LBL-580

PICASSO:
A GENERAL INTERACTIVE GRAPHICS MODELING PROGRAM

Donald M. Austin and Harvard H. Holmes

January 1972

AEC Contract No. W-7405-eng-48

LBL-580

PICASSO

UNIVERSITY OF CALIFORNIA
Lawrence Berkeley Laboratory
Berkeley, California
AEC Contract No. W-7405-eng-48

PICASSO:
A GENERAL INTERACTIVE GRAPHICS MODELING PROGRAM

Donald M. Austin
and
Harvard H. Holmes

Picasso

January 1972

PREFACE

This report is intended to serve as a user's manual for PICASSO. The detailed description of the concepts involved and the methods used in implementing those concepts will be treated in a subsequent paper and are incorporated in the Ph.D. thesis of one of the authors (HHH).

The program evolved from earlier work on a graphics program for printed circuit board design which had a fixed library of elements. The need for a dynamic graphics program (i.e., one which allows the creation of new elements as required) soon became obvious, and the generality of such a program for mathematical modeling was a major factor in the development of PICASSO.

It was thought that the present stage of development of this modeling program had reached a point that a report such as this would be useful in creating a broader interest in the field. However, work continues to further generalize the program into such areas as computer animation, 3-D display and modeling, and a more general interface to available "analysis" routines, such as the available compilers and interpreters.

It should be emphasized that the program is continually changing as new ideas and applications arise. For example, in order to create a modeling system which is complete in some sense, PICASSO was designed explicitly to interface with the CDC analysis program MIMIC. The next step, already under investigation, is to allow for user-defined output templates to specify the form of the output of a model. This feature, along with the capability for multiple definitions of an element, will allow for several types of analysis to be done on a single model.

Another innovation being studied is the provision for "robot" commands, with which the user may define a sequence of commands to be carried out on demand, in much the same way as mechanical "robots" are programmed. For this reason there will usually be several versions of PICASSO available, one of which is relatively "bug-free."

The authors wish to gratefully acknowledge support from the Math and Computing Division and the Electrical Engineering Department of the Lawrence Berkeley Laboratory, and the Department of Electrical Engineering and Computer Science of the University of California, Berkeley. Specifically, we wish to thank Ivan Wood and Horace Warnock of EE Drafting for support and many suggestions; Michiyuki Nakamura, Richard LaPierre, John Mendes, Don Evans and Frank Neu of EE Research and Development for many suggestions and interesting applications; Professor Baskin, EECS Department, UCB, James Baker and Carl Quong, Math and Computing, LBL for support and encouragement; and Norma Hart for preparing the manuscript.

TABLE OF CONTENTS

I. INTRODUCTION

An interactive graphics modeling program is designed to be an interface between the rich and complex mathematical language of the scientist and the highly structured language of machines. For many problems a diagram or schematic serves to state the essential details of the problem and is usually the medium of communication between those interested in the problem. A modeling program is intended to allow diagramatic representations to also serve as the medium of communication between man and machine.

PICASSO is a step in this direction. The first version of PICASSO described here allows a user to create and define symbols, hook them together in some meaningful fashion to define a model, and analyze the model, viewing the results almost immediately. The modeling system consists of PICASSO for the graphics, an analysis program, such as the CDC analysis program MIMIC, and the PTSS time-sharing system for the BKY operating system control. These three parts from a system which allows a relatively computer-naive problem-solver to formulate a problem, build a diagramatic model, analyze the model, view the results, change the model, reanalyze, etc., all in a couple of hours of interactive computing time. Furthermore, once a suitable library of basic elements has been created, there is no need for an understanding of any programming language by subsequent users interested in the same class of problems. The problem can be formulated graphically and the results presented graphically, and no holes need be punched in little cardboard rectangles in the process.

What is necessary, of course, is a knowledge of the graphics and text-editing commands required to build the diagram of the model. These commands are the subject of this abbreviated write-up. It is hoped that anyone with a good knowledge of the problem he wishes to solve, some small amount of artistic ability and a sense of charity toward a new system can sit down at the Vista console and master the necessary techniques in an hour.

The graphics program described here is not specialized to model building. The initial impetus for PICASSO arose from a need for a general graphic input program to function as a drafting table, capable of handling pictures of fairly high quality and accuracy. The idea of modeling is a broad one, however, and very soon the designer discovers that his picture contains much more information capacity than just the picture itself. The analysis phase of PICASSO was designed to allow that information to be converted into useable output.

The terminology used in this paper was created to make the ideas more precise. The idea behind a "picture made of other little pictures with some more lines drawn in and some extra labels on the important lines" is certainly not a profound one, but the term "Macro definition" may sound formidable, even though they mean very much the same thing. The important point is that if one has a good idea of the problem he wishes to solve, be it the analysis of a complicated logic diagram or the creation of a nice looking flow chart, an hour spent experimenting with the graphics should provide enough expertise to get the job done.

The paper is organized as follows: Section II contains a description of the data structure and most of the terminology used to

define the concepts. Section III is an abbreviated description of the commands executed by the program, and the appendices describe components of the hardware and operating system details necessary in getting the program on the air.

This writeup is not intended to be complete. The authors gladly offer their assistance in setting up all the details necessary to use the system, and maintain an active interest in any problems for which the system may be useful.

Also, in the examples extensive use is made of the analysis program MIMIC. It should be kept in mind that PICASSO's output is determined pretty much by the input, so that any analysis program, such as a WIREWRAP program, a circuit analysis program or a least-squares fitting program, may be the appropriate analysis program. PICASSO merely converts the diagram of the model into user-specified card images.

II. THE DATA STRUCTURE

The purpose of any graphic modeling system is to facilitate the creation of a diagramatic model of a problem which contains all the information essential to specifying the problem. By creating some fundamental objects and hooking them together to form a schematic representation of the problem (such as a logic diagram or a flow chart), the model builder may specify everything essential to the analysis, and the modeling system is expected to provide the uninteresting details, such as do-loop indices or statement numbers. In order to implement such a scheme, the form of the essential parts of a model must be specified by a data structure consisting of a few well-defined basic elements available to the model builder as building blocks.

The building blocks of PICASSO are called elements (or, sometimes, blocks). Collections of elements form broader structures which must be defined for convenience in storing and retrieving - these broad structures are described in the section on Libraries; in this section the composition of the fundamental building blocks is treated in detail.

An element consists of three components which must be specified by the creator; the Name, the Symbol, and the Definition.

A. The Name of an Element.

The name consists of up to seven characters typed in on the teletype. This name is used to refer to all aspects of the element.

B. The Symbol of an Element.

The symbol is a visual representation of the element created by the user to represent the element in a diagram. A symbol may consist of lines drawn in with the light pen, labels defining the parameters

of the element, and other symbols. The shape of a symbol is limited only by the artistic ability of the user and the finite core of the computer.

C. The Definition of an Element.

Familiar symbols such as NAND gates, circuit elements or flow chart symbols all carry with them an implied definition understood by those who use them. Similarly the symbols used in PICASSO are defined by their creator in one of four forms:

1. No definition: If the aim is simply to draw a picture, no definition need be supplied for an element.

2. Text definition: If the element is to represent a basic function to be performed, such as integration or logic gating, the element is thought of as a primitive element and is defined in terms of text. The text definition is typed in on the teletype in any form desired - the assumption is that the text will make sense to some analysis program such as MIMIC or a Fortran compiler. The primary use of a text definition is to specify one or more outputs as a function of some inputs. A simple example of such an element is the ADDER shown in Figure 7, where the output labeled Out is defined to be the sum of the inputs In1 and In2.

3. Macro Definition of an Element: After some primitive elements with suggestive symbols and text definitions have been defined, more complicated elements may be constructed by combining the symbols for the primitive elements into a simpler symbol whose definition refers back to the text definitions of its components. The labels on a macro definition may best be thought of as actual parameters for the formal

parameters of the primitive element.

An important feature of elements with macro definitions is the complex heirarchy which can be built from the primitive elements. Another symbol may be defined as a macro definition in terms of symbols which themselves have macro definitions, until a complex element defined in this way contains a large amount of information compressed into a single block. The heirarchy may be nested to a level 128 deep in the current version of PICASSO.

4. An Empirical Definition of an Element: An empirical definition is a special type of definiton which allows for hand-drawn or external input curves to be processed. This type of definition consists of lines defining a curve, and a set of labels determining the scale of the curve on the horizontal and vertical axes. This feature allows for models in which the input to a given element can be specified empirically, through digitized data curves read in from an external device, such as a card reader, or hand-drawn sketches of curves whose analytical form is unknown. Figures 11a, 11b, 11c, show an example of an element defined empirically, and the output produced from it.

D. A Model.

A special type of element is a model, which is simply any element with a macro definition which the model builder wants analyzed - i.e., a diagram defined by the user which will be transformed into card images suitable for input to some analysis program. In Figure 9 is an example of a simple model built from primitive elements, and the output produced by PICASSO's analyzer is shown in Figure 10.

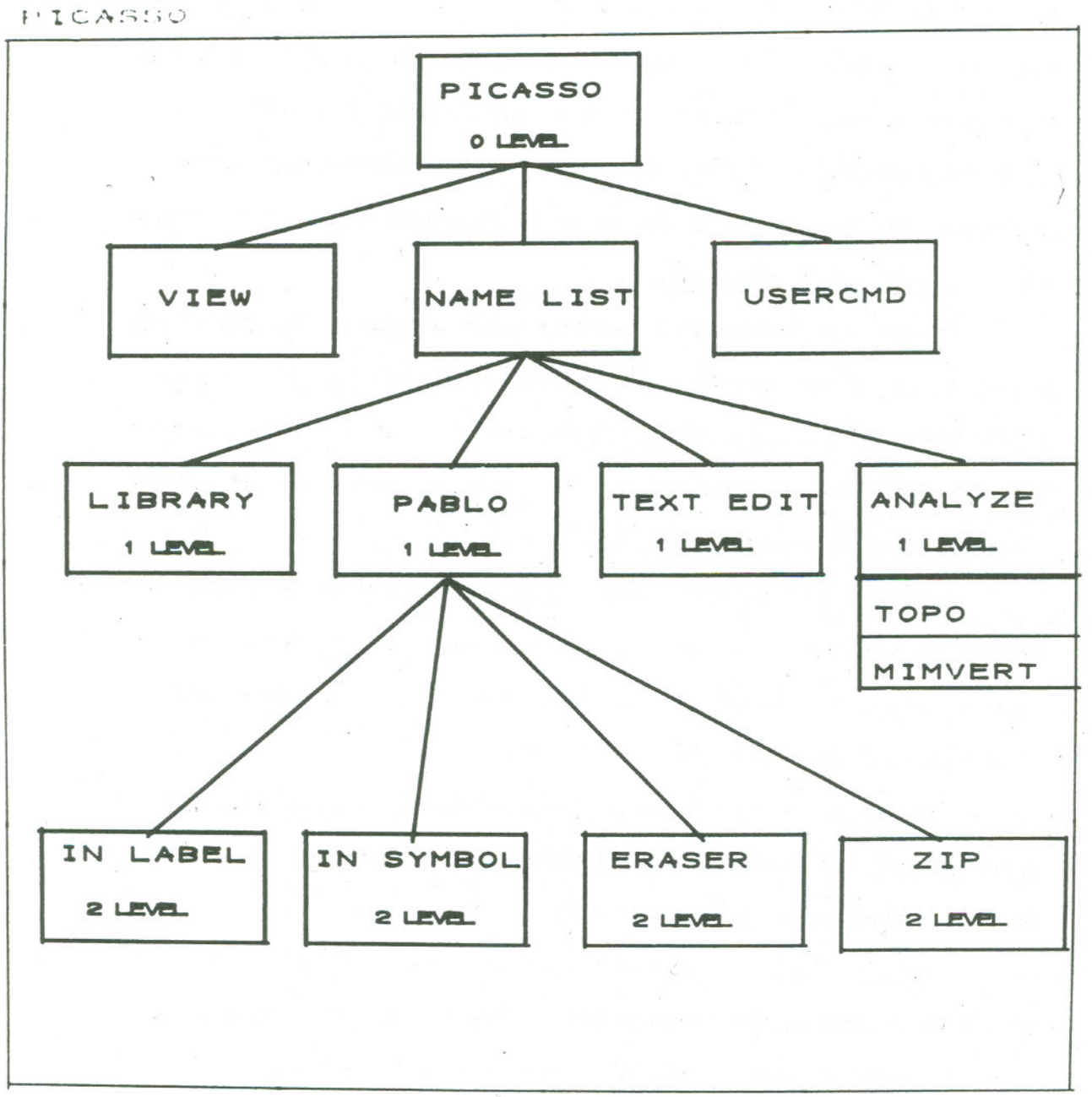


Fig. 1. Diagram of the command structure of PICASSO, showing the various editors and the flow of control.

III. OPERATING PROCEDURE

The procedure for operating PICASSO is described in an abbreviated form on the following pages. Understanding the operation of a graphics program by reading pages of text is not an easy task - nor is it a necessary one. A demonstration of the program in operation is worth many thousands of words (at the rate of a thousand words per picture). The writeup is useful as an introduction and as a reference.

A. The Comment Functions.

There are five helpful comments which appear at the top of the screen to guide the user through the command structure. (See Figures 2-5). When the going is smooth, these comments may be safely ignored; but when something strange occurs, they usually provide enough information to steer the user in the right direction.

Comment 5: The top comment displays the particular editor exercising control of the program at the moment (see Figure 1 for the command structure and the various editors). The editors have been given names to designate their functions.

1. Picasso: Picasso controls the zero-level commands which decide which part of the data structure (symbols, definitions or libraries) is to be operated upon. (See Figure 2).

2. Pablo: Pablo is the picture editor wherein symbols, macro and empirical definitions are constructed. (See Figure 4). Pablo calls some subordinate editors - In Label, to input labels; In Symbol, to input previously-defined symbols; Eraser, and Zip to perform specific tasks in constructing pictures. Each of these sub-editors has a small set of commands appropriate for executing their functions.

3. Text Edit: The Text editor controls the creation of text definitions and the editing of the text output from the analysis of a model.

4. Analysis: The analysis editor performs the topological analysis of a model, does the parameter substitution and expands the text into usable form.

Before transferring control to the appropriate editor, the Name List editor takes control for the building of a list of elements to be edited. While in the Name List editor, comment 5 indicates the command to be executed upon the list of names being created.

Comments 1-4 appear below comment 5 and may change with every command executed. In general, they are used as follows:

Comment 1: The leftmost comment is usually the name of the element being edited at the time. Thus if one is drawing the symbol for an element named "ADDER," comment 1 will read "ADDER."

Comment 2: This comment usually informs the user of the next normal action to be taken. For example, if the initial point of a line has been entered, comment 2 will read "LINE, PT 2," indicating that the second point of the line is expected next.

Comment 3: Comment 3 varies greatly in meaning. Often it is used in connection with comment 2 to indicate how the expected action may be taken. For example, if a label is to be entered, comment 2 will read "TYPE LABEL" and comment 3 will read "TELETYPE" (just in case a user may feel inclined to type the label on the function keyboard or something). A perhaps more useful function of comment 3 is to indicate an alternative action which will override the action already taken.

For example, when comment 2 reads "LINE, PT 2" indicating that the initial point of a line has been entered, comment 3 will read "IKEY=PT 1" to indicate that if one does not wish to finish drawing the line begun at the first point, he may override that point by hitting the red interrupt key on the function keyboard (called the IKEY) to discard the first point.

Comment 4: This comment informs the user of the procedure for exiting the current editor and returning to the editor which got him there. For example, in the Text editor, comment 4 reads ";R=EXIT" to indicate that by typing ;R on the teletype, one may exit from the Text editor and return to Picasso for another command. Another example is the comment "STORE-EXIT" in the picture editor Pablo. (See Figure 4). This comment indicates that in order to return to Picasso, one must execute the STORE command in Pablo, to store the current symbol away in the data structure before editing another.

It is recognized that one can say only so much in 10 letters or less, and that the comments may be somewhat vague at times. A little practice running PICASSO is the best cure for the problem.

B. The Command Structure

The command structure of PICASSO is represented in Figure 1 in terms of a level diagram showing the flow of control from the zero-level command interpreter Picasso to the various subordinate editors via the Name List editor. The level structure was created to simplify the conceptual control flow of the program and to allow display of only those commands which are relevant at the time. For example, the one-level command to "draw a line" makes sense only after a zero-level

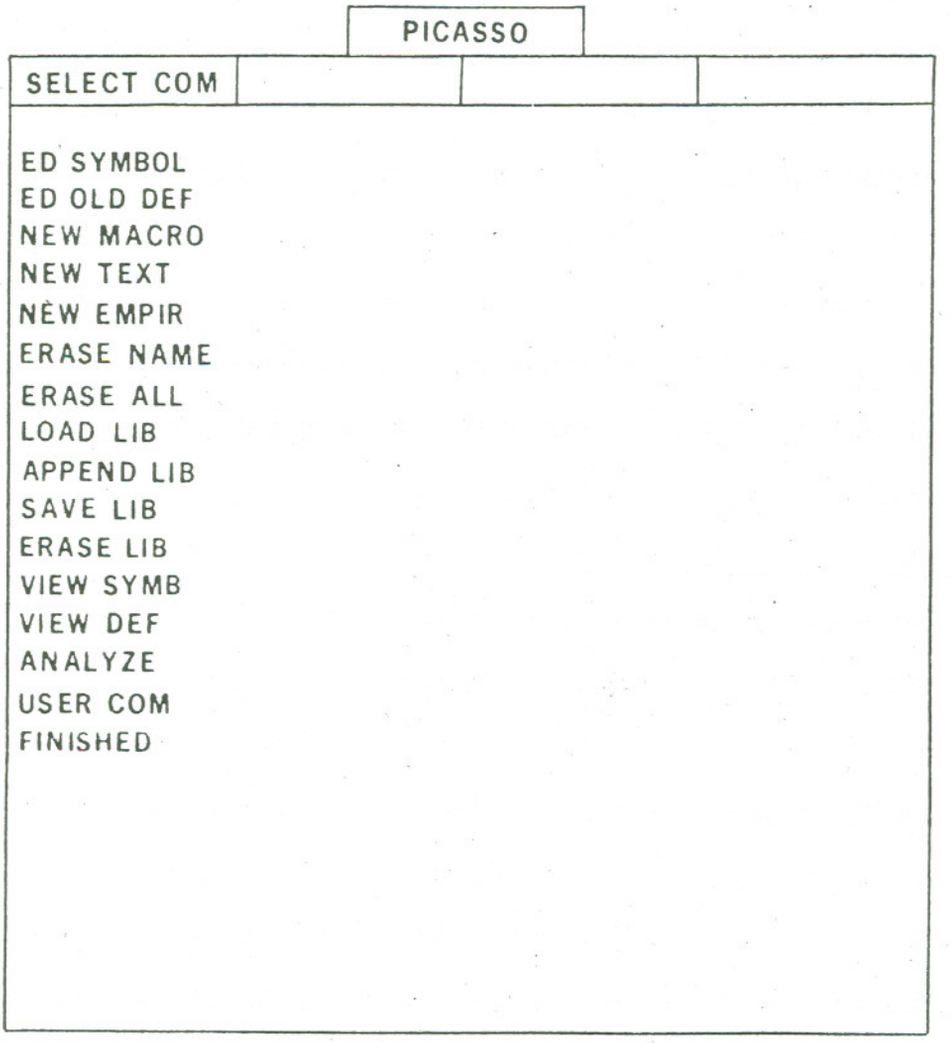


Fig. 2. The initial configuration of PICASSO the zero-level command editor Picasso waits for a light pen command.

command, such as "edit a symbol", has transferred control to the picture editor, just as a one-level command "start the engine" makes sense only after the zero-level command "get into the car" has been executed.

The commands associated with each editor are discussed in some detail in the following sections. These commands, and their associated comments, are intended to make the control flow obvious to the user after a little experience; the terminology, such as zero-level, one-level, etc., is not essential to using the program and should be ignored if it tends to obscure rather than enlighten.

a. Zero-Level Commands in Picasso:

The zero-level commands appear on the screen as in Figure 2, where comment 5 indicates the editor is Picasso. The commands are as follows:

1. ED SYMBOL: transfers control to Name List editor, where up to seven symbols may be selected for editing, then goes to Pablo.

2. ED OLD DEF: transfers control to Name List editor, where one previously defined element may be selected for editing. If the element has a Macro or Empirical definition, control goes to Pablo; if the element has a Text definition, control goes to the Text editor.

3. NEW MACRO: transfers control to Name List editor, where one element may be selected for the creation of a new Macro definition. If previously defined, the old definition is erased. Control goes to Pablo.

4. NEW TEXT: transfers control to Name List editor, where one element may be selected for the creation of a new Text definition. If previously defined, the old definition is erased. Control goes to the

Text editor.

5. NEW EMPIR: transfers control to Name List editor, where one element may be selected for the creation of a new Empirical definition. If previously defined, the old definition is erased. Control goes to Pablo.

6. ERASE NAME: transfers control to Name List editor, where up to seven elements may be selected for oblivion. All aspects of the elements (name, symbol and definition) are erased. Control returns to Picasso.

7. ERASE ALL: requests teletype confirmation (type YES) to erase all aspects of all elements currently in core. This command does not affect the Libraries, which reside on a disk file. Control returns to Picasso.

8. LOAD LIB: transfers control to Name List editor, where one Library may be selected for loading into core, replacing any elements already there. Control returns to Picasso after loading.

9. APPEND LIB: transfers control to Name List editor where one Library may be selected for loading into core along with the elements already there. Control returns to Picasso.

10. SAVE LIB: transfers control to Name List editor, where one Libname may be selected or created. All aspects of all elements currently in core are written onto disk under that Libname, and may be retrieved at any time by executing the LOAD LIB command.

11. ERASE LIB: transfers control to Name List editor, where one Libname may be selected for oblivion. This Libname is no longer available on the disk file, and when the data cell library is rewritten,

this Libname disappears forever. Confirmation is required, in the form of YES typed in on teletype. Control returns to Picasso.

12. VIEW SYMBOL: transfers control to the View editor, where all the element names in core are displayed. By selecting a name, one may view (not edit) the symbol. A light pen hit re-displays the name list and another name may be selected. This is a handy way to remind the user of what all the element names represent, and is much cheaper than editing them. An IKEY hit returns control to Picasso.

13. VIEW DEF: similar to VIEW SYMBOL, except the Macro or Empirical definitions are the only elements which will be displayed - elements with text definitions are ignored here (use ED OLD DEF instead).

14. ANALYZE: executes the topological analysis of all elements with Macro definitions and transfers control to Name List editor, where one element may be selected for analysis. The conversion routine MIMVERT substitutes the actual parameters of the model for the formal parameters of the primitive elements (with text definitions) and sends the output to the text editor, where the output may be edited, saved on disk or printed (done by default). Return from the Text editor (type ;R) returns control to the Name List editor where the next model may be selected for analysis. Returning from the Name List editor with an empty list relinquishes control to Picasso.

15. USERCMD: transfers control to a user-supplied subroutine called Usercmd, if one has been provided and has been loaded before PICASSO by a sequence of control cards; otherwise, transfers control to PICASSO's Usercmd which produces a list of Library names and element names currently available. Also an octal dump of the data structure

may be obtained, for those interested in such a feature. This subroutine was included to allow user expansion of the commands available in PICASSO, and is at present used for debugging. Most users will be uninterested in octal dumps, and by typing a minus sign (-), control returns to Picasso as if nothing had happened. Usercmd types the calling editor (in this case Picasso) and waits for a teletype message before proceeding.

16. FINISHED: comments will inform the user if he has forgotten to save his Library and/or analyze his model. This command requires teletype confirmation (type EXIT) before finishing the run. Any other characters (except SKIP) return control to Picasso. If EXIT is typed, the disk file LIBRARY is rewritten onto the disk file LIBSAVE with all the out-of-date information (erased Libnames, and the old versions of Libnames which have been re-saved) deleted. If SKIP is typed, the transfer to the disk file LIBSAVE is not done.

b. The Name List Editor:

Most zero-level commands go immediately to an editor called the Name List editor before proceeding to the subordinate editors. (See Figure 3). Name List is responsible for displaying the list of all the names available for the zero-level command (i.e., either the element names or the library names) and allowing the user to select from this list or create new names to add to it. It is a congenial namekeeper which thoughtfully displays all the names currently available, relieving the user from the chore of remembering everything he has done. Having all the names before him, the user may then select which element or library he wishes to edit, erase or whatever (the comments are

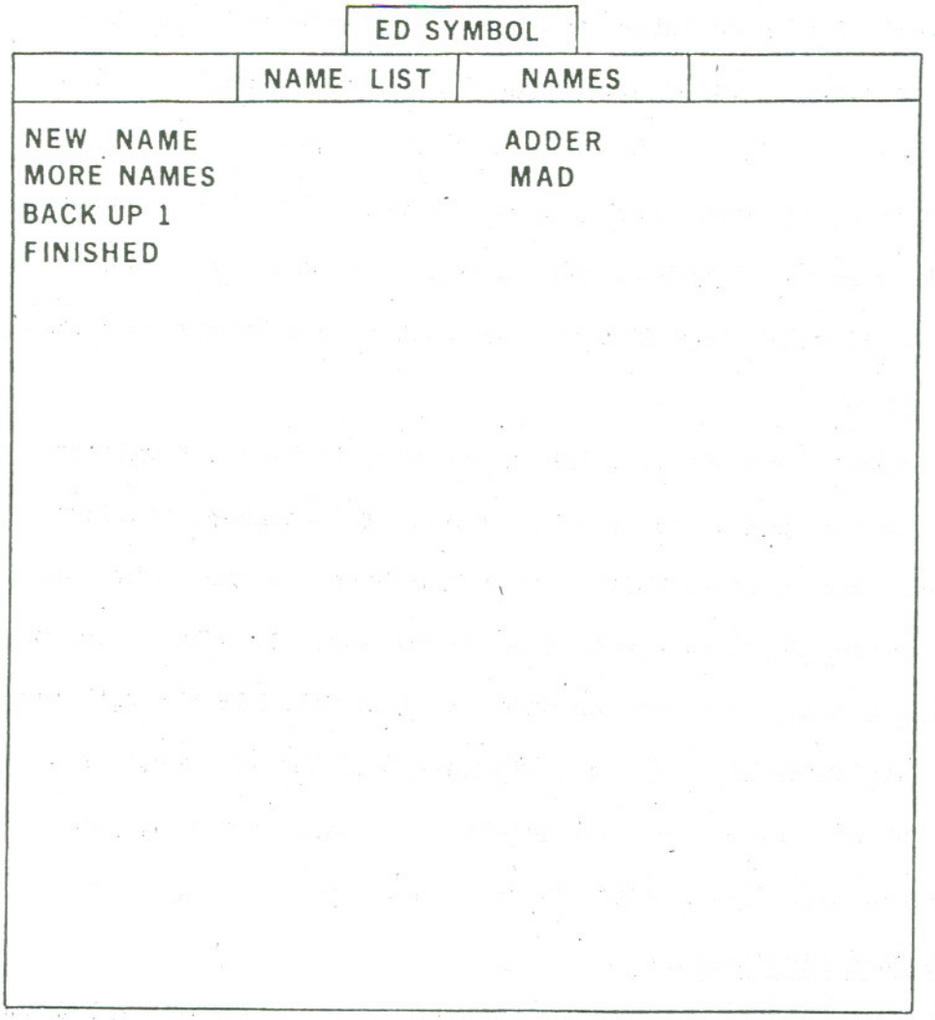


Fig. 3. The Name List editor, as it appears after being called via the ED SYMBOL command. The list of names available appears in the right column (see example in Appendix III). Names selected for editing will appear under NAME LIST in the center column

especially helpful here). Name List commands are few and simple - sort of 1/2-level commands.

1. NEW NAME: To create a new name to add to the list, this command is selected and the new name is typed in on the teletype. The new name joins the list of available names and is also placed on the list of names to be returned to the particular editor indicated by the zero-level command which sent the program to Name List (as indicated in comment 5).

2. MORE NAMES: Only 34 names can be displayed conveniently on the screen at any one time. If there are more than 34, comment 2 indicates how many are left over, and selecting the More Names command displays the next batch. The list is circular.

3. BACK UP 1: This command erases the last name added to the list to be returned to the subordinate editor - it does not erase the name from the current list of names available.

4. FINISHED: This returns the list of names in the center column to the subordinate editor. If the list is empty, control returns to Picasso; returning from Name List with an empty list is thus a do-nothing command.

Note: If element definitions or libraries are being selected in Name List, only the last name entered into the list is returned. Up to seven symbols may be edited at one time, but only one of anything else.

c. PABLO, The Picture Editor.

The zero-level commands ED SYMBOL, NEW MARCO, NEW EMPIR and ED OLD DEF (when appropriate) transfer control to Pablo, where a picture may be constructed. Pablo as 11 commands for picture editing,

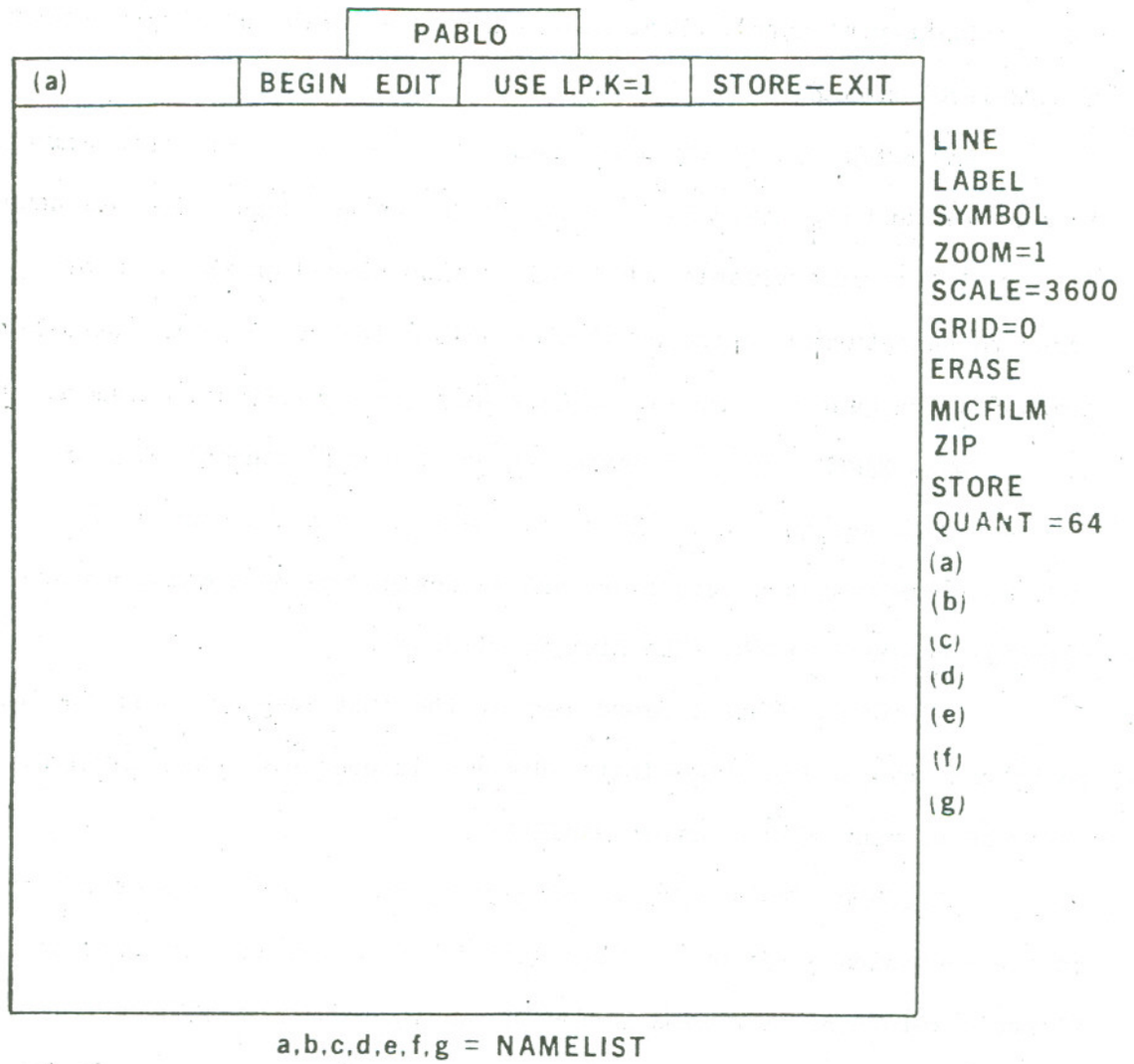


Fig. 4. The initial configuration of Pablo, the picture editor. Comment 1 will indicate which of the names in the list below the QUANTUM command is currently being edited.

most of which are graphic aids to make the picture drawing process fun and easy. (See Figure 4).

1. LINE: Draw a line. This is the default command in Pablo in the sense that, if one begins inputting light pen points, Pablo assumes that lines are being drawn and proceeds accordingly.

2. LABEL: transfers control to the Label editor, where labels are typed in and positioned.

3. SYMBOL: transfers control to the Symbol editor, where symbols are added to the picture.

4. ZOOM: This command allows zooming the picture from 1 to 99 times its CRT size. To execute the ZOOM command, the zoom factor is typed in on teletype and the zoom point is indicated with the light pen. As soon as the factor is known, a zoom box is drawn using the origin as the zoom point. The zoom box may be moved around using the light pen and tracking cross until the box encompasses that part of the picture which the user wishes to see fill the screen. A rather nice line clipper is provided which will display any portion of any line that crosses inside the zoom box. An IKEY hit effects the zoom.

Note that the Vista CRT screen has only 1024 x 1024 raster points. The ZOOM command allows coordinates to be stored in 4096 x 4096 resolution (or any other, for that matter; this resolution was chosen to approximate the film reproduction capability of the Stromberg-Carlson 4060 microfiche unit at LBL. SC4060 output will be available in a later version of PICASSO).

5. SCALE: This command appears initially in the form SCALE=3600, indicating that the scale of the "draw box" goes from 0 to 3600. The

scale may be changed by hitting the SCALE command and typing in the new scale. The scale factor is used in conjunction with GRID and ZIP.

6. GRID: This command appears initially in the form GRID = 0. To execute the command, hit the command GRID and type in the spacing (relative to the scale factor) desired. A grid of points representing that spacing will appear on the screen, up to a maximum of 100 points per line. For example, if SCALE = 5000, and GRID = 100, a 50 x 50 grid of points will fill the "draw box."

7. ERASE: transfers control to the Erase editor, where lines, labels, symbols, the zoom, the grid or the entire picture may be erased.

8. MICROFILM: The picture on the screen, minus the grid overlay and all the comments except the name of the element, is drawn on the microfilm file to be transferred to microfilm at the end of the job.

9. STORE: When a symbol is to be edited, it is copied from the data structure into a display list to facilitate the editing. When the editing is finished, it must be restored in the data structure; this is done by hitting the STORE command, which asks for an Anchor Point. The Anchor Point is the point to which the symbol is referenced when placed on the screen in subsequent use, and may be changed each time the symbol is edited. If the symbol has never been anchored (i.e., it has just been created and never been stored), Pablo stubbornly waits for a light pen point to be entered before proceeding. If the symbol is being re-edited, and an Anchor Point exists, the old point may be specified by hitting the IKEY. Macro and Empirical definitions need not be anchored - the store is immediate, and control returns to Picasso.

10. ZIP: transfers control to the Zip editor, where the picture may either be formatted in the Zip format and written to a disk file, or a Zip-formatted input may be read in to create the symbol.

11. QUANT: This command appears in the form QUANT = 64 initially to indicate that in ZOOM = 1 mode, points input closer than 64 scale units (out of 4096) will be ignored by Pablo's line drawer. This feature is to compensate for the fact that the 6600 operates in fractions of a microsecond, while the user, whose hand shakes on a scale of 10's or even 100's of raster points, usually doesn't want a bunch of short squiggly lines in his picture. In addition, lines are straightened to horizontal or vertical by a factor depending on QUANT. The QUANTum may be reset by hitting the QUANT command and typing in the new value. A small value allows for continuous drawing of short, unstraightened lines and a large value allows for easy drawing of long, straight lines. The QUANT value is zoomed inversely with the picture, so that QUANT = 64 at ZOOM = 1 becomes QUANT = 16 at ZOOM = 4.

12. NAMES: The names of the elements (1 or more) appear directly below the QUANT command. If more than one symbol is being edited simultaneously, the one currently being changed appears in comment 1. To edit a different one, a light pen hit on the name changes comment 1 to that name, and all changes made (lines drawn, etc.) affect only that name.

d. THE TEXT EDITOR.

The Text editor interprets all its commands from the teletype. The lines of text input are numbered sequentially for reference and displayed on the screen as soon as they are typed in. In the following

the notation (string) means any string of characters typed in, and the notation (n) refers to a line number assigned by the Text editor.

1. Insert - (string);(n)I

   Insert the text (string) before line (n).

   Example: This is a string;2I

   The line "This is a string" is inserted before line 2, moving all following lines down by 1.

2. Delete - ;(n)D

   Delete line (n)

   Example: ;2D

   Line 2 is deleted and following lines are renumbered.

3. Alter - (string1);(string2);(n)A

   Replace (string1) in line (n) by (string2).

   Example: shark;snarf;5A

   The word "shark" in line 5 is changed to "snarf".

4. Tabs - (n1),(n2),...,(ni);T(char)

   Set tab stops at columns (n1),(n2), ... (ni), activated by the character (char).

   Example: 7,10,72;T* sets tab stops at columns 7, 10 and 72, and tabs when the character * appears. Thus

   100*x=*y-2* comment becomes

   100 x=y-2 comment

5. Page - ;(n)P

   Start the page with line (n). Since only 35 lines of text will fit on the screen at one time, the line numbers start at (n) and go to (n)+34. The command ;P displays the next 35 lines.

6. Microfilm - ;M

Copy the entire text buffer (all lines) to microfilm.

