INTERACTIVE COMPUTER GRAPHICS WITH SPECIAL REFERENCE TO ELECTRICAL CIRCUIT DIAGRAMS

T. J. Lawrence, M.Sc.(Hons.)

Submitted in partial fulfilment of the requirements
for the Degree of Doctor of Philosophy in The
Australian National University

JULY, 1973

Department of Engineering Physics,
THE AUSTRALIAN NATIONAL UNIVERSITY,
CANBERRA, A.C.T.

DECLARATION

I declare that this Thesis reports my own original work, that no part of it has previously been accepted or presented for the award of any degree or diploma by any University, and that to the best of my knowledge no material previously published or written by another person is included, except where due acknowledgement is given.

[illegible]

T. J. Lawrence
Canberra
July, 1973

PREFACE

This Thesis is concerned primarily with problems in Interactive Computer Graphics, with special emphasis on a system which communicates in the Electrical Circuit Domain. The work is mainly within the field of Computer Science, but also draws briefly on other disciplines such as Electrical Engineering and Perceptual Psychology.

The research presented was carried out in the Department of Engineering Physics, Research School of Physical Sciences of The Australian National University (ANU), under the guidance of Professor S. Kaneff and was commenced in 1968.

Originality is suggested to stem from an attempt to rectify a shortcoming of currently available graphic systems which (as far as is known to the writer) are unable to perform "knowledgeably" in their problem domains. The novel features of the developed system arising from this objective are claimed to be:

(1) The separation of problem domain descriptions from purely pictorial descriptions with the independent generation of the two.

(2) The incorporation of "knowledge" of the problem domain into the system. This knowledge concerns the construction of circuit diagrams, and can be divided into 3 parts:

- (a) knowledge of "aesthetic" requirements for a pleasing diagram,
- (b) specific knowledge on how to draw particular types of circuits, and
- (c) specific knowledge of the connectivity of particular types of circuits.

(3) The provision of a facility to use the "knowledge" mentioned in (2) to produce a circuit diagram from purely electrical information.

(4) (Because of (3)), the ability of the user to converse in purely electrical terms.

(5) The fact that, unless he so desires, the user does not need to position components in a diagram, this being done by the nachine.

(6) The machine's "knowledge" of the connectivity of particular circuit types so that the user is freed from having to specify this information.

(7) The use of knowledge of particular circuit types, and knowledge on general layout of parts, which enables the machine to draw circuits which are modifications or combinations of known types. This helps reduce the number of specific types which nust be stored in the machine.

(8) The basic principles behind points (1) to (7) are not domain specific and are applicable to a wide variety of areas.

Two papers (related to the ideas of Chapters 5-8) resulting from this work have been accepted for publication in 1972. They are:

Lawrence, T.J., "Pictorial Organization of Electrical Circuit Diagrams"; in Pictorial Organization and Shape, Div. Computing Research, CSIRO* Canberra, P. 125

Lawrence, T.J., "Generation of Circuit Diagrams from Electrical Descriptions"; In Press in Computer Graphics and Image Processing. Academic Press, New York.

* Commonwealth Scientific and Industrial Research Organization

ACKNOWLEDGEMENTS

The author wishes to acknowledge and express thanks to his supervisor, Professor S. Kaneff, of the Department of Engineering Physics, The Australian National University, for helpful discussions and guidance throughout this work.

Thanks are also due to members of the Artificial Intelligence and Picture Processing group at ANU and CSIRO, for valuable comments from time to time.

Financial support provided by an Australian National University Scholarship is gratefully acknowledged.

CONTENTS

SUMMARY

A study in Interactive Computer Graphics is presented. A broad survey of this and related fields points out certain deficiencies in current graphics systems; these deficiencies are suggested to be due to a lack of "knowledge" within the machine of the subject about which communication is to occur. The present work attempts specifically to overcome the difficulties in the domain of electrical circuits and their corresponding diagrams. This has been done by the inclusion of models which contain information concerning connectivity and parts of various circuit types, and data on how the diagrams for these circuits may be drawn. The use of these models frees the user from the necessity of specifying detailed connectivity for circuits, and also places the onus on the machine for the actual production of diagrams.

Before developing these models, however, necessary examination of the pictorial associations found relevant by human beings in drawing and interpreting circuit diagrams is carried out. It is found that many indirect pictorial clues are used in determining function of a circuit, and these must be taken into account by the machine as it produces diagrams.

The pictorial associations thus discovered are also found useful in the production of diagrams for circuits which do not exactly correspond to known models, and where certain new parts must be suitably incorporated into a diagram.

The principles involved in the present system are believed to be general, and a discussion of extensions to the system, with possible use in various domains and environments, is given.

CHAPTER 1
INTRODUCTION

The research reported here is concerned with an apsect of Interactive Computer Graphics forming part of the field of Man-machine Communication, which in turn may be considered as a subset of the broader field of Artificial Intelligence. The approach is taken that the communication in graphic systems should be as natural as possible from the human point of view.

A shortcoming of currently available graphic systems lies in the lack of knowledge, within the machine, of the environment of operation, leading to an inability to communicate freely. This shortcoming might be alleviated by making use of developments which have been made in various aspects of Artificial Intelligence research. Natural language communication systems, for example, have already made use of this "environmental knowledge" to improve communication performance.

The present project involves an investigation of the domain of electrical circuit diagram communication; a novel incorporation of knowledge into a machine allows greater freedom of communication for a user in this domain. Two of the main features of this communication system are:-

(1) the inclusion of knowledge of the connectivity of useful circuit types, thereby freeing the user from having to specify the exact arrangement of each part of his circuit,

(2) the inclusion of knowledge regarding the construction of circuit diagrams, enabling the system to draw diagrams from circuit information specified as in (1) above. This knowledge includes not only the arrangement of particular circuit types, but also the aesthetic requirements for an acceptable diagram. The latter aids the system in drawing diagrams for circuits not conforming exactly to known types.

1.1 Relevant Fields

It has already been indicated that useful concepts for the present work have been obtained from other related fields, particularly those introduced below and surveyed in more detail in Chapters 2 and 3. This survey provides a context for discussing the present work, and particularly outlines the state of development of related systems, pointing to the need for enhancement of functional capabilities. These fields, although tightly interrelated, can with advantage be isolated for purposes of discussion.

1.1.1 Artificial Intelligence

This involves two distinct questions.

(1) What is the nature of intelligence? This is largely in the realm of the psychologist, but some achievements in simulation lend insight.

(2) How can an intelligent machine be built? Two general approaches may be identified: the first attempting simulation of the detailed organization of the human brain; and the second simulating the performance rather than the mechanism. The former method appears limited by scale and complexity. The latter, however, has been successful, for example, in the production of various game playing and problem solving systems (Newell et al, 1958; Samuel, 1967; Greenblatt et al, 1967; Elcock and Murray, 1968).

An important aspect of intelligent behaviour is communication, which is considered in succeeding sections.

1.1.2 Picture Processing

As normally interpreted, Picture Processing concerns attempts to perform, by machine, certain tasks which a human being performs with pictures, and may be viewed as part of Artificial Intelligence research (though not all workers have considered it thus).

Many approaches have emerged; early picture processing systems were concerned with classifying fairly regular images into one of a number of categories (Bledsoe and Browning, 1959; Uhr, 1963; Van der Lugt, 1964; Nagy, 1968; Dye, 1969; Andrew, 1969; Levine, 1969). More recently attention has been focussed on the description of images, and this approach has led to more flexibility being allowed in input images, as well as better methods for segmenting pictures into relevant parts (Grimsdale et al, 1959; Evans, 1967, 1968; Guzman, 1968; Narasimhan, 1966; Shaw, 1968; Stanton, 1970; Kaneff, 1970). In particular, hierarchical descriptions of pictures have followed from this approach.

1.1.3 Man-Machine Communication

Communication in natural languages (e.g. English) has been the subject of considerable interest (Weizenbaum, 1966; Bobrow, 1964; Minsky, 1969; Woods, 1967). Two important points have emerged from this research:
(i) it has so far proved impossible to develop a complete grammar representing natural language: however, extensive subsets have been formalized and used in particular cases (Woods, 1967);
(ii) incorporation of "knowledge of the environment" into a machine, greatly increases its capacity for apparently natural conversation. Systems with such knowledge are able to hold quite "natural" conversations despite their use of very fixed format language techniques.

1.1.4 Interactive Computer Graphics

Graphical man-machine communication, via Cathode Ray Tube (CRT) displays and light pens, has developed along different lines from that of natural language communication. Attempts to simulate true graphical communication, as used between human beings have proven difficult* and most systems require the selection of an object, followed by placement of that object (pick-and-place). Some systems (Sutherland, 1963) also allow the specification of constraints relating the parts of an image. A feature common to graphical systems is that they have not attempted to incorporate "knowledge of the environment" to a significant extent. Such incorporation has been found very useful in natural language communication, and there seems no reason why this should not be so also in graphical systems. Complex graphical data structures which have been developed (Williams, 1971) resemble the hierarchical structures found in the parsing approach to picture processing: this is not surprising, as the

*No known practical systems attempt the problem.

constraints and relations in these data structures are the same relations which are recovered in the parsing of pictures.

1.2 Perspectives of the Chosen Problem

Evans and van Dam (1968) have argued that for efficiency in graphical systems, data structures should be specifically designed for each application. Furthermore, in any system attempting to incorporate useful knowledge from the problem domain, a general system would appear almost impossible due to the amount and diversity of knowledge required. This argument is difficult to counter, and the generality claimed for the communications system developed herein, is not on this level. Rather, the design principles are applicable in a wide variety of domains, so that a similar program could easily be developed to operate in another field. This helps surmount the knowledge problem, as only information on one field must be incorporated for any given system.

The application chosen for the development of the present system is the electrical circuit domain: the reasons for this choice are discussed in more detail in chapter 4, but briefly, include: usefulness in terms of there being a current need for systems operating in this domain; the existence of other systems against which comparisons may be made; and the well-defined nature of electrical circuits.

An important feature addressed in this system is the "relation of representation" which considers the problem domain and pictorial domain as separate entities but adequately related. This leads to the incorporation of a capability for drawing circuit diagrams. The user is given more freedom, then, in the manner in which he may specify his input. Since the primary interest here is to investigate graphical communication, no attempt has been made to include the system into a complete environment (such as that of Computer Aided Design); this leads to some generality of environments to which this system may be applied.

To put the present work in the context of other fields: it is clearly related to computer graphics. Some of the important ideas concerning the incorporation of knowledge of the environment are related to the field of natural language communication, and form a link with this field. Connection with picture processing exists because of the necessity to investigate electrical circuit diagrams carefully to determine the various pictorial relationships of importance in producing "well-formed" diagrams. These relationships are very closely related to those considered in the analysis of images. There is an obvious connection with the general field of Artificial Intelligence.

1.3 Outline of this Report

This chapter has introduced the various relevant fields, and the present work in general terms. Chapter 2 surveys the fields of interest in some detail and provides background to the developed ideas, while Chapter 3 discusses a selection of existing graphics systems of direct relevance to this project, with the object not only of detailing these systems, but also of pointing out weaknesses and possible improvements.

Chapter 4 discusses the present problem, and the next chapter outlines the general principles of the solution used. The three following chapters describe particular aspects of the system, followed by a chapter presenting the programs which have been developed.

Chapters 10 and 11 provide a discussion of the achievements and shortcomings of the developed system and gives a number of recommendations for future development: this is followed by a concluding chapter. Two appendices, one listing programs, the other giving an example of the data structures employed are included.

Throughout the report a number of methods of solution which were not eventually incorporated in the system are discussed in detail, as they provide additional insight into the problems involved, and into the advantages and disadvantages of the various possible solutions.

CHAPTER 2
SURVEY OF RELATED WORK

2.1 Introduction

This chapter reviews the relevant fields already introduced, outlining research in each field, and indicating achievements and shortcomings.

There are several reasons for including this survey: the various fields mentioned have tended to develop independently, and this review serves to indicate relationships which exist between them. In doing so, a general context for the present work is also established. In addition, several ideas which have been developed in various independent studies and have not been used in interactive graphics (but which clearly have relevance therein), have been identified. Some of these concepts have been taken and developed in this project; the review serves to indicate their origins.

2.2 Artificial Intelligence

Attempts to provide insight into the nature of Intelligence have so far met with comparatively little success. I.Q. tests, which purport to put a measure on intelligence are generally recognized as being of questionable validity: furthermore, they do not pretend to offer any insight into the fundamental nature of intelligence. Attempts have been made also to provide an understanding of intellect from biophysical studies, which have, however, provided little information about brain mechanism (Harlow and Woolsey, 1958; Wooldridge,1963). At a low level, much is now known about the behaviour of individual neurons and this has led to a number of methods for their simulation. This work, however, does not appear to lead directly to an understanding of the nature of intelligence.

A more practical approach seems to be to take a behavioural view by defining intelligence in terms of performance, and measuring degree of intelligence by the adequacy of decision-making processes present. Stated more carefully (Fogel et al 1966):-

> Intelligence can be viewed as the ability of any decision-making entity to achieve a degree of success in seeking a wide variety of goals under a wide range of environments.

This approach is closely related to Turing's Test (Turing, 1950) to determine machine intelligence; this involves an interrogator, a human being, and the machine. The interrogator is required, by asking questions, to determine which is the man and which is the machine. If, at the end of a suitable period, he is not able to distinguish between the two, then the machine is deemed to be intelligent*.

These tests and definitions still do not indicate what intelligence is, nor do they give a quantitative measure. Nevertheless, workers in the field have been exploring many avenues of investigation, among them being:

(1) Simulation of neural networks to exhibit similar properties to the nervous system. This work is based on the contribution of McCullough and Pitts (1943), discussing the logical processing that can be achieved with neural cells. A well-known development of this type is the perceptron first described by Rosenblatt (1958, 1962) and expanded by a number of other workers (e.g. Minsky and Papert 1969). A perceptron consists of a set of sensing elements connected to a set of association elements, which are randomly connected to a set of response elements, whose task is to read out the result of a pattern recognizing operation. The response elements have inhibitory interconnections amongst themselves and feedback connections to the association units. There are two phases of operation: during the first (learning) phase the perceptron is offered a number of stimuli and a response is forced artificially, causing modification of the interconnection properties of the system. After the learning phase, the perceptron is offered further stimuli and, if the modifications have been sufficient, the resultant response will have a high probability of being correct.

Learning is represented in the Perceptron model by stating that the distribution of connectivities within the machine represents the experience of the machine up to the time in

*This test, however, emphasizes intelligence similar to human intelligence.

question. Perceptrons have been made to perform simple tasks, but could not be said to exhibit significant intelligence. Such methods of direct modelling face tremendous difficulties due to the complexity of the brain (which comprises around 10^{10} cells) and because of the simplicity of the functions used in Perceptrons.

(2) Viewing the human being as a psychological entity and attempting to simulate his performance. In this activity, simulating functions rather than the fine details of the mechanism promises to be a more successful approach. For example (Fogel et al 1966 p7), modern aircraft obey the same aerodynamic laws as birds, but they do not have oscillating wings and feathers.

This approach has led to "heuristic programming", some of the chief applications of which have been in game playing programs (Newell et al 1958, Bernstein et al 1958, Bernstein and Roberts 1958, Kister et al 1957, Samuel 1954, 1967; Greenblatt et al, 1967; Elcock and Murray, 1968) and in problem solving programs (Newell et al 1957, Gelernter 1959, Gerlernter et al 1960, Newell and Simon 1961, Slagle 1963). In essence these programs "model" intelligence in the following manner:-
A situation requiring a decision is presented to a subject who is required to make the decision; his decision is then analysed in order to reveal a consistent set of subquestions which facilitate the decision making. The rationale in the decision making process is hopefully fleeced out and incorporated into a computer program to perform the same task.

Analysis of a number of programs might indicate common features which could be included in a program to operate effectively or "intelligently" in the face of new or unexpected situations.

Programs of this type are limited to the ingenuity of the original programmer. While they can improve their performance by continued practice (in the game playing programs by playing either human being or other machine programs), all they do is increase their experience without, however, gaining new insight into the situation. This insight seems an essential part of intelligence and is yet to be achieved by machine.

(3) The evolutionary approach - taking the view that man is one product of evolution, rather than viewing man as the ultimate intelligence. The basis of this approach (Fogel et al 1966) is that an organism (such as a finite state machine) is subjected to performance tests in some environment and, subsequently, to a mutation. The mutant undergoes the same testing and, in analogy with the "survival of the fittest", is retained at the expense of the original if its performance is superior. Various sophisticated methods for obtaining "offspring" from one or more "parents" may be used, and the evaluation criteria may be more sophisticated, but in essence the principle remains the same.

Several difficulties are inherent in this approach. The observed variability in the evolutionary process makes it unlikely that an intelligence, if produced, would resemble human intelligence. This may not appear at first to be a major drawback, but if effective communication is to be achieved, it is advantageous that there should be extensive similarities in operation of the two "organisms".

To steer the evolutionary process, the programmer must decide which features of his creatures and the environment are invariant (i.e. not subject to mutation). He must also decide on the right criteria of judgment that will optimize the evolutionary process and guide it toward the desired goal. It is difficult with foresight to be certain that the correct decision has been made to these questions.

A final difficulty lies in the degree of complexity achievable. While it is not certain that the human brain is the smallest possible size for the degree of intelligence it achieves, it may be supposed that an organism of comparable complexity will be required. The time and computing power required for this would be formidable.

(4) Man-machine Communication. Communication amongst human beings is undoubtedly an aspect of intelligent behaviour. This

may take place via the five senses (sight, hearing, touch, smell and taste), with normal emphasis on the first two. In man-machine communication, two subdivisions of the form of communication can be made:- non-interactive and interactive. In the former, information (visual, auditory or other) is input for further processing, or output after processing. The emphasis here is to produce means of input and output which are both convenient to human operators and also efficient.

In interactive communication there is an emphasis on the existence of a two way flow of information between the human being and the machine. Normally, the human operator will have control over the sequence of operations of the computer and may make alterations at any time. The computer keeps the operator informed of its operations in whatever form is appropriate to the task in hand. In this environment, emphasis on user convenience is of even greater importance: the user may wish to make immediate judgments without the necessity of interpreting data output by the machine. A reasonable way to achieve this is for the machine to produce its output in a form similar to that which would have been produced by a human being.

An alternative subdivision of the forms of communication may be made in terms of the activity in which the communication takes place: for example, in terms of picture processing, voice recognition and synthesis, interactive graphics, natural language recognition and generation, some of which activities are discussed later in this chapter.

2.3 Picture Processing

Some relevant areas of research in Picture Processing, and applicable techniques, are considered. An important ingredient of picture processing is the development of ways of describing a given input picture in terms of the objects represented therein. There are some parallels between this and the coding problem, that is, the problem of minimising the amount of information necessary to represent the picture in a particular problem, but, whereas in the coding problem the ultimate aim is to synthesize a tolerable picture from the description, the aim of picture analysis is to form a description which emphasizes significant aspects of the picture. The methods used reflect this difference. In many applications a sufficient description of the picture may be to place it in one of a number of categories - that is, "pattern classification" - and much reported work is of this nature.

Attention is now directed to methods used in picture processing, most particularly to picture analysis, as this is important to the work reported herein.

A useful reference to many aspects of picture processing is found in Kaneff (1970), which also covers other fields discussed in this chapter.

2.3.1 Template Matching

The template matching technique, employed particularly in character recognition systems, is one of pattern classification (Bledsoe and Browning, 1959; Kamentsky, 1961; Dye, 1969). It is assumed that there is a set of prototype images (the templates) against which input images may be compared. The comparison is formed in many ways, a common one being two dimensional cross correlation (Vander Lugt, 1964; Andrew, 1969).

The input images may be subject to preprocessing involving picture segmentation, gray scale standardization, image enhancement, size scaling, orientation and others (Sebestyen, 1963; Abend et al, 1965; Kanal and Randall, 1969; Holmes, 1966).

Perceptrons and modifications on these have been used as pattern recognizers (Rosenblatt, 1960; Widrow, 1964; Nilsson, 1965); they operate by calculating matches with many sparse (i.e. sparse over the input field) templates and classifying according to functions calculated over these matches.

Template matching is restricted in nature, requiring fixed ranges for size, location, orientation, gray scale and background in the input. Preprocessing assists in some of these aspects, but tends to be at least as difficult to achieve as does the matching itself. Attempts to avoid preprocessing have included optical correlations, i.e. forming correlations for all positions of the template (Vander Lugt, 1964), and correlations at various orientations of the template (Harley et al, 1968; Macleod, 1970).

A difficulty with template matching is that it treats the image as a whole, forming no effective detection of sub-parts (either organized or unorganized) within the picture.

2.3.2 Property List

This approach to picture processing evaluates a number of properties of an input image and, as a result of these evaluations, makes a classification of some form. It is expected, therefore, that objects which belong to the same categories will have similar values for the various measured properties. The correct choice of properties to be measured is thus most important. The method also finds application in character recognition and several surveys exist (Uhr, 1963; Nagy, 1968; Levine, 1969).

Three parts can be identified in the process - (i) preprocessing, (ii) property measurement (feature extraction), (iii) classification. The preprocessing phase is similar to that used in template matching. The choice of properties to be measured is of extreme importance; the success of any classification depending on this choice. Useful properties depend on the application and the classifications to be made, and the range is consequently wide. Cross correlations with a number of templates have been used. These may be either fixed and imbedded into the system (Hawkins et al, 1966), or they may be generated randomly and kept if they are found to be useful (Uhr and Vossler, 1961). Another commonly used feature is the calculation of various orders of central moments for the input image (Butler et al, 1969). Both these methods rely heavily on the availability of methods for segmentation of the input (if relevant). Rosenfeld (1962) uses statistical properties of an image to provide characterizations of more complex images, such as aerial photographs and cloud patterns. He uses statistical moments of gray level distribution and changes in these indicate possible segmentations of the image.

The classification phase has received most attention as it lends itself to the use of formal mathematical techniques. A common method of decision making regards each set of N measurements on an image as a point in an N-dimensional space. Transformations on this space will, if the features are appropriate, cause the points to cluster according to the classification to which they belong. Decision surfaces must then be found which separate the clusters with minimal error.

The property list method provides more reliable results than template matching in many instances; it suffers, however, from many of the same problems. The preprocessing task remains equally difficult. The approach is still not capable of detecting organized subparts of a picture. While feature extraction determines the presence of a number of properties, these properties are classificatory rather than descriptive and convey little regarding the organization of a picture.

2.3.3 Articulation

Template matching and feature extraction may be criticized due to their inability to recover information regarding the organization of the input image. A system capable of recovering this information would assist in the alleviation of the "input segmentation problem". The approaches described in this section address themselves to this problem; because of their nature, they lean towards description of an input image, rather than classification thereof. It is reasonable to suppose that if a description of an image is formed, based on relevant descriptors, the difficulties in classification of the image will be reduced.

Articulation may be defined as the process of imposing an organization onto some object of attention. Thus a property list approach, augmented by relational properties between the parts recovered, can be called an articular analysis.

Most approaches in this field contain, either explicitly or implicitly, a model of the patterns which are of interest. Such models consist of a set of rules for the construction of patterns or arrangements from simpler components (syntax rules), together with details of the simplest components from which the patterns are built. The models may be considered adequate (Lipkin et al, 1966) if they can generate only patterns similar to those in the input images.

Most of the work in this field has been involved with line-like input images, which may result either from extensive preprocessing, or the input may be assumed to be in such form (Evans, 1964; Guzman, 1968). Because of the use of line drawings, the relations used have reflected this e.g. "joined", "crossing" and "collinear". Only occasionally have more complex relations such as "inside" been investigated (Evans, 1964).

Grimsdale et al (1959), in an early example of this method, approach character recognition by segmenting the input into line-like segments and forming a description consisting of the line segments, their lengths, and their interconnections. This is compared with known, stored descriptions of the various characters, and a classification is given.

Evans (1964) assumes appropriate preprocessing and takes a set of line drawings such as in Figure 2.1. Descriptions of these are formed in terms of the connected subfigures, their compositions and interrelations. Relations such as "inside" and "outside" are important in this context. Evans uses similarity of organization to aid in determining which diagram is the odd one out.

Roberts (1965) and Guzman (1968) have investigated the problem of describing a three dimensional scene, seen as a two dimensional image. While Guzman starts with a line drawing coded as a list of vertices, Roberts describes a preprocessing system which eventually produces a line drawing. He then extracts information on the topology of polygons in the figure and, using knowledge of the topology of projections of selected three dimensional objects e.g. cubes and wedges, tests various models against the input, looking for matches. When a model "fits" part of an image, this model is included, in proper orientation etc., in the three dimensional scene which is being

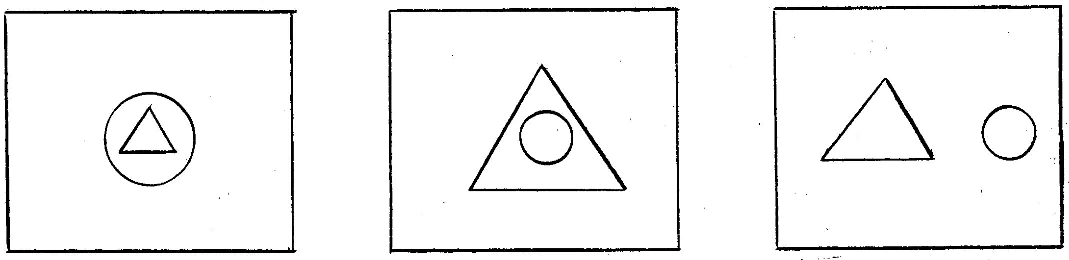


Figure 2-1   Drawings used by Evans in the geometric analogy program

built up. This work represents an important contribution through its inclusion of some "knowledge of the world" in the form of representations of cubes etc., as well as for dealing with 3-D scenes.

Guzman, in contrast, classifies the vertices in the input according to the number of lines in the vertex and the angles between them, incorporating knowledge of the world as heuristics based on the way in which three dimensional objects, when projected onto two dimensions, give rise to various types of vertices. By appropriate associations of regions whose edges form the lines at the vertices, he is able to generate a three dimensional description of the input scene.* Guzman's program is successful at this task.

The above examples represent some of the milestones in the development of the articular method, but other examples may be found in a review by Miller and Shaw (1968), and in Winston (1970).

The approach has led to better organized descriptions. There are still some shortcomings, however. The knowledge of the world which is built into these programs has contributed to their success, but it is still only very simple knowledge and improvements can certainly be expected, but conceptually and practically considerable difficulties are involved.

2.3.4 Parsing

This approach, to which the terms "syntax-directed" or "linguistic" may also be applied (Narasimhan, 1969; Miller and Shaw, 1968), is based on obtaining a specification of an image with respect to a generative model of such images. This is similar to the process of parsing a natural language with respect to a suitable grammar. The relationship between this approach and the articular analysis is clear. Picture parsing can be considered as a subset of articular analysis. The distinction is made here on the grounds that in picture parsing the syntax specifications are explicit and directly control the operation of the program. The number of levels in the description formed can

*although this 3-D description is implicit.

be arbitrary and may be several, whereas in the articular approach the descriptions are often not hierarchical, but in the event that they are, they have only a small number of levels.

The syntax rules give ways of combining various "primitive parts" into higher level constructs, and ways of combining these constructs into still higher constructs until the desired picture has been reproduced. The parser generates a sequence of applications of the rules which will produce the given input image. This represents an "hierarchical structural description" of the image. Various types of high level construct may be taken as representing different classes of interest, if this is the aim of the program.

The "primitive parts" referred to above can take many forms, but in all cases these are simple pictorial objects whose internal structure is not of interest. Such objects have a simple, fixed shape and are thus candidates for the simple recognition schemes outlined earlier.

A substantial effort has been made to develop schemes which facilitate the writing of grammars for general classes of pictures (Evans, 1968; Shaw, 1968; Stanton, 1970). Since these formalisms (or meta-languages) are intended to cover fairly general classes of pictures, there are variations amongst them reflecting their author's views on the nature of pictorial relationships. The Phrase Structure Grammar (Chomsky, 1965) was used in some early programs (Ledley et al, 1965; Narasimhan, 1966; Shaw, 1968) but more powerful techniques have been suggested (Clowes, 1968; Stanton, 1970).

Kirsch (1964) has given a context-sensitive grammar which will generate all and only 45^{o} right-angled triangles. While this grammar satisfactorily generates the triangles, it does not assign a structural description to the triangles.

Ledley et al (1965) have applied the techniques to the automated analysis of chromosome slides. Segments of boundary are classified into one of five types and the string of boundary segments (with their types) is analysed by a syntax-directed analyser according to grammars appropriate to two different types of chromosome. Measurements of certain parts of the chromosomes are made (according to the determination of parts by the analyser) for later use. Clowes (1968) points out that the grammar is not generatively adequate in that it allows the generation of non-chromosome-like pictures.

Evans (1968) has examined a meta-language and describes a picture analyser which accepts a grammar written in his formalism, and a list of primitives with their various attributes and relations. The analyser produces structural descriptions of parts of the input image which can be defined by the grammar. The grammars used by Evans' program consist of rules specifying the name of a construct formed by the rule, constituent sub-parts and significant contextual requirements, the relations necessary between the sub-parts, and attributes relevant to the new construct.

Shaw (1968) has reported a picture description language which is capable of both analysis and generation. His primitives can be of any form but must have only two distinguished positions (attributes) which are called head and tail. The primitives are assembled into constructs according to four simple rules which may be considered as forms of concatenation between the various heads and tails. Together with a few simple operators these form a phrase structure grammar. This system lends itself to the development of a number of theorems but, despite this mathematical nicety, it has shortcomings due to the simple nature of the allowed attributes and relations. Many useful descriptions can be expressed only with great difficulty or perhaps not at all.

Clowes (1969(a), 1969(b), 1970) has developed an extensive formalism for picture grammars. The capacity to name and classify images is present, but, more generally, the important aspect of recognition is seen as the capacity to compare descriptions of objects and to specify similarities and differences between them. The judgment of adequacy of description is based on the degree to which such similarities and differences are manifest in the descriptions. Clowes indicates the inadequacies of phrase structure grammars by pointing out that, in essence, the only relations that exist in such grammars are "parts of" and "followed by". His formalism is closely related to the transformational grammar (Chomsky, 1965), and his composition rules are dependent on explicit relations between the parts. Information regarding attributes of constructs is added to the description, and complex relations may be developed in terms of simple relations.

It may be noted that there is a close relationship between some of the methods used in this area and those used in the field of computer graphics (Sutherland, 1963; Williams, 1971; Maxwell, 1972). Both use highly structural descriptions whose relational information is similar.

In conclusion it is worth remembering that while the successively more complex descriptive techniques have achieved success in various domains, there has been an associated tendency to apply the techniques to images with simpler primitives. When more complex images, such as those handled by Macleod (1970) are involved, the development of adequate techniques has still a long way to go.

2.4 Man-Machine Communication

An essential aspect of communication between man and man or man and machine is the bidirectional flow of information. This section looks at natural language communication, while the next section examines graphical communication.

Before examining work in language communication, some of the problems to be faced are indicated. Firstly, it may be noted that, while "natural languages" refer here to languages such as English and French, these are not necessarily the most "natural" means of communication in every situation (Bobrow, 1970). For example, a scientist describes gravitational motion in terms of differential equations and it may be that in these mathematical terms, the most natural communication results. In other instances, repeated statements are replaced by a code or "push-button" for the sake of brevity.

To investigate the problems involved, consider a conversation in natural language between a man and a machine in more detail. Each portion of conversation by either man or machine may be called a segment. Starting with a segment from the man, the machine must act on this. The action taken is a function of the segment received and the environment in which the segment is relevant. Three requirements, in order that the machine can take action, can be identified; (i) the input segment must be parsed (analyzed); (ii) there must be a representation of the environment within the machine in a readily interpretable form; (iii) the input segment must be interpreted in terms of its parsing and the present state of the environment. Having produced this interpretation, the machine must then provide a further segment of conversation. This response should relate to the previous segment and to the present state of the environment. The state of the environment may have been modified by the input segment. There are several alternative possibilities for the type of action and response. For a declarative input segment there will almost certainly be some modification of the environment. The machine's response might then be a statement indicating that the input segment has been understood. Another possibility is that the input segment may have been a question, in which case the machine must produce an answer, and must first search the environment to determine the required information. In simple cases this might involve a search through a sequential file, but in the more general case a search through an elaborate data structure is called for. It may be necessary for some computations to be performed on the information retrieved. It cannot be assumed that the machine will be told how to perform this computation, consequently a problem solving ability will be required.

The natural language machine indicated above would be very powerful but the features involved have yet to be achieved in machines already developed. Some of these are mentioned below.

The well-known ELIZA system (Weizenbaum, 1966) converses in the manner of a psychiatrist examining a patient; it poses but does not answer questions directly. The operation of the program is simple. Each segment of text is read in and scanned for certain "key words". An important key word is chosen and an associated transformation is made on the input. These transformations are in fact text, forming the basis of the response to the input. For example the key word "can" has associated with it "Does it mean a lot to you to". Other heuristics are applied to extend the naturalness of the conversation. There may be more than one action for a given input pattern, and an element of randomness is introduced in the choice between these. ELIZA keeps a record of things considered "important" to the human being, and, in case the computer can produce no other response, returns to these and asks a question.

The ELIZA system has very little knowledge of the environment. Its analysis of the input is also very simple, choosing key words and making no attempt to parse the input. Nevertheless, by making a few assumptions and relying on some false assumptions by the human being, it produces an acceptable conversation.

STUDENT (Bobrow, 1964; Minsky, 1969) is a system with a much more limited context. This program accepts high school algebra problems posed in natural English. A typical question is

> If the number of customers Tom gets is twice the square of 30 percent of the number of advertisements he runs, and the number of advertisements he runs is 50, what is the number of customers Tom gets?

STUDENT operates on this input with a set of transformations, converting the input into a set of simple statements which can be represented as equations. The program solves the equations and outputs its answer. If it cannot find a solution, a number of transformations on the equations are tried and, if necessary, the program searches its "memory" for any relevant general information. If it still cannot find a solution, STUDENT may ask the operator for additional information.

The transformations used by STUDENT are again based on the key word concept. A selection of "operator" key words is recognized, and these are used to set up relations which form the basis of the equations. STUDENT assumes that, with few exceptions, the same string of words will be used for the same concept, and this allows it to identify variables. A number of ad hoc pieces of global information are employed rather than attempting to incorporate a true knowledge of the world. Global information such as "force equals mass times acceleration" may be accumulated by the program.

An important feature is the introduction of a specific environment or "knowledge of the world". This enables the program to perform in a more natural way than if everything were spelled out for it.

A program which attempts to make a more complete analysis of the input statement is the Question-Answering system of Woods (1967), which answers questions concerning an airline guide. Woods assumes that the input questions can be analyzed into a deep-structure parsing via a transformational grammar (Chomsky, 1965). While this is not strictly justified, sufficient approximations exist for his purpose. His data base (from which the answers to questions are obtained) is defined in terms of primitive objects, functions on these objects, and relationships between objects which can be tested by various predicates. A parsing of the input question is generated and, with the knowledge provided by this parsing and the recognition of words relating to various primitives in his data base (such as meal, flight, arrival time), a predicate is developed which, when tested, answers the question.

In all systems requiring an extensive environment or knowledge of the world, the problem of organization of this knowledge must be faced. Woods makes certain assumptions which, while making some restrictions, allow him to set up a precise data base. The organizational question is not solved by this, however, as the data must also be organized in such a manner that the predicates are readily testable. The solution to this important problem varies with the application. No general methods appear to exist for adequately representing this semantic information.

Many other examples of natural language systems (Charniak, 1969; Lemmon, 1969; Quillian, 1966, 1969; Raphael, 1964; Craig et al, 1966; Barter, 1970) exist but, for the purposes of this discussion those outlined have served to indicate typical successes and limitations. It is interesting to note that most of these are non-learning programs, and can therefore function only within a fixed range of stimulus types. While the programs may improve in performance, this is due to an accumulation of data within the fixed framework. There is a dichotomy between the processor and the data base or environment. The data base is essentially a passive body of knowledge which is modified, built up, searched, etc. by the processor. The procedures within the processor do not themselves change. Learning new forms of behaviour is an important consequence of human communication and this will be achieved in machines only if the processor itself is subject to modification. When this is so, the distinction between data base and processor will be less clear.

2.5 Interactive Computer Graphics

Many of the remarks made in the previous section on natural language communication apply also to Interactive Computer Graphics. The development of computer graphics has followed a different course, however, and many systems bear little resemblance to natural language systems. The discussion of Computer Graphics (from now on Computer Graphics or Graphics will be taken to mean Interactive Computer Graphics - all the systems considered are interactive so that no loss of meaning occurs with this abbreviation) is in general terms with some of the provisions and problems of systems being mentioned. Discussion of detailed systems is postponed until Chapter 3, when some examples of programs closely related to the present work are reviewed.

Cathode-ray tubes have been available with computers for some time, but their use was initially restricted to output functions, e.g. the Whirlwind computer at MIT had such terminals for debugging (1956). The first real use of the display of an interactive role with effective input and output was the SKETCHPAD system (Sutherland, 1963). This program demonstrated the use of the light pen in drawing pictures on the display while the computer monitored the pen's motions and internally generated a data structure representing the picture. The structure represented topological features of the picture, and elements of the structure displayed on the screen could be selected with the light pen so that non-pictorial attributes related to them could be typed in. There was also a facility for specifying relations between pictorial parts, and these served as constraints on the relative behaviour of the parts in any subsequent manipulation. The SKETCHPAD programs (also SKETCHPAD III (Johnson, 1963)) exhibit the two important functions of the display console - first as an input/output device that can display and accept pictorial data, and second as a means of controlling the sequence of a program. This latter is not unique to displays but it is convenient at the task.

Attention has often been focussed on the sketching capabilities of graphics, but it should be noted that other functions are possible and often desirable e.g. text editing (English et al, 1967), and file manipulation (Bennett et al, 1965). It has been pointed out (O'Callaghan, 1971) that SKETCHPAD did not allow "sketching" in the true sense of the word. Rather, the user constructed images in two steps. The first entailed creating a line by pushing a button, thereby producing a line with one end on the tracking cross, then moving the other end into position with the light pen. The second entailed specifying constraints applicable to the line. The majority of systems* generate their images in this manner.

The nature of the data structures used varies considerably. In simpler applications, display information may be maintained in an array with information correlating names of entities given

*which allow the user to input and edit drawings.

by the user to the representation on the CRT. In more complex applications where there may be a hierarchy of sub-pictures and where various manipulations are allowed, more powerful data structures are called for. The structure and its associated procedural support must be able to correlate the structural item and displayed item as in the simple case. Provision must be made for reference to be made to objects as parts of other objects in a hierarchy. The structure can be thought of therefore as a dynamic tree structure. This "structural description" is normally implemented in one of the many list structure techniques available (e.g. Weizenbaum, 1963; Sutherland, 1965; McCarthy, 1966) All these systems allow reference to an entire structure through a "header", permitting complex operations to be performed simply.

Until recently most systems used one data structure to contain information on the displayed picture and problem domain information. Because of the complexity and variability of much of this data, the structures had to become rather unwieldy with variable length blocks, intersecting structures etc. More recently, two independent data structures have been advocated. (Johnson, 1968). One of these refers entirely to the displayed image while the other contains problem domain information. This leads to more efficient problem domain manipulations and also to more efficient display operations. The relationship between these two structures is usually developed during the construction phase; if they are produced together step by step, there are few problems involved. If, however, only one data structure, say the problem structure, is available, there is considerable difficulty in producing a corresponding pictorial structure and its associated image. This is a manifestation of the problem of "relation of representation" which is discussed by Clowes and his co-workers (Stanton, 1970b), who view a picture as being a representation in the pictorial domain of some situation in the problem domain. These two are related by this "relation of representation" and, in general, some form of transformation must exist between the two. The nature of this transformation is only vaguely understood and, where this type of transformation is attempted, must be worked out for individual applications. The situation of transforming between two such domains is usually avoided by virtue of the simultaneous development during the construction phase as indicated above.

Considerable work has been done on program support for displays and in developing optimum system configurations for graphic systems. This is largely outside the scope of the present report. Much of this work is reviewed in Johnson (1968) and Williams (1971 - Data structure review).

Graphics is a form of communication and some remarks on this aspect are now made. Narasimhan (1969) considers graphical communication as being made up of two languages. One is the language of the graphics or drawings, and the other (which he calls the discourse language) contains methods for issuing commands, asking questions, providing additional data and so on. If the dual data structure philosophy is taken, then the discourse language may refer to either of these structures. Human beings are adept at communication in graphical language. For example, road signs are erected which instantly convey information to the driver, stories are illustrated with sketches. To date, however, this degree of adeptness does not exist in computer programs. The user is forced to specify his data by unnatural means and results are usually presented in inflexible format. In the long term, improvements in program performances may be achieved when the ability to communicate graphically is improved. It may be noted, however, that the natural language programs discussed in the preceding section performed adequately when provided with a knowledge of the environment concerned. The situation is the same with Interactive Graphics. If adequate knowledge is present, the program is able to make appropriate choices available to the user, thus easing his difficulties. A particular area in which very little information is incorporated in current systems, is that of the "relation of representation". If this information were present in machines, the user would be able to specify his problem situation directly and be relieved of the laborious task of generating the display by the "pick and place" method in common use, as the machine would have the knowledge to generate this display itself. The work reported here aims to provide just such a feature.

CHAPTER 3
SOME RELATED SYSTEMS

3.1 Introduction

The discussion of Interactive Computer Graphics given in section 2.5 introduced relevant background material; the following sections expand this aspect and discuss some programs closely related to the present work. In providing this, no attempt is made to cover all the systems available; rather, three programs have been selected for detailed discussion. These programs all provide representative but different approaches to similar problems.

3.2 The System of Bracchi and Somalvico

3.2.1 Objectives

This system is outlined in a paper entitled "An Interactive Software System for Computer-Aided Design: An Application to Circuit Project" (Bracchi and Somalvico, 1970). The orientation of the system is, therefore, towards a design environment. The authors point out in their introduction that the roles of man and machine in a design environment are complementary: the machine performing computations with speed and rigour, presenting its results without bias; the man providing invention and adaptation. Their claim is that convenient interaction in which these complementary roles may be best performed has not been realized due to the complexity and difficulty of use of the interaction methods for the user. Bracchi and Somalvico set out to produce an interactive software system which provides solutions to these problems. The approach to generality is taken by developing two levels of language - one which controls the interactive functioning of the system, and a second which provides problem-oriented programming facilities (in their case for electrical circuits).

3.2.2 The System

Their philosophy is that a designer will wish to perform a large number of calculations in differing sequences. To allow this, the application programs are segmented in a highly modular fashion. Performing the various functions in the specified order is one of the functions of a graphics monitor (INGRAM).

There are two levels of commands available within the system. The first is the language IMOL (interactive monitor language), used to control the sequence of operations of the system i.e. the interaction. The second level is the language for developing application oriented sequences, COIF (circuit oriented interactive Fortran).

IMOL

The IMOL language is intended to provide the user with a general control mechanism which can be used in several applications. Input in this language is either through a teletype or via the "light button" method. The language consists of a set of commands which can be divided into a number of functional classes (FC's). These FC's are grouped into two subsets which may be regarded as "high level" and "low level". The low level classes are:

(1) Program functions, providing entering and editing facilities for the application programs which are to be input.

(2) Operate functions, providing facilities for the selection of appropriate areas for storage of application programs and data. The order of execution of application programs may be specified.

(3) Data functions, allowing the user to enter and output data sets associated with the application programs.

Blocks of commands from each of these classes are interpreted for execution by three separate interpreters.

There are also three FC's in the high level subset, whose functions are:

(1) Construction, providing the highest level control of a session.

(2) Separation. The low level commands are given in groups according to the three FC's indicated above. The separation functions serve to deliniate these and bring the appropriate interpreter into action.

(3) Execution, providing control over the translation and execution of the various groups mentioned above.

The use of these commands is illustrated later in an example of the operation of the system.

COIF

IMOL does not provide for the graphic manipulation associated with the application; this is provided by the COIF language. Bracchi and Somalvico have imbedded this language into Fortran for the convenience of new users. Of the two approaches available to the imbedding i.e. provision of a set of subroutine packages, and additions to the Fortran language, Bracchi and Somalvico have chosen to provide extensions to the language.

The additions to Fortran are of four types - definition, specification, manipulation and control. The definition primitives provide for the establishment of structured blocks of storage representing particular circuit features, and associated with these, are circuit graphic variables which must be declared in a specification statement e.g.

```
CIRCUIT  VAR
VAR = NODE (400,200)
```

establishes a node at location (400,200) on the display.

Expressions containing circuit graphic variables may be used to create a variable representing several items. It has been indicated previously that an hierarchical organization with multiple instances of parts of structure is a common requirement. This facility is provided in COIF by the circuit functions which allow the definition of a block and the use of that block as an element at other times. For example, the function FIL is shown below and its display is shown in Figure 3-1.

```
CIRCUIT FUNCTION FIL (N1, N2, N3, N4, INST)
N1 = NODE (X,Y)
N2 = NODE (Z,V)
N3 = NODE (V,R)
N4 = NODE (T,S)
LA = INDUC (N1,N2,B1)
 C = CAPAC (N2,N3,B2)
LB = INDUC (N3,N4,B3)
FIL = N1 + N2 + LA + N3 + C + N4 + LB
RETURN
END
```

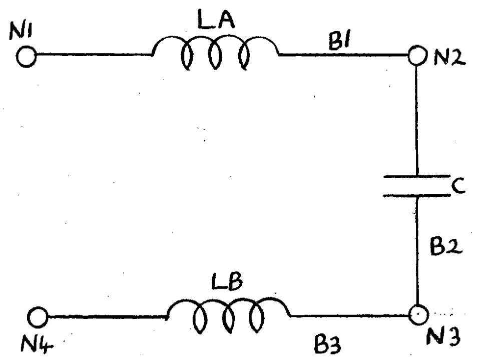


Figure 3-1 A subschematic

The manipulation primitives in COIF provide facilities for rotation, displacement, and scaling of various parts of the display. The control primitives provide for display of the generated picture, deletion of various parts of the picture, erasure of the structure and so on.

