PIXIE - A NEW APPROACH TO GRAPHICAL MAN-MACHINE COMMUNICATION

N.E. Wiseman, H.U. Lemke, and J.O. Hiles

1. Summary Interactive computer graphics is popularly thought of as an expensive luxury. There is a general agreement that interactive graphics provides a new and useful tool but widespread uncertainty regarding its economics and even it ← ?

The PIXIE system represents an attempt to single out a few applications for which a "proper role" for computer graphics can be identified and to provide a framework within which these applications may be implemented easily and economically. Particular emphasis has been placed on reducing operating costs. For example, the satellite processor supporting the graphic terminal does not normally remain connected and logged in to the main computer throughout a console session but rather will log on occasionally to start some program and then log off again. Such programs running in the main computer do not interact with the terminal during execution and may thus be arranged to operate efficiently and cheaply as remote batch jobs. On the Cambridge Multiple Access System this may be done in what is referred to as "normal mode" working, as opposed to "expensive mode" in which a user program is permitted to interact with a terminal. Because of the poor economics, expensive mode programs are limited to a maximum core space of 8K whereas normal mode programs can be large (up to 40K). Thus PIXIE is able to connect with large application programs but may invoke runs only to completion (or to some preset stop).

2. Introduction PIXIE is the name we have chosen for an operating system for the graphic terminal on the Cambridge Multiple Access System. It provides one of several man-machine interfaces with a data management system we are setting up for a number of applications in computer-aided design. In our view the computer should participate in all stages of design, development, production and administration since this eliminates many of the manually introduced clerical errors in connecting one task to the next and it also reduces the latency time of working documentation. We hope that our data management system is suitable to connect any task to any other task even though we may not have known about these tasks when we designed the system, and even though the people who use it may not know one anothers tasks in any detail.

To understand why we have designed PIXIE the way it is, it seems appropriate first to say how we think our data management system will serve the designers' needs. To be specific we shall consider electronic circuit design as the activity supported by our data management system although there are a number of application areas in which we think it could be usefully used.

The designer works at first with gross concepts and later with more and more detailed analysis and appraisal. A circuit designer for example would start with qualitative appraisals of a few topological arrangements. Next he needs a few crude analyses made to give spot

N.E. Wiseman, H.U. Lemke and J.O. Hiles are at present with the University Mathematical Laboratory at Cambridge.

checks quickly. Finally he searches for an acceptable set of parameter values and evaluates the effects of environmental changes etc. The amount of computing per decision taken increases dramatically as the task proceeds. When first drawing a proposed circuit he may choose a capacitor "there" on the basis of a hunch or his past experience. Later he may decide between one or two stages of amplification on the basis of a spot check analysis made for a particular guess at parameter values - this may invoke 10 seconds computing or so. Later still he may take 20 minutes computing to choose the best value for a resistor in the emerging design. At any time he (or someone) might decide to try to lay out the circuit on a p.c. board (or plan a set of masks if it is an i.c.), attach a memo to it and file it away, send it to a colleague in Timbucktoo or scrap it and start over again.

The response time which the designer experiences when working on-line to the computer can obviously not be less than the amount of computing time he calls for, but, if he is not to hang about needlessly, should not be too much longer. Fast response requires a certain sacrifice in computer operating efficiency and we endeavour to operate using the cheapest mode of computing consistent with the task in hand. When drawing and experimenting in a qualitative way with schematic diagrams at the graphic terminal full conversational facilities with the satellite computer are necessary but only occasional needs for access to the main computer arise. Spot check analysis however may need frequent access to the main computer and may not require the satellite at all. Finally, long bursts of computing for more detailed work can be carried out as remote batch or off-line jobs on the main computer. In this way we expect to achieve economical computing without compromising the needs of the designer.

The data passing from one job to another is formatted and structured in a task-independent manner. One job may write its results, as a data structure, to the file with no knowledge whatever of the next process to be carried out. PIXIE for example communicates with the main computer solely in terms of such data structures and needs no information about the processes to be applied to the data in the main computer (or even the area of application, although certain assumptions are built into the program).

