Organic and Biological Chemistry

Computer-Assisted Synthetic Analysis. Facile Man–Machine Communication of Chemical Structure by Interactive Computer Graphics

E. J. Corey,* W. Todd Wipke, Richard D. Cramer III, and W. Jeffrey Howe

Contribution from the Department of Chemistry, Harvard University, Cambridge, Massachusetts 02138. *Received January 30, 1971*

Abstract: The application of computer graphics to input and output chemical structures is described. The graphics program which has been developed for use in computer-assisted synthetic analysis allows a high degree of interaction between man and computer. Details are given with regard to the performance of the graphics program, its organization, structure, and adaptation to a particular machine (Digital Equipment Corporation PDP-1).

Previously we have discussed the general theory of synthetic analysis as applied to complex molecules and a particularly simple yet powerful technique for the design of organic syntheses which has been termed the "logic-centered" approach.[1] In its "pure" form this method starts from the synthetic objective or "target molecule" and is directed in a series of analytical stages to a set of possible synthetic starting points. The first step is the derivation of the set of precursor molecules which can reasonably be expected to be converted to the target by one synthetic reaction or a simple sequence of reactions. Each precursor molecule so generated is then considered to be a target and analyzed similarly, generating a "tree" of synthetic intermediates. Each precursor is in some way simpler than the target from which it was derived or leads in further analysis to precursors which are simpler. The analysis terminates when precursors are elaborated which are considered to be relatively simple or readily available.

The exceedingly large number of intermediates which would be involved in a comprehensive synthesis tree for a complicated molecule could only be generated by the expenditure of very large amounts of time and effort by a knowledgeable chemist. Some time ago we embarked upon the task of writing an experimental problem-solving computer program to assist a chemist in the more cumbersome aspects of synthetic analysis. The results of the first phase of this work and a general background of the computer-assisted synthetic analysis technique have recently been presented.[1] The early program, designated OCSS (Organic Chemical Simulation of Synthesis), was designed to be used interactively in real time by the chemist. It made possible a wider and more rapid investigation of the synthetic tree and allowed an unbiased evaluation of the principles and systematics of the program in a way which led naturally to improvement and extension. The validity of the synthetic tree generated by the machine is a true reflection of the effectiveness of the logic-centered principles, since a computer cannot rely on "prior synthetic experience" or judgement.

(1) (a) E. J. Corey and W. T. Wipke, *Science*, **166**, 178 (1969); (b) see also E. J. Corey, *Pure Appl. Chem.*, **14**, 19 (1967).

Currently a second generation program called LHASA (Logic and Heuristics Applied to Synthetic Analysis), partly the result of experience with the earlier program, is in use and is serving as a vehicle for the next stage of evolutionary development. This paper introduces a series describing the organization, operation, and performance of LHASA. This series is designed to give the reader who has some knowledge of computers and a programming language a view of how the programming was done and some feeling for the decisions and ideas which are basic to the program design. There are a number of texts at varying levels which can supply fundamental information on computer science and programming at or above the level which has been assumed in these papers.[2] The division of material in the papers parallels the program functional divisions (see Figure 2). The graphical input and output have been previously discussed in detail.[1a] This paper is concerned with general program organization, interactive graphics, program control, and data structures. Subsequent papers cover chemical perception[3a] from the connection table, representation, selection, and operation of chemical transforms and strategy,[3b] and the analysis of cyclic networks.[3c]

This paper also describes the overall environment of LHASA and the data structures and internal relationships among the functional modules. One of the most important features of LHASA is the graphical input and output devices that are used. LHASA and its predecessor OCSS were among the first artificially intelligent prob-

(2) The following references are listed approximately in order of increasing sophistication: (a) A. I. Forsyth, T. A. Keenan, E. I. Organick, and W. Sternberg, "Computer Science, a First Course," Wiley, New York, N. Y., 1969; (b) A. Ralston, "Introduction to Programming and Computer Science," McGraw-Hill, New York, N. Y., 1971; (c) I. Flores, "Computer Programming," Prentice-Hall, Englewood Cliffs, N. J., 1966; (d) A. T. Berztiss, "Data Structures, Theory and Practice," Academic Press, New York, N. Y., 1971; (e) D. G. Hays, "Introduction to Computational Linguistics," Elsevier, New York, N. Y., 1967; (f) P. Wegner, "Programming Languages, Information Structures and Machine Organization," McGraw-Hill, New York, N. Y., 1968; (g) E. A. Feigenbaum and J. Feldman, Ed., "Computers and Thought," McGraw-Hill, New York, N. Y., 1963; (h) M. Minsky, Ed., "Semantic Information Processing," MIT Press, Cambridge, Mass., 1968.

