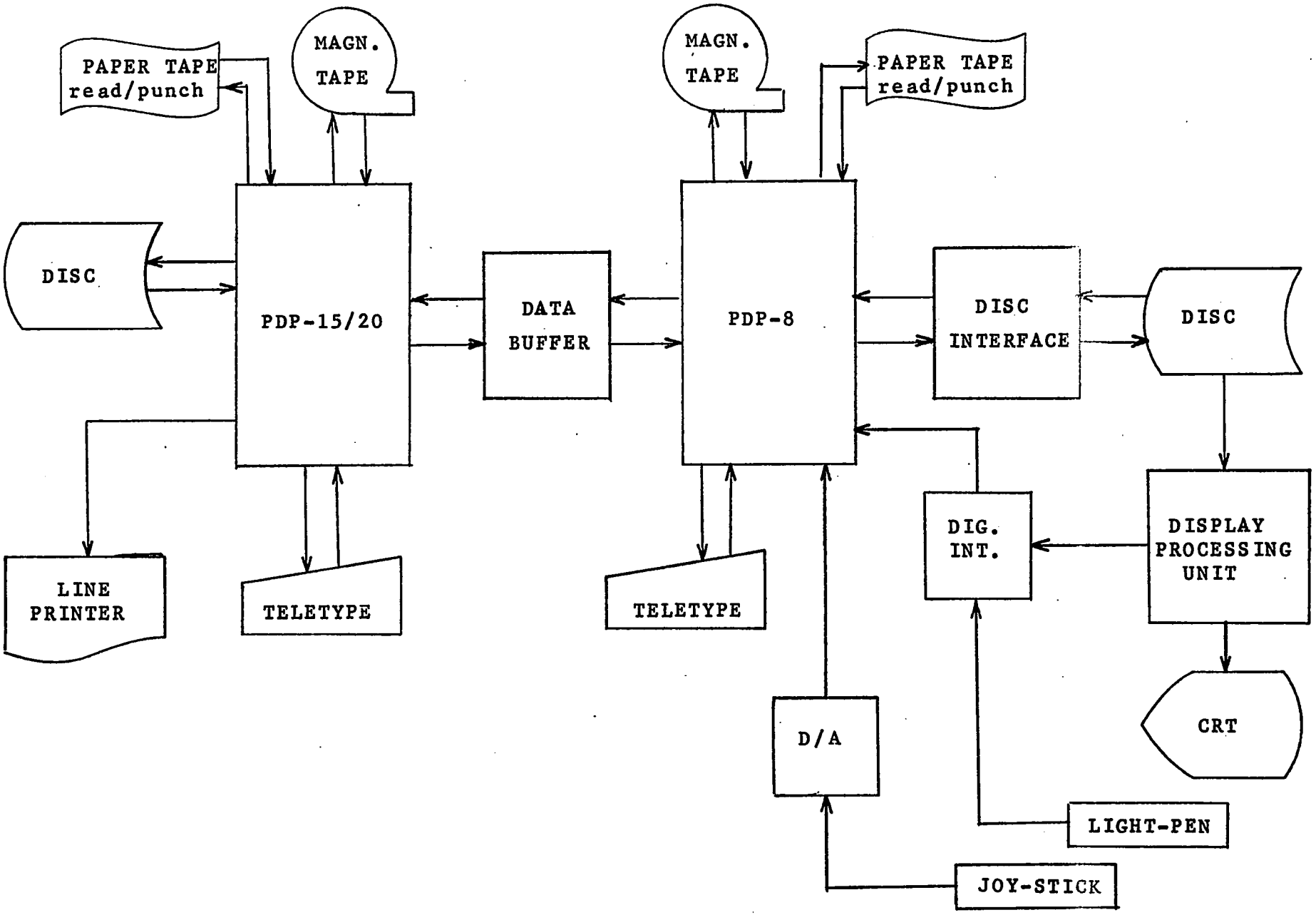


Figure 3.15 The computer facilities used.

save their own programmes on magnetic tapes. Other peripherals include a high-speed paper tape reader/punch, a Teletype keyboard-printer unit and a VERSATEC model 600 electrostatic line-printer.

The PDP-15 is used in this project as a background processor of programmes written in FORTRAN, linked with a few subroutines written in MACRO assembler language. The dialogue with the PDP-8 foreground processor is carried out via the Data Buffer (3.2.1.3). The teletype performs those input tasks which are impractical for the light-pen and is also used by the programme to print out a number of informative messages during the execution; these messages may be requested at execution time by using the console switches, as will be explained in 4.1. The fast line-printer is used when a hard copy summary of the information displayed on the CRT is desired.

3.2.1.2 The PDP-8.

This computer has a 12-bit accumulator and its core-size is only 4K. It is therefore not used for main computing tasks but, rather, to monitor a number of peripheral facilities. It is linked to the PDP-15 through the data buffer and to the disc interface (3.2.2.2). It services the light-pen through a digital interface and the joy-stick through its analog/digital converter. Other connections, not used in

this particular project, include an optical scanner and a high speed data link between the PDP-8 and the McGill Computing Center's IBM 360/75.

The PDP-8 peripherals include a teletype with slow paper tape reader/punch, two DEC-tape drives for the storage of system and user programmes and a high-speed paper tape reader/punch.

All programmes to be executed by the PDP-8 have been written in PAL-D assembler language.

3.2.1.3 The Data Buffer.

The data buffer allows the transfer of a binary word either way between the PDP-15 and the PDP-8. This transfer is done from accumulator to accumulator; because of the difference between the lengths of the two accumulators, only the 12 lower order bits (rightmost) are transmitted. The control is performed by means of a few assembler instructions; load accumulator into buffer, read buffer into accumulator, skip on "busy" flag. As the buffer capacity is only one word, it is impossible to transmit one word until the receiver has read the previous one.

3.2.2 The graphics facilities.

3.2.2.1 The Graphics Disc.

The disc used for the storage of the graphics information is a DATA DISC FPD 16. It contains one clock track and 16 user tracks which can store about 100,000 bits each. The information is fed to the disc and retrieved from it by the disc interface discussed in the next section. When the graphics unit is switched on, the information contained on one track is constantly read and transmitted to the Display Processing Unit (3.2.2.2) which processes it and draws the picture on the screen. The display is thus refreshed at every revolution of the disc. The refreshing rate (30 r.p.s) is certainly sufficient to produce a flicker-free image, with the type of phosphor used (P-31). The track being read is selected either manually on a track selector switch or by software (see subroutine LOOK in section 4.3). There is thus provision for a large number of different displays to be stored simultaneously on the sixteen tracks.

3.2.2.2 The Disc Interface.

The disc interface performs the read/write tasks between the PDP-8 core and the graphics disc. To perform either action, a number of registers have to be loaded with the necessary data by means of a set of specific assembler instructions. These data are the following:

- the address where the first word is to be taken (or stored) in the PDP-8 core;

- the total number of words to be written (or read);

- the track number and starting address on the track;

- the mode of transfer (read or write, length of word, serial or parallel; the latter mode specifies whether the words are 8-bit or 12-bit. There also is an additional option; the words can be written serially on one track or in parallel on 12 contiguous tracks. Graphics instructions are stored in serial words of 8 bits, whereas the 12-bit parallel mode is used, in other projects, for video monitoring.

When the operation is completed, the interface notifies the PDP-8 by interrupting it. More details about the disc interface, which was designed in this Department, may be found in (54). The particular software module which handles the disc interface in this project will be discussed in Chapter IV.

3.2.2.3 The Display Processing Unit.

Full details of the design of the DPU are given in references (53) and (54). We shall give here only the minimum information necessary for the understanding of the system.

The DPU allows two modes of drawing, point-plotting and vector-drawing. Points may be plotted in two different ways. In the absolute mode, the beam position is defined by its

(X,Y) coordinates; in the incremental mode, the positioning is done in reference to the current position, by specifying the changes in coordinates: ΔX, ΔY. This latter mode allows a substantial saving of code for the numerous applications where moves are relatively short (e.g. the plotting of a curve or of an alphanumeric character).

Vectors are all drawn in the incremental mode. The vector generator is of the fixed-time type (as opposed to the constant rate type) (22); any vector, regardless of its length, is drawn in 2.1 microseconds. To prevent shorter moves from looking brighter because linear drawing rate is lower, an automatic intensity compensation circuit has been included.

Overall intensity is controlled through a circuit which allows 15 levels of brightness and can monitor up to four black-and-white CRT's or else one colour and one black-and-white CRT.

The SCALE unit permits magnifications ranging from 1 to 128 and also allows the inversion of an image. In scale 1, 1024 elementary moves are possible in each of the X and Y directions. Any point on the screen may therefore be defined by two coordinate words of ten bits each.

The BRANCH circuitry permits switching the processor to a new track; this switching can be requested to take place either immediately after the proper instruction has been

decoded or at the next track origin (the beginning of the next revolution). A track selector switch allows manual track-switching as well. If the switch is set on "CONTINUOUS", the DPU executes all programmed branching instructions but, at every track origin, it restarts the track indicated by the track selector. If the switch is "OFF", branching will be entirely assumed by the software instructions.

Another circuit in the DPU processes the HEADER instructions. One such instruction is placed at the beginning of every file and identifies it by means of a file name. One of its bits specifies whether the file is to be actually displayed or not; if this bit is off, the picture is drawn but the beam is not illuminated and all programmed branching instructions contained in the file are ignored.

3.2.2.4 The CRT and graphical input devices.

The CRT is a 10 x 12 inch DUMONT display equipped with a P-31 phosphor. The light-pen is a Saunders make; it is linked to the PDP-8 through a digital interface (55). It operates as follows. Every time the DPU decodes a new HEADER instruction, the corresponding file name is instantly loaded into the File Name Register. Within one refresh cycle, the contents of the register are thus updated as many times as there are different files. When the light-pen is pointed at one element of the display, it will intercept the beam the next time

this latter refreshes that element and, within an extremely short time, it will interrupt the computer. Upon identification of this interrupt, the contents of the File Name Register are then loaded into the accumulator so that the file which is pointed at can be recognised. The whole procedure is based on the fact that the response time is short compared to the 2.4 microseconds necessary to execute one move on the screen.

The joy-stick, manufactured by Digital Equipment, is not used in this project.

CHAPTER IV

THE GRAPHICS INTERPRETER

Much of the time spent on this project has been devoted to the design of a software system which would permit a convenient handling of the graphics facility. Because the equipment used is fully dedicated, it has been found suitable to design the software support as a user-oriented interpretive package, mostly in the form of FORTRAN subroutines. The approach which has been taken starts from the lowest levels(one subroutine for each of the basic instructions acceptable by the Display Processing Unit) and progressively broadens the scope, to permit higher level programming. The reverse order has been chosen for the presentation which follows. Higher level routines will be discussed first but this will include all necessary references to the lower structures defined further.

4.1. Generalities.

The writing of a graphics programme is composed of a number of calls to special subroutines which may in fact be inserted among other FORTRAN instructions. The function of each routine is to produce one or several graphics instructions readily acceptable by the Display Processing Unit and to stack

them in a software "instruction buffer" which has a capacity of 1,000 instructions. When the buffer is full, its contents are automatically transmitted to the PDP-8; this transfer may also be requested before the buffer is full, by calling ENDGO (4.2).

The switches on the PDP-15 console may be used by the programmer to affect the programme flow at execution time. It is widely used in this project to by-pass the printing of a large number of messages which give information on the execution of the programme(e.g. one switch enables the printing of all the file indentification as soon as its header is processed; when another is up, the disc location of the last word written by ENDGO is printed). This feature has been implemented and extensively used during the development of the system and has been kept because of its potential usefulness in the training of a new user or in the debugging of some application programme.