Data structures represent with complete generality interrelated items of information in a machine readable form. For example, an electronic circuit is an array of nodes and branches associated together in some way (topologically, in the form of connectedness, geometrically as a schematic drawing or the layout of its components on a board and so on) and may be modelled with arbitrarily high precision by a data structure. Data structures are made from <u>elements</u> formed from a number of words of memory containing both the items of information and the relations with other items. In PIXIE and the data management routines it couples with, data structures are built and handled with RSP, a package of subroutines for organising ring structures (1) and in this format an element typically requires between 10 and 40 words of memory. A complete model for an electronic circuit would involve many elements, normally totalling 3000 to 20000 words of memory and an obvious function of PIXIE, and the other man-machine interfaces to our data management system, is to provide means by which a designer can create and control that much interrelated data in a meaningful way (i.e. conveniently and with low error rate).

With this preamble it is hoped that the following description of PIXIE will be seen in its proper context.

3. The PIXIE System

We can regard PIXIE as a format translator providing two-way communication between some pictorial and verbal language natural to the man and RSP data strucutres utilized by application programs in the main computer. It operates interactively and the man is able to see immediately, and understand easily, the results of his actions. He can, for example, construct a schematic for an electronic circuit on the screen of the CRT by "drawing" it, almost as readily as he might draw it with pencil and paper (and eraser). While he does this PIXIE builds a complex data structure modelling his work, a structure which he never needs to see or understand but over which he has total control.

The major routines in PIXIE are shown in figure 1 together with data paths connecting them together.