7. Cancel - ;KK

Erase the entire text buffer and start over again with line 1.

8. Write - (file name);W

Write the entire text buffer to the disk file named (file name).

Example: DATA;W writes all the text onto the disk file DATA.

9. Read - (file name);(nn)Q

Read (nn) card images from the disk file (file name), and append to the existing lines of text. The default value for (nn) is 100, or until an end of file is read.

Example: (INPUT);2Q will read 2 cards from input.

10. Rewind - (file name);X

Rewind the disk file (file name). The X command can be used along with the W and Q commands:

Example: DATA;XW rewind and write the disk file DATA.

Example: DATA;XQX rewind, read, and rewind the disk file DATA.

11. Return - ;R

Return control to the calling program.

e. IN LABEL - The Label Editor.

The Label editor is called by Pablo for creating labels on a symbol (or macro definition or empirical definition). The labels represent the formal parameters of the element and are placed at the attacher points to specify the inputs and outputs of the element. Thus there should be a one-to-one correspondence between the labels on the symbol and the parameters appearing in the text definition. Labels on

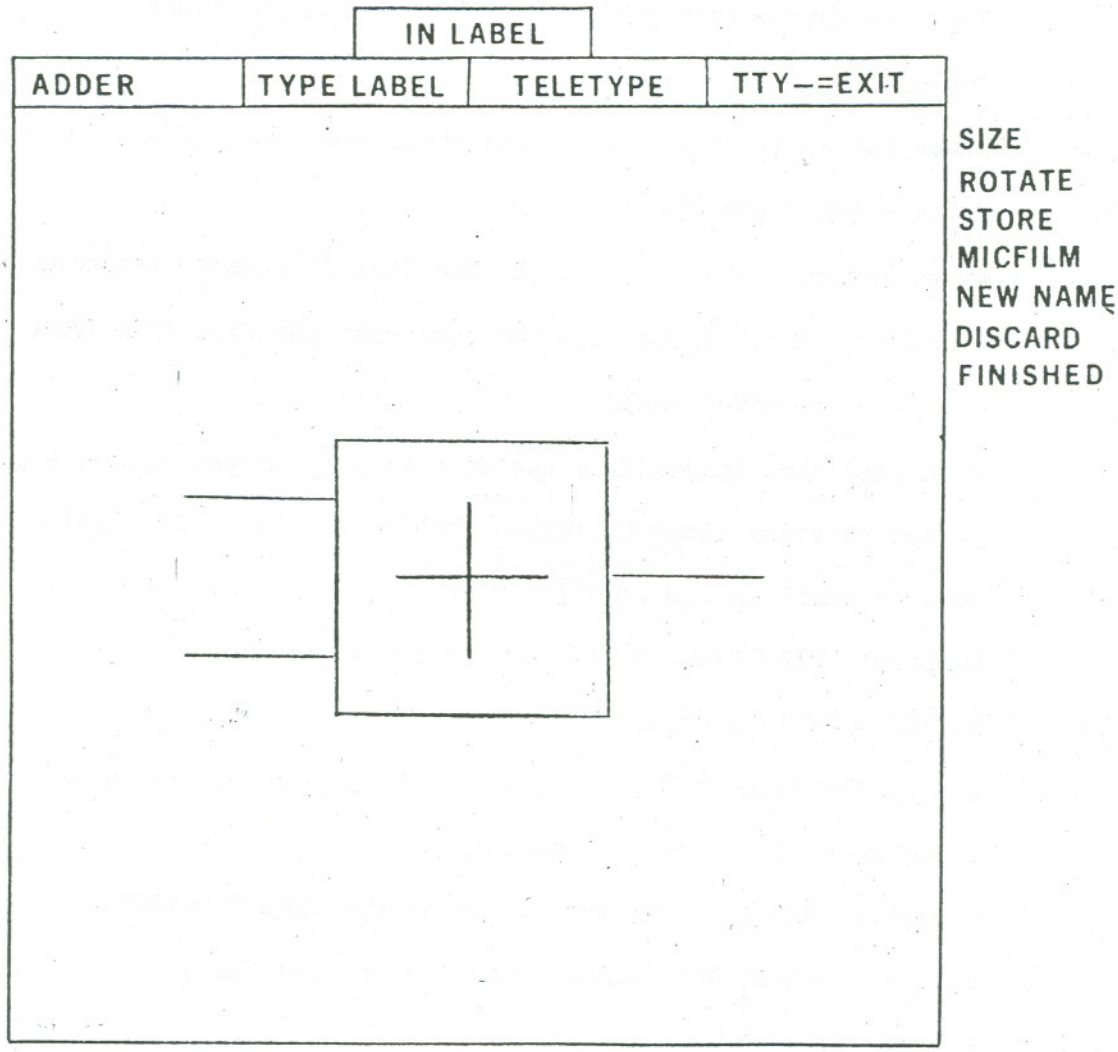


Fig. 5. The initial configuration of the Label editor after the ADDER (see Example in Appendix III) has been drawn, but before any labels have been entered.

a macro-definition (model) are the actual parameters, to be substituted for the formal parameters of the elements within the macro definition. (See example 1 in Appendix III). Labels are made of alphanumeric characters; special characters are used to specify the default value to be used if the actual parameter is not specified in a macro definition. For example, X/1.0 is a label whose default value is 1.0. If no special character appears with a label, a default name is generated internally for the actual parameter. If the default specification is a blank (e.g., X/), the formal parameter is deleted if there is a comma in the text adjacent to it, along with the comma, and a 0.0 is substituted if there is no comma. (For example, see Figure 14a). The Label editor has the following commands (see Figure 5):

1. SIZE: The hardware has four letter sizes available, in the ratio 1:1.5:2:3. An attempt is made to zoom the labels as far as possible, so a sort of logarithmic scale has been adopted which takes the values 0-14. SIZE = 10 is the smallest size which will appear when ZOOM = 1. A label with a size smaller than 10 will not appear on the screen, but will serve to parametrize the attacher point on the symbol. The size of a label is specified by hitting the SIZE command and typing the integer on the teletype.

2. ROTATE: The hardware provides two orientations for labels: horizontal and vertical. The rotate command switches from one to the other in sequence.

3. STORE: After a label has been typed in it may be positioned, rotated and its size changed continually until its appearance satisfies the user. The STORE command causes the label to be stored in the data

structure, and no further changes may be made, except by erasing the label and starting over again.

4. MICFILM: This is a dummy command in the Label editor.

5. NEW NAME: This resets the size and orientation to the default option, and waits for teletype input.

6. DISCARD: This discards the label and waits for teletype input of a new label (or the teletype escape character "- ").

7. FINISHED: stores the current label and returns control to Pablo.

Note: When waiting for teletype input, PICASSO cannot read the light pen. Thus two exit modes are used in the Label editor: store the label (or discard it) and type the escape character "- "; or position the label in the desired spot and hit the FINISHED command.

f. IN SYMBOL - The Symbol Editor.

The Symbol editor is called by Pablo for adding previously created symbols to the picture. The Symbol editor uses the same commands as the Label editor, but with a somewhat different meaning.

1. SIZE: The size of a symbol being entered may be specified by hitting the SIZE command and typing in a number on the teletype. The size is constrained to lie between 1 and 64, and simply magnifies the symbol by that factor.

2. ROTATE: Symbols are made of lines and labels; the lines can be rotated in increments of $2\pi/64$ radians (360/64 degrees) by hitting the ROTATE command and specifying points with the light pen. The anchor point of the symbol is taken as a center and the horizontal axis is taken as a base line. An IKEY hit returns the program to the positioning mode, where the symbol can be moved about or stored.

3. STORE: Symbols may be repositioned, rotated and scaled up or down continually until the STORE command is executed. No further alteration may be done except by erasing and starting over again.

4. MICFILM: This transfers the picture to microfilm without storing the symbol. It is included for fun, since it allows a crude form of animation. Serious microfilming should be done in Pablo.

5. NEW NAME: After a symbol has been stored, the Symbol editor is ready to place another symbol of the same name at the next point entered with the light pen. If a symbol of a different name is wanted, the NEW NAME command must be hit, and the name of the new symbol typed in on the teletype.

6. DISCARD: This commands discards the symbol being entered and waits for teletype input of a new name.

7. FINISHED: stores the symbol and returns control to Pablo. The first command to the Symbol editor is the name of the symbol to be entered. Only enough letters of the name need be typed in to uniquely specify the symbol. Normal exit is done via an IKEY hit, but if the Symbol editor is waiting for teletype input, the exit may be done by typing the escape character " - ".

g. ZIP - The Zip Editor.

The Zip editor allows for external input and output of pictures in a certain format. The commands for zipping in or out (using disk files as externals) are those of the Text editor - i.e., the (file name) ;W and (file name);Q commands on teletype.

Symbols may be created on cards or on-line via this editor if more accuracy than is obtainable with the light pen and grid points is

desired. A more useful function of the Zip editor is the transferring of symbols from one library to another. For example, if there is an ADDER in LIB1, and one wants an ADDER in LIB2, the procedure is as follows:

Load LIB1, edit the ADDER; while in Pablo, hit the ZIP command, and type ZIPDISK;W to write the ADDER out to the disk file ZIPDISK. Then load LIB2, create a name ADDER and while in Pablo, hit the Zip command. Now type ZIPDISK;XQ to rewind ZIPDISK and read it into the Zip editor. Magically, the ADDER symbol will appear, complete with all the lines and labels (i.e., it is not a call to a symbol, but a new symbol composed of lines and labels just as if drawn in by hand).

For on-line or card input of symbols, the Zip format is described on the next page. Figure 6 has an example of the ADDER in Figure 7 in ZIP format.

ZIP FORMAT.

The notation (x) indicates the position of the point referred to the scale set with the $SCALE card. The default values are given.

$SCALE	Default values and meaning:
(xmin),(ymin)	0,0
(xmax),(ymax)	Set to current SCALE setting in Pablo.
$LINE	Draw a series of connected lines.
(x1),(y1)	The first point of a line.
(x2),(y2)	The second point of a line.
...	More points.
(xn),(yn)	The last point.
$LABEL,(10 characters or less),(size),(orientation)	
(x),(y)	The position of the label.
$SYMBOL,(name),(size),(rotation)	
(x1),(y1)	The position of the first symbol of name (name).
(x2),(y2)	The position of the second symbol of the same name.
...	
(xn),(yn)	The position of the last one.
$END	End of Zip. Stop reading.

```
*ADDER
$SCALE
0.0,0.0
3600,3600
$LINE
14,160
64,160
$LINE
14,96
64,96
$LABEL,IN1,6,0
14,160
$LABEL,IN2,6,0
14,96
$LINE
64,176
64,80
$LINE
64,80
143,80
$LINE
80,129
129,129
$LINE
104,153
104,102
$LINE
143,80
143,176
$LINE
143,176
64,176
$LABEL,OUT,6,0
207,129
$LINE
207,129
143,129
$END
```

Fig. 6. The ADDER in Fig. 7 (see example, APPENDIX III) in ZIP format. The ADDER shown in Fig. 7 is drawn with ZOOM=16, which causes it to fill the screen; the coordinates are relative to the scale set by the $SCALE card.

h. The ERASE Editor:

The Erase editor is called by Pablo to erase components of the picture, the grid overlay or the zoom. The initial commands to the editor are those in Pablo - LINE, LABEL, SYMBOL, ZOOM or GRID. (See Figure 4). If ZOOM or GRID is specified, the erasure is immediate and control returns to Pablo. Erasing the actual components is a little more involved.

After the type of component to be erased is chosen (by hitting LINE, LABEL or SYMBOL), the Erase editor redraws the picture in relative mode. In this mode, a light pen hit with only the 1 key latched returns to the editor the origin of the component hit. Thus if a light pen hit is generated on any part of a symbol, only its anchor point is read; similarly, a hit on any part of a line or label returns the initial point to the editor. As soon as a hit is received, the editor begins searching for the particular type of component to be erased in a small region around the point. Upon finding one, the editor winks the component and comment 4 changes to "ZAP = IKEY" to indicate that if the winking component is indeed the offensive one, a hit on the IKEY will annihilate the offender. Since more than one line or symbol may share the same origin, the winking component may not be the right one; if this is the case, another light pen hit sends the editor searching for other such components nearby. If one is found, this component is then winked, and again the editor waits for IKEY confirmation before erasing. In case the editor can find no more such components, comment 4 changes to "IKEY=EXIT" to indicate that an IKEY hit will return control to Pablo with no further erasing.

To erase the entire picture, the name of the element being edited at the time (which appears in the list below the QUANT command) may be hit, and all components of the picture (lines, labels and symbols) will disappear.

IV. LIBRARIES OF ELEMENTS

The collection of elements created may be saved on a disk file named LIBRARY during the execution of PICASSO by executing the SAVE LIB command and indicating a Libname under which the collection is to be saved. Up to 62 different Libnames can coexist on this disk file at one time, and these may be continually be loaded into core, edited, saved again; in addition, Libraries may be combined via the sequence Load Lib#1, Append Lib#2, Save under Lib#3 (or Lib#1 or 2). The sample control deck provides the user with a sample library to investigate.

Before ending execution of PICASSO, one usually wants to save all the elements created for later use. The SAVE LIB command writes the elements from core to the disk file LIBRARY. When execution of PICASSO is ended by typing EXIT, the disk file LIBRARY is rewritten in garbage-collected form to the disk file LIBSAVE. The sample control card deck shows how the disk file LIBSAVE may be written onto the PSS (data cell) via a PTSS command. Each user must supply his own COPYPSS control cards with his own Group number, Account number and Ownername.

V. SAMPLE CONTROL CARD DECK FOR RUNNING PICASSO AND MIMIC VIA PTSS.

```
MODEL,17,50000.981204,AUSTIN    (supply your own job card)
REQUEST FILM,TV.  ASSIGN 42.
LIBCOPY(PICASSO,DRAW,DRAW)
LIBCOPY(PICASSO,LIBRARY,LIBRARY)
LIBCOPY(MIMIC,MIMGO,MIMNEW)
COMMON(PTSS)
PTSS.
7-8-9
7-8-9
7-8-9
GROUP=(your group nr, and group name)
ACCOUNT=(your account)
OWNER=(your name)
WRITE LIBRARY=(your library name)
REWIND INFILE=LIBSAVE
REPLACE SUBSET=LIBRARY
COPY 1F FROM CIOFILE=LIBSAVE
REWRITE LIBRARY=(your library name)
END
NOW SEE HERE, THAT NAME IS NOT ON PROTO
DRAW
SFL(55000)
DRAW.
EXIT.
CXIT.
FIN.
PTSS(E)
NOW TYPE ;G TO DRAW
MIMIC
REWIND(DATA)
SFL(100000)
MIMGO(DATA)
EXIT.
PTSS(A)     MIMIC BLEW UP
CXIT.
PTSS(A)     MIMIC COMPILATION ERROR
FIN.
PTSS(E)
NOW TYPE ;G TO MIMIC
SAVELIB
SFL(50000)
NOR.  IF PSS HANGS UP, OPERATOR SHOULD DO EXACTLY ONE DROP.
COPYPSS(PROTO)
PTSS(A)     COPYPSS SUCCESSFUL, COMMENTS ON OUTPUT
EXIT.
PTSS(A)     COPYPSS UNSUCCESSFUL
CXIT.
PTSS(A)     COPYPSS UNSUCCESSFUL
```

```
FIN.
PTSS(E)
NOW TYPE ;G TO SAVE LIBRARY ON PSS
QUIT
RETURN(FILM)
COPY(CAMERA/RB,FILM)
NOW TYPE ;G TO SAVE MICROFILM AND QUIT
6-7-8-9
```

The sample control card deck contains a jobcard (which must be supplied by each user), a REQUEST card to assign the Vista console to the job, some LIBCOPY cards to get the PICASSO object deck (called DRAW), a sample PICASSO library and the MIMIC object deck from the data cell. The COMMON card attaches the public file PTSS to the job, and PTSS. begins execution. The 7-8-9 cards separate the input records and ensure that the control card deck for running the various programs end up on a disk file called PROTO, which PTSS recognizes as a source of control cards.

To run PICASSO, connect the teletype to the job (as in Example 1 in Appendix III), turn off the RECC dayfile messages (type > N) and wait for PTSS to type BEGIN EDIT. Type DRAW;R, and wait for the reply NOW TYPE ;G TO DRAW. Type ;G and soon a picture very much like Figure 2 will appear on the screen.

To run MIMIC on a model, the analysis of the model should be written to the disk file DATA (via the DATA;XW command in the Text editor). Wait for PTSS to type BEGIN EDIT, after ending execution of PICASSO, and type MIMIC;R. PTSS will reply with NOW TYPE ;G TO MIMIC; type ;G and wait. MIMIC will come to the screen as soon as the system has core available.

To save your Library on the data cell, execute the SAVE LIB command in PICASSO, exit by typing EXIT and wait for PTSS to type BEGIN EDIT. Type SAVELIB;R wait for the NOW TYPE ;G TO SAVE LIBRARY ON PSS, and type ;G. Several minutes may be required to complete the write on the data cell. PTSS will reply with a message indicating whether or not the write was successful.

To save your microfilm and end the job, wait for the PTSS BEGIN EDIT, as usual, and type QUIT;R. Wait for the NOW TYPE ;G TO SAVE MICROFILM AND QUIT, type ;G and disconnect the teletype (type >DC). Your microfilm will appear a few hours later at the IO desk.

PICASSO

APPENDICES

APPENDIX I. OPERATING THE VISTA CONSOLE

The CDC Vista 250 Display System at LBL is the prototype system and may seem somewhat inconvenient at times. It is, nevertheless, a good interactive graphics system, especially when interfaced with the CDC 6600. The Vista operating system was written by Bill Benson of the LBL Math & Computing Systems Group, and was designed to make the best of a prototype system for general graphics users. A detailed description of the light pen and function keyboard use is given in Section N of J5 BKY TV66, a writeup available in the Computing Center Library. What follows is a brief description of the operating procedure necessary for running PICASSO.

PICASSO requires four types of Vista console input. The console is equipped with a light pen (a tube of fiber optics with a finger switch at the end) and a function keyboard containing three rows of numbered keys and red button marked I at the top (called the IKEY). The four types of interrupts generated by this equipment and read by PICASSO are:

1. IKEY Interrupts: Comment 4 usually refers to the IKEY for exiting a particular editor or storing a label or symbol. To generate an IKEY interrupt, the IKEY must be tapped firmly until the word HIT appears at the center of the screen. The position of the other keys on the function keyboard is ignored by PICASSO.

2. Light Pen Commands: The graphic command language of PICASSO is implemented in the form of phrases written on the screen, such as ED SYMBOL or FINISHED. A light pen hit on these command phrases causes the program to transfer control to the appropriate editor performing

the function indicated. In order to generate a light pen command hit, four steps are necessary.

i. Release the keys in the bottom row, if the wrong ones are latched. This is done by pressing the key marked R.

ii. Depress the key marked 1 in the bottom row until it latches.

iii. Point the light pen directly at the command word with the finger switch open.

iv. As briefly as possible, press the finger switch closed and release it again.

3. The Tracking Cross: The tracking cross is a bright cross of light generated by a peripheral processor attached to the Vista system (called equipment 40). The tracking cross does not generate any input to the 6600 - it merely provides some light at a place on the screen where a point may be entered. To generate the tracking cross, release the latched keys, if necessary, and depress keys 1 and 2 on the bottom row until they latch. Point the light pen at any available patch of light (comments are good places to find light), and close the finger switch on the light pen. As long as the finger switch is closed, the tracking cross will follow the light pen across the screen (if the motion is not too fast).

4. Light Pen Points: An interrupt is generated and the coordinates of the tracking cross are sent to the 6600 when keys 1, 2 and 3 are depressed, the light pen sees light from the tracking cross, and the finger switch on the light pen is closed. The best technique is to latch keys 1 and 2, move the tracking cross to the desired position, and gingerly tap the 3 key hard enough to generate a HIT, but not hard

enough to latch the key.

A second method of entering light pen points is exactly the same as the light pen command method (in fact, PICASSO reads the co-ordinates of the command words, not the words themselves). When only key 1 is latched, the origin of the line or word seen by the light pen is returned to the 6600.

QUIRKS

Because the Vista console is attached to a CDC 6600 with 100 nanosecond cycle time, each light pen hit attempted by most .1 second-cycle-time users results in several dozen hits being read by the peripheral processor each time a single input is intended. PICASSO deals with this sticky situation by ignoring successive hits at the same point. Most of the time this works very nicely, but at times PICASSO may seem to be ignoring the input. This quirk can easily be overriden by taking a hit from a comment at the top of the screen. This action clears the "ignore" flag, and otherwise acts as a do-nothing command.

Another cause of delay in response is the nature of the time-shared operating system of the 6600. Most of the time running PICASSO is spent in auto-recall waiting for input from the console or the teletype. During this time, PICASSO relenquishes the central processor of the 6600 to other users. When the console input is forthcoming, PICASSO must wait a finite amount of time to regain the CP. Sometimes this wait time seems too finite, and a few seconds delay becomes noticeable before the last input is acted upon. During this wait time, however, the peripheral processor is busy reading and storing away all

the input in its buffer for transmission to the 6600 when the CP is recaptured, so that all the input will be read in succession. The disadvantage of this situation is that an impatient user may generate several undesirable hits, wondering why nothing is happening, and when they all get processed, he doesn't know where he is or how he got there. On the other hand, an old timer can generate a series of interesting inputs and sit back and watch them being executed in real time.

Of course, the real advantage of the auto-recall system outweighs all the disadvantages: one can sit at the Vista console for several hours of real-time interaction and use only a few hundred CP seconds (roughly $5 an hour of real-time interaction).

APPENDIX II. THE PEOPLE'S TIME-SHARING SYSTEM

PICASSO is especially easy to use in combination with the PTSS time-sharing system developed by Bill Benson and Andy Tanenbaum. This system allows for on-line creation and execution of control cards (among other things), so that one can control the flow of the job in real time, instead of deciding exactly what he wants to do before he starts the job. A sample run might follow the sequence:

Run PICASSO to create a model, analyze it and write it to disk file;

Run MIMIC to analyze the model and display the results;

Rerun PICASSO to change the model;

Rerun MIMIC;

Save the Library on the data cell;

Rerun PICASSO;

Rerun MIMIC;

Quit.

The sample control card deck in Section V contains all the cards necessary for executing this sequence.

PTSS identifies itself by typing BEGIN EDIT on the teletype. All PTSS commands are via the teletype and are of the form NAME;R for executing a sequence of control cards identified by NAME. See Section V for more details.

A complete writeup of PTSS is available from the computing center library. It is a highly recommended debugging tool; only a small portion of its capabilities have been mentioned here.

APPENDIX III. EXAMPLES.

Several examples have been included in this appendix, ranging from an extremely simple ADDER, covered in excruciating detail, to more interesting examples of logic circuit analysis and a model of the San Francisco Bay system.

Example 1 refers to the ADDER in Figures 7 an 8 and MAD model in Figure 9. The detailed steps covered on the following pages may seem to represent far too much effort for such a simple result. The reader may be encouraged to learn that the entire example can be constructed in under 1 minute of operating time, after the mechanics of operating the Vista console have been mastered.

EXAMPLE 1: A SIMPLE MODEL IS CONSTRUCTED FROM SCRATCH.

SEE FIGURES 7-10 FOR THE PICTURES REFERRED TO IN THE TEXT.

EDITOR	COMMAND	RESULT
Read sample deck in on card reader and ask 6600 operator to start your job.		
	>CT.MODELOO	Connect teletype to your job (use your job name).
	>N	Turn off the RECC dayfile messages to avoid mess.
PTSS	"BEGIN EDIT"	PTSS reply indicates job is running and ready.
	DRAW;R	Type the DRAW command to PTSS.
	"NOW TYPE ;G TO DRAW"	replies PTSS.
	;G	
Picasso		Figure 2 appears on the screen, Picasso waits.
	ED SYMBOL	Control goes to Name List editor (Fig. 3). No names appear yet, since we are starting from scratch.
Name List	NEW NAME	Name List waits for teletype input of first name.
	ADDER	Type in ADDER- the name appears in both lists.
	FINISHED	Control goes to Pablo (Fig. 4).
Pablo		Pablo waits for light pen command or point.
	(point 1)	The tracking cross is moved to Point 1 on Fig. 7, (with keys 1 & 2 latched), and key 3 is tapped until the word HIT appears, and a little cross indicates the point Pablo read.
	(point 2)	Another point, and a line appears.
	(point 3)	Another line appears.
	(point 4)	And another.
	(point 1)	A square has been drawn in.
	IKEY	An Ikey hit to start a new line - Pablo waits.
	(point 5)	The initial point of the new line.
	(point 6)	The vertical line appears.
	IKEY	New line
	(point 7)	
	(point 8)	The horizontal line appears.

EXAMPLE 1 Continued 2

EDITOR	COMMAND	RESULT
	IKEY	New line
	(point 9)	
	(point 10)	The first input line appears.
	IKEY	New line
	(point 11)	
	(point 12)	The second input line appears.
	IKEY	
	(point 13)	
	(point 14)	The output line appears.
	LABEL	Hit the Label command (with only key 1 latched); control goes to the Label editor (Fig. 5).
In Label		The Label editor waits for teletype input.
	IN1	Type IN1 on teletype, the editor waits for a position.
	(point 9)	The label IN1 appears at point 9.
	STORE	Hit the Store command (or IKEY) to store the label.
		The editor stores the label, waits for teletype.
	IN2	Type IN2; the editor waits for a position.
	(point 11)	The label IN2 appears at point 11.
	STORE	The IN2 label is stored and the editor waits again for teletype input.
	OUT	OUT is typed in; the editor waits for position.
	(point 13)	OUT appears at point 13.
	STORE	OUT is stored away in the data structure. We have finished with the labels and wish to return to Pablo. Type in a minus sign to exit In Label.
	-	
Pablo		Pablo waits for light pen input.
	STORE	Ready to store the ADDER away in the data structure. Pablo waits for an anchor point.
	(point 9)	The ADDER is anchored at the first input (labeled IN1), and stored away. Control goes to Picasso.
Picasso		Picasso waits for light pen command (Fig. 2).

EXAMPLE 1 Continued 3.

EDITOR	COMMAND	RESULT
	NEW TEXT	Control goes to the Name List editor (Fig. 3). Now we have a name in the right-hand column.
Name List	ADDER	A light pen hit on the name ADDER puts it in the center column to be returned to the Text editor.
	FINISHED	Control goes to Text editor for creation of a new text definition for the ADDER.
Text Edit		Text editor waits for teletype input.
	OUT=IN1+IN2	Definition is typed in, appears on screen (see Fig. 8).
	;R	Exit the text editor, return to Picasso.
Picasso		Picasso waits.
	ED SYMBOL	Control to Name List editor
Name List	NEW NAME	Name List editor waits for new name on teletype.
	MAD	The name MAD is typed in, joins ADDER in list of names available and is also placed in the center list. We are creating Fig. 9.
	FINISHED	Control to Pablo.
Pablo		Pablo waits for light pen input.
	SYMBOL	Control goes to the Symbol editor, who waits for teletype input of the name of the symbol to be added to the picture.
In Symbol	ADDER	Type in ADDER (actually, just A will do); the symbol editor waits for a position of the adder (the position is where the anchor point of the ADDER will appear).
	(point)	The first ADDER appears.
	STORE	The first ADDER is stored, and the editor is ready for ANOTHER ADDER to be positioned.
	(point)	The second ADDER appears.
	STORE	The second ADDER is stored, and the editor is ready for another ADDER to be positioned.
	(point')	The third ADDER appears, but at an inconvenient place, say.

EXAMPLE 1 Continued 4

EDITOR	COMMAND	RESULT
	(point)	The third ADDER moves to a nicer place.
	STORE	The third ADDER is stored.
	IKEY	A hit of the IKEY exits the symbol editor and returns control to Pablo.
Pablo		Pablo waits for light pen input.
	LABEL	Control goes to Label editor, who waits for tty.
In Label	A	A is typed in, editor waits for position.
	(point)	A light pen hit in the first input of the first ADDER causes the label A to appear there.
	STORE	A is stored, editor waits for teletype.
	B	B is typed, editor waits for position.
	(point)	A light pen hit on the second input of the first ADDER causes the label B to appear there.
	STORE	
	1.0	1.0 typed.
	(point)	A light pen hit on the first input of the second ADDER causes the label 1.0 to appear there.
	STORE	
	C	C typed.
	(point)	C at second input of second ADDER.
	STORE	
	OUT	Out is typed.
	(point)	Light pen hit on output of third ADDER.
	STORE	The last label is stored. Editor waits for tty.
	-	Type -, the teletype escape character, to return to Pablo.
Pablo		Pablo waits for light pen.
	(point)	A light pen hit on the output of the first ADDER causes that to be the initial point of a line. The light pen hit may either be generated via the tracking cross or in the key 1 mode.
	(point)	Light pen hit on the input to the third ADDER draws a line connecting the two ADDERs.
	IKEY	New line
	(point)	Light pen hit on output of second ADDER.

EXAMPLE 1 continued 5.

EDITOR	COMMAND	RESULT
	(point)	Light pen hit on second input of third ADDER, and the line is drawn. We have finished our model.
	STORE	Pablo waits for an anchor point before storing the model MAD away.
	(point)	MAD symbol is stored, and control goes to Picasso.
Picasso		Picasso waits. We have finished Fig. 9.
	ANALYZE	Control goes to Name List (see Fig. 3).
Name List	MAD	The name MAD is hit with the light pen,
	FINISHED	and joins the center list. Control goes to the Analysis editor.
Analysis		The Analysis editor analyzes MAD - note that MAD has not been defined, so its symbol becomes a Macro definition of the model MAD. Control goes to the Text Editor where the output may be viewed (see Fig. 10).
Text Edit		The Text editor presents Fig. 10 and waits for teletype input.
	;R	;R (return) typed, control goes to Picasso.
Picasso		Picasso waits for light pen command.
	SAVE LIB	Control goes to Name List editor for a Libname under which the elements will be saved on disk.
Name List	NEW NAME	The list of Libnames appears (from the sample PICASSO Library), but we don't want to mess up those before we get a chance to play with them, so let's make a new Library.
	SIMPLE	Type in the name SIMPLE, which is added to the Libnames and appears in the center column.
	FINISHED	Control goes to the Library editor, where all the elements in core (ADDER and MAD) are written onto the disk file LIBRARY under the name SIMPLE. Control returns to Picasso.
Picasso		Picasso waits.
	FINISHED	Comment 1 says "type EXIT", which is the normal thing to do, so

EXAMPLE 1 continued 6.

EDITOR	COMMAND	RESULT
	EXIT	is typed in. Now the Library editor rewrites the disk file LIBRARY onto the disk file LIBSAVE, eliminating any out-of-date information in the process. Soon, the picture goes away and control goes to the PTSS editor.
PTSS	"BEGIN EDIT"	PTSS announces itself by typing this.
	SAVELIB;R	Type in the control word SAVELIB for PTSS.
	"NOW TYPE ;G TO SAVE LIBRARY ON PSS"	replies PTSS.
	;G	Off we go to write the disk file SAVELIB onto the data cell under your very own group number, name and account supplied as in the Sample control card deck. This may take several minutes.
	"COPYPSS SUCCESSFUL ..."	replies PTSS, to inform you of your good fortune.
	QUIT;R	Time to go home, so type the control word QUIT.
	"NOW TYPE ;G TO SAVE MICROFILM AND QUIT"	replies PTSS.
	;G	Film is saved (we forgot to make any), and the job ended.
	>DC	Give the system back their teletype by typing this in.

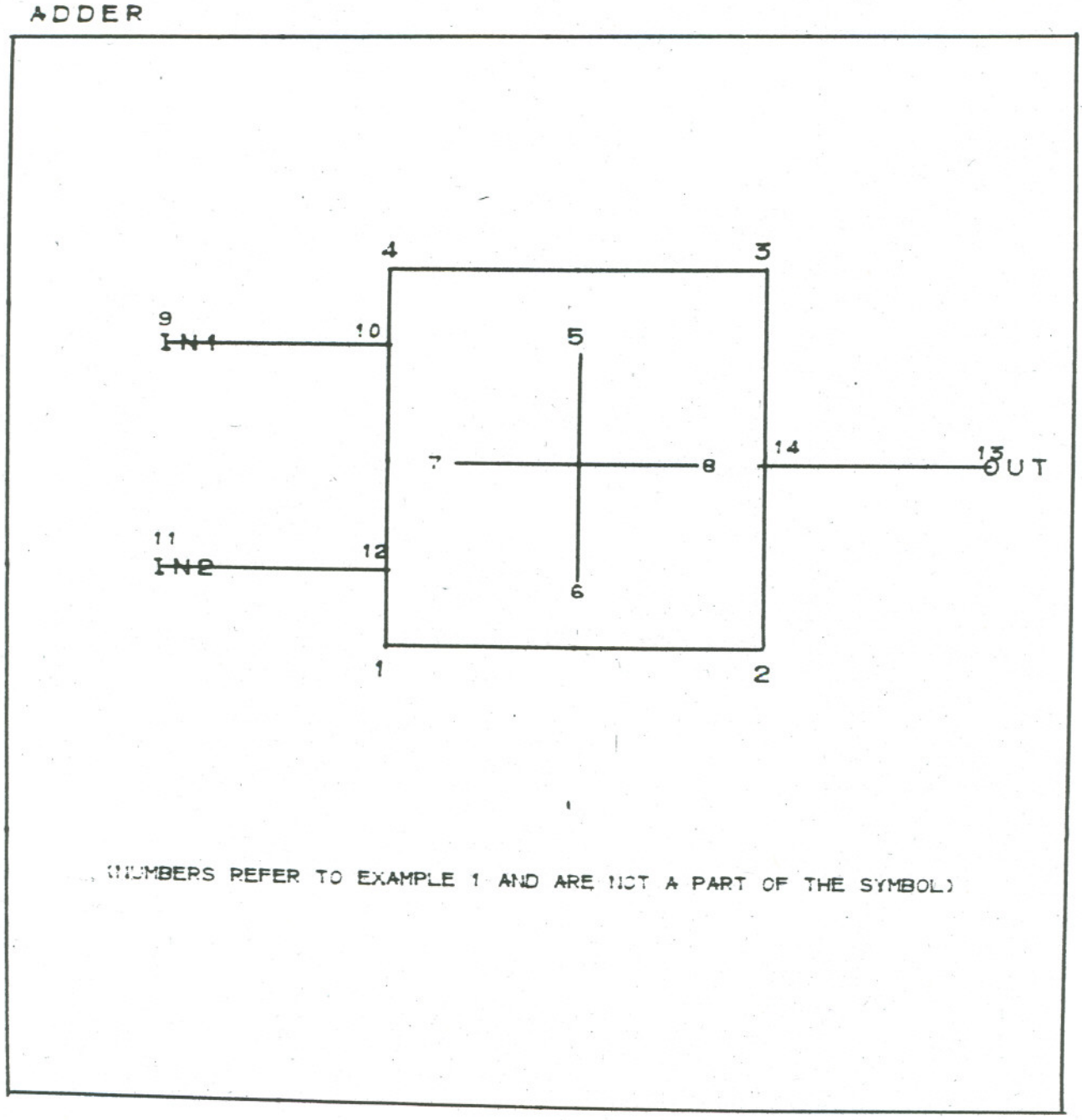


Fig. 7. The ADDER symbol (numbered points refer to the text of the Example), drawn with ZOOM = 16.

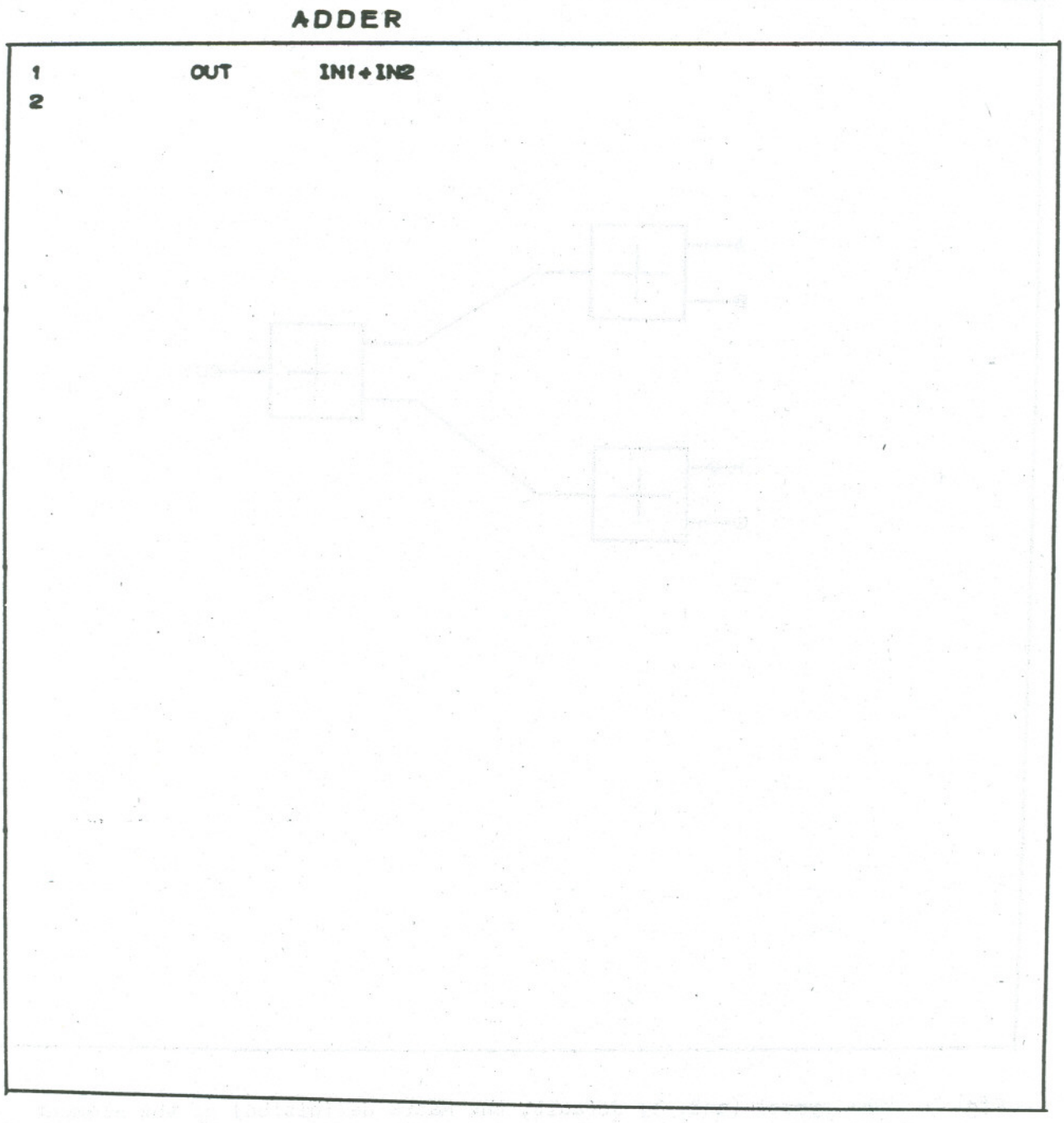


Fig. 8. The Text definition of the symbol ADDER.