4.2. Organisation of a file.

A new file is initiated by the following statement:

```
CALL HDR (file, display, interrupt)
```

where

- "file" is the octal number (0 to 7777) identifying the file,
- "display" is a boolean variable equal to 1 if the file

is to be actually displayed;

- "interrupt" is another boolean variable equal to 1 if the interrupt facility is enabled for this file.

The functions of the routine HDR are not only to issue a HEADER graphics instruction but also to perform some book-keeping. If the file name has not yet been used, a new entry is prepared in the book and includes, along with the file name, the exact position of the header on the disc(track and address) the 8-bit header instruction itself and the display status of the file(on or off).

Once a file is residing on disc, it may be switched on and off by calling

CALL DISPLAY (action, file)

where "action" equals 1 for switching on and 0 for switching off. The table is scanned until the file name is found; the status is checked and, if it is not as requested, the HEADER instruction is located on the disc and adequately changed.

The position of the HEADER on the disc is determined by means of two software registers, the track register and the address register. The track register is set to the desired value by the programmer at any point in the programme and conserves the given value until it is set differently. If

it is omitted, the files are assumed to lie on track 0. The address register may also be set by the programmer but it is updated every time the buffer is emptied so that when a header is called, its position is automatically determined knowing the number of instructions then in the buffer.

The file does not need to be closed, since the header of the next file is a terminator in itself. However, if one desires a clean display the last file on each track must be terminated with a BRANCH statement. This will have the effect of inhibiting the DPU for the remainder of the track, in order to prevent any leftover information from being interpreted as a header, thus causing unwanted display. This cautionary step is automatically taken by calling subroutine ENDGO after the last instruction of the file. The functions of this subroutine are

- to terminate the file with the adequate BRANCH.
- to store the position on the disc of the said BRANCH.
- to initiate the unloading of the buffer into the PDP-8 and then onto disc at the proper location.

Immediately after ENDGO, the programme should contain one of the three calls FORWD, BACK,RESET which specify by updating the address register where the next writing is to be performed.

- CALL FORWD will load the address register with the location of the last word written (BRANCH) so that the next time

ENDGO is called, the BRANCH instruction is deleted and the file is continued smoothly.

- CALL RESET resets the address register at the beginning of the current track.

- CALL BACK keeps the address register at its current value so that the next writing will be initiated exactly over the previous one. This is used when part of a display is fixed while one element is continuously replaced.

All three routines restart the buffer. Their use is illustrated in figure 4.1.

4.3. Organisation of the disc.

Before writing the first file of a given track, the track must be initiated by calling INIT. The effect of this routine is

- to initialise the buffer,
- to prepare the track for track switching operations
- to issue a minimum of standard instructions permitting subsequent displays. The latter instructions include the clearing of the X and Y registers and the setting of the SCALE and INTENSITY registers at their most commonly used levels.(respectively 0 and 15).

The track switching system makes use of the first four

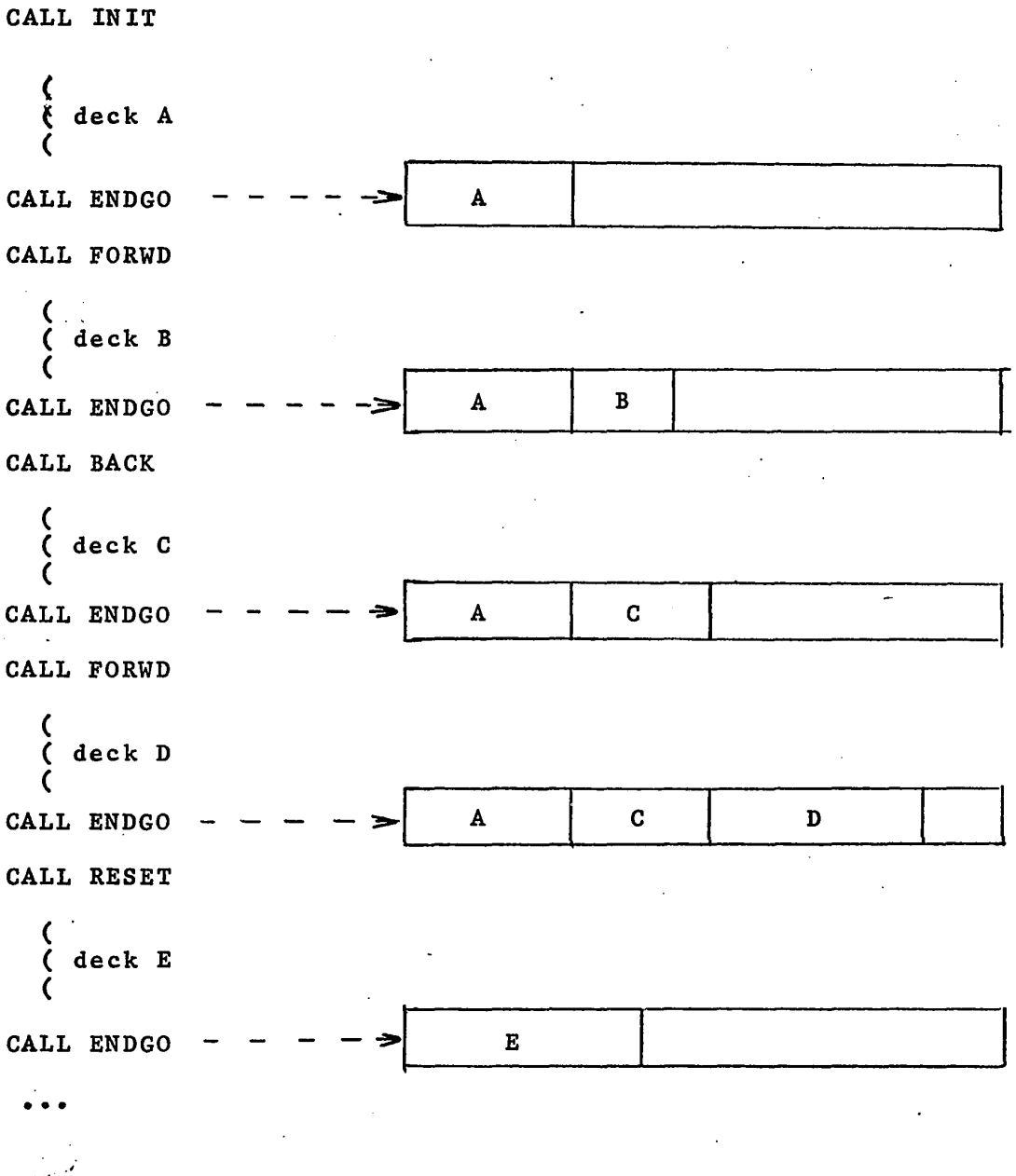


Figure 4.1 The positioning of a file on a track.

locations on each track. The track selector must be set on CONTINUOUS and indicate track 16. In this manner, the DPU restarts every revolution on track 16 and executes all programmed BRANCH statements. If one wishes that track i be displayed at some point in the programme, one inserts

CALL LOOK (i).

This routine writes the proper BRANCH statement at the beginning of track 16 so that the DPU starts decoding track i immediately after track origin (Figure 4.2). Independently of this system, one may jump from track to track within one revolution by calling

CALL BRANCH (j,wait,interrupt).

- j is the track to which the DPU should jump.
- "wait" indicates whether the jump is to be performed immediately (0) or at the next track origin(1). (Here,"wait"=0).
- the interrupt action is normally ignored (interrupt = 0).

This may be used to compose a display from several elements stored on different tracks but because of the necessity to synchronize the various files this method makes a wasteful use of disc space. It is much more efficient to keep all the elements on one single track and to switch them on and off by means of the DISPLAY routine.

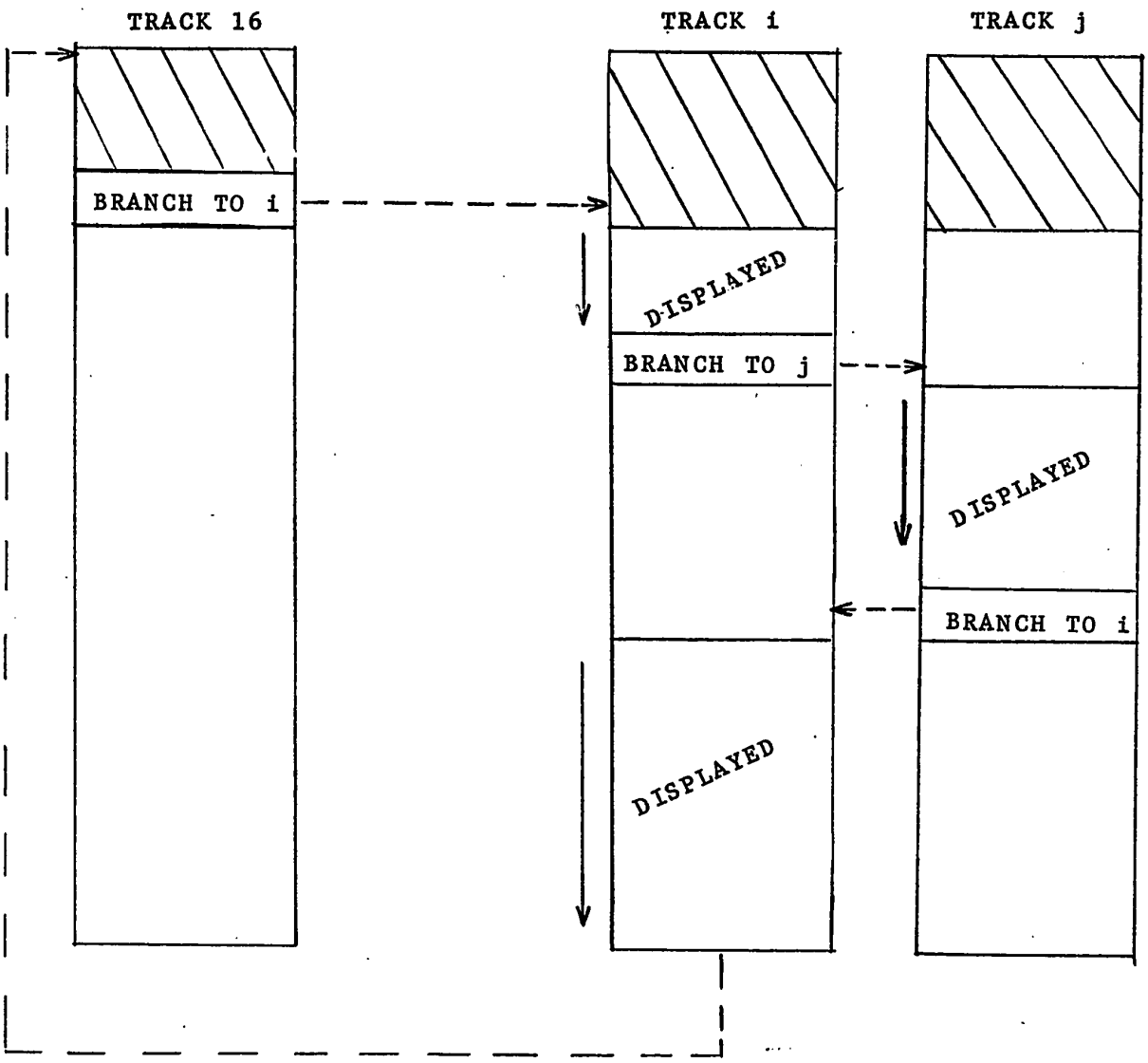


Figure 4.2 The track-switching mechanism.