As an example of the use of the system, a portion of a session is shown in Table 3-1.

3.2.3 Discussion

The stated intention of developing a system which is effective in its application and is easily adaptable to other types of problem has been achieved by the use of a separate control language IMOL. While this language is general in the sense of its application independence, it could best be considered as a form of high level editor rather than as a communication language. Its interactive features are not directed towards a display, and could be used without a display at all.

One of the main purposes of the COIF language is to develop the electrical connectivities which are to be fed into an analysis routine. It could, therefore, be input along with the analysis routine and be run in batch mode. The major advantage of the system is the immediate turnaround available in the display environment. Display of the circuit serves as confirmation that the COIF code has been written correctly.

An important requirement of display environments is the ability to make rapid modifications to the circuit and observe their effects. To make such changes in Bracchi and Somalvico's system would require rewriting the COIF program. Depending on the changes made, this may be a lengthy task and the system cannot be considered an example of natural communication.

The system has no specific knowledge of circuits; consequently there is no facility for it to aid the designer in the production of either his circuit or the corresponding circuit diagram. Each individual element (with the exception of those elements which are parts of blocks defined as circuit functions) must be selected, its position on the screen determined, and its

Table 3-1 A Run of the System

Statement	Comment
INIT	Initialize software
USER BLOGGS	declare designer name
STARTS	between this and ENDS defines a session
SESSION CIRCAN	define name of session
ENTER(P) PROG 1	the following statements will define an application analysis program
MEM(P)(PROG1,MEM1)	storage for PROG1 is assigned
EXEC(P) PROG1	PROG1 edited, compiled and stored in MEM1
ENTER(D) DATA1	the following statements define the data to be used by the application program
MEM(D)(DATA1,MEM2)	
EXEC(D) DATA1	Output of DATA1 stored in MEM2
ENTER(O) OPER	
DATA(DATA1,MEM2)	DATA1 is selected from memory
PROGRAM(PROG1,MEM1)	PROG1 is selected. It will use DATA1
RESUME	end at operate segment 'OPER'
MEM(O)(OPER,MEM3)	select file for subsequent output
EXEC(O)OPER	PROG1 now runs with output onto MEM3
SAVE CIRCAN	keep this section
ENDS	end of session
EXIT	leave software package

symbol placed accordingly. This procedure becomes lengthy even for relatively simple circuits.

3.3 Belady et al - DESIGNPAD

3.3.1 Objectives

DESIGNPAD is outlined in "A Computer Graphics System for Block Diagram Problems" (Belady et al, 1971). The authors start from the premise that human beings prefer to communicate complex ideas symbolically by graphical means. They consider that such communication with a computer is still in a very primitive stage, and furthering this evolution is an important goal. They note that many systems have been outlined for communication in various applications: all these suffer, however, from the need for frequent major reprogramming in the face of small changes in application. (Kuo et al, (1969) and Sutherland (1966) are quoted as examples of electrical network and flowchart programs which, although they have similarities in the problem area, are completely incompatible programs). Because a large number of applications use labelled block diagrams, the authors set out to develop a general system for such diagrams.

3.3.2 The System

A labelled block diagram is defined as a set of interconnections (or lines) between "attacher points" of blocks, with text possibly associated with any of the elements. Some examples of the types of diagram of this form are shown in Figure 3.2, which shows the wide range of applicability of the system. DESIGNPAD is intended to operate with a large scale host machine, together with a small scale satellite terminal. The division of labour between units is achieved by allowing the structuring of block diagrams (or modelling) to be restricted to the satellite, with a work load being sent to the host when application analysis is required. In this arrangement the major portion of DESIGNPAD is concentrated in the modelling subsystem within the satellite.

A major principle behind the operation of DESIGNPAD is the use of "modelling sheets". These supposedly provide an easy transition for users accustomed to drafting boards and those

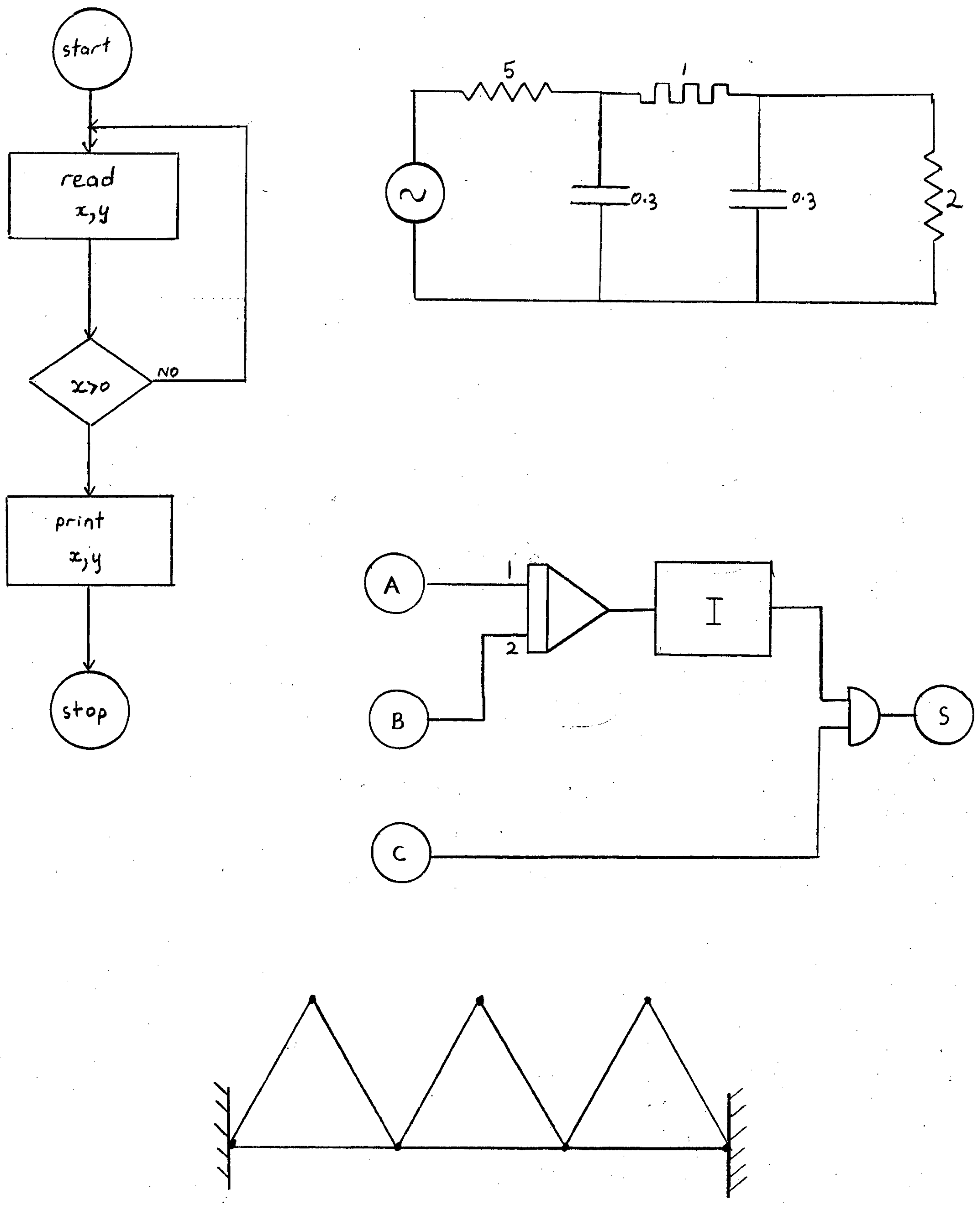


Figure 3-2   Some labelled block diagrams

accustomed to conventional time-sharing systems. The sheet acts like a large piece of paper (about sixty feet square) and it is on this sheet, a portion of which is displayed on the C.R.T. screen, that all graphical information (either textual or purely graphic) is entered. There may be many sheets per user, each with its own label. In operation, a sheet is displayed on the screen, a model constructed thereon by the user, and the contents sent to the host for analysis. A library sheet, containing names of other sheets, names of the various blocks available to the user, and other names, is kept as a special sheet. To create a model, the user draws lines and selects and positions blocks which have been chosen from the library sheet. The blocks may be sheets defined by the user at another time. This allows a hierarchical and recursive model structure to be built. These features are illustrated in Figure 3.3, together with the "viewport" and "window" concepts which are explained next.

To examine input or output, facilities are provided which allow the screen to be divided into as many as four viewports, each displaying part of a different sheet. Typically, one viewport may contain the model under construction, another the various blocks available, a third a selection of "light buttons", and a fourth the results of a previous analysis. Since scaling is not available within the system (as built), only a portion of a sheet may be displayed in a viewport at any one time. By constructing within a window and then moving the window, the entire sheet may be utilized.

A drawing package is provided to assist the user in the construction of his model. With this package, individual blocks are copied from a reference sheet to a specified position on the model sheet. Each block has associated "attacher points", between which lines may be constructed. A line may possibly not end on an attacher point, in which case it is called an endpoint and may be connected to further lines. As it is often desirable to add text to a diagram for its completion, text editing facilities are provided. If text is mixed with blocks and lines, the text may be associated with particular blocks or

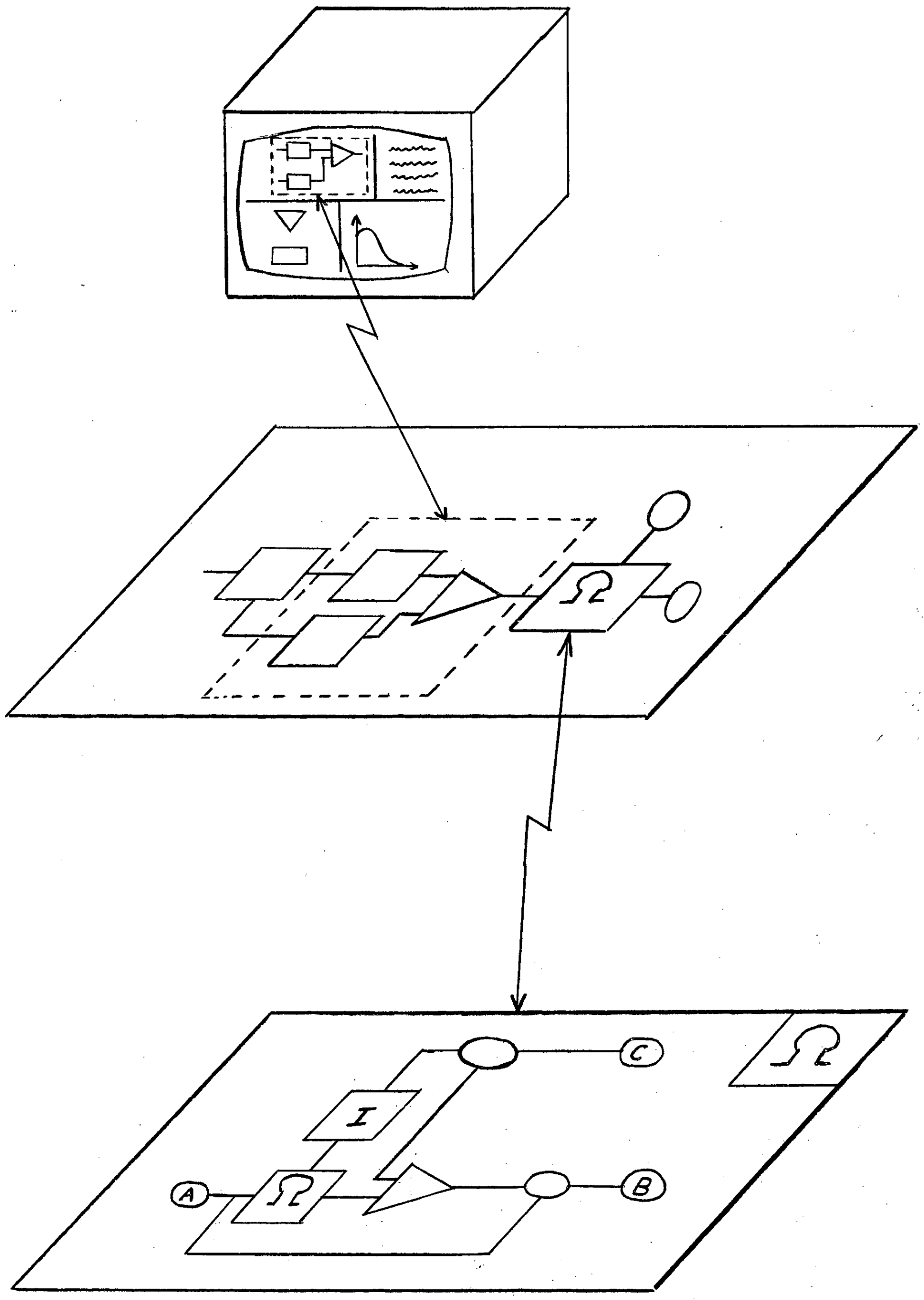


Figure 3-3 Illustrating windows, viewports, and hierarchical, structures in DESIGNPAD

attacher points. These links are provided by graphical entities called "hooks". Objects treated by the drawing package are shown in Figure 3.4.

During the drawing operation, two viewports are reserved; one is for a reference viewport displaying a sheet containing available blocks, the other is for the modelling sheet. Most drawing packages require that specific commands be given for each operation, along with appropriate operands. The DESIGNPAD drawing package, however, does not need explicit commands. It is able to determine the required operation from the nature of the operands. For example, the selection of a block in the reference viewport, followed by the selection of a point in the modelling viewport, indicates that the appropriate block is to be copied into the modelling sheet at the specified position.

Extensibility in DESIGNPAD is provided by the provision of a "block specification" mode of operation. There are two phases in this mode: in the first, the user specifies the shape and number of attacher points of the symbol which is to represent a new block; the second phase involves the specification of the function of the block. One way of specifying this function is to draw a network on a sheet, thereby providing the hierarchical facilities. Since it is possible to specify the block being defined in this network, recursion is possible.

DESIGNPAD provides a mechanism for saving individual sheets. During the construction phase of a model, two separate files are maintained within the system. The first is the actual display file, the second is an action file which is a list, in the order of creation, of all the actions taken by the user during the construction of the model. While a sheet is being displayed, it is kept in the satellite's main memory, but after this it is held on the satellite's disk. The capacity of this disk is such that files can be expected to be erased after a short time. Upon receipt of a retrieval command, the system first looks on the satellite's disk, but if the file is not present, requests the action file from the host. Since this file contains precisely

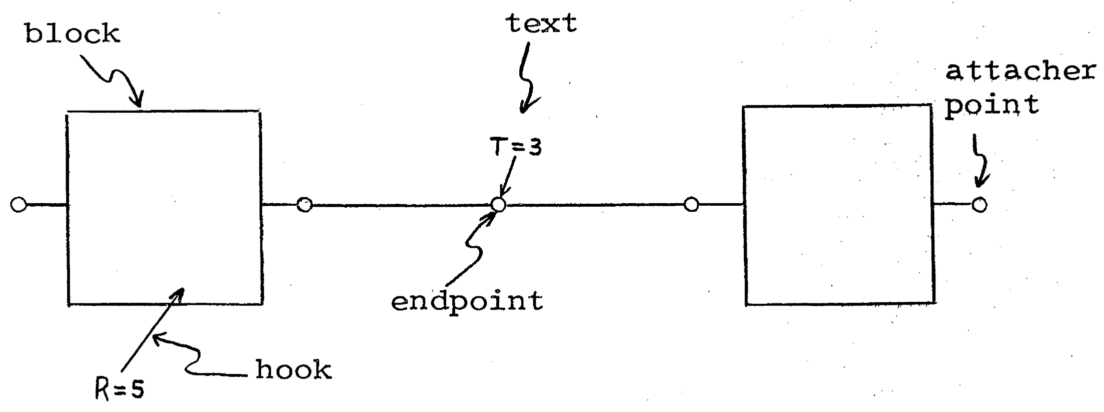


Figure 3.4   Graphical entities in DESIGNPAD

the same information as specified by the user, the same routines may be employed to recreate the display file as were used originally.

A final mode of operation of DESIGNPAD is entered when the user has completed his model and wishes to perform some analysis. Application programs in DESIGNPAD are normally supplied by the user, and therefore some form of interprogram communication is needed which is independent of the detailed internal operation of DESIGNPAD and is provided by the "interface file". Associated with this mode of operation is an output package which provides facilities for representing output from the application programs. Facilities exist for plotting two dimensional graphs, bar graphs, three dimensional projections, and so on. These forms of output are converted into a sheet which may be inspected by the user as in the case of any other sheet. An additional facility allows the application program to output an action file so that block diagrams, text etc. may be produced as output. Table 3.2 contains a selection of operations from the modelling subsystem - descriptions are given for some of these.

The interface file, mentioned above is of special interest as it is this file which forms the link between the purely graphical descriptions in DESIGNPAD and the application programs or real world environment. Section 1.2 indicated that this linkage between the pictorial domain and the problem domain is of special interest to the present project. The interface file is built up from the individual transactions as they are sent to the host i.e. it is effectively built from the action file. The interface file is more complex than the others and is made up of a set of objects, attributes of those objects, and relationships between them. This contrasts with the other files which are essentially sequential.

The file starts out with an object entry for each block, text and hook. Various attributes such as number of attacher points are associated with each block. Two further objects are added. The first is a "set of all blocks" object, and "member

Table 3-2 Some DESIGNPAD Commands

Facility	Command	Comments
Start-up	Cold start	complete start from scratch
	Warm start	start as left by user
Control	Viewport size	viewport boundaries move with pen. A "windowing control" appears in viewport and use of light pen moves sheet continuously through the viewport
	Clear sheet	
	Analyse	choose application program and model in modelling viewport is analyzed
	Move	Selected block moves with pen. Attached lines stretch appropriately.
Drawing	Draw line	select start then move pen to required end of line
	Copy block	point to block in reference viewport, then move to desired position in modelling port
	Erase line/block	
Text editing	Begin text	select point in modelling viewport then enter text
	Edit text	
	Relate text	Select text then line, block or attacher point. A hook relates the two
Block definition	Specify shape	Draw large version of block. Scaled (to actual size) version appears in corner
	Label block	enter alphanumeric label
	Specify function	modelling mode entered to define content of a block

of" relations are created between it and each block. A similar block is defined for lines. The next step is the creation of "point objects". These represent either attacher points or endpoints or both. A further set of relationships is added which relates each "point object" with the lines and/or blocks with which it is associated. The first step in the creation of the interface file is the creation of a set of objects which represent attacher points which are "wired" together by lines. A relation is created between each such attacher point and the associated linkage object.

In many applications the positioning of lines joining attacher points is of no importance - only the connection itself is relevant. The last objects (i.e. those representing "wired" attacher points) added to the structure are particularly useful in such applications, and they represent perhaps the only non-pictorial information present in the interface file. In some applications the positioning of lines may be important and it is thus necessary to leave all the individual lines in the structure.

3.3.3 Discussion

The authors have succeeded rather well in their attempt to provide a block diagram system which has general applicability. Critical discussion of DESIGNPAD project should be viewed in the light of this latter objective, as well as in the terms of the system itself.

Before discussing design objectives of DESIGNPAD, comments will be made on the system as it stands. Belady et al suppose that human beings show a preference for communication with the use of sketches, often hastily drawn. If it is assumed that a similar form of communication would be desirable when trying ideas on a computer, then the question may be asked as to whether the DESIGNPAD system actually provides this facility - it does not. The actual drawing operations used are a process of selection of object types, positioning of blocks and endpoints of lines and so on. While this type of operation is quite commonplace among graphics systems, it could barely be called sketching. This process has been criticized (O'Callaghan, 1971) on the basis of its being an unnatural and frequently time consuming operation. The language of communication is not graphic but rather is instructional, that is, giving instructions on the creation of the diagrams. The alternative to this is true sketching. Sketching implies that the system must distinguish or recognize imperfectly drawn objects, implying pattern recognition of some complexity and, while a large number of elaborate procedures exist in this field, none appear to be up to the task of successfully and continually recognizing the content of a large sketch. While certain simplifying assumptions may make the task feasible, the computational complexity of many of those procedures would probably make them unsuitable for a rapid response graphic system.

Turning to the question of generality, this DESIGNPAD objective has had a major effect: it seems impracticable for the system to include knowledge of the problem domain - this places an extra load on the user. Each application has its own conventions, typical groupings etc., and human beings assume these to simplify communication. In DESIGNPAD the user must spell everything out, as the machine cannot make these assumptions. The inclusion of problem domain knowledge would make it feasible for the machine to make similar assumptions, considerably reducing the quantity of information which the user must transmit. The inclusion of such knowledge might also be of assistance when an attempt is made to develop a system in which true sketching is permitted.

3.4 The System of Evans and Katzenelson

3.4.1 Objectives

Evans and Katzenelson (1967) describe a communication program intended to be used in conjunction with the AEDNET non-linear network simulator. It is convenient in this discussion to refer to the communications program as AEDNET, although this usage is strictly incorrect. The environment for this system is similar to that used by Bracchi and

Somalvico, their primary objective being the same - viz. to provide convenient interaction allowing full advantage to be taken of various available analysis programs. Evans and Katzenelson place special emphasis on "convenient and natural" communication and lay down three specific requirements:-

(1) The network is specified and modified graphically by drawing on the oscilloscope; results are plotted as graphs.

(2) The user communicates in network terms which are as close as possible to the terms he normally employs when thinking about network design.

(3) A complete mechanism of error checking and error comments so that the system will be "fail-safe" in case of human operator errors.

3.4.2 The System

There are four features which characterize the command language which forms the interaction with the system. These are:-

(1) Operations are specified by single commands followed by sets of arguments. Each command corresponds to a "network operation".

(2) Error checking and error comments exist for each command.

(3) The use of implicit arguments reduces the amount of specification required by the user.

(4) The user can queue commands and their arguments.

The communication program is built from a set of elementary operations relating to manipulations of the data structure. Groupings of these operations form more complex functions which represent typical network operations. These complex operations correspond to the various commands available to the user. As an example, an elementary data structure operation is "remove an element from a ring". A complex operation making use of this simple operation is "delete a component from the network". This forms the basis of a user command in which the user selects the operation, then indicates which component is to be removed.

Commands are selected using a set of push buttons associated with the display, each command requiring a number of operands which may be entered in one of two ways:- the first is by selecting the appropriate object with the light pen; the second by typing in the object label on the teletype. Either method avoids having to specify formal arguments for each command.

A network is generated in AEDNET by a method which is conceptually similar to that described in DESIGNPAD; that is, a process of "pick-and-place" is involved. A difference here, however, is that the terms used relate more to application than is possible in DESIGNPAD (the items are more application oriented). The operation can be seen from the following set of commands which generate a small portion of a network.

(1) "Create a node and display it at the location of the tracking cross"

(2) "Create an element, connect to it to the two nodes to be specified by the light pen, and display the result".

(3) "Give a label (from the teletype) to an element or device to be specified by the light pen and display the label".

The error-checking facilities are based on an examination of the arguments of the various commands. The first level of examination involves inspection of the type of incoming items (is it characters on teletype, integer number, floating point number, light pen "see", button push?). A match must exist between the incoming item and the desired argument type to prevent the occurrence of an error condition. The second level of checking involves the nature of objects selected (via light pen or teletype label input). For example, if an operation requires an element as an operand, then a node will be rejected as an error. The existence of queued commands means that an error in a command may affect subsequent queued commands;

therefore an error causes the current command and all subsequent queued commands to be ignored.

There are a number of network concepts in AEDNET which are fundamental to the organization of the data structures. Corresponding to each concept, an "entity" is defined which has associated properties and may be the subject of various relations. The entities are node, element, generic element, network, device, and generic device. A node corresponds to the normal electrical concept and is represented as a point in the display. An element is a network component that can be analyzed directly by the analysis program. An element has three properties - a type, an identifying name or label, and a relation between the network variables at its poles. In these definitions the poles are properties of elements as opposed to nodes (which are separate entities). Elements are connected to nodes by attaching one of their poles to the node. AEDNET recognizes only a small set of simple elements (resistor, capacitor, inductor, voltage source, current source). A "short circuit" element is included for graphical convenience and serves to relate nodes in the analysis system.

The generic element is associated with a particular element type, and its properties are those common to all elements of the given type. For example, the resistor generic element has properties indicating the number of poles on a resistor (i.e. 2), the resistor's variables (i.e. voltage and current), and the resistor's display symbol.

The concepts "network" and "device" are defined recursively. A network is a configuration of nodes, elements and devices. Each pole of each element and device is associated with one of the network nodes. A device is defined as an entity similar to an element but with slightly different associated properties. The three main properties of a device are:- a type, a name or label, and a network which is the device's equivalent circuit. A generic device is also defined which serves a similar function relative to devices, to the function served by generic elements relative to elements. With this recursive arrangement, complex circuits may be defined as devices and then used in subsequent circuits, allowing for the development of very complex circuits.

Three forms of relation between entities emerge from the above definitions: the first relates a node with all poles incident upon it (connectivity); the second associates members of a given set; and the third associates devices with their equivalent circuit.

The data structure of AEDNET is based on the network entities discussed in preceding paragraphs. Various entities within a network are represented by "beads" (consecutive groups of locations containing information pertinent to the entity, and also pointers to associated "beads"). In AEDNET, unidirectional rings are formed from beads which are related (in one of the ways mentioned above) and a header bead supplies information regarding the nature of the relation. This structure is principally a network structure but display (i.e. pictorial) information is tied tightly in with the structure.

3.4.3 Discussion

Evans and Katzenelson set out within the framework of a pre-existing analysis system to develop an interactive communication system which allows convenient operation for the user. The system is in some respects related to the DESIGNPAD system but, by considering a restricted application domain, it has been possible to include a small amount of problem domain knowledge. This knowledge manifests itself in the data structure which is primarily problem domain oriented. The objects represented in the data structure are oriented towards networks, and consequently user commands can be made to appear as network commands.

The use of one data structure for both pictorial and electrical information leads to a close intermingling of graphic and network commands. For example, the commands given in the previous section for generating a portion of a circuit, combined both pictorial aspects and electrical aspects. The electrical aspect was the connection of a resistor between nodes, while the pictorial aspect was the positioning of those nodes on the screen. Since the machine's pictorial and electrical information is viewed as one body of knowledge, it seems difficult to provide assistance in the construction of a diagram. If the machine knew how circuit diagrams were arranged, it would ease the "pick-and-place" task placed on the user.

3.5 Summary

Three systems have been studied, and it may be seen that, although they investigate similar problems, each has taken a different approach to solution. All have claimed to be interested in "natural" communication between user and machine, and this should apply to pictorial and discourse communication.

Bracchi and Somalvico have done little to achieve natural pictorial communication. Their system could best be described as a convenient, interactive programming system. A picture is generated by writing a section of COIF code, and this in turn generates the diagram. A disadvantage of their method is that the COIF code is not executed at the time that it is compiled; this does not allow the user to detect errors immediately. Their technique cannot be considered as a candidate for natural pictorial communication.

While certain primitives in COIF could be considered to be circuit oriented, these primitives are not conveniently available to the user (the problem domain portion of the communication also involving the creation of code). Circuit communication is also, therefore, not "natural" in this system.

The DESIGNPAD system achieves greater success in graphical communication (than does the COIF language). Here, the user is able to create his diagram directly on the screen, without the necessity of specifying coordinate values and so forth. This system uses the pick-and-place method which has been criticized in this report and elsewhere (O'Callaghan, 1971). Since no problem domain knowledge is available, the user must employ purely graphical terms to create his network, so that no assistance in the layout can be given by the system.

Evans and Katzenelson also use the pick-and-place method in the construction of circuit diagrams and networks. In their case, however, the specific domain has enabled the use of problem oriented terms in the discourse. These terms are merely labels for the structural entities - the machine contains no information as to the meaning of the terms; high level names represent only pieces of picture. The user's task is, therefore, only slightly eased.

All these systems have the need for, and suffer because of the lack of, adequate graphical communication methods. A possible solution to this problem involves addition of a true sketching capability. As previously indicated, this requires presently unavailable recognition techniques, and the user still has to handle the complete layout burden.

In order to alleviate the above limitations, the writer believes that a more satisfactory approach is to shift the emphasis of communication from the graphical domain to the problem domain, by allowing the user to communicate both simple and complex application situations to the machine directly. Achieving this aim requires the incorporation of knowledge not present in any of the systems discussed (or any other known to the writer). This necessity is illustrated by considering the following example; a user may say "I am interested in a two-stage transistor amplifier with a pi-filter on the input". Such a command is purely application oriented and contrasts with those of the systems described (they are pseudo-graphical). The machine must know not only the make-up of various types of circuit, but also must know how to draw their corresponding diagrams: this is so because the consequence of the above command must be that the circuit diagram for such a network appears on the display with no further assistance from the user.

While the systems described in this chapter allow for the definition of blocks of circuit and diagram, these blocks do not satisfy the above requirements as the systems cannot automatically make significant changes to the blocks in order to handle similar, but nevertheless different, circuit segments: they also have no capability for automatically constructing a diagram from purely electrical information.

Another advantage which the writer's system offers, is that the user need not specify detailed connectivity of circuits, which is obtained from the machine's knowledge of network arrangements, and the user's communication burden is thereby greatly relieved. Such a feature is not provided by other systems, and this report is concerned with the development of such a system.

CHAPTER 4
THE CIRCUIT DIAGRAM COMMUNICATION PROBLEM

4.1 Introduction

Following the discussion of many of the problems (solved and unsolved) in the fields associated with interactive computer graphics, given in the preceding chapters, it is now proposed to discuss the problems confronted in the present work, and the environment which is chosen for the development of the programs. The emphasis here is on the nature of the problems rather than on methods of solution (which are discussed in Chapter 5).

4.2 Man-machine Communication

4.2.1 Motives for Communication

The primary interest here is in communication with a machine. It is useful, therefore, to indicate the reasons for human beings wishing to communicate with machines, and to state the objectives in the formulation of such communication systems.

If human beings are to use computing machines to assist in their various tasks, then the machines must receive information on the problems faced, and must inform the human operators of solutions found.

In normal batch computing systems there is not a great range of methods for presenting problems to the machine, and a form of high level language program is usually used for this purpose. The question of the output of solutions is more complex. If an ultimate decision required by a user is, for example, the type of transistor to use in a particular circuit to perform a given job, then the answer need only be "2N3904". While a computer is capable of finding such solutions and giving the answers, many users are not satisfied unless they receive more detailed information, such as the performance curves obtained by using each of the possible transistors in the particular circuit. The human being wants, in effect, to look into the machine's detailed operations to see how it has arrived at its solution. This form of probing is not common when people are set similar tasks (except perhaps in a psychiatrist's office, or in the case of a new recruit in a design office) and represents a waste of the computer's resources. In an interactive environment the same statements apply, although the waste in resources may not be so severe.

A question related to the efficient use of resources is that of the form in which interactive communication should take place. A common assumption is that communication should resemble inter-human communication closely: a basis for this argument is that human communication has evolved over a long period and can be expected to be fairly efficient. Bobrow (1970) points out, however, that there are many tasks, such as those involving mathematical development, where natural language communication is not the most convenient or efficient. This is due to the fact that newly developed areas of study have not been subject to an extensive evolution in communication and are not generally considered as natural communication. A better statement of the assumption would be to assume inter-human communication as a guideline, but to allow other convenient, efficient methods to be called natural. Natural communication makes it easier for the human operator to perform his task. While it may in some cases be more efficient to allow the machine to dictate the means of communication, the above assumption is generally admitted.

4.2.2 Communication Media and Languages

The question of the language to be used for communication is closely related to the communication media to be used. This, in turn, is connected with the objectives of communication discussed in the previous section. If the guideline of inter-human communication is taken, then the two major media should be spoken natural language and pictures, as these are dominant in man-man communication. The use of spoken natural language implies the existence of useable speech recognition and speech synthesis methods. These do not yet exist*; thus spoken language is, for the present, replaced by typed natural language. At present systems using typed natural language

*except perhaps for specialized applications, operate only on a restricted basis (see section 2.4) and most interactive systems tend to use the fixed command type of communication - these restrict free communication but, as is shown in section 4.2.3, the inconvenience of the restrictions can be eased. With regard to pictorial languages, simple methods exist (sections 2.3.3 and 2.3.4) for the description of pictures, but no adequate communication of pictorial information has yet been achieved.

Thus far the two media (language and pictorial) have been treated separately. In reality they are related in their functions. Information transmitted via one language is relevant to the other domain and vice versa.

4.2.3 Internal Machine Knowledge

The existence of internal knowledge regarding the world of interest is important if effective communication is desired. Human beings make assumptions about the world of interest during communication, and it is consequently possible to transfer rapidly ideas from one person to another because of the reduction. As well as coping with this reduction of information transmitted, the machine must, after receiving a communication, interpret it in the real world environment - such interpretation is not possible without knowledge of the environment (section 2.4).

Many graphics researchers have not made specific use of the fact that incorporated knowledge of the environment can be of assistance in reducing the amount of transmitted information necessary, although in the field of natural language communication, the need for environmental knowledge in interpreting statements has been appreciated (Bobrow, 1964; Woods, 1967). Graphic systems have been able to avoid incorporating such knowledge because the discourse between man and machine has been in the form of commands which have a direct interpretation in either the graphic or application (or both) environments. Relieved of the interpretation burden, graphics designers have not attempted to address the communication problem. This is considered to be an important shortcoming in current graphic systems, which it is well worth alleviating.

4.2.4 Organization of Internal Knowledge

Thus far the necessity for knowledge of the application field has been indicated. Nothing has been said, however, about the way in which such knowledge should be organized within the machine. To point out the significance of this to ease of communication, consider the following example:

Suppose a particular line drawing of a country scene is the subject of discussion between man and machine. This picture forms the environment for the discussion. A possible machine view of this environment is as a set of line segments, each with certain X,Y coordinates for its two endpoints. This contains all the available data on the environment. The human being, on the other hand, can be expected to describe the picture in terms of such objects as houses, trees, fences, animals. As well as identifying the various objects, he will consider the various spatial relationships between them, such as the car is just in front of the house, the cow is tied to the tree. If the human being starts the dialogue with the question "How many windows are on the top floor of the house?" the machine will not be able to extract this information readily from its "environment", but will first have to perform some elaborate processing to determine which lines make up houses, windows, etc., and then it must manipulate this information to obtain the answer. This supposes that the machine knows what is meant by window, top floor, and so on. The machine might want help and might come back with the question "Do the line segments which start with the coordinates (324, 19.8) .... form a window?" The man has now received a question not in his descriptive realm, and he must use a ruler to work out what the machine requires.

This example is perhaps extreme, but it illustrates the result of having a "foreign" description of the environment within the machine. This aspect of descriptions and data bases has not received adequate attention from graphics researchers, although natural language researchers have at times paid heed to the problem (e.g. Woods (1967) attempts to use a data base related to the questions which will be asked of his system).

4.3 A Sub-class of Communication Systems

4.3.1 Comments on Complex Human Though Processes

Human thought processes, for simple problems, are characterized by rapid response. However, as the complexity of the manipulations required for solution increase, the response slows rapidly and the human being makes use of a number of aids to the thinking process. It appears that the human brain can maintain only a few concepts concurrently and, in the face of situations involving many more concepts, the process of manipulating a few ideas, storing them away, manipulating a few more and so on, becomes very lengthy and unreliable. The methods used in circumventing this difficulty usually involve the development of some form of symbolic representation of the problem. The data for the problem can then be recorded in this representation and retrieved and modified as required.

4.3.2 Sketches as an Aid to Thought Processes

There are many forms of the symbolic representation referred to above. Written natural language can be considered as such a representation for thoughts, although this is used more often as a permanent method of recording. The method which is of greatest interest here, however, is the use of pictorial sketches. In general, these sketches are not direct transformations such as those produced by an artist, but rather they are abstract symbolic entities. Examples of this type of sketching are electrical circuit diagrams, program flow charts, logic network diagrams, algebraic expression notation, and a great many others in common use.

The relationship which these bear to the application involved is important to their use. In most cases there is one application configuration for every pictorial configuration, but there is not necessarily only one pictorial arrangement for each application situation (for example each program flow chart represents a particular organization of the program, but the program logic could be represented by many pictorial arrangements). In general there are few explicit "rules" governing the structure of symbolic sketches, but the sketch usually conveys more than these rules alone would indicate. The general pictorial layout of such a sketch conveys information regarding higher level organization in the application domain. No fixed rules govern this layout, but established convention plays an important part.

4.3.3 The Use of Sketches in Inter-human Communication

The assistance provided to human thought processes by sketches indicates that they might be of assistance in communicating complex ideas between human beings. This would be the case only if the communicating human beings placed the same interpretation on the sketches placed before them. That such similar interpretations are indeed formed is the result of the rules and conventions for assembling the sketches mentioned above. The rules and conventions have thus arisen from the desire to use sketches in communication.

When symbolic sketches (diagrams) are used in communication, the overall communication system becomes quite complex. There are now two tightly interrelated communication media. One is the medium of the diagrams themselves, which communicate pictorial and application information, and the second is a discourse language (usually natural, such as English) which serves to transmit information directly in the application domain. It is possible that this medium may also transmit information relevant to the pictorial domain. The two media are often so tightly interwoven that a statement in the discourse language may not make sense unless taken together with some associated pictorial indications.

4.3.4. Diagrams in Man-machine Communication

Since diagrams form such a useful part of human communication, the question naturally arises as to whether a similar system of communication could be of use in man-machine interaction. The question to be considered is to determine the requirements for such a system.

Requirements for this system may be discussed by examining the capabilities exhibited by human beings in this field. One requirement is that the machine must "know the meaning of a diagram", that is, given a particular diagram as input, the machine must be able to interpret this as an application situation. This does not necessarily mean that the machine must have complex recognition procedures for freehand sketches. The input may be in the form of well-defined symbols, lines, and so on. The machine should in general, however, be able to interpret the meaning of the pictorial layout in application terms, just as a human observer does.

A complementary requirement is to be able to generate a diagram given some form of application description. It was stated previously that the transformation from application to diagram is frequently one to many, consequently this task is not necessarily similar to the diagram to application transformation. The choice of the exact diagram to produce is made by human beings on the basis not only of the rules and conventions, but also on certain aesthetic grounds related to the overall neatness and intelligibility of the diagram. Many of these aesthetic requirements have been formed into conventions, but nevertheless they still impose a significant, separate constraint on the choice of diagram construction. The machine must also take these points into account if it is to produce a diagram which is acceptable to the user.

The requirements, as set out so far, suggest that the machine should maintain two separate descriptions of the situation - one is of the problem situation, the other is a pictorial description of the diagram. These two descriptions should be couched, if possible, in the same terms as human beings use.

To date the internal processes required have been considered, but nothing has been said of the actual communication between man and machine. This may take several forms, some or all of which may be relevant to particular applications. Firstly the machine communicates graphically with the user by displaying its current version of the diagram on the display screen. Since this display is representative to the user of the problem situation as held in the machine, it is not generally necessary for the machine to transmit its problem description to the user. Application oriented statements by the machine will normally be concerned with results of analyses performed for the user.

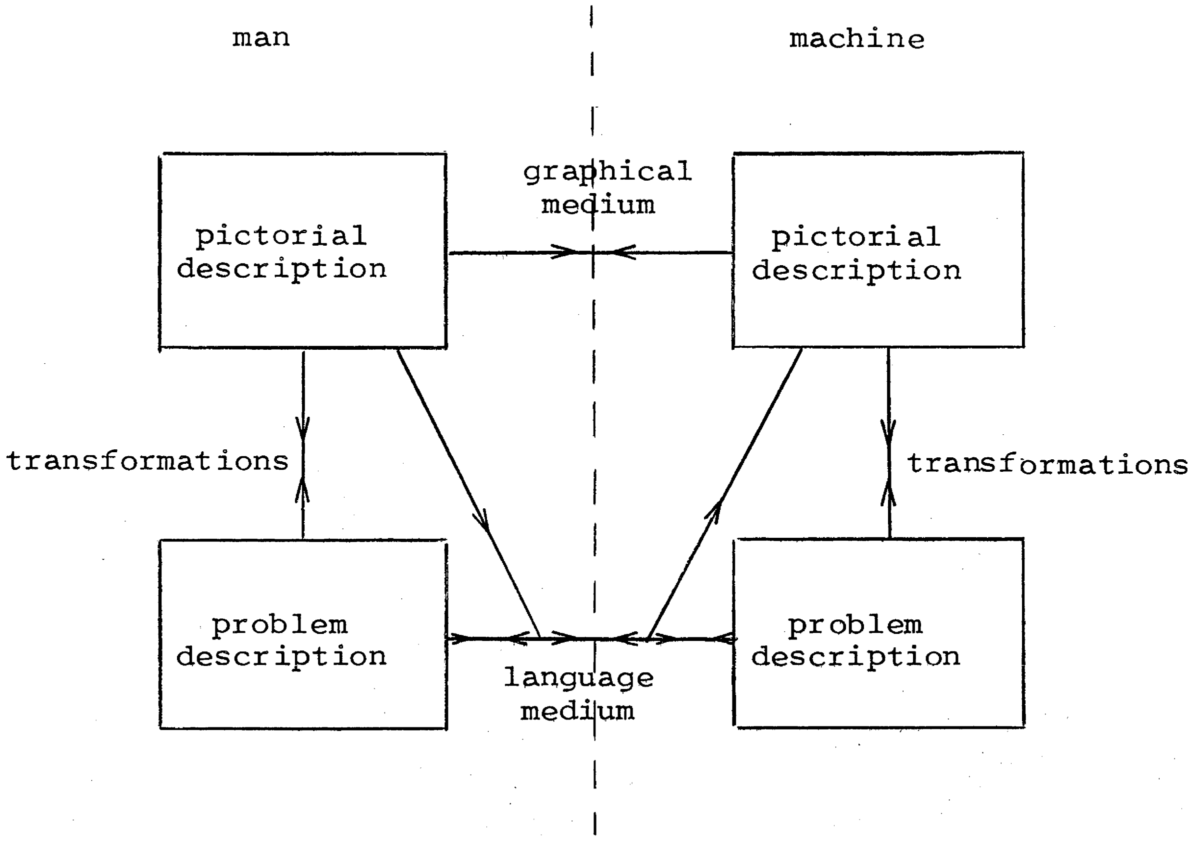
There are more ways in which a user may wish to communicate.

(1) He may communicate graphically by constructing a diagram of interest, and do this via the "pick-and-place" method which, although not ideal, is the only developed practical method.

(2) He may also communicate over the language interface, and this may take several forms depending on the subject: a statement concerning graphical information may be given - this could come about if, because of personal idiosyncrasy, the user was not satisfied with the diagram produced by the machine; a statement in the problem domain could also be made - an example of this is the specification of a problem situation to be studied. This alternative to generating a diagram can shorten the user's task considerably and puts the onus on the machine to produce the diagram. These various forms of communication involve the machine in exercising all of the facilities mentioned previously as requirements. The relationship of these features to one another is indicated approximately in Figure 4.1.

4.4 Circuit Diagram Communication

Questions involved in graphical communication have been discussed from the general point of view; no mention has yet been made as to whether or not different fields have identical problems. This question will now be viewed with regard to a particular application field, namely electrical networks and electrical circuit diagrams.



<u>Figure 4-1</u> An arrangement for man-machine communication involving diagrammatic representations

In attempting to develop systems of the type outlined in the previous sections of this chapter, the question of generality occurs. The developers of DESIGNPAD, for example, consider that it is important to produce a system which is capable of handling many domains without modification (section 3.3). It has been indicated, however, that effective communicating machines should have useful knowledge of the environment in which they are to operate. Since this knowledge, involving that of how to draw appropriate diagrams, varies from one field to the next, there appears to be no simple method of achieving the aim. It seems that separate sets of knowledge for each of the anticipated environments must be included, making the system equivalent to several almost independent systems. One way to overcome the difficulty is to incorporate the ability to learn how to operate in new fields, in which case the designers of the system at least do not have to incorporate knowledge beforehand on all fields in which the machine may possibly operate. Addition of learning cannot be done effectively, however, until some idea has been obtained of exactly what type of knowledge is necessary.

A full circle has now been completed in the generality argument, and the position has been reached once more in which it seems that an individual application should be studied. (Learning abilities are discussed in a later chapter when reviewing the capabilities and possible extensions of the present system.)

Having decided to investigate a single application, a domain must be chosen: for this work electrical networks and their associated circuit diagrams have been selected for study. There are several reasons for making this choice:

(1) The interest currently existing in graphic systems involving circuits - because of the development of computer-aided-design techniques which can be operated most efficiently in an interactive mode.

(2) As a consequence of this interest, a number of systems already have been developed in this field, so that it is possible

to assess their performance and to compare their performance with that of the system developed in this report.

(3) Electrical circuit communication seems to be a field which contains many features of more general applicability.