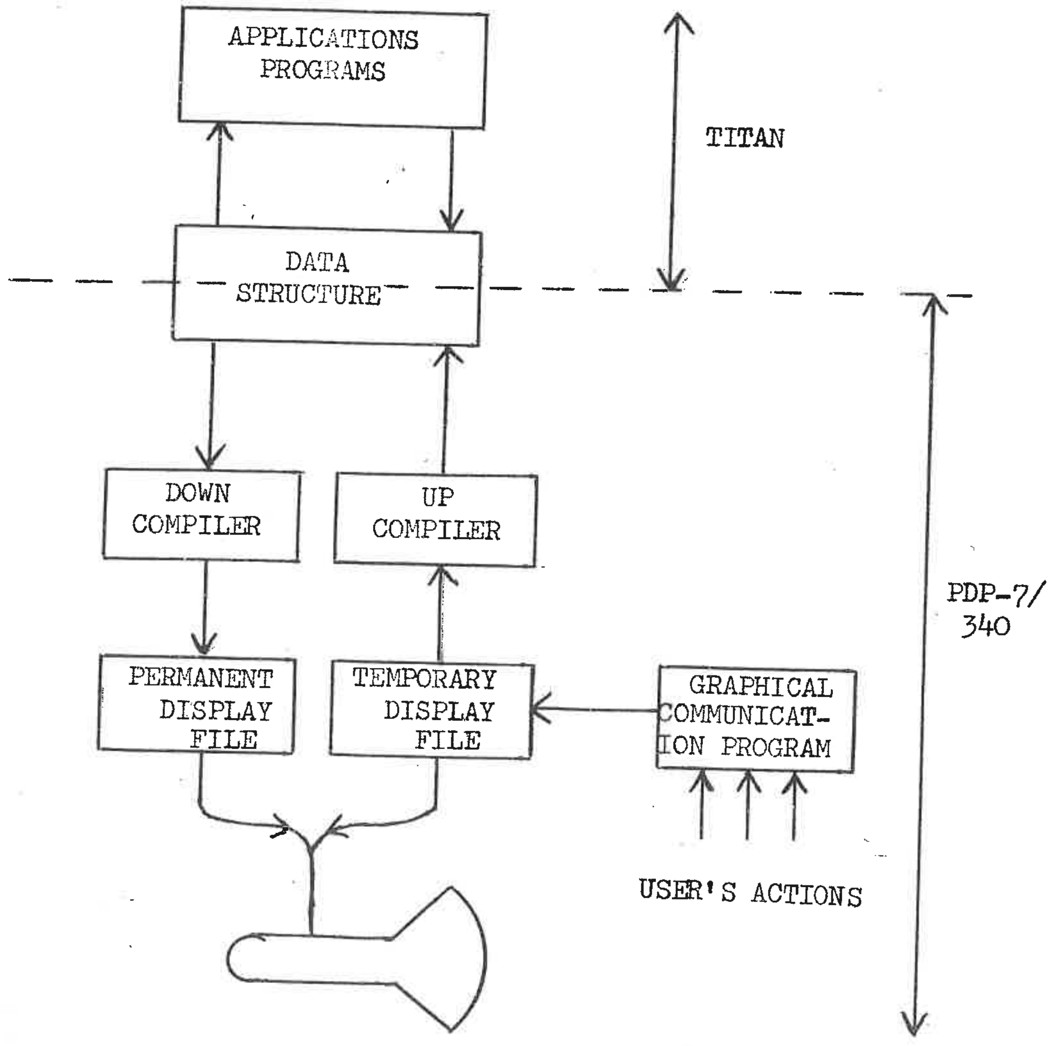

FIGURE 1

In discussing the function of each of these routines we shall refer to the data input to a routine as its source and the data output from a routine as its object. Thus the graphical communication routine has the user's actions, and corresponding computer interrupts as source and the temporary display file as object.

The user's actions comprise typing commands and messages on the teletype, pointing the lightpen at lightbuttons on the screen and trailing a tracking cross around the screen by moving the lightpen over it. Two sets of lightbuttons are provided:

(1) The command lightbuttons which are for inducing gross actions such as changing mode displayed down the right edge of the screen.

(2) The control lightbuttons which are used frequently within a mode to carry out a sequence of operations. These buttons are displayed around the tracking cross and move about with it so that the user's hand is always close to these buttons when he needs them. To avoid clustering a large number of control buttons around the tracking cross, it is arranged that only buttons for those actions which are legal at any given time are displayed and also that the user may select different sets of legal buttons by pointing at one of them (which acts as a sort of rotating switch for the rest).

To illustrate an action on the screen consider how the user might draw the fragment of circuit shown in figure 2.

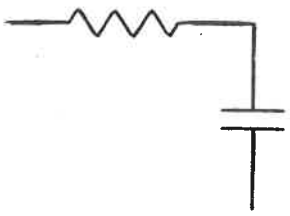

FIGURE 2

He points the lightpen at command buttons EN (end previous work, make clean start); DR (enter drawing mode); and ST (set constraints on line-drawing to be straight horizontals or verticals). Next he picks up the tracking cross and moves it to a position on the screen where he wants to start drawing and points at the control button S (start drawing). When he next tracks the cross, in say roughly a W $\rightarrow$ E direction, the graphical communication routine constructs segments of line in the temporary display file in exactly W $\rightarrow$ E direction (since horizontal-vertical constraints are selected) and displays them on the screen. When he wants to introduce a symbol he selects the appropriate control lightbutton (in this case R for resistor) and the symbol is inserted in the display file in the appropriate orientation. As he continues to track, further line segments are added, changing direction (in this case to vertical) when his hand motion strays by more than about a quarter inch from the line being drawn. Next he selects a capacitor by pointing at the control button C: note that it is inserted in vertical orientation because he is drawing vertically now - and finally draws a further short segment of line and finishes by pointing at the button F. Up to this time he has been able to erase and correct the drawing simply by retracing the pen over its path (a technique we call "undrawing"). The graphical communication program does this simply by maintaining a stack recording the items to be put in the temporary display file and pushing or popping the stack according to the motion of the pen.