4.4. Light-pen handling.

4.4.1 Preparing a light-button

Each item of a light-button is created by

CALL LB (group,file,out, text)

- "group" is the identification tag carried by all the items of the same group.
- "file" is the file name of the particular item.
- "out" is the numerical output which one wishes to associate with the item.
- "text" is the 5-letter Hollerith string to be displayed on the menu, starting at the current position of the beam.

Before defining the first item of a light-button, the programmer should specify on which track the light-button is to be stored. A table is made of all the items of a group and contains all the necessary information for the use of the light-button. The same file may be referred to in up to three different light-button groups, without having to be repeated on the disc. Any instructions following a LB call (additional text or picture) will belong to the same file. Only another LB call or HEADER will initiate a new file.

4.4.2 Using a light-button.

A light-pen selection is initiated by

CALL LPSELX (group,out).

The functions of this routine are

- to identify all the item files belonging to the given group and to switch them on (by means of DISPLAY);
- to switch the display on the proper track (LOOK);
- to wait for the user's action with the light-pen (LPEN);
- to identify the file selected with the light-pen, retrieve the corresponding output from the LB table and store it as the argument "out" of the routine.

An example of light-button use is illustrated in figure 3.4.

4.5. Writing aids.

4.5.1. Character generation.

The sets of instructions which constitute the drawing of each character are stored sequentially in a MACRO language programme (CHAR). The first word of each set is the length of the set, available for the handler routine which resides at the beginning of the programme. The input to subroutine CHAR is a code identifying the character called; the output is an array containing the corresponding graphics instructions.

The characters have been designed so that in scale 0, they

are about ¼ inch high. The generator contains all alphanumerics plus a few special characters. A semi-colon is arbitrarily used to specify that a new line should be started. The # symbol is used for filling blanks in a Hollerith string.

In the programme, a character may be called individually by

CALL TYPE (n)

where n is the order of the character in the storage sequence. The one to one correspondence is readily available through a simple arithmetic manipulation of the ASCII code.

A higher level of programming is possible by using

CALL PRINT (text)

where text is a five letter Hollerith string which should have been predefined somewhere in the programme. The routine unpacks the string and successively calls subroutine TYPE for each of the five characters.

4.5.2. Handling of numbers.

Two subroutines are available for the display of numerical variables on the CRT. Both of them initiate the display at the current position of the beam, as do TYPE and PRINT. An integer variable i will be displayed by calling IDISP (i). A variable format is available for the display of real numbers.

A real variable X is displayed with n decimal places by calling RDISP (X,n).

4.6 Drawing aids.

A number of subroutines have been written to assist the user in his graphical designs. This package has been set up progressively as a function of the needs of the project and is to be expanded.

4.6.1 Plotting package.

As graphics are often used to plot data, it is convenient to have a general plotting technique applicable to as many formats as possible. This has been realised in the following way. The total screen (but for narrow margins) is supposed to be the frame of an Y vs X plot. Four software registers hold the boundary values of each coordinate. The programmer may change these values at any point in his programme. A statement

```
CALL DOT (X,Y)
```

will cause a dot to be plotted at the correct position, with respect to the frame defined by the current contents of the registers mentioned above. If one ordinate falls off-range the dot is arbitrarily plotted in the corresponding margin, at the correct value of the other ordinate.

A set of axes may be drawn in the current frame, by

CALL AXES (X_o, Y_o)

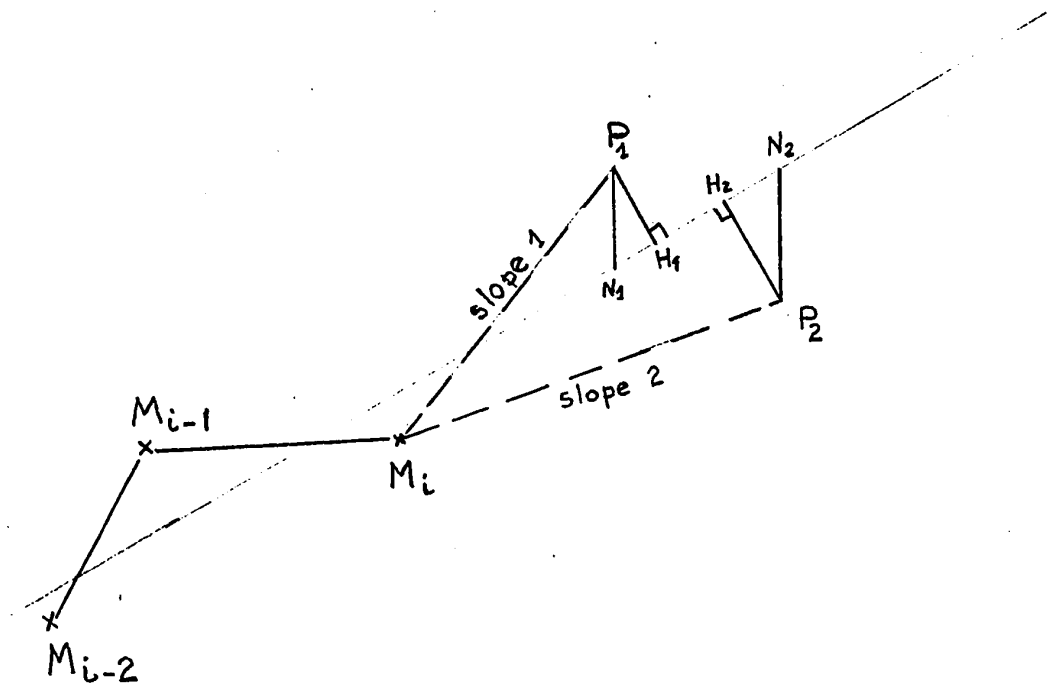
where X_o, Y_o is the origin point.

4.6.2. Drawing of a straight line.

The subroutine JOIN is used to join two points defined in the current frame by a straight line. The calling statement is

CALL JOIN (X1,X2,Y1,Y2)

Because of the discrete aspect of the drawing system, only a limited number of slopes are available. In fact, when drawing a vector, the x and y components can be chosen among seven values.(see section 4.7.1). The number of distinct slopes is therefore thirty-six per quadrant. An algorithm must be used to approximate the required line by the "closest" broken line made of available slopes. The criterion used in this algorithm is the one of the least distance between the desired line and the approximated line. For every new segment, the two slopes which bracket the desired line are first determined. The positions of the two possible end-points are calculated and the one closer to the desired line is chosen. The corresponding vector instruction is issued and the current position updated. The algorithm is illustrated in Figure 4.3.



By similarity $$\frac{P_1 N_1}{P_2 N_2} = \frac{P_1 H_1}{P_2 H_2} .$$

The algorithm minimizes the vertical distance $P_j N_j$ instead of the perpendicular distance $P_j H_j$, as the computations involved are much easier.

Figure 4.3 The algorithm joining two points by a line.

4.7. The basic instructions.

In the following, we shall describe the lower level subroutines of the system. They are often called indirectly, through the higher level subroutines discussed in the previous sections. They include the elementary graphics instructions and also a number of service subroutines.

4.7.1. Elementary graphics.

As described in 3.2.2.3, the Display Processing Unit accepts a limited number of instructions classified into two different modes. Their bit format is shown in reference (53) and a subroutine has been written for each of them.

4.7.1.1 The COMMAND mode.

In addition to the two instructions which have been already discussed (HEADER and BRANCH), this mode contains four "operate" instructions which are used to switch the DPU into one of the four DATA modes described in the next section. They are the following:

EPM Enter Point Mode

EVM Enter Vector Mode

EZM Enter Intensity Mode

ESM Enter Scale Mode

4.7.1.2 The DATA Mode.

Four instructions belong to this group: POINT, VECTOR, SCALE, INTENSITY and are issued by the corresponding subroutines PNT, VEC, SCL, ISTY. They have in common a number of features which are handled by the DATMOD subroutine. This routine is called at the beginning of the execution of each DATA routine. Its flow chart is shown in Figure 4.4 and its functions are the following:

- to check the mode in which the DPU currently is. If it is not the correct one, a " return to command mode" and the appropriate " operate" instruction are issued;

- to handle the data register in the way required; the increment may be added to or subtracted from the current contents (incremental mode) or else the register may be cleared first and then the increment loaded (absolute mode); for subroutines PNT and VEC, this applies to both the X and the Y registers;

- to check that the increments specified are within acceptable limits i.e.:

-2^{10}	to	2^{10}	for PNT,
- 7	to	+7	for VEC,
0	to	15	for ISTY,
0	to	7	for SCL

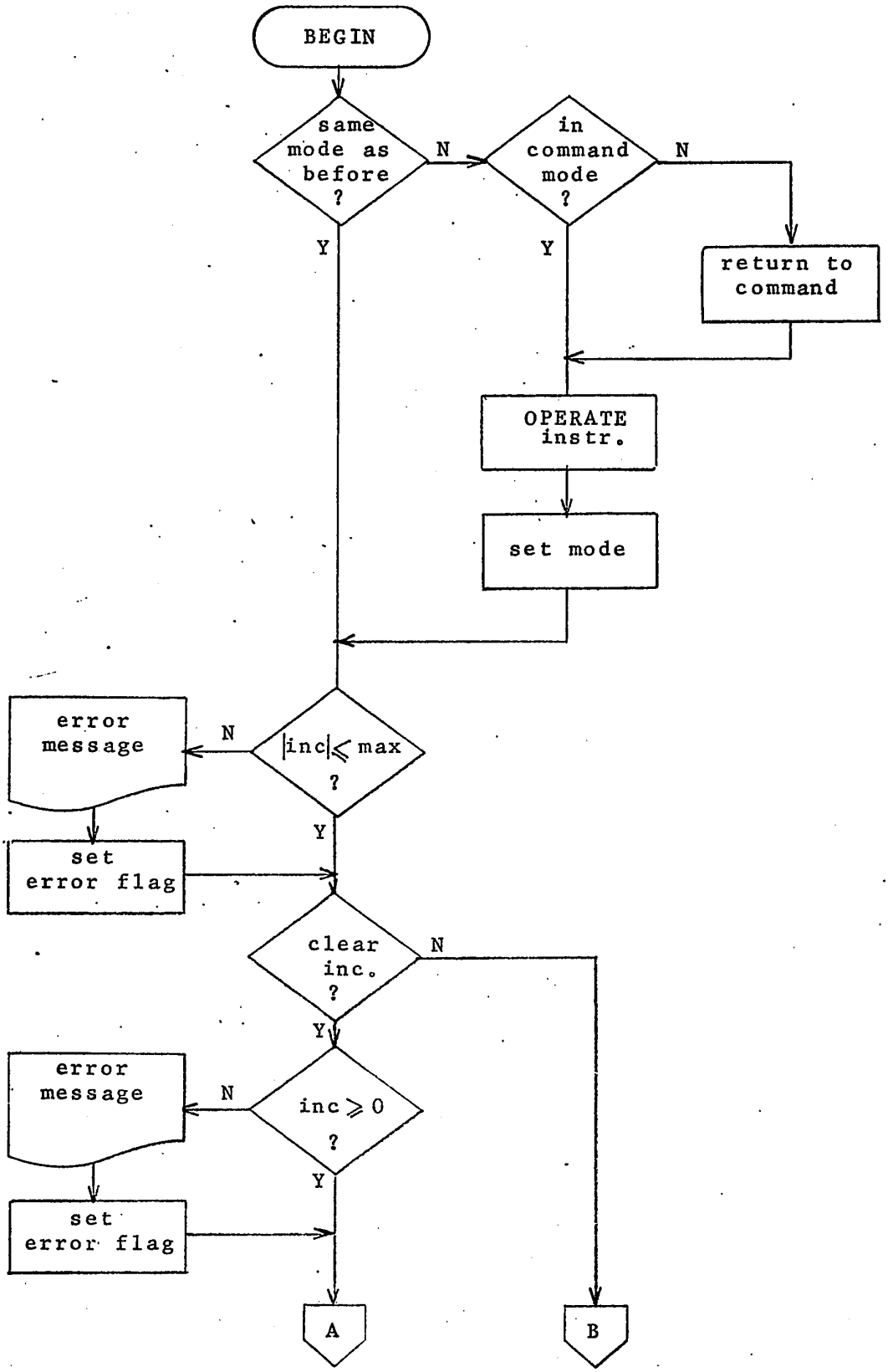
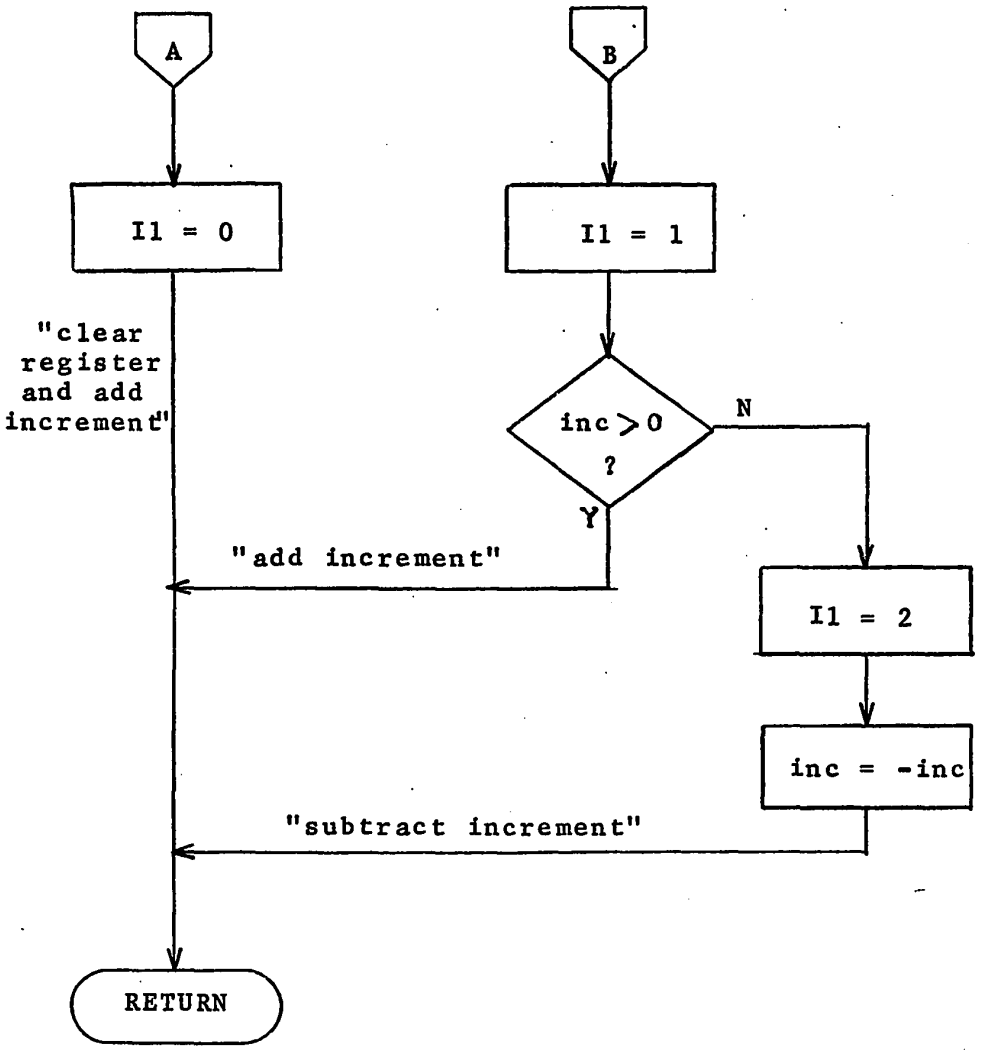


Figure 4.4 The DATMOD subroutine (Part 1)



Note. I1 is the number formed by the bits 8 and 7 of the data instruction.

Figure 4.4 The DATMOD subroutine (Part 2)

Should an error be detected, an explicit message is printed out and the DEPOSIT and SDFILE facilities (4.7.2) are disabled. There is no disabling, however, if the erroneous instruction is graphically meaningful (e.g. a zero-long vector).

The PNT subroutine

CALL PNT (CLR, X, Y, D)

separately processes the X and Y moves. If the boolean argument CLR is equal to 1, both registers are cleared before the increments are loaded (absolute mode) whereas CLR = 0 calls for the incremental mode. If the boolean argument D equals 1 the point is illuminated after the move has been effected. Two kinds of moves are available: a short (3 bits) move requiring one instruction and a long (10 bits) move requiring two instructions. According to the values of X and Y, the subroutine automatically chooses the correct length. If either X or Y is zero, the corresponding instruction is ignored; one has the ability however of making a dummy zero move by requesting both X and Y to be zero. This has been found necessary as a software "patch" and is automatically effected if a long move is initiated after clearing the registers. The flow-chart for PNT is shown in figure 4.5.

The VECTOR subroutine

CALL VEC (CLR,X,Y,D)

is very much similar to the PNT subroutine. A "short" move is one in either the X or the Y direction, whereas a "long" move is a diagonal one. If CLR equals 1, both registers are cleared except in the case where one of the two increments is zero; in such a case, only the non-zero ordinate is cleared before adding the increment and the move is considered a short one. The flow-chart for VEC is shown in figure 4.7.

The SCALE subroutine

CALL SCL (CLR,X,Y,INV)

permits different scalings in each direction. The normal scale, in which the full screen corresponds to 2^{10} bits in each direction, is scale 0. The boolean argument INV may be set to 1 if an inverted picture is desired.

The INTENSITY routine

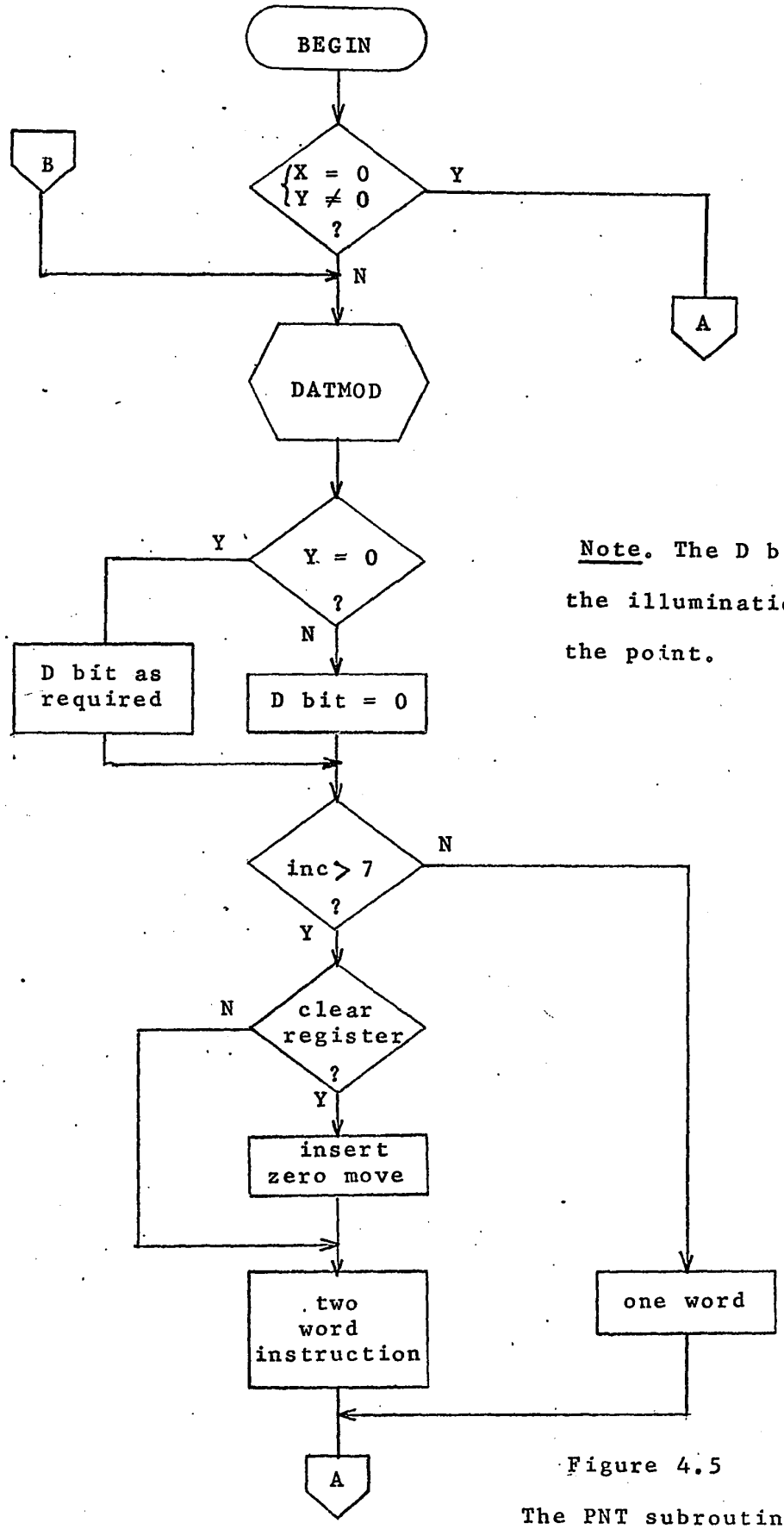
CALL ISTY (CLR,INT,CRT)

allows 15 levels of brightness. Four CRT units may be monitored, corresponding to values 0,1,2,3 of argument CRT.

4.7.2. The service routines.

These routines are mainly concerned with the handling of display files.

Subroutine DEP (name) allows the saving of the contents



Note. The D bit controls the illumination of the point.

Figure 4.5

The PNT subroutine (Part 1)

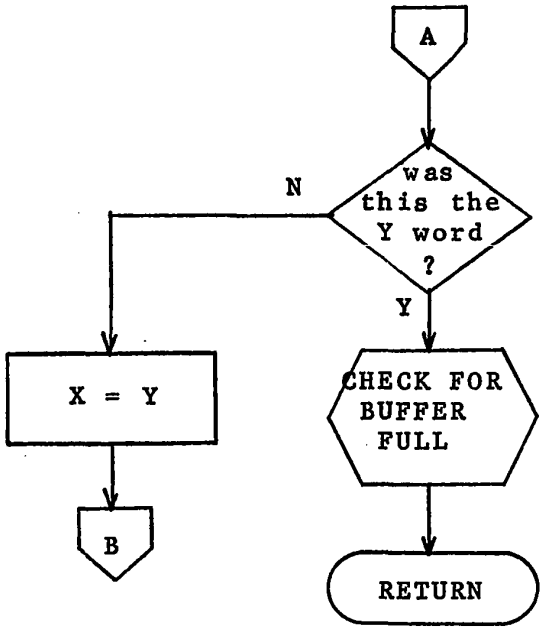


Figure 4.5 The PNT subroutine (Part 2)

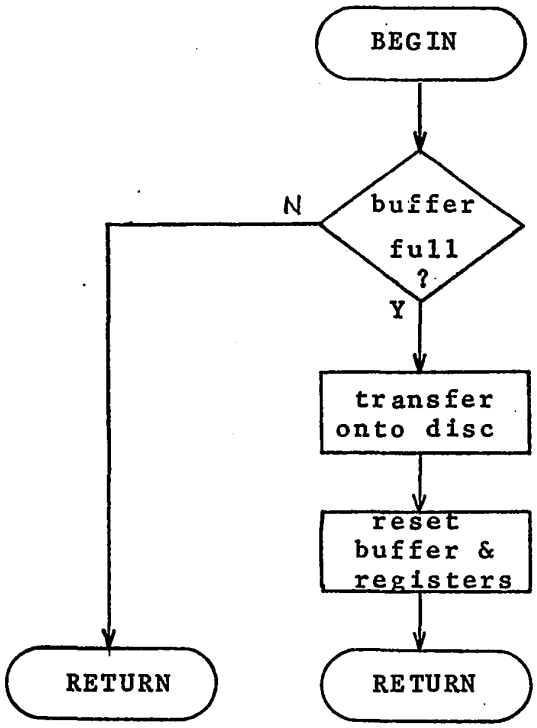


Figure 4.6 Checking for instruction buffer full.