The fundamental rules regarding the construction of electrical circuit diagrams are well defined. These are discussed in detail in Chapter 6, but for the present purpose, it may be noted that there are certain fixed symbols used to represent electrical elements; electrical connectivity is represented by joining of attachment points of these symbols, either directly or via a number of straight line segments. A diagram constructed according to these rules represents a valid diagram, although it may not be acceptable to a human being. Human operators, trained in handling circuit diagrams, are able to obtain a great deal more information than simply connectivity, provided that the diagram satisfies certain other criteria. In general the layout of the various components can be expected to convey information regarding the function of the electrical circuit. A simple example of this type of situation is shown in Figure 4-2. Drawing (b) conveys the idea of parallel impedance much more rapidly than does the unconventional drawing (a). Figure 4-3 shows another effect of pictorial layout. Diagram (a) may be interpreted as a potential divider of some kind, while (b) may be considered as a pi-filter. The pictorial layout has thus influenced the electrical interpretation given. It should be noted that this is a high level interpretation involving intended function, it does not involve connectivity. Both these examples involve the aesthetic arrangement of elements as well as certain layout conventions.

Circuit diagram communication exhibits not only the characteristics of the general symbolic diagram with regard to complexity, but also the same general methods of communication. The user may wish to construct a representative diagram, he may wish to indicate a particular circuit type in which he is interested, or he may desire to inform the machine of an unsatisfactory aspect in the diagram created. The machine may

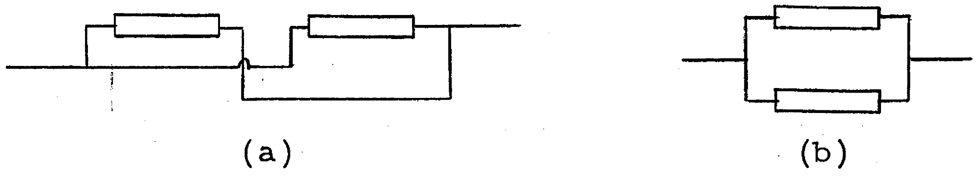


Figure 4-2 A simple example of the role of pictorial layout

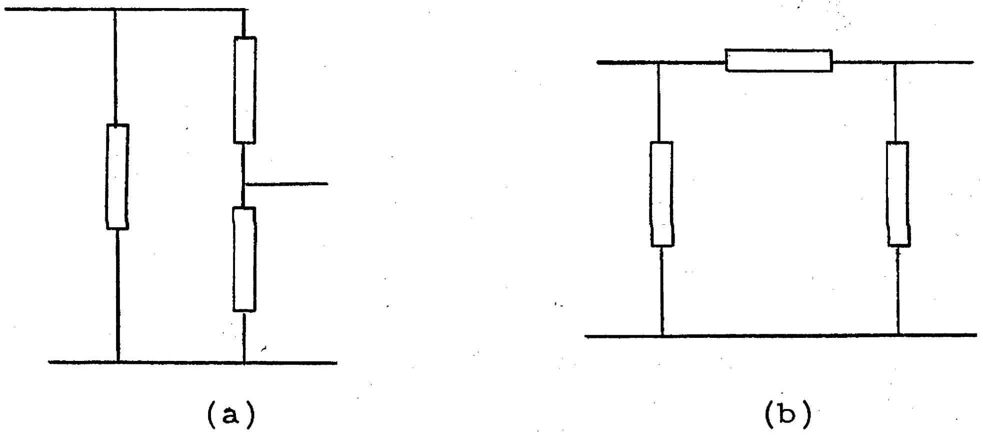


Figure 4-3 A more complex example showing the role of pictorial layout

display a diagram which it has produced and may give answers to electrical analyses performed. In other words, communication of circuit diagrams contains all the elements of the general problem discussed previously, and is a suitable field for investigation of graphical communication systems.

4.5 The Environment of the System

The terms "application field", "domain", and "environment" have previously been used interchangeably. More precise meanings are now attached to each of these terms. "Application field" and "domain" are taken to be synonymous, and to refer to the subject in which communication is to occur. In the present work the domain is electrical circuit networks and their associated diagrams. From now on "environment" is taken to mean the situation which leads to the desire for the man-machine communication (which is somewhat different from the meaning of "domain" and "application field") - there may be many different environments for a given domain. A common environment for circuit communication is, for example, the computer-aided-design environment.

No mention has so far been made of the environment of the present system. This has been done deliberately as the environment does not have a major effect on the nature of the necessary communication; the environment affects only the distribution of the communication amongst the various possible forms. To illustrate this, consider the types of communication required in some possible environments of a circuit diagram system:

In the computer-aided-design environment the principal requirement is for the user to transmit some form of circuit to the machine for analysis. This may be accomplished either graphically via the generation of a diagram, or electrically by informing the machine of the network required. The machine is then required to perform the analysis and to present the (electrical) results to the user. It must also display any diagram, whether generated by the user or produced by the machine from electrical information. A feature of this environment is the absence of questions - all communication is in the form of statements, either instructions to the machine or presentation of results.

Another possible environment for a "circuit communicator" is in the development of an information system supplying answers to various questions on electronic topics. Such a system might take the place of a designer's reference text book and would contain a large data base holding the required information, together with an appropriate system for operating on the data base (c.f. Woods (1967) Airline Guide Q.A. system). In this environment the user makes statements and asks questions in the electrical domain, perhaps also producing a circuit of relevance to the question. The data base system must develop the answer (in electrical network terms, for this is its only realm) and pass this to the communicator. The communicator must generate any diagrams relevant to the answer, and display these at the same time as giving additional information in the answer. The requirements of this environment are thus much the same as before but with a slight shift of emphasis from one type of communication to another.

A further environment, also characterized by questions, is that of a teaching system for circuit theory. The machine may ask questions such as "What type of circuit is this?" The teaching system then presents a circuit organization to the communicator, which generates an appropriate diagram and displays it to the student. Another type of question could be "Draw a circuit of type X". The student would produce a diagram which the communicator, using its knowledge of circuit diagrams, translates into a network description. The "teacher" can then readily compare this with the required solution.

All these systems require the same facilities in the communicator. Consequently it appears reasonable to develop a communicator without including a precise environment. It is advantageous for development, however, to keep an actual environment in mind. This is particularly true if any simplifications are to be made in the system. In the work presented in this report the design environment has been kept in mind, although the system has not been extended to include the complete design system. Rather, the communication is considered as a "front end" for several possible systems (but more particularly for a design system).

Consideration of the design environment allows the introduction of some simplifications which assist in the programming task. If it is assumed that a satisfactory method is devised by which the machine can produce circuit diagrams, then it will only rarely be necessary for a designer to request that the system modify the drawing of a particular circuit. These rare instances can be expected to be due to particular problems or whims of the designer, rather than to gross error by the system. This feature is then an extra, and not an essential part of the system.

When a user wishes to analyze a circuit which is unusual (to the machine) he may input his circuit either electrically (giving connectivities) or by the pick-and-place method graphically. The former method places the onus on the machine to use "common sense" (or aesthetic rules) to work out a layout for this diagram. Since this can be expected to be a more convenient method for the user, it is possible to restrict the system, so that only this "electrical input" is allowed, and not lose too much generality.

The main effect of the above two restrictions, as far as the system is concerned, is that it is then not necessary to develop bi-directional transformations between the circuit description and the pictorial description, but it is sufficient to provide a transformation from network description to pictorial description. This does not affect the "knowledge" required by the machine, for it must still know how diagrams are drawn. While this simplification may appear to be a major restriction, it is not as serious as it may appear: Chapter 10 discusses how the transformation technique developed may be used to assist in the reverse transformation.

4.6 Summary

Problems involved in graphical communication have been discussed above, and a type of communication system has been introduced which is designed to overcome these difficulties. The main novel feature of the system put forward is in the introduction of a form of knowledge relating to the way symbolic diagrams are drawn. This knowledge allows the machine to relieve the user of the task of generating his diagram point by point, because the machine can itself generate this diagram from application knowledge. For the particular application of interest, viz. electrical circuit diagrams, some indications of the complexities of the knowledge necessary have been given.

With regard to generality, it has been indicated that the system is, by nature, environment independent, but is specific to a particular domain. The principles involved, however, are general and applicable to many domains. The effects of simplifications introduced in the actual system have been discussed.

CHAPTER 5
APPROACH TO A CIRCUIT DIAGRAM COMMUNICATION SYSTEM

5.1 Introduction

The discussion in Chapter 4 centred on the features and requirements of the communication system under consideration, but with little attention being given to how the various requirements might be met. This chapter addresses these problems, summarizing the philosophy of approach adopted (section 5.2) and giving a general overview of the system developed, emphasizing important and novel features. Section 5.3 presents a general overview with block diagrams indicating system structure, and section 5.4 introduces some of the problems faced in realizing the system.

5.2 Philosophy

The philosophy taken in the design of the present communication system differs considerably from that used by other researchers. Since this philosophy has an important effect on the approaches used to solve the various problems in the system, the major points of view are summarized here before outlining the approaches to solution.

A general notion considered important is that graphical communication is viewed as an aspect of Artificial Intelligence research. This has led, as has been indicated previously, to the objective of a communication system in which the interaction methods available to the user are as natural and convenient (for the user) as possible.

The desire for convenient and efficient communication has caused an investigation to be initiated into means for replacing the graphical communication methods currently available. The extensive knowledge of subject matter, used by human beings in communication, is believed to be an important factor in the achievement of convenient human interaction, and free man-machine communication, either in graphics or elsewhere, will not be realized until this knowledge has been incorporated in machines. It is considered that the above graphical communication problem may be alleviated by consideration of knowledge of the subject matter - the "relation of representation" concept is relevant here. A complex problem statement carries with it corresponding implications in the pictorial domain. An adequate "communicator" must be able to determine these implications automatically as does a human being.

An important outcome of these considerations is that it appears necessary to maintain two separate descriptions within the machine. These descriptions, which should resemble the corresponding human descriptions, need not be generated simultaneously as is required in current systems. Rather, the "relation of representation" knowledge is invoked to create one from the other at appropriate times.

The incorporation of knowledge as outlined above, is believed to be a useful method of providing more convenient interaction while remaining within the constraints of fairly inflexible command mechanisms. This is so as the machine may now assume a large portion of information which must otherwise be provided by the user each time it is needed.

Within this framework, the following sections outline solutions employed in the present system.

5.3 General Organization

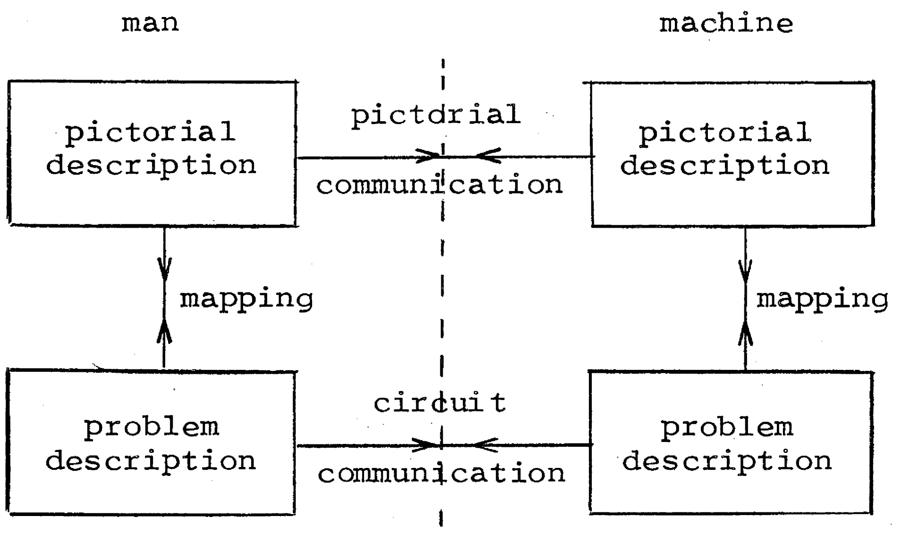
In discussing the requirements of a general communication system, a pattern emerged indicating the various paths by which information may flow between the user and the machine, and also within the machine itself. This pattern was summed up in Figure 4-1,* about which two points can be noted: firstly, the organization shown in the "man" is not intended to indicate a hypothesis regarding the operation of the brain, but rather to represent the apparent functions performed, as seen by the machine or other external observer. Secondly, there is no fundamental reason for the existence of unidirectional paths in this diagram; theoretically, communication could occur in the reverse direction along all paths. This situation is not, however, expected to be common. Such a situation might occur, for example, if the machine was not satisfied with the way that

*P59

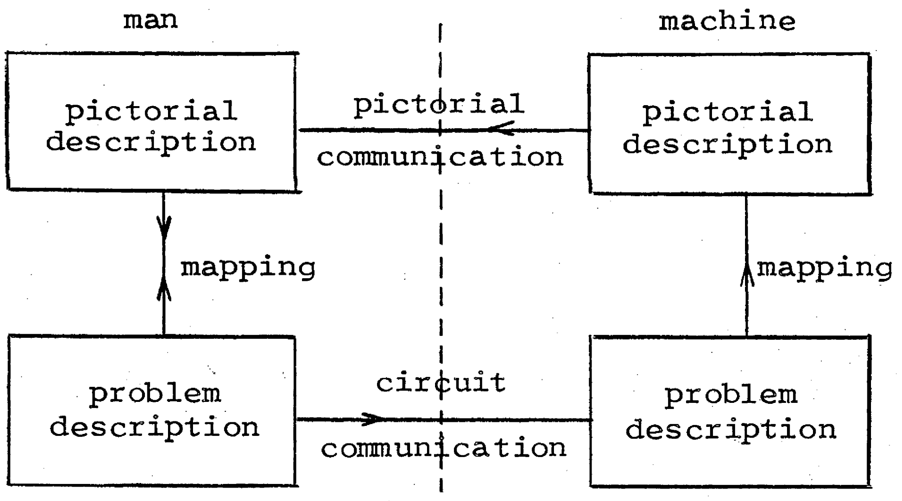
the user had drawn his circuit diagram. While this is feasible, it is not considered useful to add such a path at present.

It is valuable for the development which follows, to regroup the communication media shown in Figure 4-1, in which there are two forms of communication passing through the language interface: firstly, electrical network statements, such as the specification of a network of interest; and secondly, graphical command statements, such as corrections to a diagram. It is useful to group these latter commands along with the graphic interface at the top of the diagram because, although they may go through the language interface, their origin is in the man's pictorial description and their ultimate destination is in the machine's pictorial description. The resulting combination of communication types, while not entirely graphical, is concerned with pictorial information and will be called pictorial communication. Similarly the language interface, while not now representing all the language discourse, is accordingly entirely concerned with the circuit (or application domain) communication and will be called circuit communication. The modified diagram is shown in Figure 5-1(a).

The simplifications in communication discussed in section 4.5 allow further modifications to this diagram. It has been assumed in this investigation that the user will not need to communicate graphically to the machine (that is, not draw circuits himself) because it is capable of constructing the necessary diagrams itself. This makes the pictorial communication channel unidirectional, and it now consists simply of the machine displaying its currently held diagram on the screen for the user. This restriction means that the mapping between pictorial description and circuit description will no longer be necessary, and this may also become one way. Communication in the electrical domain still remains two way when a complete system("communicator" and environment programs) is in use. Since communication between machine and man concerns solutions to problems posed in the environment, and since only the "communicator" is of prime interest in this work, it is possible



(a) a modification of Figure 4-1 grouping the two forms of pictorial communication



(b) simplification of the above diagram as indicated in text

Figure 5-1 Regrouped paths of information flow

to restrict the direction of communication here also. The flow of information, once these simplifications have been made, is shown in Figure 5-1(b). The bi-directional mapping in the user remains as an indication of his thought processes.

Figure 5-1(b) represents the flow of information (i.e. communication paths) within the total system of user and machine, and does not necessarily give any indication of the machine structure, which must now be considered.

A basic machine requirement is a form of monitor which controls the operation of various parts of the system, and through which various commands from the user must pass. These user requests can be divided into two categories:-

(a) Those commands of interest to the monitor only and concerning control of the system;

(b) Circuit data from the user which must be converted into the circuit data structure.

At the other end of the system, it will be necessary at various times to display the diagram represented in the pictorial description. To this end a display package must be included, and this can be expected to be closely related to the particular graphic facilities available in any installation.

The transformations or mappings which will generate a circuit diagram must have access to knowledge of the nature of circuit diagrams. There are two possible ways in which this information can be supplied:

(1) By building the information directly into the procedures. The knowledge is then implicit rather than explicit and it may be difficult to change the information, either for improvements to the system or for a change in environment.

(2) by including the information in some form of passive data base. This method allows modifications to be made more readily. In either case the nature of this information must be determined.

The method which has been used in the present system is a combination of (1) and (2) above. That portion of knowledge which is inbuilt is relatively free from change, even between different domains. It concerns the various aesthetic requirements in the construction of a diagram. Such features as symmetry and proximity can be expected to be the same irrespective of the domain of operation of the system. It has also been found that these features are well suited to incorporation within the mapping procedures (see section 8.4).

The portion of knowledge which is in the form of a data base is connected specifically with circuit diagrams. This data base includes information on the pictorial organizations required for particular circuit organizations.

The various parts of the system which have been introduced so far can be brought together into an overall block diagram (Figure 5-2). The division of information between the procedures and the data base for the mappings has been indicated by the overlap of the two blocks concerned. In order to incorporate this system into some application environment, an extra communication link must be added to the environment programs. This linkage would occur to and from the circuit data structure, again via a form of data converter which is controlled by the monitor. This converter may bear some resemblance to the input converter, although the actual form of data transmission need not be the same.

5.4 Problems to be Solved

The communicator outlined in the preceding section contains features which have not been directly included in previous systems. These features present certain new problems which have been solved in the development of the new system. These problems are now discussed.

The nature of the circuit data structure should be considered. While other systems have included a circuit structure, there are some features required by the present system which have not been included in previous systems. The most important of these is the requirement that the description used by the machine must closely resemble that which might be

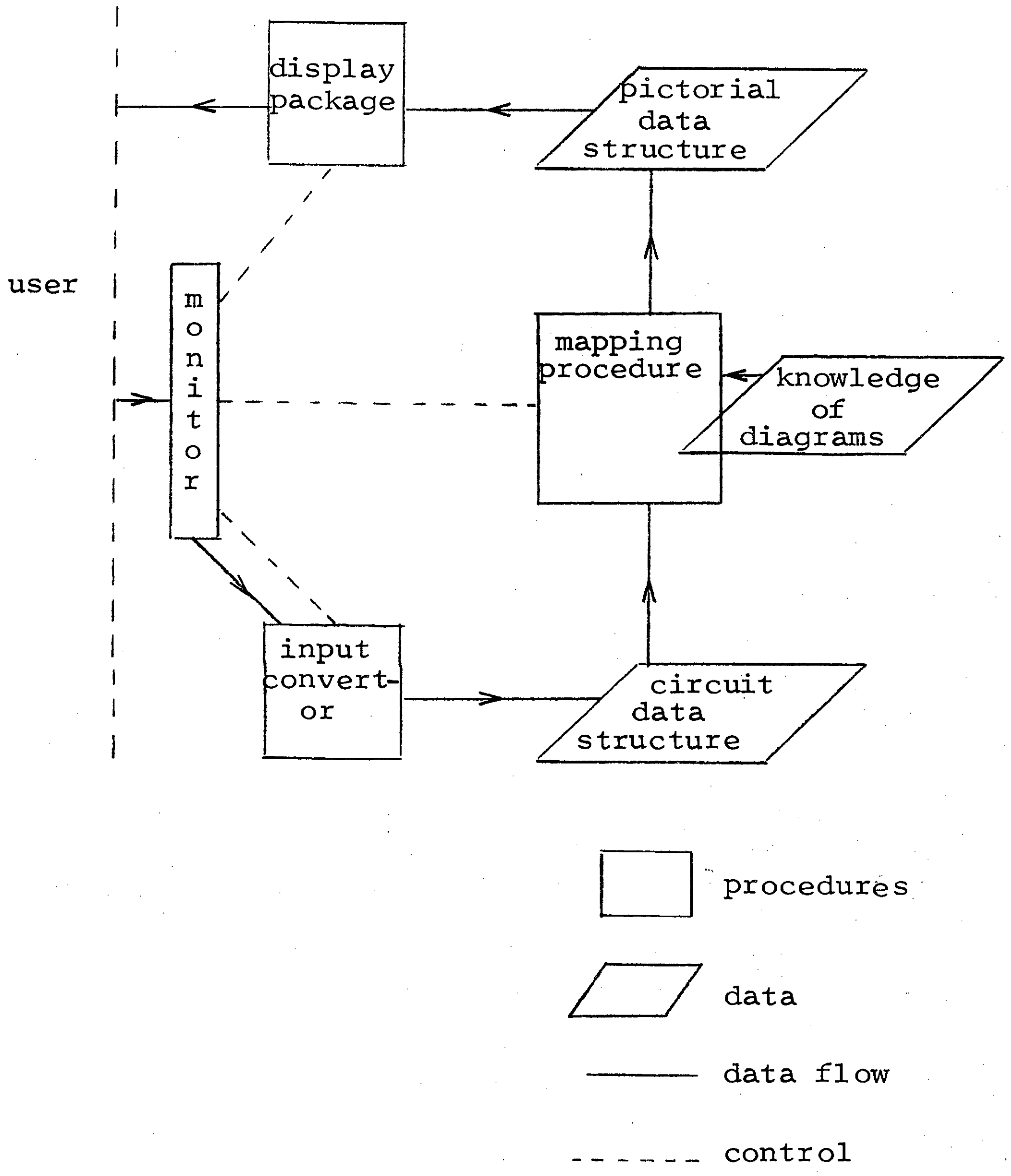


Figure 5-2 A block diagram indicating the internal structure of the communication system

used by a human being for the circuit. Some idea of the higher order structure, the functioning, and the flow of information through the circuit, all appear to be essential parts of the human description. The problem which must be solved is to determine exactly what information is required to form such a description, and how this information may be organized into a satisfactory structure. This is discussed further in Chapter 6.

A further feature of the system which must be investigated is the nature of the pictorial description of the diagram maintained by the machine. Other systems have not attempted to maintain an elaborate pictorial description, rather (as discussed in section 3.4 for example) they have intermingled both pictorial and circuit aspects within one structure (except systems such as DESIGNPAD, which do not have an explicit problem domain). The resulting data structure is biassed towards the circuit structure, and consequently cannot exhibit many purely pictorial relationships. Such relationships as "adjacent" and "collinear" supply useful information in interpreting circuit diagrams. The problem of determining the nature of these useful relationships, and incorporating them into a suitable data structure, is considered in Chapter 7.

The most important novel feature of the present system is the provision of the ability to map between the circuit description and the pictorial description, thus generating a circuit diagram. This aspect of a graphical communication system has not been considered before*, and therefore must be considered from first principles. It has been stated that the mapping process requires a knowledge of the way diagrams are drawn, and that this knowledge is incorporated both implicitly in the procedures, and explicitly through the use of a data base. The knowledge which is inbuilt in the procedures is of an "aesthetic" nature. It may be argued that features such as these hardly amount to knowledge. It is believed, however, that without such features the diagram would be unacceptable to a human user. The aesthetic features can therefore be considered to add to the knowledge of how to draw circuit diagrams correctly.

*Clowes and Stanton have considered the question in general terms but not in detail

The data base part of the knowledge is based specifically on the nature of circuit diagrams. The form of data in this base depends greatly on the methods used in the mapping, and the two must therefore be considered together. Whatever the actual form, however, some effort must be expended in determining how and why various types of circuit are drawn as they are, and what is the nature of the clues which draw attention to the various circuit organizations and functions. When answers to these questions have been found, it is then possible to devise a mapping procedure using this information (as outlined in Chapter 8).

5.5 Summary

A general organization for a communicating system has been developed here. The simplifications to the general system which have been introduced are not believed to constitute major restrictions, and are discussed in Chapter 10, together with achievements and possible extensions to the system.

Three portions of the system which are novel, and constitute the major problems to be solved, have been introduced. The following three chapters discuss these areas in detail, together with the solutions adopted.

CHAPTER 6
CIRCUIT DESCRIPTIONS

6.1 Introduction

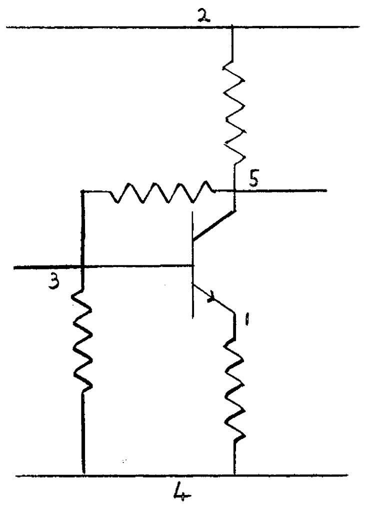
Previous chapters have briefly indicated some of the problems of determining the nature of suitable circuit descriptions: this chapter investigates these more fully. As the major objective is the development of a form of description resembling that formed by a human being, human interpretation of circuits is examined.

The first method of description investigated bears some relationship to those used in other systems. Although this method includes some of the necessary features, it is shown to become clumsy and unnatural, and is therefore abandoned in favour of a new approach, which is developed into a complete descriptive system. The chapter concludes with the presentation of a formalism developed to define this descriptive mechanism.

6.2 Interconnection Matrices

The first method considered starts with a simple form of network description, and attempts to extend this to include the features found necessary. The term "interconnection matrices" has been used to denote this form of description. While this latter may not in fact be represented in the machine as matrices, it may be considered as equivalent to this, and it is in these terms that the descriptive method will be discussed.

The starting point is to consider that the network is made up of a set of nodes, interconnected by various branches. The connectivity can be represented by a matrix in which the row and column number both represent nodes in the circuit. A cross is placed in the i,j th position of the matrix if a connection exists between the i th and j th nodes. In the present discussion the presence or absence of a branch is indicated by a cross, but in an actual implementation, this cross can be expected to be replaced by some information regarding the nature of the branch represented. Figure 6-1 shows a simple circuit and the corresponding matrix, which is symmetric about the



	1	2	3	4	5
1	\		X	X	X
2		\			X
3	X		\	X	X
4	X		X	\	
5	X	X	X		\

<u>Figure 6-1</u> A simple example of the interconnection matrix method of description

diagonal. This symmetry will be destroyed, however, if information concerning branches which are not bi-directional is included. The transistor in Figure 6-1 introduces such branches.

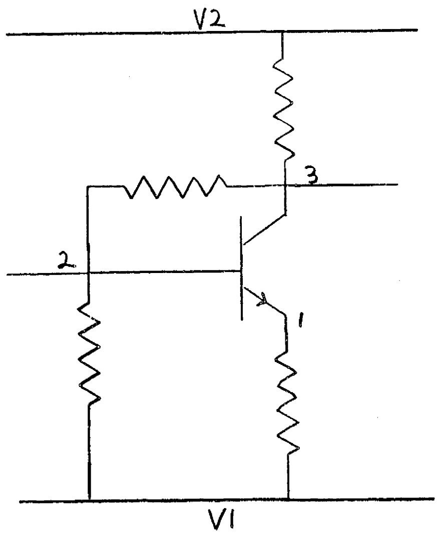
The description in this form contains complete information about the network, if by completeness it is meant that all the information needed to specify the operation of the network is present. This form of description (or a very similar one) is the required input for some circuit analysis programs, and it is because of this fact that the particular form of description has been investigated.

Functional, relational and structural organizations (with which human beings endow circuits) are important aspects lacking in the description as shown in Figure 6-1; this is a major drawback, and some possible additions to the descriptive method have therefore been considered in the next section.

6.3 Node Labelling

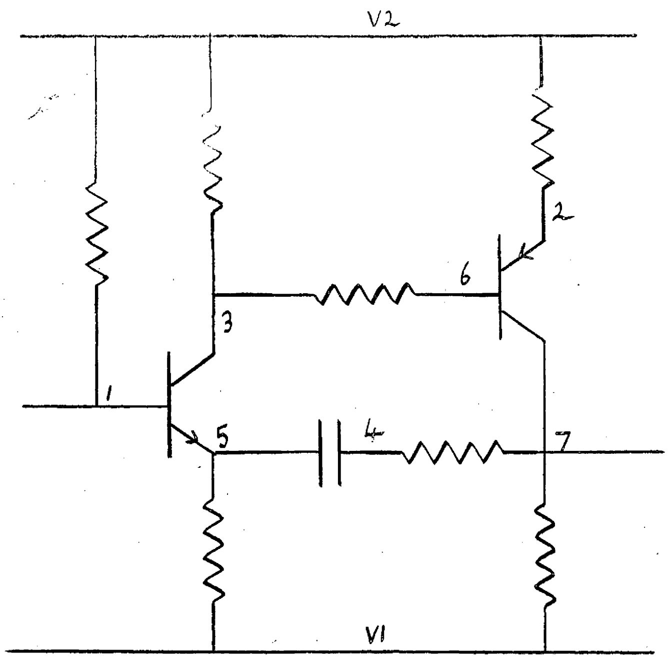
A major factor in the lack of organization of the matrix in Figure 6-1 results from the random labelling of nodes. It may be expected that some organization in the labelling of the nodes might therefore introduce order into the resultant matrix. As a starting point for this, it is generally assumed that "earth lines" and "power supply lines" are a special form of node and they may thus be specially labelled to show this. It is also possible to partition the matrix into regions indicating connections to the supply line, and interconnections of components. Figure 6-2 illustrates this procedure for the previously illustrated circuit.

While these additions introduce some organization into the circuit, this is by no means sufficient. Since all of the internal connections between nodes 1, 2, and 3 have been made in Figure 6-2, consideration of a more elaborate circuit is necessary to further the investigation. In the two stage circuit shown in Figure 6-3, for example, the expected separation of the connection of various nodes to the supply lines is clearly shown, but there is no apparent organization within the internal portion of this



	V1	V2	1	2	3
V1	\		X	X	
V2		\			X
1	X		\	X	X
2	X		X	\	X
3		X	X	X	\

Figure 6-2 The special treatment of supply lines



	V1	V2	1	2	3	4	5	6	7
V1	\						X		X
V2		\	X	X	X				
1		X	\		X		X		
2		X		\				X	X
3		X	X		\		X	X	
4						\	X		X
5	X		X		X	X	\		
6				X	X			\	X
7	X			X		X		X	\

Figure 6-3 A more complex example of the procedure

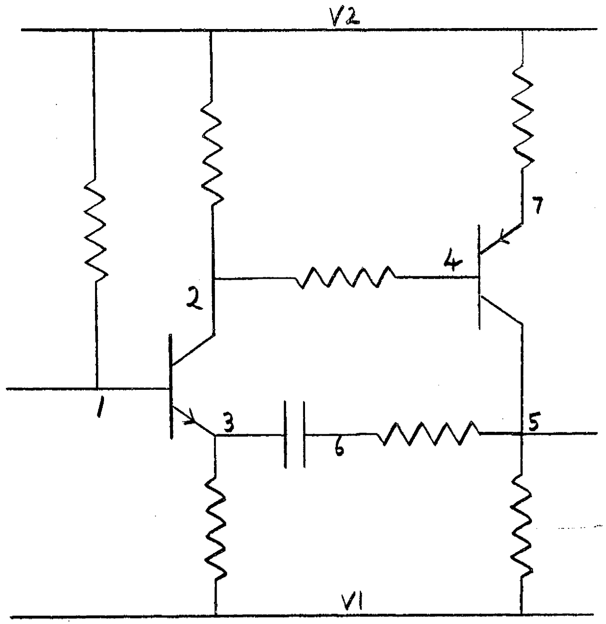
matrix, however. Because this is a two stage circuit, there should be some representation of this in the matrix description.

It is possible to label the nodes of each stage separately, and this procedure is straightforward except for nodes such as node 4 (in Figure 6-3) for which there is no clear cut assignment. For the present, a random assignment to the second stage may be made as in Figure 6-4.

Some interesting features emerge from this new labelling. The power supply separation remains the same as it was in the earlier matrix, but the internal connections section shows significant change. There are two major clusterings of interconnections between nodes. This is to be expected, as the two clusters represent the connections within each stage of the circuit. The two remaining regions of the matrix (only one of which is independent) represent the interstage connections in the circuit. That this region of the matrix is only thinly marked with connections is again to be expected, as a reflection of the way circuits are normally designed. A usual procedure is to design a circuit in functional sections, each of which is sufficiently complex to be not trivial, yet sufficiently simple that it may all be comprehended at once. These sections are then interconnected, and it is to be expected, therefore, that the number of such interconnections will be small compared to the number of internal connections.

To this point then, it is possible to construct a matrix representation which reflects the structuring of a circuit into a number of stages, and also indicates the portions which represent interstage connections. The representation does not, however, give any indication of the flow of information from stage to stage, nor does it give any indication of the nature of the function of each individual stage within the circuit.

It is possible to obtain some indications on the flow of information between stages by specially marking the nodes which represent the input and output of the circuit. If the circuit contains a number of cascaded stages, it will be possible to use



	V1	V2	1	2	3	4	5	6	7
V1	\				X		X		
V2		\	X	X					X
1		X	\	X	X				
2		X	X	\	X	X			
3	X		X	X	\			X	
4				X		\	X		X
5	X					X	\	X	X
6					X		X	\	
7		X				X	X		\

Figure 6-4 Node labelling according to stages

this terminal information, together with the interconnection information as shown in Figure 6-4, to obtain some notion about the order of the various stages. A difficulty arises, however, when appreciable amounts of feedback occur, as there is no indication in the matrix representation, of the direction of signal flow through any interconnection branch. The existence of information regarding the components making up the branch is not relevant, as the components will probably be normal bidirectional passive elements. In Figure 6-4 there are two paths between the first and second stages: since this figure represents a simple circuit, it is possible to conclude that there is a forward path between the first stage and the second (assuming the input is known) and that the other path could be either forward or feedback. This conclusion, however, has been logically deduced and is not explicitly available in the representation. It is possible to increase the ease with which the information is extracted by means of the following heuristic procedure:

Most stages will have a "dominant" component, probably a transistor, and a more explicit labelling procedure may be used within a stage (A) by labelling the emitter of the transistor A1, the base A2, the collector A3, and the remaining nodes systematically outwards from this dominant component. The label of the node then gives some indication of the node's position in the stage. The heuristic then suggests that within a stage one can usually expect a collector to be associated with output, a base to be associated with input, and an emitter may handle either input or output. This procedure, as has been indicated, helps in the decision concerning interconnections of stages but does not add explicit information to the representation. (The heuristic procedure solves the problem of Figure 6-4).

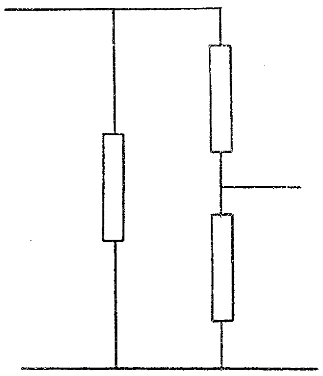
A more serious problem occurs when trying to represent the circuit function within the matrix. At first it appears that, with the fixed node labelling technique, it might be possible to distinguish between various functions by considering the nature of that portion of the matrix representing the stage. This procedure breaks down completely when faced with circuits such as those shown in Figure 6-5, which are simply pictorial rearrangements of the same circuit diagram. They would normally be taken, however, to represent different circuit functions. Since they are equivalent in terms of network interconnections, there is no way that they could be distinguished in the matrix representation, without attaching information external to the matrix itself.

From this and other points discussed, it appears extremely difficult to form a suitable circuit representation based on the matrix interconnection approach. The representation of the required information is clumsy and "unnatural" compared with the manner in which human beings appear to view circuits. It is for this reason, therefore, that this approach to description of circuits was abandoned in favour of a more suitable descriptive mechanism.

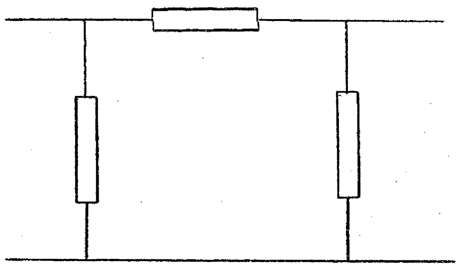
6.4 The Subdivision of Circuits

The arguments in the previous section indicate that it is worthwhile examining the viewpoint of a circuit taken by a human being more closely, and from this, attempting to establish a circuit representation. The two features which caused most difficulty with the matrix representations have been the representation of function within a segmented circuit, and the representation of the way in which the various parts in this segmentation are related.

Consider the manner in which circuits are segmented by reference to the reasonably complex arrangement shown in Figure 6-6. The complete circuit is first segmented into a number of large subdivisions, such that the number and organization of these can be readily understood. Once this is done, attention is focussed on each of the subdivisions in turn. Depending on their complexity, each of these may itself be split into a number of blocks. This process is continued until finally the sections of circuit may be comprehended in detail. Figure 6-6 indicates this procedure for a crystal clock configuration in which two levels of subdivision have been used (leaving three levels of blocks). There is no essential reason

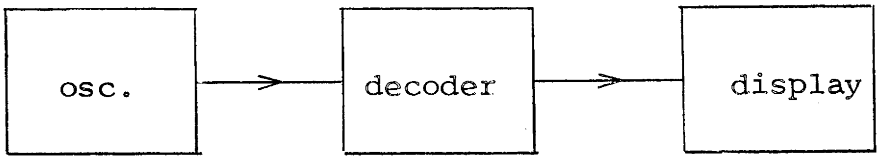


(a) a potential divider circuit

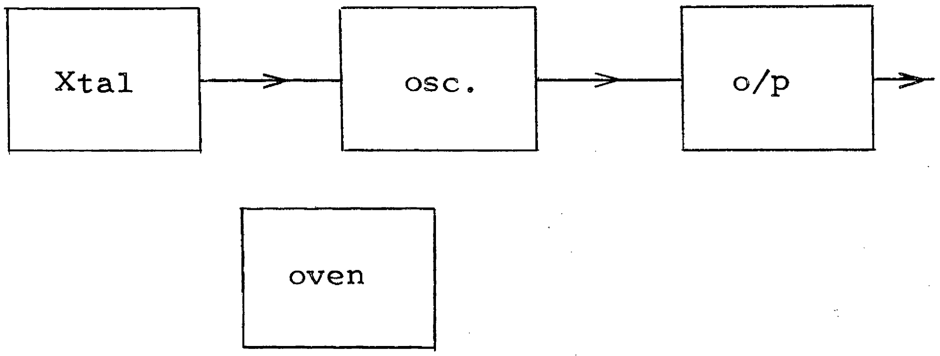


(b) a pi-filter

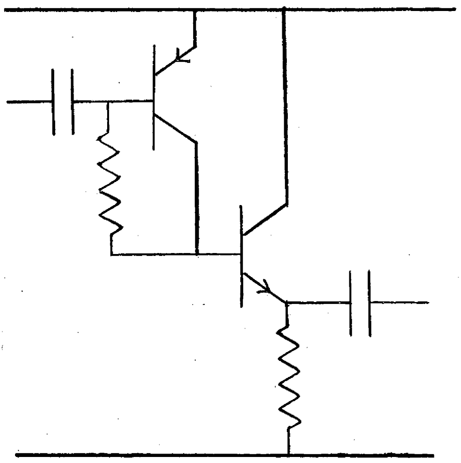
Figure 6-5 Topologically equivalent circuits which have apparently different functions



(a) a crystal clock block diagram



(b) a more detailed view of the oscillator section of (a)



(c) the o/p amplifier in (b)

Figure 6-6 Illustrating the human subdivision of circuits

to restrict the number of levels to three; this depends entirely on the complexity of the circuit. In Figure 6-6(c) for example, it is possible to subdivide again into two separate stages within the amplifier. If the emitter resistor of the n-p-n transistor had been paralleled by a capacitor, it would be possible to subdivide yet again and to consider this parallel impedance as a separate entity.

The position of each subdivision of a circuit is such that each subdivision represents a functional unit. This is so in the example shown, and the labels given to each of the subdivisions reflect this function. This association of a function and functional label seems to be an important aspect of human view of circuits.

6.5 Relational Information

The information present in a circuit consists of more than a set of objects grouped into appropriate subdivisions. The objects are organized in various ways and the organization is fundamentally a function of connectivity, which can be considered on two levels. At the first level this connectivity is the actual electrical connectivity between components via some node. This viewpoint is the reverse of that of the matrix description, in that components are considered the primary objects, with nodes being the means of interconnecting them, and as such appears to agree with the human view of circuits which seems to consider components as the primary entity.

The higher level form of connectivity concerns the flow of signal information through the circuit and is related to the stage interconnections mentioned in discussing the matrix representation. An important feature of this connectivity is that it is, in contrast to the lower form, directional, and any representation must include this directionality explicity, so helping to overcome the interconnection problem as it exists for matrix representations.

The signal interconnection relation can, if desired, be subdivided into two types. The first indicates the normal forward flow of signal, and the second represents any feedback paths present. This explicit division allows the organization of a circuit to be seen according to explicit information in a description.

6.6 The Descriptive Scheme

The necessary features of the description, as outlined above, seem to be well suited to incorporation in some form of hierarchical tree structure of objects, having associated attributes, and with relations between the objects. This form of description can be readily implemented in any of a number of list processing languages which is available; the actual implementation is discussed in Chapter 9.

In a descriptive scheme of this kind, it is conceptually possible to continue subdividing the structure to any desired level. In practice a certain number of "primitive" objects are chosen as the endpoints of the division process, and the description is formed in terms of these. In the present scheme the primitive elements chosen are the various circuit components of interest, together with nodes, which are given a separate existence. Three forms of nodes are recognized - it is not absolutely necessary to do so, but it is done to tie in with the accepted view that power supply nodes and earth nodes are of a special form. Since the objective is to reflect the human view of the circuit, it is acceptable to make this distinction between nodes. The components chosen for representation in the scheme do not comprise an exhaustive list, but represent a useable subset suitable for illustrative development of the system. The primitives may each have associated attributes which represent their points of connection with the rest of the circuit. A resistor, therefore, has two such attributes. Nodes have no attributes. The primitive objects, names and attributes used in the system are:

N	ordinary node
E	earth node
V	power supply node
R(1,2)	resistor with terminals (1,2)
C(1,2)	capacitor with terminals (1,2)
L(1,2)	inductor with terminals (1,2)
D(1,2)	diode with terminals (1,2) being (cathode, anode)
T(1,2,3)	transistor with terminals (1,2,3) being (emitter, base, collector)
TR(1,2,3,4)	transformer with terminals (1,2,3,4) being in pairs (1,2) and (3,4)

Generally the order of the terminals is important but for bilateral components, this is not so.

The development of higher level objects depends on the existence of certain relationships between the parts. Consequently it is necessary to incorporate certain primitive relations from which to build higher level objects and higher level relations. It was mentioned earlier that there are two forms of connectivity to be found in a circuit. Two fundamental relations may be defined to correspond to these, and are called "connect" and "stage connect". The "connect" relation between a component and a node indicates the obvious simple connection. The "stage connect" relation, while considered fundamental to the system, is defined on higher level objects, and implies that the output of one stage is connected to the input of another. It is possible to define this relation in terms of more primitive elements but, since it is considered to represent a separate concept in the human view, it is preferable to keep it as a fundamental relation.

Having established the fundamental objects and relations, it is now possible to take any interconnection of components and define it as a high level object. In practice, however, an objective is to restrict the structure to high level objects which are usually regarded as functional units, and consequently the definition process is constrained accordingly. The present descriptive mechanism contains a restricted set of such objects which are sufficient for the development of a workable system.

A useful high level relation is the "component connect" relation, developed out of two "connect" relations, and indicating the connection of two components via a particular node. Typical of the possible definitions of high level objects are the "parallel impedor[1]", formed by two appropriately connected two terminal components; the "series impedor", also formed by a connection of impedors; the "divider" consisting of an appropriate arrangement of two impedors with an input and an output; the "pi-filter"; the "T-filter"; the "common emitter stage" and so on.

In the current implementation of the system (developed for illustrative purposes), the only objects containing transistors are common emitter amplifying stages. A "stage" is defined as any of a number of objects with well-defined inputs and outputs. The relation, "stage connect", is predicated on stages, and the object, "cascade", is constructed from stages satisfying the "stage connect" relation. The precise definitions of these and other objects and relations is given below with a formal notation for writing these definitions.

6.7 A notation for the Descriptive Mechanism

The final paragraphs of the preceding section indicate that it is difficult to write down the definitions of objects and relations precisely. Consequently it is worthwhile, for purposes of documentation, and as a possible method to assist in the input of models, to develop a notation which is capable of defining precisely the various objects and relations involved in the mechanism.

In defining a high level object, the following features are required:

(1) an indication of the name of the new object,

[1] The term impedor is used in analogy with the words resistor and resistance. In this description it is the components and not their corresponding mathematical functions which are of interest, and so the term impedor is preferred to impedance.

(2) an indication of the attributes of the new object,

(3) a list of the sub-objects from which the new object is formed,

(4) the relations which must be satisfied by the sub-objects,

(5) any definitions of attributes which are appropriate to the new object.

Clowes (1970) has used a notation which incorporates most of these features, but does not normally include definitions of attributes. As attributes are important for the descriptions used here and must be included, a modification of this notation has been used. For the definition of an object the format is

N(attributes) sub-objects (relations;attribute definitions)

where N is the name of the new object, (attributes) is a list of the names used in the definition for the attributes of the new object, [sub-objects] lists the objects from which the new object is constructed, "relations" specifies the relations which must be satisfied by the sub-objects (here S1, S2, ... Sn are used to refer to the 1st, 2nd ... subobjects in the list given), and "attribute definitions" defines the attributes of the new object. An example of this is the following definition of a parallel impedor:

ZP(A1,A2)[Z,Z,N,N](<u>para</u>(S1,S2,S3,S4); A1←S3,A2←S4)

In this definition the four sub-objects must be of the type indicated (i.e. 2 impedors and 2 nodes), and they must satisfy the relation "<u>para</u>", which is defined later. The attribute definitions follow. The backward arrow means that the item on the L.H.S. is defined as the item on the R.H.S. Thus attribute 1 is the sub-object 3 (i.e. is a node).