(3) (a) E. J. Corey, W. T. Wipke, R. D. Cramer III, and W. J. Howe, *J. Amer. Chem. Soc.*, **94**, 431 (1972); (b) E. J. Corey, R. D. Cramer III, and W. J. Howe, *ibid.*, **94**, 440 (1972); (c) E. J. Corey and G. A. Petersson, *ibid.*, **94**, 460 (1972).

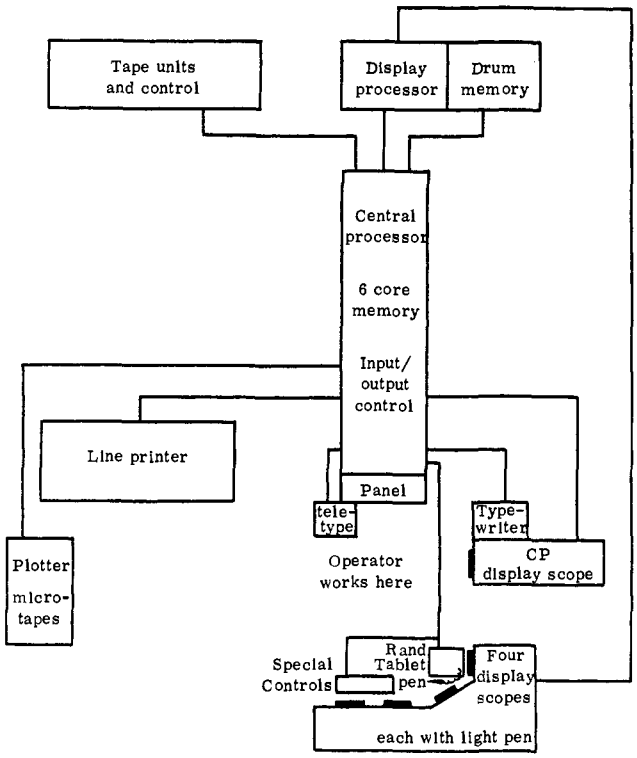


Figure 1. PDP-1 installation at Harvard's Cruft Laboratory. The diagram shows the layout of principal units of the computer. All peripherals are connected to the central processor (CP) except the four display scopes that are driven by a separate processor. The scopes receive instructions from the CP and can share the core memory.

lem solvers to use interactive graphics and the first in the chemical area to accept chemical structural diagrams hand-drawn in the normal way using an electrostatic stylus and tablet (Rand).[1a]

Hardware Organization. The computing hardware used in this research consists of a Digital Equipment Corp. (DEC) PDP-1 computer with a 24,576 (18-bit) word core memory. This computer is a single address machine with a basic cycle time of 5 μsec and *add* time of 10 μsec. It has two registers, the usual complement of fixed point arithmetic, logical, shift, rotate, and skip instructions, single level indirect addressing, and a 16-channel priority interrupt system. Associated with the system is a DEC tape transport capable of holding two microtapes, a 300-lpm (Analex) printer, a 131,072-word, slow-speed, magnetic drum (35 msec/256 words), a DEC type-30 scope for program editing, a DEC 340 display with four slave scopes and light pens, a Data Equipment Corporation Grafcon (Rand) tablet, a Cal Comp plotter, and various assorted devices for interactive input. The particular configuration is shown in Figure 1.

Software Organization. The PDP-1 system software includes one programming language called DECAL, which provides procedures; "for" loops; and conditional, arithmetic, and assignment statements, and which in these respects is similar to ALGOL. Subscripting and floating point operations can be executed interpretively. DECAL also permits intermixing of assembly language with DECAL source statements and can be extended by user-defined "action operators" and "instruction generators."

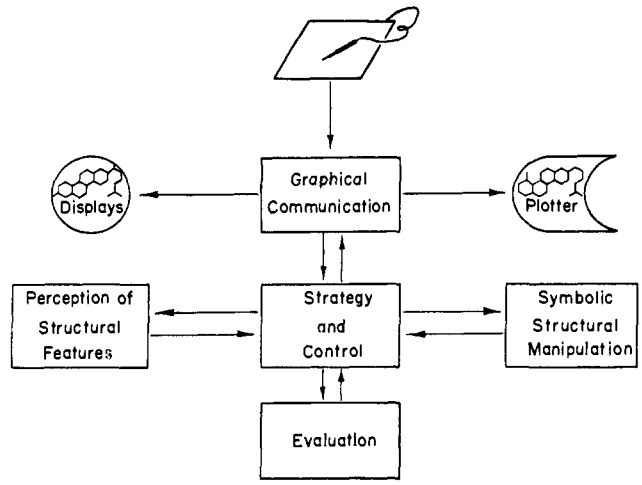


Figure 2. Functional subdivisions of LHASA.