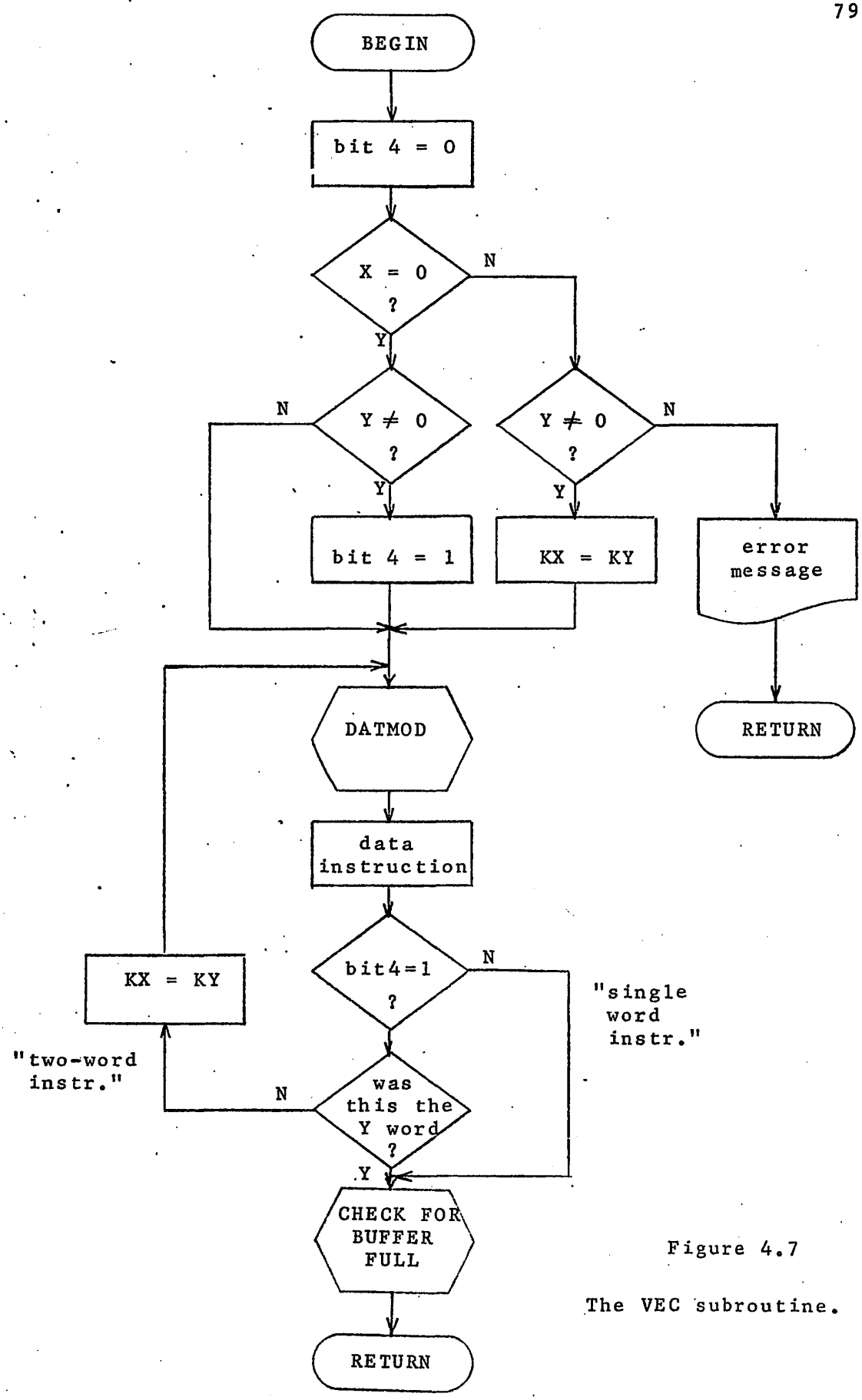


Figure 4.7

The VEC subroutine.

of the instruction buffer on the PDP-15 disc (or on DEC-tape) under the file name "name". It may be called at any point in the programme.

Subroutine LIST prints the current contents of the instruction buffer in the form of an array of octal instructions wearing mnemonic tags indicating the type of instruction (PNT, VEC, etc).

These two latter routines were very useful for debugging the system and they may constitute a useful instructional aid for a new user of the system.

Subroutine SDFILE (name, CA,WC) transmits the file "name" from the PDP-15 to the PDP-8. CA is the core address in the PDP-8 where the storage should be initiated and WC is the total number of words to be transmitted. If "name" is specified as "CURRT", the current contents of the instruction buffer are transmitted. Otherwise, a display file should have previously been saved under "name", by using subroutine DEP. This routine SDFILE is called by the higher level routine ENDGO.

Subroutine WS8 (CA, WC, T, DA) initiates the writing of the previously transferred file (core address CA, length WC) from the PDP-8 onto disc (track T, location DA). This routine is also called by subroutine ENDGO.

Subroutine CLR allows the filling with zeros of the track indicated by the track register (as explained in section 4.2). This subroutine may be called by subroutine INIT if requested by means of a console switch.

Subroutine LPEN (file) signals the PDP-8 that a light-pen selection is to be expected and, after this selection is made, gets the selected file name from the data buffer. This routine is called by subroutine LPSELX.

4.8 The slave programme on the PDP-8.

This programme, written in PAL-D assembler language, handles all transfers of data between the PDP-8 and the disc interface. Its normal use is as a slave programme monitored by the PDP-15's master programme. However, mere transfers (read from or write onto disc) may be performed without the PDP-15.

The size of this programme is approximately 400 words (octal). It is saved on DEC-tape as a "user" programme named DISC. The operation of the programme is shown in the flow chart of Figure 4.8. As soon as it is started, it enters a waiting loop, attempting to read the contents of the data buffer which links the PDP-15 to the PDP-8. When a significant control code is read, the corresponding routine is called and the programme re-enters the waiting loop as soon as the task is completed.

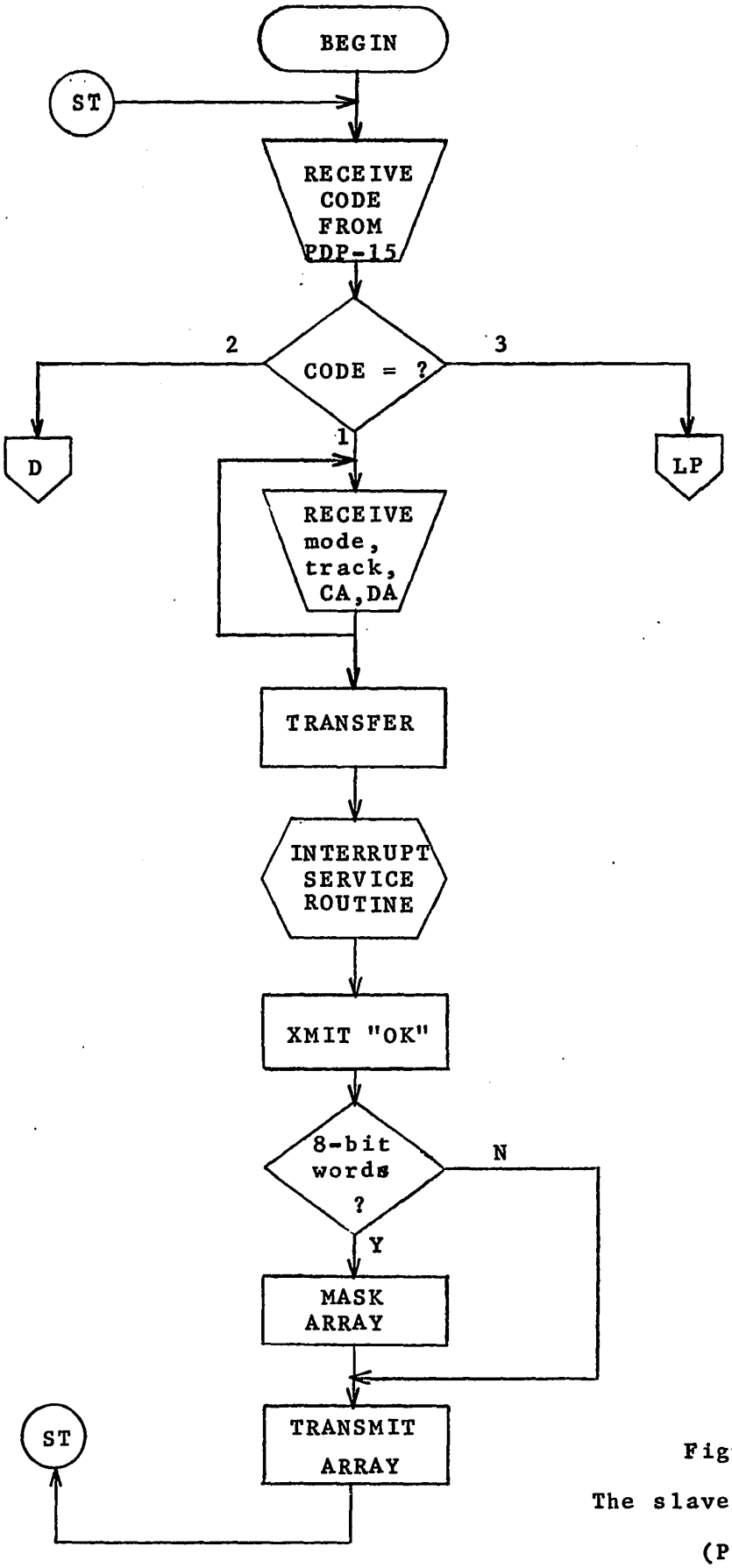


Figure 4.8

The slave programme DIS

(Part 1)