A similar method of definition is used for relations. All relation names are underlined to distinguish them from objects. The primitive "connect" relation is written:

<u>con</u> (S1,S2)

where S1 must be an appropriate component, and S2 a node. The format for the definition of relations is:

<u>rel</u> (OBJECTS) [defn. of relation]

Here "rel" is the name of the new relation, "OBJECTS" is a list of objects on which the relation is predicated.

Using this notation it is now possible to list in a reasonably rigid form, a number of the relations and objects used in the descriptive scheme.

ccon (Z,Z,N) [con(S1,S3),con(S2,S3)]

para (Z,Z,N,N) [ccon(S1,S2,S3),ccon(S1,S2,S4)]

series (Z,Z,N) [$\forall$(M$\neq$S3)ccon(S1,S2,S3) and not ccon(S1,S2,M) and $\nexists$ Z such that con(Z,S3)]

The predicate in the latter may be interpreted as "For every node M, other than the node S3, a ccon exists between S1, S2 and S3, but a ccon does not exist between S1, S2 and M. Also there is no component Z such that Z and S3 are connected". This prevents connection of S1 and S2 in parallel, and prevents a third component being joined to the centre node.

ZS(A1,A2)[Z,Z,N](series(S1,S2,S3); A1$\leftarrow$other term(S1), A2 $\leftarrow$other term(S2))

Here "other term" is defined as a terminal not previously involved in a relation.

Z   any of R, L, C, ZS, ZP

This allows recursive definition of two terminal impedors.

DIV(in,out)[Z,Z,E,N,N](ccon(S1,S2,S4),con(S2,S3),con(S1,S5); in $\leftarrow$ (S5,S3),out $\leftarrow$ (S4,S3))

PIFIL(in,out)[Z,Z,Z,E,N,N,N](ccon(S1,S2,S5),ccon(S2,S3,S6), con(S1,S4),con(S3,S4);in$\leftarrow$(S5,S4),out $\leftarrow$(S6,S4))

CEST(in,out)[T,Z,ZM,Z,E,V,N,N,N](S3(1,2,3)[Z,Z,N,N,N] (con(S2,S5),A1$\leftarrow$S3,A2$\leftarrow$S4,A3$\leftarrow$S5), ccon(S1(A3),S4,S9),ccon(S1(A1),S2,S7), ccon(S3,S1(A2),S8),con(S4,S6),con(S3,S6), con(S3,S5),con(S2,S5); in$\leftarrow$(S8,S5),out$\leftarrow$(S9,S5))

The "CEST" definition is two level, since it contains a definition of ZM. The term S1(A1) means the 1st terminal of object 1.

These definitions give an indication of the capabilities of this notation, and from them it will be clear that the use of this notation for the specification of individual circuits can become unwieldy. In the specification of individual circuits, it is not necessary to include all the information found in the above definitions, however. The statement ZP(R,C), for example, is sufficient to define a parallel impedor, and this is the approach which will be used in the remainder of this report: only that information which cannot be obtained from definitions is incorporated. This may include modified information or information unspecified in the model, as in the above example.

CHAPTER 7
PICTORIAL DESCRIPTIONS

7.1 Introduction

This chapter discusses the features considered necessary in a pictorial description of a circuit diagram. It is worth pointing out that a distinction exists between the pictorial description of a diagram, and the diagram itself. The diagram consists of a set of straight or curved line segments (with their appropriate coordinates) assembled in their correct positions. The pictorial description, on the other hand, must contain an articulation of the various significant groupings of parts within the picture, together with any important relations which exist within the picture, and other items, abstract or real, which carry information useful to the viewer. Knowledge of these features will be of assistance in formulating the mapping methods in the next chapter, as well as in designing a pictorial description scheme.

The nature of the exact form of pictorial description is closely related to the methods used for the mapping. It is difficult, therefore, to separate the discussions completely, but for convenience of presentation, this has been done. Those features of the circuit descriptions which depend heavily on the method of mapping are here stated as unsupported facts, and elaborated later (Chapter 8).

This chapter introduces the low level terms in which the description is formed, then digresses into the nature of circuit diagrams, commencing with a discussion of the "rules" for the construction of circuit diagrams, and then examining the various influences which affect the interpretation of circuit diagrams. Having established a sufficient set of these influences, methods for their inclusion into a pictorial description are outlined.

It should be emphasized that well formed line segments are available from the start, and recourse does not need to be made to the determination of lines in terms of gray scale values of points, as needs to be done in many picture processing problems.

It is assumed that methods are available for the direct generation of lines of precisely known shape, size and position.

7.2 The Primitives

Having established the foundation for the description, it is possible to introduce the required primitive objects. These must be drawn from various line segments, but the information required to draw the primitives is invariant and need be known only by the drawing package (see Figure 5-2). The parts of a circuit diagram which remain fixed are the symbols representing the various circuit components. The straight line segment is also a separate primitive part of the diagram, for although its length may change, this should properly be considered an attribute and not, therefore, indicative of a change in shape or structure. The straight line segment does not have a directly corresponding circuit element but, although this has no effect on its role as a pictorial primitive, it greatly affects the mapping procedures (as indicated in Chapter 8).

The various primitive symbols used in the system as it is now implemented, are shown in Figure 7-1. The inductor symbol has been selected in this form (rather than the more usual circular arcs form), for convenience in the drawing package. Since these are the primitive elements, any shape may be used for a given symbol without affecting the descriptions.

The primitives may have attributes; in this system they have several, involving orientation, size and position. These attributes are discussed in detail throughout this chapter, along with those of other pictorial objects, and also in Chapter 8.

As the origin of the shape of the symbols used bears a relationship to the discussion of high level features, some comment is pertinent here. The origin of the symbols is largely historical - when components such as resistors and capacitors were first represented symbolically on paper, the diagrams were symbolic pictures of the constructed circuit, i.e. a resistor was commonly made of a zig-zag length of resistance wire, and a capacitor was formed from two large plates in close proximity.

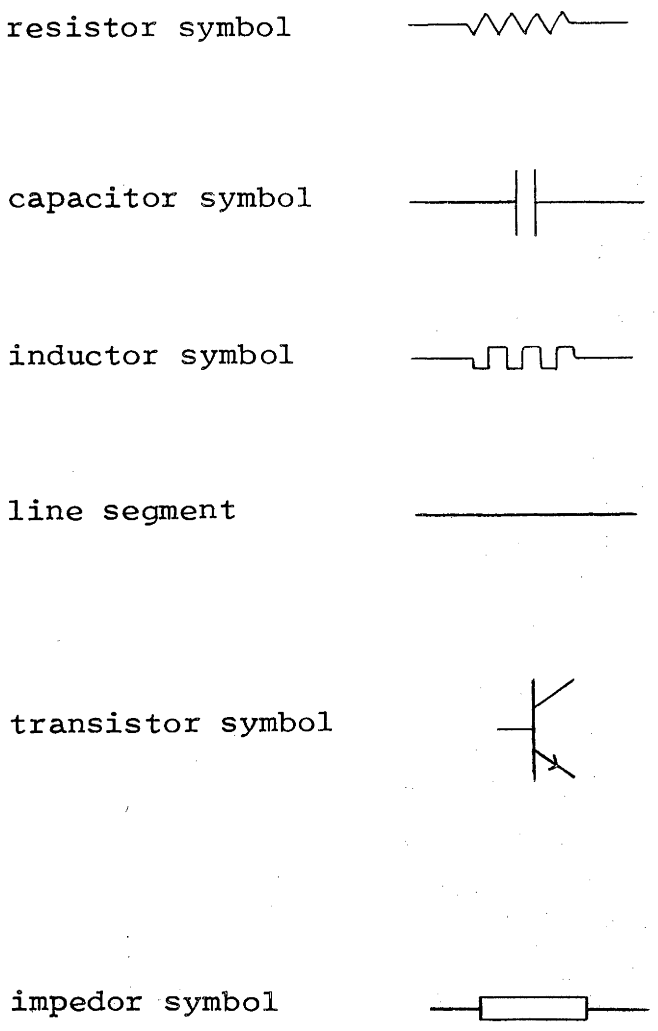


Figure 7-1 The primitive pictorial objects

The symbols can therefore be considered representative of the function their corresponding circuit elements perform, even though they now do not resemble the original in detail.

The same principle extends to various high level features of circuits. In early examples, the layout of a circuit diagram reflected the layout of the actual circuit. With modern printed circuit and microcircuit techniques, this is no longer true, but the historical aspects of the various conventions should be borne in mind.

7.3 "Correct" Circuit Diagrams

Having established the fundamental objects (the primitives) on which the pictorial description is to be based, it is now necessary to consider the nature of a "well constructed" circuit diagram, for this determines the information which must be included in the pictorial descriptions.

The fundamental rules for the construction of a circuit diagram are very simple, and may be stated as follows:

For each component in the circuit, there must be a corresponding symbol (taken from Figure 7-1) in the diagram. Those components which are connected via a node must either have the appropriate attachment points of their symbols coincident or joined by an unbroken path of straight line segments. For convenience no line or symbol may overlap another symbol, although a line may overlap a line.

The above rules are sufficient for a circuit diagram to be theoretically <u>interpretable</u>. Most graphic systems involved in the field contain at most ~~only~~ such "knowledge", and rely on the human user to arrange for any further organization. Figure 7-2 indicates that these rules are not sufficient to produce a satisfactory diagram: most observers could not immediately state that Figure 7-2 represents a form of common emitter amplifying stage - such recognition usually requires a careful tracing of the various paths to reach a decision. If, however, the circuit is redrawn in the form shown in Figure 7-3, most observers can immediately grasp the probable function of the circuit.

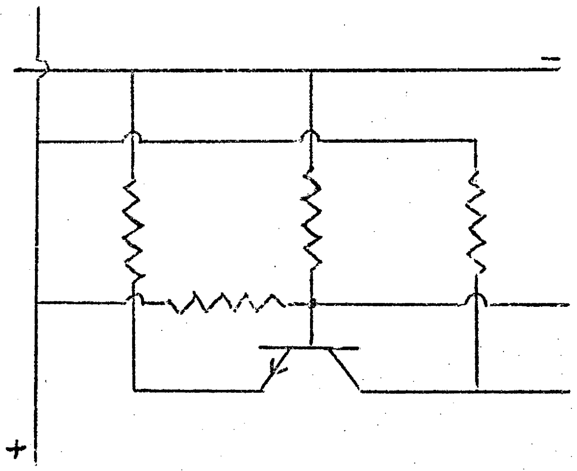
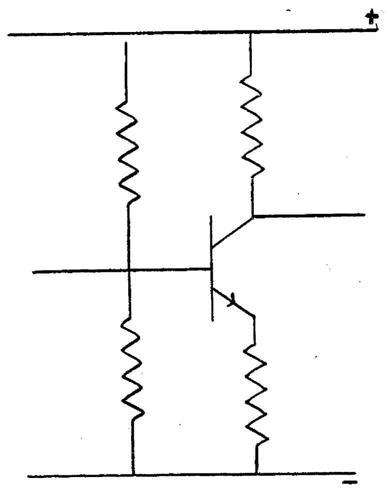


Figure 7-2 A diagram resulting from the application of the basic rules



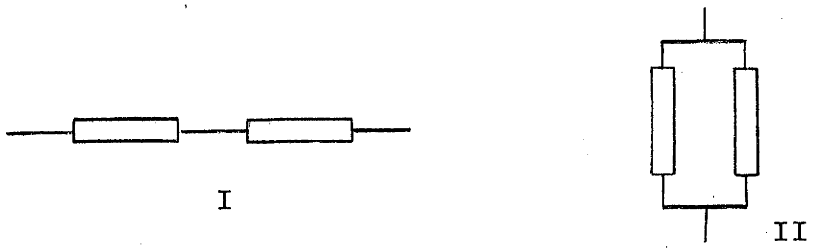
<u>Figure 7-3</u> A redrawn version of Figure 7-2 according to normal conventions

While this is only one example, the argument can be seen to apply to a large number of commonly used circuit types, for each of which there is an overall impression gained from the picture which immediately conveys information regarding the possible circuit function. While this effect is due to the existence of certain conventions, there are a number of influences underlying the conventions; accordingly the influences are discussed prior to the examination of the conventions themselves.

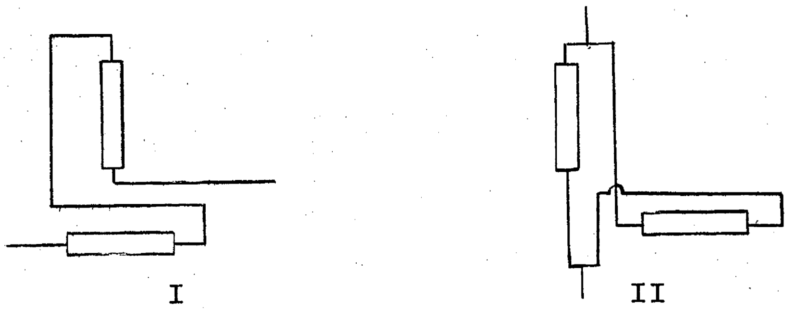
7.4 The Effect of Psychological Laws

Certain properties of human visual systems, which are represented by Psychological laws of visual perception, have an important influence on the organization of symbolic diagrams of all types. These laws indicate a number of global properties of a scene which are relevant to human descriptions of that scene. It is sufficient here to know that the laws suggest that certain properties, namely similarity, proximity, continuation, simplicity and symmetry, are recognized immediately in a scene, and that these properties assist in the formation of associations between the objects concerned.

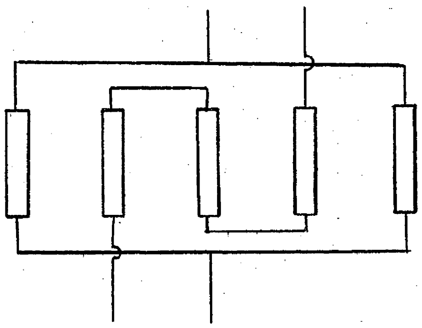
In drawing satisfactory diagrams, the above pictorial associations are used to assist in the association of certain electrical components with one another. The examples in Figure 7-4 help clarify this situation. In Figure 7-4(a)I, the notion of continuity exists between the two impedor symbols; this is used to form an appropriate electrical association. In Figure 7-4(a)II, the symbols are associated on the basis of adjacency, and the existence of a parallel impedor is implied from this. Usually it appears that the association is formed, the implication in the circuit domain made, and only then is the connectivity checked out as a verification of the association. In Figure 7-4(b), the associations no longer exist, and the electrical function must be determined by tracing out the connectivities. In Figure 7-4(c), a parallel impedor is again drawn, but the notion of proximity is destroyed and the association is not made as readily as previously.



(a) two simple examples of associations formed on the basis of Gestalt principles.



(b) the previous examples redrawn so that the associations are not formed.



(c) drawing (a)II, but with the proximity notion destroyed

Figure 7-4 The formation of Gestalt associations

Figure 7-5 shows a circuit in which an important association is made on the basis of symmetry. Because the symmetry here is at a higher level, the implication formed from the association is less direct and indicates simply that the two parts are related. The actual function is determined by a closer examination of the two parts. (In this particular case of a multivibrator, the overall pattern of the diagram may also be recognized as a single functional unit).

Proximity is an important feature in making associations. The symmetry relationship in Figure 7-5, for example, would not be formed as strongly if the two parts were separated and had other pictorial objects interposed between them.

Proximity is also useful in implying the order of the various functional operations in a circuit. Two adjacent pictorial objects, for example, are assumed to follow each other in the performance of their function*.

Simplicity is in a different class to the above principles, and associations are rarely, if ever, formed on this basis alone. In general, however, diagrams are more acceptable and more readily interpreted if they are drawn as simply as possible.

The implication of circuit relationships from these pictorial associations is based on established convention; this convention is usually historical in origin as indicated above. It remains true, however, that pictorial association principles play an important role in the interpretation of diagrams.

7.5 The Influence of Function

It has been indicated previously that, under a number of circumstances, the intended function of the circuit has an influence on the way in which the diagram is drawn. This influence occurs at a number of different levels of complexity.

The lowest level of this influence has already been indicated in the historical connection between the shape of the primitive symbols used, and the original form of the actual components. This historical connection also has relevance to

*Continuation along a line is also important here.

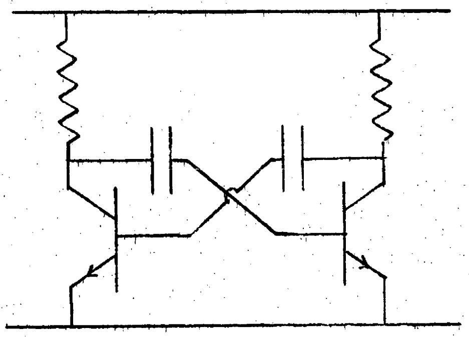


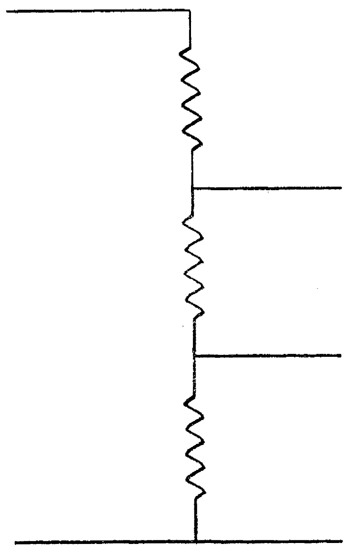
Figure 7-5 A diagram in which a symmetry association is made

the layout of circuits. It was common previously, for example, to arrange an earth line and a power supply line, and to construct the circuit between the two. This type of construction is now to be found only in "bread-board" experimentation, where it is important to maintain the easy association between components so that experimentation may proceed rapidly.

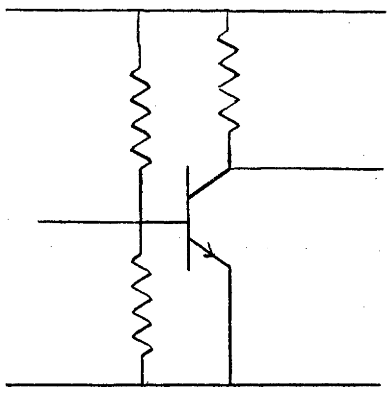
At a different level, the "flow of signals" between the various stages is implied from certain pictorial clues. As indicated previously, this information is closely linked with the notion of proximity, as it is normally assumed that adjacent stages perform successive functions. The more specific adjacency relationship of "left-right" is also used to imply the order of these functions. That is, the flow of information processing in a circuit proceeds from left to right in the circuit diagram. A circuit diagram in which this arrangement is not satisfied is difficult to interpret. Simplicity is also important here, as it is usually not reasonable to arrange the stages randomly and still connect them correctly.

In certain passive circuits, such as the divider shown in Figure 7-6(a), the notion of "signal" is involved in another form of pictorial association. In this case the "above-below" relationship is used to indicate a relationship between the "magnitudes" of the signal at various points. This form of association is used in a number of passive circuit types, but is not completely applicable for active circuits where, for example, there may be a conflict between the pictorial arrangements for biassing and signal (as the power supply line and the earth line are at the same signal potential).

The "above-below" relationship is also involved in interpreting the d-c conditions throughout a circuit. This is an extension of the principle used above, applied to a different parameter (i.e. d-c volts), and is illustrated in Figure 7-6(b), which also shows the discrepancy between the requirements for signal potential and d-c potential.



(a) as applied to signal division



(b) as applied to d-c conditions

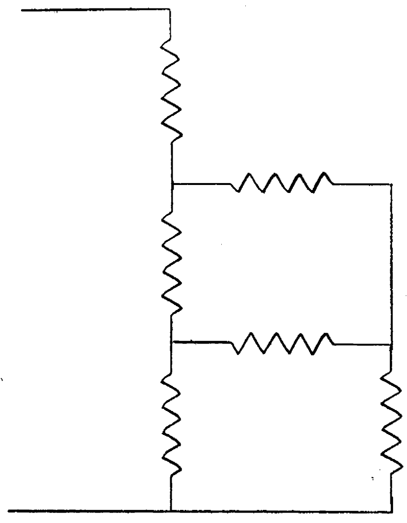
Figure 7-6 The application of the "above-below" relationship to circuit diagrams

The representation of feedback in a circuit is related to the representation of signal flow. Since signal flow is usually from left to right, it can be expected that a feedback interconnection will be from right to left. Further, since the main "path" from left to right will be occupied by a number of components, a direct feedback path is frequently not possible. An interconnection path which is parallel to, but outside the main flow, is therefore taken as a clue to a possible feedback path. If, however, a feedback path is placed within the body of the circuit, no simple pictorial clues exist and the path is discovered only by more intensive "analysis" in the electrical domain.

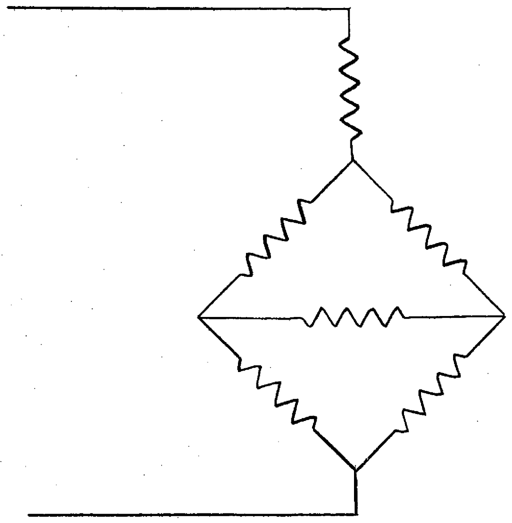
One further pictorial phenomenon concerned with the interpretation of function should be mentioned here. This is the situation in which two similar connectivities in a circuit have different intended functions. Figure 7-7 illustrates such an example - the first diagram includes the "above-below" relationship and suggests a signal divider with two tap-off points feeding a common load: the second diagram, while having the same connectivity as the previous diagram, indicates a bridge detector. Connectivity, therefore, is not sufficient to determine the correct way of drawing a diagram. Information regarding function is essential and is available from the descriptive scheme presented in Chapter 6.

7.6 Conventions

The features discussed above can be taken as a guide to the construction of a circuit diagram, but they do not precisely define how such diagrams should be drawn - thus there may still be an appreciable number of "acceptable" ways of drawing circuit diagrams. In general practice the meaning of the words "acceptable diagram" has been expanded. The various influences have been crystallized into a set of conventional ways of drawing certain functional circuit configurations. The conventional diagrams thus produced are not independent influences on drawing diagrams, but are combinations of the various influences into a configuration which has become accepted as normal.



(a)



(b)

Figure 7-7 A circuit in which the pictorial layout suggests different functions

The existence of these conventions greatly simplifies the task of drawing a diagram, as a person drawing a diagram does not need to apply each of the separate influences and attempt to combine them harmoniously. He need only configure his diagram according to a known convention. The use of these conventions also makes the task of interpreting a diagram simpler, and is of particular importance in the interpretation of circuits such as those shown in Figure 7-7.

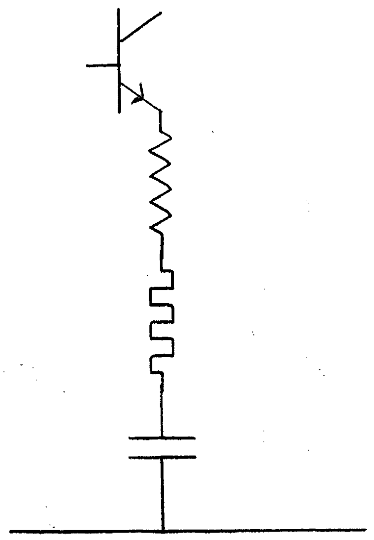
7.7 High Level Pictorial Objects

The above discussion on the nature of circuit diagrams has revealed the types of relations which exist and which should be included in any pictorial description. Little has been said about possible subdivisions into pictorial objects. For example nothing shows whether a simple one-level organization of pictorial objects is satisfactory, or whether a highly organized structure is called for. In an attempt to solve this dilemma, it is possible to inspect a number of diagrams for noticeable groupings, but no such groupings can usually be found.

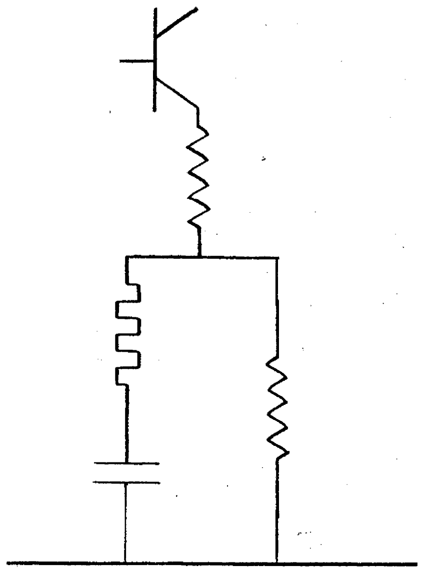
Before any decision on groupings is made, it should be remembered that the method of production, and the eventual use of the descriptions, affect the form of description, and are themselves affected by the description. Since little guidance is available, the description used is closely tied in with the mapping method employed. The full reasons for using this form of description are made clear in section 8.3. A hierarchical form of description is used in which the various pictorial symbols are grouped according to their origin in the circuit description. This appears reasonable if the manipulations on a diagram are considered. The changes called for at any time are usually related to functional units. Thus a parallel impedor may be replaced, an extra stage inserted, and so on. Figure 7-8 illustrates the effect, with the addition of a component in parallel with a series combination (which is manipulated as a unit).

7.8 The Pictorial Descriptions

The previous section indicated the hierarchical nature of



(a) the original circuit



(b) the modified circuit

Figure 7-8 Illustrating the manipulation of a series combination as a unit

the descriptions used, and the nature of the objects within the description. Earlier sections have given some indication of the relations which must be included. Some details of attributes and actual relations will now complete the discussion of pictorial descriptions.

There is no need for a notation for pictorial descriptions as occurred for electrical descriptions, because human beings usually communicate circuit information via circuit diagrams. Since this work implies a distinction between the two, some other method had to be found in Chapter 6. Pictorial descriptions may usually be conveyed via the diagram itself, so that a formal notation is unnecessary.

For primitive objects, the positional attributes used are XMAX, XMIN, XSIZE, YMAX, YMIN, YSIZE, and orientation. The redundancy is purely for convenience in programming. Attributes (other than positional) are the number of attachment points of a symbol, the type of the attachments (e.g. to a line or to a point), and the positions of the attachment points. Of these attributes only those related to attachment points are subject to variation when the system is applied to other domains, although this variation is only in detail of attachment. Higher level objects use the same attributes, and it is for these that the "attachment type" attribute is important (e.g. for a parallel impedor the attachment is to a line segment at either end of the object).

The necessary relations for the description have been indicated above; a number of such relations is used - the most important is "join" (note that this is not the same as the "con" relationship, but they may frequently imply one another during the mapping). The others used are "adjacent", "collinear", "left", "right", "above" and "below". The meanings of these are clear, and precise definition is left to the discussion of the programs. It may be stated here, however, that "left" and "above" are defined by simple inequalities which, although satisfactory for this purpose, would not be satisfactorily defined for the general picture processing problem, where no satisfactory definition yet appears to exist (Macleod, 1970).

CHAPTER 8
MAPPING

8.1 Introduction

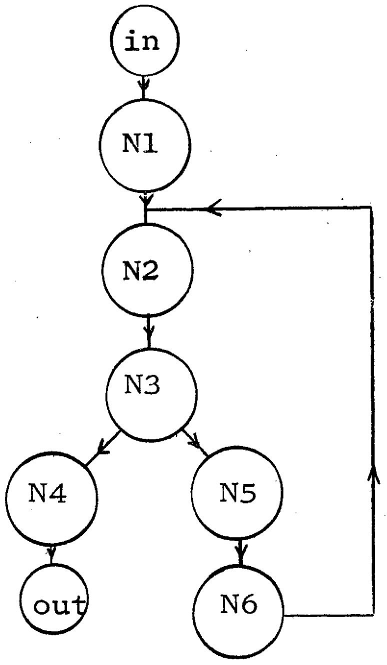
The mapping from circuit descriptions to pictorial descriptions is here discussed. As in Chapter 6 (Circuit Descriptions), two different methods* are described, hopefully adding insight into the problems involved, more so than would be the case with discussion of the chosen scheme alone. It has previously been indicated that the form of the descriptions used, and the method of mapping selected, are closely interrelated. The first method of mapping discussed is not strictly applicable to the descriptions finally selected. The description required for this first method is outlined, however, and the method developed to the point where its merits and demerits may be discussed. Some of the difficulties encountered with the first method are overcome by the second method which is developed into the mapping used by the present system.

8.2 Key Component Mapping

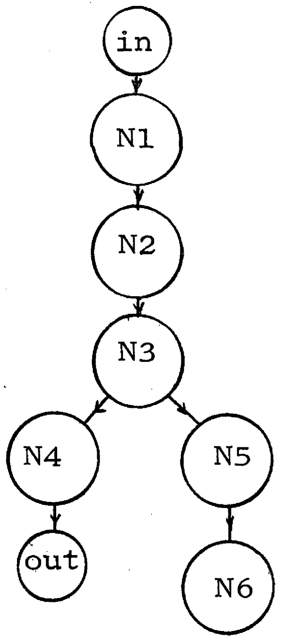
Key Component Mapping is based on the notion of "dominant components", introduced briefly in connection with the matrix method of circuit description (section 6.3). Additions to the circuit description, however, must be introduced first.

An attempt is made to introduce explicit information regarding "signal flow" in the circuit, which is available readily at interstage level, but is not so clear within individual stages. Since signal flow information appears to be useful, some explicit statement of it is attempted. To each object in the electrical description, a "signal flow list" is appended. For high level objects, this may be simply a sequence of stages. At a lower level, when an object is described in terms of components and nodes, the list is a sequence of nodes. Special markers for the input and output are included, and there may be branches in the list. Two forms of such node lists are shown in Figure 8-1. A node list for a circuit including feedback is shown in (a), with N5 and N6 being the feedback path, and N1, N2, N3 and N4 representing the normal forward flow. Figure 8-1(b)

*developed by the writer



(a) including feedback path



(b) including "dead-end" path

Figure 8-1 Examples of signal flow lists

also includes a branch in the signal flow, but in this case the branch is a "dead-end". That circuits with the "dead-end" type of path exist, is illustrated by the circuit shown in Figure 8-2 - with the node labelling shown in the figure, the signal flow list for this circuit is given by Figure 8-1(b). Here the path N5,N6 represents a bootstrap path, and contributes nothing further beyond N6.* It may, therefore, be considered as a dead-end path.

The second addition to the circuit description classifies individual components according to whether they are part of the signal flow path (an F component) or whether they are part of a shunt path (S component). In Figure 8-3 Z1 and Z3 are S components, and Z2 is an F component, while in Figure 8-2 the components C1, C4 and C7 are F components, while C2, C3, C5 and C6 are S components. There is a connection between this labelling and the flow lists, but they are not identical, and both assist in the mapping process.

Where a transistor is used as an active element, it may be considered an F component in the sense that signal flows from base to collector (in a common emitter amplifying stage). However there is also a shunt path from base to emitter, allowing a simultaneous classification as both S and F. It is sufficient here to resolve the difficulty by not classifying transistors.

A byproduct of these additions is the ability to determine readily the type of an amplifying stage (i.e. C.E., C.B., E.F.) in a given circuit; the signal flow lists provide this information.

Having established the above mentioned additions to the circuit description, the "key components" mapping approach may now be developed. This method is based on examination of the ways in which a human being produces a circuit diagram. The circuit is "grown" around a central "key" component. As the growth proceeds, a table of correspondences between electrical entities and their mappings in the diagram is kept. This is most necessary for nodes - which can become a set of connected line

*Only important contributions to flow are included in the list, so this statement is considered valid

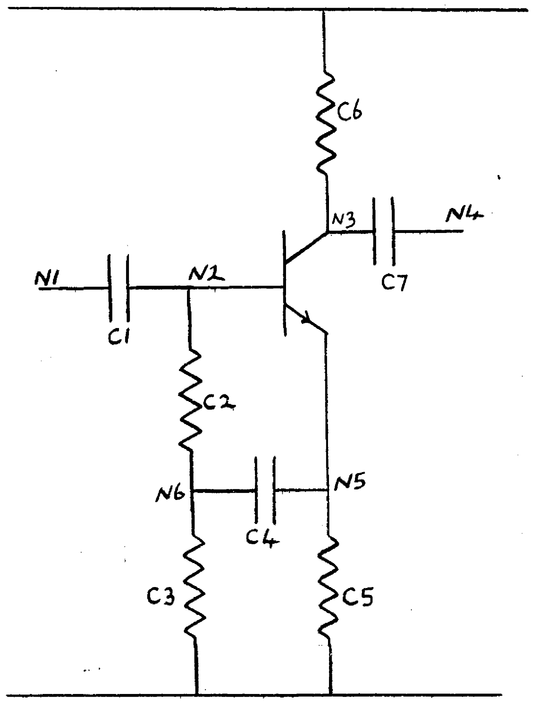


Figure 8-2 Illustrating dead-end paths, and also F and S components

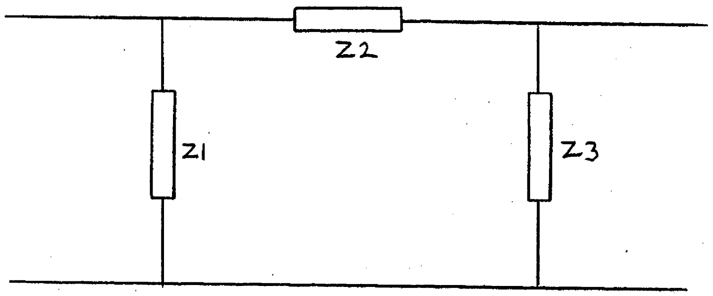
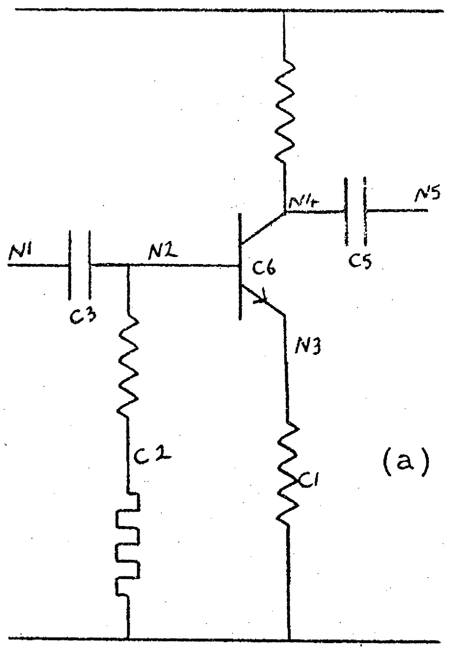


Figure 8-3 Illustrating F and S components

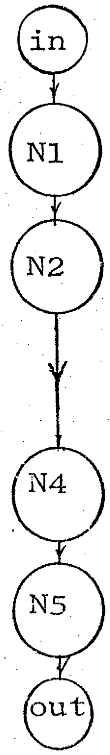
segments - and to which component symbols must be attached as the growth proceeds. The following example illustrates the method (Figure 8-4(a)). The flow list for this is shown in Figure 8-4(b). The only F components in this circuit are the two capacitors. Paths, such as represented by the impedor C2, are not included as dead-end branches on the signal flow list, because they merely form a shunt to earth.

The transistor is the key component here, and it is positioned centrally (e.g. (o,o)) in the first step. The correspondence table may be set up with the symbol corresponding to the transistor and the nodes N2, N3, N4 being represented by attachment points on the symbol for C6. The circuit is now "grown" by taking the first terminal of C6 (i.e. N3), and finding the components connected to it. The resistor C1 is labelled as an S component and should therefore be drawn vertically: its sense (i.e. up or down from the node) is determined by the fact that it eventually leads to earth. A check is now made to ensure that there is no clash with the remaining circuit diagram. In this case there is no clash, and the component C1 may be inserted. Since the other node of C1 is the earth line, a division at this point adds the earth line to the diagram. Figure 8-5(a) represents the circuit drawn so far. (Attachment points and "joins" between pictorial symbols are marked with dots).

Moving on to the next terminal of C6, two connections are found. The first is a series impedor whose length is two basic units (a passive symbol is arbitrarily called 1 unit). This impedor is an S component which is connected to earth, and must, therefore, be connected vertically downwards. Such a requirement clashes with the present state of the diagram as there is insufficient space available. The solution is to "stretch" the diagram by the addition of short line segments to the representations of the appropriate nodes. Once this is done, the impedor may be inserted, and Figure 8-5(b) shows the result of this step. The rest of the circuit may be mapped by a similar procedure.

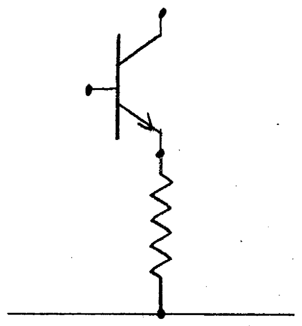


(a) the circuit

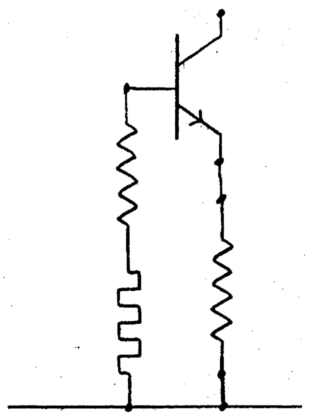


(b) signal flow list for the circuit

Figure 8-4 An example to illustrate the "key component" mapping method



(a)



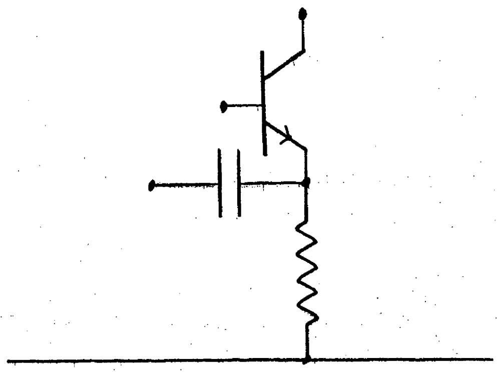
(b)

Figure 8-5 Steps in the generation of Figure 8-4(a)

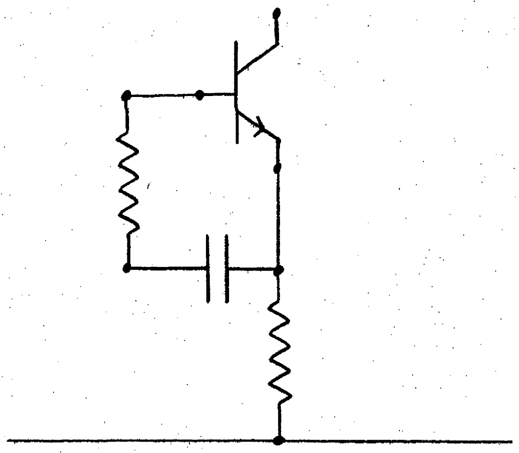
The circuit of Figure 8-2 throws further light on this mapping approach: it maps as in the previous example with the construction of the transistor, C5 and the earth. There is now a dead-end F component connected to the emitter terminal, and this may be added horizontally as shown in Figure 8-6(a). On mapping the node N2, it is found that C2 is an S connection between nodes N2 and N6 - some adjustments must be made to accommodate C2. By considering horizontal adjustment first, a horizontal line segment may be added to N2 which will allow a vertical interconnection to be made. The vertical adjustment cannot be made by considering only the current node, and the change must be made elsewhere in the diagram. A vertical line segment added to N5 solves the difficulty, and C2 may be inserted. This is shown in Figure 8-6(b) and provides an illustration of one of the problems which must be overcome in this method. If a symbol will not fit into the diagram as it stands, then a decision must be made as to what change will resolve this problem. An attempt is made first to make the adjustment by additions or changes to the node currently under consideration, as this will not affect the rest of the diagram - otherwise nodes further afield must be investigated. Often, several nodes must be modified simultaneously.

The necessity for modification of the previously drawn diagram is a serious drawback to this method of mapping. The origin of the difficulty is in the "bottom-up" nature of the mapping technique. When a particular area of the diagram is under consideration, only local information is used in the processing, and the drawing produced must be subject to change when relevant information from further afield is treated. This problem is not unique to the machine - human beings suffer from the same lack of detailed global insight, and often have to redraw portions of the circuit accordingly. The machine, however, need not suffer from such a difficulty, as the necessary global information is available in the circuit description.

Associated with the global information difficulty, is the problem encountered when there are changes in size in a diagram. If a component is replaced by another requiring more or less



(a)



(b)

<u>Figure 8-6</u> Steps in the generation at Figure 8-2

space, then the system cannot make the changes simply, without going through the whole procedure again: this deficiency is also due to a lack of utilization of global information.

8.3 Models

The difficulties in the mapping method described above can at least be partially overcome if a method of "top-down" mapping is devised. The most global information will be utilized first and so on down until, when the individual components are to be treated, sufficient information is available to permit correct positioning to be achieved.

In order to realize this objective, some method for the manipulation of global information must be found. The information required is organizational in nature for the level in the structure concerned. For example, if a cascade of several stages is to be mapped, then at a high level the information required is such that the individual stages can be organized relative to one another. At the next level down, the organization of each of the stages will be considered, and so on down until individual components are placed.

The necessary information can be obtained from a knowledge of the conventions discussed in section 7.6. These conventions provide information concerning the general organization of a particular type of circuit. They do not, however, give exact placement of symbols within the diagram. For example, in a simple common emitter amplifying stage, the convention is that the transistor appears somewhere between supply and earth lines; the base bias circuitry is to be found to the left of the transistor; the collector load circuitry is usually above the transistor, between the latter and the supply line; and the emitter connections are arranged between the transistor terminal and the earth line. The above information is that necessary for the top-down mapping method. There remains the question of how this information is to be available to the system.

The solution used by the present system is to provide a set of models, one for each of the circuit types known thereto. The models take the form of a prototype for a pictorial object of the type concerned. Certain detailed information is stripped from these prototypes, allowing generality in the application of each model. The models indicate the allowable types of sub-object, and the way in which such sub-objects are normally related to one another.

The provision of models in this manner leads to some further idea of what a suitable pictorial subdivision of the diagram might be. This concept was introduced in section 7.7, in which the solution to the pictorial subdivision problem was given, together with information concerning reasons for selecting this solution. With the introduction of these models, it is clear that the mapping procedures are likely to treat each stage as a unit, producing the pictorial organization for that unit, then moving on to the next stage. Therefore, it is convenient for the high level pictorial objects to correspond to the circuit objects, in agreement with the previous reasons.

8.4 Mapping with Models

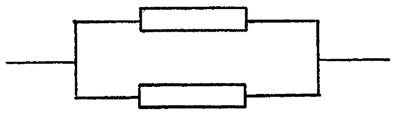
Having introduced the notion of models, it is now possible to outline the method by which the mapping process operates. In this section, only the general principle is outlined, but subsequent sections discuss some of the difficulties encountered and their solutions.

The highest level in the circuit description contains a single object ("CIRCUIT" for example). The only organizational information necessary here is to place the whole circuit somewhere. This is achieved by saying that the lower left hand corner is at (o,o) say. (The coordinates used throughout need only be arbitrary, since it is a simple matter for a display package to displace and scale a picture to fit onto a screen. The generation process therefore uses a fixed origin, and fixed sizes for the primitive components.) At the next level down, the circuit is found to be a cascade of two stages (for example). Information in the cascade model requires the first stage to be constructed to the left of the second. The appropriate information for the connection of the input of one to the output of the previous stage is included, as well as information tying the position of each stage to that of its immediate higher level object (the "CIRCUIT"). The linkage of information between levels in this manner allows "global" information to flow down through the mapping procedure, and thus to affect the ultimate creation of the various primitives. The problems of the previously described mapping method may therefore be overcome. The onus is placed on the models, rather than on the procedures, to provide sufficient relational information so that clashes do not occur in the diagram.

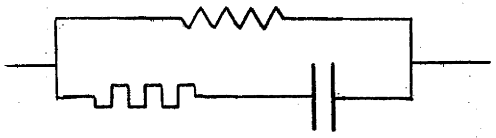
It may be noted that this method of mapping is very closely related to the use of a context free grammar for the generation of such arrangements of symbols. The similarity exists because one of the main functions at each level is to allot space in which sub-objects may later be constructed: thus the content of one of these spaces is generated largely independently of what goes on in the surrounds. The context is not, however, entirely irrelevant and a later section indicates that while such simple methods may work for simple objects, more complex context dependent manipulations are required for other circuits.

8.5 Chaining of Constraints

The change from a "bottom-up" to "top-down" approach to the mapping has, while solving many of the problems, introduced certain difficulties. Consider the segment of a circuit shown in Figure 8-7(a). Information obtained from the model for a parallel impedor indicates that the two branches must be drawn of equal length (to preserve the neatness of the diagram). The size of the constructed parallel impedor will depend on the smallest possible size of the longer of the two arms. This information is not at this stage known. Figure 8-7(b) indicates that the lower arm is itself a series impedor, and until this has been treated, the higher level information cannot be obtained. The components of the series impedor may themselves be complex impedors, thus burying the necessary information deeply in the structure. This situation is the analagous problem for



(a)



(b)

Figure 8-7 Illustrating the need for chaining of constraints

"top-down" mapping to that which existed for "bottom-up" mapping. The problem here is aesthetically better, however, because the possible solutions to this problem do not involve the creation of an unsatisfactory diagram with subsequent corrections being necessary. No diagram construction is actually performed until the lowest level in the structure has been reached, and all information can be collected. The situation is handled in the present system by creating "constraint chains". In the above example, the size of the parallel impedor is constrained to equal the largest of the minimum possible sizes of the two sub-impedors. In this way, a chain of constraints may permeate the structure. Two possible methods are available for the resolution of these constraint chains. The first is to traverse the chain backwards as soon as all the necessary information is available. This method requires back-tracking information to be kept for all chains. The second method waits until the entire structure is completed and then goes back and examines each constraint chain. In this latter case, backtracking information need be kept only for the current chain. Since chains may branch out when two or more parameters are involved in a constraint, the backtracking data may become extensive. The latter method is used in the present system, largely for convenience.