The monitor system provides a linking loader which permits references between memory banks, a simple text editor which uses Type 30 scope, a debugging package called DDT, input–output routines, and a subroutine package called LIB340 to simplify use of the 340 display. The capabilities of the display system and a description of the LIB340 routines are given in Appendix I.

Organization of LHASA. The program is organized into five basic functional modules: (1) graphical communication, (2) strategy and control, (3) perception of structural features, (4) symbolic structural manipulation, (5) evaluation of precursors. The interrelation of these modules is shown in Figure 2. Each module performs a specific function and each has well-defined inputs and outputs. The control module contains a common data area, a library of basic support routines, and the master executive. Consequently, the control module is always resident in memory to coordinate the activities of the rest of the programs. All other modules are called by the executive and in principle can share memory with one another, *i.e.*, only one of the other four modules need be in core memory at a time—the time during which that module is being executed. Modular construction not only simplifies the programming but also provides a way to fit a large program into a small computer memory. In this way LHASA, a program of 50 K words, is able to execute on a computer with a memory of only 24 K words.

Each module consists of many subroutines and one or more main routines. Groups of these routines may form subfunctions which are mutually exclusive in execution and can also overlay one another in memory. Currently there are eleven overlay segments, each of which fits into one of three special regions of core memory. LHASA contains an overlay handler which assures that a segment is in core before passing control to it and also avoids rereading the segment from the drum if the segment is already present in core. The time required to bring an overlay segment (3 K words long) from the drum into core memory is about 0.23 sec.

Typical Analysis. A typical analytic session with the computer will now be described from two vantage points: that of the chemist treating the program as a

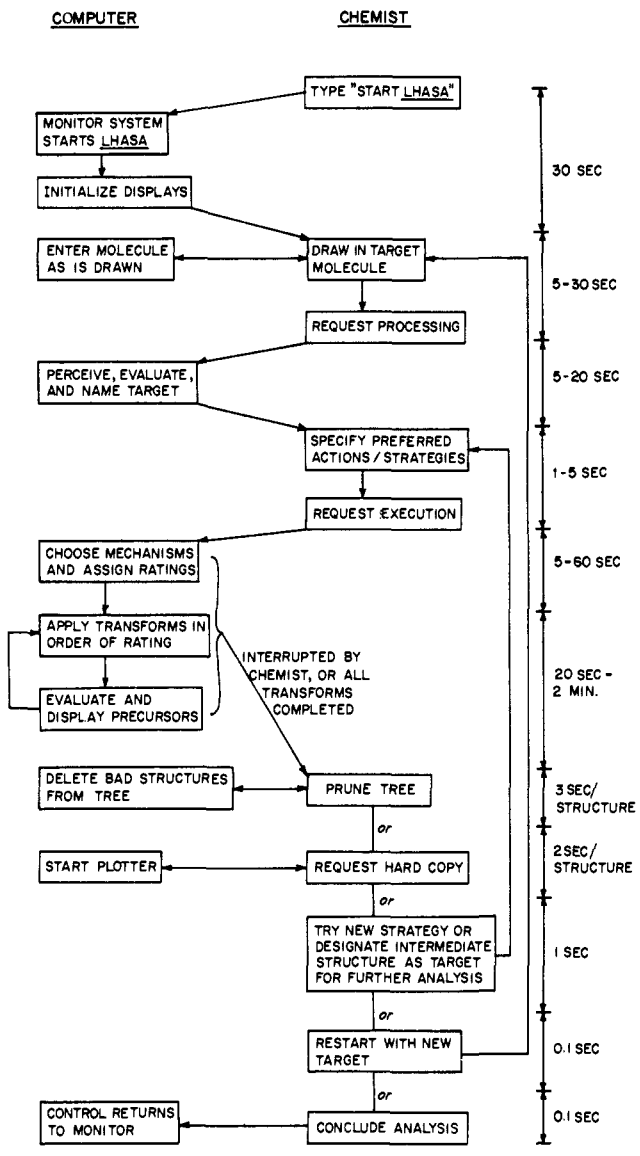


Figure 3. Man–machine interaction diagram. Arrows indicate the flow of control between the chemist and the computer during an analysis session. Approximate times required for each stage of processing are shown at right.

"black box" and that of the program's internal operations in response to the chemist's requests.