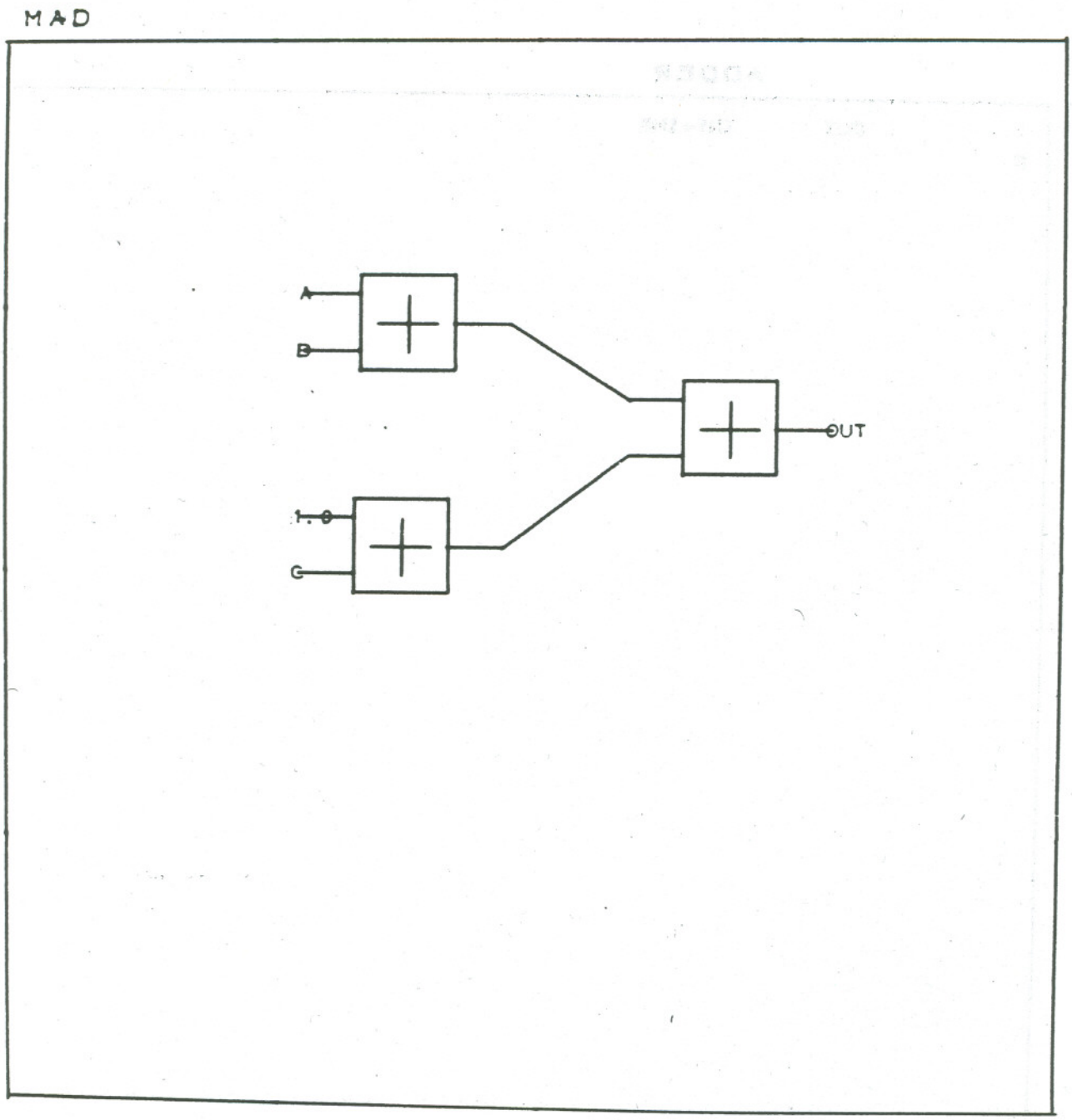


Fig. 9. The symbol (and, by default, the Macro definition) of the element MAD, drawn with ZOOM = 4.

MAD

```
1  $IAM       MIMIC RUN OF MAD      PRODUCED BY MIMVERT
2             G003      1.0+C
3             G004      A+B
4             OUT       G004+G003
5                       END
6
```

Fig. 10. The output of PICASSO's analyzer MIMVERT. The header card is generated internally, as are the line numbers and END card. Only lines 2, 3 and 4 pertain to the example. The names G001 and G002 are internally generated names for unspecified formal parameters.

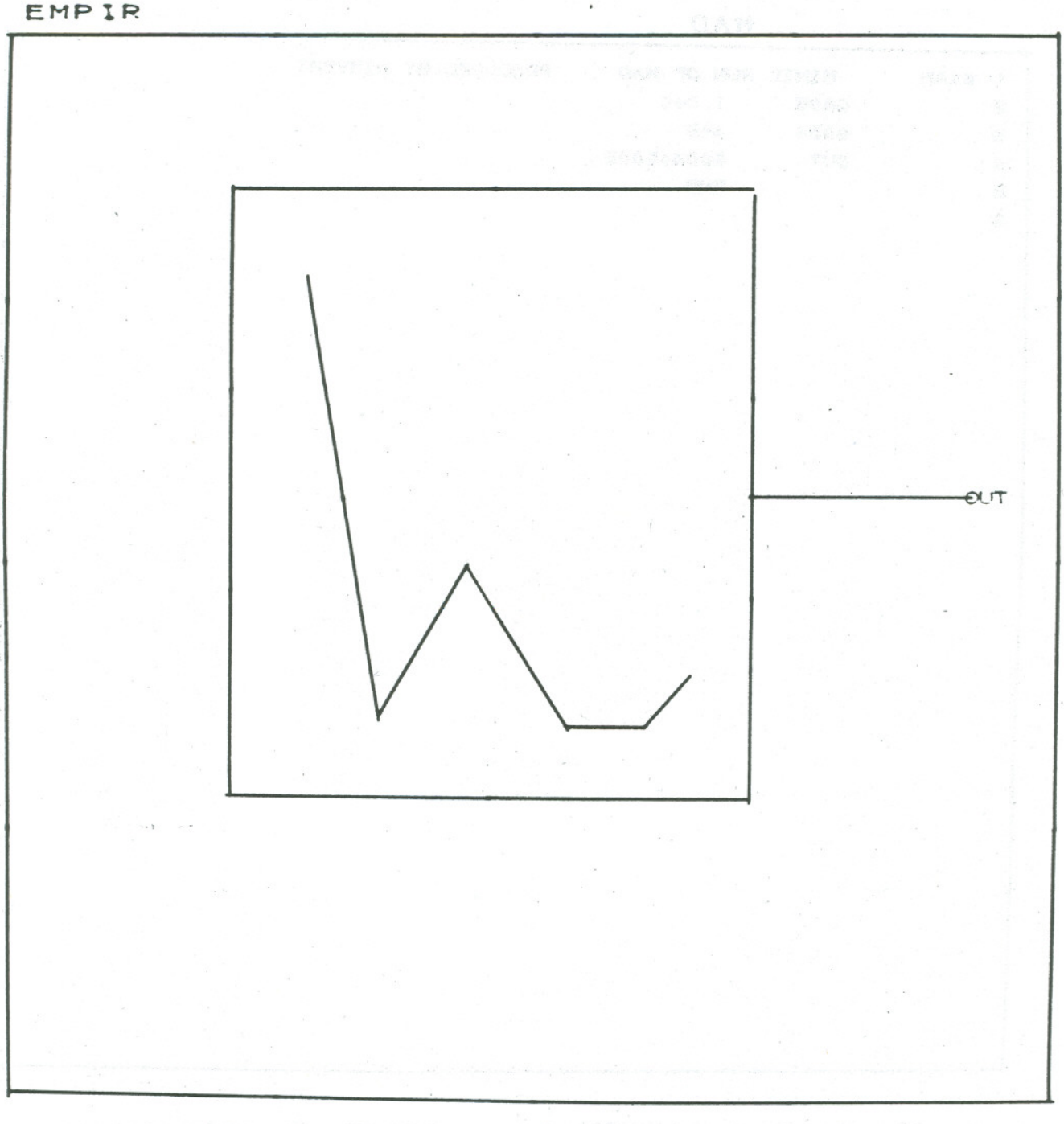


Fig. 11a. The symbol for an element named EMPIR. The curve is merely a visual reminder of the definition of the symbol.

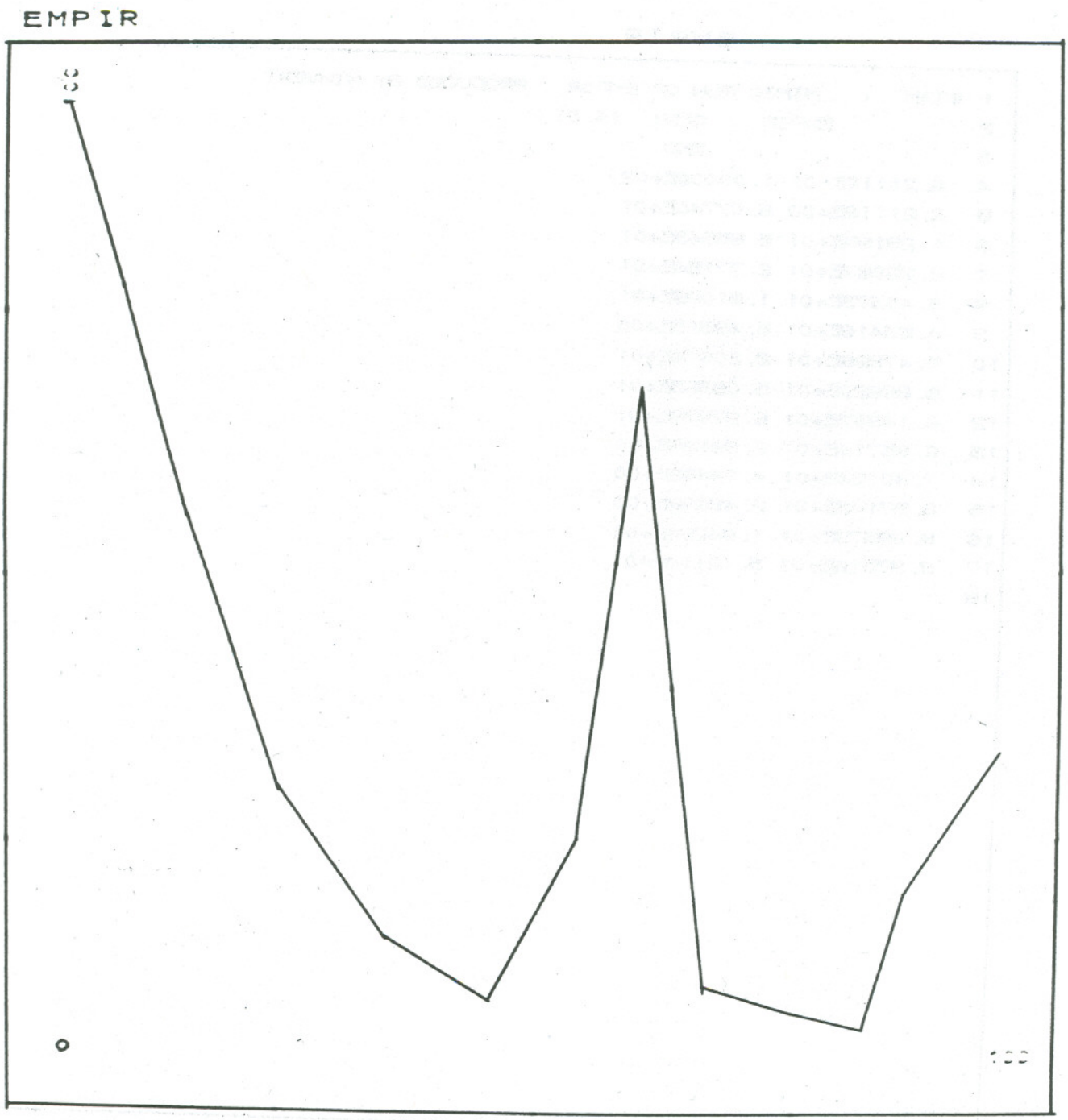


Fig. 11b. The empirical definition for the element EMPIR is a hand-drawn curve with the scale specified by horizontal and vertical labels.

EMPIR

```
 1 *IAM       MIMIC RUN OF EMPIR   PRODUCED BY MIMVERT
 2           EMPIR    OPN( 14.0)
 3                    END
 4  6.21118E-01 1.00000E+02
 5  6.21118E+00 8.07740E+01
 6  1.29193E+01 5.68643E+01
 7  2.26087E+01 2.77154E+01
 8  3.40575E+01 1.21099E+01
 9  4.53416E+01 5.49313E+00
10  5.47826E+01 2.20974E+01
11  5.88820E+01 5.09365E+01
12  6.14907E+01 6.97878E+01
13  6.85714E+01 6.99126E+00
14  7.80124E+01 4.24469E+00
15  8.57143E+01 2.49688E+00
16  8.99379E+01 1.64794E+01
17  9.97516E+01 3.12110E+01
18
```

Fig. 11c. The output of PICASSO's analyzer in a form suitable for input to MIMIC.

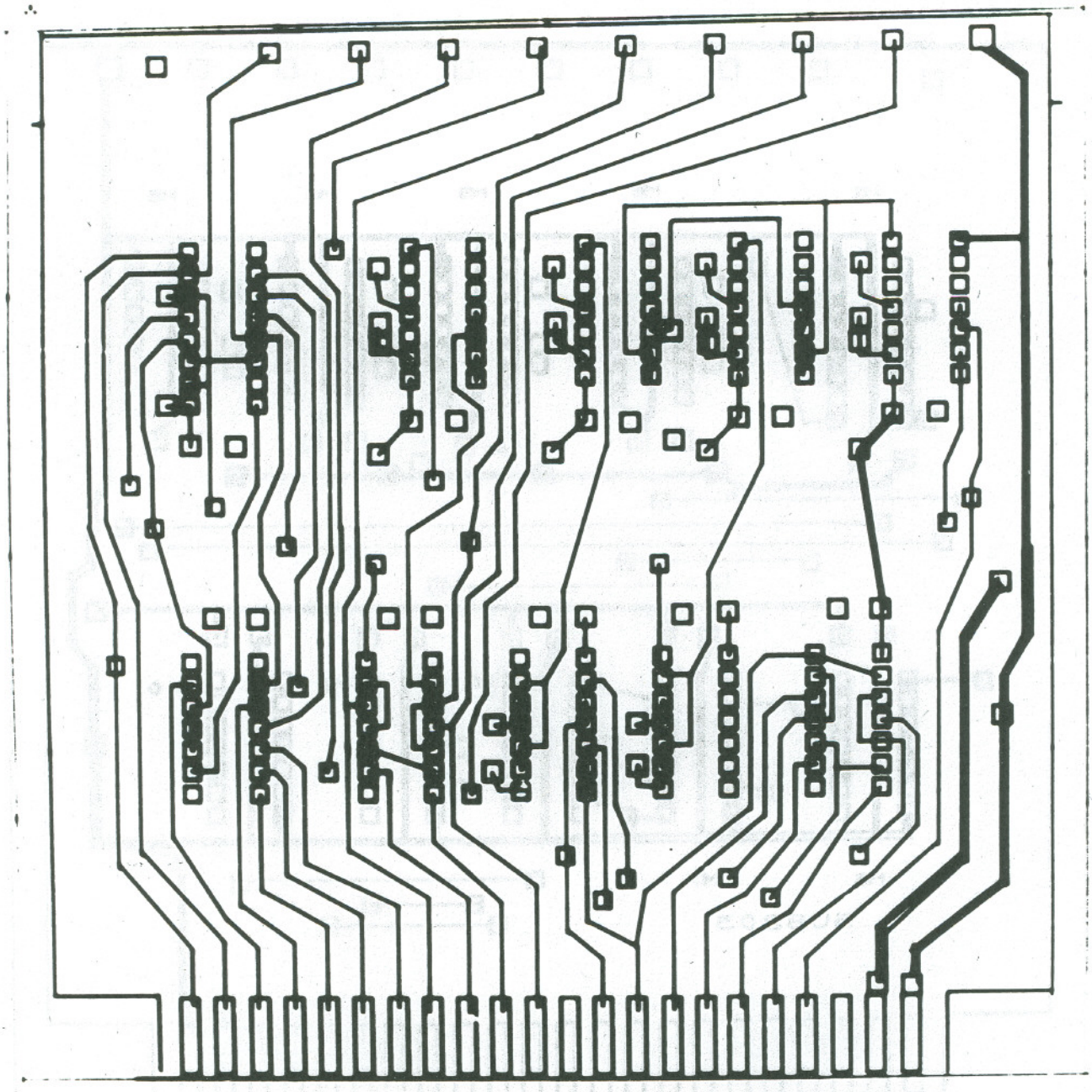


Fig. 12a. The wiring side of a prototype printed circuit board layout designed by Horace Warnock. When completed the picture will be redrawn on microfilm with the proper line widths, and a film mask created directly from the microfilm.

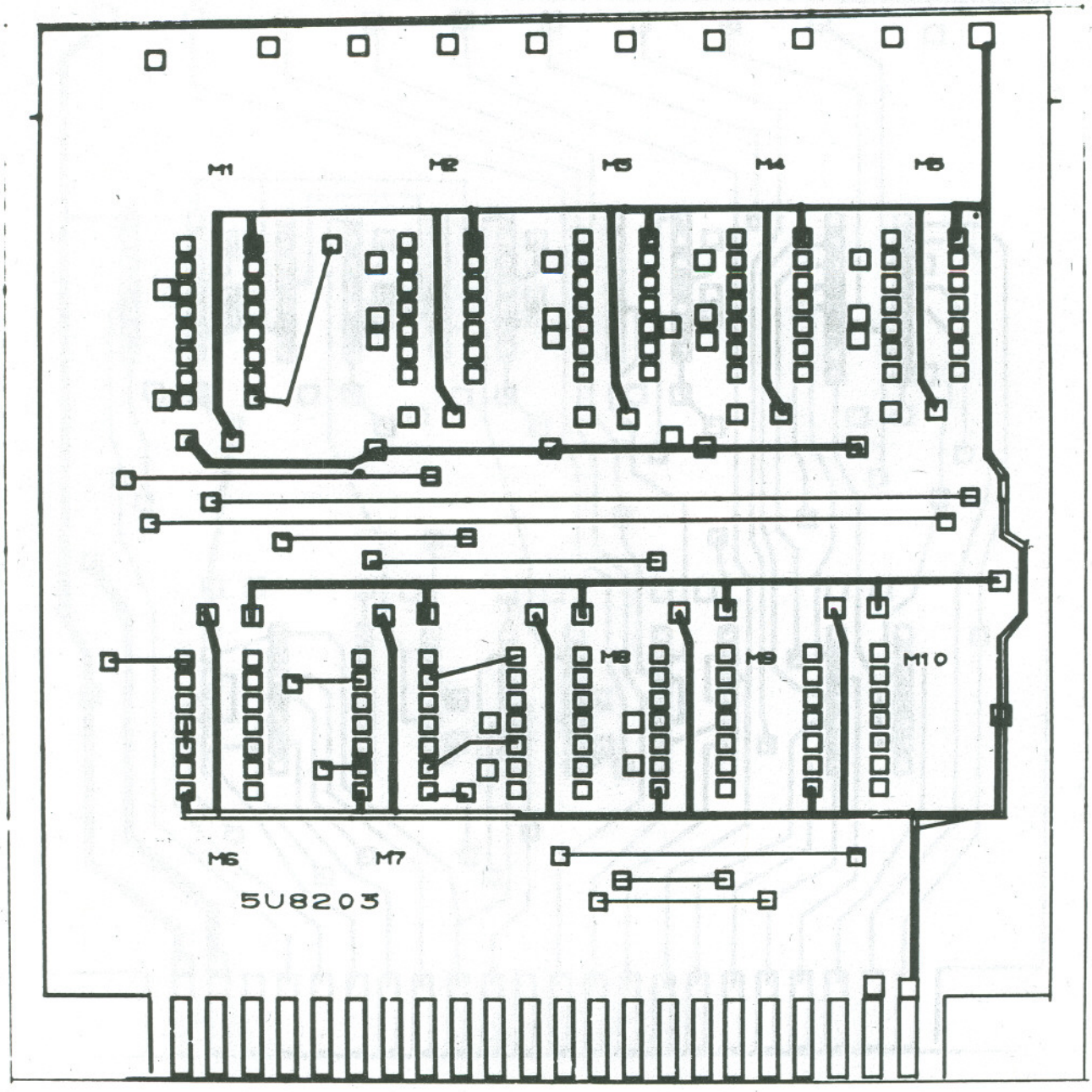


Fig. 12b. The component side of the board.

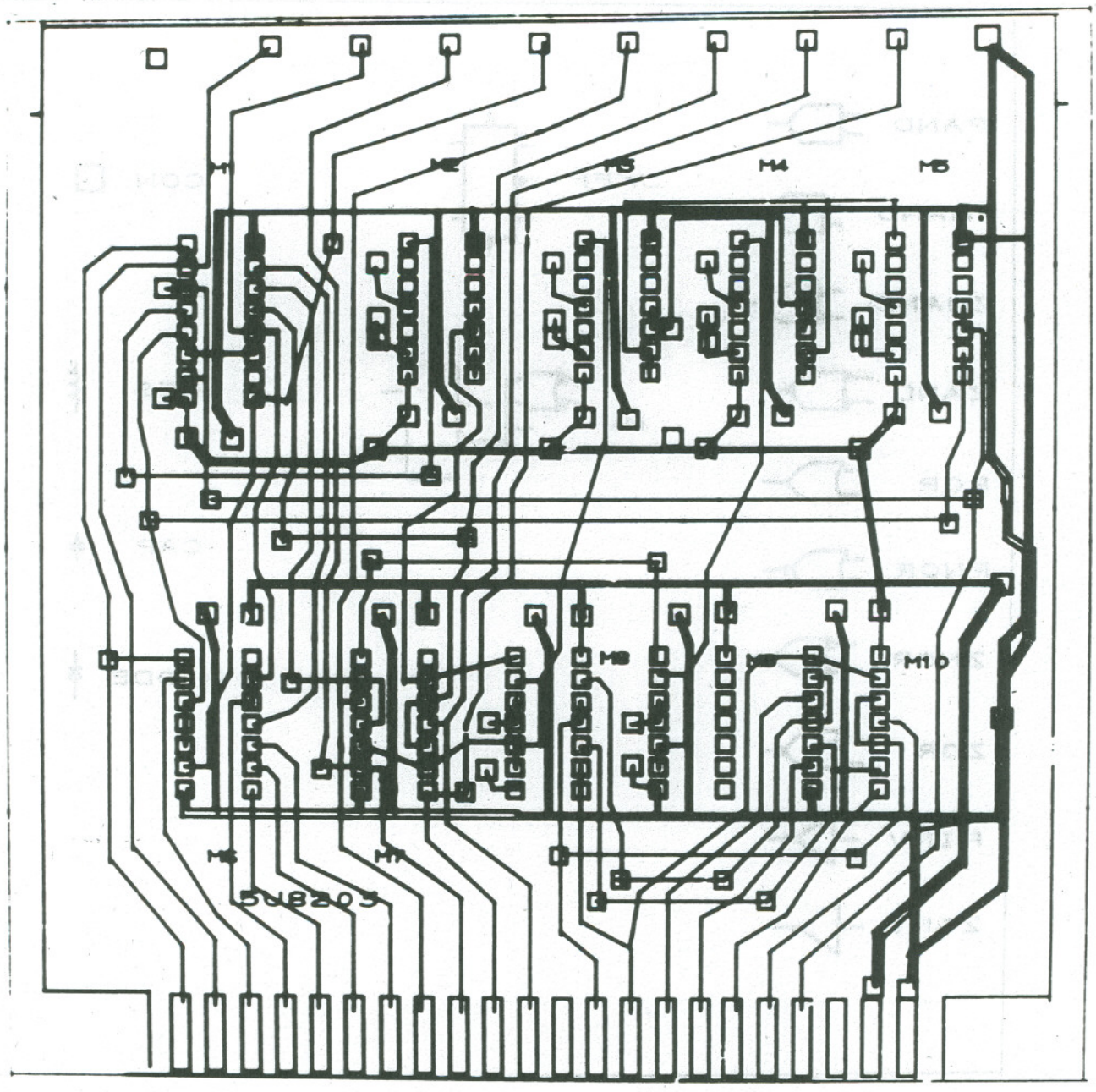


Fig. 12c. A composite showing the connections on both sides, for documentation.

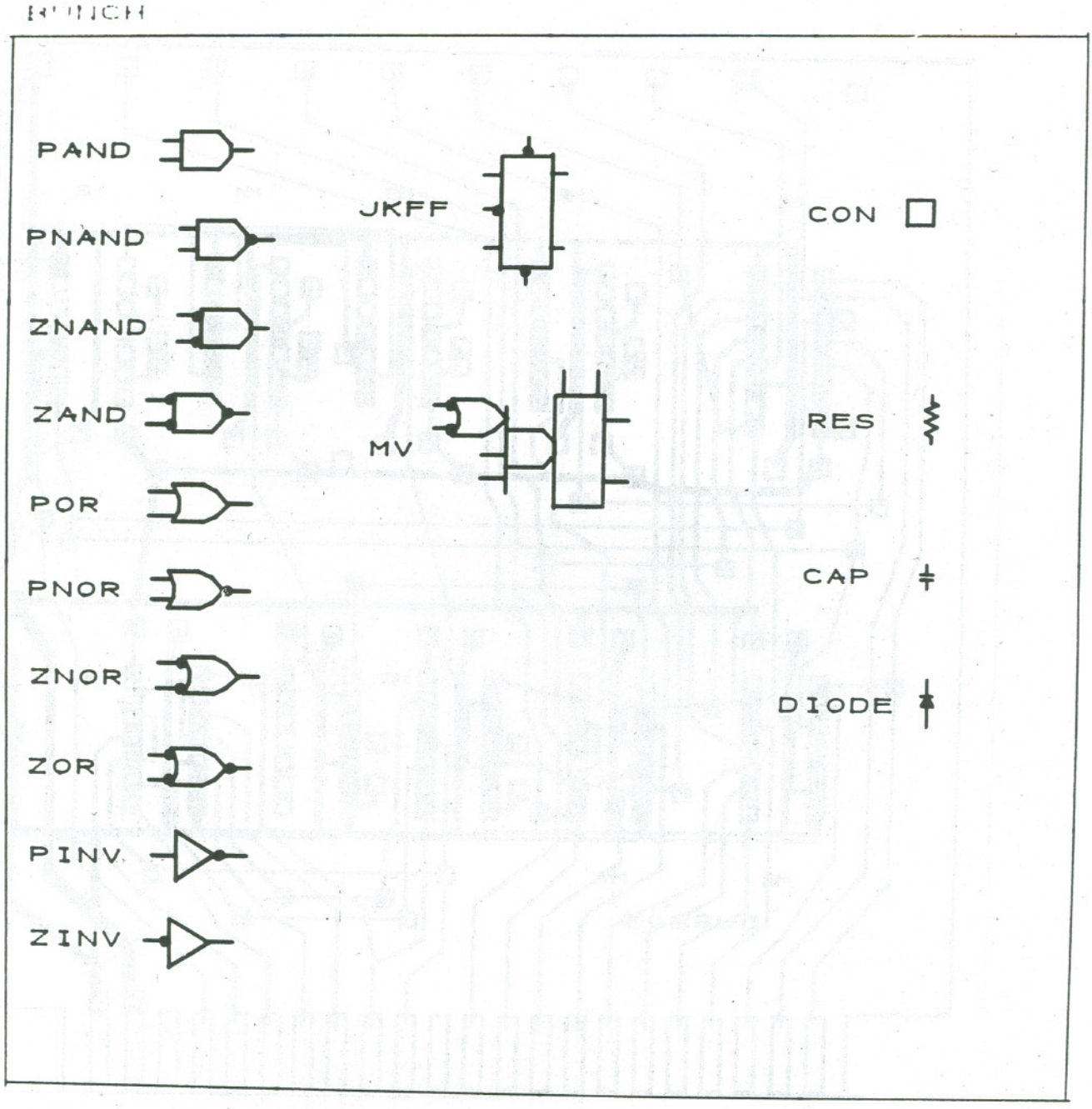


Fig. 12d. Some symbols used in constructing logic diagrams and circuit board layout documentation.

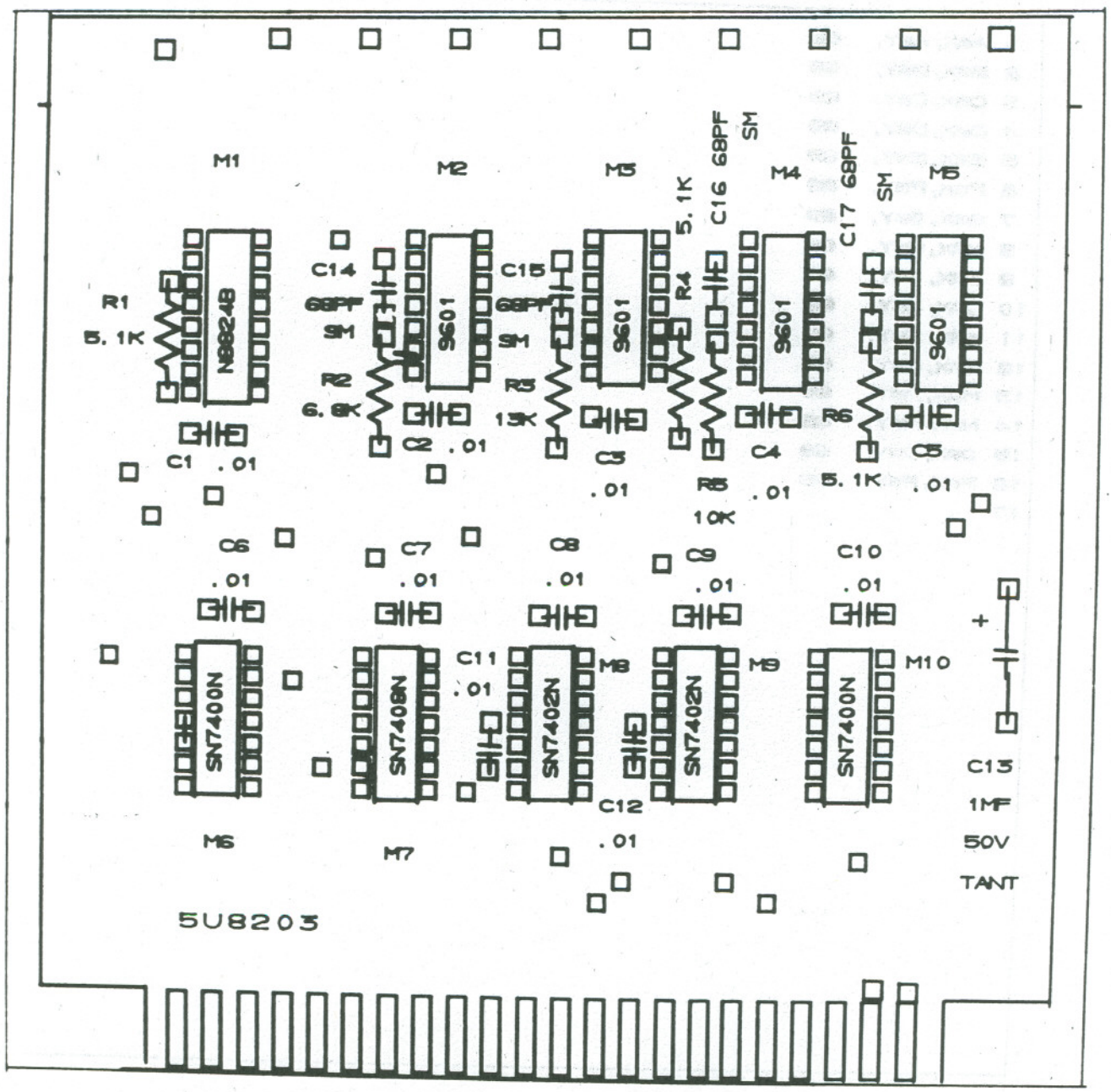


Fig. 12e. The component layout of the board.

M16

```
 1 A≠X, A≠Y,   69
 2 B≠X, B≠Y,   69
 3 C≠X, C≠Y,   69
 4 D≠X, D≠Y,   69
 5 E≠X, E≠Y,   69
 6 F≠X, F≠Y,   69
 7 G≠X, G≠Y,   69
 8 H≠X, H≠Y,   69
 9 I≠X, I≠Y,   69
10 J≠X, J≠Y,   69
11 K≠X, K≠Y,   69
12 L≠X, L≠Y,   69
13 M≠X, M≠Y,   69
14 N≠X, N≠Y,   69
15 O≠X, O≠Y,   69
16 P≠X, P≠Y,   69
17
```

Fig. 12f. The text definition of the 16-pin can M16 in drill tape format. The notation A ≠ X indicates that the X coordinate of the attacher point labeled A is to be output.

SU8203 DRILL TAPE

487,	3468, 62
862,	3506, 56
1162,	3506, 56
1462,	3506, 56
1762,	3506, 56
2062,	3506, 56
2362,	3506, 56
2662,	3506, 56
2962,	3506, 56
3262,	3506, 39
332,	1408, 69
392,	2032, 69
466,	1884, 69
516,	2306, 62
520,	2684, 62
584,	1416, 69
584,	1341, 69
584,	1266, 69
584,	1191, 69
584,	1116, 69
584,	1041, 69
584,	966, 69
809,	966, 69
809,	1041, 69
809,	1116, 69
809,	1191, 69
809,	1266, 69
809,	1341, 69
809,	1416, 69
590,	2168, 62
658,	1568, 62
670,	1954, 69
742,	2164, 62
806,	1564, 62
590,	2830, 69
590,	2755, 69
590,	2680, 69
590,	2605, 69
590,	2530, 69
590,	2455, 69
590,	2380, 69
590,	2305, 69
815,	2305, 69
815,	2380, 69
815,	2455, 69
815,	2530, 69
815,	2605, 69
815,	2680, 69
815,	2755, 69
815,	2830, 69
1814,	2764, 62
1818,	2572, 62
1818,	2508, 62
1840,	752, 69
1914,	1562, 62
1920,	2840, 69
1920,	2765, 69
1920,	2690, 69
1920,	2615, 69
1920,	2540, 69
1920,	2465, 69
1920,	2390, 69
2145,	2390, 69
2145,	2465, 69
2145,	2540, 69
2145,	2615, 69
2145,	2690, 69
2145,	2765, 69
2145,	2840, 69
1922,	2240, 62
1968,	600, 69
2046,	672, 69
2070,	2232, 62
2086,	1054, 62
2088,	1198, 62
2168,	1744, 69
2174,	1426, 69
2174,	1351, 69
2174,	1276, 69
2174,	1201, 69
2174,	1126, 69
2174,	1051, 69
2174,	976, 69
2399,	976, 69
2399,	1051, 69
2399,	1126, 69
2399,	1201, 69
2399,	1276, 69
2399,	1351, 69
2399,	1426, 69
2216,	2540, 62
2220,	2166, 62
2244,	1568, 62
2330,	2130, 62
2332,	2762, 62
2334,	2580, 62
2340,	2494, 62
2396,	670, 69
2400,	1574, 62
2438,	2834, 69

Fig. 12g. The first page of the drill tape output for the board. This information will be sorted by drill code and punched on paper tape to drive the drill.

```
 908,   1816, 69
 938,   1332, 69
1040,   1038, 69
1074,   2832, 69
1170,   1416, 69
1170,   1341, 69
1170,   1266, 69
1170,   1191, 69
1170,   1116, 69
1170,   1041, 69
1170,    966, 69
1395,    966, 69
1395,   1041, 69
1395,   1116, 69
1395,   1191, 69
1395,   1266, 69
1395,   1341, 69
1395,   1416, 69
1206,   1750, 69
1220,   2132, 62
1224,   2766, 62
1228,   2580, 62
1226,   2508, 62
1246,   1562, 62
1328,   2836, 69
1328,   2761, 69
1328,   2686, 69
1328,   2611, 69
1328,   2536, 69
1328,   2461, 69
1328,   2386, 69
1553,   2386, 69
1553,   2461, 69
1553,   2536, 69
1553,   2611, 69
1553,   2686, 69
1553,   2761, 69
1553,   2836, 69
1330,   2242, 62
1394,   1566, 62
1406,   2038, 69
1482,   2244, 62
1522,    962, 69
1528,   1828, 69
1596,   1040, 62
1602,   1198, 62
1692,   1422, 69
1692,   1347, 69
1692,   1272, 69
1692,   1197, 69
2663,   2834, 69
2438,   2244, 62
2540,    602, 69
2588,   2244, 62
2690,   1426, 69
2690,   1351, 69
2690,   1276, 69
2690,   1201, 69
2690,   1126, 69
2690,   1051, 69
2690,    976, 69
2915,    976, 69
2915,   1051, 69
2915,   1126, 69
2915,   1201, 69
2915,   1276, 69
2915,   1351, 69
2915,   1426, 69
2766,   1580, 62
2842,   2128, 62
2840,    744, 69
2850,   2570, 62
2854,   2758, 62
2848,   2494, 62
2898,    314, 62
2912,   1578, 62
2954,   2248, 62
2958,   2828, 69
2958,   2753, 69
2958,   2678, 69
2958,   2603, 69
2958,   2528, 69
2958,   2453, 69
2958,   2378, 69
3183,   2378, 69
3183,   2453, 69
3183,   2528, 69
3183,   2603, 69
3183,   2678, 69
3183,   2753, 69
3183,   2828, 69
3014,    308, 62
3106,   2250, 62
3140,   1872, 69
3220,   1956, 69
3314,   1668, 62
3318,   1216, 62
        END
```

Fig. 12h. The second page of the drill tape output for the board.