When the F control button is hit, radical changes in the format of data in core occur, although the picture on the screen remains unchanged. The up-compiler (2) translates the information from the temporary display into a structured form representing the nodes and branches, lengths and positions of line segments and so on and appends it to the existing data structure (unless of course, as is the case for the example above, there is no existing structure, in which case a new one is started). The up-compiler has display file as source and RSP data structure as object and is primarily concerned with extracting features about the circuit from what is in the temporary display file and the existing data structure. It is at this point for example that notions of connectivity first enter into the format of the data being put together in the machine. We shall see later how the user's actions make reference to such features. After the amended data structure is formed the down compiler (3) is entered to generate a new display file (the "permanent display file") after which the temporary display file is emptied. The drawing action may then be recommenced. Information in the data structure is stored in a hierarchical form (circuits, branches and nodes, squiggly lines) and the permanent display file is linked with the data structure at the various levels to assist in the identification of pen hits from the picture. This is the purpose of the permanent display file and, until some action arises which needs such identification aids, the appearance of the picture is unchanged by transitions from temporary to permanent status.

Suppose next that the program enters pointing mode (user points at the command button PO). The menu of control buttons now changes to display the options available in this mode, such as T (for track) and E (for erase). When a hit from the permanent display file occurs the context of the hit is displayed by causing all or part of the picture to blink on and off at about twice per second. The context can be moved up and down in the hierarchy by hitting further command buttons until the desired picture part is blinking. For example, a line,or node which contains the line, or a subcircuit incorporating that node can be identified, and made to blink, when the line is pointed at and this picture part can then be the operand for a further function. Examples of functions available are

(1) track the picture part around on the screen (point at T and track the cross).
(2) erase the picture part (point at E)
(3) give a name or label to the picture part (type it in on the teletype.)
(4) attach arbitrarily formatted messages to the picture part (type them in with particular prefix characters)
(5) list all the data about the picture part on the teletype (type a space).

When tracking or erasing, a new permanent file is compiled for each change to the data structure. This frequently takes a few seconds during which time the permanent display file is not displayed. It was to avoid introducing such delays into the drawing action that a temporary display was generated in DR mode by the graphical communication routine. This approach is, we believe, novel and certainly makes for a most pleasing action.

Provided nothing on screen is blinking, messages entered on the teletype are normally regarded as commands. In addition to the operations of the command lightbuttons it is possible to annotate the picture with all its labels (type LABEL), remove labels from the picture (UNLABEL) or freeze PIXIE in preparation for communication with Titan (type TITAN).

The control flow for PIXIE is quite complicated but it may be of interest to outline the operation of the main routines. Flow diagrams for the gross actions of the main program and interrupt routines are shown in figures 3 and 4 respectively. In most cases the number of instructions obeyed to service any given interrupt is kept to the absolute minimum and all long routines are designed to be interruptable. If an interrupt requires to initiate a long program then a request is set in the interrupt routine and sensed later in an interruptable control program. Two chains of registers for holding requests and locks are used to remember requests and activate the programs at the appropriate time. The important ones are indicated in figures 3 and 4. The action is rather like that of a dual processor sharing the use of core (one processor being the PDP-7 the other the 340 display). It has been with the design and debugging of these administrative features that we have had the greatest trouble in getting PIXIE operating. Real-time bugs, always difficult to uncover, are much more troublesome in a dual processor situation.

4. Utilizing PIXIE originated data A TITAN command to PIXIE followed by a link write or read command from Titan can be used to transfer a data structure between the two machines. For example a structure built in the PDP-7 can be sent to a file in Titan or retrieved from a file, edited and returned. In order to carry out some useful process on such a structure an application program would normally pick it up from file, run the process and write a results file, also in the form of a data structure. To facilitate this an RSP processor exists for Titan as well as the PDP-7 and the operations of extracting data from or writing data to an RSP structure are carried out using a MOUSE routine (after Ross (4)). The mouse may be called by FORTRAN or assembly-code programs and can be steered over a victim structure by a list of parameters, rather in the manner of a treasure hunt.