The overall flow of control between man and machine during analysis is diagramed in Figure 3. Typical elapsed times for each phase are shown at the right of the figure. While the computer has control, the chemist's controls are deactivated. Lengthy processes may be interrupted by the chemist by depressing the pen anywhere in the drawing box. The program then returns control to the chemist, flashing a message (*You called?*) when it reaches a convenient stopping place. As much as half of the time for an analysis is that required for the chemist to make a decision and notify the program.

Viewpoint of the Chemist. Once the program has been started *via* teletype commands to read the contents of a microtape into the computer and to start the program, there are two displays on the scopes at any one time (see Figure 4). Scope 1 contains the input display (display 1), and scope 2 has the title and buttons on what will later become the tree display (display 3).

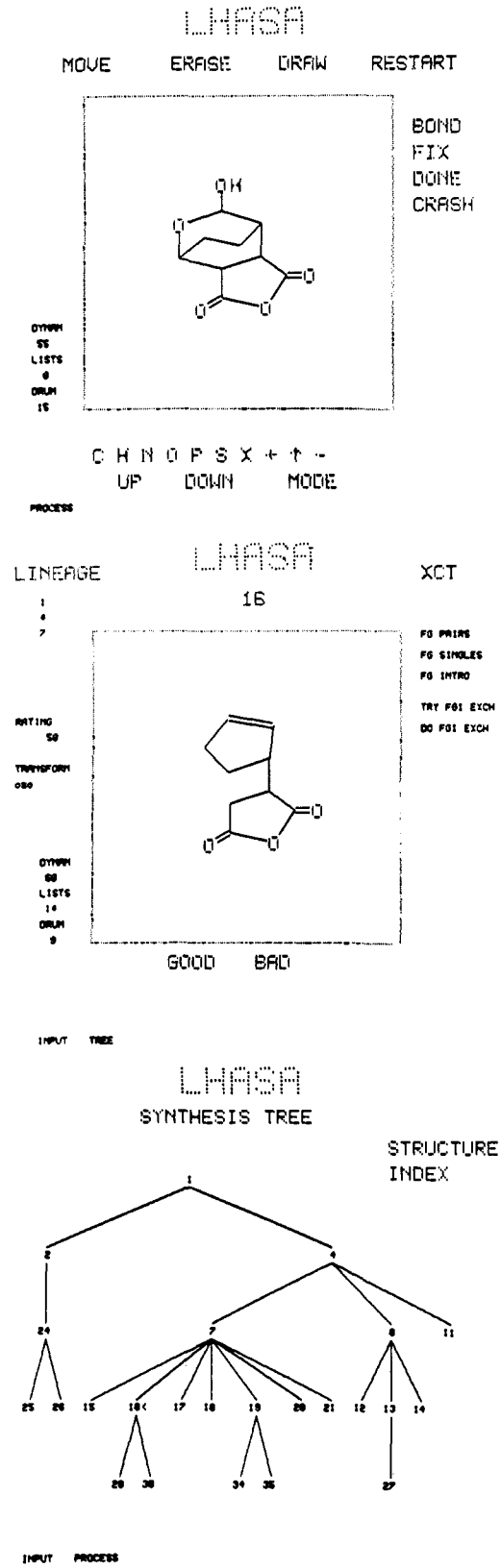


Figure 4. (Top) Display 1. (Middle) Display 2. (Bottom) Display 3.

(Display 2 replaces display 1 at a later part in the analysis.) The position of the tip of the Rand pen in x,y coordinates is displayed on scope 1 as a small cross. Whenever the chemist presses the pen down on the tablet, the cross becomes brighter. Most of the words and symbols on the display are "buttons" (control

words) which can be activated by moving the tracking cross *via* the pen to the first letter in one of the words and depressing the pen. The word touched will become brighter and remain so until the pen is depressed outside the drawing box. Whenever this operation is applied to the word DRAW, the program is ready to accept new structural input. Structures are input within the center box, using the standard organic chemical format, where a line represents a bond, and nodes, or the ends of lines, represent atoms. All atoms are assumed to be carbon unless specified otherwise. Since the program assumes saturation of all atoms, hydrogen atoms are drawn ordinarily only to specify stereochemistry or to create familiar-looking functional groups.

To draw a bond, the pen is depressed inside the drawing box and moved along in this "*pen-down*" state. A line appears joining the point where the pen was pushed down with the current coordinates of the pen. When the pen is lifted, the line is "frozen." If the end point of a line is reasonably close to the end point of another line, the end of the new line is shifted to that of the old. Thus drawings can be somewhat sloppy, but the display will be automatically corrected. Drawing additional bonds between two atoms which are already connected creates the display of a double bond, and then a triple bond. If one attempts to draw excess bonds to any atom, the message VALENCE EXCEEDED is flashed on the screen and the offending bond is erased.