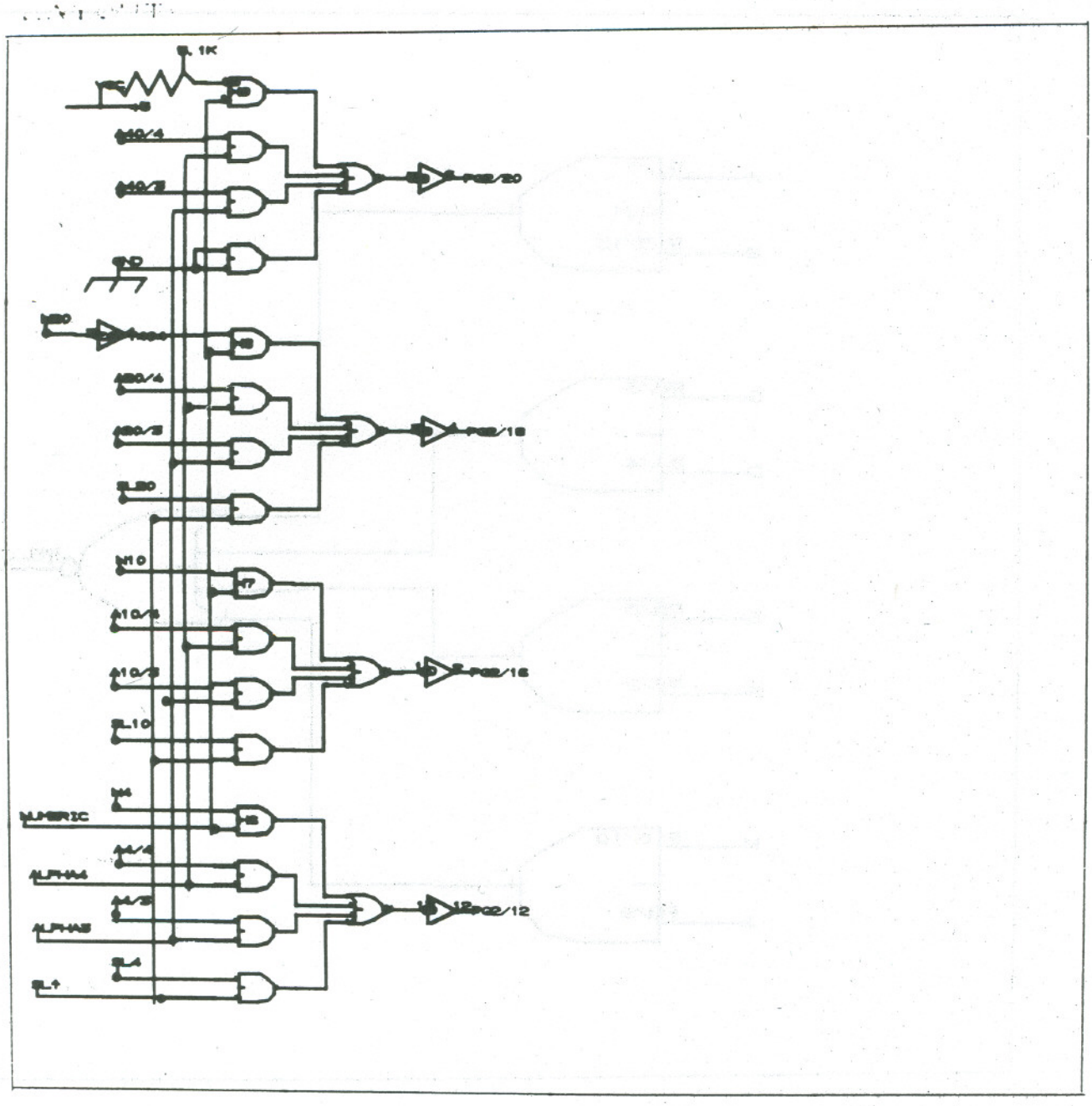


Fig. 13a. A logic layout problem provided by Frank Neu.

C7453

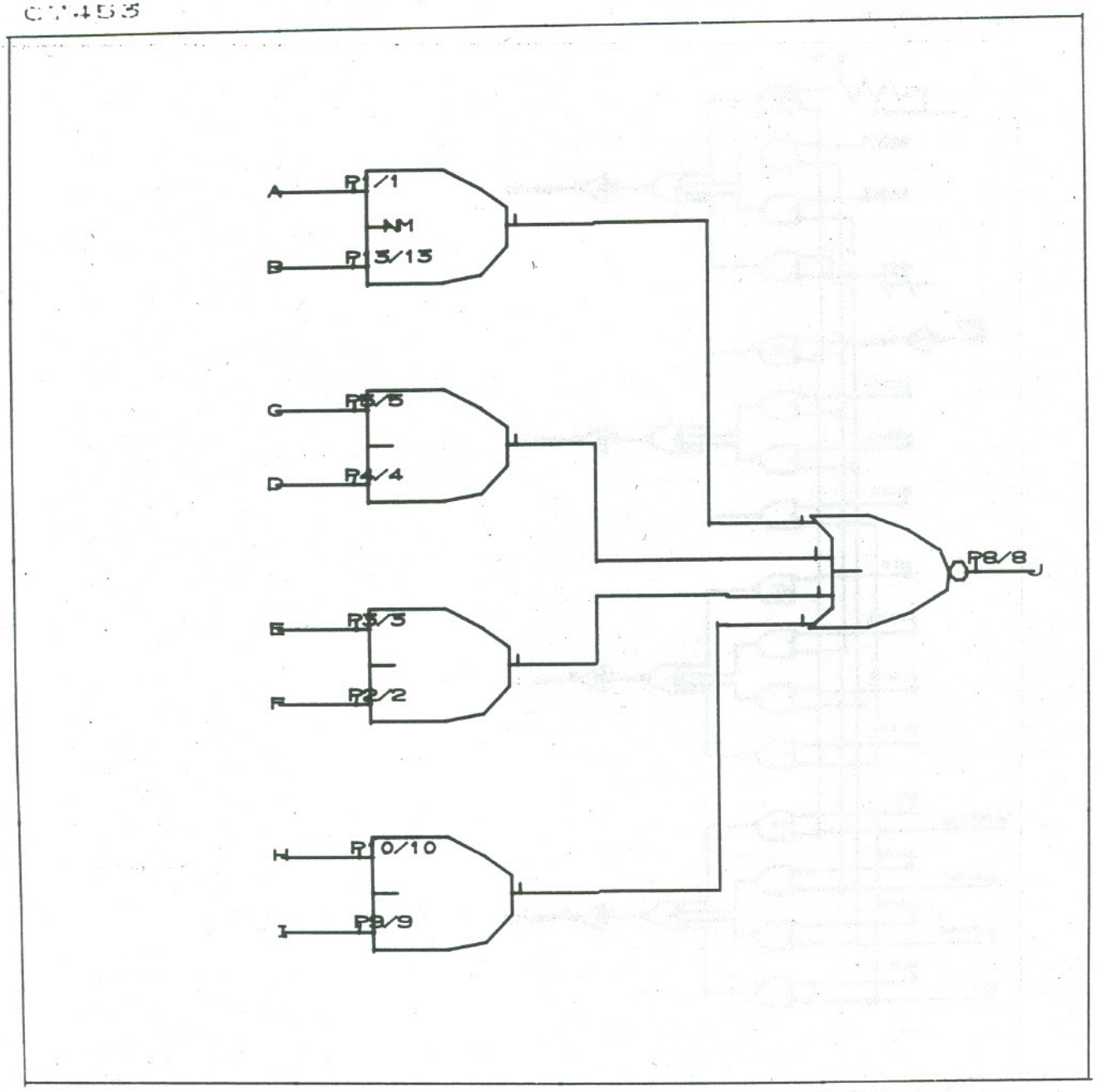


Fig. 13b. The symbol for the C7453 element used in the logic layout. The labels P1/1 define the default values for the pin numbers.

C7453

```
 1   A, 7453/-NM-0-P1
 2   B, 7453/-NM-0-P13
 3   C, 7453/-NM-0-P5
 4   D, 7453/-NM-0-P4
 5   E, 7453/-NM-0-P3
 6   F, 7453/-NM-0-P2
 7   H, 7453/-NM-0-P10
 8   I, 7453/-NM-0-P9
 9   J, 7453/-NM-0-P8
10
```

Fig. 13c. The text definition of C7453 in a form suitable for a Connect-List or wire wrap program.

LAYOUT

```
 1 CONNECT LIST FOR LAYOUT (PARTIALL DRAWING) PRODUCED BY PICASSO
 2  N20, C6-7404-05
 3  G002, C6-7404-04
 4  VCC, +5VOLTS
 5  GND, GROUND
 6  G004, 7453/H9-013
 7  NUMERIC, 7453/H9-01
 8  A40/4, 7453/H9-05
 9  ALPHA4, 7453/H9-04
10  A40/3, 7453/H9-03
11  ALPHA3, 7453/H9-02
12  GND, 7453/H9-010
13  GND, 7453/H9-09
14  G005, 7453/H9-08
15  G002, 7453/H8-01
16  NUMERIC, 7453/H8-013
17  A20/4, 7453/H8-05
18  ALPHA4, 7453/H8-04
19  A20/3, 7453/H8-03
20  ALPHA3, 7453/H8-02
21  SL20, 7453/H8-010
22  SL↑, 7453/H8-09
23  G006, 7453/H8-08
24  N10, 7453/H7-01
25  NUMERIC, 7453/H7-013
26  A10/4, 7453/H7-0G003
27  ALPHA4, 7453/H7-0G003
28  A10/3, 7453/H7-03
29  ALPHA3, 7453/H7-02
30  SL10, 7453/H7-010
31  SL↑, 7453/H7-09
32  G007, 7453/H7-08
33  N4, 7453/H6-01
34  NUMERIC, 7453/H6-013
35  A4/4, 7453/H6-05
```

Fig. 13d. Page 1 of the output from the analysis of LAYOUT indicating the signal names associated with the pins of the elements. The information can easily be sorted and converted to the proper format for a wire wrap program.

LAYOUT

```
36   ALPHA4, 7453/HS-04
37   A4/3, 7453/HS-03
38   ALPHA3, 7453/HS-02
39   SL4, 7453/HS-010
40   SL↑, 7453/HS-09
41   G008, 7453/HS-08
42   VCC, 5. 1KOHM
43   G004, 5. 1KOHM
44   G005, D-05
45   PG2/20, D-06
46   G006, D-03
47   PG2/18, D-04
48   G007, D-01
49   PG2/16, D-02
50   G008, D-013
51   PG2/12, D-012
52                    END
53
```

Fig. 13e. Page 2 of the output from LAYOUT.

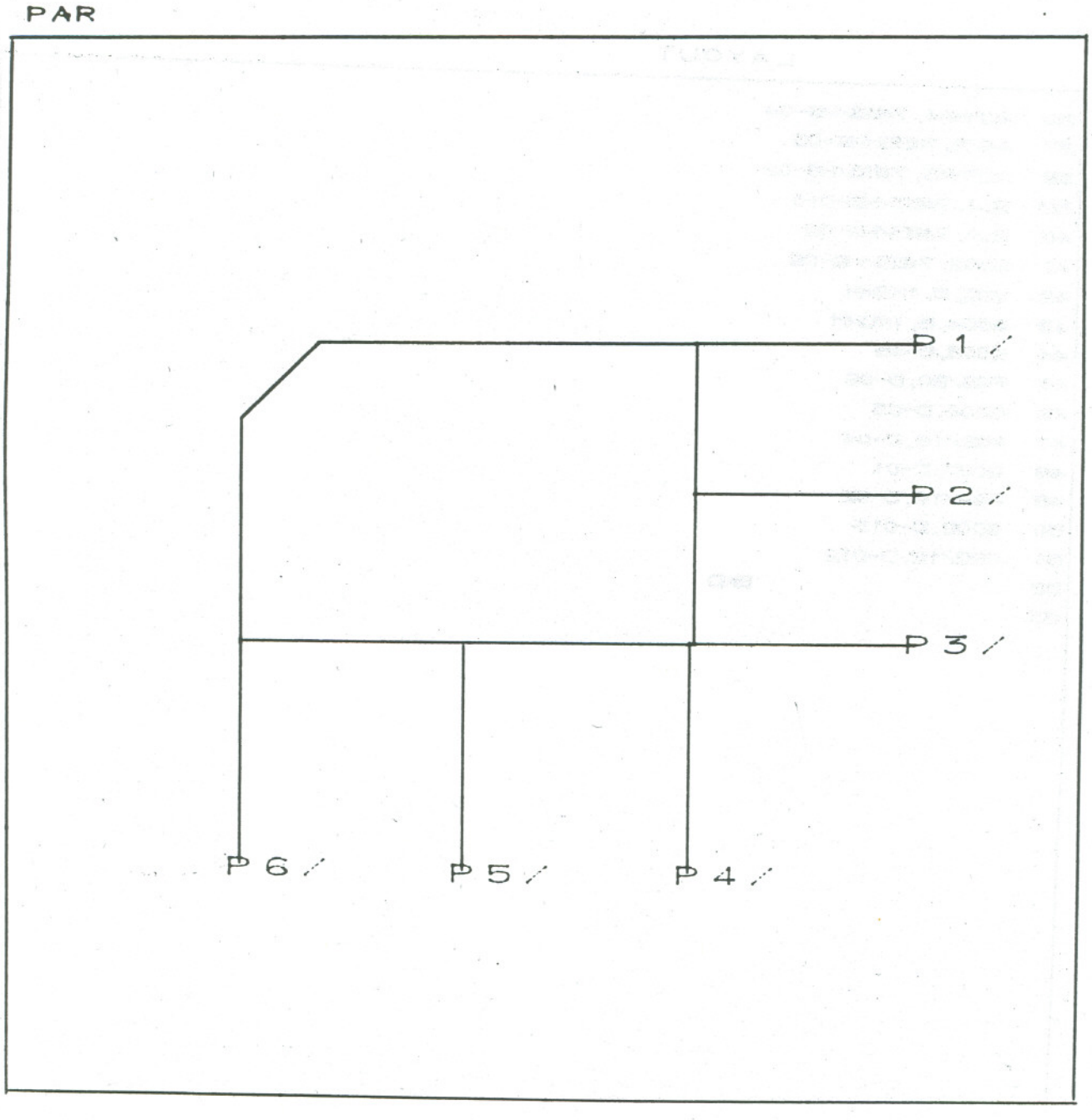


Fig. 14a. The symbol for the MIMIC PAR function, which allows parameters to be set on-line. Note the default specification P1/ which indicates that the parameter and its preceding comma are to be deleted if left an specified.

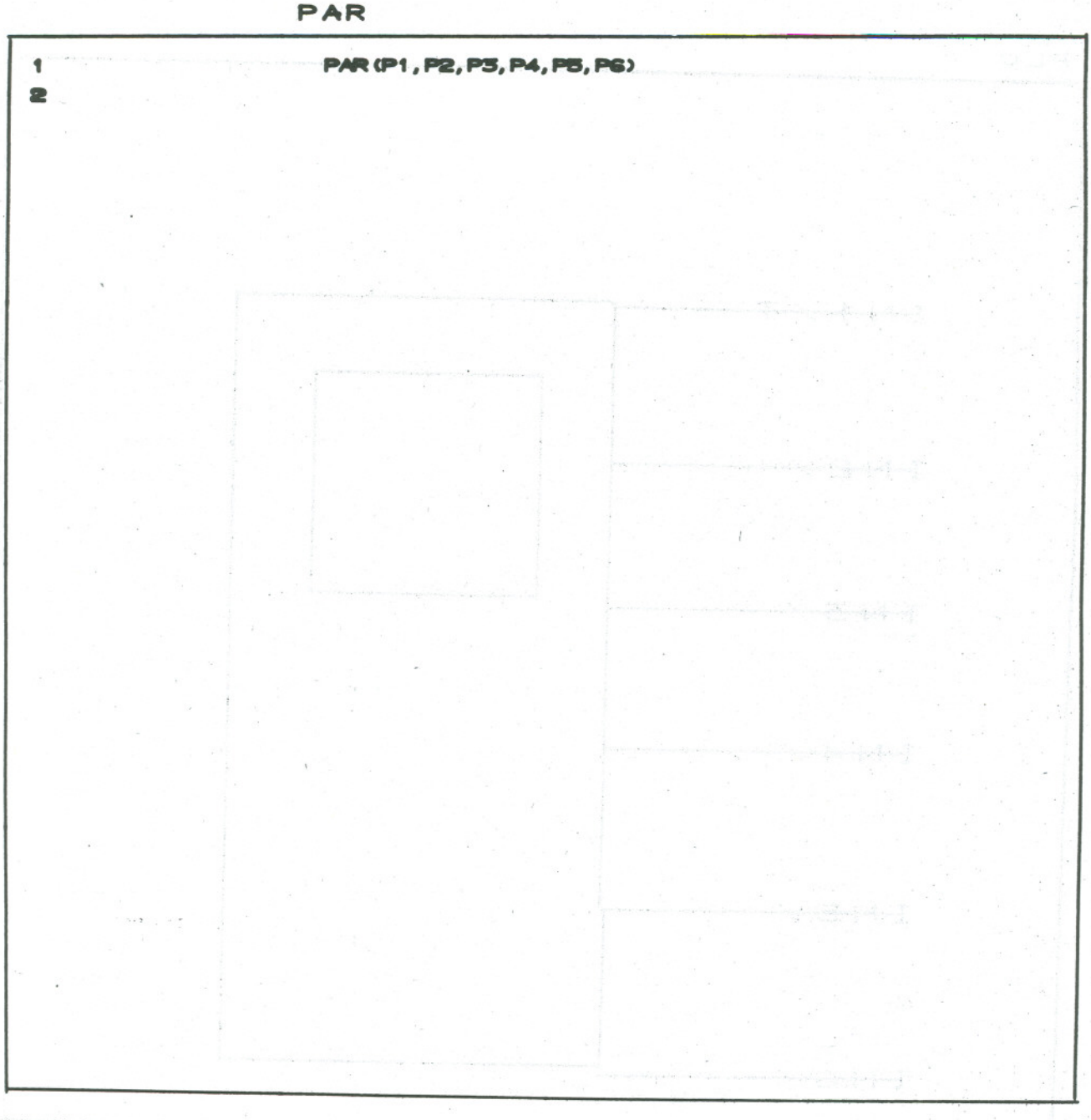


Fig. 14b. The text definition for PAR.

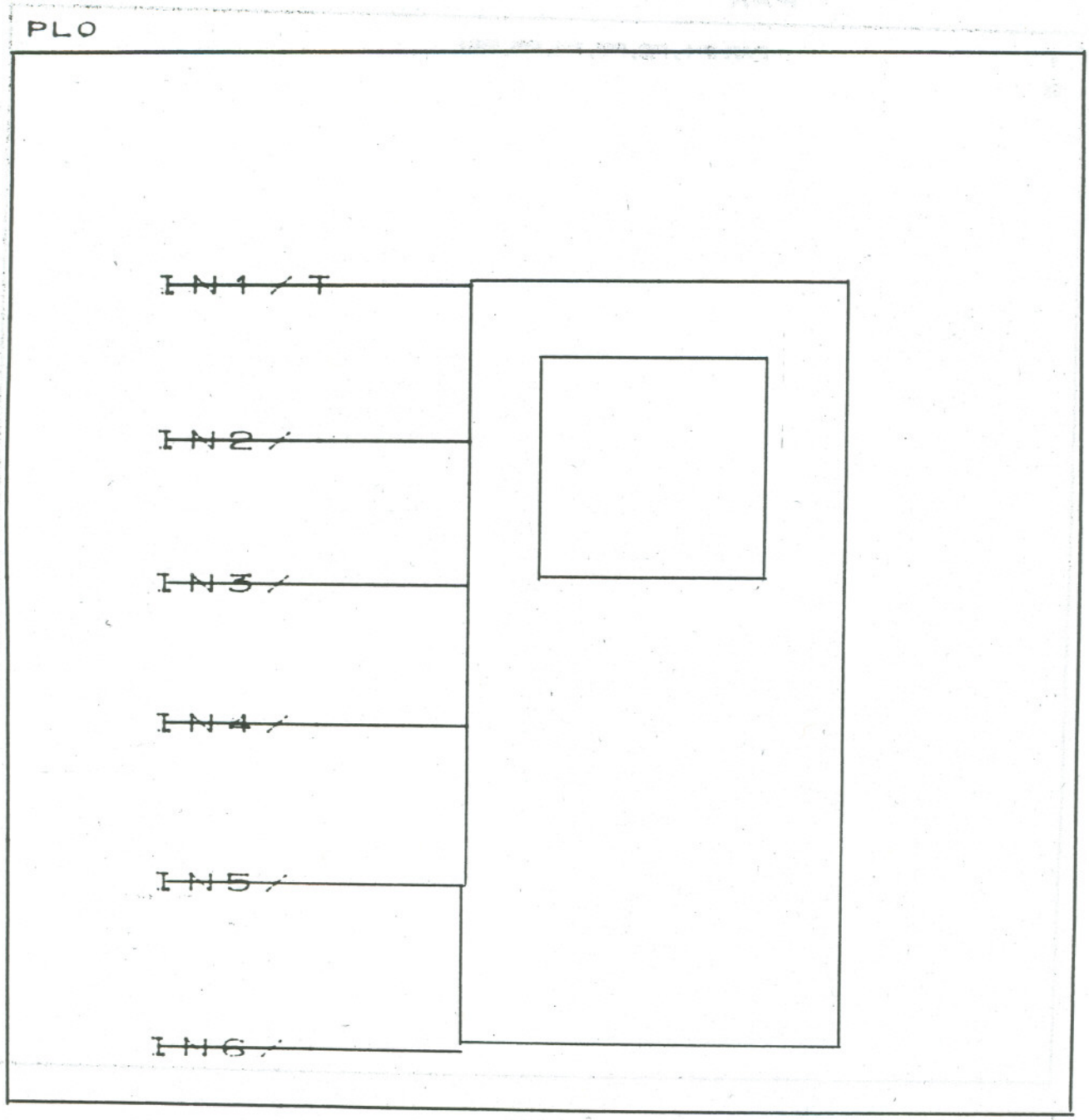


Fig. 15a. The symbol for the MIMIC PLO (plot) function, which specifies which functions are to be plotted. The default specification IN1/T indicates that if IN1 is not specified, the variable T is to be substituted for IN1.

PLO

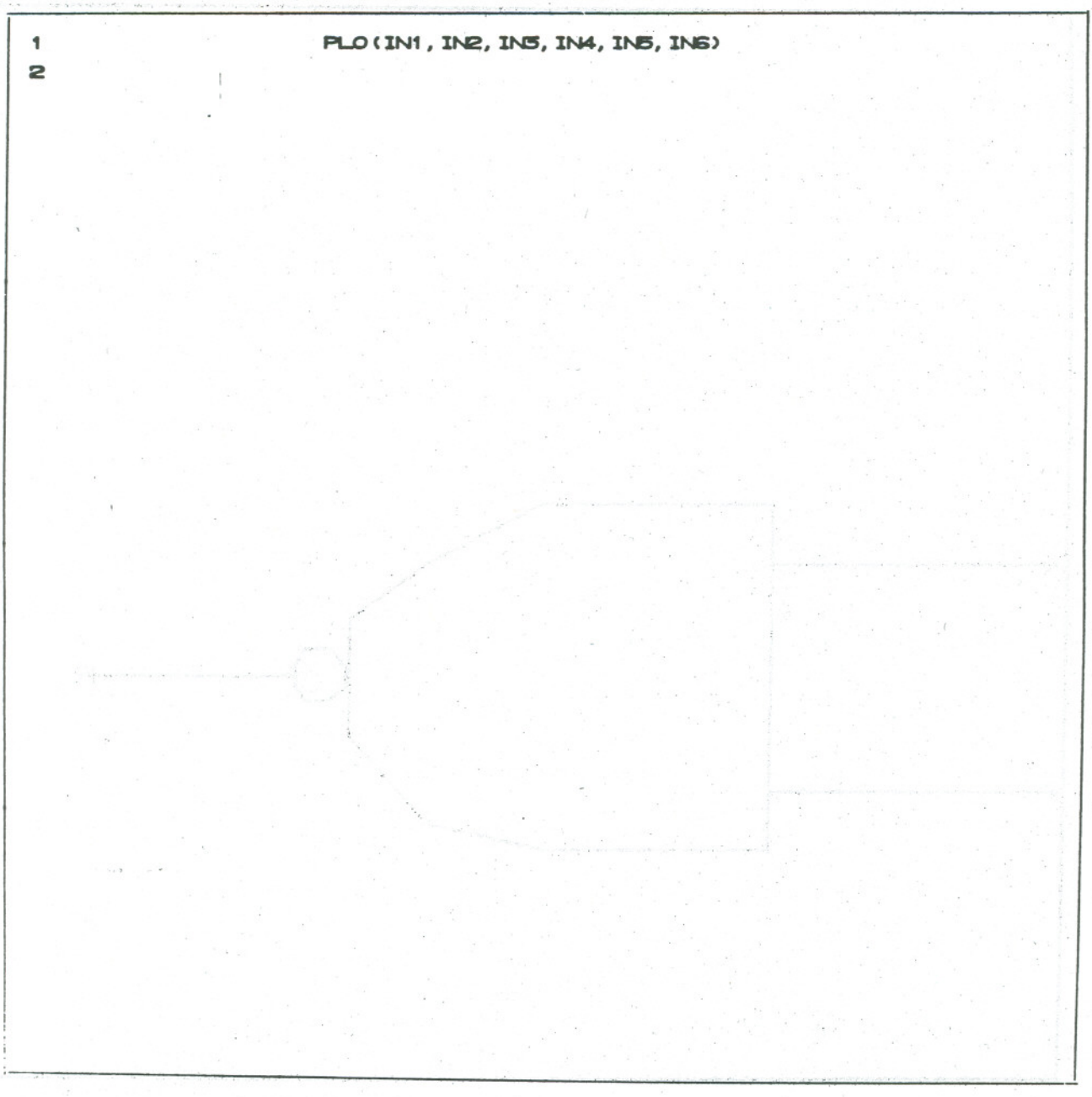


Fig. 15b. The text definition for PLO.

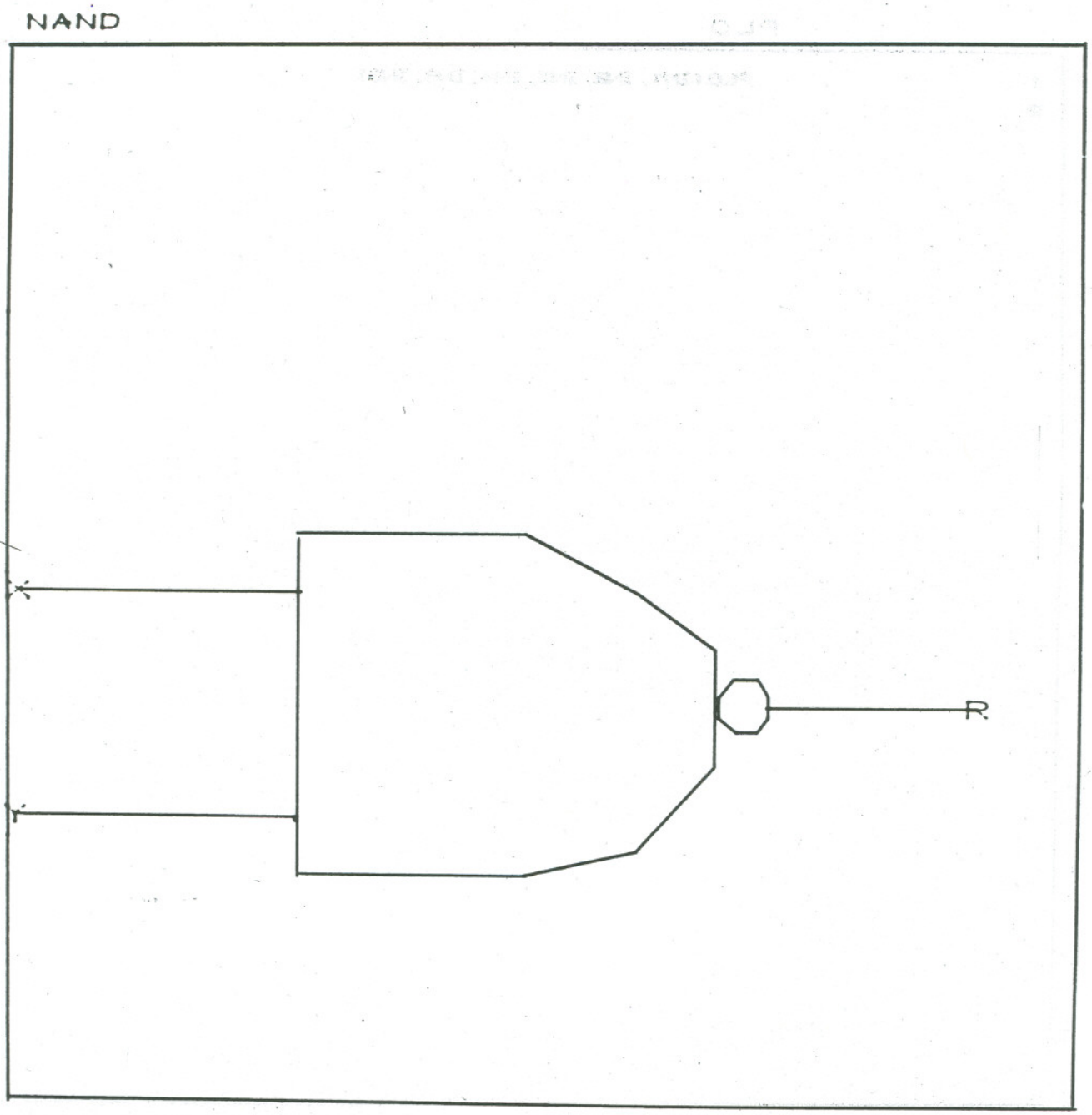


Fig. 16a. The symbol (zoomed by a factor 12) for the NAND gate.

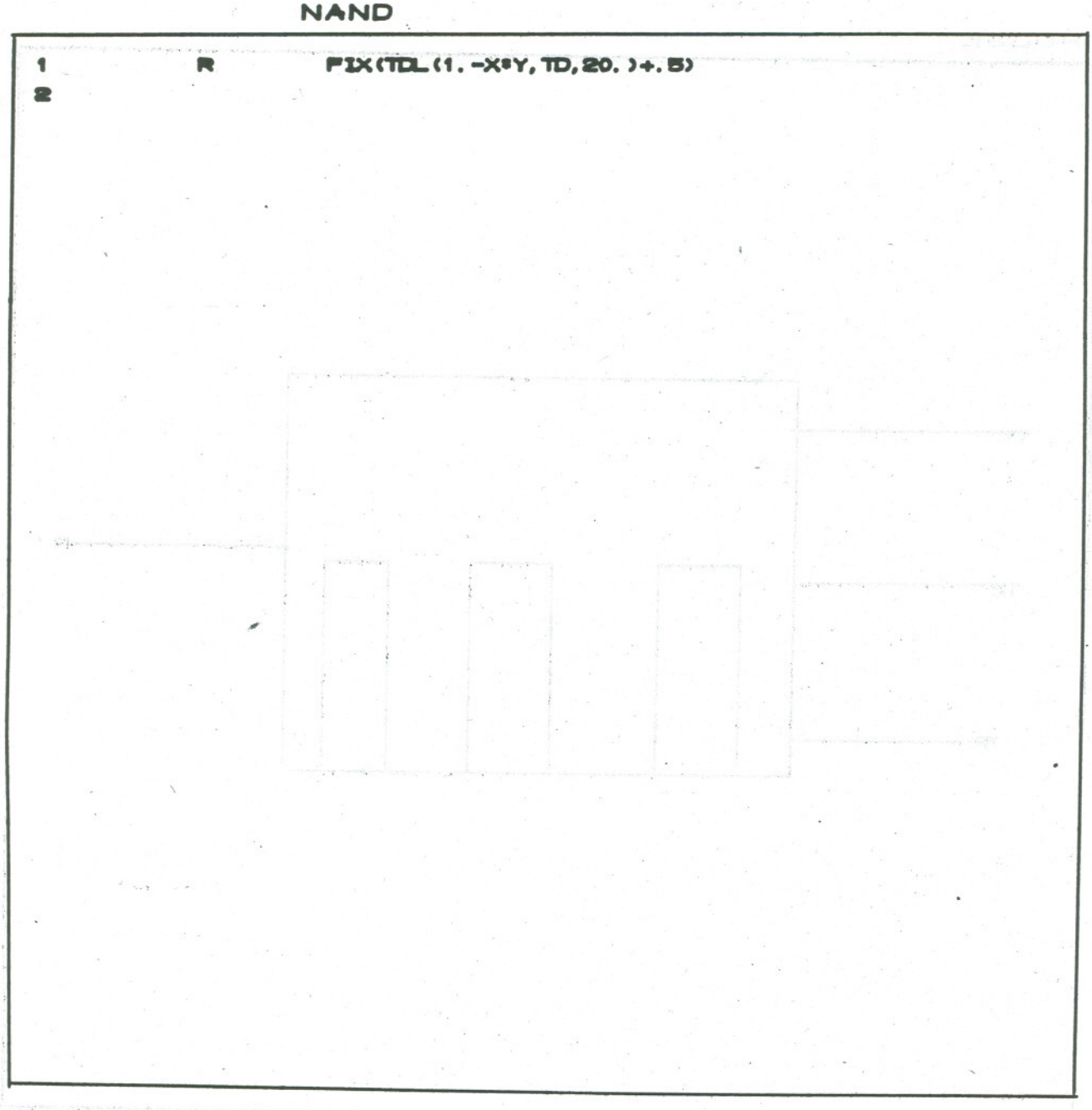


Fig. 16b. The text definition for the NAND gate in terms of the MIMIC time-delay functions TDL. The logic function is 1-X*Y, and the result, delayed by a time TD is FIXed to a value of 0.0 or 1.0.

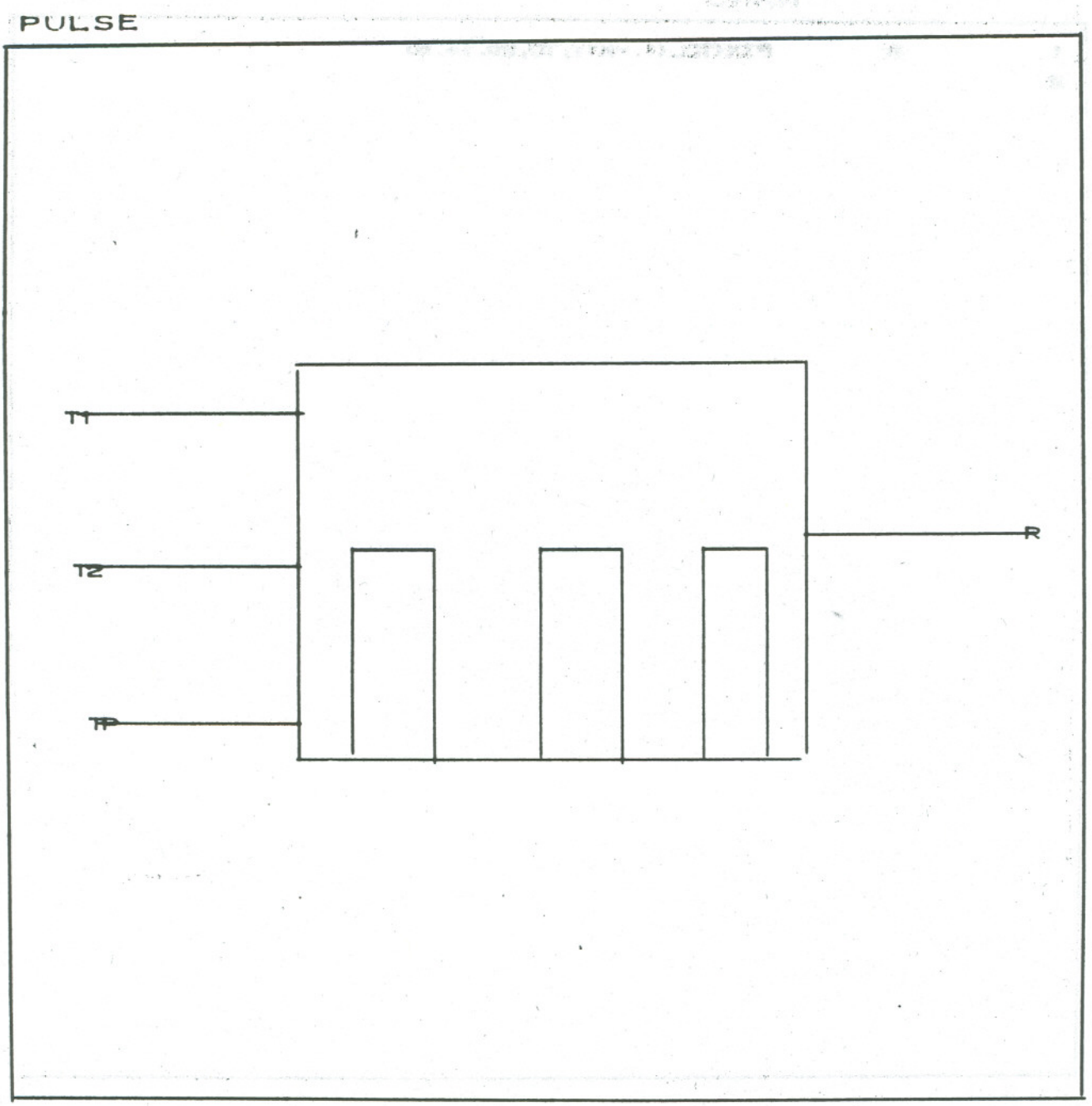


Fig. 17a. The symbol for the PULSE function, which generates a square pulse from T1 to T2, with a period TP.