As an initial application of PIXIE a generalized circuit analysis program is being written which can input and output data structures. This program need not be machine dependent in the way that RSP is and in fact is being written almost entirely in ASA FORTRAN. A very real possibility therefore exists of transferring the large analysis program from the Titan to some other computer and developing the mouse subroutine as circumstances permit. We have not allowed the detailed implementation of the data structure package to manifest itself in the FORTRAN code but have designed calls to the mouse subroutine that are as general as possible. Thus only the basic mechanisms of structure traversal are being written in machine code so that a FORTRAN program may be produced which refers to any relational concepts contained within the data structure that are of interest to an application program. In the case of an analysis program for example the mouse will initially be situated on an element and will be asked to find another element of

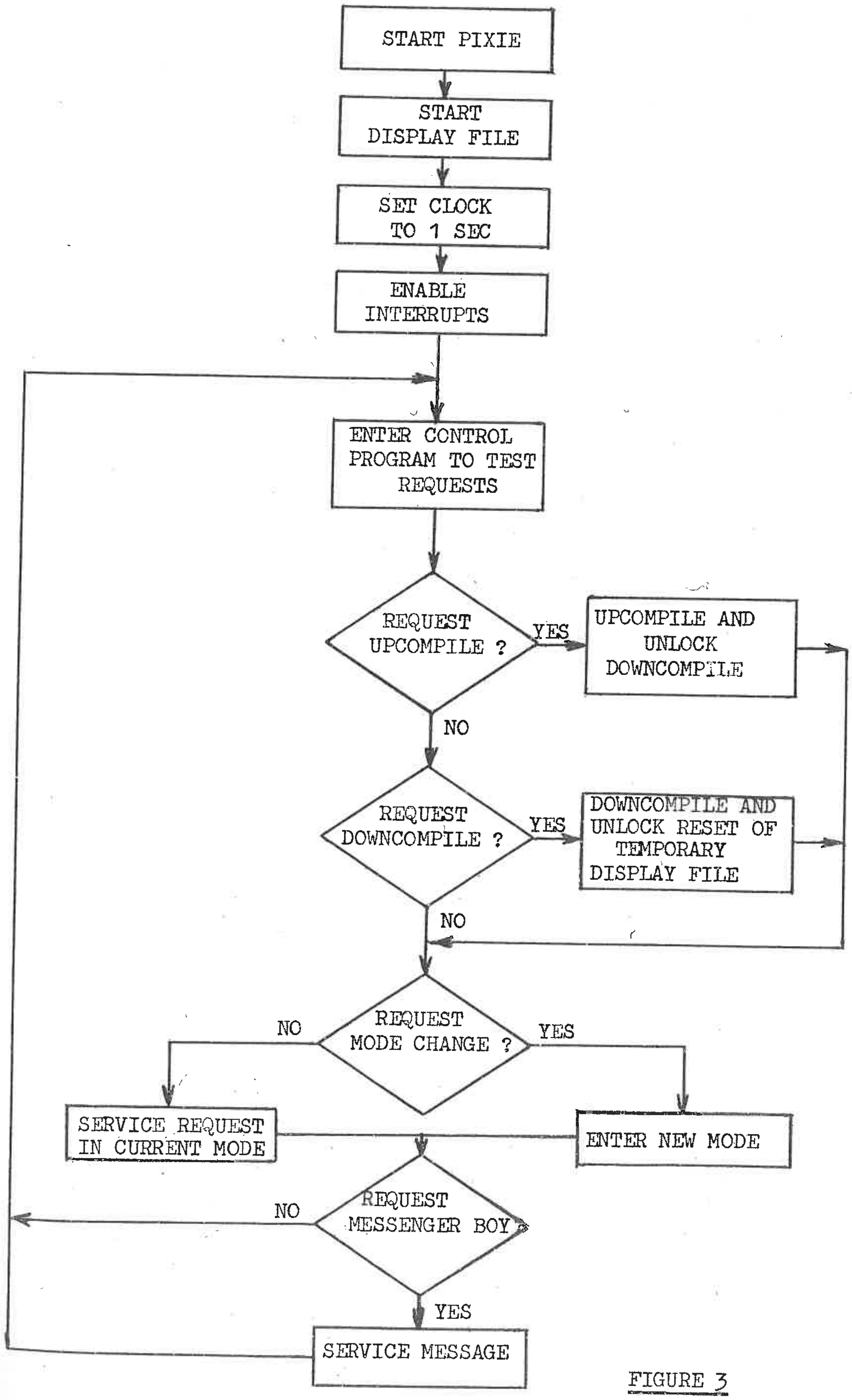
START PIXIE
START DISPLAY FILE
SET CLOCK TO 1 SEC
ENABLE INTERRUPTS
ENTER CONTROL PROGRAM TO TEST REQUESTS
REQUEST UPCOMPILE ?
YES
UPCOMPILE AND UNLOCK DOWNCOMPILE
NO
REQUEST DOWNCOMPILE ?
YES
DOWNCOMPILE AND UNLOCK RESET OF TEMPORARY DISPLAY FILE
NO
REQUEST MODE CHANGE ?
NO
YES
SERVICE REQUEST IN CURRENT MODE
ENTER NEW MODE
REQUEST MESSENGER BOY ?
NO
YES
SERVICE MESSAGE