Atoms may be specified as noncarbon by pushing the pen down on one of the six atomic symbols at the bottom of the drawing box. This transforms the tracking cross into whichever symbol was pushed. When the chemist then depresses the pen on a node in the molecule, a copy of the symbol "sticks to" the node. The three remaining symbols in the atom symbol line at the bottom of the box—plus, up-arrow, and minus—are used similarly to designate atoms that are positively charged, free-radical, and negatively charged. The presence of a charge on an atom allows attachments to that atom in excess of the normal valence.

Once a structure has been drawn, the chemist may straighten up the molecule or move certain atoms out of the way of others to make a drawing less confusing by pushing the MOVE "button." Then, when the pen is depressed on an atom, the atom appears glued to the tracking cross. As the depressed pen is moved, the atom and all bonds attached to it follow the pen tip until the pen is raised.

Input mistakes may also be corrected by pushing the ERASE button. While the ERASE word is bright, pushing the pen down on an atom causes deletion of that atom and all bonds connected to it. A bond may also be deleted by depressing the pen on the middle of a bond. Any isolated atoms remaining after an atom or bond deletion are automatically removed. Artistic disasters may be destroyed by pressing the RESTART button, which completely clears the drawing box and reinitializes the atom and bond tables.

Pressing DOWN and then pointing to a bond in the structure turns that bond into a dotted line. Dotted bonds may be restored to their normal state by the use of the UP button. This process is useful only to give a three-dimensional appearance to a display, since this program does not yet understand stereochemistry.

The chemist may specify particular parts of the molecule to be worked on. To do this, he indicates one or more sites of strategic disconnection by pushing BOND and then pointing to the desired bonds. These bonds become bright, and all nonstrategic bonds dim slightly. Pointing to a bond again restores it to its original state. The appearance of the display may be returned to normal by touching the MODE button, with the knowledge of the strategic bonds being retained. They can be shown as bright bonds again by pushing the MODE button once more.

At this input stage, preceding any analysis, there are no restrictions on the order or number of times that these input commands can be used. However, once the input stage is ended, only the MOVE and the BOND commands can be used to "fix up" structures, and if such later changes are to be made permanent, FIX must be pushed after the changes have been made.

The remaining buttons, DONE or CRASH, can be pushed at any time in the analysis to stop the program and return control of the computer to the system programs.

Once the target molecule has been completely drawn, the chemist begins analysis by pointing to PROCESS. The commands on scope 1 (display 1) are replaced by a new set of commands (display 2). The target molecule and the box remain unchanged by this operation. Meanwhile the program is "processing" the target, perceiving functional groups and rings[3a] and assigning an identification number. Return of control to the chemist is signalled by a message.

Ordinarily the chemist will wish to specify a strategy and request a synthetic analysis. The available strategies[3b] at present consist of one-group and two-group analyses, the latter with or without detection of valuable functional group exchanges. Pushing FG PAIRS or FG SINGLES selects a strategy; TRY FGI EXCH adds the search for group exchanges. DO FGI EXCH requests performance of a previously created list of exchanges. Pushing XCT then starts the selected analysis. The computer's default strategy is to try all strategies sequentially until one succeeds in generating a new intermediate.

Synthetic analysis proceeds in three stages: further perception, matching of the perception results against data tables,[3b] and display of the results. An appropriate message is flashed at the beginning of each stage. Each structure which results from the analysis of a given target molecule is displayed (display 2 of Figure 4) along with a three-letter mnemonic designating the transform (retrosynthetic process, see ref 3b) involved and a number designating the rating for that transform. In the example shown in Figure 4, the target molecule input by the chemist, a tricyclic anhydride, is shown in display 1; one of the intermediates (no. **16** in the synthesis (a cyclopentene produced by OZO, a transform corresponding to the synthetic ozonolysis of CH=CH to form CHO OHC) is shown in display 2; and the complete synthetic tree appears in display 3.

At the same time as a new intermediate is displayed, a new node bearing this structure's "index number" is added to the synthetic tree in display 3 (see below). The index number (in this case "**16**") appears in display 2 above the new intermediate. The index num-

bers for every ancestor of this intermediate are displayed in the upper left corner of display 2 under LINEAGE. For this example the ancestors of the intermediate **16** shown in display 2 are structures **7**, **4**, and **1**.

The display of each new intermediate remains on the screen while the computer is "evaluating" the structure, checking it for valence violations and possible duplication of an earlier structure. When evaluation is finished, a display of the current target (*i.e.*, the particular structure in the synthetic tree which is the subject for analysis) replaces the new structure in display 2 while another intermediate is being generated. If the structure failed its evaluation, it is deleted from the synthetic tree. Otherwise it may be recalled by the chemist after the analysis is finished, or interrupted by the chemist.