PULSE

```
1          R       FSW(SIN(PI*(T-T1)/TP)*SIN(PI*(T-T2)/TP),1.,0.,0.)
2
```

Fig. 17b. The text definition of the PULSE in terms of the MIMIC function switch FSW.

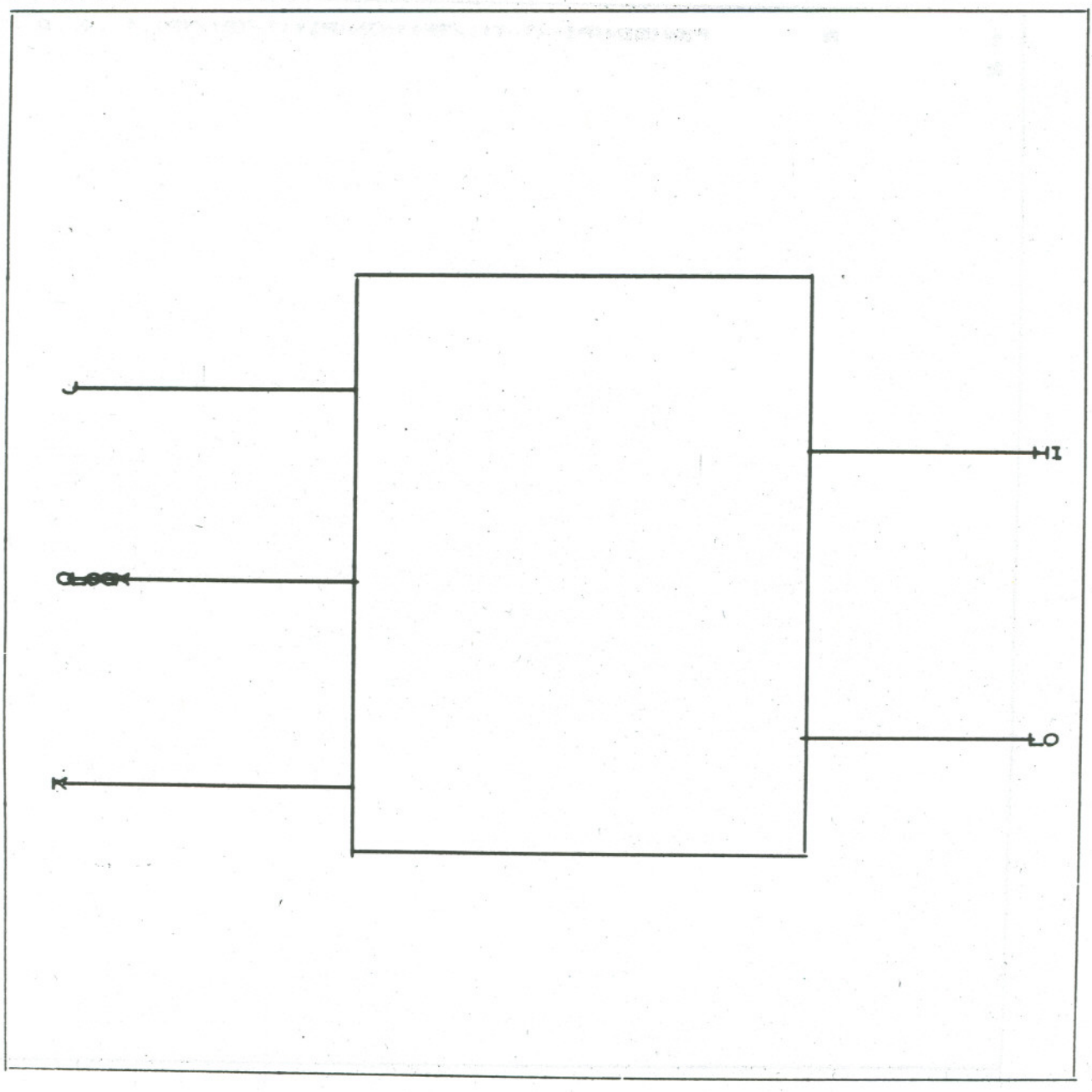


Fig. 18a. The symbol for JKFF (a Master-Slave J K Flip-Flop).

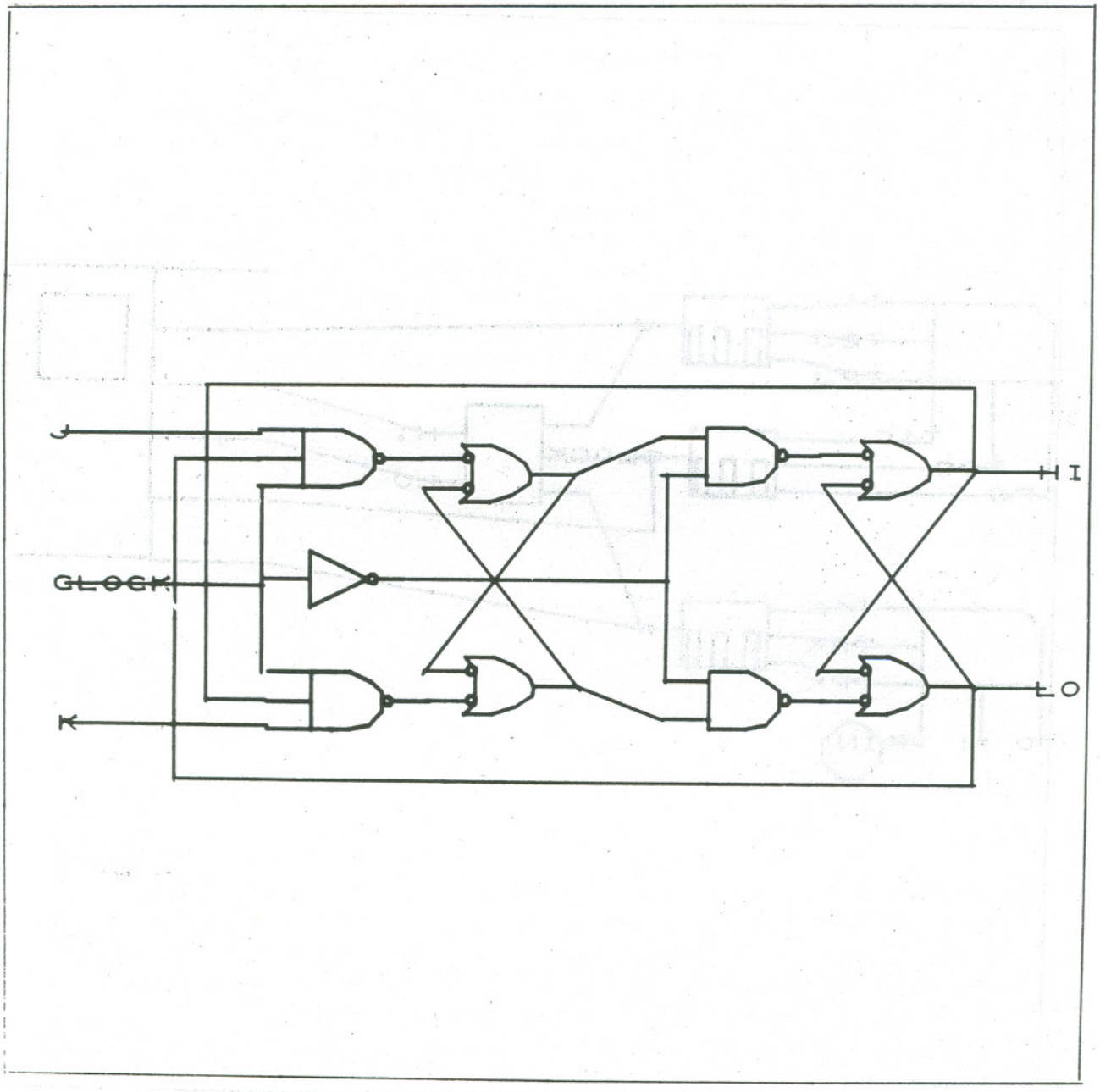


Fig. 18b. The macro definition of JKFF in terms of NAND, NOR and INV gates. The gates are primitive elements defined by MIMIC time-delay functions. This macro was constructed as specified in Digital Logic Handbook, 1970, p. 13 (Digital Equipment Corporation, Maynard, Massachusetts.)

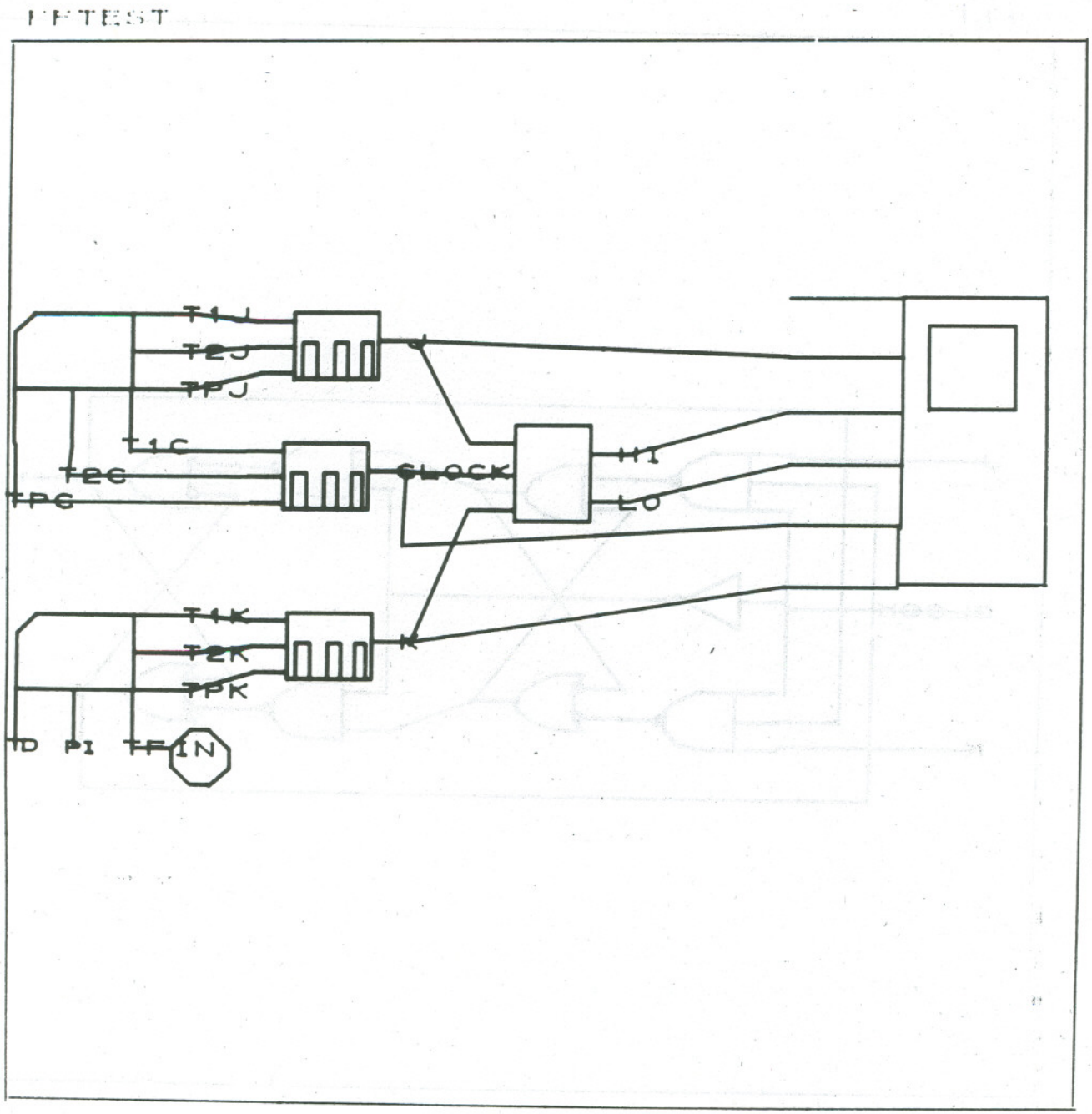


Fig. 18c. The model FFTEST designed to test the JKFF. Three input pulses (with parameters to be set on-line using interactive MIMIC program) and a plotter have been connected to the JKFF.

FFTEST

```
 1 *IAM      MIMIC RUN OF FFTEST  PRODUCED BY MIMVERT
 2           JKFF      BML(CLOCK ,HI     ,J     ,K     ,LO     )
 3           G003      FIX(TDL(1.-J*LO*CLOCK,TD,20.)+.5)
 4           G001      FIX(TDL(1.-CLOCK,TD,20.)+.5)
 5           G004      FIX(TDL(1.-CLOCK*HI*K,TD,20.)+.5)
 6           G005      FIX(TDL(1.-G003*G002,TD,20.)+.5)
 7           G002      FIX(TDL(1.-G005*G004,TD,20.)+.5)
 8           G006      FIX(TDL(1.-G005*G001,TD,20.)+.5)
 9           G007      FIX(TDL(1.-G001*G002,TD,20.)+.5)
10           HI        FIX(TDL(1.-G006*LO,TD,20.)+.5)
11           LO        FIX(TDL(1.-HI*G007,TD,20.)+.5)
12                     EMA
13                     FIN(T,TFIN)
14                     PAR(T1J,T2J,TPJ,T1C,T2C,TPC)
15                     PAR(T1K,T2K,TPK,TFIN,PI,TD)
16           CLOCK     FSW(SIN(PI*(T-T1C)/TPC)*SIN(PI*(T-T2C)/TPC),1.,0.,0.)
17           K         FSW(SIN(PI*(T-T1K)/TPK)*SIN(PI*(T-T2K)/TPK),1.,0.,0.)
18           J         FSW(SIN(PI*(T-T1J)/TPJ)*SIN(PI*(T-T2J)/TPJ),1.,0.,0.)
19           JKFF      CML(CLOCK ,HI     ,J     ,K     ,LO     )
20                     PLO(T,J,HI,LO,CLOCK,K)
21                     END
22
```

Fig. 18d. The output of PICASSO's analyzer for the model FFTEST. This program was written onto a disk file which MIMIC reads as input.

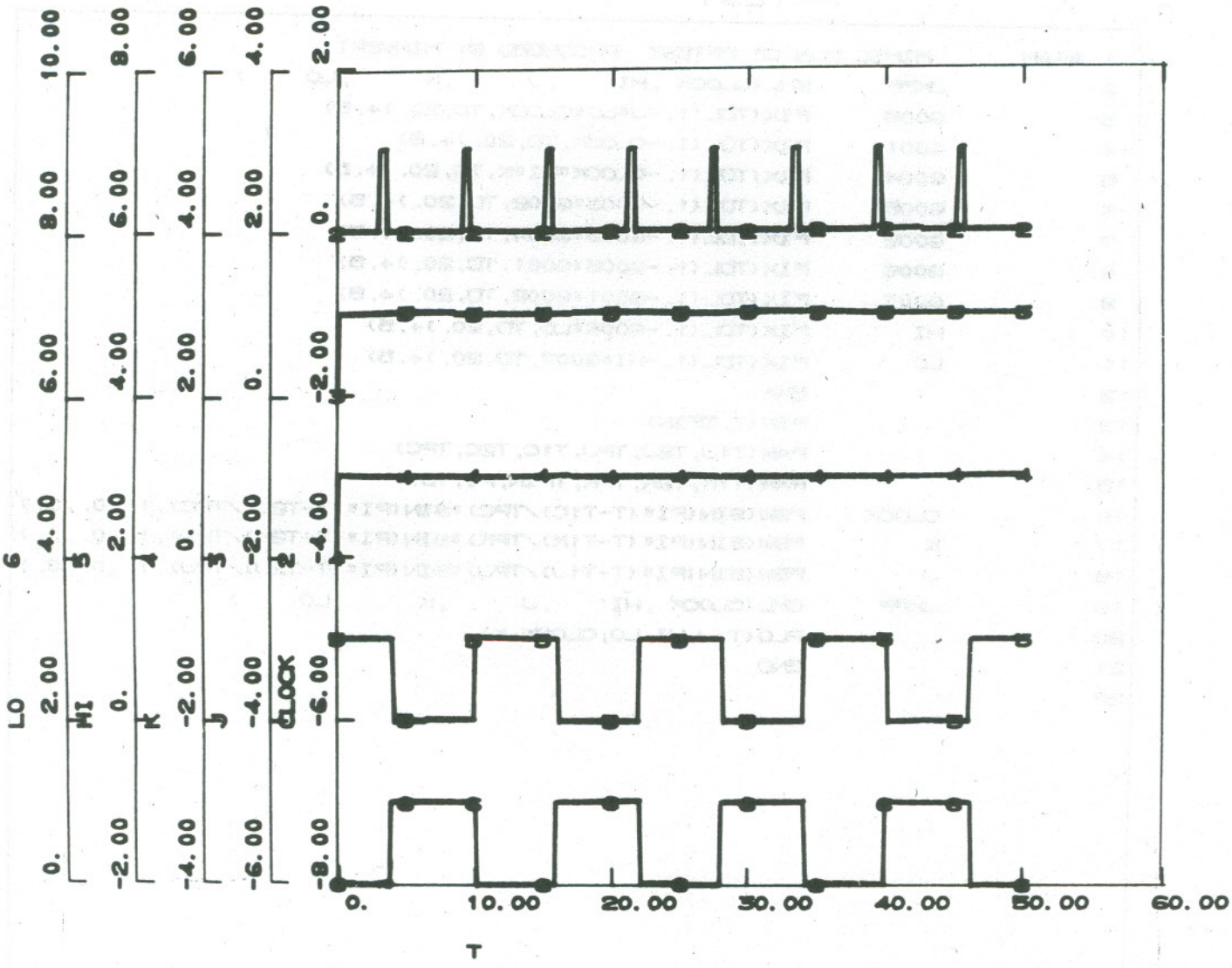


Fig. 18e. MIMIC output curves for one set of input pulse parameters, demonstrating the output of the JKFF for constant J and K inputs.

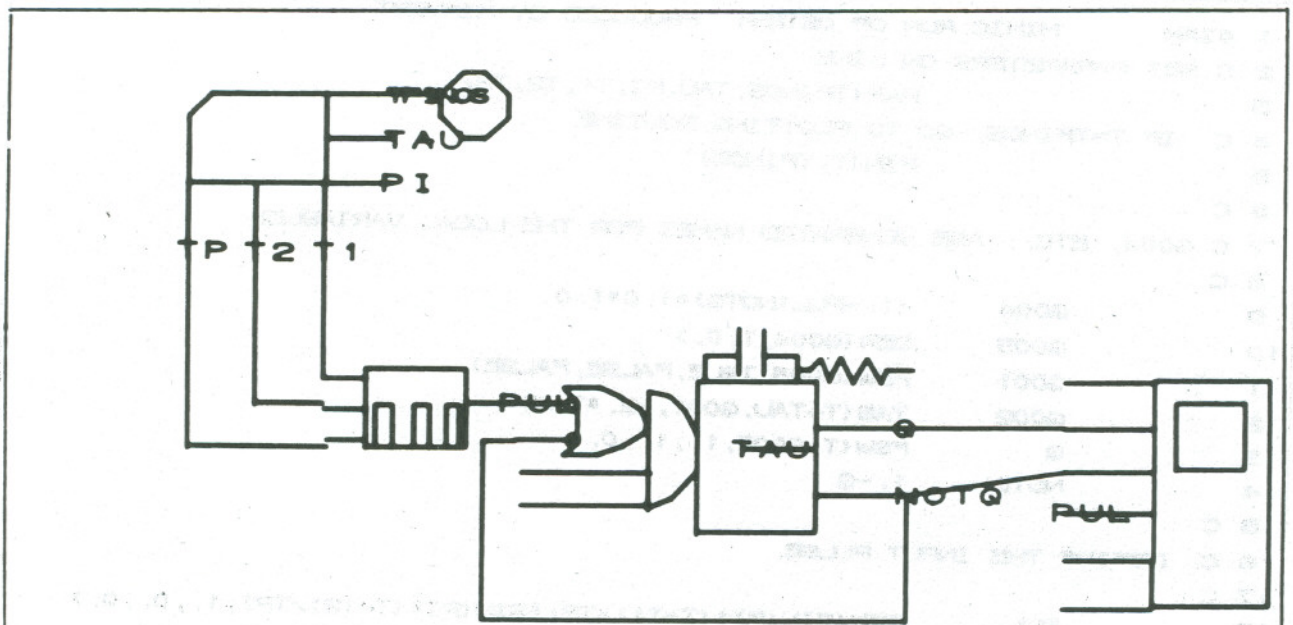
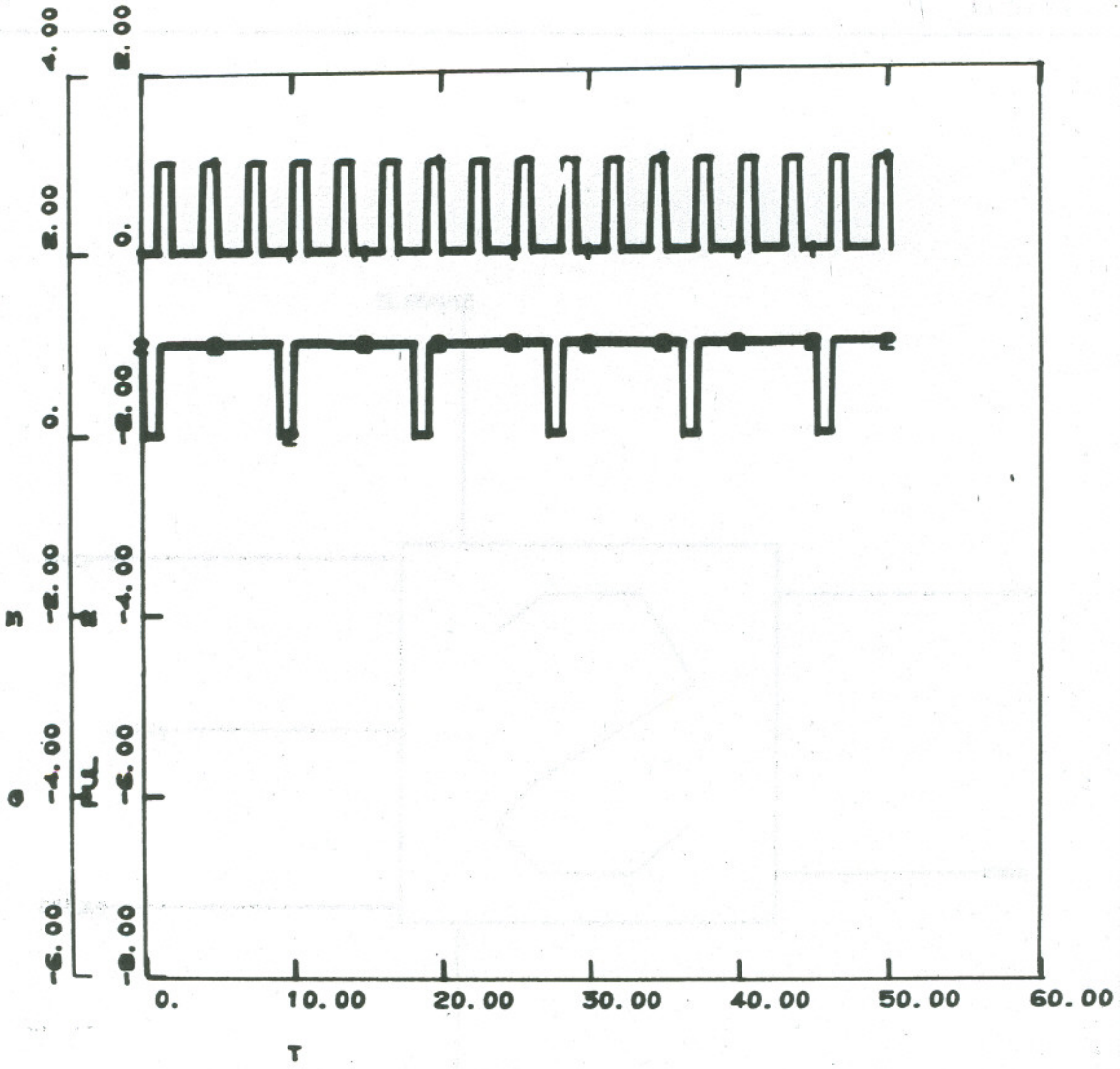


Fig. 19a. A model to test an element names ONESHOT. OSTEST is a ONESHOT wired as a frequency selector. This problem was suggested by Richard La Pierre.

OSTEST

```
 1 *IAM       MIMIC RUN OF OSTEST  PRODUCED BY MIMVERT
 2 C SET PARAMETERS ON LINE
 3                     PAR(TFINOS, TAU, PI, T1, T2, TP)
 4 C  IF T=TFINOS, GO TO PLOTTING ROUTINE
 5                     FIN(T, TFINOS)
 6 C
 7 C G004, ETC., ARE GENERATED NAMES FOR THE LOCAL VARIABLES
 8 C
 9           G004      (1.-PUL*NOTQ)*1.0*1.0
10           G003      DER(G004, T, 0.)
11           G001      FSW(G003, TRUE, FALSE, FALSE)
12           G002      TAS(T+TAU, G001, -2.*TAU)
13           Q         FSW(T-G002, 1., 1., 0.)
14           NOTQ      1.-Q
15 C
16 C  DEFINE THE INPUT PULSE
17 C
18           PUL       FSW(SIN(PI*(T-T1)/TP)*SIN(PI*(T-T2)/TP), 1., 0., 0.)
19 C
20 C   SPECIFY THE FUNCTIONS TO BE PLOTTED
21 C
22                     PLO(T, Q, NOTQ, PUL)
23                     END
24
```

Fig. 19b. The MIMIC program produced by the analysis of OSTEST (comments added).



ONESHOT WITH TAU=8.. FREQUENCY SELECTOR

12/20/71. 16.45.44.

Fig. 19c. MIMIC output curves for one set of input parameters.

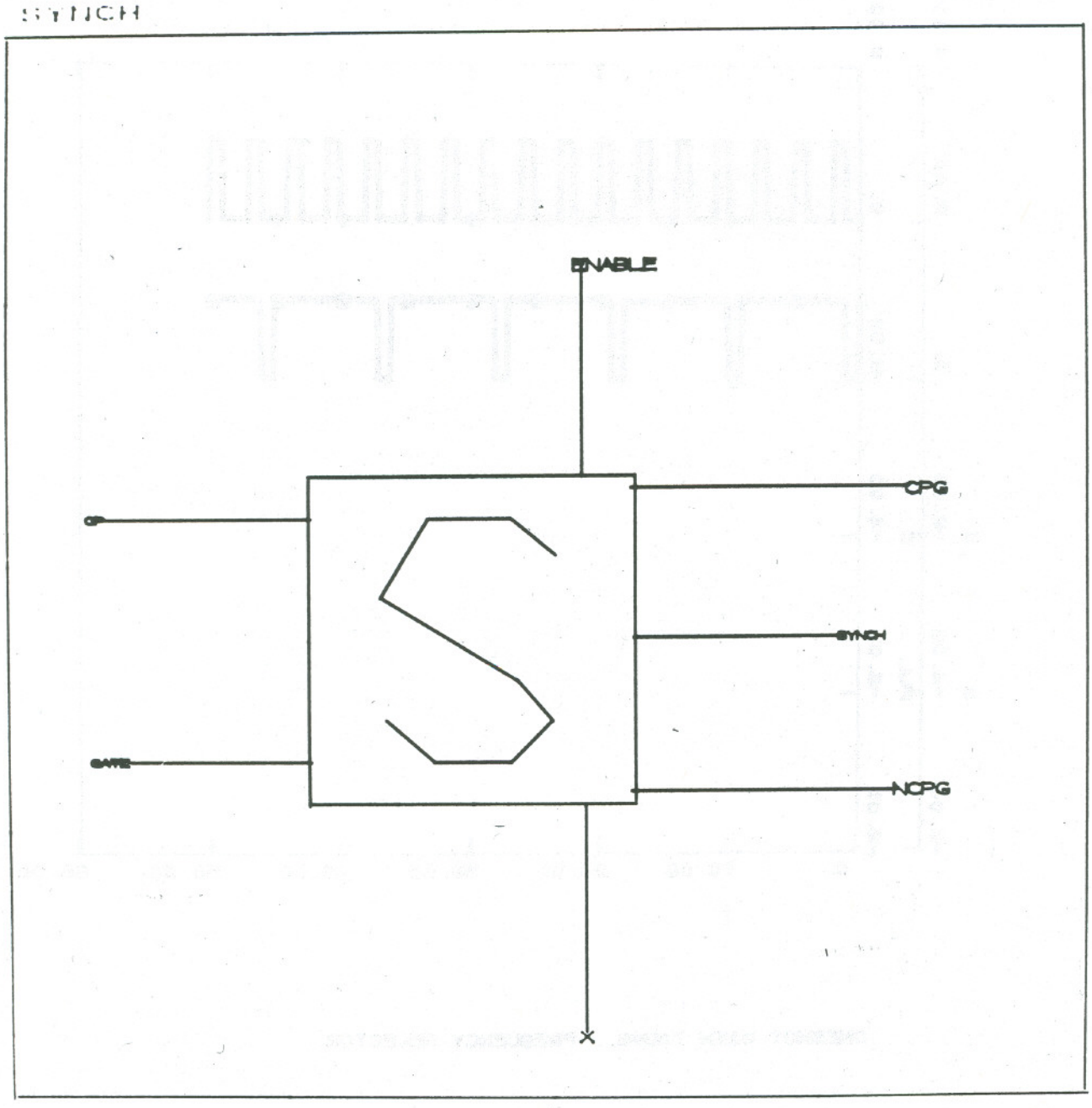


Fig. 20a. The symbol SYNCH for a next pulse synchronizer. Some "debugging" outputs, labeled ENABLE, CPG, NCPG and X, have been added to allow for viewing intermediate signals.

SYNCH

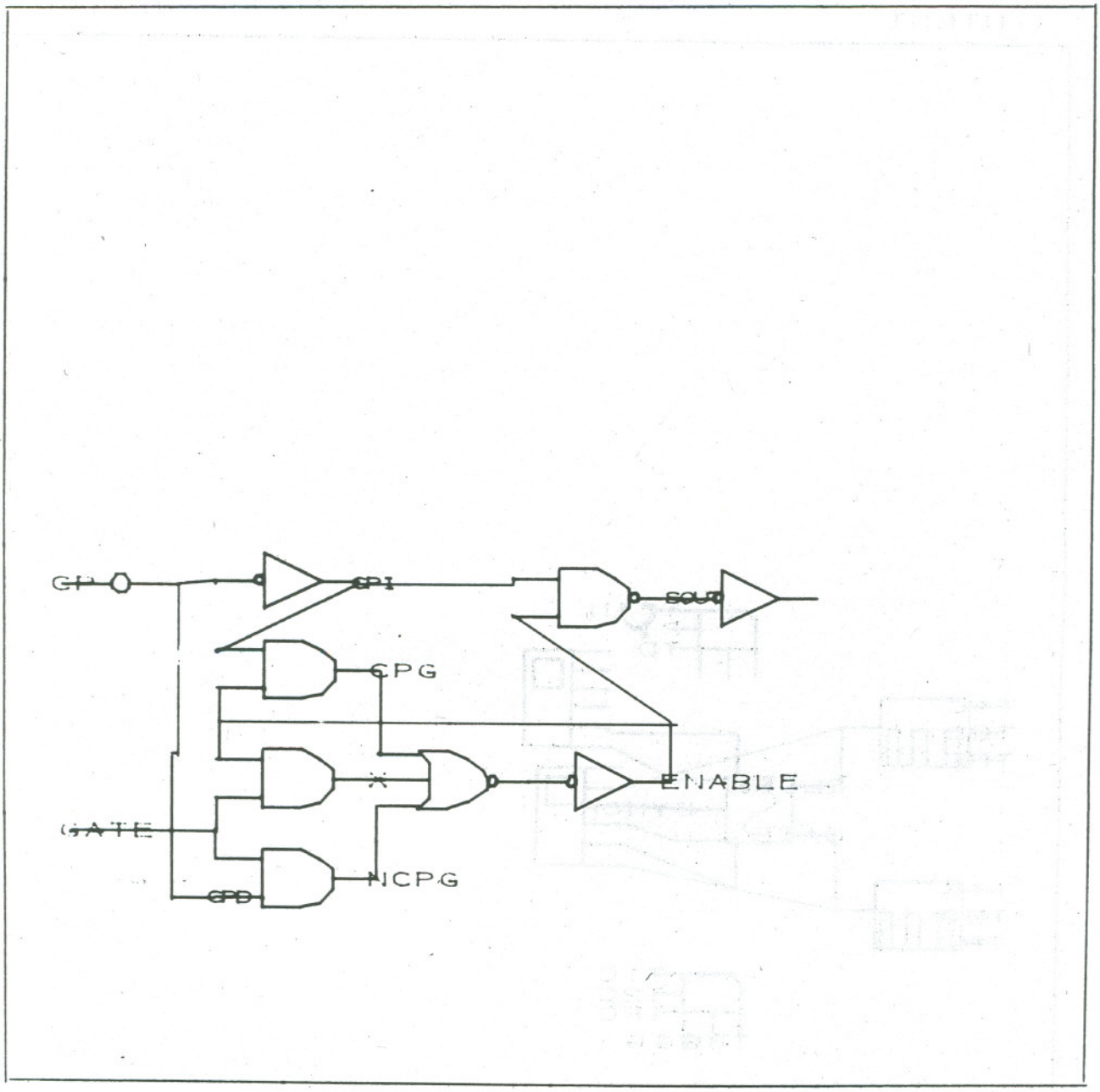


Fig. 20b. The macro definition of SYNCH in terms of gates defined by MIMIC time-delay functions. This element was designed by Don Evans to illustrate problems in asynchronous logic design.

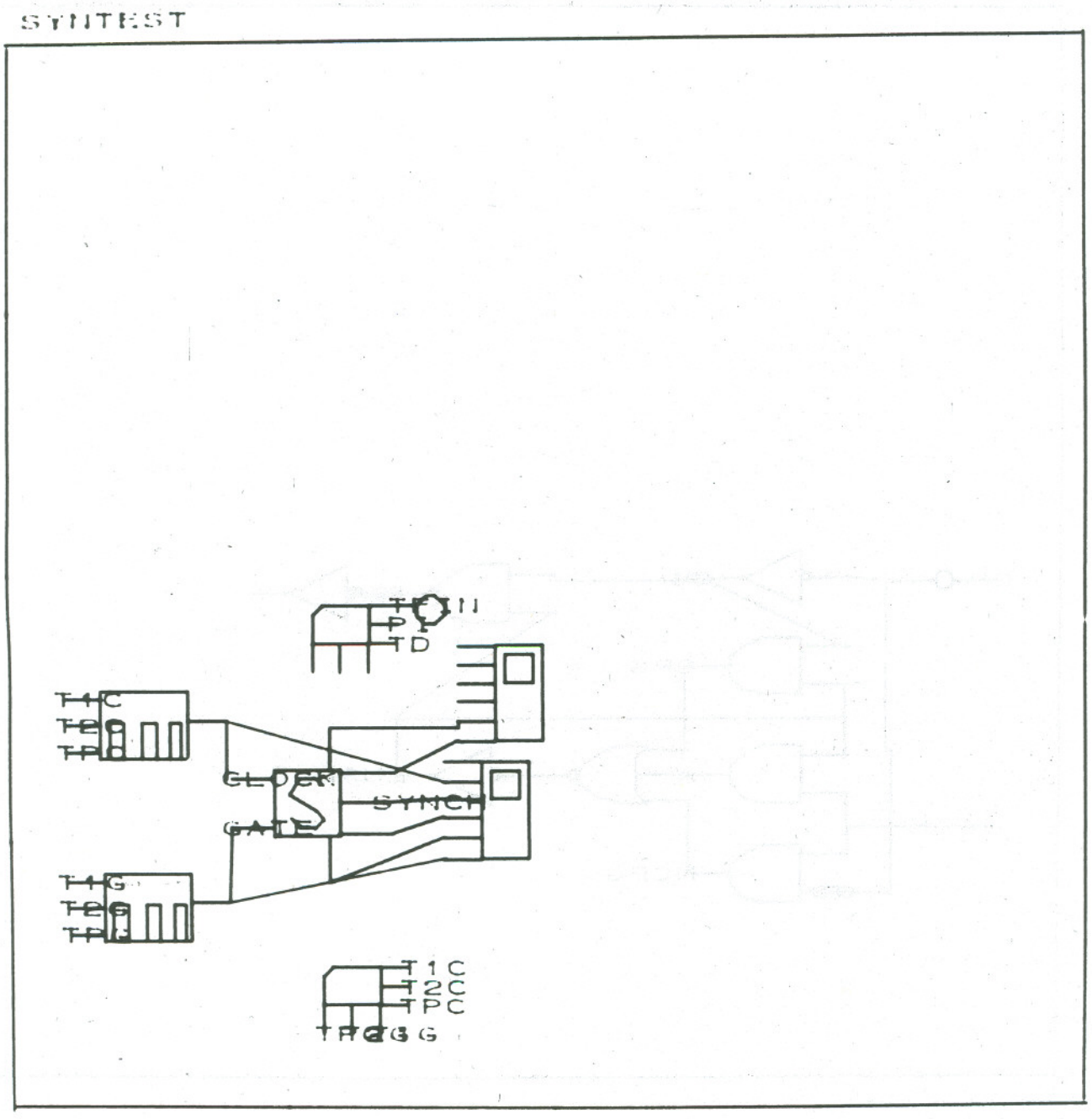


Fig. 20c. A model designed to test the SYNCH. The input pulse parameters and gate delays can be set on-line in the interactive MIMIC program.