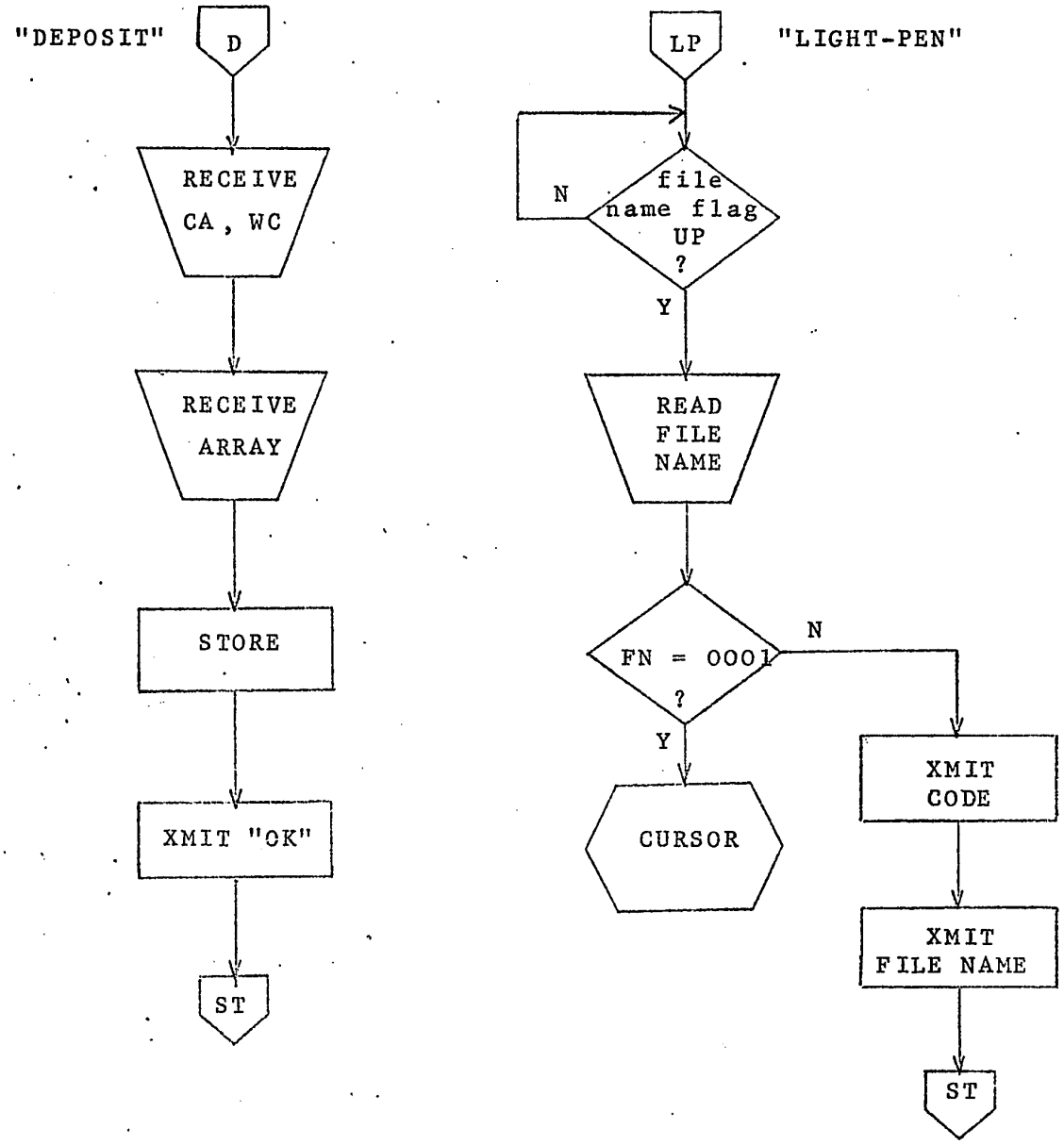


Figure 4.8 The slave programme DISC. (Part 2)

4.8.1 Code 1 - Transfer.

When a 1 is received, transmitted from the PDP-15 by subroutine WS8, the programme performs the following:

- read the initialization data (3.2.2.2) from the data buffer and store them;

- format those data as to make them acceptable to the disc interface;

- initialize the transfer;

- signal the PDP-15 when the transfer is completed.

To permit debugging, the array which has just been transferred is then transmitted back to the PDP-15; if the mode of transfer is a 8-bit mode (which is the case for graphics instructions) the four higher order bits of each word are masked before transmission.

4.8.2 Code 2 - Deposit.

This routine performs the transfer of an array from the PDP-15 to the PDP-8. It is called by subroutine SDFILE. The location data i.e.

- the first address of the array in the PDP-8 core,
- the number of words to be deposited,

are first obtained through the data buffer. Then, a looping

routine reads each word from the data buffer and stores it. An OK code is sent to the PDP-15 after normal completion.

4.8.3 Code 3 - Light-pen handling.

When the code 3, transmitted from the PDP-15 by subroutine LPEN, is received, the programme enters a waiting loop in which it checks the light-pen flag. As soon as this flag is raised (by a light-pen intercept on the screen) the contents of the File Name Register of the interface are loaded into the PDP-8 accumulator. If the file name is recognized as 0001 the programme executes the CURSOR routine (used for pen-tracking). Otherwise it transmits a signal to the PDP-15, followed by the file name itself.

4.8.4 Manual operation.

The initialization data may be toggled in on the PDP-8 console and the programme may be started manually. This allows the transfer of data to and from the disc without needing the PDP-15 and it has been extremely convenient during the testing phase of this project.

CHAPTER V

EVALUATION OF THE SYSTEM

In this chapter, we shall attempt to evaluate the system discussed in the previous chapters. This evaluation will be done both from the designer's and from the first user's points of view. Unfortunately, those two roles have been assumed by the same person and, to date, no comments are available from a second user unrelated to the development stage. In fact, the duality designer-user has been present at all stages of this project and most of the software has been tailored to the needs of the application programme.

When the project was initiated, the Display Processing Unit had been completed and its logic had been tested in simulation but the disc interface had still to be tuned up. As soon as the basic graphics instructions (4.7) had been written, they were used to debug the hardware, mainly the disc interface, and while the Display Processor was being tuned up, the writing of higher level routines was undertaken. This simultaneity of hardware and software debugging, with all the ambiguity and doubts which it entails, has been one of the major problems arising in this project. Long hours have been spent in arguments between the software man and the hardware people in an attempt to define responsibilities and it has been a rewarding experience to have all problems eventually solved to the satisfaction of all.

Some hardware bugs have been patched by software and conversely a few hardware parts have been redesigned to ease the programmer's task.

The present size of the PDP-15 core, 24K, certainly is appreciable. When this project was started, only 8K were available; the additional core purchased since then has been a considerable relief to the programmer, as well as the disc memory. The latter permits loading and unloading of the CPU core during execution, without noticeable delay for the user. The disc memory is used in this programme for the storage of data. The data matrix is kept on disc and each column (representing one variable) is loaded into core only when needed by the programme. The data used during the development of the system filled a real matrix of 22 variables by 99 cases, i.e. more than for 4K (decimal). However, most regression analyses would make use of many more data points and it might be necessary to split one column into several loads. Data storage for statistical programmes is such a problem that Muller (57) proposes to pass programmes over data instead of passing data over programmes, as is usually done. This means that each data record, once identified, would call all the programmes in which it is needed and all analyses would be updated for this record before the next record be loaded into core. This drastic step in programme organisation has not been taken here because of the limited size of the data matrices to be analysed, but it might be considered in further work.

Also, it has not been found necessary yet to chain the programmes. But, should the application programme grow in size, this chaining could be efficiently performed, thanks to the disc, with virtually no difference in the system's response time. Because they are largely interrelated, most graphics subroutines could not overlay each other, and in any case the possible saving in space would be of little value, due to their relatively small size. Furthermore, since the basic routines are called very often in a programme the total loading time could end up affecting the response time of the application programme. On the other hand, the regression package subroutines are large enough to allow a substantial saving to be gained from chaining. This has been considered in the organisation of the package and only a few changes would have to be made in data handling procedures.

The core size on the PDP-8 is even less problematic since the disc handler (4.8) takes only 400 locations (octal), out of the total 10K. Although some 1,000 locations are needed for the system's resident monitor, the remaining space is sufficient to house a number of functions such as for instance a pen-tracking programme and a joy-stick handler.

The disc buffer has proven to be a very convenient refreshing device, in spite of a few hardware malfunctions which are still being experienced; in particular, some cross-talk has been observed at times between certain tracks. The disc's response time is very satisfactory. With its revolution speed

of 30 r.p.s. and its writing function being enabled every two revolutions, the picture changing rate is 15 times per second. This is fast enough for all user requirements, even animation, and slow enough to ensure that in most applications, the computer time necessary to compute a new picture will be able to match this maximum rate. When a light-button is put out, items are switched on one at a time at a rate which turns out to be slightly faster than the user's reading rate. This can be clearly tested with the display shown in Figure 3.4 with its 21 choices.

Disc space brings no limitation either. Each track has a capacity of about 12,000 graphics instructions. This is more than sufficient, not only to store the most complicated display, but also to permit the selective use of prestored displays. This technique is used in this project for all the light-buttons which are stored on track 16. The totality of the information necessary for the displays shown in Figures 3.2 through 3.7 represents 2451 words, i.e. one fifth of the track's capacity. A display such as the plot of Figure 3.10 takes 725 words, while the histogram of Figure 3.8 takes 653 words.

It is clear that much space is saved thanks to the stroke vector generator. A two-word vector instruction draws in scale 2 a 1/4 in. long vector which would otherwise necessitate a minimum of 10 points. The most complicated alphanumeric character (8) necessitates 26 words.

Partly because disc space is no problem, the absence of a hardware character generator does not really affect the system's efficiency. The software character generator even has the advantage of a greater flexibility. The ASCII codes of characters which are not in common use may be reserved for conveniently calling special graphical entities.

Maybe one of the most interesting peculiarities of this system is that all its tracks are available and accessible to the user. Most multi-track systems are used in a time-shared environment where each user has access to only one track. Here, either by software control or by manually switching the track selector, one may use or display any of the 16 tracks. This is very useful in a number of applications (such as modeling) and considerably reduces the need for hard copy. This property has been widely used in this project. All light-buttons are saved on one dedicated track (no.16), inaccessible to the user, thus eliminating the need for rewriting them every time they are called for. During the preliminary examination of the data, it is convenient to store on contiguous tracks the plots corresponding to the projections of the data points on various planes (e.g. X_1 vs X_2, X_2 vs X_3, X_1 vs X_3). This is also very useful for residual plots and for the comparison of the analyses of variance for several models.

The operation of the light-pen seems quite satisfactory in view of the use which has been made of it in this project.

The small beam of light which is projected on the screen by the pen is a convenient help for locating the exact point which one wishes to intercept. Although no specific test has been conducted to check the accuracy of the detection, only a minor problem has been encountered. It seems that the response is not quite fast enough to guarantee the correct identification of a display file when the pen intercepts its very last strokes. This fault has been noticed while attempting to choose item "1" in the light-button of Figure 3,4. Whenever the vertical stroke of the digit was hit, item"2"was actually picked up. This is likely to happen in the very rare cases where item files are particularly short. One specific case might be in a pattern such as the one shown in Figure 1.1 and used for pen-tracking. In any case, a software patch has been implemented. The HEADER subroutine inserts a couple of dummy instructions before the actual header instruction, so that an appropriate delay is secured between two files.

The Regression Analysis Package itself has been debugged and evaluated with a set of industrial data supplied by Canada Cement Lafarge. The goal of the study was to derive a linear model for the compression strength of the cement after 3, 7 and 28 days. The independent variables are the physical characteristics and chemical composition of the samples and a number of process parameters (e.g. the kiln output rate). As already mentioned in Chapter II the study had been started using a BMD programme before the development of this system was undertaken. This previous experience has been decisive

for evaluating the usefulness of interaction in this regression analysis package.

It had been found during the previous stepwise regression studies and it has been confirmed on this system that, most of the time, bivariate models were representative enough of the cement process, with the data available. This is the reason why, in order to conserve core space, the implementation of the least squares technique has been initially limited to the case of one or two independent variables. It is felt, however, that this would not be sufficient for a general purpose regression package, and a higher order least squares algorithm or a stepwise regression algorithm will be considered when programme chaining is implemented in the system.