8.6 The Finite Set of Models

Since the mapping procedure is based on a model for each particular circuit type, the machine will sooner or later be presented with a circuit which does not fall exactly into any of the circuit categories known to the system. There are two possible ways of handling this eventuality, the applicability of either depending on the difference between the new circuit and the nearest known type. If there is no similar circuit type, then the new circuit is out of the range of knowledge of the machine, and should be formed into a new model, thereby effectively extending the range of the machine's knowledge of circuits.

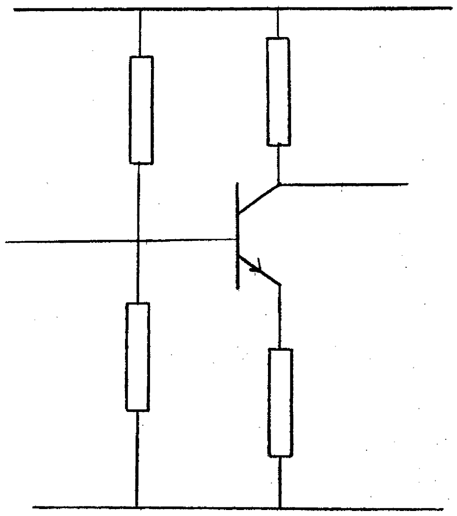
The second possibility is that the new circuit closely resembles (structurally) a type which is known to the system. In this case the new circuit is probably labelled as functionally the same as the model, so that the selection of the relevant model is not difficult. An attempt is made to make changes to the layout suggested by the model, in order to accommodate the new circuit, which is simply regarded as a modification of a known type. Figure 8-8(a) shows a possible arrangement specified by a model for a common emitter stage. A variation which may be submitted is shown in Figure 8-8(b). The procedure compares the model with the circuit and attempts to arrange all parts of the circuit which are common to the model, in the same way in which they would have been organized in a standard example of the model. For changes which are not drastic, this can be expected to provide sufficient general relational information to position the changed components, when taken in conjunction with their new connectivity information. In the example of Figure 8-8, the model will indicate to the system that the base bias circuit should be to the left of the transistor; the new connectivity information provides for the insertion of the extra line segment.

This approach represents an attempt to make intelligent use of information which the machine has on closely related problems, in the solution of a particular problem at hand. Since more general considerations of context are involved here, the comparison with context free grammar generations, no longer applies.

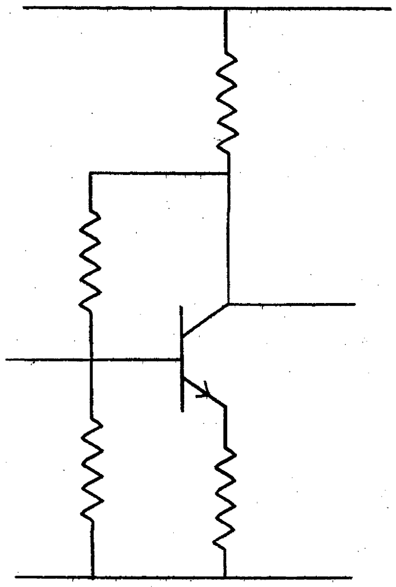
8.7 Example of Models

More detailed information is now given on the data in the models, and an example is provided to illustrate this. Since the principles involved in all models are the same, and since elaboration of some of the more complex models is space consuming, only a simple illustrative example is shown here.

It has been stated that models are generalized pictorial descriptions of an example of the circuit in question. Consequently the form of description and the relations used are all the same as described in Chapter 7 (Pictorial Descriptions). In general, a model has two levels of object, the top level consisting of the object which the model represents, the lower



(a)



(b)

Figure 8-8 Showing an arrangement from a model and a possible variation

level consisting of the sub-objects involved in the model. There are several forms in which information may be provided. One is a fixed datum, i.e. an attribute which can have no other value for this object. Another is a default datum - used in the absence of any other information. For example, a resistor has a default orientation of horizontal - normally the orientation would be specified by the context, but if the resistor exists with no constraining context, horizontal is assumed.

Another form in which information may be provided is an upward pointer from one of the lower level objects in the model to the top object. In a parallel impedor, for example, each arm must be constrained to be equal in size to the overall parallel impedor. The constraint on the size of the overall impedor will involve the minimum possible sizes of each of the two arms. A further common example is that orientations of sub-objects are usually related to the orientations of the higher object.

Another type of constraint is one which points down the structure. The example previously given for the overall size of a parallel impedor includes this type of constraint.

In addition to the provision of information in the form just described, there are also the various pictorial relations existing between the sub-objects. (See section 7.8 for the available relations.) These relations eventually manifest themselves in the form of further constraints on the various attributes in the structure, but this occurs only during mapping and need not be considered at present.

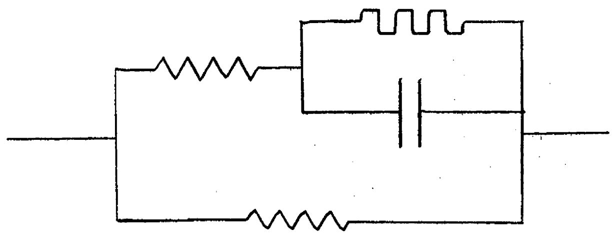
To clarify some of these ideas, consider the model for a series impedor - this consists of the object representing the series impedor, and two sub-objects. Each of these sub-objects is constrained to be an impedor, but may be any form of impedor. One relation (apart from the "join" relation) exists between the sub-objects and is, "collinear". This will eventually be converted into constraints on the placement of the symbols. The orientations of the two sub-objects must be the same as the higher object. The size (in the longitudinal direction) of the series impedor is constrained to equal the sum of the sizes of the two sub-objects. The width constraint is slightly more complex. The width of the whole impedor must equal the larger of the minimum possible sizes of the two sub-objects. (The determination of x and y sizes is useful in ensuring the non-overlap of symbols). The relations and constraints indicated so far have been sufficient to locate the two sub-objects relative to one another, but they do not specify position of the combination to the already determined position of the series impedor. To achieve this position specification, it is sufficient to add an up-pointer to the first sub-object, relating its minimum coordinates to those of the series impedor.

The above completes the information present on the series impedor model. Some of the constraints indicated (particularly size constraints) will become part of chains as the mapping process is expanded down through the sub-objects.

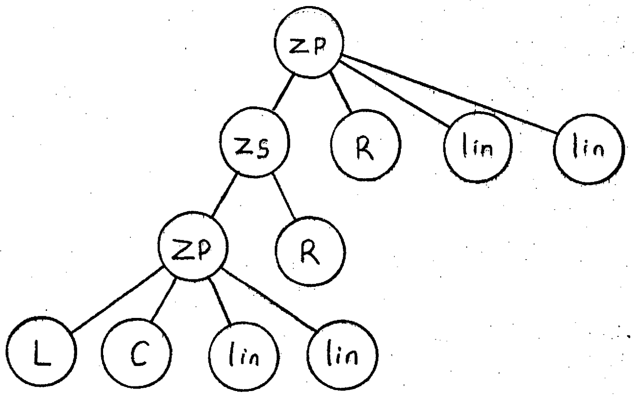
8.8 A Mapping Example

To complete the clarification of the mapping processes and models, an example is given of the operation of the mapping process.

Consider Figure 8-9(a). The hierarchical organization of the pictorial objects is shown in Figure 8-9(b). The objects only are shown in this diagram. The mapping process starts with the highest level ZP (parallel impedor) in the circuit description, and invokes the model for a ZP, from which a duplicate is made. More complete information is called from the electrical description as to the exact nature of the sub-objects. Various constraints from the model are added (such as orientation constraints pointing up to the ZP which has its orientation chosen by default as horizontal). The relation "adjacent" is placed between the two main sub-objects, and "join" relations are placed between these objects and the lines. Such constraints as - "the XSIZE of the ZP must be at least equal to the size of each of the two branches" - are added as indicated in the previous section. The system moves down to the next level



(a)



(b)

Figure 8-9 An example to illustrate the mapping procedures

and each of the sub-objects is mapped in a similar way. The process is continued until the lowest level is reached.

A second phase is then entered. All the relations inserted into the structure are converted into appropriate constraints on the attributes affected. This could not have been done previously because the precise constraints depend on the nature of the objects concerned (e.g. the constraints for a "join" relation between a resistor and an impedor depend on whether the impedor is a parallel impedor).

Once this is completed, the third phase is entered. The necessary information to satisfy the constraint chains is now available, and the chains are traced and replaced by the actual data required. All the data concerning position and so on for each object is then present, and the picture structure is available for use by the drawing package (see section 4.2).

CHAPTER 9
PROGRAM DESCRIPTION

9.1 Introduction

Chapters 6, 7 and 8 developed novel portions of the communication system first introduced and outlined in Chapters 3 and 4. A set of programs which has been developed embodying these principles is described here. Certain features of the system outlined in Chapter 5 have subsequently not been discussed; these are now presented with regard to their implementation. These aspects of the system, since they do not constitute the novel portions of the project, have received less attention than might otherwise have been the case in a complete communicator and environment system.

Before attempting to write programs, the question of "why simulate the system at all?", should be considered. It may be argued that programming the system involves embodying the ideas developed into a set of routines, and therefore contributes no further information. There are, however, reasons (both obvious and subtle) for actually writing programs, and certain of these reasons affect the approach taken here:

(1) To prove that the system can actually be built. In the conceptual development of a system, ideas are not always complete; by programming, the designer is forced to specify his principles precisely and debug his programs.

(2) A programmed system allows various assumptions to be tested. It may have been assumed, for instance, that a certain mode of communication is desirable. This may be verified by trial of the assumption.

(3) The ease with which new ideas may be tried on a programmed system. An example of this is the correction of unforeseen difficulties involved in practical use.

(4) Programming of a system is viewed as a vehicle for test and experimentation. Because ease of modification is a useful development criterion, developed programs are not necessarily as efficient as they might be, but may include redundant information,

and perform certain tasks in an indirect manner, to maintain the advantages of ease of development and modification. This is in contrast with programming a working system for a user environment, where speed and efficiency are important criteria and ease of modification is not as important. The above principles may be seen in the programs outlined in the present chapter. Examples of the programs themselves may be found in Appendix 1.

9.2 General Organization

The system has been developed in the form of a relatively large number of short routines, each of which performs a specific task on one or more of the data structures. These routines combine in a hierarchy of more complex operations until, at the top level, a very short main program controls the sequence of operations, making the task of modification or adding new features easier, but causing the programs themselves to be rather more difficult to be understood.

The sequence of operations available to the user is biassed towards the use of the mapping procedures. Actual communication methods are discussed in section 9.4 but, for the present, the general sequence is to input either a new circuit description or a modification of the current one, and for the machine to produce and display the associated diagram. The sequence is then repeated.

Before outlining the structure of the system, the computing facilities available to the writer are indicated as they influence the programming. The system has been developed on a CDC3600 computer with 64K words of core storage each of 48 bits (1K=1024). 32K words are reserved for the operating system, leaving 32K for the user. The relevant facilities include a general purpose CRT display (DD250) which is capable of point plotting (1024 x 1024 grid), vector generation, and character writing. A light-pen is able to detect specific elements in the display, and a set of function keys is provided. Available independently, but physically nearby, is an alphanumeric keyboard display with a capacity of 1000 characters. The operating system restricts the use of the alphanumeric displays to high priority interactive programs, which cannot communicate directly with a main job using the general purpose display. Communication is possible, however, by storing data generated at the keyboard display on a drum backing store, and accessing this from the main program. Standard batch-processing peripherals also exist.

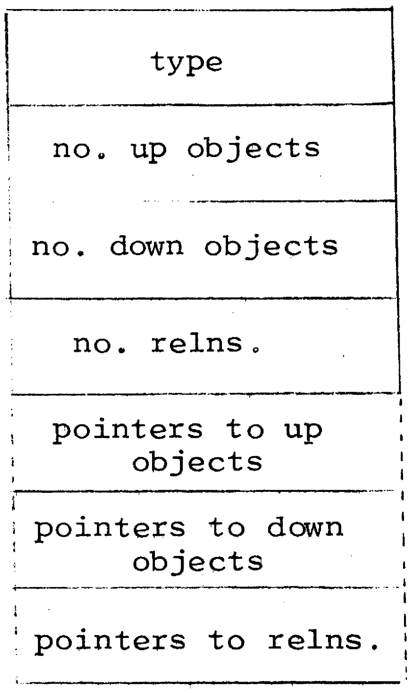
In the present system, models are generated in the form of a pictorial data structure (as described in section 8.7) by a separate generation subroutine for each model known to the system. Initially these models were generated at the start of a run, and maintained for reference throughout the run. Space requirements, however, have led to a procedure whereby each model is generated when needed, and then erased after use.

The operation of the mapping procedures has been explained in Chapter 8, and will consequently receive little further attention here.

9.3 List Structure

The existence of two fairly complex data structures in the system (as outlined in Chapters 6 and 7) indicates that some use should be made of one of the various data structure or list processing languages available. Features required include the representation of a hierarchy of objects with linkages between them. The objects in the descriptions all have a set of attributes, and relations must be set up between the objects. Because of the flexibility of the various languages, they are all able to represent these entities in one way or another. The choice therefore must be made on such grounds as convenience and availability . The Symmetric List Processor (Weizenbaum, 1963) has been chosen mainly for its availability on the computing system used.

A well-defined format has been employed for each type of entity and the same format is used for both the circuit structure and the pictorial structure. All entities in the structure are represented by a list. The structure of an object list is shown in Figure 9-1(a). Note that only the data cells i.e. not the



(a) an object list

orientation	2
XMAX	3
XMIN	4
YMAX	5
YMIN	6
YSIZE	7
XSIZE	8
no. connections	9
connection type	10-13
connection value	14-17

(b) the attributes used for each object

Figure 9-1 An object list and its attributes

header are shown in this diagram. The SLIP system, while symmetric within the lists, is not symmetric across a list structure. To maintain ease of tracing through a structure, it is necessary to include both up and down pointers. This leads to some problems with list erasure and the writer had to develop special methods to achieve effective erasure of lists. Attributes of an object are not maintained on the main object list, but the description list facility of SLIP is used for this purpose. Each of the attributes is represented by a code number which is used to identify the associated datum in the description list. A list of the attributes and their codings is shown in Figure 9-1(b). This list represents attributes of pictorial objects only, as no attributes are necessary for circuit objects. Component value, as previously indicated, has not been included in the present system but could be included if desired as indicated in section 6.6. The only other attributes of circuit objects concern the number of connection points, and this information is available from the appropriate models as required.

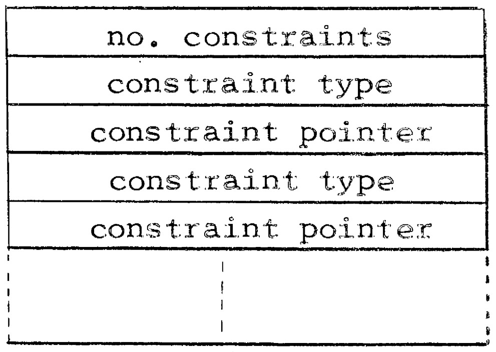
During the mapping process, the datum representing a given attribute may take on one of many forms. It may not be present: usually indicative of its not having been processed as yet. It may take on a negative value: indicating that the magnitude of this datum is the default option for the given attribute (no attribute values in the system are negative). A positive datum indicates an absolutely specified datum, which could indicate a non-variable quantity, or an attribute which has been completely processed.

If the attribute is a pointer to another list, then the attribute is not yet specified but is determined by one or more constraints. Constraints are specified in the system by a two level arrangement of lists called "constraint lists" and "constraint sublists". An attribute can be the subject of more than one constraint; such constraints are collected together in the constraint list, only one of which may exist for each attribute of an object. The constraint list consists of a cell indicating the number of constraints, and cell pairs, the first of which contains the nature of the constraint e.g. greater than, less than, equal to, not equal to, and the second, a pointer to the associated constraint sublist (Figure 9-2(a)).

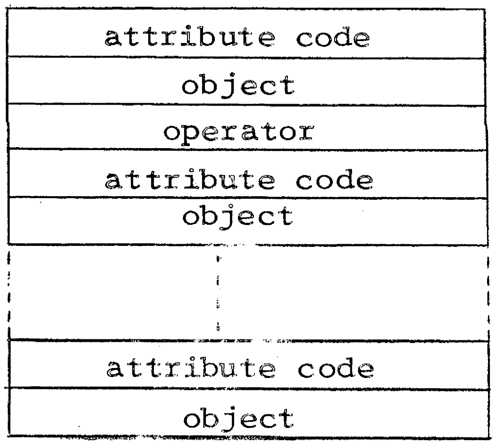
The information in the constraint sublist specifies the value to which the current attribute is constrained in the manner indicated by the constraint list. This value will in general involve an attribute of another object in the structure. Thus, cell pairs exist in constraint sublists, the first containing the code for the attribute in question, the second containing a pointer to the object. Constraints may also involve functions of several quantities. Constraint sublists must therefore provide information for the evaluation of these functions. This is done by interspersing each of the cell pairs in the constraint sublists with codes representative of arithmetic operators which are to be applied in the evaluation of the function.

To clarify the situation consider a parallel impedor. The width of this must equal the sum of the widths of the two sub-objects. If no other constraints exist for this width, (the YSIZE, if the impedor is horizontal), then the constraint list will contain three cells. The first indicates one constraint, the second indicates "equals", and the third points to the constraint sublist. The constraint sublist contains five cells and may be read as "the YSIZE of Z1 plus the YSIZE of Z2".

The attributes indicated in the constraint sublists may themselves be the subject of constraints and the constraint chains indicated previously are thereby set up. There is no indication of whether a chain is pointing up or down a structure, and the two cases are treated in the same way. Since a given chain may point both up and down the structure, there is the possibility that a chain may be set up which points in a loop to itself. There is no protection at present against this occurrence. However, if the models are correct, this situation will not arise except in the case of program failure. This is because the mapping is closely related to context-free generation.



(a) a constraint list



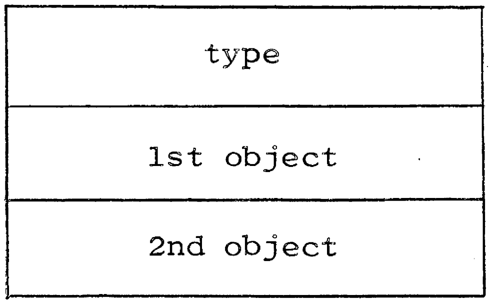
(b) a constraint sublist

Figure 9-2 Format of lists involved in constraint chains

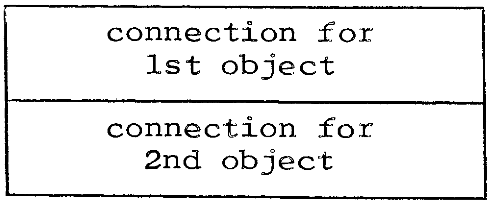
Another possibility is that two or more constraints may conflict. Again this does not occur in the present system and is not handled. If, however, any learning features are included in the future, (see section 10.6), it is feasible that the situation may arise. Sutherland (1963) attempts to solve the problem of conflicting constraints by an iteration technique in which all the effected variables are subject to modification in the hope of finding a set which satisfies all the constraints. A similar approach may be necessary here if the problem arises.

In the description of the data structures so far, relations have not been considered. The linkage between the remainder of the structure and the relations has been indicated in the form of relation pointers on object lists. All the relations involved are predicated on two objects. Both of these objects contain pointers to the relation to maintain symmetry in the structure. In most cases the relation list itself is very simple, and contains only three cells - the first indicating relation type, and the second two being pointers to the two objects involved in the relation (Figure 9-3(a)). The reverse pointers are in accordance with the policy of ensuring that the data structure can be traversed completely, starting from any point, and without any additional tracing information. This simplifies the writing of routines for new functions. (Constraint chains do not have reverse pointers, but they are not considered part of the structure proper.) The information is essential for relations, because when a relation is encountered in the mapping via one of its objects, the only way of determining the other object involved is through the relation list reverse pointer.

Most relations are predicated on objects themselves. The relation lists provide sufficient information for these. "Join" relations, however, should be strictly predicated on the connection points of the objects. The extra information for "join" relations is provided by a description list which specifies which connection point is involved for each object. Figure 9-3(b) illustrates this.



(a) a relation list



(b) description list for "join" relation

Figure 9-3 Format of relation lists

At some stage during the mapping process, the relations must be converted into appropriate constraints on attributes (see section 8.8). This task is performed by subroutines, one of which must exist for each relation type in the system. The "knowledge" of which pictorial organization corresponds to each relation is thus built into the programs. It is here that the "aesthetic requirements" mentioned previously are incorporated, for these routines are responsible for the generation of constraints which will satisfactorily arrange the components.

9.4 Communication

So far in the discussions in this chapter, the parts of Figure 5-2* which involve control of the system, input of commands and data, and output, have received no attention. The work reported here is concerned with aspects of interactive graphical communication; actual communication with the machine is therefore important - this is true in the development of a user system, but as pointed out in section 9.1, programming is here considered as a vehicle on which to experiment with ideas. The objectives of this project have been to investigate the incorporation of novel knowledge into a communication system, and a bias towards these features exists in the experimental programs. It has been also pointed out that one of the anticipated results of this system is the easing of the burden on a user who must communicate with a sequence of light key selections - consequently control of the machine and transmission of data have been developed in this simple form.

During the earlier development of the system, when work concentrated on structures and methods for manipulating the structures, the need for interactive operation did not arise. Consequently a large part of the system was developed in a batch environment with input information on cards, and output on a line printer. The form of this output was initially as diagnostic listings of the generated structures. An example of such a structure is shown in Appendix 2, where the pictorial structure shown is at a stage at which a number of constraint chains have developed and untreated relations are present. The structure is

*Page 73

almost at its maximum size here, because subsequent operations eventually remove all the constraint chains. Later, as the mapping procedures become capable of producing complete diagram information, a method was developed for taking the pictorial structures and producing a diagram on a line printer. A line printer was used rather than an incremental plotter, as it enabled less expensive operation and also quicker turnaround. An example of this form of output is shown in Figure 9-4. This routine represents the earliest form of the display package shown in Figure 5-2, and was written in such a way that it could be readily modified to produce its output on any medium. The actual output was achieved by a short routine for drawing line segments; this could be readily replaced for a change of medium. Consequently these routines were also used as the basis of the display package when the system was updated to operate in the interactive environment. While the system operated in the batch environment, the control information was of a simple form obtained from cards. In essence, this was simply a statement of the circuit which was to be drawn. A set of mnemonics, which represented the various circuit objects, was known to the programs, and the input was punched on cards using a nested expression to represent the circuit of interest. It was mentioned in Chapter 6 that the description of a particular circuit need not contain every piece of information about a circuit, as the system was expected to "know" most of this. Consequently only that information relevant to the particular instance needs to be included. The nested expression for a parallel impedor consisting of a resistor and a capacitor, for example, need only include:

ZP(R,C)

This information must pass through an "input converter" (Figure 5-2), during which process the appropriate circuit structure is produced, involving a relatively simple task of analyzing the input expression and generating the appropriate structure. The data structures produced in this phase are simplified, again on the assumption that the machine will "know

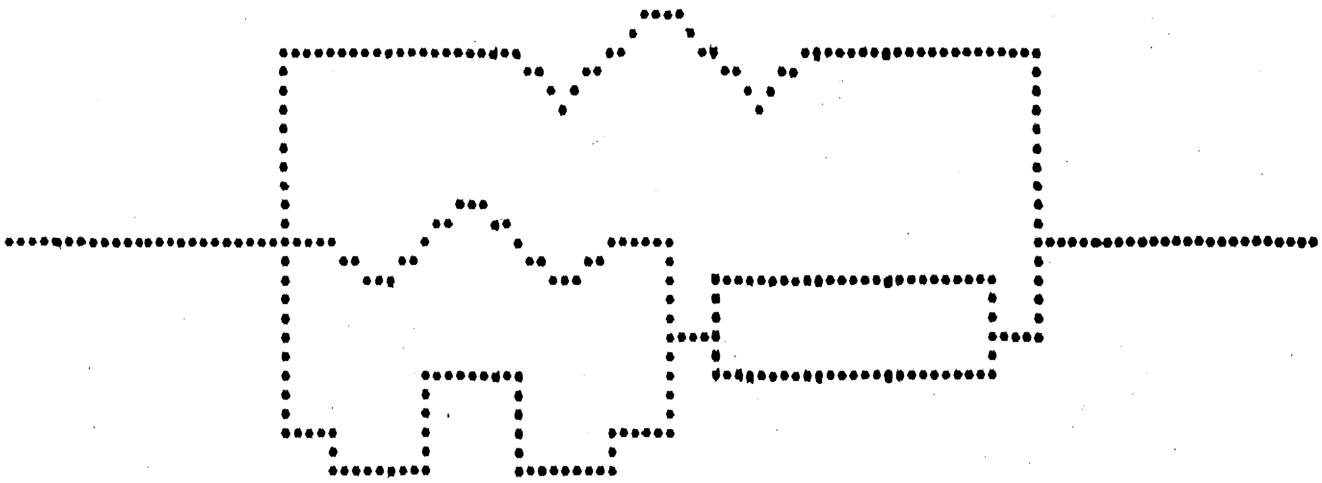


Figure 9-4 An example of the line printer output of diagrams

what is meant". This structure generation represents the first form of the input converter.

Even in a card input system, the input converter described above does not provide sufficient facilities, as it is necessary to input a complete circuit with every new input. A normal operation involves the input of a complete circuit, followed by a sequence of additions and modifications to the circuit; this feature must therefore be included. Two pieces of information are required:

(1) a specification of the part of the circuit subject to modification, and

(2) the modified information.

Once the system is in an interactive environment, there are no difficulties with the specification of the portion subject to change - this is achieved using a light pen pointing operation, which cannot be done in the card environment. Since the card form of input is only a temporary measure, a simple method of input suffices. The objects in the circuit description are numbered sequentially, numbers being used to specify which portions of the circuit are to be modified. The new specification for this part is then given in the same format as before. Thus the inputs:

```
ZP(R,C)
2:ZS(L,R)
```

are equivalent to the single specification:

```
ZP(ZS(L,R),C)
```

The above represents the state of the input converter as developed for operation in the batch environment.

When conversion is made to an interactive environment, the major portion of the system remains unchanged. The display package is subject to minor modification; provision must be made to allow various symbols in the diagram to be detected by the light pen.

The input converter is subject to greater change. Initially, the input of nested expressions was completely replaced by light pen manipulations. A set of light buttons is provided, one for each circuit object known to the system. Using these light buttons, the circuit may be developed one step at a time. The highest level object in the circuit structure is selected, and the system attempts to produce a corresponding diagram. Since a large portion of the information is missing at this stage, the displayed diagram will contain a number of general impedance symbols. The operator selects one of these with the light pen, and then selects the light button corresponding to the desired expansion of this portion of the circuit. The mapping procedure is then invoked once more, and a new diagram produced. This cycle is repeated until the desired diagram is produced. At any time during this process, a similar procedure may be used to change parts of the circuit, or to abandon the circuit and start again.

This form of input communication is evidently rather wasteful of computing resources, however. While a circuit is being generated, it is not necessary to pass through the mapping process at every stage and display a diagram. Some means should therefore be provided to input a complete circuit before the mapping procedures are invoked. This is achieved by providing the facility for inputting a nested string as used before, on the adjacent alphanumeric display (see section 9.2). By using a combination of the two methods, it is possible to efficiently input a circuit via the keyboard and then to use the light pen to perform various manipulations on the circuit.

The monitor portion of Figure 5-2 does not require much discussion here. The operations are simple, and the cycle of events has already become apparent. The control functions are in fact built in to various parts of the system. The main tasks are to cycle through the various portions of the system, and to display the appropriate items and light buttons for light pen detection.

9.5 Examples of Operation

So far this chapter has been concerned with the description of the various parts of the system, in particular the list structures and methods of communication. To complete description of the programs, this section covers the extent of the present capabilities of the system and illustrates these with examples of the operation of the programs.

With a system such as the one developed, capabilities are dependent upon two major factors:

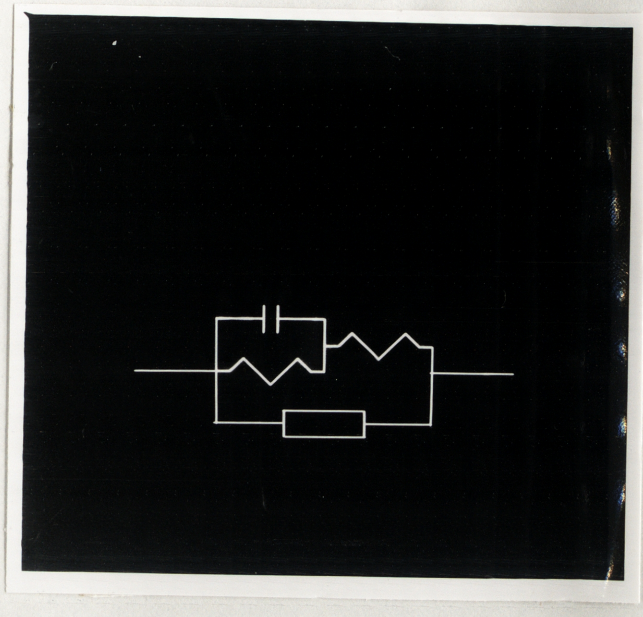
(1) The range of models included, and the ease of addition thereto

(2) The extent to which the procedures are able to perform the organizational tasks required.

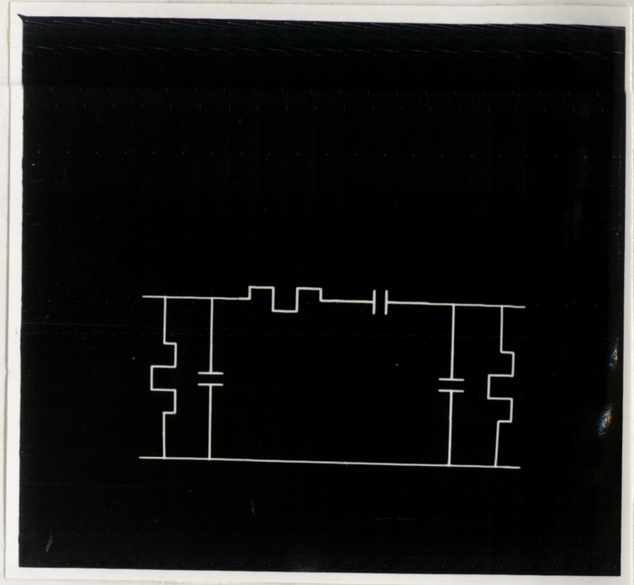
To alter the capabilities of the system, therefore, one or other of these areas must be upgraded. Extension of the procedure capability in general requires further inventive programming and is the more difficult of the two. An example of this type of extension is the provision of more elaborate procedures for handling circuits which do not conform exactly to a model. Extension of the range of models available is a much simpler task, and involves only the provision of data (in this case in the form of code to generate the data) of a type which already exists.

Because of the experimental nature of these programs, and the ease of addition of extra models, the models which have been initially incorporated are in some ways restricted, but include enough circuit types to test the procedures developed and to allow a certain amount of generality in the circuits available.

The existing models start with all of the primitive objects, and models for series and parallel impedors which allow recursive definition of a large class of two-terminal impedors. One of these, as generated by the program, is shown in Figure 9-5(a). The capabilities with passive circuits is extended by the definition of certain filter stages. Models for T and Pi filter stages are included. While these two circuits are equivalent when a number of stages are cascaded, this does not affect their



(a) An impedor



(b) A Pi-filter

Figure 9-5 Some generated diagrams for passive circuits

separate existence, because, when used in a small number of stages, they are thought of as being separate, though dependent, circuit types. An example of a Pi-filter stage is shown in Figure 9-5(b).

The models currently available for the use of active elements are also a restriction as only common emitter stages may be handled. This, however, allows experimentation with all the features of the system. The cascading of a number of stages in a circuit is provided by a model for a cascade of two stages. More stages may be cascaded by virtue of the fact that a cascade itself is, for the purposes of the cascade model, also a stage, and therefore recursive definition can cascade any number of stages. This represents an inefficient method of cascading several stages from both the core storage, computation time and user points of view. In a system programmed for a real environment, the cascade definition would allow several stages at the one level.

With regard to the procedural aspects of the system's capabilities, the programs are capable of taking any circuit description which exactly fits any valid models, and producing a diagram from this, implying the satisfactory manipulation of the aesthetic organizational requirements as dictated by the relations. The ability to handle circuits which are similar to, but not the same as, known models has been discussed. This has been implemented to a limited degree, and allows useful variations to be made in the production of circuits without the need for new models to be included. This feature is illustrated in the example given later.

At the present time the provision of feedback paths over several stages cannot be achieved - discussion of proposed methods for achieving this is given in section 10.6.

During operation of the system, light pen selections are used for a number of different tasks. Four phases of operation may be identified as follows:-

(1) Initialization: in which the system starts with no current

circuit, and presents a selection of light buttons to the user. Each button represents a model and, if selected, the model becomes the current circuit, which is then mapped into a diagram.

(2) Generation: this is entered after the initialization phase, and the circuit description is expanded by the specification of detailed information for various parts of the circuit. Two forms of light pen operation are performed: the first is the selection of an object (in the displayed diagram) which is to be the subject of expansion; the second is the selection of a light button corresponding to a model which is to be used in the expansion. After the new diagram is displayed, new objects therein may themselves be the subject of further expansion, and so on.

(3) Deletion: at any time during the generation phase, the deletion phase may be entered by the selection of a light button. Once in this mode of operation, the light pen is used to indicate a component which is to be removed from the circuit. Such deletion operations cause an appropriate redrawing of the circuit.

(4) Insertion: this phase may also be entered at any time, and the light pen is used here to select a button corresponding to a circuit element (or type) which is to be inserted into the circuit. The pen is then used to select the nodes to which the new component is to be connected. It should be noted that this phase does <u>not</u> involve the pictorial placement of a symbol. The user may point to any portion of a node, and the system interprets this as an electrical domain connection. The new circuit is then mapped into a new diagram, and the symbol may, in fact, be connected to quite different points chosen by the machine. Other parts of the diagram may also have undergone suitable modification.

As previously mentioned, groups of the above light pen operations may be replaced by nested expressions input from the keyboard, thereby speeding the generation of a particular circuit.

In order to demonstrate the use of the various features of the programs, an example is now given in which a circuit is developed using the light button method alone. While this is not the quickest way of developing the circuit, it demonstrates the system more completely. The series of photographs in Figure 9-6 illustrate the output. To obtain Figure 9-6(a) the operator enters the initialization phase and selects a common emitter amplifying stage with the light pen. The extent of the machine's knowledge of these circuits is used to produce the figure shown. To obtain Figure 9-6(b), the generation phase is entered, the emitter impedor selected for expansion, and the "parallel impedor" light button selected as the model to use in the expansion. The same procedure is used in the specification of an R and C as the parts of the parallel impedor, and of an R as the collector load. This produces Figure 9-6(e). Attention then focusses on the base circuitry with the expansion of the lower bias impedor. The next step is to enter the deletion phase, removing the upper bias impedor, and then to enter the insertion phase to add the new impedor. This gives Figure 9-6(g). Finally the generation phase is used to complete the diagram. As previously indicated, this circuit could have been input as in Figure 9-6(f), say, via the keyboard in one step, and the change then carried out. The method shown is more illustrative, however.

Figure 9-7 shows a more complex example, used by the writer, of a magnetic phono cartridge equalization preamplifier. This is obtained by a fairly straightforward application of the above method, and it consists of a cascade of two stages. The emitter impedors of each stage have been generated as

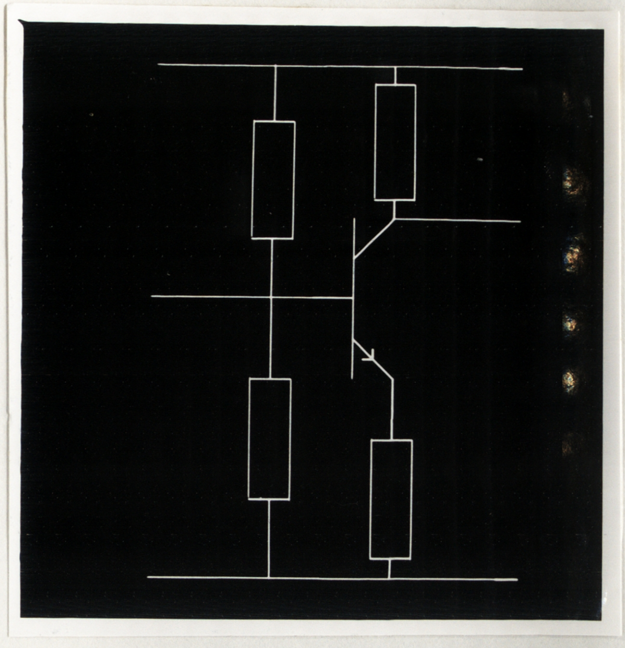
ZS(R,ZP(R,C))

while the collector load of the first stage has been expanded as

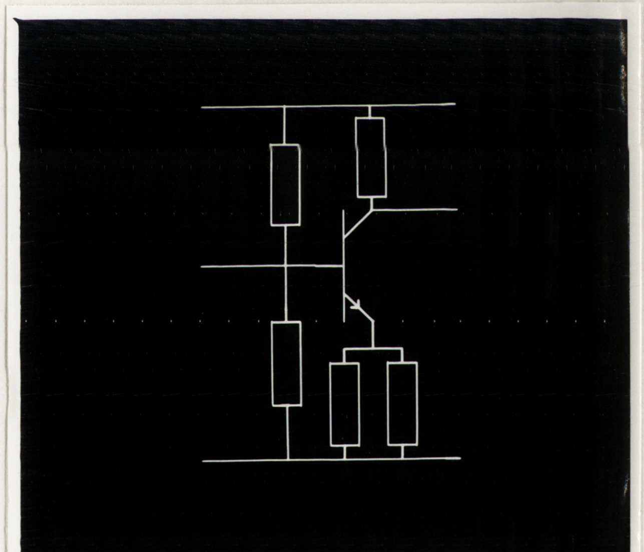
ZS(ZP(R,C),ZP(R,C))

The latter provides the necessary frequency response of the amplifier.

The above examples indicate that, while the system outlined in this chapter is experimental, it is capable of convenient

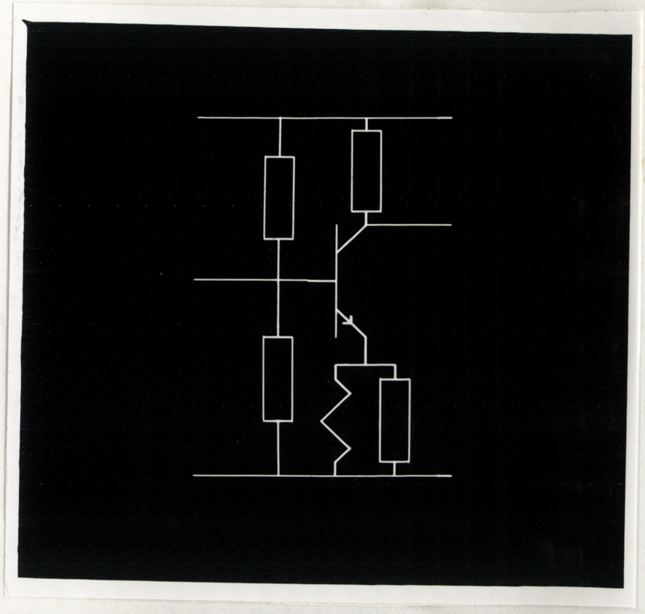


(a)

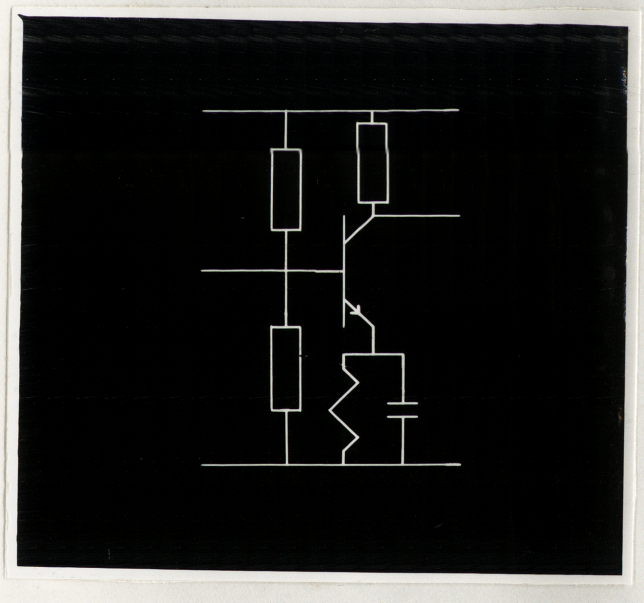


(b)

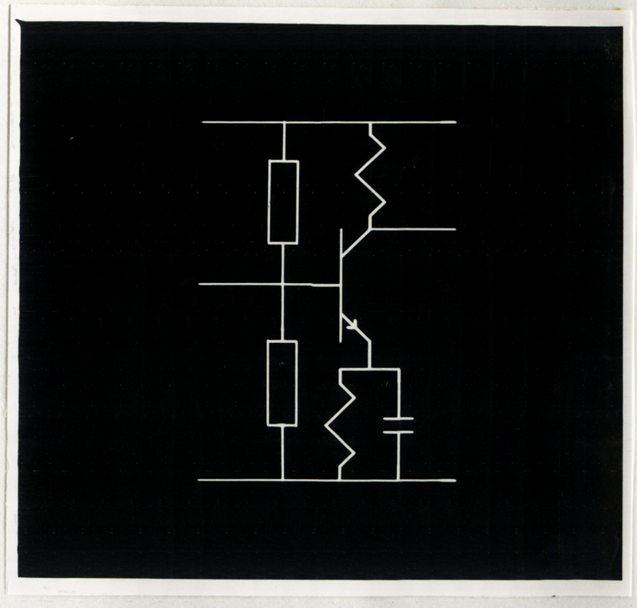
Figure 9-6 Generation of a circuit using the light pen alone



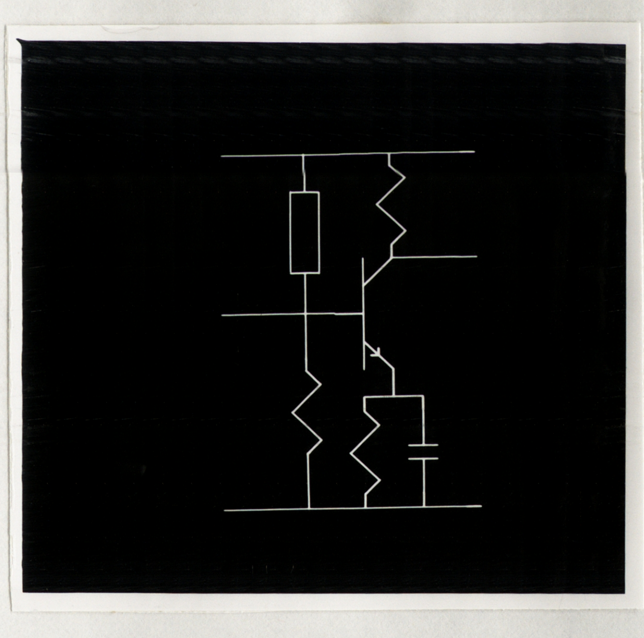
(c)



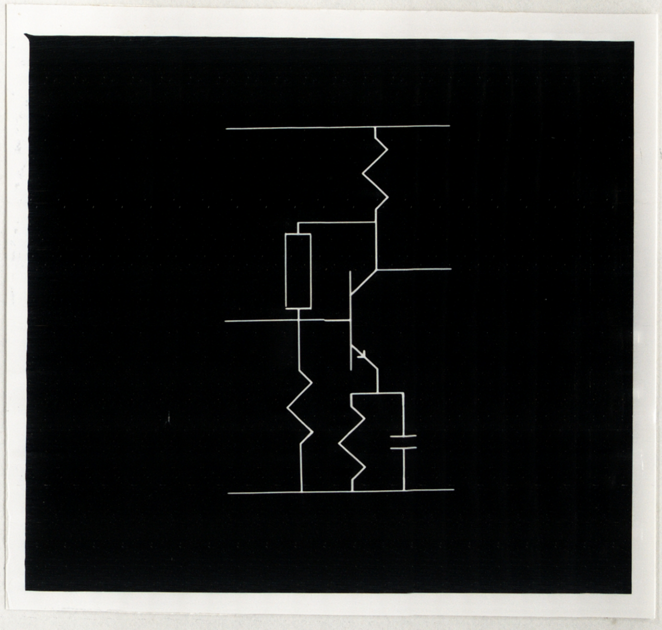
(d)



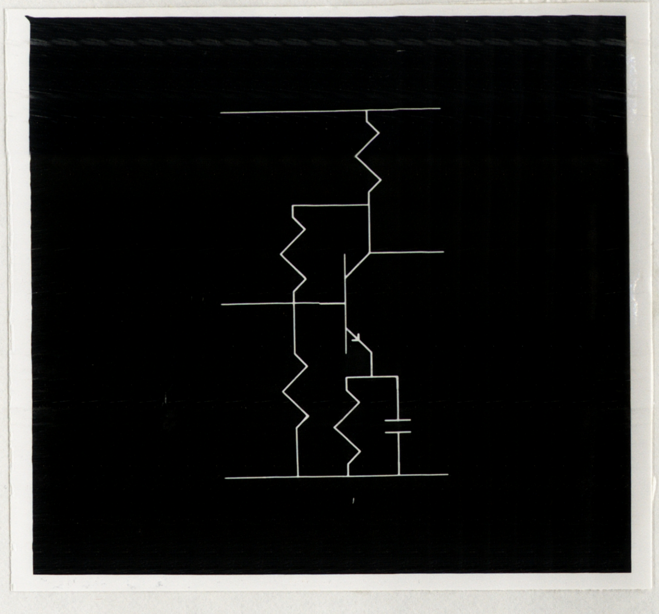
(e)



(f)



(g)



(h)

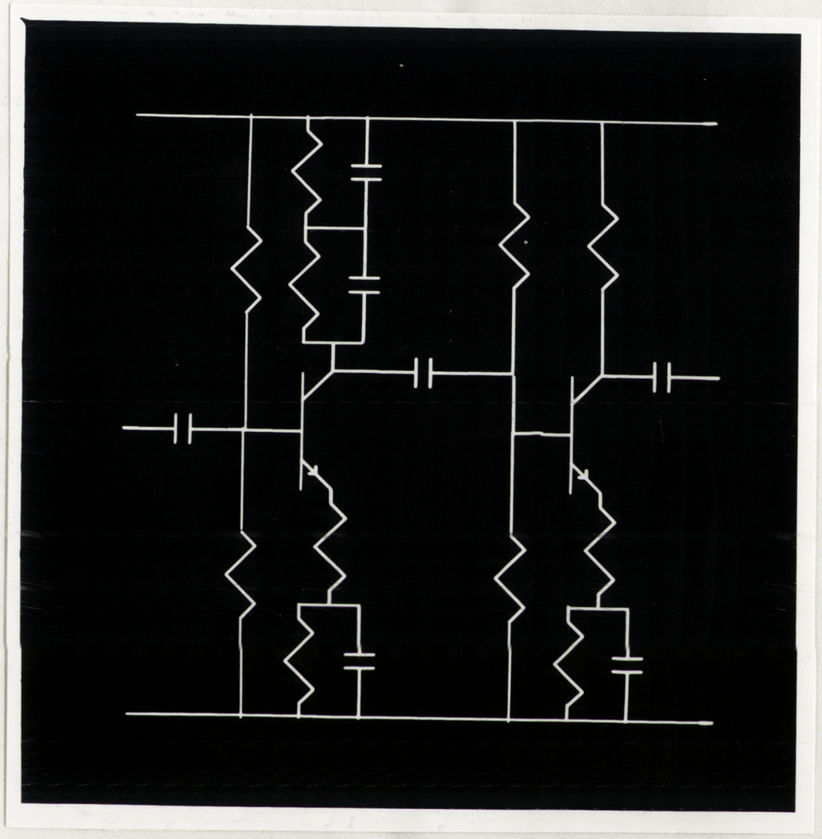


Figure 9-7 A magnetic phono cartridge equalization preamplifier circuit diagram produced by the system

communication over a fairly wide range of circuits.