SYNTEST

```
 1 *IAM      MIMIC RUN OF SYNTEST PRODUCED BY MIMVERT
 2 C
 3 C  BEGIN MIMIC MACRO DEFINITION OF SYNCH
 4           SYNCH    EMA(CPG   ,CP    ,ENABLE,GATE  ,NCPG  ,SYNCH )
 5                    EML(X     )
 6           CPD      FIX(SUB(1.,CP)+.5)
 7           CPI      FIX(TDL(1.-CPD,TD,20.)+.5)
 8           CPG      FIX(TDL(CPI*ENABLE,TD,20.)+.5)
 9           X        FIX(TDL(ENABLE*GATE,TD,20.)+.5)
10           NCPG     FIX(TDL(GATE*CPD,TD,20.)+.5)
11           G001     FIX(TDL(1.-MAX(CPG,X,NCPG),TD,20.)+.5)
12           CPI      FIX(TDL(1.-SOUT*ENABLE,TD,20.)+.5)
13           ENABLE   FIX(TDL(1.-G001,TD,20.)+.5)
14           SYNCH    FIX(TDL(1.-SOUT,TD,20.)+.5)
15                    EMA
16 C  END OF MACRO DEFINITION
17 C
18 C BEGIN PROGRAM
19 C
20           CLOCK    FSW(SIN(PI*(T-T1C)/TPC)*SIN(PI*(T-T2C)/TPC),1.,0.,0.)
21           GATE     FSW(SIN(PI*(T-T1G)/TPG)*SIN(PI*(T-T2G)/TPG),1.,0.,0.)
22 C
23 C  CALL SYNCH MACRO
24           SYNCH    CMA(CPG   ,CLOCK ,ENABLE,GATE  ,NCPG  ,SYNCH )
25                    CML(X     )
26 C END OF SYNCH MACRO CALL
27 C
28                    PAR(TFIN,PI,TD)
29                    FIN(T,TFIN)
30                    PAR(T1C,T2C,TPC,T1G,T2G,TPG)
31                    PLO(T,CLOCK,SYNCH,NCPG,X,GATE)
32                    PLO(T,ENABLE,CPG)
33                    END
34
```

Fig. 20d. The MIMIC program produced for SYNTEST.

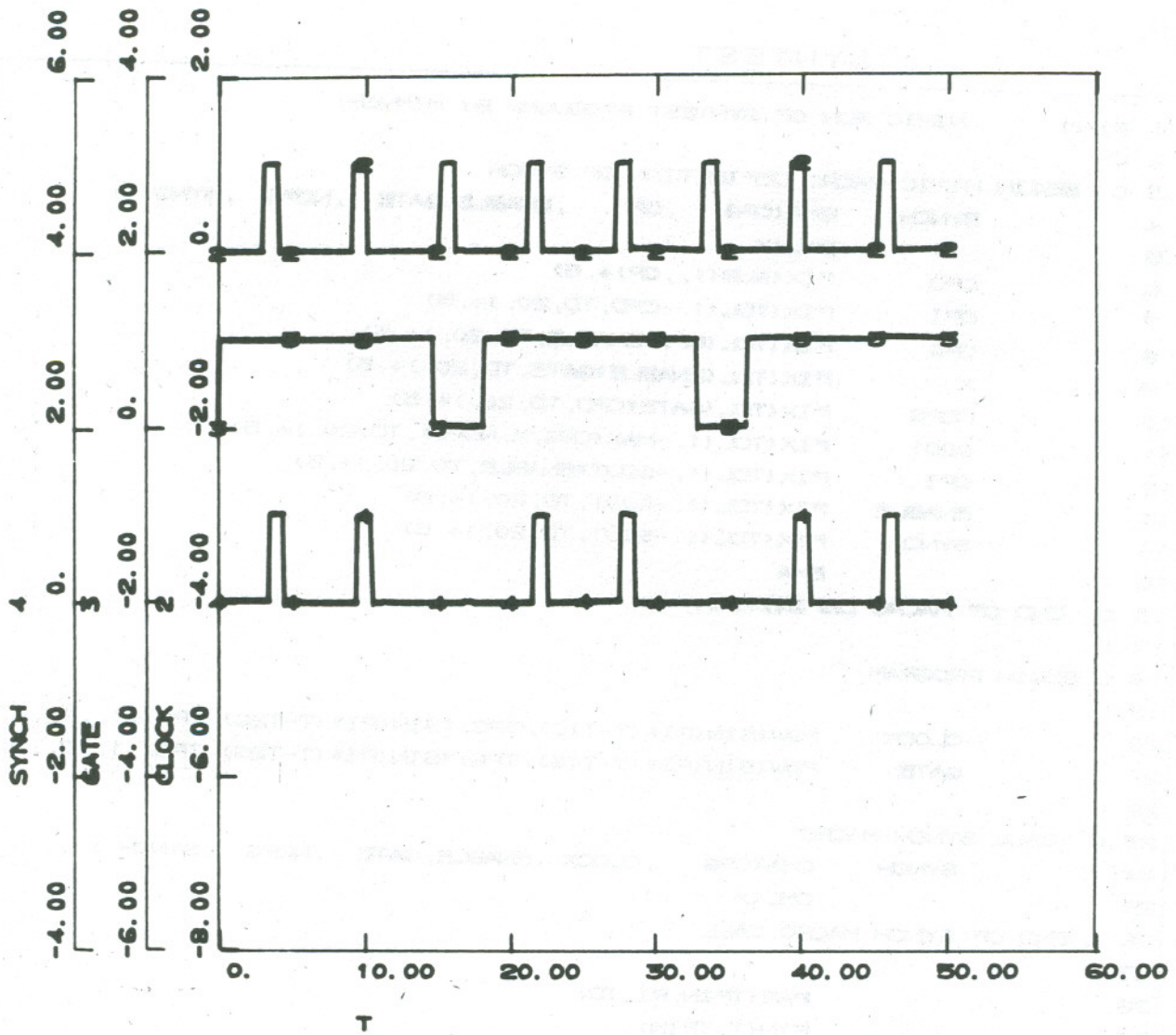


Fig. 20e. MIMIC output curves for SYNTEST. The GATE and CLOCK pulses have been specified such that no timing problems occur.

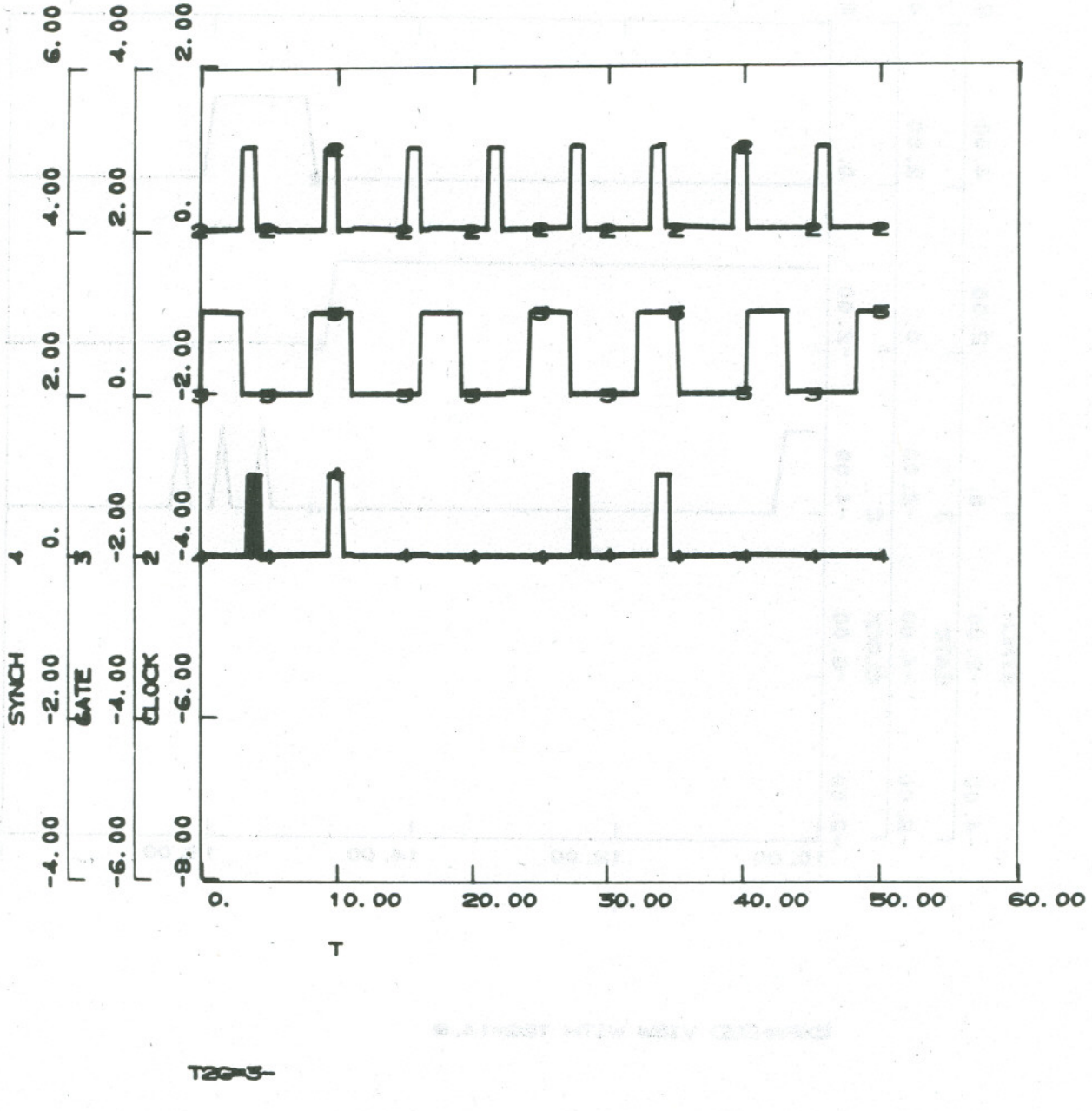


Fig. 20f. MIMIC output curves for SYNTEST. The GATE and CLOCK pulses have been specified such that a timing glich occurs.

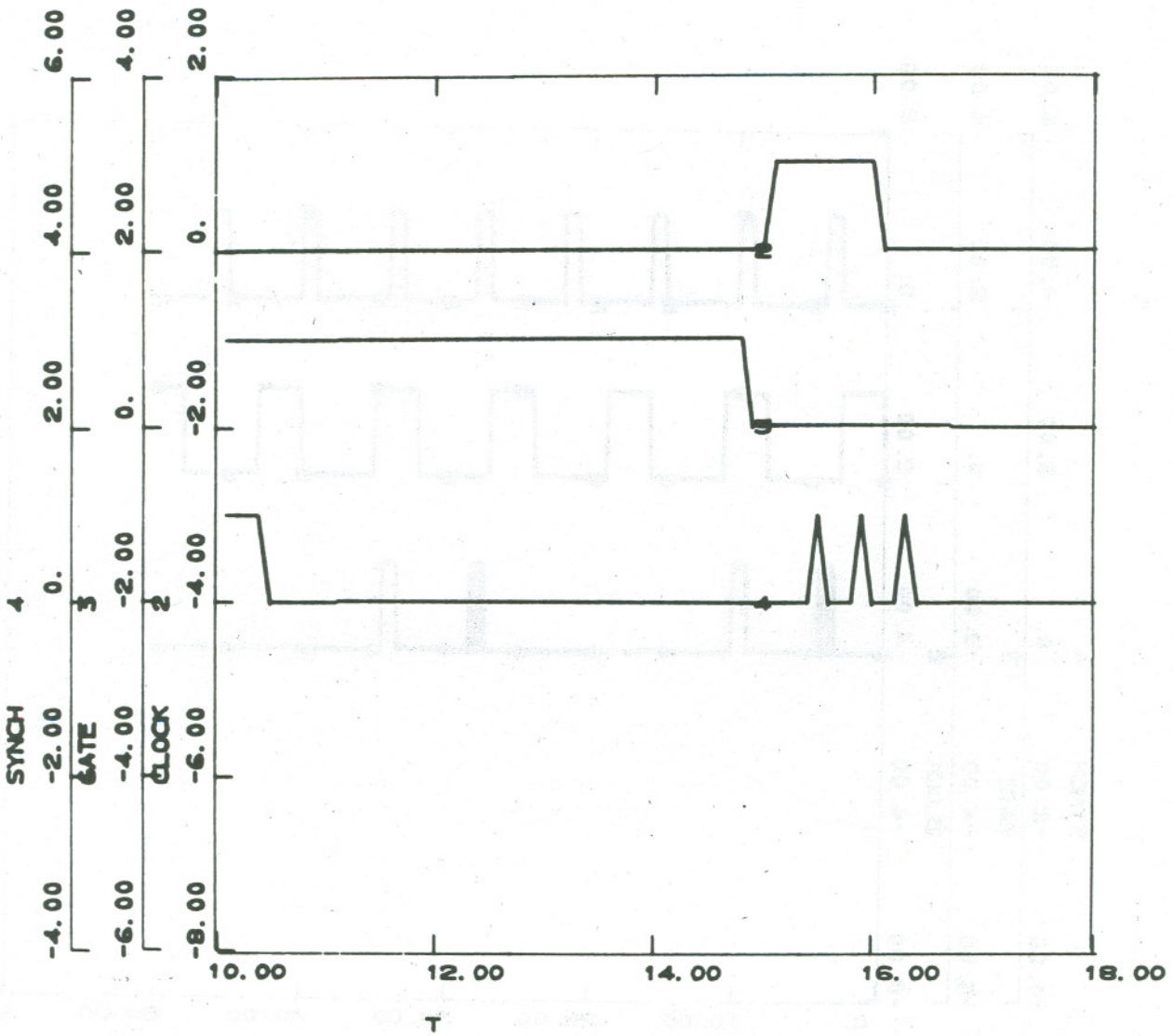


Fig. 20g. MIMIC output curves for SYNTEST. This is an expanded view of the glich in the last figure.

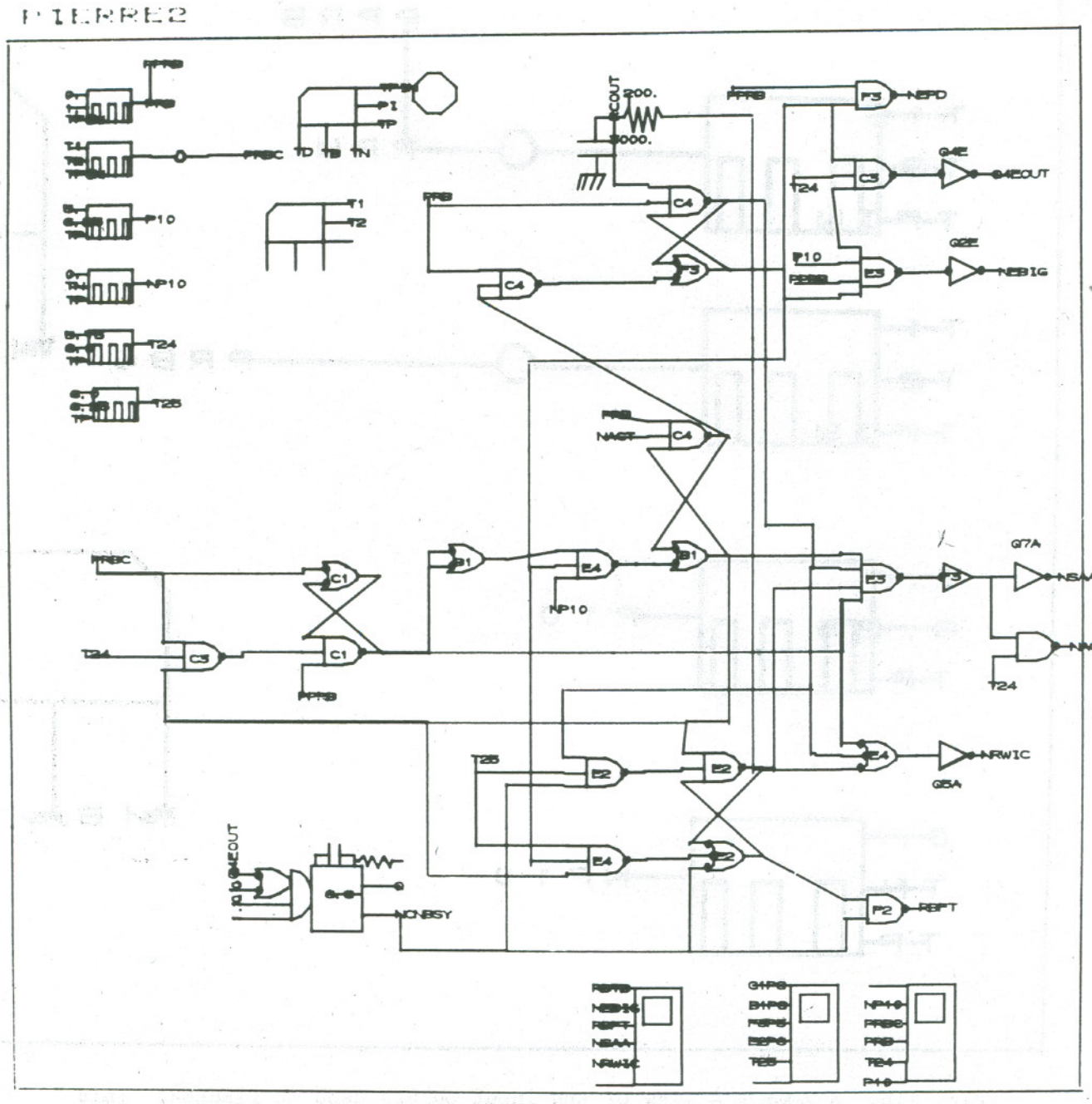


Fig. 21a. The model PIERRE2 for a logic problem, designed by Richard La Pierre.

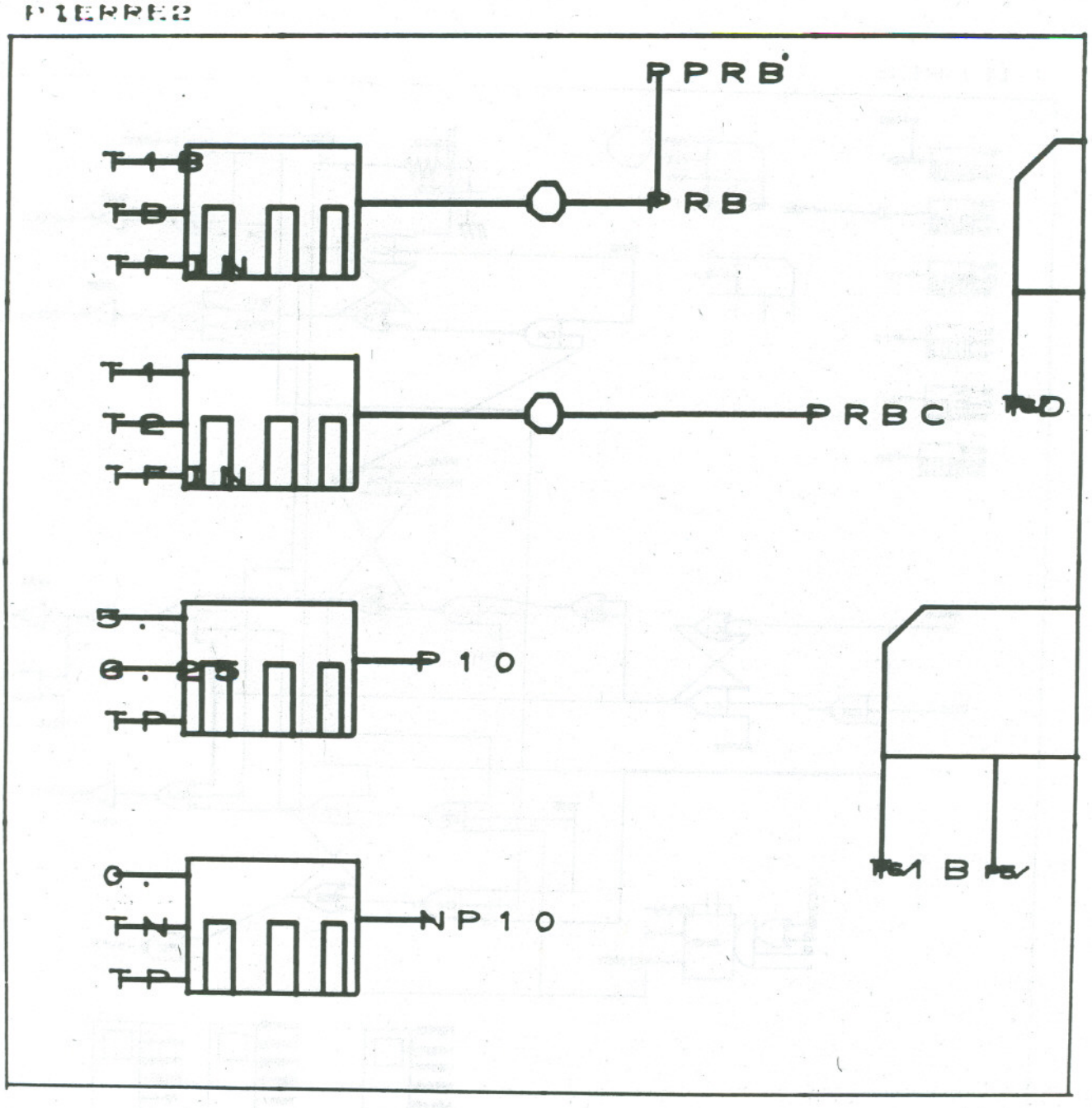


Fig. 21b. A ZOOM = 4 view of the input pulses used in PIERRE2. This illustrates the use of identical labels (instead of lines) to make topological connections.

PIERRE2

```
 1 $IAM       MIMIC RUN OF PIERRE2 PRODUCED BY MIMVERT
 2 C   INPUT PULSES ARE SPECIFIED
 3             PRB       PPRB
 4             P10       FSW(SIN(PI*(T-5.)/TP)*SIN(PI*(T-6.25)/TP),1.,0.,0.)
 5             NP10      FSW(SIN(PI*(T-0.)/TP)*SIN(PI*(T-TN)/TP),1.,0.,0.)
 6             T24       FSW(SIN(PI*(T-5.75)/TP)*SIN(PI*(T-6.0)/TP),1.,0.,0.)
 7             G001      FSW(SIN(PI*(T-T1)/TFIN)*SIN(PI*(T-T2)/TFIN),1.,0.,0.)
 8             PPRB      FSW(SIN(PI*(T-0.)/TFIN)*SIN(PI*(T-1.)/TFIN),1.,0.,0.)
 9             T25       FSW(SIN(PI*(T-6.0)/TP)*SIN(PI*(T-6.25)/TP),1.,0.,0.)
10 C
11             C3P8      FIX(TDL(1.-PRBC*T24*B1P8,TD,20.)+.5)
12             PRBC      FIX(SUB(1.,G001)+.5)
13 C   THE ONESHOT IS DEFINED
14             G005      (1.-Q4EOUT*1.0)*1.0*1.0
15             G004      DER(G005,T,0.)
16             G002      FSW(G004,TRUE,FALSE,FALSE)
17             G003      TAS(T+6.8,G002,-2.*6.8)
18             Q         FSW(T-G003,1.,1.,0.)
19             NCNBSY    1.-Q
20 C
21             C1P8      FIX(TDL(1.-C1P6*C3P8*PPRB,TD,20.)+.5)
22             C1P6      FIX(TDL(1.-PRBC*C1P8,TD,20.)+.5)
23 C PARAMETERS TO BE SET ON LINE ARE SPECIFIED
24                       PAR(T1,T2)
25                       PAR(TFIN,PI,TP,TN,TB,TD)
26 C
27                       FIN(T,TFIN)
28             B1P6      FIX(TDL(1.-C1P8*C1P8,TD,20.)+.5)
29             C4P8      FIX(TDL(1.-PRB*C4P12*C4P12,TD,20.)+.5)
30             E4P8      FIX(TDL(1.-B1P6*F3P3*NP10,TD,20.)+.5)
31             E2P12     FIX(TDL(1.-C4P6*T25*NCNBSY,TD,20.)+.5)
32             E4P12     FIX(TDL(1.-T25*F3P3*B1P8,TD,20.)+.5)
33                       PLO(RBTB,NBBIG,RBFT,NSAA,NRWIC)
34             C4P12     FIX(TDL(1.-PRB*NACT*B1P8,TD,20.)+.5)
35             C4P6      FIX(TDL(1.-RCOUT*PRB*F3P3,TD,20.)+.5)
```

Fig. 21c. The first page of the MIMIC program produced by analyzing PIERRE2.

PIERRE2

```
36          B1P8      FIX(TDL(1.-C4P12*E4P8,TD,20.)+.5)
37          F3P3      FIX(TDL(1.-C4P6*C4P8,TD,20.)+.5)
38          RCOUT     FIX(FTR(E2P8,200.*3000./1000./1000.)+0.5)
39          E2P8      FIX(TDL(1.-B1P8*E2P12*E2P6,TD,20.)+.5)
40          E2P6      FIX(TDL(1.-E2P8*E4P12*NCNBSY,TD,20.)+.5)
41                    PLO(C1P8,B1P8,F3P3,E2P6,T25)
42          C3P6      FIX(TDL(1.-F3P3*T24*B1P8,TD,20.)+.5)
43          PPRB      FIX(TDL(1.-NEPD*F3P3,TD,20.)+.5)
44          E3P6      FIX(TDL(1.-B1P8*C4P6*E2P8*C1P8,TD,20.)+.5)
45          E4P6      FIX(TDL(1.-C1P8*C4P6*E2P8,TD,20.)+.5)
46          E3P8      FIX(TDL(1.-B1P8*P10*PPRB*F3P3,TD,20.)+.5)
47          E2P6      FIX(TDL(1.-RBFT*NCNBSY,TD,20.)+.5)
48                    PLO(T,NP10,PRBC,PRB,T24,P10)
49          Q4BOUT    FIX(TDL(C3P6,TD,20.)+.5)
50          NRWIC     FIX(TDL(1.-E4P6,TD,20.)+.5)
51          F3P11     FIX(TDL(1.-E3P6,TD,20.)+.5)
52          NEBIG     FIX(TDL(E3P8,TD,20.)+.5)
53          NSAA      FIX(TDL(1.-F3P11,TD,20.)+.5)
54          F3P11     FIX(TDL(1.-NACT*T24,TD,20.)+.5)
55                    END
56
```

Fig. 21d. The second page of the MIMIC program for PIERRE2.

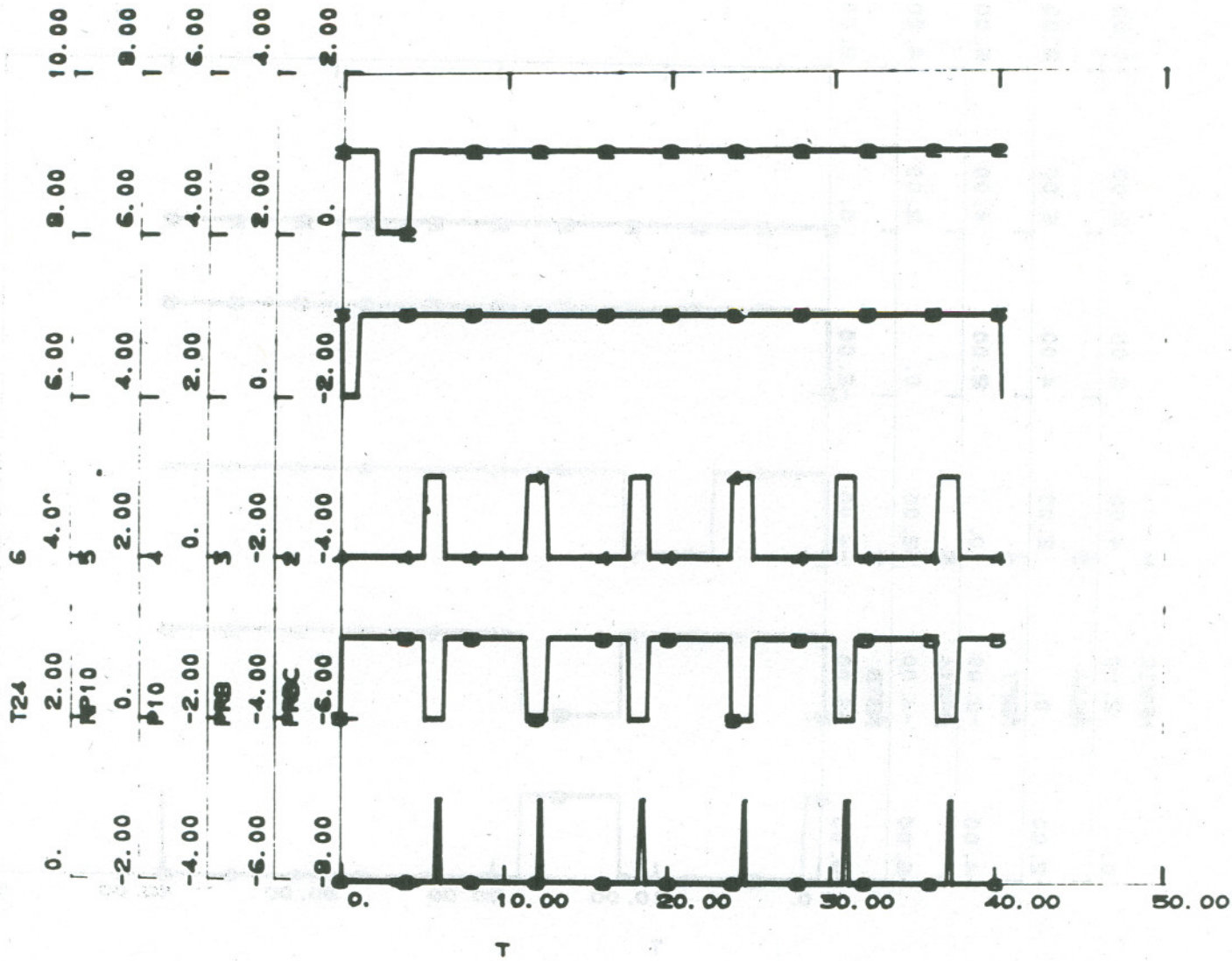


Fig. 2le. MIMIC output curves showing some of the input pulses for PIERRE2.

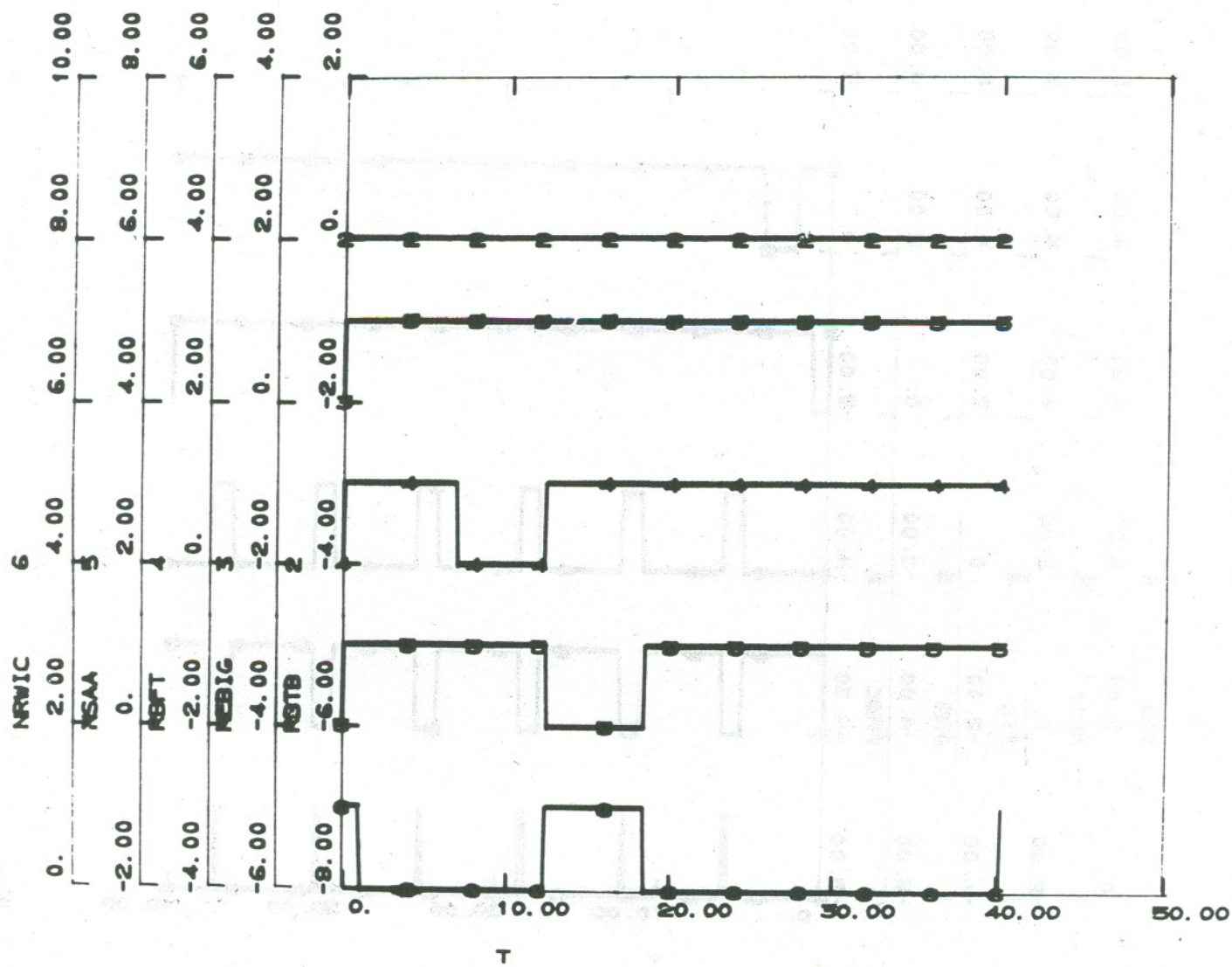
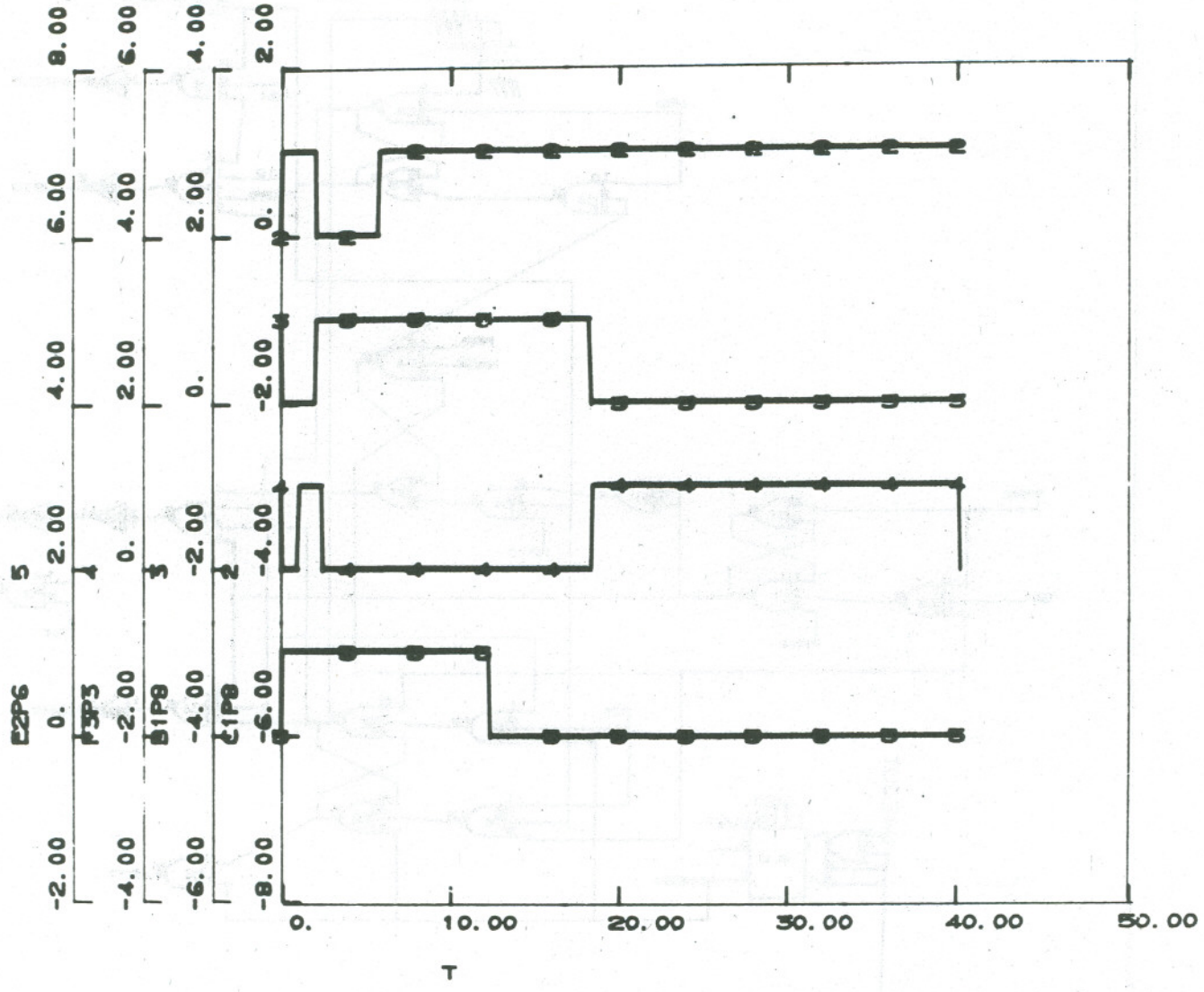


Fig. 21f. MIMIC output curves for PIERRE2.