The modular organization of the package is very convenient, as it permits a great variety of paths in the analysis and thus results in a saving of time for the user.

Sitting at the display and running this fully operational programme was the last stage of a project which had started, some time ago, by entering bit by bit into the PDP-8 a few graphics instructions, with the hope that a square pattern of 4 points would appear on the screen. It cannot be remembered now whether it had actually appeared that day but what can be said in conclusion is that this work has constituted a challenging but extremely rewarding experience.

BIBLIOGRAPHY

1. Davis R. M. A history of automated displays. Datamation, 11, 24. Jan. 1965.

2. Loewe et al. Computer generated displays, Proc IEEE, 49,185. Jan. 1961.

3. Sutherland I.E. SKETCHPAD: A man-machine graphical communication system. Proc. AFIPS 1963 FJCC, 23, 329.

4. Watkins G.S. A real time surface algorithm. University of Utah UTEC CS c-70-101.

5. Univ. of Illinois Conference on Computer Graphics."Emerging concepts in Computer Graphics", Proc., 1968.

6. Jordan J.A. Kinoforms and holograms as I/O devices. Computer designer's conference and exhibition (abstracts). Anaheim, 19-21 Jan. 1971.

7. Merritt M.J. and Sinclair R. INSIGHT - An interactive graphic instructional aid for systems analysis. Proc. AFIPS 1971 FJCC, 39,351.

8. Schwartz J.L. and Taylor E.F. Computer displays in the teaching of Physics. AFIPS 1968 FJCC, 33, 1285.

9. Weiner D. D. Computer animation. An exciting new tool for Education. IEEE Trans. on Ed. vol E-14, 4, 202. Nov. 1971.

10. Uber G.T. et al. The organisation and formating of hierarchical displays for the on-line input of data. Proc. AFIPS 1968 FJCC, 33,219.

11. Williams R. A Survey of Data Structures for Computer Graphics Systems. Computing Surveys.Vol.3, 1, 1. 1971.

12. Bracchi G. et al. An interactive software system for computer-aided design. An application to circuit-design project. Comm. ACM, 13, 9,537. 1970.

13. Saillard J.C. Utilisation d'un terminal graphique dans la conception assistee par ordinateur. L'Onde Electrique, 52,2,75. 1972.

14. Jarvis J.F. The design of interactive graphical aids to mask lay-out. Proc. IEEE,60,1,35. 1972.

15. Howe R.M. et al. Time-shared hybrid computers: a new concept in computer-aided design. Proc. IEEE, 60, 1, 71. Jan. 1972.

16 Showalter A.K. Display Systems. Datamation,22,28, 1971.

17. Boyle A.R. Computer-aided map compilations. Man-Computer Communications Seminar. Ottawa, May 31-June 1, 1971.

18. Anderson R.H. A selective bibliography of computer graphics. RAND Corporation. 1971.

19. Covvey H.D.J. et al. The television/computer system. The acquisition and processing of cardiac catheterisation data using a small computer. Proc. AFIPS 1971 FJCC,39,455.

20. Coulam C.M. et al. Three-dimensional computerized display of physiological models and data. Computers and Bio-Medical Research,5, 1972.

21. Smith L.B. A survey of interactive graphical systems for mathematics. Computing Surveys,2,4,261. 1970.

22. Lewin M.H. An introduction to computer graphics terminals. Proc. IEEE,55,1544. 1967.

23. Webber D. Some applications of computers in traffic engineering. Traffic Engineering and Control. Feb. 1972.

24. Matsushita Y. Hidden lines for a rotating object. Comm. ACM,15,4,245. 1972.

25. Birnbaum J. et al. An Interactive Graphics System for Nuclear Data Acquisition. IBM J. of R. & D. Jan. 1969.

26. Christensen C. and Pinson E.N. Multi-function graphics for a large computer system. Proc. AFIPS 1967 FJCC,31,697.

27. Newman W.M. Display Prodedures. Comm. ACM,14. 1971.

28. de Hollan A.N. et al. A network model of the U.S. Air Transportation System. 1971 Winter Simulation Conference.

29. Gurley B.M. and Woodward C.E. Light-pen links computer to operator. Electronics,32,47,85. 1959.

30. Davis M.R. and Ellis T.D. The Rand tablet: a man-machine graphical communication device. Proc. AFIPS 1965 FJCC,26,325.

31. Groner et al. BIOMOD - An interactive computer graphics system for modeling. Proc. AFIPS 1967 FJCC,39,369.

32. Joyce J.D. and Cianciolo M.J. Reactive displays: improving man-machine graphical communication. Proc. AFIPS 1967 FJCC,31,713.

33. Young D.W. Organisation of Information into Displays. Computers and Bio-medical research,5. 1972.

34. Ophir D. et al. BRAD: The Brookhaven Raster Display. Comm. ACM,11,6,415. 1968.

35. Terlet R.H. The CRT display sub-system of the IBM 1500 instructional system. Proc. AFIPS 1967 FJCC,31,169.

36. Lovercheck L.R. Raster scan technique provides multicolor graphics displays. Electronics, June 5, 1972.

37. Shannon and Beckner. Program controlled colour computer graphics. Proc. 9th UAIDE Ann. Meeting, 1970.

38. Johnson R. H. Programming for computer output microfilm. Computer designer's conference and exhibition (abstracts) Anaheim 19-21 Jan, 1971.

39. Pitteway M.L.V. A simple data compression technique for graphic displays or incremental plotters. Symp. on Comp. Proc. in Comm., Brooklyn, Apr. 8-10, 1969.

40. Savas E.S. Computer control of industrial processes. McGraw-Hill, 1965.

41. Hamaker H.C. On multiple regression analysis. Neerlandica, 16 (1962).

42. Efroymson M.A. Multiple regression analysis. Ch.17 in Ralston A. and Wilf H. (eds.), "Mathematical methods for digital computers". Wiley, 1960.

43. Draper and Smith Applied Regression Analysis. Wiley 1966.

44. Brownlee K. Statistical Theory and Methodology in Science and Engineering, Wiley, 1965.

45. Daniel C. and Wood F.S. Fitting equations to data. Wiley 1971.

46. Anscombe F.J. and Tukey T.W. The examination and analysis of residuals. Technometrics, 5, 1963.

47. Cochran and Cox Experimental Designs. Wiley, 1964.

48. Anscombe F.J. Rejection of outliers. Technometrics 2,1960.

49. Johnson and Leone Statistics and experimental design in engineering and the physical sciences. Wiley.

50. Schatzoff M. et al. Efficient calculation of all possible regressions. Technometrics 10. no. 4, Nov. 1968.

51. Dixon W.J. (ed.) BMD, Bio-medical computer programs. University of California in L.A. 1964.

52. Dixon W.J. Use of displays with packaged statistical programs. Proc. AFIPS 1967 FJCC,31,481.

53. Fabi R.J. The design and construction of a disc-oriented graphics system. Thesis, Dept. Elec. Eng., McGill University. 1971.

54. Malowany A.S. et al. A disc-oriented graphics display system. Man-computer Communications Seminar. Ottawa, May 31-June 1, 1971.

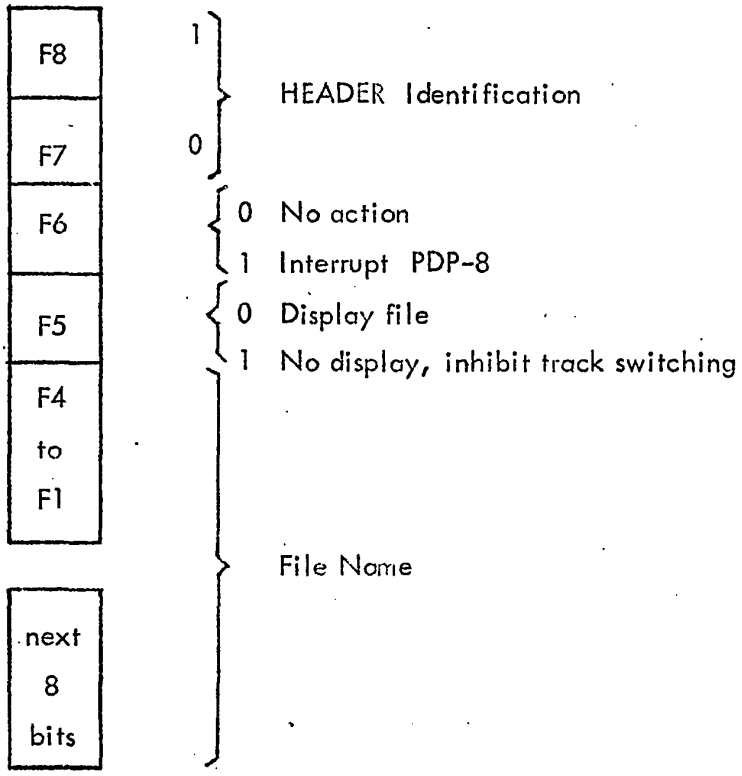
55. Knox J. and Ichiyen N. Assembling and testing of an interface to a PDP-8 computer. Report, course 499T, Dept. Elec. Eng., McGill U., April 1970.

56. Bell T.E. Computer graphics for simulation problem-solving. RM 6112-PR. RAND Corporation, 1969.

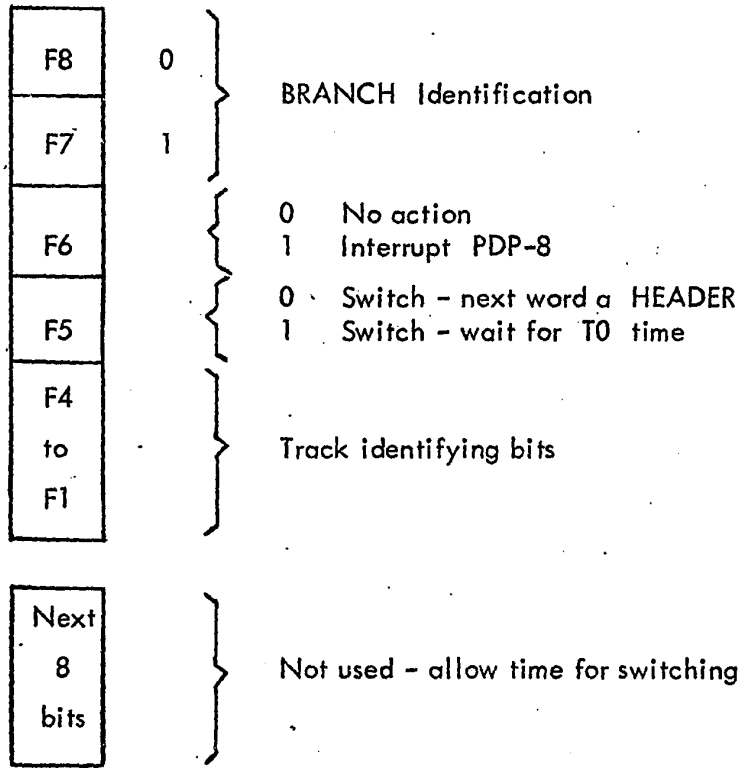
57. Muller M.E. Computers as an instrument for data analysis. Technometrics,4,12. 1970.