When the computer has finished its analysis of the current target, the chemist's control buttons are reactivated. The numerous options now available to him will be discussed.

The TREE button, in the lower left corner, transfers the display of the cross (and control by the chemist) onto scope 2, where the current synthesis tree appears (display 3). Each node in the tree is a control button, labeling one of up to one hundred intermediates in the computer's structure index. Pushing a node button is the way the chemist designates a new "current" target. The molecule corresponding to that node in the tree reappears on scope 1 and a cursor symbol "<" is generated next to the tree button being pushed. The new current target can be subjected to synthetic analysis just as before, by selecting a strategy and requesting execution. A permanent copy of the current target can be made on the Cal Comp plotter by pushing the display 3 button STRUCTURE, whereas pushing INDEX makes a hard copy of the synthesis tree display. A series of structures can be "stacked" for hard copy without waiting for earlier drawings to be finished and while a subsequent synthetic analysis is going on. A structure for which a plot has been requested is indicated by a "." symbol after the corresponding node in the synthesis tree.

Pushing PROCESS on scope 2 returns the cross display to scope 1. From display 2 the chemist can delete the current target from the synthesis tree by pushing BAD, while GOOD intensifies display of the synthetic tree branches leading from the input target down to the current target. INPUT allows return to the structural input display, to make any of the changes to the current target previously described as legal when the initial input phase has ended. From the input display the chemist may also RESTART the program with a new target structure or CRASH back to the monitor system.

Machine Execution. The flow of control in the man–machine system has just been examined. In discussing the flow of control within the machine, it is helpful to utilize a state diagram as shown in Figure 5.[4] The executing program may be treated as a limited-state machine which can only be in one state at a given time. The states are represented by ellipses. The labeled lines leading from each state represent the possible changes-in-state that may occur. A label on a line specifies the input condition that must exist for a given

(4) W. M. Newman, *AFIPS Conf. Proc., Spring Joint Computer Conf.*, **32**, 47 (1968).

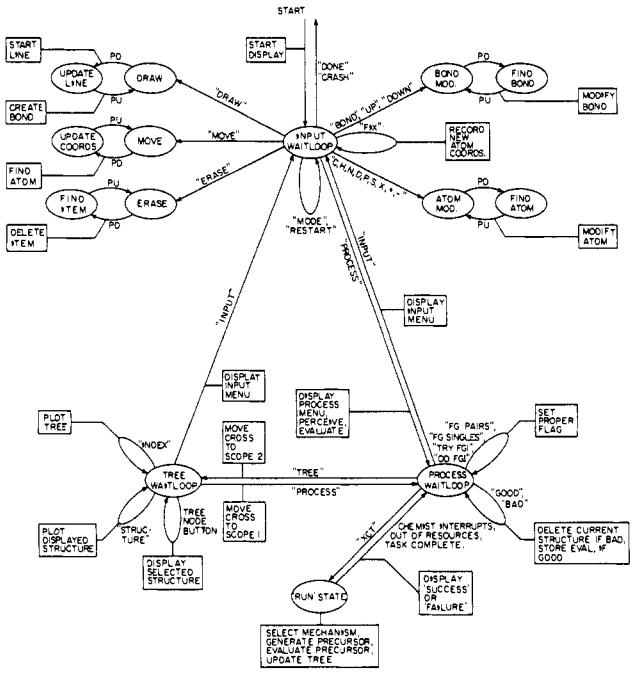


Figure 5. State diagram of LHASA. The result of "pushing" one of the graphics buttons is indicated by an arrow with the name of that button on it. Return to the INPUT WAITLOOP state from any of the five states attached to it (DRAW, MOVE, etc.) is accomplished by pushing the pen down anywhere outside the drawing box. The terms "PD" and "PU" mean "pen down" and "pen up," respectively.

change of state to occur. Thus, while in the INPUT WAITLOOP state, the program continually interrogates the RAND tablet for instructions from the chemist to enter a new state. For example, when the chemist points to the word DRAW, the DRAW state is entered. While in this state whenever the pen is pressed down inside the drawing box, a temporary line is started, and the line is updated until the pen is lifted. On the way back to the DRAW state, the line is entered as a bond. Drawing is allowed only within the drawing box. If the pen is pressed down outside the drawing box, the program returns to the INPUT WAITLOOP state, from which it may then enter any of the other connected states.