SOME INTERMEDIATE SIGNALS FOR PIERRE2

01/16/72. 19.42.30.

Fig. 21g. MIMIC output curves for PIERRE2.

PIERRE2

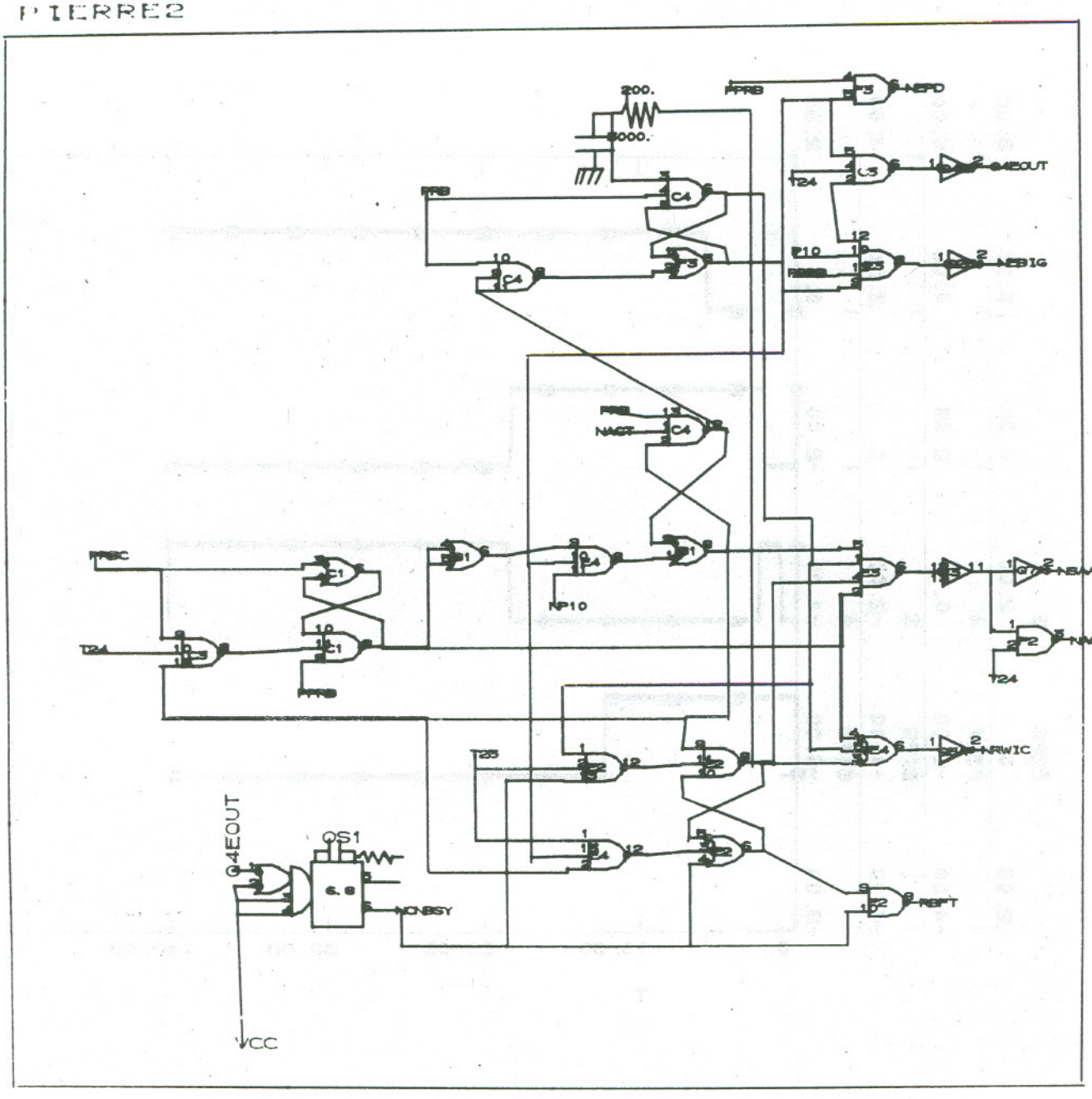


Fig. 21h. The model PIERRE2 altered to produce a connect-list output (an alternate set of definitions for the elements has been specified replacing the MIMIC functions used in the preceding figures).

PIERRE2

```
 1 CONNECT LIST FOR PIERRE2 PRODUCED BY PICASSO
 2 PRBC, C3-09
 3 T24, C3-010
 4 G001, C3-013
 5 G002, C3-08
 6 Q4EOUT, OS1-01
 7 VCC, OS1-02
 8 VCC, OS1-03
 9 VCC, OS1-04
10 G025, OS1-05
11 NCNBSY, OS1-06
12 G003, C1-010
13 G002, C1-011
14 PPRB, C1-09
15 G004, C1-08
16 PRBC, C1-03
17 G004, C1-04
18 G003, C1-06
19 G004, B1-04
20 G004, B1-05
21 G006, B1-06
22 PRB, C4-010
23 G005, C4-09
24 G005, C4-011
25 G008, C4-08
26 G006, E4-09
27 G007, E4-010
28 NP10, E4-011
29 G011, E4-08
30 G009, E2-01
31 T25, E2-02
32 NCNBSY, E2-013
33 G012, E2-012
34 T25, E4-01
35 G007, E4-013
```

Fig. 2li. The first page of the connect list output produced by analyzing PIERRE2. The format is signal name, element name-pin number.

PIERRE2

```
36 G001, E4-02
37 G013, E4-012
38 PRB, C4-013
39 NACT, C4-01
40 G001, C4-02
41 G005, C4-012
42 G010, C4-03
43 PRB, C4-04
44 G007, C4-05
45 G009, C4-06
46 G005, B1-09
47 G011, B1-010
48 G001, B1-08
49 G009, F3-01
50 G008, F3-02
51 G007, F3-03
52 G010, 3000.PF
53 G014, 200.OHM
54 G001, E2-09
55 G012, E2-011
56 G015, E2-010
57 G014, E2-08
58 G014, E2-05
59 G013, E2-03
60 NCNBSY, E2-04
61 G015, E2-06
62 G007, C3-05
63 T24, C3-04
64 G016, C3-02
65 G017, C3-06
66 PPRB, F3-04
67 G007, F3-05
68 NEPD, F3-06
69 G001, E3-05
70 G009, E3-01
```

Fig. 21j. The second page of the connect list for PIERRE2.

PIERRE2

```
71 G014, E3-04
72 G004, E3-02
73 G018, E3-06
74 G004, E4-04
75 G009, E4-05
76 G014, E4-03
77 G020, E4-06
78 G016, E3-012
79 P10, E3-010
80 PPRB, E3-013
81 G007, E3-09
82 G021, E3-08
83 G015, F2-09
84 NCNBSY, F2-010
85 RBFT, F2-08
86 G017, Q4E-01
87 Q4EOUT, Q4E-02
88 G020, Q5A-01
89 NRWIC, Q5A-02
90 G018, F3-012
91 G022, F3-011
92 G021, Q2E-02
93 NEBIG, Q2E-02
94 G022, Q7A-01
95 NSAA, Q7A-02
96 G022, F2-01
97 T24, F2-02
98 NACT, F2-03
99                    END
00
```

Fig. 21k. The last page of the connect list for PIERRE2.

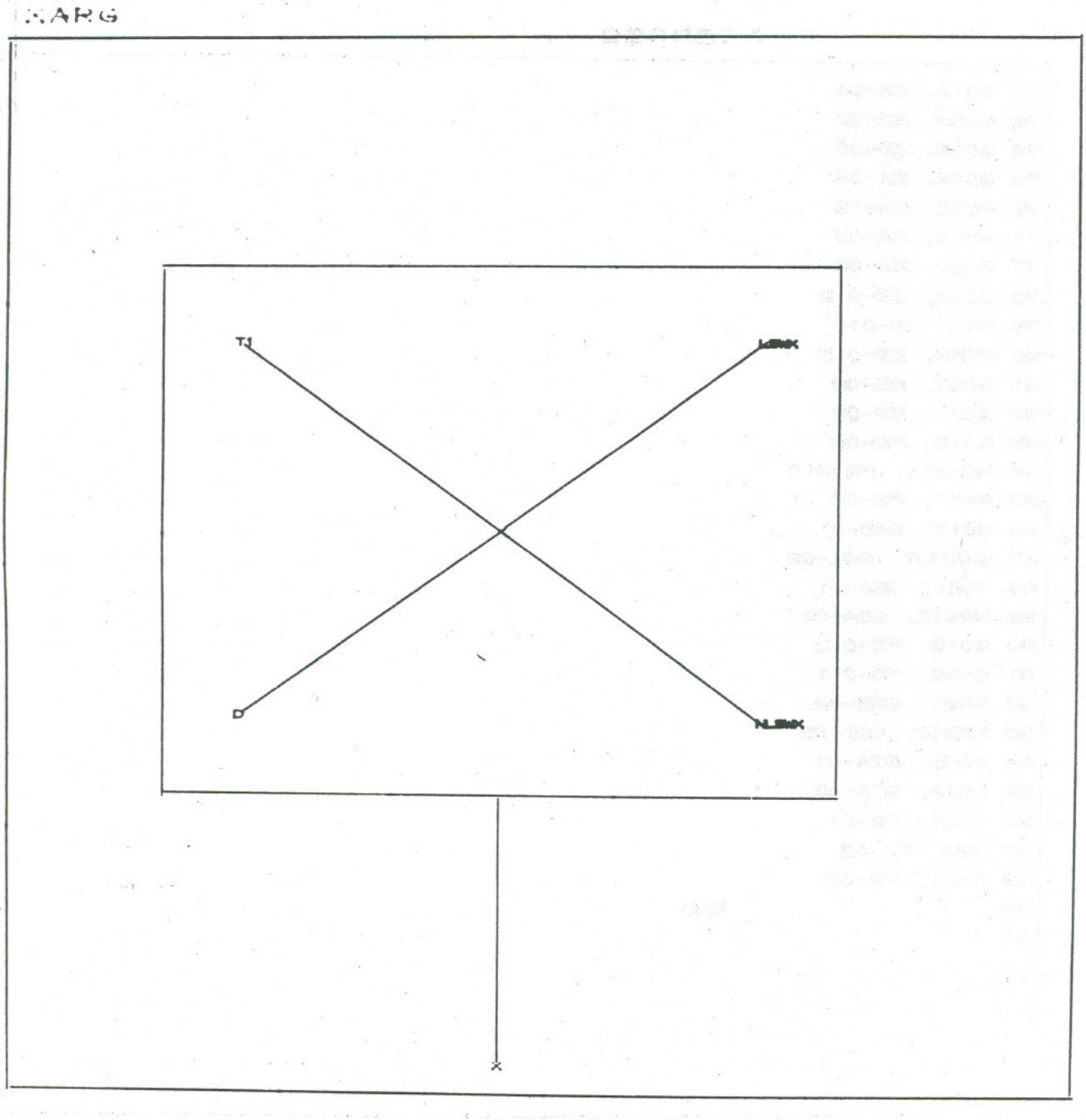


Fig. 22a. A symbol XARG with two inputs, labeled T1 and D, some local labels LSWX and NLSWX and an output X.

XARG

```
1                LSWX      FSW(D,FALSE,FALSE,TRUE)
2                NLSWX     NOT(LSWX)
3    NLSWX       X         0.0
4    LSWX        X         3.14159*EXP(.693*(T1-T)/D)
5
```

Fig. 22b. The text definition of XARG in terms of MIMIC functions and logical variables.

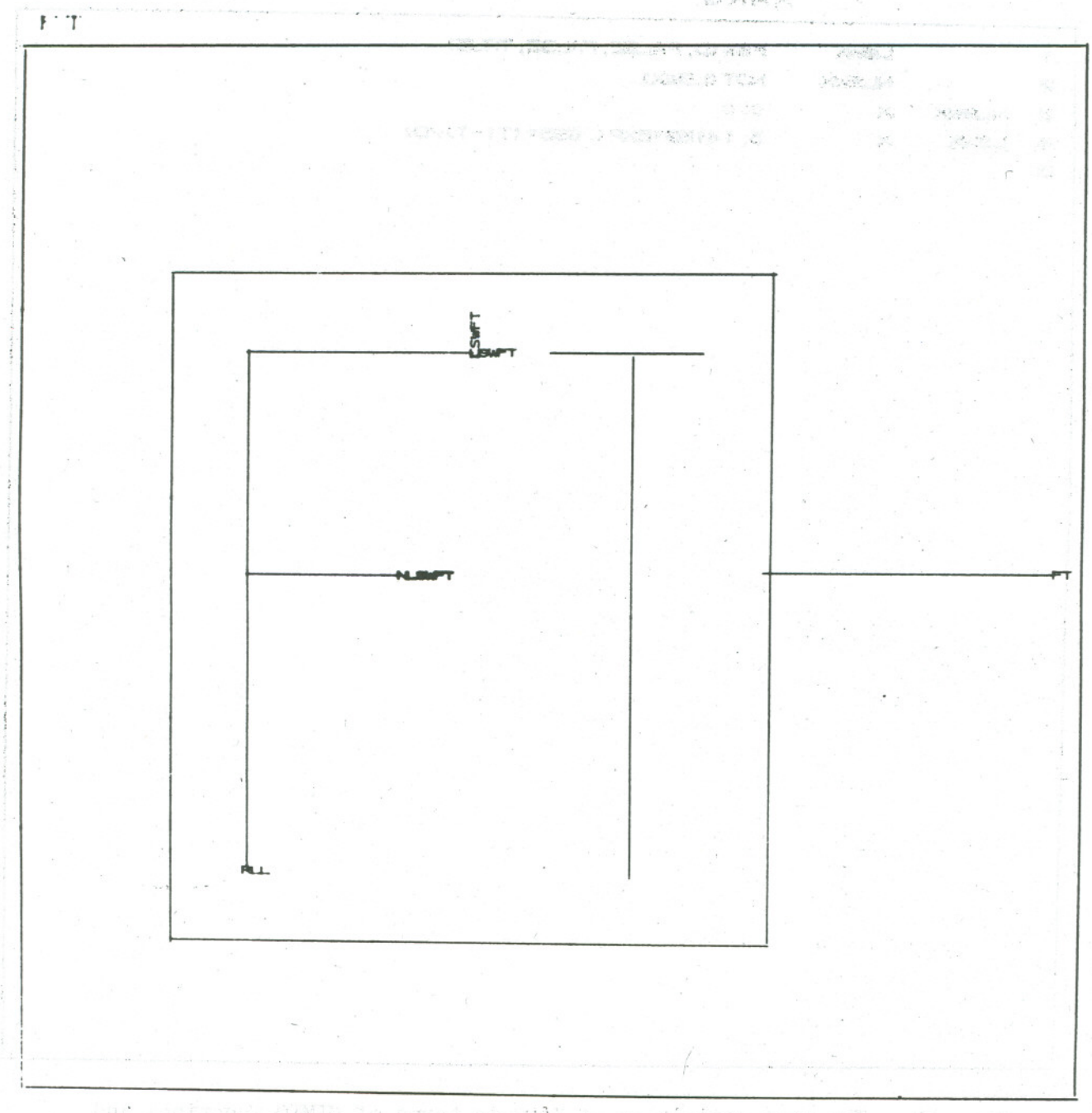


Fig. 22c. A symbol FT.

FT

```
1                LSWFT     FSW(RLL, TRUE, FALSE, TRUE)
2                NLSWFT    NOT(LSWFT)
3    LSWFT       FT        (SQR(T1*(T1-D)*(1.+RLL/.9))+.385*D)*SQR(1.+RLL/.9)
4    NLSWFT      FT        0.0
5
```

Fig. 22d. The text definition of FT in terms of MIMIC functions SQR (square root) and logical variables.

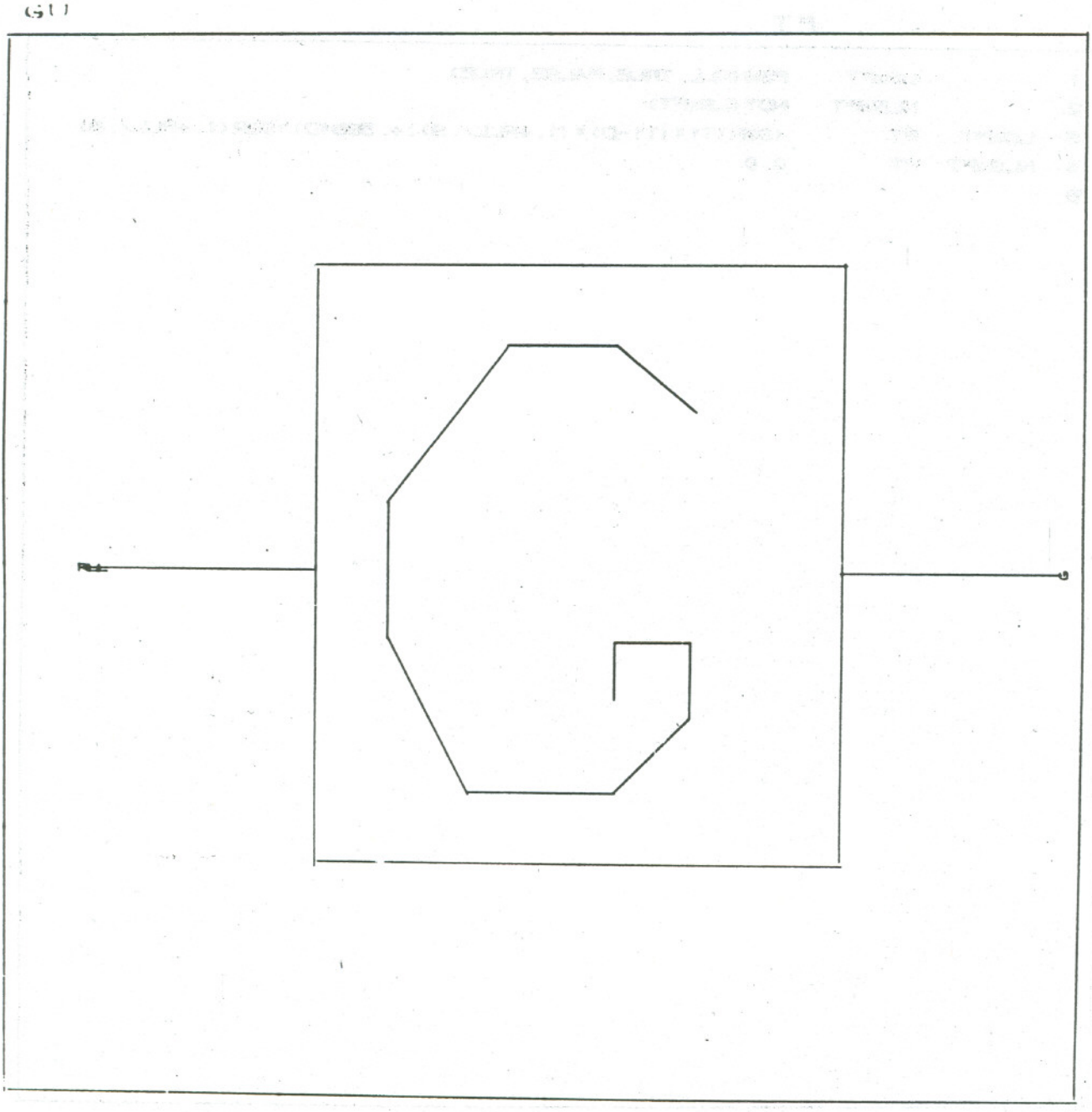


Fig. 22e. The symbol GU.

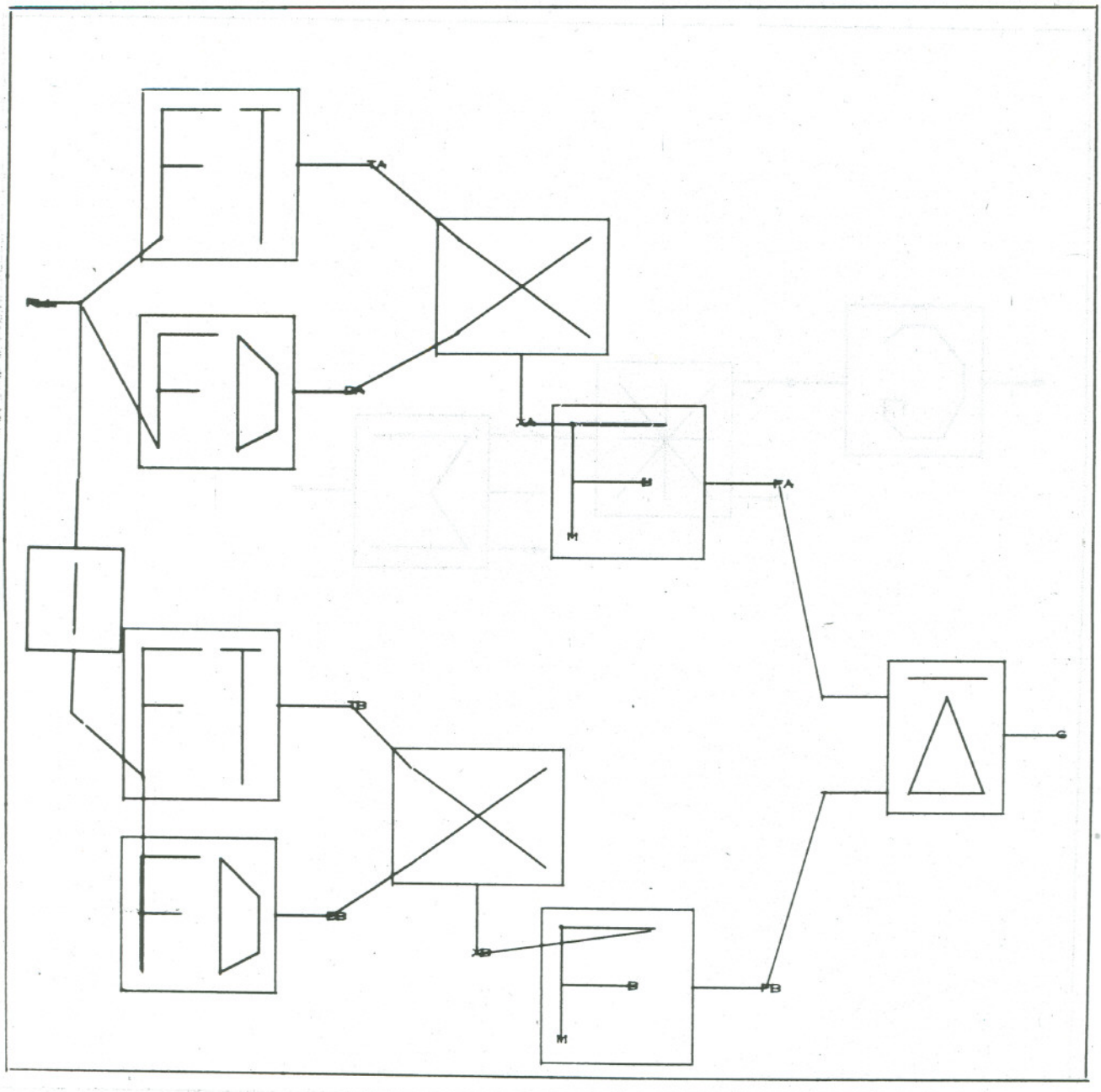


Fig. 22f. The macro definition of GU in terms of primitive elements defined by MIMIC functions. GU is a macro definition nested to level 1.

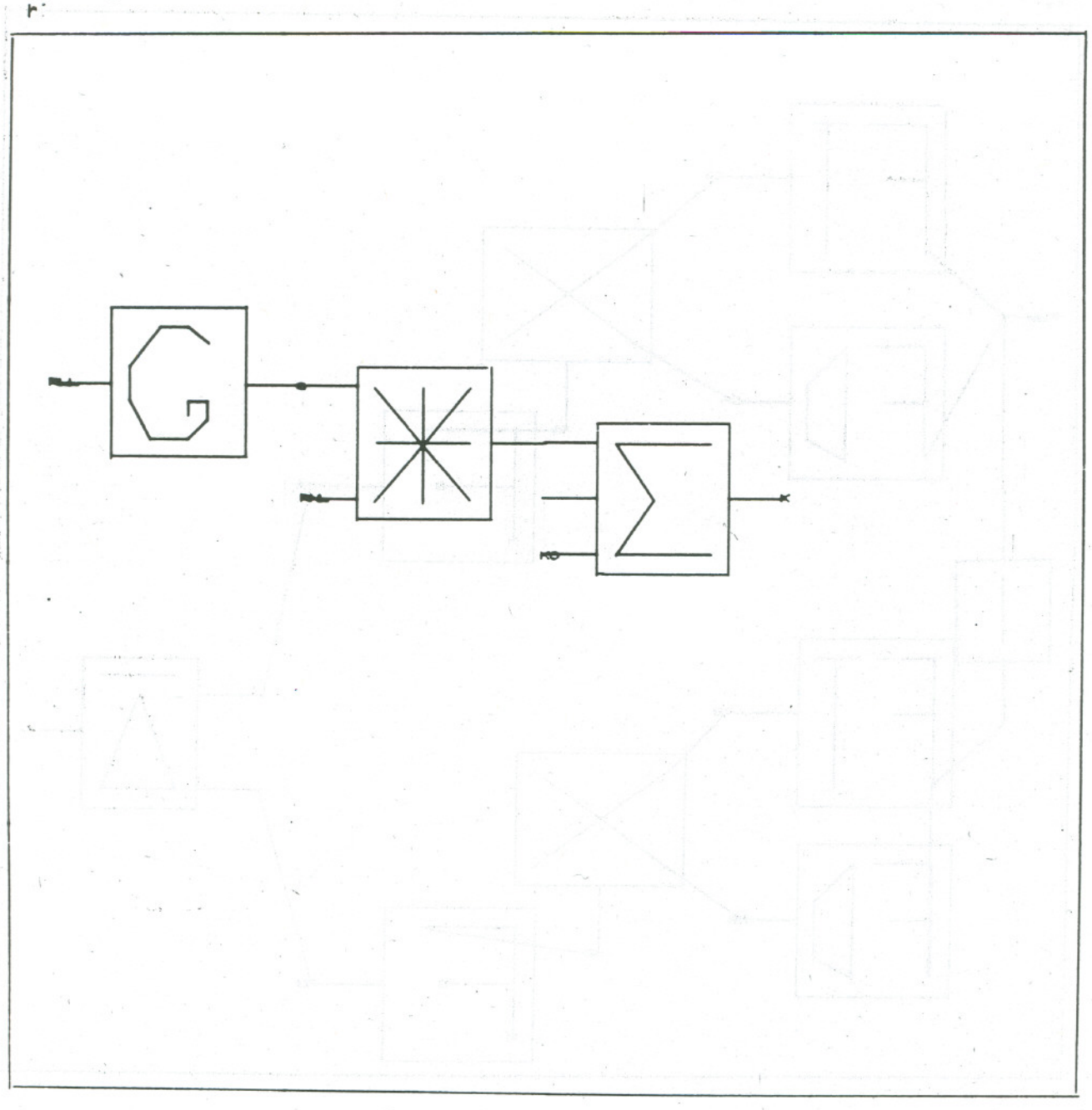


Fig. 22g. The macro definition of the element K in terms of another macro defined element GU and some primitive elements. This illustrates the nesting of macro definitions to level 2.

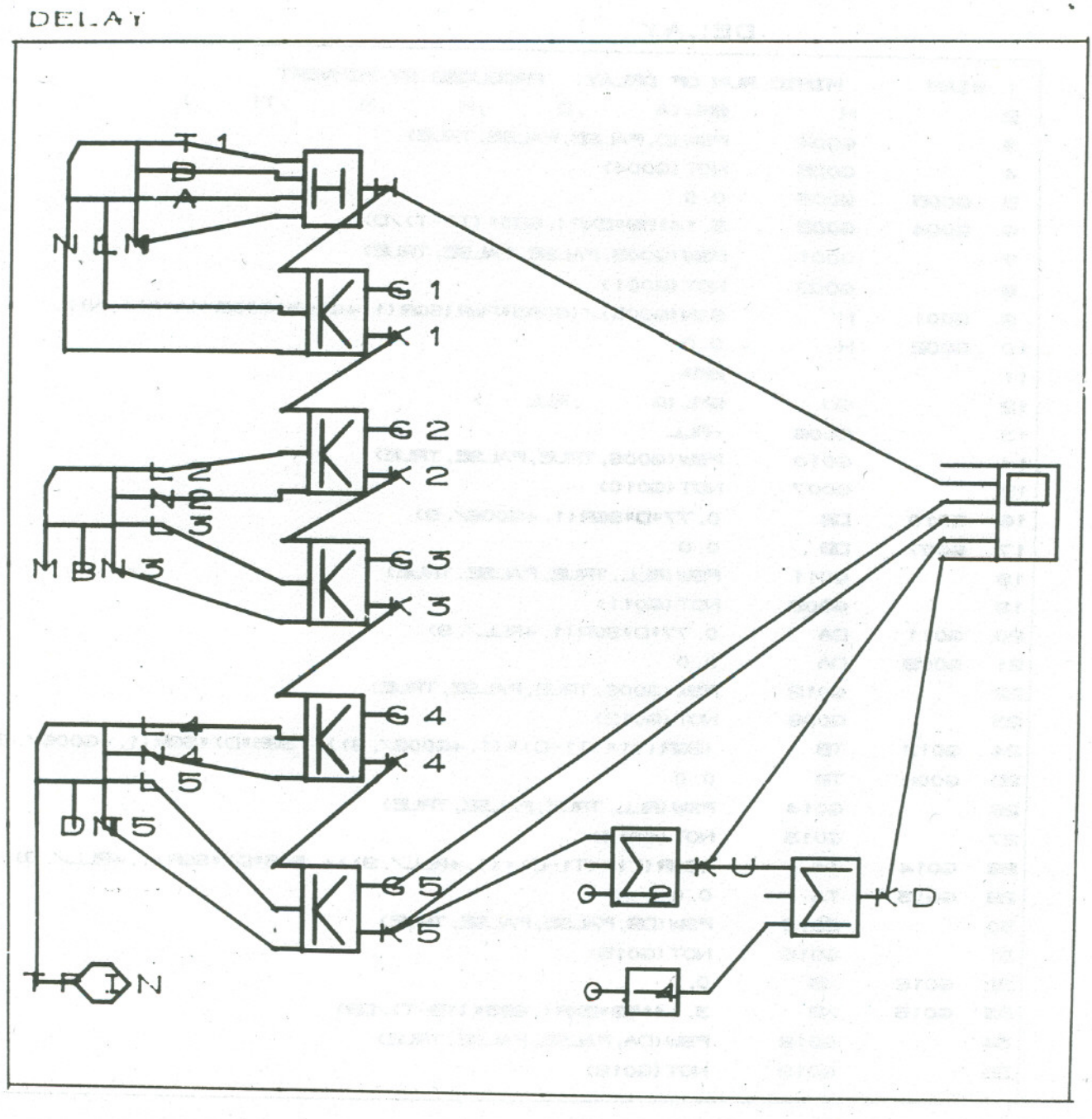


Fig. 22h. The model DELAY designed by Dan Maeder, University of Geneva to test a delay line model. DELAY is a macro definition nested to level 3.

DELAY

```
 1 *IAM       MIMIC RUN OF DELAY   PRODUCED BY MIMVERT
 2            H          BML(A      ,D      ,H      ,N      ,T1     )
 3            G004       FSW(D,FALSE,FALSE,TRUE)
 4            G005       NOT(G004)
 5  G005      G003       0.0
 6  G004      G003       3.14159*EXP(.693*(T1-T)/D)
 7            G001       FSW(G003,FALSE,FALSE,TRUE)
 8            G002       NOT(G001)
 9  G001      H          SIN(G003)/(G003*PWR(SQR(1.+G003*G003/(A*A)),N))
10  G002      H          0.0
11                       EMA
12            GU         BML(G      ,RLL    )
13            G006       -RLL
14            G010       FSW(G006,TRUE,FALSE,TRUE)
15            G007       NOT(G010)
16  G010      DB         0.77*D*SQR(1.+G006/.9)
17  G007      DB         0.0
18            G011       FSW(RLL,TRUE,FALSE,TRUE)
19            G009       NOT(G011)
20  G011      DA         0.77*D*SQR(1.+RLL/.9)
21  G009      DA         0.0
22            G012       FSW(G006,TRUE,FALSE,TRUE)
23            G008       NOT(G012)
24  G012      TB         (SQR(T1*(T1-D)*(1.+G006/.9))+.385*D)*SQR(1.+G006/.9)
25  G008      TB         0.0
26            G014       FSW(RLL,TRUE,FALSE,TRUE)
27            G013       NOT(G014)
28  G014      TA         (SQR(T1*(T1-D)*(1.+RLL/.9))+.385*D)*SQR(1.+RLL/.9)
29  G013      TA         0.0
30            G015       FSW(DB,FALSE,FALSE,TRUE)
31            G016       NOT(G015)
32  G016      XB         0.0
33  G015      XB         3.14159*EXP(.693*(TB-T)/DB)
34            G018       FSW(DA,FALSE,FALSE,TRUE)
35            G019       NOT(G018)
```

Fig. 22i. The first page of the MIMIC program produced by analyzing the model DELAY. Here the MIMIC macros H and part of GU are defined.

DELAY

```
36  G019    XA      0.0
37  G018    XA      3.14159*EXP(.693*(TA-T)/DA)
38          G017    FSW(XB,FALSE,FALSE,TRUE)
39          G020    NOT(G017)
40  G017    FB      SIN(XB)/(XB*PWR(SQR(1.+XB*XB/(B*B)),M))
41  G020    FB      0.0
42          G021    FSW(XA,FALSE,FALSE,TRUE)
43          G022    NOT(G021)
44  G021    FA      SIN(XA)/(XA*PWR(SQR(1.+XA*XA/(B*B)),M))
45  G022    FA      0.0
46          G       0.5*(DER(FA,T,0.)+DER(FB,T,0.))+0.5
47                  EMA
48          K       BML(G     ,K0    ,K     ,RLL   ,RNL   )
49          GU      CML(G     ,RLL   )
50          G023    G*RNL
51          K       G023+G024+K0
52                  EMA
53                  FIN(T,TFIN)
54                  PAR(RLL4,RNL4,RLL5,RNL5,DT,TFIN)
55                  PAR(RLL2,RNL2,RLL3,RNL3,B,M)
56                  PAR(T1,D,A,N,RLL1,RNL1)
57          H       CML(A     ,D     ,H     ,N     ,T1    )
58          K       CML(G1    ,H     ,K1    ,RLL1  ,RNL1  )
59          K       CML(G2    ,K1    ,K2    ,RLL2  ,RNL2  )
60          K       CML(G3    ,K2    ,K3    ,RLL3  ,RNL3  )
61          K       CML(G4    ,K3    ,K4    ,RLL4  ,RNL4  )
62          K       CML(G5    ,K4    ,K5    ,RLL5  ,RNL5  )
63          KU      K5+G026+0.2
64          G025    -0.4
65          KD      KU+G027+G025
66                  PLO(H,T,G5,K5,KU,KD)
67                  END
68
```

Fig. 22j. The second page of the MIMIC program produced from DELAY. Note that the MIMIC macro definition for K contains a macro call (CML) GU. The executable program begins on line 53 and consists mainly of macro calls on K.

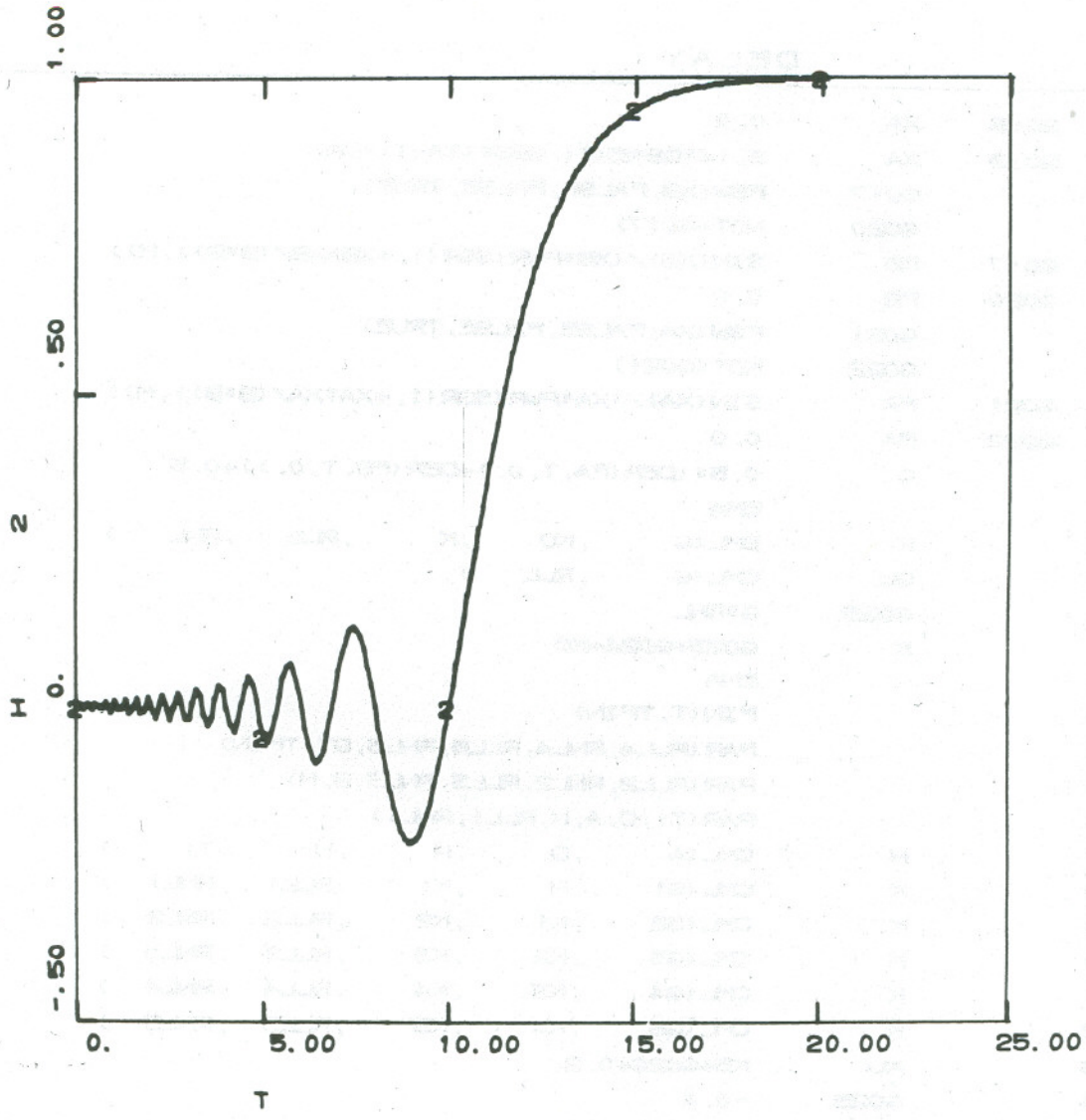


Fig. 22k. MIMIC output curves for DELAY, showing the form of H - the input signal.