APPENDIX I

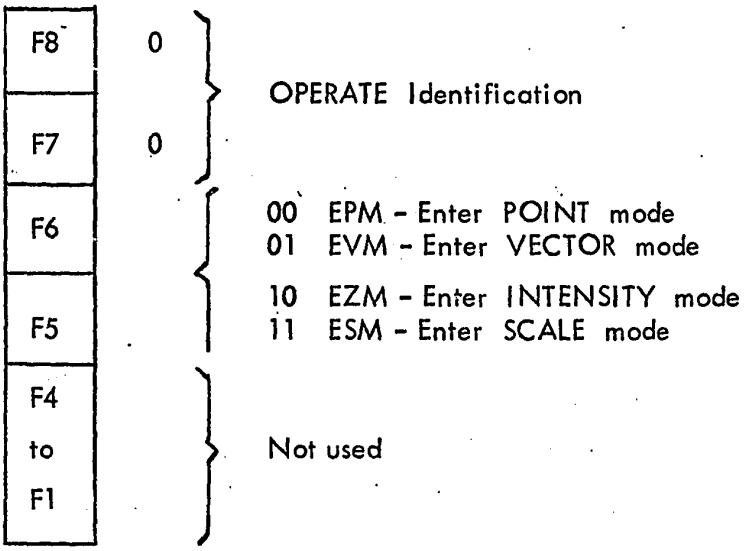
THE INSTRUCTION FORMATS



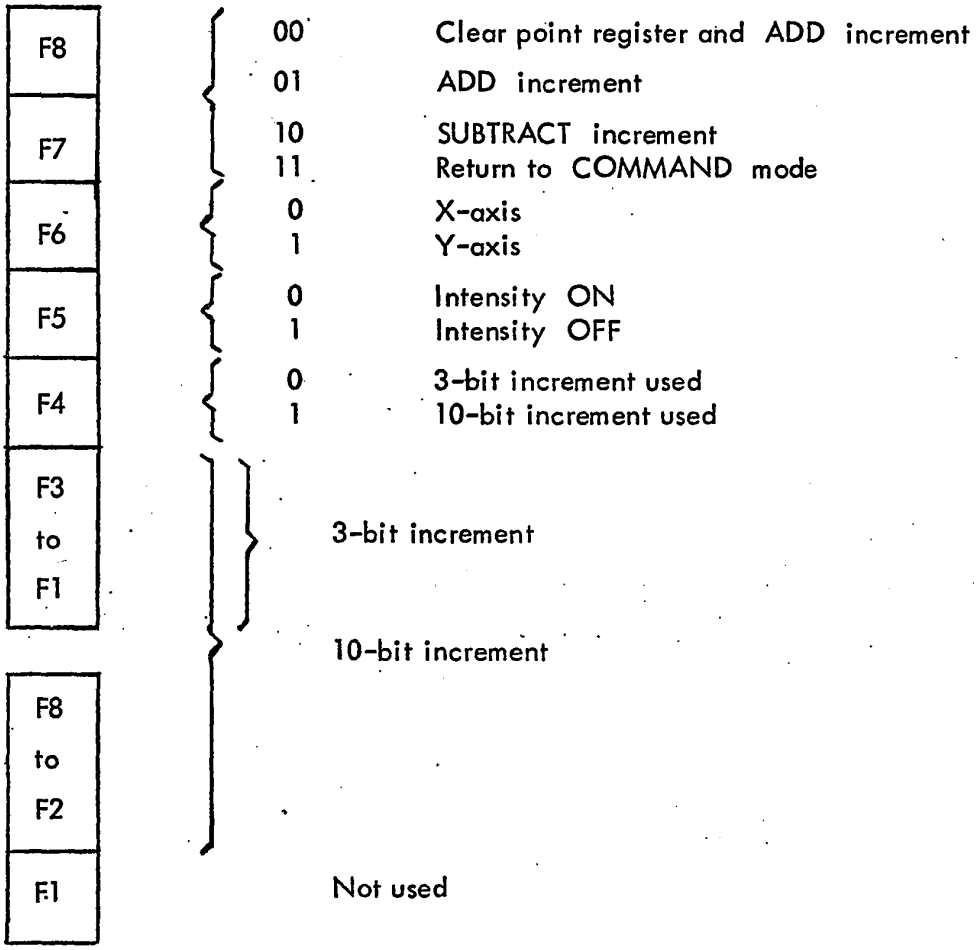
HEADER INSTRUCTION



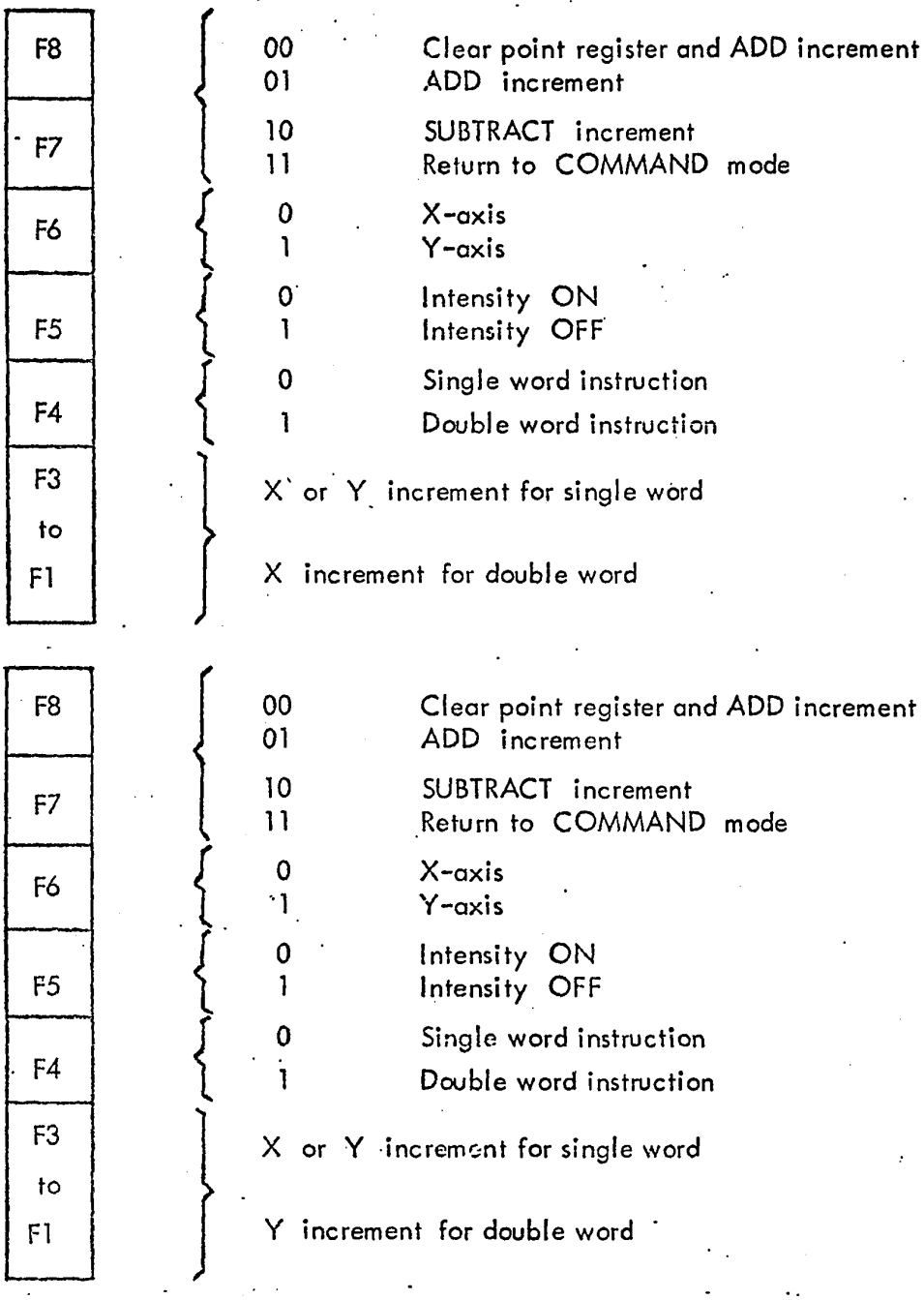
BRANCH INSTRUCTION



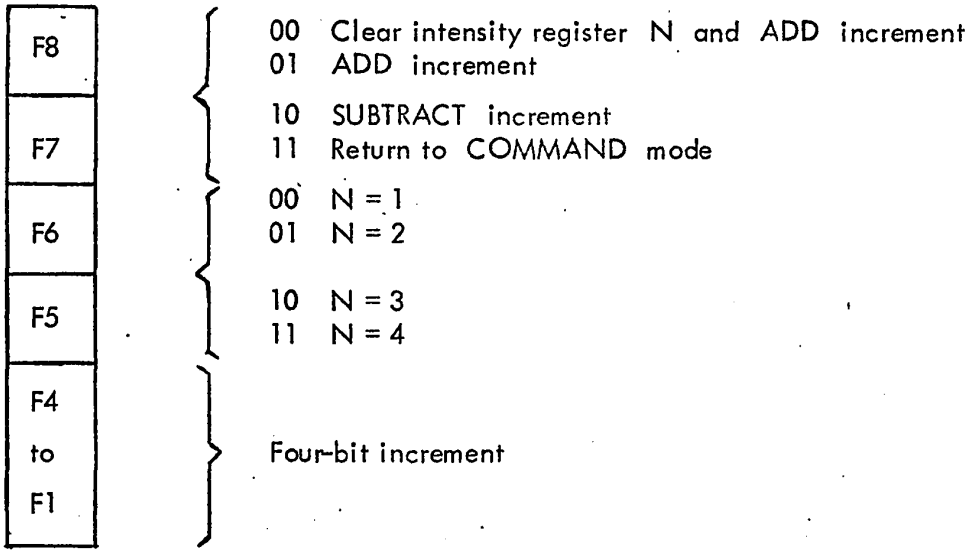
OPERATE INSTRUCTION



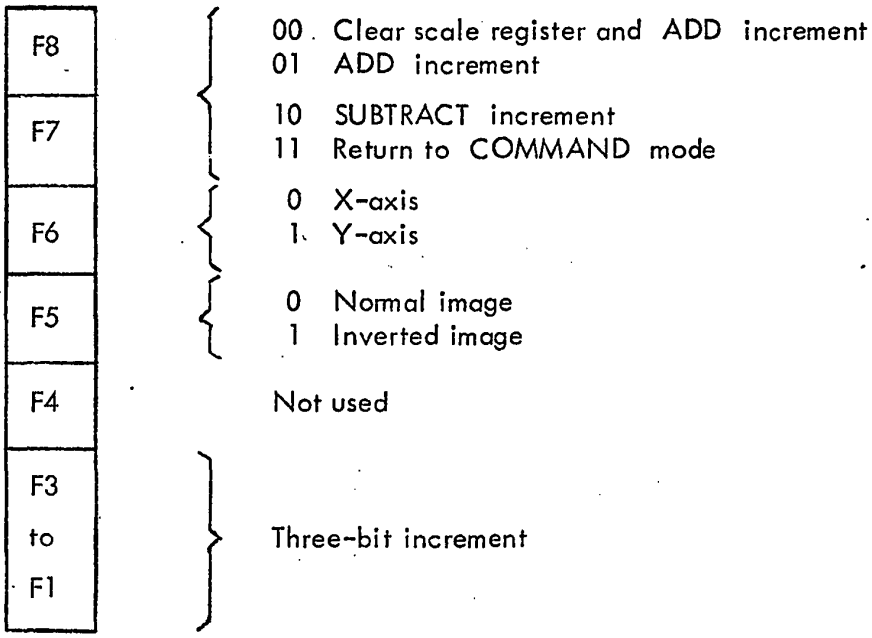
POINT INSTRUCTION



VECTOR INSTRUCTION



INTENSITY INSTRUCTION



SCALE INSTRUCTION