After a molecule is drawn and the PROCESS button is pressed, the input display is replaced by display 2, and the target molecule is perceived, evaluated, given a canonical representation, and entered at the top of the synthetic tree. These operations are described in later papers. The PROCESS WAITLOOP state is then entered. The program now waits for special requests or for XCT, the command to proceed. Buttons labeled DO FGI EXCH and TRY FGI EXCH cause certain flags to be set and then return the program to the PROCESS WAITLOOP state. The XCT button releases LHASA to enter the RUN state where it begins analysis of the current target structure in accordance with any special request flags that were set. If no flags were set, RUN chooses a strategy based on the structural features of the target molecule and guides execution according to that strategy (see ref 3b for details). The executive directs the selection of transforms, generation and display of precursors, evaluation of precursors, and updating of the synthesis tree. Return from the RUN state occurs automatically when it has completed generation of one level of precursors,

but may occur prematurely if the chemist interrupts or if the program is out of core memory or drum memory.

Graphics Programming. There are three basic control displays in LHASA, although only two scopes are used. These are DISPLAY 1, the input controls, visible on scope 1 when in the INPUT WAITLOOP state; DISPLAY 2, the processing controls, visible on scope 1 after LHASA enters the PROCESS WAITLOOP state; and DISPLAY 3, the synthesis tree and output controls, visible on scope 2 at all times (see Figure 4).

The graphics module consists of three types of programs, those which create static displays such as control words, those which create dynamic displays such as structural diagrams, and those which handle graphical interaction with the chemist. The static displays are created only once by GENFRM during the initialization process. GENFRM generates the display code executed by the 340 scopes for each of the three basic displays, including control words, the drawing box, the tracking cross, and the main *Frame* program shown below.

```
Frame
  top   Display LHASA              Title of program
        Call DISPLAY1/DISPLAY2     Control of INPUT/PROCESS
        Call STRUCTURE             Current structure pict
        Call INDEXPICT             Synthesis tree pict
        Call TEMPBOND              "Rubber band bond"
        Call DISPLAY3              Control of TREE state
        Call MESSAGE               Flashing messages to chemist
        Call STAT                  Statistics of memory resources
        Call TRACKER               Tracking symbol for pen
        Go to top                  Loop to refresh display
```

Substitution of DISPLAY 2 for DISPLAY 1 is accomplished by simply substituting the "call" in the appropriate position in the frame. The other picture calls are self-explanatory from the comments beside the calls to them. Pictures are removed from view by replacing the call to that picture by a call to a dummy picture which displays nothing.

GENFRM also sets up the lines of communication to be followed when any of the control buttons are pushed, by storing the name of the action routine with the location of the displayed control word text. Both the display and the lines of communication are established by a simple call to BUTTON ("*text*," *x*, *y*, *action routine*, *arbitary parameter*).

The remaining routines in the graphics module dynamically create the code necessary to display the structure (GENSTRUC), the synthesis tree (structure index) (GENIDX), and the lineage of the current structure (GENLIN). GENSTRUC operates from the connection table and creates display instructions to display the structure represented by the tables. As indicated above, bonds that are "down" are displayed dotted, and "strategic bonds" are displayed brighter than others. When a new picture code has been completed, GENSTRUC substitutes the new picture for the old.

Similarly, GENIDX operates from the actual tree structure of intermediates and creates a display program to display the structure of the current synthesis tree. Each node of the tree is labeled with the unique number designating the structure it represents. When a structure is evaluated by the chemist as being good, the corresponding symbol in the tree and the path from the target to that symbol are displayed more brightly than the others. In this way the chemist can easily keep track of those synthetic paths in which he is most interested even when the synthetic tree contains many nodes. The tree appears with each parent centered over its sons, and the sons are evenly spaced. The tree thus is always centered and balanced on the screen regardless of structure.

Table I. Summary of LHASA Control "Buttons"

DISPLAY 1	
DRAW	Enter the DRAW state
ERASE	Enter the ERASE state
MOVE	Enter the MOVE state
RESTART	Erase structure and tree and start fresh
C,H,N,O,P,S,X,+, ↑,−	Attach this symbol to the pen point for atom modification
UP	Enter mode for designating "up" bonds
DOWN	Enter mode for designating "down" bonds
MODE	Display/don't display specified strategic bonds
PROCESS	Leave INPUT state and enter PROCESS state
BOND	Enter mode for designating strategic bonds
FIX	Record new atom coordinates of modified intermediate
DONE	Terminate analysis session and return to monitor
CRASH	Return to monitor immediately
DISPLAY 2	
XCT	Begin analysis (Enter RUN state)
FG PAIRS	Try only 2-group chemistry
FG SINGLES	Try only 1-group chemistry
FG INTRO	Try functional group introduction (dummy)
TRY FGI EXCH	During analysis of possible transforms find group interchange subgoals
DO FGI EXCH	Try group-interchange subgoals if there are any
GOOD	Designate current structure "good"
BAD	Delete current structure and subtree if any
INPUT	Leave PROCESS WAITLOOP state and enter INPUT WAITLOOP state
TREE	Leave PROCESS WAITLOOP state and enter TREE WAITLOOP state
DISPLAY 3	
STRUCTURE	Queue current structure for hard copy output
INDEX	Queue picture of current tree for hard copy output
INPUT	Leave TREE WAITLOOP state and enter INPUT WAITLOOP state
PROCESS	Leave TREE WAITLOOP state and enter PROCESS WAITLOOP state
TREE NODE	Display this structure and make it the current target

The tree thus is always centered and balanced on the screen regardless of structure.