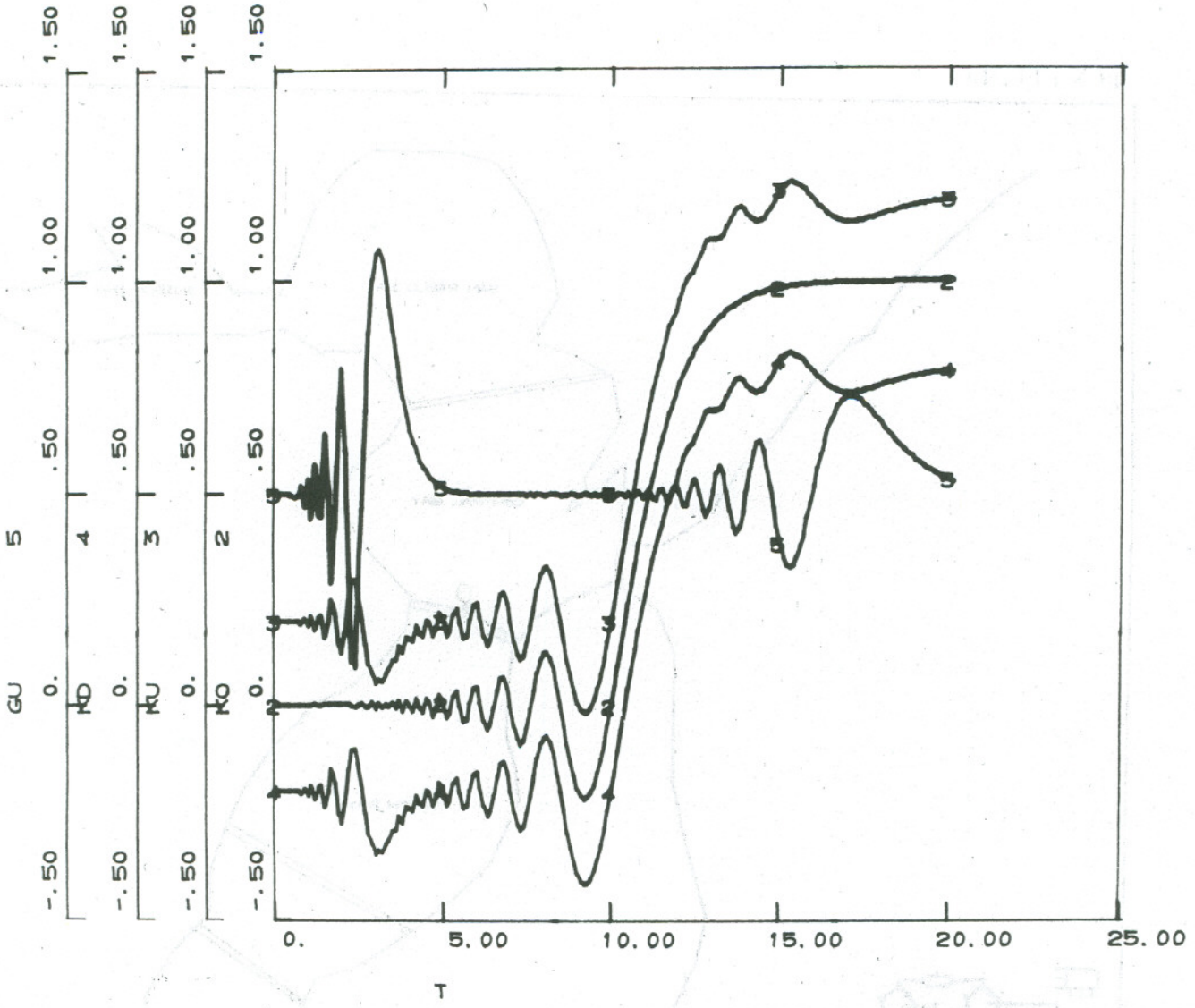


Fig. 22ℓ. MIMIC output curves for DELAY showing the effect of adding the input KO to some delayed derivatives (KU and KD).

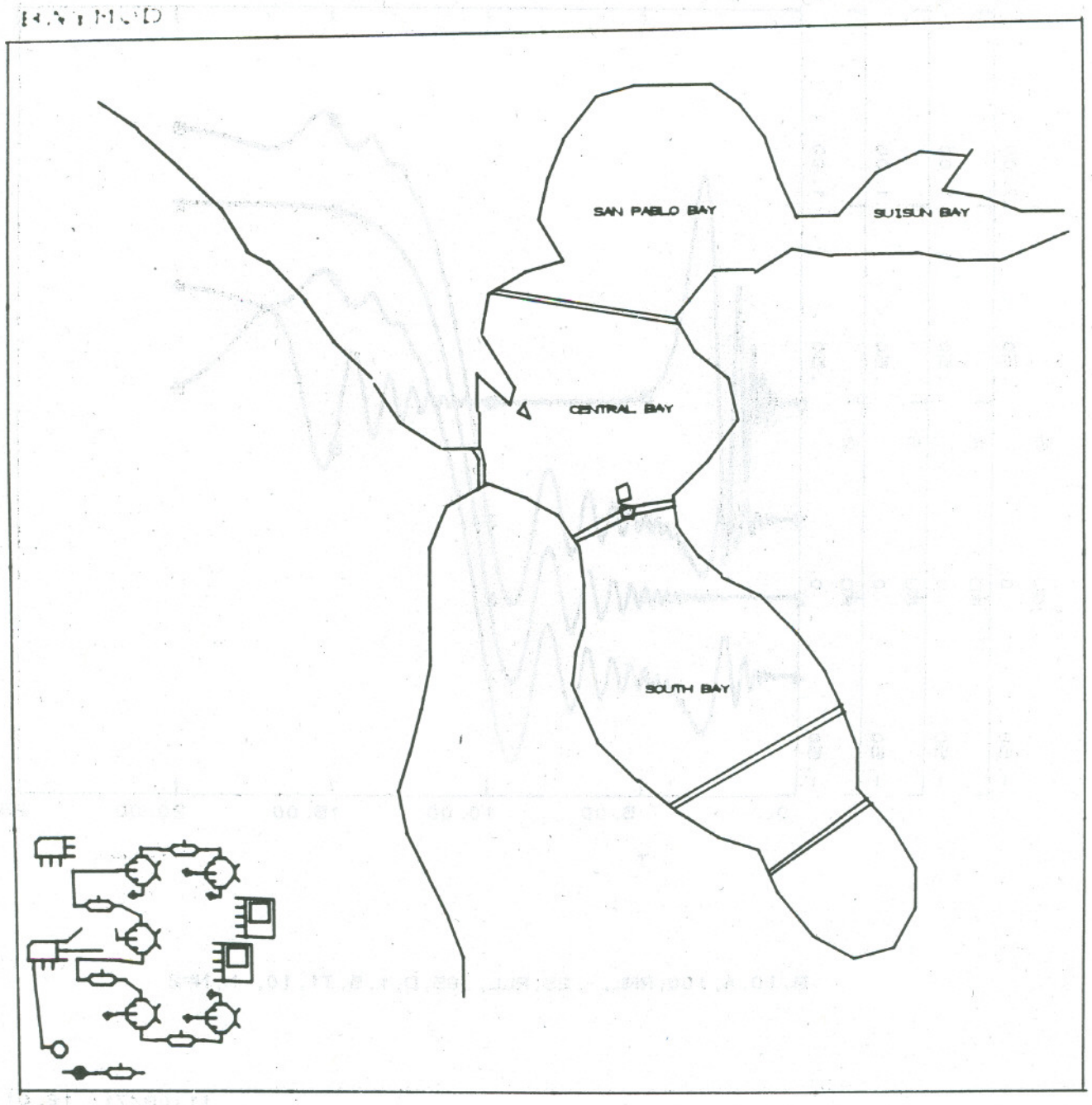


Fig. 23a. The model BAYMOD designed to study the flow of pollutants between the bodies of water comprising the San Francisco Bay system. The hand-drawn picture of the geography serves as a reminder of the meaning of the nodes in the model.

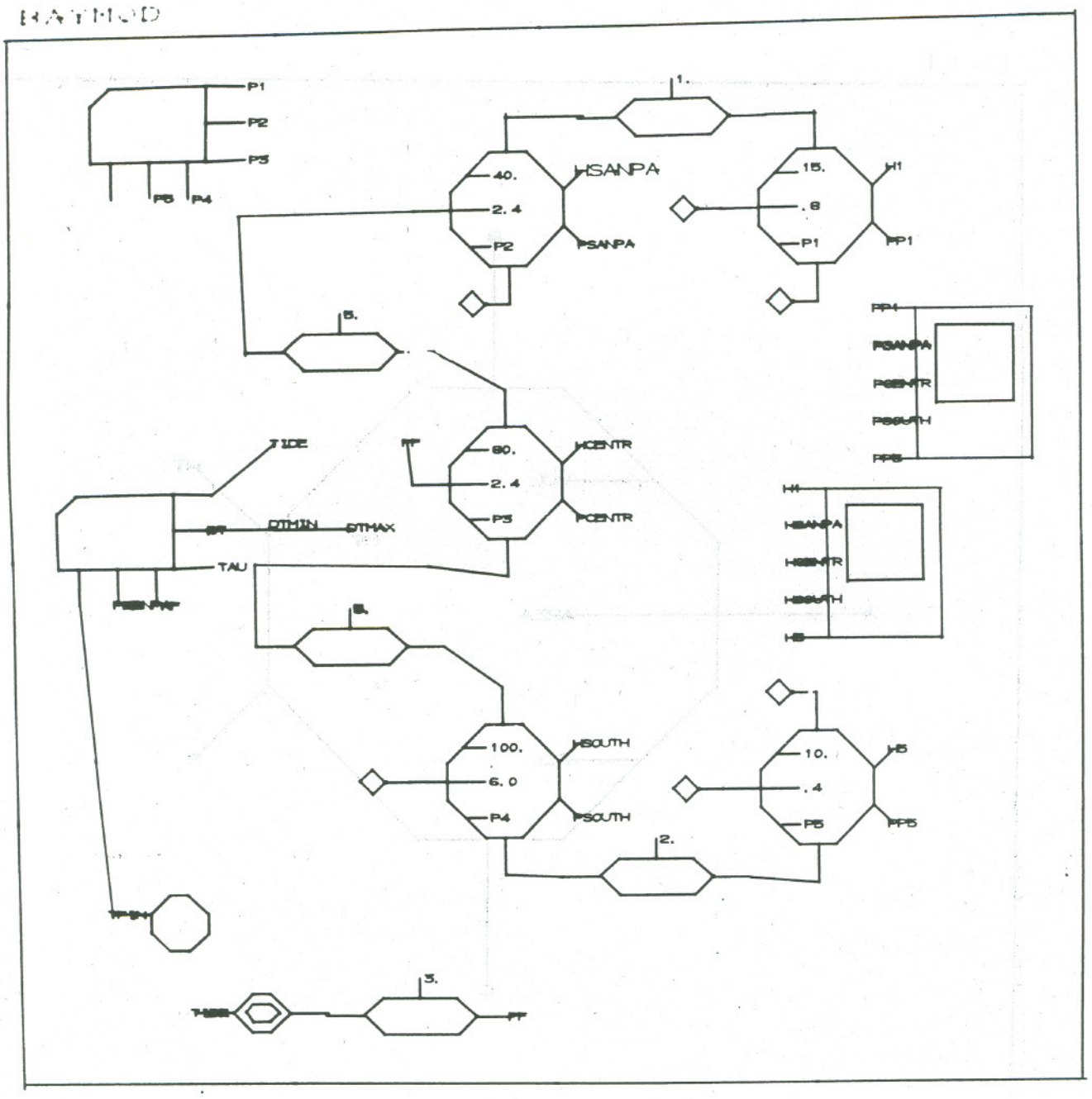


Fig. 23b. The BAYMOD model in ZOOM = 4, showing the labels (actual parameters) of the model.

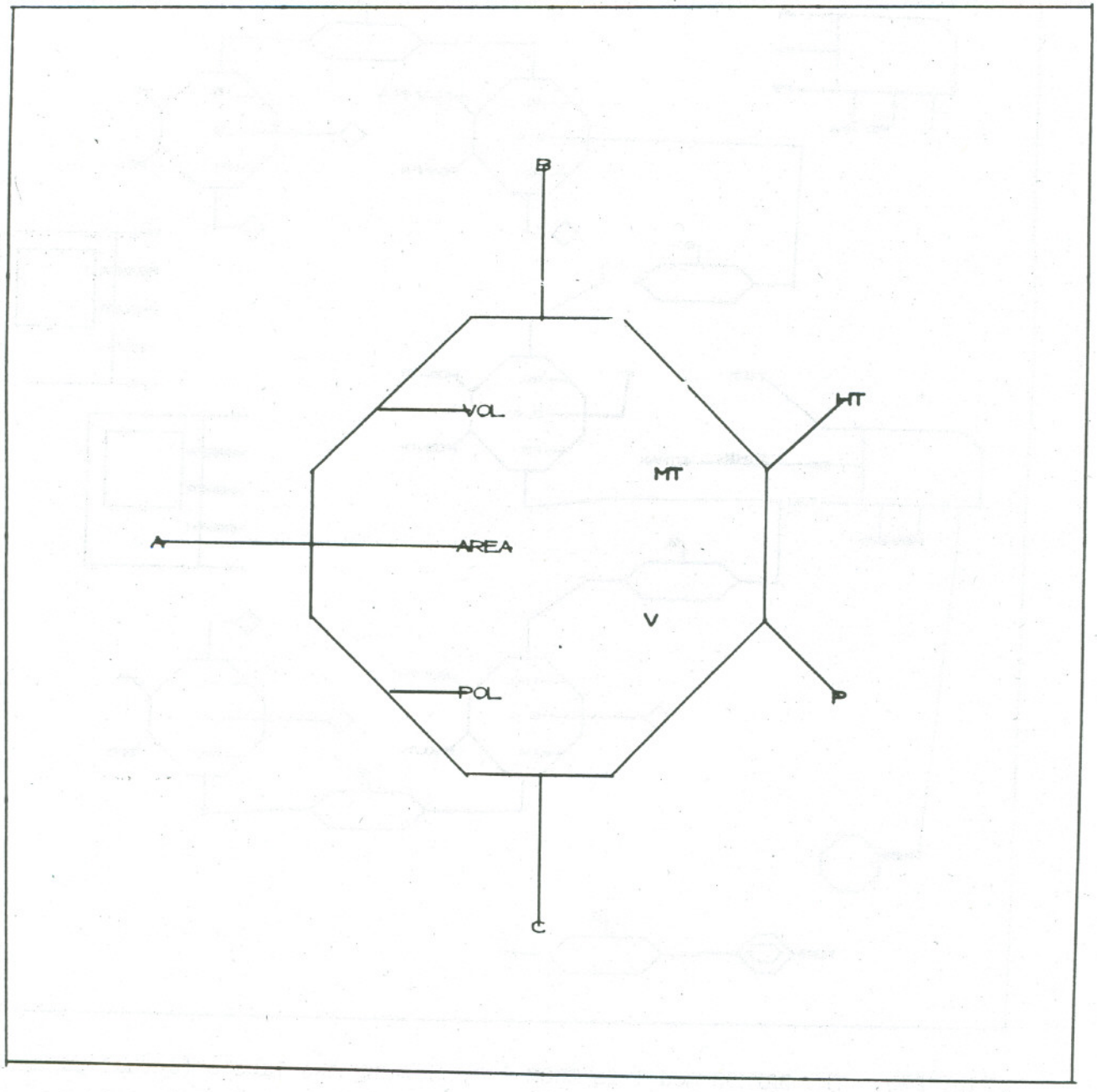


Fig. 23c. The symbol NODE used to represent the features of a bay.

NODE

1	V	INT (L→A+L→B+L→C, VOL)
2	HT	(V-VOL) / AREA
3	H→A	HT
4	H→B	HT
5	H→C	HT
6	P	INT (F→A+F→B+F→C, POL)
7	MT	P/V
8	M→A	MT
9	M→B	MT
10	M→C	MT
11		

Fig. 23d. The text definition of NODE in terms of the MIMIC Runge-Kutta integration function (INT). Note the use of PICASSO's concatenation character (→) which allows a single label to specify several variables.

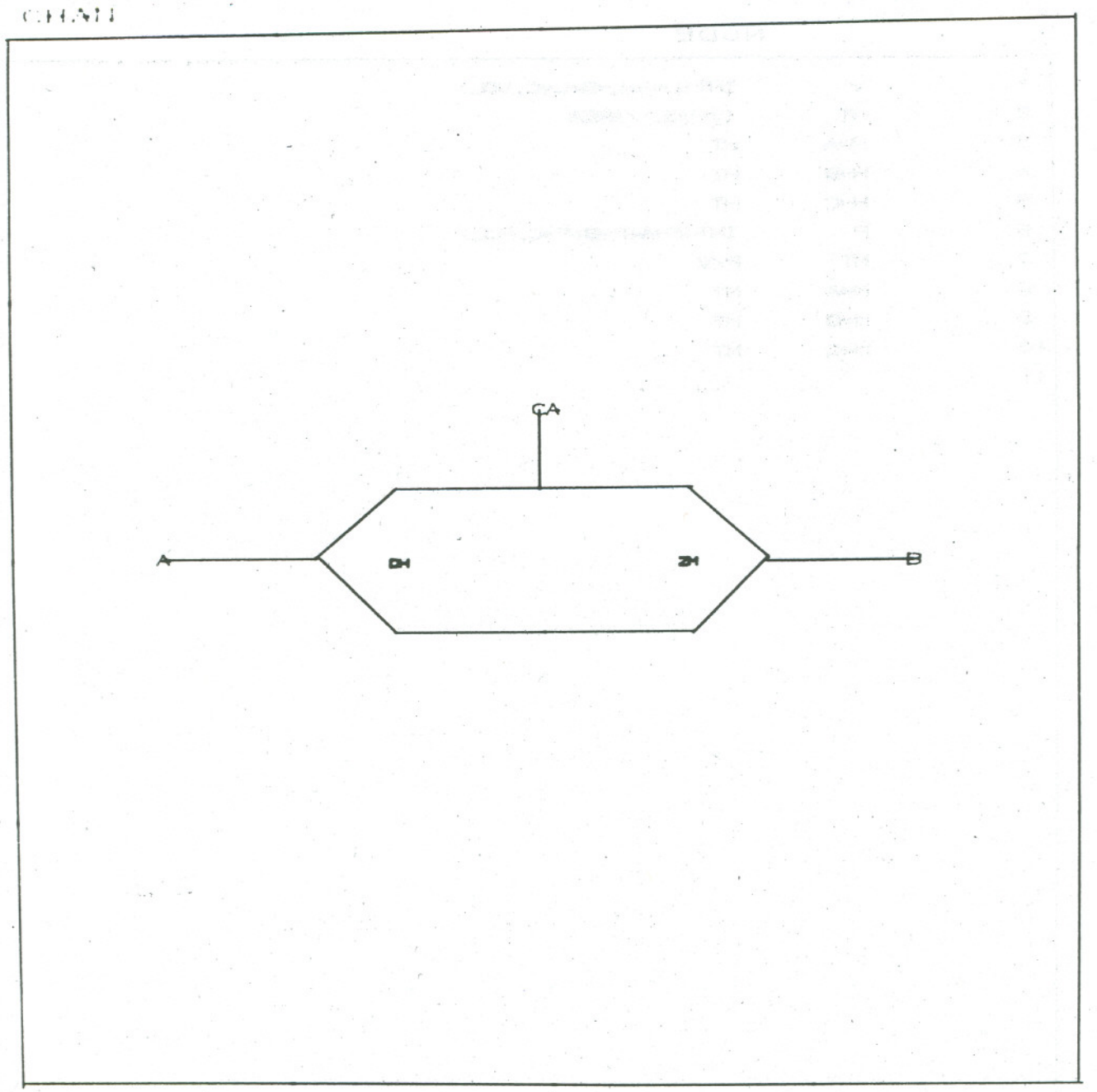


Fig. 23e. The symbol CHAN used to represent the flow channel between bays.

CHAN

```
1        DH      FTR(H→A-H→B, TAU)
2        L→B     DH*CA*KF
3        L→A     -L→B
4        ZM      MAX(DH+KD, 0.)*M→A
5        F→B     (ZM-MAX(-DH+KD, 0.)*M→B)*CA*KF
6        F→A     -F→B
7
```

Fig. 23f. The text definition of CHAN in terms of MIMIC functions. The first-order Laplace transform function FTR provides a model of the inertia of water driven by tides.

BAYMOD

```
 1 *IAM      MIMIC RUN OF BAYMOD  PRODUCED BY MIMVERT
 2           DTMIN     DTMAX
 3           DT        DTMAX
 4                     FIN(T,TFIN)
 5           HG004     TIDE=SIN(T*.5)
 6           MG004     0.
 7                     PAR(TIDE,DTMAX, TAU, KF, KD,TFIN)
 8           G003      FTR(HG001-HG009,TAU)
 9           LG009     G003* 5.*KF
10           LG001     -LG009
11           G006      MAX(G003+KD,0.)*MG001
12           FG009     (G006-MAX(-G003+KD,0.)*MG009)* 5.*KF
13           FG001     -FG009
14                     PAR( P1, P2, P3, P4, P5)
15           G007      FTR(HG002-HG010,TAU)
16           LG010     G007* 5.*KF
17           LG002     -LG010
18           G008      MAX(G007+KD,0.)*MG002
19           FG010     (G008-MAX(-G007+KD,0.)*MG010)* 5.*KF
20           FG002     -FG010
21           G011      FTR(HG004-HFF,TAU)
22           LFF       G011* 3.*KF
23           LG004     -LFF
24           G014      MAX(G011+KD,0.)*MG004
25           FFF       (G014-MAX(-G011+KD,0.)*MFF)* 3.*KF
26           FG004     -FFF
27           G017      INT(LG015+LG010+LG012, 100.)
28           HSOUTH    (G017- 100.)/ 6.0
29           HG015     HSOUTH
30           HG010     HSOUTH
31           HG012     HSOUTH
32           PSOUTH    INT(FG015+FG010+FG012, P4)
33           G016      PSOUTH/G017
34           MG015     G016
35           MG010     G016
```

Fig. 23g. The first page of the MIMIC program produced from BAYMOD.

BAYMOD

```
36          MG012     G016
37          G019      INT (LFF+LG009+LG002,  80. )
38          HCENTR    (G019- 80. )/ 2.4
39          HFF       HCENTR
40          HG009     HCENTR
41          HG002     HCENTR
42          PCENTR    INT (PFF+PG009+PG002,  P3)
43          G018      PCENTR/G019
44          MFF       G018
45          MG009     G018
46          MG002     G018
47          G024      INT (LG001+LG013+LG022,  40. )
48          HSANPA    (G024- 40. )/ 2.4
49          HG001     HSANPA
50          HG013     HSANPA
51          HG022     HSANPA
52          PSANPA    INT (PG001+PG013+PG022,  P2)
53          G023      PSANPA/G024
54          MG001     G023
55          MG013     G023
56          MG022     G023
57          LG015     0.
58          PG015     0.
59          LG022     0.
60          PG022     0.
61          G025      FTR (HG012-HG020, TAU)
62          LG020     G025* 2. *KF
63          LG012     -LG020
64          G026      MAX (G025+KD, 0. ) *MG012
65          PG020     (G026-MAX (-G025+KD, 0. ) *MG020) * 2. *KF
66          PG012     -PG020
67          G027      FTR (HG013-HG021 , TAU)
68          LG021     G027* 1. *KF
69          LG013     -LG021
70          G028      MAX (G027+KD, 0. ) *MG013
```

Fig. 23h. The second page of the MIMIC program from BAYMOD.

BAYMOD

```
71          FG021     (G028-MAX(-G027+KD, 0.)*MG021)* 1. *KF
72          FG015     -FG021
73          G032      INT(LG029+LG030+LG020, 10.)
74          H5        (G032- 10.)/ .4
75          HG029     H5
76          HG030     H5
77          HG020     H5
78          PP5       INT(FG029+FG030+FG020, P5)
79          G031      PP5/G032
80          MG029     G031
81          MG030     G031
82          MG020     G031
83          G036      INT(LG033+LG021+LG034, 15.)
84          H1        (G036- 15.)/ .8
85          HG033     H1
86          HG021     H1
87          HG034     H1
88          PP1       INT(FG033+FG021+FG034, P1)
89          G035      PP1/G036
90          MG033     G035
91          MG021     G035
92          MG034     G035
93          LG033     0.
94          FG033     0.
95          LG029     0.
96          FG029     0.
97                    PLO(T, H1, HSANPA, HCENTR, HSOUTH, H5)
98          LG034     0.
99          FG034     0.
100         LG030     0.
101         FG030     0.
102                   PLO(T, PP1, PSANPA, PCENTR, PSOUTH, PP5)
103                   END
104
```

Fig. 23i. The third page of the MIMIC program for BAYMOD.

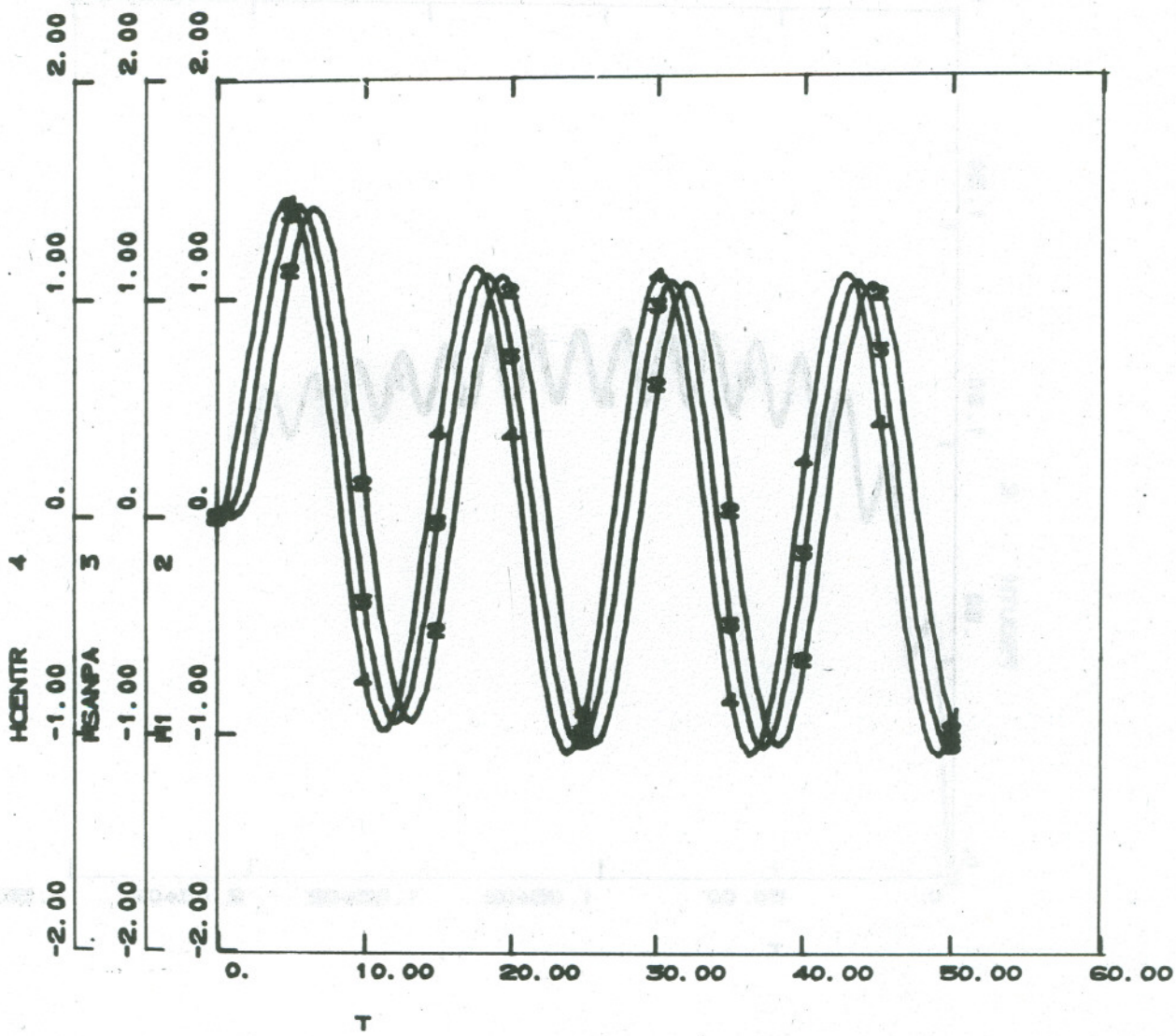


Fig. 23j. MIMIC output curves for BAYMOD.

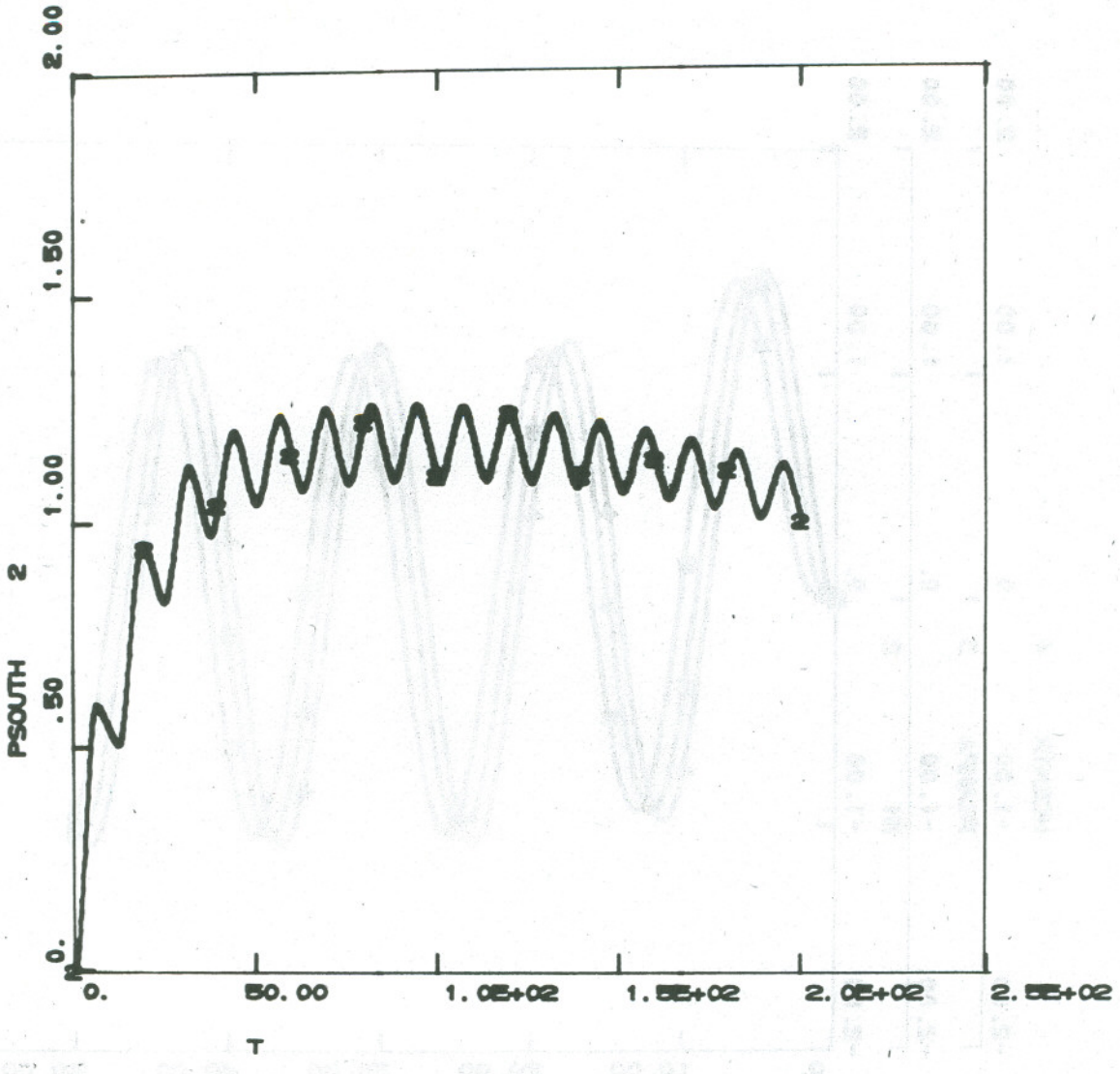


Fig. 23k. MIMIC output curves for BAYMOD.

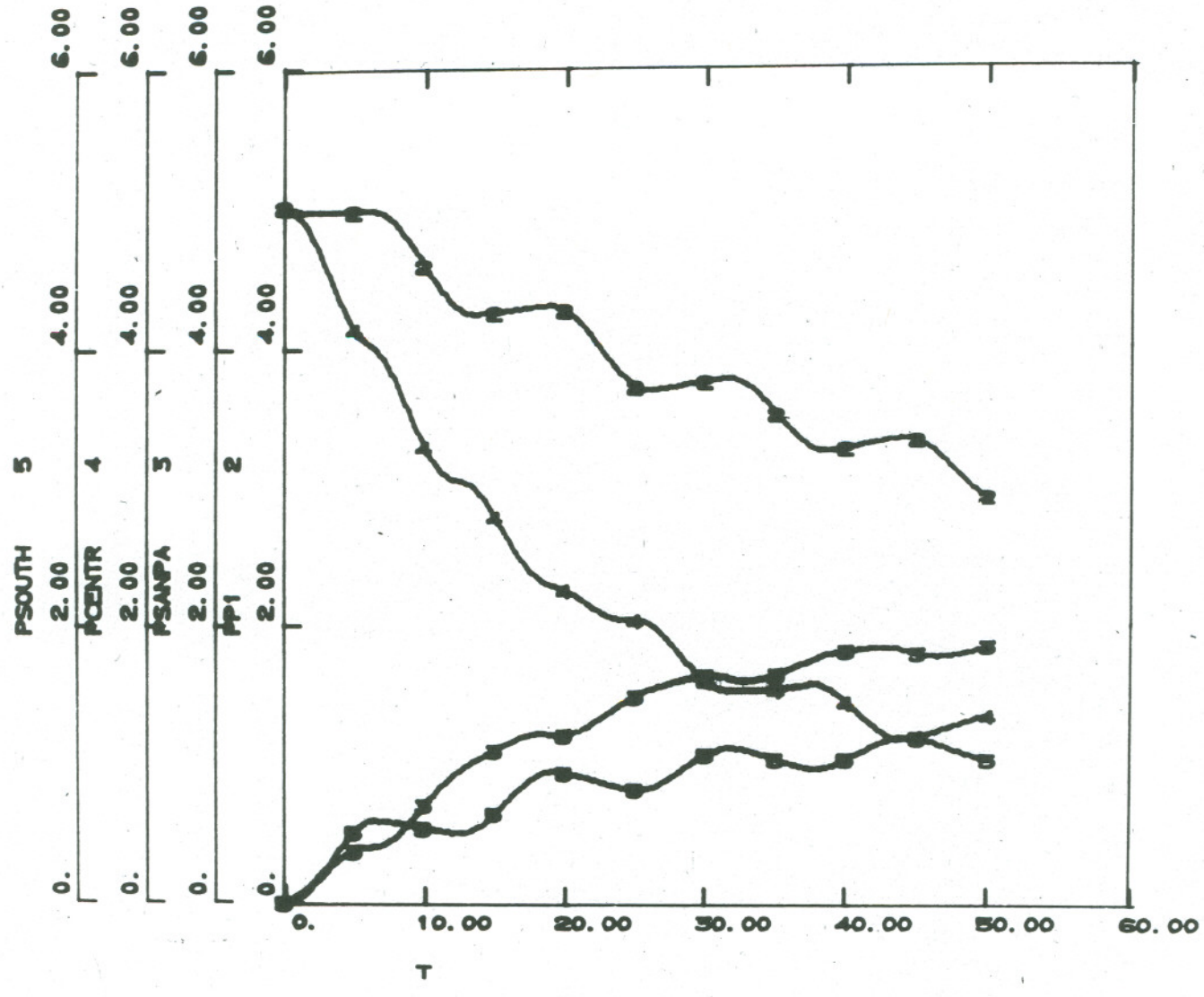


Fig. 23ℓ. MIMIC output curves for BAYMOD.

LEGAL NOTICE