It should be emphasized that the use of the light pen in this system does not correspond to the "pick-and-place" method criticized earlier (e.g. pp.26-27). In this system the light pen is used for the selection of a portion of a circuit to be modified, and for specifying what is to be done (electrically) with that portion. This may result in considerable change to the diagram layout, none of which has been explicitly specified. The operation is thus fundamentally electrical in nature. Light pen operations in other systems are graphical in nature, and layout changes due to an electrical change must be spelled out step by step at considerable length.

The achievements and shortcomings of the present system are discussed in detail in chapters 10 and 11, along with a number of possible extensions.

CHAPTER 10

EXTENDED APPLICATION OF THE COMMUNICATION MODEL CONCEPTS

10.1 Introduction

The earlier chapters of this work addressed themselves to a general problem existing in interactive computer graphics - that is, the communication methods employed are unnatural to the human user and frequently very tedious to use. General considerations of the requirements for more natural communication led to the development of a model for an interactive computer graphic system. This model has been applied to the particular case of a system for communicating in the field of electrical circuit diagrams. The system design was thus directed to demonstrating the principles of the model, rather than on producing a system optimized for electrical circuit design.

This chapter examines the general concepts used in the communicator (and also many of the more detailed aspects) and illustrates their use in systems designed for communication in other fields. The areas covered are widely disparate and the generality of the communication concepts presented herein is thereby established.

10.2 General Communicator Concepts

Absolute generality in a communication system is clearly an unachievable aim in the near future, and the communication model proposed here makes no such claim. Rather attention is focussed on a set of communication domains characterised by the use of abstract sketches or diagrams to represent some real world situation. The connection between the diagram and the real situation is not, as has been indicated, straight forward, and methods are proposed in the present model for effecting the transformation from one to the other. As indicated in chapters 2, 3 and 4, this transformation has been avoided rather than addressed in other systems, by forcing the user to simultaneously supply both diagrammatic and real world information whenever he makes a modification to his structure. Usually the real world modification information provided is adequate for other human beings; the author's system attempts to ensure that it is also adequate for the machine.

This basic concept (the provision of a transformation between the problem representation and the diagrammatic representation) is applicable to a large number of areas which use sketches or diagrams, and underlines all of the more detailed concepts developed in the writer's model, thus indicating that, provided they have not been formulated in a specific manner, they should be applicable to many systems.

To assist in the consideration of other domains in the later sections of this chapter, some of these general features are now collected and discussed together.

In this communication model the machine's knowledge of a particular situation is divided into two completely separate structures. While this in itself is not completely novel, the information contained in the pictorial structure is of a more complex nature than in other systems, containing as it does such information as logical groupings within the pictorial structure, the existence of psychologically perceived relationships such as nearness, adjacency, collinearity and so forth. The facts that these relationships have been considered by psychologists to be fundamental to the human being's mechanism for structuring pictures, and that these relationships are used to convey certain meanings in the problem domain, indicate that this form of pictorial structure should be of broad applicability.

The exact nature of the pictorial relationships as perceived by human observers has been the subject of much research, but the problem has yet to be adequately unravelled (Macleod, 1970). The present system uses a simple approach based on the size of the objects related. This preserves a necessary relativity, and is sufficient for the purpose required because diagrams are not intended to be compared with real life pictures - that is, the degree of a relationship such as "adjacent" is not as important as the observation that the relationship exists in a diagram. Consequently the approach used here for pictorial relationships will be satisfactory for many domains.

The problem domain structures used are in the form of multi-level trees, and a particular feature of these structures is that the higher level objects are each categorized in a manner which provides the machine with an indication as to the purpose of that particular object. These categorisations provide global as well as local information on the nature of the problem structure, and may be used outside the "communicator" to provide assistance in processing the problem structure. In the communicator itself, this information is used to assist in the construction of diagrammati representations of the structure.

The form of the structure used is applicable outside the circuit domain, for the functional information in other fields is used in a similar manner for the production of diagrams. This is demonstrated in the examples used in later sections of this chapter. Even if the structure is essentially one level in nature, the present scheme is still applicable as it is just a special case of the many-level structures which have been used. The necessary information is then contained within the relationships between the parts.

It has been stated herein that the communicator must contain some knowledge of the domain of operation in order to effectively shorten the human user's communication task, and to produce the diagrams corresponding to the current problem structure. The communication model contains this information in two ways

1) As indicated earlier, the aesthetic knowledge in the relation subroutines provided is based on certain psychological principles, and consequently will find application in other domains. The mapping procedures based on these relations are therefore of a generally useful nature.

2) Knowledge is contained in the set of models, which are application specific. Each model represents the conventions for representing a particular application object, but the format of these models is not application dependent, involving as it does

the pictorial arrangement of primitives (whose exact nature is irrelevant) and the application of various constraints and relations between them. These are the same relations which were indicated previously in this section as being pictorial phenomena and not application dependent phenomena.

The mapping procedure, which produces diagrams from problem structures, is dependent heavily on the existence of the above models. The communication model proposed in this thesis can be easily adapted for use in other areas if a suitable set of models can be devised for the new area. Even if such models cannot be found, the communication model may still be useful as it is unlikely that there will not be a set of problem relationships which are represented in some specific manner in the pictorial domain. The system will then rely largely on its attempts to produce aesthetic layouts using the relation routines, along with the layout requirements of the problem relations.

Bearing the above discussion in mind, the remaining sections of this chapter point to the usefulness of the communicator model to various areas by examining the existence and format of the necessary models. To illustrate the general use of the models, some of the areas covered are outside the subclass of section 4.3 (that is, the subclass of domains using abstracted sketches or diagrams), despite the fact that the model was originally intended solely for that subclass.

10.3 Other Electrical Circuit Systems

In building a demonstrative communication system for electrical circuit diagrams, a particular environment, namely design, was kept foremost in mind. The general communicator configuration, as developed in section 4.5, was restricted for this environment because of the limitations in the functions required. A principal aim of the designer is to get a circuit description into the machine for analysis and subsequent manipulation. The diagram is a subsidiary entity, providing convenience and a thinking aid for the designer. The design environment, therefore, need not allow for the construction of the diagram by the user, and the uni-directional transformations of the developed system are sufficient.

This restriction need not be as severe as it at first seems as a large proportion of current applications are of the design type. Nevertheless this section looks at extensions to remove the limitation.

Consider once again the computer aided teaching environment. The modes of communication which may be desirable in this situation are (as indicated in section 6.5):

1) The machine transmits circuit information to the student. This may be required in answer to a student's question, or as part of a question posed by the machine to the student. The present system is able to handle such data by a simple transfer of information from teaching programs to the student.

2) The student transmits circuit information to the machine. This is required in similar situations to the above.

3) The machine may display diagrams to the student. This again is useful in posing and answering questions, and is within the capabilities of the present system. Other possibilities also occur, however. The teaching system may wish to test a student's ability at manipulating circuits. An example such as Figure 7-7 (Page 107) is useful here. When the communicator is asked to display a circuit consisting of a divider with two taps, the student is presented with Figure 7-7(a). If the student fails in the analysis, the "teacher" may request a bridge circuit to be displayed to assist the student. Fugure 7-7(b) will then appear. The present system is capable of this mode of operation.

4) The student may wish to draw a circuit for the "teacher". This is useful in answer to a request, or as part of a question which the student wishes to pose. In either case the student's understanding of the circuit will be reflected in part by the way the diagram is drawn. The communicator would be expected to produce a circuit description and pass it on to the "teacher". The system is, as yet, not capable of this function.

To incorporate the latter facility, it is not sufficient to recognize each of the components and their interconnections, and to form these into a network. Rather, the machine must be able to discover the various pictorial relationships present, and make appropriate implications about the high-level circuit structure. A suggested technique for this purpose uses the models already present in the system. The pictorial description representative of the input diagram is first produced, involving the recovery of the pictorial relationships present between the various pictorial symbols. A search must then be initiated amongst the known models in an attempt to match various portions of the structure against the models. A complete match would indicate that the appropriate circuit type was intended by the drawer. In the case of an imperfect match, the nearest match should be determined, and the circuit specified as modifications thereof. The problem of finding a given structure within a large structure, although not trivial, is not impossible. Macleod (1970) discusses the problem of describing an image in terms of variations on another image. It is believed that such an approach may be fruitful in the attempt to develop the pictorial description to circuit description transformation.

Once this transformation is developed, the full communicator model of Figure 5-1(a) (Page 70) can be programmed, and the system could then operate in many environments. The data base environment (discussed in chapter 4) provides a further environment in which the system, with the above mentioned extension, is capable of operating.

10.4 Diagrammatic Representation of Logic Networks

This domain can be considered to be closely related to the electrical circuit domain. There are, however, a number of important differences which imply that this must be considered as a distinct and different domain.

The primitive application objects here may be taken as AND gates, OR gates, and inverters. Other objects in the domain such as NAND gates may be considered in terms of these basic elements, despite the fact that they are frequently drawn with their own special symbol - i.e. as if they were primitives. This feature is discussed presently.

The basic application relationship is once again connectivity - connectivity appears as a relationship in many domains and in most cases it is represented in the same manner in diagrams; consequently the pictorial "join" relation is important. The meaning of "connectivity" may vary between domains, but in logic networks it has a similar meaning to the electrical circuit connectivity.

The pictorial primitives for logic networks are symbols representing AND gates OR gates and inverters, plus line segments. There must be other pictorial elements, however, as it is nowadays common, particularly since the advent of medium and large scale integration, to represent certain higher level problem domain objects by a single symbol in the diagram. In fact it becomes necessary to have a pictorial symbol for each of the higher level objects defined by a model within the system. These symbols are involved by allowing the network-to-diagram transformation to proceed to a certain depth in the problem structure and then stop. Those pictorial elements at the bottom of the resultant pictorial structure are displayed according to their corresponding symbols. This mechanism allows for the diagram produced to contain elements of widely varying complexity, the criterion for the expansion of non-expansion of a given item being the importance it has been given in the network description. The importance is manifested by the depth in the structure of the element. This mechanism is important as it is the mechanism used in practice. e.g. a single AND gate may control the enabling of an entire Arithmetic and Logic Unit, and it would be necessary to display the AND gate alongside a box containing the complete Arithmetic Unit. Any attempt to perform the transformation by specifying that the process is to continue until objects of a given complexity only are present wo-ld be unsatisfactory in these circumstances.

The suggested transformation constraint is easily incorporated into the communication model as it is now defined, so that this peculiarity of logic networks presents no problems to the system.

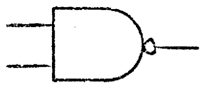
The problem domain objects defined within the system are quite readily obtained. These can be a set of standard logic modules in fairly common usage - including such items as decoders, multiplexers, parity modules, dual rank flip-flops of various types, registers, counters, adders, and so on. Possible configurations for each of these in terms of lower level objects may be produced, and these can be used to form the necessary set of models in exactly the same way as was done with electrical circuits. Two simple examples are illustrated in Figure 10-1.

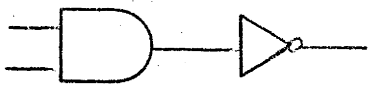
The conflict between d-c information and signal information, which exists for circuit diagrams, does not affect logic networks, as only signal information is represented therein. This alleviates some of the difficulties found with circuit diagrams, and may indicate that a system for logic diagrams could handle entirely new networks more effectively. The same rules regarding proximity in the diagram and sequence in the problem domain apply, and therefore there are no real difficulties involved in using the present system, with new models and primitives, in this domain.

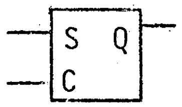
The possibilities for a teaching system on the above lines are currently being examined by the author. Such a system would be directed to second semester students who are taught logic network theory along just those modular lines.

10.5 Flowchart Systems

Another example of symbolic diagrams to which the present system is applicable is program flow charts. In this domain primitive symbols such as rectangles, parallelograms, diamonds and circles represent functional blocks such as computation, input/output, decision making and beginning or end points of procedures. In the problem domaim, each of those functional blocks has some corresponding function, and this function is represented pictorially as text within (or associated with) the symbol. This function corresponds to the component value in the circuit diagram domain, and the text corresponds to the component value labels in a diagram; both should be represented as attributes of their appropriate objects in the descriptions. The mechanism for including such attributes has not been included in the illustrative circuit diagram system developed,


(a) A NAND gate symbol


(b) Corresponding lower level representation



(c) A dual rank S-C flip flop symbol

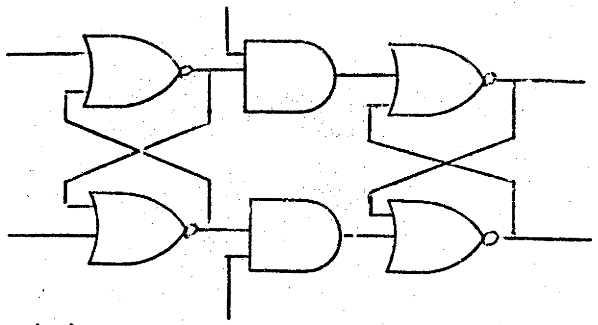

(d) Corresponding lower level representation

Figure 10-1 Some logic objects and the diagrams corresponding to their models.

but it is more important for flow charts; the mechanism for the inclusion of such information, and its placement on resultant diagrams, has been indicated in section 6.6.

The major problem domain relation present in flow charts is that of program flow from block to block, and is normally represented in the pictorial domain by an "above-below" relationship, in which pictorial adjacency is used to associate functional blocks which are normally performed one after the other. As with electrical connectivity, the exact flow of program, indicated by the connecting line segments, is used for confirmation, except in cases where the chart is not conventionally laid out.

Program loops represent a problem corresponding to feedback in electrical circuits. In contrast to the latter, however, there is usually only one way to represent the loop. This is done by developing a path "travelling" up the page, and either to the left or right of the main "stream". The information concerning the existence of loops is already imbedded within the connectivity of blocks, but as with feedback in circuit diagrams, loops represent a higher level of functional organisation and this should be represented explicitly in the problem domain structure. Given that such information is included in the problem description, then it may be seen that the present system has the tools required to arrange the blocks in the diagram according to the above criteria. The loop may be represented as a relationship amongst several objects or blocks. When this relationship is encountered during the mapping process, the appropriate pictorial relationships are set up, and it is the task of the existing pictorial relationship procedures to determine the detailed pictorial layout.

It may be noted that the hierarchical depth of the structures does not at first sight appear to be as great as that for circuits and logic networks. If this were so, it would simply mean that less emphasis is placed on a large set of models, and more is placed on the relational procedures. In many instances, however, there can be a considerable depth in the problem structure. If a reasonably large system is programmed using the desirable practices of developing many modules of increasing complexity and combining these into a readily understood overall system, then the depth within the structure may be considerable. Each of the program modules is represented by an object in the program structure, and invocations of the module appear as instances of the structural object. As with logic networks, it is feasible to allow partial transformations, stopping at a given depth, and allowing various degrees of detail to be displayed.

Despite a reasonable complexity in the problem structures, there is not necessarily a large number of models required. Each module is really only a different instance of the general program model "procedure module". When these objects are to be displayed as the lowest level in a diagram, they may be displayed as one of the basic block types, the choice depending on whether the main purpose of the module is computational, Input/output, and so on. This corresponds to the definition of symbols for high level objects in logic network diagrams.

This domain thus may be treated similarly to circuits and logic networks, and fits within the capabilities of the general model proposed in this work.

10.6 Landscape Planning and Environmental Architecture

These two domains are not infact distinct, as they are manifestations of the same domain at a different level. Since they have differing objects of interest, however, they are considered separately here, following an outline of their common features. This discussion considers landscape planning at the level of the design of layouts for small areas, in which individual objects such as trees are each considered. In environmental architecture it is assumed that the area of concern is somewhat larger, and detail required may not go below specifying that a block of home units is placed in a certain area.

These domains are somewhat different from those considered previously, as the representations to be displayed must bear a closer relationship to the real situation - spatial relationships in the diagram correspond to spatial relationships within the domain. While it would appear that it is unnecessary to maintain separate descriptions of the situation, there is some advantage to be gained from continuing with a pictorial and a problem structure.

The diagram that is displayed for the user as he proceeds with designing a garden layout, for example, contains symbolic representations of the items he is manipulating, and it is necessary to get across the spatial relationships which may exist in the final area. Since symbols commonly used can differ considerably in appearance from the real object, the spatial relationship is not necessarily manifested in the same way (A tree and a shrub near a fence may be equidistant from the fence in the diagram because of the nature of the symbols used, but in reality, since the tree is much larger than the shrub, it will have to be planted further from the fence).

A hierarchy of objects may be set up similarly to the hierarchie used previously. Objects such as a tree screen (made up of a group of appropriately spaced trees), a barbeque area (made up of a barbeque, concreted area, seats, windbreak, table) are problem domain objects useable in landscape design for small areas. By setting up a hierarchy it is possible to readily move groups of objects about in the design in order to produce required effects. Each of the objects mentioned above can be set up as a model within the system in order to give it the knowledge to construct situations in detail from small amounts of high level information.

While the use of this system in small scale landscape design may be too expensive at present, the requirement for professional advice in this area is rapidly growing, and professional consultants might be able to use this approach economically in the future.

The large scale manifestation of this domain, here called environmental architecture, is an area where the economics are more favourable. Large developments are the subject of intensive environmental design studies, and a system based on the above principles would be of assistance. The principles outlined above are equally applicable; all that needs changing are the particular objects involved. For an urban environment one can define apartment blocks, skyscrapers, open pedestrian areas, etc. On top of this, blocks of skyscrapers and so on, can be defined, leading ultimately to complete large scale development objects. Size of the various components is of particular relevance and this is included readily as an attribute. Particular spatial relationships which may be required include insuring adequate sunlight filtering between complexes, and this form of relationship, which can be mapped into a combination of the existing pictorial relationships for diagrammatic purposes, can be included in the problem structure. Modifying items in the structure is then performed within the constraints of those relationships. Models can be defined for various types of area to be found, and these used to assist in the rapid specification of a structure. As well as the production of a plan of the structure as currently represented, these systems may be required to produce "views" of the area from various positions. Since a complete problem description (not just the plan) is maintained, it is possible to generate a three dimensional model, projected onto the display screen. This again requires knowledge in models of the typical actual appearance of various objects.

These two areas, then, can make use of many of the features of the present communicator model, provided a number of provisions are made to allow for these domains not fitting exactly within the original premises.

10.7 Engineering Drawings

The domains considered in this and the following section are not strictly of the symbolic diagram type, and it is not expected that they should be directly amenable to solution using the model proposed. They are considered here, however, to indicate that some of the principles involved in the model are not restricted in applicability purely to the area originally considered.

Engineering drawings do not fall into the symbolic diagram class as they involve a projection of a three dimensional object, and this can be obtained by a direct transformation. It is not clear, then, that two descriptions are required. For convenience sake a three dimensional description and a display description may be kept, but it is on the three dimensional structure that attention may be focussed.

In any particular field in which engineering drawings are to be produced, it may be expected that certain types of part, and certain arrangements of parts, are common. These parts and their arrangements may be the subject of models which contain three dimensional information relating the parts to one another. The production of a drawing may then involve the generation of a three dimensional description using the models, and then obtaining the two dimensional diagram therefrom. It must be stressed that one could not expect a completely general drawing system, as not all models for every purpose could be included. Rather models for a particular environment would be developed to ease specification time.

The contribution to this field is seen as the provision of machine knowledge in the form of models for typical arrangements, to assist the user in the development of his drawing.

10.8 Printed Board Layout

Layout of printed boards (and similarly micro circuit design) represents a somewhat different problem again. No major advantage is seen in the maintenance of two distinct descriptions, as the picture displayed exactly resembles the problem domain object. The problem involved in this domain is the placement of components and leads on boards so that path lengths, number of crossovers, and (for double sided boards) number of through connections, are minimized. The constraints here are much more rigorously defined than for circuit diagrams, and are of a different nature. Minimum path length is amenable to some mathematical treatment, but ingenious search remains the key to success. Current programs are inefficient unless aided by human intervention. Previous experience is an important part of this ingenuity, and is amenable to incorporation in the machine. Many circuit arrangements are commonly encountered, and may be laid out similarly each time. Models representative of commonly occurring situations are seen as a feature of the present system for circuit diagrams, which may be applicable to the board layout problem, thereby improving the performance of such programs, both when unaided or aided by a human operator.

Improving performance of programs in this domain would be of great benefit as the number of workshops and laboratories involved in board layout throughout the world is large.

10.9 Summary

This chapter has discussed the communication model with regard to its applicability to other domains, and has discussed how it may be applied to a number of particular fields. The main features of the system which are applicable to other classes of problems may be summarized as follows:-

(1) There are two separate structures, which are generated separately (not generated concurrently as in other comparable systems).

(2) The pictorial structure used is based on the existence of psychologically perceived relationships which are of general applicability, rather than solely on linking together sets of coordinates of objects.

(3) For the class of problems of prime interest, the diagram is an abstract representation of a situation, and the occurrence of the above-mentioned pictorial relationship in a diagram is of major importance, rather than the degree of the relationship. Consequently fairly simple definitions of the relationships may be used to cover a wide range of domains.

(4) The problem domain structures explicitly contain higher level functional information. This is required in many domains because the purpose of a structure is not always deducible from the basic interconnections, and the purpose or function of a structure often governs the way a diagram should be laid out.

(5) The basic principles of the mapping procedure are not specific to circuit diagrams. Rather they depend on the existence of procedures for interpreting the above-mentioned pictorial relationships, and on the existence of a set of models indicating the layout of a number of common constructs in the problem domain.

Some fields will require routines for problem specific relations, but these may be written to interpret the problem relation in pictorial relation terms, and the procedures for handling the latter are general and have already been provided.

(6) The models required are different for different domains; but are constructed in a similar way for all domains: they are in the form of an exemplary pictorial layout for the object in question, using the supplied pictorial relationship tools common to many domains.

(7) The exact nature of the pictorial and problem primitives does not affect the communication model. For each problem primitive there will usually be a corresponding pictorial symbol whose exact structure is known only to the final display package. The pictorial primitive's attributes will include size, orientation, position and method of interconnection in the diagram and will be common to many fields. Problem domain attributes will vary, but only those involved in linking objects into the problem structure are used by the communicator, the rest being present solely for the use of any analysis system to which the communicator is attached.

CHAPTER 11
DISCUSSION

11.1 Introduction

The research reported here has attempted to identify and investigate solutions to some of the problems involved in interactive computer graphics - a system has been developed to this end. The work is reviewed in relation to its own merits as well as in relation to other work in the field. Use of the system (or modifications thereof) in other environments and domains is discussed, and aspects of generality of the principles involved are brought out. A discussion of possible ways of extending and improving the system, leads to recommendations for further research.

11.2 Review of Project

The initial investigations, which provided the background to the development of this project, examined the field of interactive computer graphics within the general framework of Artificial Intelligence and its subfields. It has become apparent that the various subfields have become, to some extent, independent areas of research, with less than optimal interchange of ideas and techniques between them. A number of deficiencies in graphic systems, as previously discussed in sections 2.5 and 4.3, is apparent, including:

A) "Graphical" communication is usually not really graphical in nature - graphical input into a computer is usually achieved by a series of "light pen commands" to place various symbols; such communication is slow, and in a "discourse language" rather than in a graphical language.

B) No attempt is made to assist the user in the communication task by the incorporation of knowledge of the environment into the programs. Examples (such as ~~ELIZA~~ STUDENT) in natural language communication indicate that a small amount of "environment" can produce large improvements in communication performance.

C) No distinction is made between the pictorial domain and the problem domain. Communication tends to be in statements involving a mixture of both. Research in picture processing, particularly the parsing approach, has indicated that these two domains can profitably be considered as separate, with a means of relating one to another.

This project has attempted to remedy some of these deficiencies and so improve graphical communication systems. The important novel features of the present communication system are:

(i) On choosing a specific problem domain in which to work, an effort has been made to incorporate some knowledge of this domain into the machine. This information is used to relieve the human user of some of the labour involved in communicating his problem to the computer.

(ii) Communication between man and machine has been separated into two parts, one purely pictorial, the other purely problem-oriented. This allows the user to specify only that information which needs to be communicated, and it may be couched in appropriate terms.

The approach taken in the present research towards the deficiency mentioned in (A) above has been that the question of general pictorial communication and pictorial languages is one which will not be solved in the short term; consequently effort has been directed to improve the ease of communication by means of the methods introduced in (ii).

Once the overall objectives of the project were established it was necessary to consider the individual problems involved in the development of such a system. The establishment of the requirements of the system revealed problems in 3 areas.

a) Because of the requirements for free communication, the system should use a method for the representation of circuits which mirrors the view taken by the human user, thereby allowing the user to communicate in the terms in which he thinks of the circuit. Providing such a system necessitated an investigation of human representations of circuits, as discussed in Chapter 6.

b) An independent pictorial description must be maintained in the system. This description must not simply be in terms of the set of line segments in the diagram, but must represent all the forms of relation and structure which the human observer is able to utilize in interpreting a circuit diagram. The nature of this information is not obvious, and an investigation of the pictorial properties of circuit diagrams was necessary (as discussed in Chapter 7) in order to formulate adequately a pictorial description.

c) Once the two descriptions were established, a method by which the computer could generate a circuit diagram description from the circuit description, was necessary: to achieve this the machine needs more information than the class of relations involved in a diagram. Some precise knowledge is required on the connection between a given circuit description and the corresponding diagram. Chapter 8 investigated this and developed a method of mapping with models.

Based on the principles outlined above, a system (developed in Chapter 9) has been programmed. Despite the experimental nature of these programs, they have achieved many of the initial objectives of the project. Significant features of the developed programs include:

1) The user is completely relieved from the task of generating a circuit by the "pick-and-place" method. If a light pen is used as the sole input medium, the process employed in other systems is reversed, as the generation of circuit and diagram starts at the top level and works down.

2) The user does not need to specify every piece of interconnection information needed to form the circuit. He need only specify the type of circuit and the parts of that circuit. The machine then uses its own "knowledge" to construct the required interconnections.

3) The user need not specify his circuit step by step as it is possible to input a complete circuit on the keyboard in one step - this facility relies heavily on the machine's knowledge of

circuit organization as mentioned in 2) above. Without such a knowledge in the machine, specification of a complete circuit would be very cumbersome (section 6.7).

4) The communication channels are separated into problem and pictorial communication, as was set out in requirement (a) above. In the present system the information communicated by the user to the machine is entirely problem oriented, whereas the machine communicates pictorial information to the user.

11.3 Comparison with Other Work

To the writer's knowledge, no other systems presently available place emphasis on the same problems. In comparing the present work with that of others, therefore, program comparison may conveniently be on the basis of ease of communication for the user.

Of the systems described in Chapter 3 as representative of different approaches to interactive graphics, the programs of Bracchi and Somalvico bear least resemblance to the present work. While their system aims at use in a specific problem domain, no clear attempt is made to assist the user in communication in this domain. Circuits are developed and drawn by generating a portion of program which is later executed and, at that time, produces the required circuit and diagram. The user is therefore at a disadvantage, as he cannot detect mistakes at the time they are made, and this negates many of the advantages of the interactive environment. By contrast, the present programs maintain a continuous indication of the currently held circuit, thereby allowing immediate recognition of errors made either by man or by machine. The language used for communication by Bracchi and Somalvico is, despite its modifications, simply a variation of Fortran. As such it does not represent a suitably natural communication medium. The present programs attempt to provide more suitable input terminology by the use of circuit entities closer to those in terms of which the user thinks.

The DESIGNPAD system, in contrast to that of Bracchi and Somalvico, attempts to be a general graphics program, and not a specific problem domain program. One of the DESIGNPAD objectives was the achievement of the ease of communication via sketching which human beings achieve. DESIGNPAD does not, however, allow the user to sketch diagrams, but rather, enforces the "pick-and-place" method. The system makes no attempt to provide alternative aids to circumvent this problem. The writer's system, however, while acknowledging that any method other than sketching is perhaps not ideal, attempts to alleviate the situation by the incorporation of some knowledge of the problem domain. The writer's system then behaves rather more like an "intelligent" communicator, in that it is able to make a number of assumptions concerning the input data, and is able to perform a number of the tasks which would otherwise fall on the user. DESIGNPAD is unable to achieve this because of its aim at complete generality, and therefore cannot readily contain the necessary problem domain knowledge.

The AEDNET system of Evans and Katzenelson is a compromise between the system of Bracchi and Somalvico and DESIGNPAD; natural communication is more satisfactorily achieved and operation is in a specific problem domain. Since a single data structure is used, it is difficult to separate the graphic and electrical aspects of the various commands. The existence of knowledge in the program is limited to the use of certain names which are the same as those used by human beings. For example, a command to create a node appears to be a natural command until it is realized that this command involves the creation of the electrical entity in the data structure and the creation of a point in the display at a position specified by the user. The knowledge is therefore limited to the existence of the set of problem oriented primitives. No knowledge exists as to the organization of these primitives either electrically or pictorially.

A useful feature of the AEDNET system is its ability to construct a portion of the circuit and to use this portion as a block in further manipulations. This, in part, allows the user to omit redundant data, a property which is incorporated in the present writer's system, but in AEDNET it does not represent a true knowledge of circuit diagrams because AEDNET is not capable of building the block into a diagram automatically, nor is it capable of performing any major changes within the block.

On comparing other existing systems (Thomas, 1967; Robbins and Beyer, 1970; Schwinn, 1967; Kulsrud, 1968; Bracchi and Ferrari, 1969; Kuo et al, 1969) with the system presented in this report, it may be noted that similar deficiencies to those already discussed for Bracchi and Somalvico, DESIGNPAD and AEDNET exist, and the various comparative remarks made above apply essentially also to these other existing systems. No existing system attempts the incorporation of some form of knowledge to assist the user in his diagram production as does the present communicator.

11.4 Program Results

The operation of the present programs may be examined on two levels: the actual diagrams produced, and the functional capabilities of the whole system.

The responsibility for aesthetic arrangement of diagrams lies (as previously outlined in Chapter 8) in two areas: the models, and the relation procedures. In the first instance it is the duty of the models to specify the pictorial organization necessary in a diagram, and this specification may be altered reasonably readily at any time, to obtain an alternative organization for a particular circuit type. The task of detailed organization and placement within a diagram is the responsibility of the relation subroutines, as they must specify exact positional information for symbols and lines. As a simple example of this responsibility, an "adjacent" relation exists between symbols in a parallel impedor. The values of the maximum and minimum coordinates of the symbols are constrained by the "adjacent" subroutine, to be such that the rectangles enclosed by those coordinates are contiguous. The symbols eventually drawn by the drawing package do not completely fill the rectangles; the symbols are therefore adequately separated, but are sufficiently close to be associated with one another. If a programmer, with tastes other than those of the writer, wishes to modify the aesthetic arrangement produced by the above procedure, he may do so in one of two ways. The first is to modify the size and shape of the symbol drawn within the rectangle; and the second is to modify the precise constraints set up by the relation subroutine. Both of these tasks involve only small modifications to existing routines, and are not difficult to effect. Similar comments apply to other aesthetic procedures produced by the program: the exact arrangements produced represent the writer's taste and may be readily modified.

Turning to the functional capabilities of the system, it may be seen that, as indicated in section 5.3, the communication is electrical in nature between man and machine, but pictorial in nature in the reverse direction. The electrical communication involves the generation of circuits according to models, and the addition to, and deletion of parts from, the circuit, thereby allowing fairly convenient development of any particular circuit. The necessity for more-extended electrical communication may be found in some environments, and this necessity is discussed in the following two sections where other functional extensions, such as specification of modifications to pictorial arrangements, are also discussed.

11.5 Environment and Domain Dependence

The early chapters of the thesis outlined a communication model for a class of domains. An example of this has been discussed in detail. The question of whether the system can be used elsewhere was discussed in chapter 10, where it was shown that the system can be applied almost directly to other domains within the symbolic representation subclass. The concept of the incorporation of knowledge as models, in order to simplify the communication task of a user, was also shown to be applicable in a range of other domain types.

These discussions indicated that the model should not be considered as applicable to circuit diagrams only, but rather as the contribution of a fairly generally useable model for interactive computer graphic systems.

11.6 Extensions and Recommendations for further Work

Throughout this report, various areas have arisen as possible extensions to the present system, or as ideas for developing more elaborate systems. Some of the more important of these are listed, together with other possibilities which have not been discussed previously.

11.6.1 More-Extensive Models

The most obvious extension which arises is the provision of a more extensive set of models. While this will certainly improve the performance of the programs, it should not be considered as an extension of the features of the system, but rather, more examples of the features will be available. This extension is desirable for a working system but is not necessary for continued experimentation.

11.6.2 Enhancement of Mapping for Non-Conforming Circuits

Improved ability of the program to handle circuits which do not exactly conform to the layout of a model will also enhance performance. At present the system can handle changes only in which the new component connections are close to those of the model. A more general ability to fit a component into an existing circuit would be desirable, but poses several problems.

Although it is easy to identify the nodes between which the component must be connected, and to determine, from the pictorial description, which objects (i.e. terminals on components or line segments) to which the symbol must be joined, the problem exists as to whether or not the component will fit directly into the diagram. This requires a search amongst the neighbouring pictorial objects, constraints and relations. If the symbol will fit, it is easy to add it to the description. If the symbol will not fit directly, the system may do one of two things:-

(1) Attempt to see if the circuit can be "stretched" or "squashed" to fit the component or

(2) see if additions can be made to fit the new component.

(1) involves inspection of the relations to see if changes in these will allow the symbol to fit. For example, if two symbols are adjacent, and the new symbol must fit between them, then the adjacent relation must be removed and replaced by two adjacent relations between the new symbol and each of the others: even this is not always possible - it may happen that the connection points are so placed that there is no path between them that does not cross other symbols or lines. Then the problem becomes one of determining a route for the connection which will cause least interference with the remaining circuit. Some modifications may also be necessary to the rest of the diagram.

At present the programs can handle only a simple form of this problem. Components must be so placed that they are approximately in their original position, and some simple line insertions can be made, as shown in Figure 9-6. Further work in this area is expected to be fruitful.

11.6.3 Feedback Interconnections

Feedback interconnections have been discussed briefly at several places in this report. This section brings together the points raised there, and discusses methods of solution to the problem of placing feedback components into a diagram.

In Section 6.3 (p.83) feedback (and signal flow in general) was indicated as being a major problem of the node-labelling approach to circuit description, as there was no mechanism for incorporating circuit function information into the description. An attempt to overcome this difficulty was discussed in Section 8.2 (p.111), in which a signal flow list was associated with the interconnection matrix. This method allowed some attempt to be made at distinguishing feed-back paths and placing the corresponding components appropriately in the diagram; however the whole approach was unsatisfactory due to the non-hierarchical nature of the descriptions, and due to the bottom-up mapping used (as indicated in these sections). An important point to come out of Section 8.2, however, is that the feedback information should be explicitly available in the circuit structure, and not buried within the interconnections, because feedback components are frequently placed differently in the diagram.

The application of the above principle to the final communication model, for the domain of flow charts, was discussed in Section 10.5. Flow chart loops were explicitly included by a special "feedback connection" form of the "connect" relationship. With this information available, the layout method indicated there allowed the mapping procedures to produce a flow chart for an algorithm containing loops.

An exactly analagous method applies to circuit diagrams. If a "feedback connection" relationship is made available, then exact knowledge of the components involved in feedback is available. Since mapping is "top-down" in nature, general positioning information is available when a "feedback connection" is encountered. Depending on the particular circumstances involved, one of two procedures may be followed:-

(1) If the interconnection is within a stage or between adjacent stages, then the methods for modifying a diagram discussed in Section 11.6.2 may be invoked.

(2) If the feedback is over a number of stages, it is unlikely that a suitable path would be found to make a direct connection.

(If the algorithms outlined in Section 11.6.2 were invoked, a path which weaved through the diagram would be produced and this is considered aesthetically unsatisfactory by most human observers.) The solution usually used in this circumstance is to take the feedback path above or below the rest of the diagram, allowing a straight path to be followed.

A decision for procedure (1) or (2) is made for each individual case, and the exact point at which the changeover occurs could profitably be considered with the modification problem discussed in Section 11.6.2.

An important point to note is that, while feedback involves a form of loop in the electronic sense, it need not involve circular structures in machine representations of the circuit. This is so because the major connective between objects in a structure is the "is a part of" connection. Objects which have feedback connections between them are not parts of one another, but are parts of some common higher level object. The electrical loop is manifested in the structure by the relationships between objects therein. The relationships are "hung onto" the basic structure. Feedback might be indicated by a combination of relations such as:-

A is connected to B, B is feedback connected to C, C is feedback connected to A - indicating a main path through A and B, with C as a feedback object. Relation loops such as this are already common within the structures and represent no new problem (a simple parallel impedor has 2 connections in what might be termed a loop).

The necessary further development required for handling feedback fully lies, then, in the routine to handle the "feedback connect" relation, and this routine should operate according to the principles outlined above.

11.6.4 Mapping from Pictorial Descriptions to Circuit Descriptions

Mapping from pictorial descriptions to circuit descriptions was discussed in chapter 10 where a possible method was indicated for providing this capability to allow operation in a greater range of environments.

11.6.5 Learning

One area which has not yet been considered is that of including learning in the system. There are several ways in which learning could possibly improve the system's potential. Since it has been suggested that the procedures developed in the mappings are principally concerned with constraints and relations which are quite general in their application, it would not be expected that learning could profitably be applied in this area because the necessary principles are already included. This is perhaps fortunate, for it is difficult, with current techniques, to build procedures capable of modifying themselves significantly.

The models, being in essence a form of data, are candidates for learning techniques. The models also represent the problem specific information in the system, and it is therefore desirable for methods to be devised for the system to learn new models. Either aided or unaided learning may be considered, but it is believed that for the present application, user assisted learning is more suitable. A user is then able to control the generation of models suited to his purpose, and can prevent the generation of unwanted models. If the program were to find groupings and generate models for itself, it might easily produce circuit objects which the user does not consider relevant. While the possibility of the machine presenting a new and improved viewpoint or model to the benefit of the user, is not ruled out, it is expected that other more important features should be first attended to in the system during its relatively early stages of development, and for these reasons assisted learning is preferable.

It is envisaged that the procedure for generating new models might be as follows: The user generates a circuit and diagram by a combination of the use of known models and the "pick-and-place" method. Once the diagram is complete, the system analyzes this for the significant pictorial relationships and constraints present. This would be greatly facilitated if the user has been able to make use of previous models, for significant circuit and pictorial structure will already be present. Once these relationships have been determined, they are combined into a model which is then added to the data base. In determining the important relationships, the program may make errors which will become apparent during operation, and some provision must be made for the user to correct these errors.

Some work has been done (Winston, 1970) on forming models of object types from raw data, but only in the field of line drawings of three dimensional objects. The indications from the above kind of work are that, with the assistance of the reverse transformation methods outlined above, a successful model learning technique could be developed.

Model learning need not be restricted to a particular domain. It is possible that, with the provision of a method for defining new primitives, models may be defined for other domains. The problem here is that the meaning of various relations to the problem domain must also be included: while this problem is difficult, it does not appear insuperable once model learning within a domain has been achieved.

11.6.6 Inclusion of Labels and Component Values

In Chapter 6, the omission of component values as attributes of circuit objects was justified on the grounds that the primary interest in this work has been in producing diagrams. In a working system, however, component values must be included. The mechanism for this has been indicated, but the inclusion also raises the question of placement of these values (and other labels) on the diagram. The question should be investigated. As a starting point, it is believed that the rectangles surrounding each symbol allow sufficient space for the placement of labels, and such a method could be readily implemented. If this does not turn out to be satisfactory, a more general investigation, using the "proximity" notion, must be undertaken.

11.6.7 Extended Problem Domain Knowledge

It is worth mentioning the inclusion of more elaborate problem domain knowledge as a possible area for fruitful development.

At present the communicator knows only sufficient to allow it to draw diagrams. It knows nothing about circuit analysis and function. Inclusion of more detailed knowledge of these areas may be of great assistance in both the problem to pictorial and pictorial to problem domain mappings already discussed. This knowledge would also be of great assistance in communication. For example, ultimately the user may be able to ask the machine for a high input impedance amplifier with a specified gain, and the system, with its more extensive knowledge of function, may suggest a circuit to the user. When this stage is reached, the distinction between the communicator and application programs has become vague because the communicator will "know" just as much as the application program.

Although this stage is in the future, the principle of improved problem knowledge remains applicable as an area for investigation. One effect of this approach, however, must be that the generality evidence for the present system, will become more and more difficult to maintain, as the problem domain data structures would tend to become more application dependent.

The work reported here has set out to provide a graphical communication system with a more extensive knowledge of the problem area than has previously been provided. This has been done, for the electrical circuit layout, as well as by the use of natural pictorial relations in the development of the diagrams. A major effect of this incorporation has been the provision of a convenient interaction between user and machine, in which the user needs only to specify that information considered absolutely necessary.

CHAPTER 12

CONCLUSIONS

The present study commenced with a survey of several fields of research, some of which are not directly related to the project. This broad survey has allowed a general perspective to be placed on the role of the work undertaken, and has indicated ideas from other fields which are relevant to Interactive Graphics, but have not been taken up therein previously. The review has also made it possible to evaluate currently available graphic systems in their own right, and in relation to the broader perspective indicated above.

A feature arising from the survey is that so-called graphical communication falls short of true pictorial communication from man to machine. Two reasons are evident for this: firstly, there is not sufficient knowledge at present on the fundamental nature of the organization involved in pictures; secondly, current graphical systems do not attempt to improve performance by the incorporation of "knowledge of the world" (or environment data) into the system.

The problem studied here arose from the above difficulties and the major objectives of the new system have been:

(1) To incorporate electrical circuit diagram knowledge into a communication system.

(2) To use this knowledge to draw circuit diagrams, thus removing this task from the user.

In developing the communication system according to the listed objectives, the major achievements which have emerged are:

(1) An examination of requirements for graphical communication systems, with special reference to "symbolic diagram" problems, has been made. This investigation has shown the various modes of communication necessary, and indicated that the notion of "relation of representation", which has been used by some workers in picture processing, but rarely by graphics workers, is of central importance.

(2) For the particular domain chosen for investigation (viz. electrical circuits and their associated diagrams), a detailed investigation of the requirements of the "relation of representation" notion has indicated that certain "knowledge" of the problem domain is necessary in the machine, and that this "knowledge" is in three parts.

(a) knowledge of the components and interconnections for particular circuit types,

(b) knowledge of the "aesthetic" requirements for a pleasing diagram,

(c) specific knowledge on how particular circuit types are usually drawn.

None of these three areas has been included as such in previous systems: part (c) has not been included at all; part (b) has been included only to the extent that certain circuits may be defined as blocks and used subsequently - no flexibility is allowed in this provision by other systems, as they are unable to make use of this "rigid block of data" in the manipulation of similar portions of circuit. The writer's system includes all three areas of "knowledge".