FIGURE 3

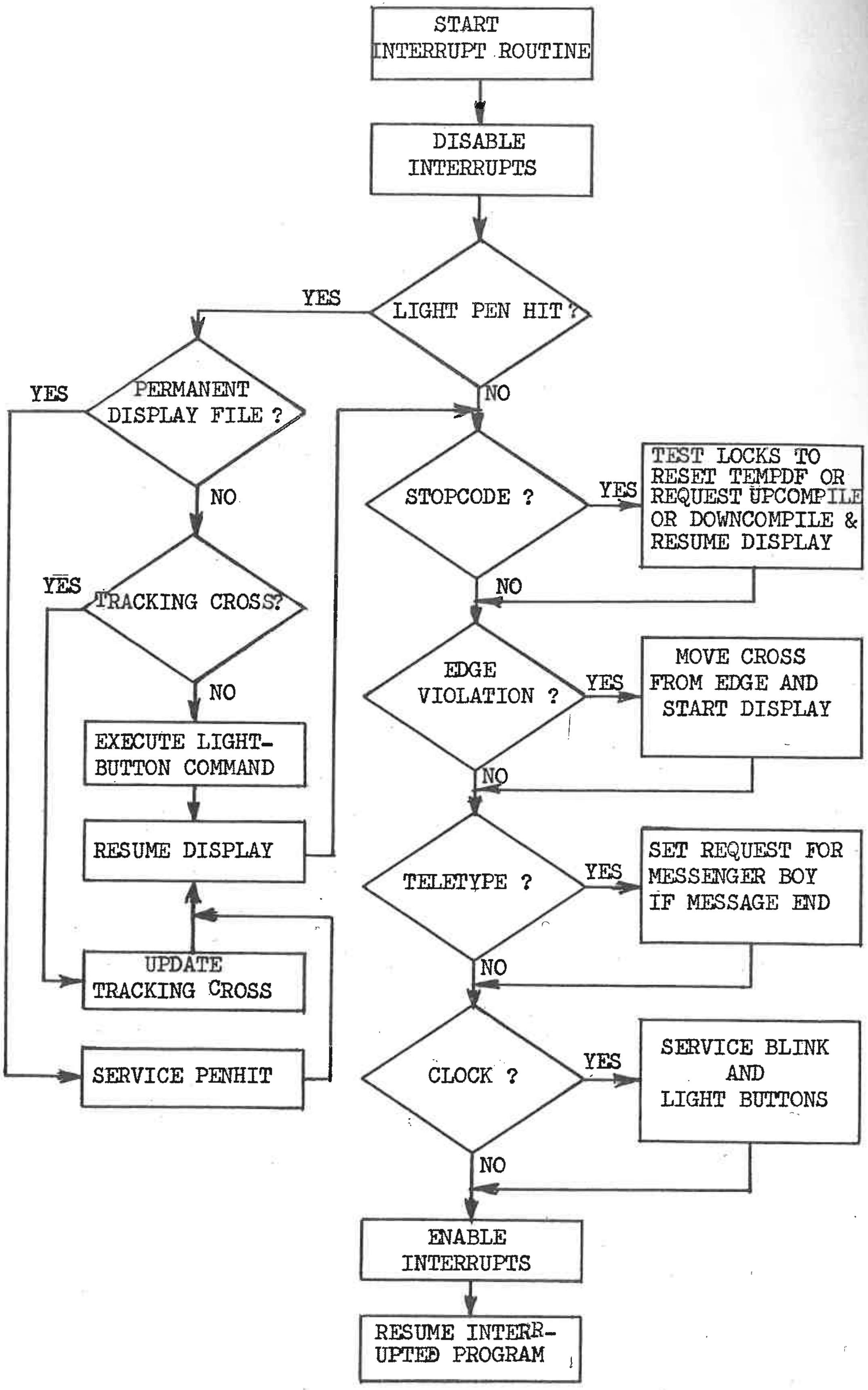
START
INTERRUPT ROUTINE
DISABLE INTERRUPTS
LIGHT PEN HIT ?
YES
NO
PERMANENT DISPLAY FILE ?
YES
NO
TRACKING CROSS?
YES
NO
EXECUTE LIGHT-BUTTON COMMAND
RESUME DISPLAY
UPDATE TRACKING CROSS
SERVICE PENHIT
STOPCODE ?
YES
TEST LOCKS TO RESET TEMPDF OR REQUEST UPCOMPILE OR DOWNCOMPILE & RESUME DISPLAY
NO
EDGE VIOLATION ?
YES
MOVE CROSS FROM EDGE AND START DISPLAY
NO
TELETYPE ?
YES
SET REQUEST FOR MESSENGER BOY IF MESSAGE END
NO
CLOCK ?
YES
SERVICE BLINK AND LIGHT BUTTONS
NO
ENABLE INTERRUPTS
RESUME INTERRUPTED PROGRAM