(3) A method for forming circuit descriptions has been developed, which takes account of the high level structure and relational information used by human beings in describing circuits, and forms circuit descriptions in similar terms.

(4) By investigating the circuit diagrams produced by human beings, it has been possible to determine a number of pictorial relationships which are important in aiding interpretation of diagrams. This information has led to the formation of pictorial descriptions for diagrams, which contain significant structural and relational data.

(5) A mapping between the circuit description and pictorial description has been developed; this mapping is the manifestation of the "relation of representation". The necessary knowledge is incorporated in the form of models of typical

circuit organizations and their pictorial arrangements. Procedures, based on the pictorial relations mentioned in (4), have also been developed which allow the system to handle circuits similar to, but not the same as, circuits represented by the models. This feature, which is not present in other systems, not only allows a reduction in the number of models necessary, but also represents "intelligent" use of data, known from related situations. Because of the existence of this mapping, it is not necessary to develop the two descriptions associated with a circuit simultaneously, and this has produced associated advantages indicated below.

(6) The use of the above mentioned models has allowed a considerable reduction in the amount of data which a user must supply to the machine in order to input a circuit of interest. This reduction is due to the fact that the models contain a considerable "knowledge" of the connectivity and component types in various circuits, and only that data not present in the models, or differing from them, must be specified.

(7) The provision of the mapping, frees a user from the burden of having to place each component or block in a diagram. This is performed by the developed system, using its "knowledge" of this (mapping) task.

(8) An experimental and illustrative system, based on the above principles, has been implemented. Despite its experimental nature, the system exhibits the anticipated reduction in necessary communication, by producing the drawing layout automatically, and gives satisfactory diagrams within its range.

(9) An examination of the principles used in the communication system has indicated that they are based on general notions of human comprehension of problem domain items, and on pictorial phenomena of a general nature. The communication model thus has applicability over a wide range of domains.

10) A discussion of the performance of the developed system has led to a number of possibilities for extension (for example to environments such as teaching and an "electronic handbook", and to domains such as logic networks, flow charts, engineering drawings and printed board layout) and improvement of the approach. A number of interesting extensions arising from the work has led to suggestions for further research (as listed in Chapter 11).

11) A major objective (see pp49-50) of shifting the communication burden from the graphical to the problem domain, thereby easing the communication task and making it more natural, has been achieved as outlined in points (1)-(10).

In conclusion, this work has indicated the feasibility of the incorporation of signi~~ficant~~ problem domain knowledge into graphical communication systems, and has illustrated the benefits to be obtained therefrom in an experimental set of programs, which have also demonstrated suitable techniques for the implementation and later extension of the approach. This work is thought to have great potential in practical systems applicable in a number of fields.

REFERENCES

ABEND, K., HARLEY, T.J., and KANAL, L.N. (1965) "Classification of binary random patterns"; Trans. IEEE, IT-11, 4, pp 538-544.

ABRAHAMS, P.W., et al. (1966) "The LISP programming language and system"; AFIPS, 29, pp 661-676.

ANDREW, A.M. (1969) "Pattern Recognition"; Physics Bull., 20, 8, pp 316-320.

AROYEN, G.F. (1959) "The technique of spatial filtering"; Proc. IRE, 47, 9, pp 1561-1568.

BARKDOLL, I.H. and McGLAMERY, B.L. (1968) "An on-line image processing system"; Proc. 23rd Nat. Conf. ACM, pp 705-716.

BARTER, C.J. (1970) "Data structure and question answering"; in Kaneff, 1970, pp 341-374.

BELADY, L.A., BLASGEN, M.W., EVANGELISTI, C.J. and TENNISON, R.D. (1971) "A computer graphics system for block diagram problems"; IBM Systems Journal, 10, 2, pp 143-161.

BENNETT, E., HAINES, E., and SUMMERS, J.E. (1965) "AESOP: A prototype for on-line user control of organizational data storage, retrieval and processing"; AFIPS, 27, pt 1, pp 435-456.

BERNSTEIN, A. and ROBERTS, M. (1958) "Computer vs. chess player"; Scientific American, 198, 6, pp 96-105.

BERNSTEIN, A. et al (1958) "A chess playing program for the IBM 704 computer"; Proc. WJCC, pp 157-189.

BLEDSOE, W.W. and BROWNING, I. (1959) "Pattern recognition and reading by machine"; Proc. EJCC, 16, pp 225-232.

BOBROW, D.G. (1964) "A question-answering system for high school algebra word problems"; AFIPS FJCC, 26, pt 1, pp 591-614.

BOBROW, D.G. (1970) "Natural language interaction systems"; in Kaneff (1970), pp 31-62.

BRACCHI, G. and FERRARI, D. (1969) "A graphic language for describing and manipulating two dimensional patterns"; Int. Rept. No. 69-35, Politecnico di Milano, Instituto di Elettrotecnica ed Elettronica.

BRACCHI, G. and SOMALVICO, M. (1970) "An interactive software system for computer aided design: An application to circuit project"; Comm. ACM, 13, 9, pp 537-545.

BUTLER, J.W., BUTLER, M.K. and STROUD, A. (1967) "Automatic classification of chromosomes - III"; Data Acquisition and Processing in Biology and Medicine, 5, pp 261-275.

CHARNIAK, E. (1969) "Computer solution of calculus word problems"; Proc. Int. Jt. Art. Intel. Conf., Washington D.C., pp 303-316.

CHENG, G.C., LEDLEY, R.S., POLLOCK, D.K., ROSENFELD, A. (eds) (1968) Pictorial Pattern Recognition, Thompson, Washington.

CHOMSKY, N. (1965) Aspects of the Theory of Syntax, MIT Press, Cambridge, Mass.

CLOWES, M.B. (1969a) "Transformational Grammars and the Organization of Pictures", in Graselli, 1969.

CLOWES, M.B. (1969b) "Pictorial relationships - a syntactic approach"; Machine Intelligence, 4, pp 361-383.

CLOWES, M.B. (1970) "Picture Syntax"; in Kaneff, 1970, pp 119-149.

CRAIG, J.A. et al (1966) "DEACON: Direct access English control"; Proc. FJCC, pp 365-380.

CUTLER, C.C. (ed) (1967) Special Issue on Redundancy Reduction, Proc. IEEE, 55, 3, pp 251-406.

DYE, M.S. (1969) "Character recognition by means of optical filtering"; Marconi Review, 32, 173, pp 111-128.

ELCOCK, E.W. and MURRAY, A.M. (1968) "Automatic description and recognition of board patterns in Go-Moku"; Machine Intelligence, 2, pp 75-88.

ENGLISH, W.K., ENGELHART, D.C., and BERMAN, M.L. (1967) "Display selection techniques for text manipulation"; Trans. IEEE, HFE-8, 1, pp 5-15.

EVANS, T.G. (1964) "A heuristic program to solve geometric analogy problems"; AFIPS, 25, pp 327-338.

EVANS, T.G. (1968) "A grammar controlled pattern analyser"; Preprint of IFIP Congress 68, pp H152-H157.

EVANS, D.S. and KATZENELSON, J. (1967) "Data structure and man-machine communication for network problems"; Proc. IEEE, 55, 7, pp 1135-1144.

EVANS, D. and VANDAM, A. (1968) "Data structure programming system" IFIPS Congress 68, pp 557-564.

FOGEL, L.J. et al (1966) Artificial Intelligence through Simulated Evolution, Wiley and Son.

GRASELLI, A. (ed) (1969) "Automatic interpretation and classification of images"; Academic Press, New York.

GREENBLATT, R.D., EASTLAKE, D.E.III, CROCKER, S.D. (1967) "The Greenblatt chess program"; AFIPS, 31, pp 801-810.

GRIMSDALE, R.L., SUMNER, F.H., TUNIS, C.J., KILBURN, T. (1959) "A system for the automatic recognition of patterns"; Proc. IEE, 106B, 26, pp 210-221.

GUZMAN, A. (1968) "Decomposition of a visual scene into three dimensional bodies"; AFIPS, 33, pt 1, pp 291-304.

HARLEY, T.J., KANAL, L.N. and RANDALL, N.C. (1968) "System considerations for automatic imagery screening"; in Cheng et al, pp 15-31.

HARLOW, H.F. and WOOLSEY, C.N. (eds) (1958) Biological and Biochemical Bases of Behaviour, Univ. of Wisconson Press, Madison.

HAWKINS, J.K., ELERDING, G.T., BIXBY, H.W. and HAWORTH, P.A. (1966) "Automatic shape detection for programmed terrain classification"; Paper XVI, Proc. Soc. Photo. Inst. Engnrs. (on Filmed Data and Computers), Boston.

HOLMES, W.S. (1966) "Automatic photointerpretation and target location"; Proc. IEEE, 54, 12, pp 1679-1686.

JOHNSON, C.I. (1968) "Principles of interactive systems"; IBM Systems Journal, 7, 3 and 4, pp 147-173.

JOHNSON, T.E. (1963) "Sketchpad III: A computer program for drawing in three dimensions"; AFIPS, 23, pp 347-353.

KAMENTSKY, L. (1961) "The simulation of three machines which read rows of handwritten arabic numbers"; Trans. IRE, EC-10, 4, pp 489-501.

KANAL, L.N. and RANDALL, N.C. (1964) "Recognition system design by statistical analysis"; Proc. 19th Nat. Conf. ACM, paper D2.5.

KANEFF, S. (ed) (1970) Picture Language Machines, Academic Press, London.

KISTER, J. et al (1957) "Experiments in Chess"; JACM, April, pp 174-177.

KULSRUD, H.E. (1968) "A general purpose graphic language"; Comm. ACM, 11, 4, pp 247-254.

KUO, F.F., MAGNUSON, W.G.Jr., WALSH, W.J. (1969) "Computer graphics in electronic design" Datamation, 15, 3, pp 71-79.

LAWRENCE, C.H. (1967) "Stereomat IV, automatic plotter"; Photogramm. Engng., 33, 4, pp 394-402.

LAWRENCE, T.J. (1972a) "Pictorial organization of electrical circuit diagrams"; Proc. of Workshop on Pictorial Organization and Shape, Div. Computing Research, CSIRO, Canberra.

LAWRENCE, T.J. (1972b) "Generation of circuit diagrams from electrical descriptions"; Accepted for publication in Computer Graphics and Image Processing.

LEDLEY, R.S., ROTOLO, L.S., GOLAB, T.J., JACOBSEN, J.D., GINSBERG, M.D. and WILSON, J.B. (1965) "FIDAC: Film input to digital automatic computer and associated syntax-directed pattern recognition programming system"; in Tippett et al, pp 591-613.

LEMMON, A. (1969) "QUASI-NET, a question answering system with input of natural English text"; Ph.D. thesis, Harvard Univ., Div. of Engrng. and App. Physics, Cambridge, Mass.

LEVINE, M.D. (1969) "Feature extraction: A survey"; Proc. IEEE, 57, 8, pp 1391-1407

LIPKIN, L.E., WATT, W.C. and KIRSCH, R.A. (1966) "The analysis, synthesis and description of biological images"; Ann. N.Y. Acad. Sci., 128, Art. 3, pp 984-1012.

McCARTHY, J. et al (1966) LISP 1.5 programmer's manual, MIT Press, Cambridge, Mass.

McCULLOUGH, W.S. and PITTS, W.H. (1943) "A logical calculus of the ideas immanent in nervous activity"; Bull. of Maths. and Biophysics, 5, p 115.

MACLEOD, I.D.G. (1970) "A study in automatic photo-interpretation"; Ph.D. thesis, Dept. Engrng. Physics, ANU.

MAXWELL, P.C. (1972) "The perception and description of line drawings by computer"; Ph.D. thesis in preparation, Dept. Engrng. Physics, ANU.

MILLER, W.F. and SHAW, A.C. (1968) "Linguistic methods in picture processing - A survey"; AFIPS, 33, pt. 1, pp 279-290.

MINSKY, M. (1969) Semantic Information Processing, MIT Press, Cambridge, Mass.

MINSKY, M. and PAPERT, S. (1969) Perceptrons, MIT Press, Cambridge, Mass.

NAGY, G. (1968) "State of the art in pattern recognition"; Proc. IEEE, 56, 5, pp 836-862.

NATHAN, R. (1968) "Picture enhancement for the moon, mars and man"; pp 239-266 in Cheng et al.

NARASIMHAN, R. (1966) "Syntax-directed interpretation of classes of pictures"; Comm. ACM, 9, 3, pp 166-173.

NARASIMHAN, R. (1969) "On the description, generation and recognition of classes of pictures"; in A. Grasselli.

NARASIMHAN, R. (1970) "Computer simulation of natural language behaviour" in S. Kaneff, pp 257-290.

NEWELL, A., SHAW, J.C., and SIMON, H. (1959) "Chess playing programs and the problem of complexity"; IBM J. of R. and D., 3, pp 211-229.

NILSSON, N.J. (1965) "Learning machines: Foundations of trainable pattern-classifying systems"; McGraw Hill, N.Y.

O'CALLAGHAN, J.F. (1971) "Graphical communication in interactive computer graphics"; Aust. Computer Journal, 3, pp 12-19.

PRATT, W.K. (1967) "A bibliography on TV bandwidth reduction studies"; Trans. IEEE, IT 13, 1, pp 114-115.

QUILLIAN, M.R. (1966) "Semantic Memory"; Ph.D. thesis, Carnegie-Mellon Univ., Pittsburgh, Pa.

QUILLIAN, M.R. (1969) "The teachable language comprehender"; Comm. ACM, 12, 8, pp 459-476.

RAPHAEL, B. (1964) "A computer program that 'understands'"; Proc. FJCC, pp 577-589.

ROBBINS, M.F. and BEYER, J.D. (1970) "An interactive computer system using graphical flow chart input"; Comm. ACM, 13, 2, pp 115-119.

ROBERTS, L.G. (1965) "Machine perception of three-dimensional solids" in Tippet et al, pp 159-197.

ROSENBLATT, F. (1958) "A probabilistic model for information storage and organization in the brain"; Psychological Review, 65, p 386.

ROSENBLATT, F. (1960) "Perceptron simulation experiments"; Proc. IRE, 48, 3, pp 301-309.

ROSENBLATT, F. (1962) Principles of Neurodynamics, Spartan Books, Washington D.C.

ROSENFELD, A. (1962) "Automatic recognition techniques applied to high information pictorial inputs"; IRE Conv. Recd., 10, pt. 4, pp 114-123.

ROSENFELD, A. (1968) "Picture processing by computer"; Tech. Rept. 68-71, Computer Sc. Centre, Univ. of Maryland.

SAMUEL, J. (1959) "Some studies in machine learning using the game of checkers"; IBM J. of R. and D., 3, pp 211-229

SAMUEL, A.L. (1967) "Some studies in machine learning using the game of checkers. II - recent progress"; IBM J. of R. and D., 11, 6, pp 601-617.

SCHROEDER, M.R. (1969) "Images from Computers"; IEEE Spectrum, 6, 3, pp 66-78.

SCHWINN, P.M. (1967) "A problem oriented graphic language"; Proc. ACM 22nd Nat. Conf., pp 471-477.

SEBESTYEN, G. (1963) "Adaptive memory search"; Tech. rept. AD427825, CFSTI.

SHAW, A.C. (1968) "The formal description and parsing of pictures"; Tech. Rept. CS-94, Comp. Sc. Dept., Stanford University.

STANTON, R.B. (1970a) "Computer Graphics"; Ph.D. thesis, Univ. N.S.W.

STANTON, R.B. (1970b) "Plane regions: A study in graphical communication" in Kaneff (ed) pp 151-188.

SUTHERLAND, I.E. (1963) "Sketch-pad: A man-machine graphical communication system"; AFIPS, 23, 329-346.

SUTHERLAND, W.R. (1965) "The Coral language and data structure"; Appendix C, Lincoln Lab. Tech. Rept. 405.

SUTHERLAND, W.R. (1966) "On-line graphical specification of computer procedures"; Technical Report 405, Lincoln Lab., MIT, Cambridge, Mass.

TIPPETT, J.T., BERLCOWITZ, D.A., CLAPP, L.C., KOESTEV, C.J., and VANDENDURGH, A.J. (Eds.) (1965) Optical and Electro-Optical Information Processing; MIT Press, Cambridge, Mass.

THOMAS, E.M. (1967) "GRASP: a graphic service program"; Proc. ACM 22nd Nat. Conf. pp 395-402.

TURING, A.M. (1950) "Computing Machinery and Intelligence"; Mind <u>59</u>, pp 433-460.

UHR, L. and VOSSLER, C. (1961) "A pattern-recognition program that generates, evaluates, and adjusts its own operators"; Proc. WJCC, <u>19</u>, pp 555-569.

UHR, L. (1963)""Pattern recognition" computers as models for form perception"; Psychol. Bull. <u>60</u>, 1, pp 40-73.

VANDER LUGT, A. (1964) "Signal detection by complex spatial filtering"; Trans. IRE, IT-10, 2, pp 139-145.

WEIZENBAUM, J. (1963) "Symmetric list processor"; Comm. ACM, <u>6</u>, 9, pp 524-544.

WEIZENBAUM, J. (1966) "ELIZA - a computer program for the study of natural language communication between man and machine"; Comm. ACM, <u>9</u>, 1, pp 36-45.

WIDROW, B. (1969) "Pattern recognition and adaptive control"; IEEE Trans. on Applications and Industry, <u>83</u>, 74, pp 269-277.

WILLIAMS, R. (1971) "A survey of data structures for computer graphics systems"; Computing Surveys, <u>3</u>, 1, pp 1-21.

WINSTON, P.H. (1970) "Learning structural descriptions from examples"; Ph.D. thesis, MIT.

WOODS, W.A. (1967) "Semantic interpretation of English questions on a structural data base": Mathematical Linguistics and Automatic Translation, Rept. NSF-17, prepared by Computing Lab., Harvard Univ., Cambridge, Mass.

WOOLRIDGE, D.E. (1963) The Machinery of the Brain, McGraw Hill.

Appendix 1 - COMPUTER PROGRAM LISTINGS

A number of the routines making up part of the system described in Chapter 9, are listed here. Due to space requirements only an illustrative sample is included here - low level structure manipulation, mapping procedures, and communication routines are illustrated.

The routines shown make considerable use of SLIP routines, and routines in a particular graphic display package. The purpose of each of the routines shown is listed below:

MAIN	main program, controlling sequencing
LUPOB	make a list of "up" objects of given object
LSUBOB	make a list of "down" objects of given object
NUOB	form a new object list
JUPDN	link two objects into the structure
LRELN	make a list of all relations of given object
CONSTR	form a new constraint for an attribute
GETRELN	get special list of relations
PROBJ	output lists associated with one object
PROBJS	output entire list structure
ADJAC	"adjacent" relation subroutine
JOIN	"join" relation subroutine
MODLIN	model routine for a line
MODCOP	copies a given model
MAPCCT	uses MODCOP on entire structure
MAKPIC	invokes relation routines and sets up constraints
NCOORD	traces constraint chains
GETCHAN	in generation mode, obtains a circuit expansion
RDCCT	in initialization mode, obtains a new circuit
CHOOSE	sets up light buttons
PICOUT	outputs picture data to drum for display package

```
      PROGRAM MAIN
      INTEGER LL(8000)
      COMMON /VISTRAN/ KA,KB,KC,LX,LY,LD
      COMMON /MODELS/NOMODS,IK(10)
      DATA(NOMODS=8)
      DATA (IK=4HQUIT,3HRST,2H R,2H L,2H C,3H ZS,3H ZP,3H PI)
      CALL TVREADY(4HMAIN,4)
      CALL MODARG(0)
      I=1
      CALL ERFIX
      IF(I.NE.1)GO TO 4
      I=2
5     CALL INITAS(LL,8000)
1     A=GETCHAN(CCT)
      IF(A.NE.0.)GO TO 10
      IF(MAPCCT(CCT,NUOB(PIC,0)).NE.0)GO TO 2
      IF(MAKPIC(PIC).NE.0)GO TO 2
      REWIND 55
      CALL PICOUT(PIC)
      REWIND 55
      CALL DRCCT
      CALL IROBJ(PIC)
      GO TO 1
10    IF(A.LT.0.)GO TO 3
2     IF(NAMTST(CCT).EQ.0)CALL IROBJ(CCT)
      GO TO 1
3     CALL VISTOP
4     CALL ERASEALL
      CALL CLEAR(LL,100)
      CALL MOVE(300,500)
      CALL VTEXT(13HERROR TRAPPED,13,3)
      CALL DISPLAY(LL)
      CALL BREAKOUT
      IF(KC.EQ.63)CALL VISTOP
      CALL ERASEALL
      GO TO 5
      END


      FUNCTION LUPOB(L)
      LUPOB=LIST(M)
      NUM=LDATT(L,2)
      DO 1 I=1,NUM
    1 CALL NEWROT(LDATB(L,1+NUM+LDATT(L,3)+LDATT(L,4)-I),M)
      RETURN
      END


      FUNCTION LSUBOB(L)
      LSUBOB=LIST(M)
      NUM=LDATT(L,3)
      DO 1 I=1,NUM
    1 CALL NEWROT(LDATB(L,1+NUM+LDATT(L,4)-I),M)
      RETURN
      END
      FUNCTION NUOB(L,ITYP)
      NUOB=LIST(L)
      CALL NEWBOT(ITYP,L)
      DO 1 I=1,3
    1 CALL NEWBOT(0,L)
      RETURN
      END


      SUBROUTINE JUPDN(L,M)
      CALL NXTLFT(M,MADNBT(L,LDATT(L,4)))
      CALL SUBST(1+LDATT(L,3),MADNTP(L,3))
      CALL NXTLFT(L,MADNBT(M,LDATT(M,4)+LDATT(M,3)))
      CALL SUBST(1+LDATT(M,2),MADNTP(M,2))
      RETURN
      END


      FUNCTION LRELN(L)
      LRELN=LIST(M)
      NUM=LDATT(L,4)
      DO 1 I=1,NUM
    1 CALL NEWBOT(LDATB(L,1+NUM-I),M)
      RETURN
      END


      SUBROUTINE CONSTR(I,L,J,M)
C
C     ADDS CONSTRAINT LIST M, OF TYPE J, TO THE ATTRIBUTE I OF THE OBJECT L
C
      INTEGER MES(3)
      DATA(MES=8HCONSTRAI,8HNT NOT A,5HDDED,)
      IF(IFATR(I,L).NE.0)GO TO 1
      NAM=ITSVAL(I,L)
      IF(NAMTST(NAM).EQ.0)GO TO 2
      CALL Q8QERROR(1,MES)
      RETURN
    1 CALL NEWVAL(I,LST(NAM),L)
      CALL NEWBOT(0,NAM)
    2 CALL SUBSTP(1+LDATT(NAM,1),NAM)
      CALL NEWBOT(J,NAM)
      CALL NEWBOT(M,NAM)
      RETURN
      END


      SUBROUTINE GETRELN(NAME,LALREL,LCNREL)
      INTEGER TOP
      LALREL=LRELN(NAME)
      N1=LSTLN(LALREL)
      CALL LIST(LCNREL)
      DO 1 K=1,N1
      IF(TOP(LDATT(LALREL,K)).NE.2)GO TO 1
      NAMJR=LDATT(LALREL,K)
      IF(LDATT(NAMJR,2).EQ.NAME)NAMOB=LDATT(NAMJR,3)
      IF(LDATT(NAMJR,3).EQ.NAME)NAMOB=LDATT(NAMJR,2)
      DO 2 K1=1,N1
      NAMR=LDATT(LALREL,K1)
      IF(LDATT(NAMR,2).EQ.NAMOB)CALL NEWBOT(NAMR,LCNREL)
    2 IF(LDATT(NAMR,3).EQ.NAMOB)CALL NEWBOT(NAMR,LCNREL)
    1 CONTINUE
      RETURN
      END



      SUBROUTINE PROBJ(L)
      INTEGER TOP
      PRINT 900
  900 FORMAT(*1 OBJECT STRUCTURE*)
      CALL PRLSTL(L,1H0)
      N=NAMEDL(L)
      IF(N.NE.0)GO TO 1
      PRINT 901
  901 FORMAT(*0 NO DESCRIPTION LIST*)
      GO TO 2
    1 PRINT 902
  902 FORMAT(*0 DESCRIPTION LIST*)
      CALL PRLSTL(N,1H0)
      DO 3 I=1,100
      IF(NAMTST(MADNTP(N,I)).EQ.0)GO TO 2
      LPTR=LDATT(N,I)
      IF(NAMTST(LPTR).NE.0)GO TO 3
      PRINT 903
  903 FORMAT(*0 CONSTRAINT LIST*)
      NUM=TOP(LPTR)
      CALL PRLSTL(LPTR,1H0)
      DO 5 J=1,NUM
      ITYP=LDATT(LPTR,2*J)
      LATT=LDATT(LPTR,2*J+1)
      PRINT 904,ITYP
  904 FORMAT(*0 CONSTRAINT SUBLIST, TYPE= *I2)
      CALL PRLSTL(LATT 1H0)
    5 CONTINUE
    3 CONTINUE
    2 LDOWN=LSUBOB(L)
      IF(LISTMT(LDOWN).EQ.0)GO TO 20
      DO 6 I=1,100
      IF(NAMTST(MADNTP(LDOWN,I)).EQ.0)GO TO 7
      IJK=LDATT(LDATT(LDOWN,I),1)
      PRINT 905,I,IJK
  905 FORMAT(*  SUBOBJECT *I2*  TYPE *I2)
    6 CONTINUE
    7 CALL LIST(LST)
      DO 8 I=1,100
      IF(NAMTST(MADNTP(LDOWN,I)).EQ.0)GO TO 9
    8 CALL IRALST(INLSTL(LRELN(LDATT(LDOWN,I)),LST))
    9 DO 10 I=1,100
      IF(NAMTST(MADNTP(LST,I)).EQ.0)GO TO 12
      IDAT=LDATT(LST,I)
      DO 11 J=1,100
      IF(NAMTST(MADNTP(LST,I+J)).EQ.0)GO TO 10
      IF(IDAT.NE.LDATT(LST,I+J))GO TO 11
      CALL DELETE(MADNTP(LST,I+J))
      J=J-1
```

```
 11 CONTINUE
 10 CONTINUE
 12 IF(LISTMT(LST).EQ.0)GO TO 21
    PRINT 906
906 FORMAT(*0 RELNS BETWEEN SUBOBJECTS*)
    DO 13 I=1,100
    IF(NAMTST(MADNTP(LST,I)).EQ.0)GO TO 14
    CALL PRLSTL(LDATT(LST,I),1H0)
    IF(NAMTST(NAMEDL(LDATT(LST,I))).NE.0)GO TO 13
    PRINT 909
909 FORMAT(*0*24X*DESCRIPTION LIST*)
    CALL PRLSTL(NAMEDL(LDATT(LST,I)),1H )
13  CONTINUE
14  GO TO 22
 20 PRINT 907
907 FORMAT(*0 NO SUBOBJECTS*)
    GO TO 22
 21 PRINT 908
908 FORMAT(*0 NO RELNS BETWEEN SUBOBJECTS*)
22  IF(NAMTST(LDOWN).EQ.0)CALL IRALST(LDOWN)
    IF(NAMTST(LST).EQ.0)CALL IRALST(LST)
    RETURN
    END


    FUNCTION PROBJS(PIC)
    INTEGER LEV(20)
    CALL LEVELS(PIC,LEV)
    DO 1 I=1,20
    IF(NAMTST(LEV(I)).NE.0)GO TO 2
1   CONTINUE
    I=21
2   LVNO=I-1
    DO 3 I=1,LVNO
    N=LSTLN(LEV(I))
    DO 3 J=1,N
3   CALL PROBJ(LDATT(LEV(I),J))
    DO 4 I=1,LVNO
4   CALL IRALST(LEV(I))
    PROBJS=0
    RETURN
    END


    FUNCTION ADJAC(NAME1,NRELN)
    INTEGER TOP
    EQUIVALENCE (IOR1,IOR)
    ADJAC=0.
    IF(LSTMRK(NRELN).NE.0)RETURN
    CALL MRKLST(1,NRELN)
    IF(LDATT(NRELN,2).NE.NAME1)NAME2=LDATT(NRELN,2)
    IF(LDATT(NRELN,3).NE.NAME1)NAME2=LDATT(NRELN,3)
    IF(TOP(NRELN).EQ.1)GO TO 1
    PRINT 100
100 FORMAT(* -ADJAC- ILLEGAL RELN TYPE*)
2   ADJAC=1.
    RETURN
1   IOR1=ITSVAL(2,NAME1)
```

```
      IOR2=ITSVAL(2,NAME2)
      IF(NAMTST(IOR1)*NAMTST(IOR2).NE.0)GO TO 3
      PRINT 101
 101  FORMAT(* -ADJAC- ORIENTATION UNFIXED*)
      GO TO 2
 3    IF(IOR1.EQ.IOR2)GO TO 10
      PRINT 102
 102  FORMAT(* -ADJAC- ORIENTATIONS UNEQUAL*)
      GO TO 2
C
C     FIND IF LEFT-RIGHT EXISTS FOR
C     CASE OF VERT OBJECTS, OR ABOVE-BELOW
C     FOR HORIZONTAL OBJECTS
C
 10   LALREL=LRELN(NAME1)
      N=LSTLN(LALREL)
      DO 4 I=1,N
      NREL1=LDATT(LALREL,I)
      IF(TOP(NREL1).NE.3+IOR)GO TO 4
      IF(LDATT(NREL1,2).EQ.NAME2)GO TO 5
      IF(LDATT(NREL1,3).NE.NAME2)GO TO 4
C
C     SET PRIMARY AND SECONDARY OBJECTS
C     ACCORDING TO LEFT-RIGHT, ABOVE-BELOW RELNS
C     OR BY DEFAULT
C
C     MARK ANY SUCH RELNS AS DONE
C
      NOB2=NAME2
      NOB1=NAME1
      GO TO 6
 5    NOB1=NAME2
      NOB2=NAME1
      GO TO 6
 4    CONTINUE
      NOB1=LDATT(NRELN,2)
      NOB2=LDATT(NRELN,3)
      GO TO 7
 6    CALL MRKLST(1,NREL1)
 7    IF(IOR.EQ.1)GO TO 8
C
C     DO VERTICAL OBJECTS
C
      CALL LST(M)
      CALL NEWBOT(3,M)
      CALL NEWBOT(NOB1,M)
      CALL CONSTR(4,NOB2,3,M)
      GO TO 9
C
C     DO HORIZONTAL OBJECTS
C
 8    CALL LST(M)
      CALL NEWBOT(5,M)
      CALL NEWBOT(NOB1,M)
      CALL CONSTR(6,NOB2,3,M)
C
C     DONE.  RETURN
C
 9    CALL NEWBOT(2+2*IOR,LST(M))
      CALL NEWBOT(NOB1,M)
      CALL CONSTR(2+2*IOR,NOB2,3,M)
      RETURN
      END


      FUNCTION JOIN(LNAM,NRELN)
      INTEGER TOP,POPTOP,ICT(2),BOT,TYP1,TYP2
      JOIN=0
      IF(LSTMRK(NRELN).NE.0)RETURN
      CALL MRKLST(1,NRELN)
      NAME1=LDATT(NRELN,2)
      IF(TOP(NRELN).EQ.2)GO TO 4
      PRINT 101
 101  FORMAT(* -JOIN- ILLEGAL RELN TYPE*)
      GO TO 2
 4    IF(NAMEDL(NRELN).NE.0)GO TO 1
      PRINT 100
 100  FORMAT(* -JOIN- NO ATTRIBUTE LIST*)
 2    JOIN=1
      RETURN
 1    IF(LDATT(NRELN,2).EQ.NAME1)GO TO 3
      NAME2=LDATT(NRELN,2)
      ICNVL2=ITSVAL(1,NRELN)
      ICNVL1=ITSVAL(2,NRELN)
      GO TO 5
 3    NAME2=LDATT(NRELN,3)
      ICNVL1=ITSVAL(1,NRELN)
      ICNVL2=ITSVAL(2,NRELN)
 5    ICT(1)=ICNVL1-4
      ICT(2)=ICNVL2-4
      TYP1=ITSVAL(ICT(1),NAME1)
      TYP2=ITSVAL(ICT(2),NAME2)
      IF(NAMTST(TYP1)*NAMTST(TYP2).NE.0)GO TO 17
      PRINT 102
 102  FORMAT(* -JOIN- CONTYP NOT PROPERLY SPECIFIED. *)
      GO TO 2
 17   IOR1=ITSVAL(2,NAME1)
      IOR2=ITSVAL(2,NAME2)
      IF(IOR1.NE.IOR2)GO TO 18
C
C     BOTH VERT OR BOTH HOR
C
      IF(TOP(NAME1).NE.7)GO TO 40
      NAME10=NAME2
      NAME2=NAME1
      ITMP=ICNVL1
      ICNVL1=ICNVL2
      ICNVL2=ITMP
      GO TO 44
 40   NAME10=NAME1
 44   CALL LST(M)
      CALL NEWBOT(18+2*(IOR1-1)-ICNVL1,M)
      CALL NEWBOT(NAME10,M)
      CALL CONSTR(18+2*(IOR2-1)-ICNVL2,NAME2,3 ,M)
      RETURN
 18   IF(TYP1+TYP2.NE.2)GO TO 19
C
C     DIFFERING DIRECTIONS, POINT - POINT
C
```

```
35    IF(TOP(NAME1).EQ.7.OR.TOP(NAME2).EQ.7)GO TO 41
      IF(IOR1.EQ.1)GO TO 20
      NHOR=NAME2
      NVERT=NAME1
      IOH=IOR2
      IOV=IOR1
      ICNHOR=ICNVL2
      ICNVER=ICNVL1
      GO TO 21
20    NHOR=NAME1
      NVERT=NAME2
      IOH=IOR1
      IOV=IOR2
      ICNHOR=ICNVL1
      ICNVER=ICNVL2
21    DO 22 I=1,2
      NN=-(I-2)*NVERT+(I-1)*NHOR
      NM=-(I-2)*NHOR+(I-1)*NVERT
      IJ=-(I-2)*ICNHOR+(I-1)*ICNVER
      II=-(I-2)*IOH+(I-1)*IOV
      CALL LST(M)
      CALL NEWBOT(2*I+1,M)
      CALL NEWBOT(NN,M)
      CALL NEWBOT(1,M)
      CALL NEWBOT(2*I+2,M)
      CALL NEWBOT(NN,M)
      CALL NEWBOT(4,M)
      CALL NEWBOT(-1,M)
      CALL NEWBOT(2,M)
22    CALL CONSTR(18+2*(II-1)-IJ,NM,3,M)
      RETURN
19    IF(TYP1+TYP2.EQ.4)GO TO 200
C
C     DIFFERING DIRNS, POINT - LINE
C
      GO TO 35
200   PRINT 103
103   FORMAT(* -JOIN- CAN'T DO THIS YET*)
      RETURN
C
C     IF JOIN IS DIFFERING DIRNS, PT-PT, AND 1OBJ IS A LINE        DO HERE
C
41    IF(TOP(NAME1).EQ.7)GO TO 42
      NOB=NAME1
      NLIN=NAME2
      ICNL=ICNVL2
      ICNO=ICNVL1
      IOL=IOR2
      IOO=IOR1
      GO TO 43
42    NOB=NAME2
      NLIN=NAME1
      ICNL=ICNVL1
      ICNO=ICNVL2
      IOL=IOR1
      IOO=IOR2
43    CALL NEWBOT(5-2*(IOO-1),LST(M))
      CALL NEWBOT(NOB,M)
      CALL NEWBOT(1,M)
      CALL NEWBOT(6-2*(IOO-1),M)
      CALL NEWBOT(NOB,M)
      CALL NEWBOT(4,M)
      CALL NEWBOT(-1,M)
      CALL NEWBOT(2,M)
      CALL CONSTR(18+2*(IOL-1)-ICNL,NLIN,3,M)
      CALL NEWBOT(20-2*(IOL-1)-ICNO,LST(M))
      CALL NEWBOT(NOB,M)
      CALL CONSTR(8-2*IOL,NLIN,3,M)
      RETURN
      END


      FUNCTION MODLIN(L)
      CALL LIST(L)
      MODLIN=L
      CALL NEWBOT(-7,L)
      DO 1 I=1,3
    1 CALL NEWBOT(0,L)
      CALL NEWVAL(2,-1,L)
      CALL NEWVAL(7,0,L)
      CALL NEWVAL(9,2,L)
      CALL NEWVAL(10,1,L)
      CALL NEWVAL(11,1,L)
      CALL NEWVAL(8,LST(M),L)
      CALL NEWBOT(1,M)
      CALL NEWBOT(2,M)
      CALL NEWBOT(LST(N),M)
      CALL NEWBOT(-1,N)
      CALL NEWBOT(0,N)
      DO 2 I=1,2
      CALL LST(M)
      CALL NEWBOT(2*I+2,M)
      CALL NEWBOT(L,M)
      CALL NEWBOT(1,M)
      CALL NEWBOT(9-I,M)
      CALL NEWBOT(L,M)
      CALL CONSTR(2*I+1,L,2,M)
      CALL NEWVAL(13+I,LST(M),L)
      CALL NEWBOT(2,M)
      CALL NEWBOT(6,M)
      CALL NEWBOT(LST(N1),M)
      CALL NEWBOT(9,M)
      CALL NEWBOT(LST(N2),M)
      CALL NEWBOT(6,N2)
      CALL NEWBOT(L,N2)
      CALL NEWBOT(2*I,N1)
    2 CALL NEWBOT(L,N1)
      RETURN
      END


      FUNCTION MODZP(L)
      MODZP=NUOB(L,-6)
      CALL JUPDN(L,NUOB(L1,4))
      CALL JUPDN(L,NUOB(L2,4))
      CALL JUPDN(L,NUOB(L3,7))
      CALL JUPDN(L,NUOB(L4,7))
      CALL NEWVAL(2,-1,L)
      CALL NEWVAL(10,2,L)
      CALL NEWVAL(11,2,L)
      CALL NEWBOT(18,LST(K))
      CALL NEWBOT(L3,K)
      CALL CONSTR(14,L,3,K)
      CALL NEWBOT(18,LST(K))
      CALL NEWBOT(L4,K)
      CALL CONSTR(15,L,3,K)
      CALL NEWVAL(8,LST(M1),L)
      CALL NEWBOT(2,M1)
      CALL NEWBOT(2,M1)
      CALL NEWBOT(LST(M11),M1)
      CALL NEWBOT(2,M1)
      CALL NEWBOT(LST(M12),M1)
      CALL NEWBOT(8,M11)
      CALL NEWBOT(L1,M11)
      CALL NEWBOT(8,M12)
      CALL NEWBOT(L2,M12)
      CALL NEWVAL(9,2,L)
      CALL LST(M)
      CALL NEWBOT(2,M)
      CALL NEWBOT(L,M)
      CALL CONSTR(2,L1,3,M)
      CALL LST(M)
      CALL NEWBOT(2,M)
      CALL NEWBOT(L,M)
      CALL CONSTR(2,L2,3,M)
      CALL NRELN(1,L1,L2)
      NR=NRELN(2,L1,L3)
      CALL NEWVAL(1,15,NR)
      CALL NEWVAL(2,14,NR)
      NR=NRELN(2,L1,L4)
      CALL NEWVAL(1,14,NR)
      CALL NEWVAL(2,14,NR)
      NR=NRELN(2,L3,L2)
      CALL NEWVAL(1,15,NR)
      CALL NEWVAL(2,15,NR)
      NR=NRELN(2,L2,L4)
      CALL NEWVAL(1,14,NR)
      CALL NEWVAL(2,15,NR)
      CALL LST(M)
      CALL NEWBOT(2,M)
      CALL NEWBOT(L,M)
      CALL CONSTR(2,L3,10,M)
      CALL LST(M)
      CALL NEWBOT(2,M)
      CALL NEWBOT(L,M)
      CALL CONSTR(2,L4,10,M)
      DO 1 I=1,2
      CALL LST(M)
      CALL NEWBOT(2*I+2,M)
      CALL NEWBOT(L,M)
      CALL NEWBOT(1,M)
      CALL NEWBOT(9-I,M)
      CALL NEWBOT(L,M)
      CALL CONSTR(2*I+1,L,2,M)
1     CONTINUE
      CALL NEWBOT(7,LST(M))
      CALL NEWBOT(L1,M)
      CALL NEWBOT(1,M)
      CALL NEWBOT(7,M)
      CALL NEWBOT(L2,M)
      CALL CONSTR(7,L,2,M)
      CALL NEWBOT(4,LST(M))
      CALL NEWBOT(L,M)
      CALL CONSTR(4,L1,3,M)
      CALL NEWBOT(6,LST(M))
      CALL NEWBOT(L,M)
      CALL CONSTR(6,L1,3,M)
      DO 2 I=1,2
      IF(I.EQ.1)LI=L1
      IF(I.EQ.2)LI=L2
      CALL NEWBOT(4,LST(M))
      CALL NEWBOT(LI,M)
      CALL NEWBOT(1,M)
      CALL NEWBOT(8,M)
      CALL NEWBOT(L,M)
2     CALL CONSTR(3,LI,3,M)
      RETURN
      END


      FUNCTION MODCOP(CCT,OBJ)
      INTEGER M(10),TYP,TYP1,TOP,CONTYP,CONSUB,OBJ,OBJ1
      IF(LSTMRK(OBJ).NE.0)GO TO 19
      IF(NAMTST(CCT).NE.0)GO TO 20
      TYP=TOP(CCT)-20
      TYP1=TOP(OBJ)
      IF(TYP1.EQ.TYP)GO TO 16
      IF(TYP1.NE.0)GO TO 17
      CALL SUBSTP(TYP,OBJ)
      GO TO 16
17    IF(TYP1.NE.4)GO TO 18
      IF(TYP.GE.7)GO TO 19
      CALL SUBSTP(TYP,OBJ)
      GO TO 16
18    IF(TYP.NE.4)GO TO 19
      IF(TYP1.GE.7)GO TO 19
      TYP=TYP1
      GO TO 16
19    MODCOP=1
      RETURN
16    MODCOP=0
      CALL MRKLST(1,OBJ)
      CALL MODLST(MODNAM,TYP)
      NDOWN=LDATT(MODNAM,3)
      L=LSUBOB(MODNAM)
      DO 1 I=1,NDOWN
1     CALL JUPDN(OBJ,NUOB(M(I),LDATT(LDATT(L,I),1)))
      IJK=1
      OBJ1=OBJ
      MOD1=MODNAM
52    DO 2 I=1,18
      IF(IFATR(I,MOD1).NE.0)GO TO 2
      IVAL=ITSVAL(I,MOD1)
      IF(IFATR(I,OBJ1).EQ.0)GO TO 30
40    IF(NAMTST(IVAL).EQ.0)GO TO 3
      CALL NEWVAL(I,IVAL,OBJ1)
      GO TO 2
```

```
3       NCON=LDATT(IVAL,1)
        DO 4 J=1,NCON
        CONTYP=LDATT(IVAL,2*J)
        CONSUB=LDATT(IVAL,2*J+1)
        CALL LST(NCNSUB)
        N=LSTLN(CONSUB)
        DO 5 K=1,N
        NDAT=LDATT(CONSUB,K)
        IF(NAMTST(NDAT),EQ,0)GO TO 6
        CALL NEWBOT(NDAT,NCNSUB)
        GO TO 5
6       IF(NDAT,EQ,MODNAM)CALL NEWBOT(OBJ,NCNSUB)
        DO 7 IJ=1,NDOWN
        IF(NDAT,EQ,LDATT(L,IJ))CALL NEWBOT(M(IJ),NCNSUB)
7       CONTINUE
5       CONTINUE
        CALL CONSTR(I,OBJ1,CONTYP,NCNSUB)
4       CONTINUE
2       CONTINUE
        GO TO (50,51),IJK
50      CALL LIST(LSTT)
        DO 8 I=1,NDOWN
8       CALL IRALST(INLSTL(LRELN(LDATT(L,I)),LSTTT)
        N=LSTLN(LSTT)
        DO 9 I=1,N
        IDAT=LDATT(LSTT,I)
        K=N-I
        DO 10 J=1,K
        IF(NAMTST(MADNTP(LSTT,I+J)),EQ,0)GO TO 9
        IF(IDAT,NE,LDATT(LSTT,I+J))GO TO 10
        CALL DELETE(MADNTP(LSTT,I+J))
        J=J-1
10      CONTINUE
9       CONTINUE
        NN=LSTLN(LSTT)
        DO 11 I=1,NN
        NAM=LDATT(LSTT,I)
        TYP=TOP(NAM)
        LST1=LDATT(NAM,2)
        LST2=LDATT(NAM,3)
        DO 12 J=1,N
        IF(LST1,EQ,LDATT(L,J))GO TO 13
12      CONTINUE
13      DO 14 K=1,N
        IF(LST2,EQ,LDATT(L,K))GO TO 61
14      CONTINUE
61      NR=NRELN(TYP,M(J),M(K))
        IF(NAMTST(NAMEDL(NAM)),NE,0)GO TO 11
        DO 62 IJ=1,20
        IF(IFATR(IJ,NAM),NE,0)GO TO 62
        CALL NEWVAL(IJ,ITSVAL(IJ,NAM),NR)
62      CONTINUE
11      CONTINUE
        DO 49 II=1,NDOWN
        MOD1=LDATT(L,II)
        OBJ1=M(II)
        IJK=2
        GO TO 52
51      CONTINUE
49      CONTINUE
```

```
      CALL IRALST(L)
      CALL IRALST(LSTT)
      CALL IROBJ(MODNAM)
      RETURN
20    IF(TOP(OBJ).EQ.0)GO TO 19
      TYP=TOP(OBJ)
      GO TO 16
30    IVL1=ITSVAL(I,OBJ1)
      IF(IVL1.LT.0)GO TO 40
      IF(IVL1.EQ.IVAL)GO TO 2
      IF(NAMTST(IVL1).NE.0)GO TO 41
42    IF(NAMTST(IVAL).EQ.0)GO TO 3
      CALL NEWBOT(-1,LST(MM))
      CALL NEWBOT(IVAL,MM)
      CALL CONSTR(I,OBJ1,3,MM)
      GO TO 2
41    CALL NOATVL(I,OBJ1)
      CALL NEWBOT(-1,LST(MM))
      CALL NEWBOT(IVL1,MM)
      CALL CONSTR(I,OBJ1,3,MM)
      GO TO 42
      END


      FUNCTION MAPCCT(CCT,PIC)
      INTEGER LEV(20)
      INTEGER TOP
      COMMON /COR/CORLST,COORD
      IF(NAMTST(CORLST).EQ.0)CALL IRALST(CORLST)
      CALL NEWBOT(CCT,LIST(CORLST))
      CALL NEWBOT(PIC,CORLST)
      CALL LEVELS(CCT,LEV)
      DO 10 I=1,20
      IF(NAMTST(LEV(I)).NE.0)GO TO 12
10    CONTINUE
      I=21
12    LVNO=I-1
      DO 2 I=1,LVNO
      NN=LSTLN(LEV(I))
      DO 2 J=1,NN
      N=LSTLN(CORLST)
      IDAT=LDATT(LEV(I),J)
      DO 3 K=1,N,2
      IF(LDATT(CORLST,K).EQ.IDAT)IPIC=LDATT(CORLST,K+1)
3     CONTINUE
      IF(MODCOP(IDAT,IPIC).NE.0)GO TO 50
      L1=LSUBOB(IDAT)
      L2=LSUBOB(IPIC)
      N=LSTLN(L1)
      N2=LSTLN(L2)
      IF(N2.EQ.N)GO TO 23
      N3=N+1
      DO 24 N4=N3,N2
      IF(TOP(LDATT(L2,N4)).EQ.7)GO TO 24
      CALL JUPDN(IDAT,NUOB(N5,20+TOP(LDATT(L2,N4))))
      IF(NAMTST(LEV (I+1)).EQ.0)GO TO 25
      CALL LIST(LEV(I+1))
25    CALL NEWBOT(N5,LEV(I+1))
```

```
      CALL IRALST(L1)
      L1=LSUBOR(IDAT)
      N=LSTLN(L1)
23    DO 4 K=1,N
      CALL NEWBOT(LDATT(L1,K),CORLST)
4     CALL NEWBOT(LDATT(L2,K),CORLST)
      CALL IRALST(L1)
      CALL IRALST(L2)
2     CONTINUE
20    DO 5 I=1,LVNO
5     CALL IRALST(LEV(I))
      ISW=0
      CALL LEVELS(PIC,LEV)
      DO 6 I=1,20
      IF(NAMTST(LEV(I)),NE,0)GO TO 7
6     CONTINUE
      I=21
7     LVNO=I-1
      DO 8 I=1,LVNO
      N=LSTLN(LEV(I))
      DO 9 J=1,N
      IF(LSTMRK(LDATT(LEV(I),J)),EQ.1)GO TO 9
      ISW=1
      IF(MODCOP(0,LDATT(LEV(I),J)),NE,0)GO TO 51
9     CONTINUE
8     CONTINUE
      IF(ISW,EQ,1)GO TO 20
      DO 21 I=1,LVNO
21    CALL IRALST(LEV(I))
      MAPCCT=0
      RETURN
50    PRINT 101
101   FORMAT(*0ONE OR BOTH OF THE FOLLOWING LISTS*
     1* IN A CALL TO MODCOP INVALID*)
      CALL PRLSTL(IDAT,1H0)
      CALL PRLSTL(IPIC,1H0)
      MAPCCT=1
      RETURN
51    PRINT 100
100   FORMAT(*0THE PICTORIAL OBJECT IN A CALL TO MODCOP HAS*
     1* ALREADY BEEN MAPPED*)
      CALL PRLSTL(LDATT(LEV(I),J),1H0)
      MAPCCT=1
      RETURN
      END



      FUNCTION MAKPIC(PIC)
      INTEGER POPTOP,BOT
      INTEGER LEV(20),TYPE,TOP
      MAKPIC=0
      LVNO=LEVELS(PIC,LEV)
      DO 1 I=1,LVNO
      N=LSTLN(LEV(I))
      DO 1 J=1,N
C
C     CHECK FOR D,L,
C     MAKE LIST OF ALL RELNS (LALREL)
C     THEN LIST OF RELNS TO CON OBJECTS (LCNREL)
```

```
C
      NAME=LDATT(LEV(I),J)
      IF(NAMEDL(NAME).NE.0)GO TO 10
      PRINT 101
 101  FORMAT(* -MAKPIC- NO DESCRIPTION LIST*)
      GO TO 9
 10   CALL GETRELN(NAME,LALREL,LCNREL)
C
C     PROCESS ORIENTATION
C
      IF(IFATR(2,NAME).EQ.0)GO TO 11
      PRINT 102
 102  FORMAT(* -MAKPIC- (WARNING) NO ORIENTATION SPEC,*)
      TYPE=TOP(NAME)
      CALL NEWVAL(2,-ITSVAL(2,MODLST(M,TYPE)),NAME)
      CALL IROBJ(M)
      GO TO 12
 11   IOR=ITSVAL(2,NAME)
      IF(NAMTST(IOR).EQ.0)GO TO 13
      IF(IOR.LT.0)IOR=-IOR
      IF(IOR.GT.2.OR.IOR.LT.1)GO TO 14
      CALL NEWVAL(2,IOR,NAME)
      GO TO 12
 14   PRINT 103
 103  FORMAT(* -MAKPIC- BAD ORIENTATION*)
      GO TO 9
 13   NCLST=IOR
      NCONSTR=TOP(NCLST)
      CALL LIST(ORVALS)
      DO 15 K=1,NCONSTR
       ICNTYP=LDATT(NCLST,2*K)
       IF(ICNTYP.EQ.3)GO TO 16
       IF(ICNTYP.EQ.10)GO TO 17
 25    PRINT 104
 104   FORMAT(* -MAKPIC- BAD ORIENTATION CONSTRAINT*)
       GO TO 9
 16    JSW=1
       GO TO 19
 17    JSW=-1
 19    NCSLST=LDATT(NCLST,2*K+1)
       IF(TOP(NCSLST).NE.-1)GO TO 18
       NUM=LDATT(NCSLST,2)
       NUM1=ISIGN(1,NUM)*(IABS(NUM)*JSW-3*(JSW-1)/2)
       CALL NEWBOT(NUM1,ORVALS)
       GO TO 15
 18    IF(TOP(NCSLST).NE.2)GO TO 25
       NUOR=ITSVAL(2,LDATT(NCSLST,2))
       IF(NAMTST(NUOR).EQ.0)GO TO 15
       NUOR2=ISIGN(1,NUOR)*(JSW*IABS(NUOR)-3*(JSW-1)/2)
       CALL NEWBOT(NUOR2,ORVALS)
 15    CONTINUE
       NOR=LSTLN(ORVALS)
       NUM=0
       DO 20 K=1,NOR
        IF(LDATT(ORVALS,K).LT.0)GO TO 20
        IF(NUM.EQ.0)NUOR=LDATT(ORVALS,K)
        NUM=1
        IF(NUOR.NE.LDATT(ORVALS,K))GO TO 25
 20     CONTINUE
        IF(NUM.EQ.0)GO TO 22
      CALL NEWVAL(2,NUOR,NAME)
      CALL IRALST(ORVALS)
      GO TO 12
 22   CALL NEWVAL(2,-TOP(ORVALS),NAME)
      CALL IRALST(ORVALS)
 12   CONTINUE
      CALL IRALST(LCNREL)
      CALL IRALST(LALREL)
 1    CONTINUE
C
C     REVERSE CONSTRAINTS AS REQD, BY ORIENTATIONS
C     DO SIZES FIRST (7 AND 8)
C
      DO 40 I=1,LVNO
      M=LSTLN(LEV(I))
      DO 40 J=1,N
      NAME=LDATT(LEV(I),J)
      IF(ITSVAL(2,NAME).EQ.-ITSVAL(2,MODLST(M,TOP(NAME))))GO TO 333
      CALL IROBJ(M)
      IFST=ITSVAL(7,NAME)
      ISCND=ITSVAL(8,NAME)
      I1=IFATR(7,NAME)
      I2=IFATR(8,NAME)
      IF(I1.EQ.0)CALL NEWVAL(8,IFST,NAME)
      IF(I2.EQ.0)CALL NEWVAL(7,ISCND,NAME)
      IF(NAMTST(IFST).NE.0)GO TO 31
      NCLST=IFST
      ISW=1
 32   NCONSTR=TOP(NCLST)
      DO 33 K=1,NCONSTR
      NCSLST=LDATT(NCLST,2*K+1)
      LNCSL=LSTLN(NCSLST)
      DO 34 K1=1,LNCSL
      IF(NAMTST(LDATT(NCSLST,K1)).NE.0)GO TO 34
      IATR=LDATT(NCSLST,K1-1)
      IOBJ=LDATT(NCSLST,K1)
      IF(ITSVAL(2,IOBJ).EQ.-ITSVAL(2,MODLST(M,TOP(IOBJ))))GO TO 332
      CALL IROBJ(M)
      IATR=IATR+(1-IFIX(SIGN(1.,6.5-IATR)))*IFIX(SIGN(1.,4.5-IATR))
     1   +(1-IFIX(SIGN(1.,4.5-IATR)))/2*(-1)**(1+IATR)
      CALL SUBST(IATR,MADNTP(NCSLST,K1-1))
      GO TO 34
 332  CALL IROBJ(M)
 34   CONTINUE
 33   CONTINUE
      GO TO (31,30),ISW
 31   IF(NAMTST(ISCND).NE.0)GO TO 30
      NCLST=ISCND
      ISW=2
      GO TO 32
 30   CONTINUE
C
C     NOW DO CONVALS, (14,15,...,) WITHIN
C     SAME LOOP AS SIZES
C
      NOCONS=ITSVAL(9,NAME)
      DO 43 K=1,NOCONS
      NCNVAL=13+K
      NCNTYP=9+K
      IF(ITSVAL(NCNTYP,NAME).EQ.2)GO TO 43
      IF(NAMTST(ITSVAL(NCNTYP,NAME)),EQ,0)GO TO 43
C     CONVAL IS A POINT AND SPECIFIED HERE,
      NCLST=ITSVAL(NCNVAL,NAME)
      NCONSTR=TOP(NCLST)
      DO 44 K1=1,NCONSTR
      CALL SUBST(15-LDATT(NCLST,2*K1),MADNTP(NCLST,2*K1))
      NCSLST=LDATT(NCLST,2*K1+1)
      N1=(1+LSTLN(NCSLST))/3
      DO 44 K2=1,N1
      IAT=LDATT(NCSLST,3*K2-2)
      INAM=LDATT(NCSLST,3*K2-1)
      IF(IAT,EQ,-1)GO TO 44
      IF(INAM,EQ,NAME)GO TO 45
      PRINT 105
 105  FORMAT(* -MAKPIC- (WARNING) PECULIAR CONVAL CONSTRAINT LIST,*)
 45   IOUT=0
      IF(IAT,EQ,3)IOUT=5
      IF(IAT,EQ,4)IOUT=6
      IF(IAT,EQ,5)IOUT=4
      IF(IAT,EQ,6)IOUT=3
      IF(IOUT,NE,0)GO TO 46
      PRINT 106
 106  FORMAT(* -MAKPIC- CONVAL CONSTRAINT, BAD ATTRIBUTE,*)
      GO TO 9
 46   CALL SUBST(IOUT,MADNTP(NCSLST,3*K2-2))
 44   CONTINUE
 43   CONTINUE
      GO TO 40
 333  CALL PROBJS(M)
      CALL IROBJ(M)
 40   CONTINUE
C
C     MAP ALL RELNS EXCEPT JOIN
C
      DO 41 I=1,LVNO
      N=LSTLN(LEV(I))
      DO 41 J=1,N
      NAME=LDATT(LEV(I),J)
      CALL GETRELN(NAME,LALREL,LCNREL)
      N1=LSTLN(LCNREL)
      DO 4 K=1,N1
      IT=TOP(LDATT(LCNREL,K))
      GO TO (8,5,4,7,8),IT+1
 5    CALL ADJAC(NAME,LDATT(LCNREL,K))
      GO TO 4
 7    CALL COLLIN(NAME,LDATT(LCNREL,K))
      GO TO 4
 8    PRINT 100
 100  FORMAT(* -MAKPIC- ILLEGAL RELN, TYPE *)
 9    MAKPIC=1
      RETURN
 4    CONTINUE
      CALL IRALST(LCNREL)
      N1=LSTLN(LALREL)
      DO 200 K=1,N1
      IT=TOP(LDATT(LALREL,K))
      GO TO (8,201,200,202,8),IT+1
 201  CALL ADJAC(NAME,LDATT(LALREL,K))
      GO TO 200
 202  CALL COLLIN(NAME,LDATT(LALREL,K))
 200   CONTINUE
       CALL IRALST(LALREL)
  41   CONTINUE
C
C      TRACE DOWN STRUCTURE TO DETERMINE ALL CONTYPS,
C
       DO 51 I=1,LVNO
       N=LSTLN(LEV(I))
       DO 51 J=1,N
       NAME=LDATT(LEV(I),J)
       NCON=ITSVAL(9,NAME)
       DO 51 K=1,NCON
       CALL LIST(STACK)
       IC=9+K
       NM=NAME
  54   IF(NAMTST(ITSVAL(IC,NM)).NE.0)GO TO 52
       CALL NEWTOP(NM,STACK)
       CALL NEWTOP(IC,STACK)
       NCLST=ITSVAL(IC,NM)
       IF(TOP(NCLST).EQ.1)GO TO 53
  56   PRINT 107
 107   FORMAT(* -MAKPIC- BAD CONTYP CHAIN*)
       GO TO 9
  53   NCSLST=LDATT(NCLST,3)
       IF(LDATT(NCLST,2).NE.3)GO TO 56
       IF(LSTLN(NCSLST).NE.2)GO TO 56
       IC=TOP(NCSLST)
       NM=BOT(NCSLST)
       IF(10.LE.IC.AND.IC.LE.13)GO TO 54
       GO TO 56
C
C      REACHED BOTTOM, NOW COME UP
C
  52   ICV=ITSVAL(IC,NM)
       NCL=ITSVAL(IC+4,NM)
  60   IF(LISTMT(STACK).EQ.0)GO TO 55
       IC=POPTOP(STACK)
       NM=POPTOP(STACK)
       CALL NEWVAL(IC,ICV,NM)
       LNT1=LSTLN(NCL)
       CALL LST(M0)
       DO 57 IJ=1,LNT1
       IF(NAMTST(LDATT(NCL,IJ)).EQ.0)GO TO 58
       CALL NEWBOT(LDATT(NCL,IJ),M0)
       GO TO 57
  58   CALL LST(M1)
       M2=LDATT(NCL,IJ)
       CALL NEWBOT(M1,M0)
       LNT2=LSTLN(M2)
       DO 59 IJK=1,LNT2
  59   CALL NEWBOT(LDATT(M2,IJK),M1)
  57   CONTINUE
       CALL NEWVAL(IC+4,M0,NM)
       GO TO 60
  55   CALL IRALST(STACK)
  51   CONTINUE
C
C      NOW DO JOINS
C
       DO 42 I=1,LVNO
```

```
      N=LSTLN(LEV(I))
      DO 42 J=1,N
      NAME=LDATT(LEV(I),J)
      CALL GETRELN(NAME,LALREL,LCNREL)
      N1=LSTLN(LCNREL)
      DO 21 K=1,N1
21    IF(TOP(LDATT(LCNREL,K)).EQ.2)CALL JOIN(NAME,LDATT(LCNREL,K))
      CALL IRALST(LALREL)
      CALL IRALST(LCNREL)
42    CONTINUE
      CALL NCOORD(PIC)
      DO 203 I=1,LVNO
203   CALL IRALST(LEV(I))
      RETURN
      END



      FUNCTION NCOORD(PIC)
      INTEGER LEV(20),TOP,POPTOP
      NCOORD=0
      IFST=0
      LVNO=LEVELS(PIC,LEV)
      DO 1 I=1,LVNO
      N=LSTLN(LEV(I))
      DO 1 J=1,N
      NAME=LDATT(LEV(I),J)
      DO 2 IJ=3,6
      JJ=3-IJ+(IJ-1)/2*4
      IF(IFATR(JJ,NAME).NE.0)GO TO 3
      IVAL=ITSVAL(JJ,NAME)
      IF(NAMTST(IVAL))2,4,2
3     IF(IFST.NE.0)GO TO 5
      IFST=1
      CALL NEWVAL(JJ,20,NAME)
      GO TO 2
5     IF(IFST.NE.1)GO TO 6
      IFST=2
      CALL NEWVAL(JJ,20,NAME)
      GO TO 2
6     PRINT 100
100   FORMAT(* -NCOORD- (WARNING) MORE THAN 2 ARBITRARY COORDS*)
      NTST=TOP(NAME)
      PRINT 500,NAME,NTST,JJ
500   FORMAT(* NAME,TOP(NAME),ATTRIBUTE  *3I10)
      CALL NEWVAL(JJ,20,NAME)
      GO TO 2
4     CALL LIST(STACK)
      IATO=JJ
      INAMO=NAME
      NCLST=IVAL
15    NCONS=TOP(NCLST)
      IEQSW=0
      DO 7 II=1,NCONS
      IBRSW=1
      IF(LDATT(NCLST,2*II).NE.3)GO TO 7
      IEQSW=1
21    NCSLST=LDATT(NCLST,2*II+1)
C
C     CHECK IF CAN BE DONE
```

```
C
      NDATS=(LSTLN(NCSLST)+1)/3
      DO 8 IJ1=1,NDATS
      IF(LDATT(NCSLST,3*IJ1-2).EQ.-1)GO TO 8
      IAT=LDATT(NCSLST,3*IJ1-2)
      IOB=LDATT(NCSLST,3*IJ1-1)
      IF(NAMTST(ITSVAL(IAT,IOB)).EQ.0)GO TO 9
      IDAT=ITSVAL(IAT,IOB)
      CALL SUBST(-1,MADNTP(NCSLST,3*IJ1-2))
      CALL SUBST(IDAT,MADNTP(NCSLST,3*IJ1-1))
 8    CONTINUE
C
C     CAN DO,    DO
C
      NDATS=NDATS-1
      IARG=LDATT(NCSLST,2)
      DO 10 IJ1=1,NDATS
      NARG=LDATT(NCSLST,3*IJ1+2)
      NOP=LDATT(NCSLST,3*IJ1)
      GO TO (11,12,13,14),NOP
 11   IARG=IARG+NARG
      GO TO 10
 12   IARG=IARG-NARG
      GO TO 10
 13   IARG=IARG*NARG
      GO TO 10
 14   IARG=IARG/NARG
 10   CONTINUE
      CALL MTLIST(NCSLST)
      CALL NEWBOT(-1,NCSLST)
      CALL NEWBOT(IARG,NCSLST)
      GO TO (7,20),IBRSW
C
C     CAN'T DO.  PUSH STACK
C
 9    CALL NEWTOP(INAMO,STACK)
      CALL NEWTOP(IATO,STACK)
      NCLST=ITSVAL(IAT,IOB)
      IATO=IAT
      INAMO=IOB
      GO TO 15
 7    CONTINUE
C
C     DONE EQUALS,  CHECK IF ANY
C
      IF(IEQSW.EQ.0)GO TO 16
      DO 17 IJ1=1,NCONS
 17   IF(LDATT(NCLST,2*IJ1).EQ.3)GO TO 18
      IJ1=NCONS
 18   IDAT=LDATT(LDATT(NCLST,2*IJ1+1),2)
 26   CALL NEWVAL(IATO,IDAT,INAMO)
C
C     POP STACK AND RESTART
C
      IF(LISTMT(STACK).EQ.0)GO TO 19
      IATO=POPTOP(STACK)
      INAMO=POPTOP(STACK)
      NCLST=ITSVAL(IATO,INAMO)
      GO TO 15
```

```
C       NO EQUALS,  DO OTHERS
C
 16     DO 20 II=1,NCONS
        IBRSW=2
        IF(LDATT(NCLST,2*II),EQ,3)GO TO 20
        GO TO 21
 20     CONTINUE
C
C       RUN THRO' AND FIND GT AND LT CONSTRAINT
C
 22     NGT=-10000
        NLT=100000
        LGT=LLT=0
        DO 23 II=1,NCONS
        IF(LDATT(NCLST,2*II),EQ,2)GO TO 24
        IF(LDATT(NCLST,2*II),NE,1)GO TO 23
        LLT=-1
        NN=LDATT(LDATT(NCLST,2*II+1),2)
        IF(NN.LT.NLT)NLT=NN
        GO TO 23
 24     NN=LDATT(LDATT(NCLST,2*II+1),2)
        LGT=-1
        IF(NN,GT.NGT)NGT=NN
 23     CONTINUE
        IF(LGT,EQ,0)GO TO 25
        IDAT=NGT
        IF(LLT,EQ,0)GO TO 26
        IF(IDAT,LT,NLT)GO TO 26
        PRINT 101
 101    FORMAT(* -NCOORD- (WARNING) CONFLICTING CONSTRAINTS, GT CHOSEN*)
        GO TO 26
 25     IDAT=NLT
        IF(LLT,EQ,0)IDAT=0
        GO TO 26
C
C       DONE COMPLETELY
C
 19     CALL IRALST(STACK)
 2      CONTINUE
 1      CONTINUE
        RETURN
        END


        FUNCTION GETCHAN(L)
        INTEGER PIC(50),IK(8)
        COMMON /COR/ CORLST,COORD
        COMMON /VISTRAN/KA,KB,KC,LX,LY,LD
        DATA (IK=2H R,2H L,2H C,2H Z,3H ZS,3H ZR,3HRST,4HQUIT)
        IF(NAMTST(L),EQ,0)GO TO 1
        GETCHAN=-RDCCT(L)
        RETURN
 1      CALL CLEAR(PIC,50)
        CALL MOVE(100,975)
        CALL VTEXT(25HSELECT OLD CIRCUIT OBJECT,25,2)
        CALL DISPLAY(PIC)
        CALL DETECT
        CALL ERASE(PIC)
        IF(KC,EQ,63)CALL VISTOP
```

```
      IF(LD.EQ.0)CALL VISTOP
      IX=LX
      IY=LY
      CALL CLEAR(PIC,50)
      CALL MOVE(100,975)
      CALL VTEXT(22HSELECT NEW OBJECT TYPE,22,2)
      CALL DISPLAY(PIC)
      CALL CHOOSE(IK,8)
      CALL ERASE(PIC)
C
C     FIND WHICH OLD OBJECT IS SELECTED
C
      NN=LSTLN(COORD)
      DO 2 I=1,NN,2
      IF(LDATT(COORD,I).NE.IX)GO TO 2
      IF(LDATT(COORD,I+1).NE.IY)GO TO 2
      NOBOLD=LDATT(CORLST,I)
      GO TO 3
  2   CONTINUE
      CALL VISTOP
  3   IF(LD.EQ.0.OR.KC.EQ.63.OR.LD.EQ.8)CALL VISTOP
      IOUT=(I+1)/2
      PRINT 100,IOUT,IK(LD)
 100  FORMAT(* GETCHAN, OLD OBJ IS *I4* NEW OBJ IS *A4)
      IF(LD.EQ.7)GO TO 6
      IF(NOBOLD.NE.L)GO TO 4
      CALL IROBJ(L)
      CALL NUOB(L,20+LD)
      GO TO 5
  4   NPREV=LDATT(NOBOLD,5)
      II=5+LDATT(NPREV,2)
      JJ=II+LDATT(NPREV,3)
      DO 12 IJ=II,JJ
 12   IF(LDATT(NPREV,IJ).EQ.NOBOLD)GO TO 13
 13   IAD=MADNTP(NPREV,IJ-1)
      CALL RSTRUC(NOBOLD)
      CALL NUOB(NEW,20+LD)
      CALL SUBST(1+LDATT(NPREV,3),MADNTP(NPREV,3))
      CALL SUBST(1+LDATT(NEW,2),MADNTP(NEW,2))
      CALL NXTRGT(NEW,IAD)
      CALL NXTRGT(NPREV,MADNTP(NEW,4))
  5   GETCHAN=0.
      RETURN
  6   GETCHAN=1.
      RETURN
      END


      FUNCTION RDCCT(L)
      INTEGER II(9)
      INTEGER PIC(50)
      COMMON /VISTRAN/KA,KB,KC,LX,LY,LD
      DATA (II=2H R,2H L,2H C,2H Z,3H ZS,3H ZP,3HRST,3H CI,4HQUIT)
      CALL ERASEALL
      CALL CLEAR(PIC,50)
      CALL MOVE(180,500)
      CALL VTEXT(27HSELECT NEW CCT TYPE OR QUIT,27,3)
      CALL DISPLAY(PIC)
      IF(KC.EQ.63)CALL VISTOP
      IF(LD.EQ.0)CALL VISTOP
      PRINT 100,II(LD)
100   FORMAT(* RDCCT SELECTION IS *A4)
      GO TO (2,2,2,2,2,2,3,5,1),LD
1     RDCCT=1.
      GO TO 4
3     RDCCT=-1.
      GO TO 4
2     CALL NUOR(L,20+LD)
      RDCCT=0.
4     RETURN
5     CALL ERASEALL
      CALL CLEAR(PIC,50)
      CALL MOVE(150,500)
      CALL VTEXT(33HPREPARE DOC 'DATA' THEN INTERRUPT ,33,3)
      CALL DISPLAY(PIC)
      CALL BREAKOUT
      IF(KC.EQ.63)CALL VISTOP
      RDCCT=CIRDCCT(L)
      GO TO 4
      END


      SUBROUTINE CHOOSE(A,N)
      INTEGER P(100),A(1)
      COMMON /VISTRAN/KA,KB,KC,LX,LY,LD
      CALL CLEAR(P,100)
      DO 1 I=1,N
      IF(N.EQ.1)GO TO 4
      IX=10+900*(I-1)/(N-1)
5     CALL MOVE(IX,10)
      CALL RVECTOR(0,30)
      CALL RVECTOR(60,0)
      CALL RVECTOR(0,-30)
      CALL RVECTOR(-60,0)
      CALL MOVE(IX+11,24)
      CALL PENABLE(1)
      CALL VTEXT(A(I),4,2)
1     CALL PENABLE(0)
      CALL DISPLAY
      CALL DETECT
      IF(LD.EQ.0)RETURN
      IF(LY.NE.24)GO TO 3
      IF(N.EQ.1)GO TO 6
      LD=(LX-10)*(N-1)/900+1
      GO TO 2
3     LD=0
2     CALL ERASE(P)
      RETURN
4     IX=500
      GO TO 5
6     LD=1
      GO TO 2
      END
      INTEGER LEV(20),TYPE,TOP,NAMTYP(7)
      INTEGER OUT(5)
      COMMON /COR/CORLST,COORD
      TYPE BYTE5 (/6) FOUT(26),FIX(26),A
      DATA (NAMTYP=8H RESIST,,8H  CAPAC,,8H INDUCT,,8H     IMP,,
     18HSER,IMP,,8HPAR,IMP,,8H    LINE   )
      DATA (A=1HA),(FIX=26H(*0*40X,A8,I8,I8,I8,I8,I8))
      LVNO=LEVELS(L,LEV)
      CALL OUTLINE(-1,0,OUT)
      IF(NAMTST(COORD).EQ.0)CALL IRALST(COORD)
      CALL LIST(COORD)
      NNN=LSTLN(CORLST)
      DO 7 I=1,NNN
7     CALL NEWROT(0,COORD)
      PRINT 100
100   FORMAT(*1*55X*COT, DIAGRAM OUTPUT*)
      PRINT 101
101   FORMAT(*0*40X*    TYPE  ORIEN.    XMAX    XMIN    YMAX    YMIN*)
      DO 1 I=1,LVNO
      N=LSTLN(LEV(I))
      DO 1 J=1,N
      NAME=LDATT(LEV(I),J)
      IF(LDATT(NAME,3).NE.0)GO TO 1
      DO 6 K=1,26
6     FOUT(K)=FIX(K)
      TYPE=TOP(NAME)
      IF(NAMTST(NAMEDL(NAME)).NE.0)GO TO 2
      DO 3 K=2,6
      IF(IFATR(K,NAME).NE.0)GO TO 4
      IF(NAMTST(ITSVAL(K,NAME)).EQ.0)GO TO 5
      OUT(K-1)=ITSVAL(K,NAME)
      GO TO 3
4     OUT(K-1)=8H  UNSPEC
      FOUT(3*K+6)=A
      GO TO 3
5     OUT(K-1)=8H   UNFIX
      FOUT(3*K+6)=A
3     CONTINUE
      DO 8 K=2,NNN,2
      IF(NAME.NE.LDATT(CORLST,K))GO TO 8
      X=(OUT(2)+OUT(3))/2.
      Y=(OUT(4)+OUT(5))/2.
      CALL SUBST(X,MADNTP(COORD,K-1))
      CALL SUBST(Y,MADNTP(COORD,K))
      GO TO 9
8     CONTINUE
9     CONTINUE
      PRINT FOUT,NAMTYP(TYPE),OUT
      CALL OUTLINE(0,TYPE,OUT)
      GO TO 1
2     PRINT 900,NAMTYP(TYPE)
900   FORMAT(*0*40X,A8*     NO SPECIFICATION*)
1     CONTINUE
      CALL OUTLINE(1,0,OUT)
      PICOUT=0.
      RETURN
      END
```

Appendix 2 - A DATA STRUCTURE EXAMPLE

This appendix lists the pictorial data structure, when partially processes, for a series impedor made up of a capacitor and an inductor. The structure may be followed using the list formats outlined in Chapter 9 (assuming a knowledge of SLIP). A simple structure is given here, again because of space requirements.

```
OBJECT STRUCTURE

                           BEGIN LIST 29975 2 30045 30009 1 30053      5
30009 0 29975 29979                      5          0000000000000005
29979 0 30009 30047                      0          0000000000000000
30047 0 29979 29983                      2          0000000000000002
29983 0 30047 30027                      0          0000000000000000
30027 1 29983 30045            03601722689          0007250100072501 LIST ADDRESS= 30017
30045 1 30027 29975            03903712595          0007252300072523 LIST ADDRESS= 30035
                           END LIST

DESCRIPTION LIST

                           BEGIN LIST 30053 2 30165 30055 0      0      0
30055 0 30053 30057                      2          0000000000000002
30057 0 30055 30067                     -1          7777777777777776
30067 0 30057 30069                      7          0000000000000007
30069 1 30067 30091            04407029105          0007256100072561 LIST ADDRESS= 30065
30091 0 30069 30093                      9          0000000000000011
30093 0 30091 30103                      2          0000000000000002
30103 0 30093 30105                     10          0000000000000012
30105 1 30103 30123            05011008917          0007262500072625 LIST ADDRESS= 30101
30123 0 30105 30125                     11          0000000000000013
30125 1 30123 30143            05346553257          0007265100072651 LIST ADDRESS= 30121
30143 0 30125 30145                     14          0000000000000016
30145 1 30143 30163            05682097597          0007267500072675 LIST ADDRESS= 30141
30163 0 30145 30165                     15          0000000000000017
30165 1 30163 30053            06017641937          0007272100072721 LIST ADDRESS= 30161
                           END LIST

CONSTRAINT LIST

                           BEGIN LIST 30065 2 30089 30085 0      0      1
30085 0 30065 30075                      2          0000000000000002
30075 0 30085 30077                      2          0000000000000002
30077 1 30075 30087            04306365803          0007255300072553 LIST ADDRESS= 30059
30087 0 30077 30089                      2          0000000000000002
30089 1 30087 30065            04641910143          0007257700072577 LIST ADDRESS= 30079
                           END LIST

CONSTRAINT SUBLIST, TYPE=  2

                           BEGIN LIST 30059 2 30063 30061 0      0      1
30061 0 30059 30063                      7          0000000000000007
30063 1 30061 30059            03601722689          0007250100072501 LIST ADDRESS= 30017
                           END LIST

CONSTRAINT SUBLIST, TYPE=  2

                           BEGIN LIST 30079 2 30083 30081 0      0      1
30081 0 30079 30083                      7          0000000000000007
30083 1 30081 30079            03903712595          0007252300072523 LIST ADDRESS= 30035
                           END LIST

CONSTRAINT LIST

                           BEGIN LIST 30101 2 30113 30109 0      0      1
30109 0 30101 30111                      1          0000000000000001
30111 0 30109 30113                      3          0000000000000003
30113 1 30111 30101            04910345615          0007261700072617 LIST ADDRESS= 30095
                           END LIST

CONSTRAINT SUBLIST, TYPE=  3
```

```
                              BEGIN LIST 30095 2 30099 30097 0      0      1
30097 0 30095 30099                      10            0000000000000012
30099 1 30097 30095              03601722689            0007250100072501 LIST ADDRESS= 30017
                              END LIST

CONSTRAINT LIST

                              BEGIN LIST 30121 2 30133 30129 0      0      1
30129 0 30121 30131                       1            0000000000000001
30131 0 30129 30133                       3            0000000000000003
30133 1 30131 30121              05245889955            0007264300072643 LIST ADDRESS= 30115
                              END LIST

CONSTRAINT SUBLIST, TYPE=  3

                              BEGIN LIST 30115 2 30119 30117 0      0      1
30117 0 30115 30119                      11            0000000000000013
30119 1 30117 30115              03903712595            0007252300072523 LIST ADDRESS= 30035
                              END LIST

CONSTRAINT LIST

                              BEGIN LIST 30141 2 30153 30149 0      0      1
30149 0 30141 30151                       1            0000000000000001
30151 0 30149 30153                       3            0000000000000003
30153 1 30151 30141              05581434295            0007266700072667 LIST ADDRESS= 30135
                              END LIST

CONSTRAINT SUBLIST, TYPE=  3

                              BEGIN LIST 30135 2 30139 30137 0      0      1
30137 0 30135 30139                      14            0000000000000016
30139 1 30137 30135              03601722689            0007250100072501 LIST ADDRESS= 30017
                              END LIST

CONSTRAINT LIST

                              BEGIN LIST 30161 2 30173 30169 0      0      1
30169 0 30161 30171                       1            0000000000000001
30171 0 30169 30173                       3            0000000000000003
30173 1 30171 30161              05916978635            0007271300072713 LIST ADDRESS= 30155
                              END LIST

CONSTRAINT SUBLIST, TYPE=  3

                              BEGIN LIST 30155 2 30159 30157 0      0      1
30157 0 30155 30159                      15            0000000000000017
30159 1 30157 30155              03903712595            0007252300072523 LIST ADDRESS= 30035
                              END LIST
SUBOBJECT  1  TYPE  2
SUBOBJECT  2  TYPE  3

RELNS BETWEEN SUBOBJECTS

                              BEGIN LIST 30189 2 30195 30191 0      0      4
30191 0 30189 30193                       2            0000000000000002
30193 1 30191 30195              03601722689            0007250100072501 LIST ADDRESS= 30017
30195 1 30193 30189              03903712595            0007252300072523 LIST ADDRESS= 30035
                              END LIST

                              BEGIN LIST 30205 2 30211 30207 0      0      4
30207 0 30205 30209                       3            0000000000000003
```

```
OBJECT STRUCTURE

                               BEGIN LIST 30017 2 30215 30241 1 30245     18
30241 0 30017 30033                           2           0000000000000002
30033 0 30241 30023                           1           0000000000000001
30023 0 30033 30213                           0           0000000000000000
30213 0 30023 30031                           2           0000000000000002
30031 1 30213 30199                 02897079575           0007242700072427 LIST ADDRESS= 29975
30199 1 30031 30215                 06487404013           0007275500072755 LIST ADDRESS= 30189
30215 1 30199 30017                 06755839485           0007277500072775 LIST ADDRESS= 30205
                               END LIST

DESCRIPTION LIST

                               BEGIN LIST 30245 2 30341 30247 0       0      0
30247 0 30245 30249                           2           0000000000000002
30249 0 30247 30251                          -1           7777777777777776
30251 0 30249 30253                           7           0000000000000007
30253 0 30251 30263                           1           0000000000000001
30263 0 30253 30265                           8           0000000000000010
30265 1 30263 30275                 07695363637           0007306500073065 LIST ADDRESS= 30261
30275 0 30265 30277                           9           0000000000000011
30277 0 30275 30279                           2           0000000000000002
30279 0 30277 30281                          10           0000000000000012
30281 0 30279 30283                           1           0000000000000001
30283 0 30281 30285                          11           0000000000000013
30285 0 30263 30295                           1           0000000000000001
30295 0 30285 30297                          14           0000000000000016
30297 1 30295 30339                 08232234581           0007312500073125 LIST ADDRESS= 30293
30339 0 30297 30341                          15           0000000000000017
30341 1 30339 30245                 08970432129           0007320100073201 LIST ADDRESS= 30337
                               END LIST

CONSTRAINT LIST

                               BEGIN LIST 30261 2 30273 30269 0       0      1
30269 0 30261 30271                           1           0000000000000001
30271 0 30269 30273                           2           0000000000000002
30273 1 30271 30261                 07594700335           0007305700073057 LIST ADDRESS= 30255
                               END LIST

CONSTRAINT SUBLIST, TYPE=  2

                               BEGIN LIST 30255 2 30259 30257 0       0      1
30257 0 30255 30259                          -1           7777777777777776
30259 0 30257 30255                           2           0000000000000002
                               END LIST

CONSTRAINT LIST

                               BEGIN LIST 30293 2 30329 30325 0       0      1
30325 0 30293 30303                           2           0000000000000002
30303 0 30325 30305                           6           0000000000000006
30305 1 30303 30327                 08131571279           0007311700073117 LIST ADDRESS= 30287
30327 0 30305 30329                           9           0000000000000011
30329 1 30327 30293                 08467115619           0007314300073143 LIST ADDRESS= 30307
                               END LIST

CONSTRAINT SUBLIST, TYPE=  6

                               BEGIN LIST 30287 2 30291 30289 0       0      1
30289 0 30287 30291                           4           0000000000000004
30291 1 30289 30287                 03601722689           0007250100072501 LIST ADDRESS= 30017
```

```
30209 1 30207 30211          03601722689          0007250100072501 LIST ADDRESS= 30017
30211 1 30209 30205          03903712595          0007252300072523 LIST ADDRESS= 30035
                         END LIST
```

```
                                END LIST

CONSTRAINT SUBLIST, TYPE=  9

                                BEGIN LIST 30307 2 30323 30309 0      0      1
30309 0 30307 30311                             5           0000000000000005
30311 1 30309 30313                   03601722689           0007250100072501 LIST ADDRESS= 30017
30313 0 30311 30315                             1           0000000000000001
30315 0 30313 30317                             6           0000000000000006
30317 1 30315 30319                   03601722689           0007250100072501 LIST ADDRESS= 30017
30319 0 30317 30321                             4           0000000000000004
30321 0 30319 30323                            -1           7777777777777776
30323 0 30321 30307                             2           0000000000000002
                                END LIST

CONSTRAINT LIST

                                BEGIN LIST 30337 2 30373 30369 0      0      1
30369 0 30337 30347                             2           0000000000000002
30347 0 30369 30349                             6           0000000000000006
30349 1 30347 30371                   08869768827           0007317300073173 LIST ADDRESS= 30331
30371 0 30349 30373                             9           0000000000000011
30373 1 30371 30337                   09205313167           0007321700073217 LIST ADDRESS= 30351
                                END LIST

CONSTRAINT SUBLIST, TYPE=  6

                                BEGIN LIST 30331 2 30335 30333 0      0      1
30333 0 30331 30335                             5           0000000000000005
30335 1 30333 30331                   03601722689           0007250100072501 LIST ADDRESS= 30017
                                END LIST

CONSTRAINT SUBLIST, TYPE=  9

                                BEGIN LIST 30351 2 30367 30353 0      0      1
30353 0 30351 30355                             5           0000000000000005
30355 1 30353 30357                   03601722689           0007250100072501 LIST ADDRESS= 30017
30357 0 30355 30359                             1           0000000000000001
30359 0 30357 30361                             6           0000000000000006
30361 1 30359 30363                   03601722689           0007250100072501 LIST ADDRESS= 30017
30363 0 30361 30365                             4           0000000000000004
30365 0 30363 30367                            -1           7777777777777776
30367 0 30365 30351                             2           0000000000000002
                                END LIST

NO SUBOBJECTS
```

```
OBJECT STRUCTURE

                               BEGIN LIST 30035 2 30219 30381 1 30385     18
30381 0 30035 30051                        3         0000000000000003
30051 0 30381 30041                        1         0000000000000001
30041 0 30051 30217                        0         0000000000000000
30217 0 30041 30049                        2         0000000000000002
30049 1 30217 30203              02897079575         0007242700072427 LIST ADDRESS= 29975
30203 1 30049 30219              06487404013         0007275500072755 LIST ADDRESS= 30189
30219 1 30203 30035              06755839485         0007277500072775 LIST ADDRESS= 30205
                               END LIST

DESCRIPTION LIST

                               BEGIN LIST 30385 2 30481 30387 0       0       0
30387 0 30385 30389                        2         0000000000000002
30389 0 30387 30391                       -1         7777777777777776
30391 0 30389 30393                        7         0000000000000007
30393 0 30391 30403                        1         0000000000000001
30403 0 30393 30405                        8         0000000000000010
30405 1 30403 30415              10044174017         0007330100073301 LIST ADDRESS= 30401
30415 0 30405 30417                        9         0000000000000011
30417 0 30415 30419                        2         0000000000000002
30419 0 30417 30421                       10         0000000000000012
30421 0 30419 30423                        1         0000000000000001
30423 0 30421 30425                       11         0000000000000013
30425 0 30423 30435                        1         0000000000000001
30435 0 30425 30437                       14         0000000000000016
30437 1 30435 30479              10581044961         0007334100073341 LIST ADDRESS= 30433
30479 0 30437 30481                       15         0000000000000017
30481 1 30479 30385              11319242509         0007341500073415 LIST ADDRESS= 30477
                               END LIST

CCNSTRAINT LIST

                               BEGIN LIST 30401 2 30413 30409 0       0       1
30409 0 30401 30411                        1         0000000000000001
30411 0 30409 30413                        2         0000000000000002
30413 1 30411 30401              09943510715         0007327300073273 LIST ADDRESS= 30395
                               END LIST

CCNSTRAINT SUBLIST, TYPE=  2

                               BEGIN LIST 30395 2 30399 30397 0       0       1
30397 0 30395 30399                       -1         7777777777777776
30399 0 30397 30395                        2         0000000000000002
                               END LIST

CCNSTRAINT LIST

                               BEGIN LIST 30433 2 30469 30465 0       0       1
30465 0 30433 30443                        2         0000000000000002
30443 0 30465 30445                        6         0000000000000006
30445 1 30443 30467              10480381659         0007333300073333 LIST ADDRESS= 3042
30467 0 30445 30469                        9         0000000000000011
30469 1 30467 30433              10815925999         0007335700073357 LIST ADDRESS= 3044
                               END LIST

CCNSTRAINT SUBLIST, TYPE=  6

                               BEGIN LIST 30427 2 30431 30429 0       0       1
30429 0 30427 30431                        4         0000000000000004
30431 1 30429 30427              03903712595         0007252300072523 LIST ADDRESS= 3003
```

```
                                  END LIST

CONSTRAINT SUBLIST, TYPE=  9

                                  BEGIN LIST 30447 2 30463 30449 0      0      1
30449 0 30447 30451                           5            0000000000000005
30451 1 30449 30453                 03903712595            0007252300072523 LIST ADDRESS= 30035
30453 0 30451 30455                           1            0000000000000001
30455 0 30453 30457                           6            0000000000000006
30457 1 30455 30459                 03903712595            0007252300072523 LIST ADDRESS= 30035
30459 0 30457 30461                           4            0000000000000004
30461 0 30459 30463                          -1            7777777777777776
30463 0 30461 30447                           2            0000000000000002
                                  END LIST

CONSTRAINT LIST

                                  BEGIN LIST 30477 2 30513 30509 0      0      1
30509 0 30477 30487                           2            0000000000000002
30487 0 30509 30489                           6            0000000000000006
30489 1 30487 30511                 11218579207            0007340700073407 LIST ADDRESS= 30471
30511 0 30489 30513                           9            0000000000000011
30513 1 30511 30477                 11554123547            0007343300073433 LIST ADDRESS= 30491
                                  END LIST

CONSTRAINT SUBLIST, TYPE=  6

                                  BEGIN LIST 30471 2 30475 30473 0      0      1
30473 0 30471 30475                           5            0000000000000005
30475 1 30473 30471                 03903712595            0007252300072523 LIST ADDRESS= 30035
                                  END LIST

CONSTRAINT SUBLIST, TYPE=  9

                                  BEGIN LIST 30491 2 30507 30493 0      0      1
30493 0 30491 30495                           5            0000000000000005
30495 1 30493 30497                 03903712595            0007252300072523 LIST ADDRESS= 30035
30497 0 30495 30499                           1            0000000000000001
30499 0 30497 30501                           6            0000000000000006
30501 1 30499 30503                 03903712595            0007252300072523 LIST ADDRESS= 30035
30503 0 30501 30505                           4            0000000000000004
30505 0 30503 30507                          -1            7777777777777776
30507 0 30505 30491                           2            0000000000000002
                                  END LIST

NO SUBOBJECTS
```