FIGURE 4

which it is the son. The argument list of the subroutine hands back the identifier for the element found and the FORTRAN code may decide as the result of a test to probe still further or try another father of the original element. In this way the mouse may be guided from an instance of a node to the instances of the various branches connected to it, systematically discovering and handing back the nature of the connections.

Much of the data structure ceases to be of value in the analysis once the connectivity has been expressed in terms suited to fast numerical techniques, but it provides a side benefit in greatly simplifying the procedure required for choosing a tree. This stems from the fact that connectivity is explicitly represented by the structure.

The complicated features of non-linear circuit elements are handled by sections of the original data which are retained to be accessed by the control program. Small routines will be provided to feed back the current values for the circuit Jacobean and the break point situations for piecewise linear elements. The tabular form of piecewise linear characteristics allow them to be treated as multi-dimensional arrays which are easy to represent as a data structure and access through a FORTRAN called subroutine. However, such problems are separate from the primary tasks of the Mouse and merely indicate the rather obvious advantages of extending data structure principles to the realm of control flow within an application program.

5. References

1. N.E. Wiseman and J.O. Hiles "A Ring Structure Processor for a small Computer"; Computer Journal, Vol.10, p.338. February 1968.

2. H.U. Lemke "The PIXIE up-compiler". Paper in preparation.

3. N.E. Wiseman "A note on compiling display file from a data structure"; Computer Journal, Vol.11, p.141. August 1968.

4. D.T. Ross "The AED approach to generalized computer-aided design" MIT Electronics Systems Laboratory technical Memorandum ESL-R-305, 1967.

5. D.S.Evans and J. Katzenelson "Data Structure and Man-machine Communication for Network Problems" Proc.IEEE, Vol.55, p.1135, July 1967.