GENLIN traces the lineage of the current structure up the tree to the target structure and displays this under the word LINEAGE on display 2. The number of the structure is displayed directly above the drawing box. It also displays the name and the rating of the transform[3b] which produces the current structure. This information comes from the structure data block (see Figure 6). The numbers at the lower left of displays 1 and 2 indicate, from top down, the per cent of dynamic storage currently in use, the per cent of list storage currently in use, and the number of unused drum tracks (each track is 256_{10} words) (see below).

HIT5 is a collection of over 25 programs which handle interaction by servicing the numerous buttons on the displays. The main routine, called WAITLOOP, simply watches the pen status until it changes to *pen down*. At that time checks are made to see what state LHASA is in and whether the pen is in or out of the drawing box area. If the pen is outside the drawing box, then the pen coordinates are compared with the list of button coordinates. If the coordinates of the pen and

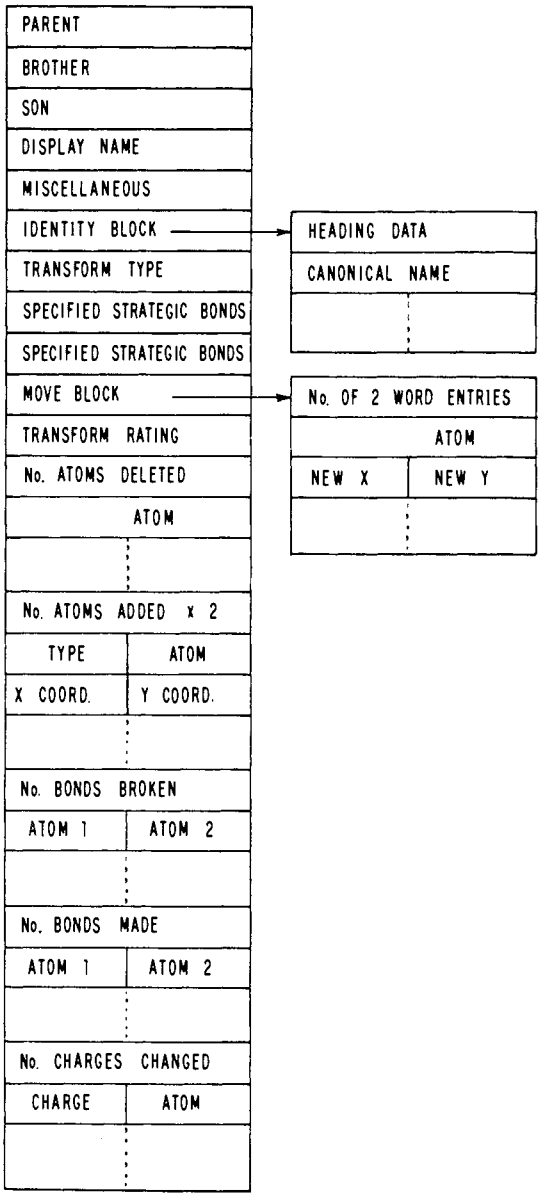
a button ~orrespond, control is transferred to the action address stored with those coordinates.

The action routine then may change the display, change the state of LHASA, or begin a long sequence of computation. Eventually the routine returns control to WAITLOOP, which waits for more interaction.

Two of the buttons call routines which initiate yet another type of graphic program, the plotter program. When a permanent (hard) copy of a structure is requested, RELOC is called which builds a relocatable version of the display code for the requested picture. The new display list is written onto the drum, and an asynchronous program called PLOT is initiated. This routine reads the display file on the drum and interprets it just as the DEC-340 hardware does, except that the program then outputs instructions to the plotter to simulate the actual display by the 340 scope. Whenever the plotter completes a command, it interrupts the computer, which may have been doing chemistry, asks PLOT to provide the next instruction, and then allows the computer to return to whatever it was doing. This programming technique allows the chemist to request many structures and then to continue chemical processing. A queue of structures is created on the drum. When PLOT finishes one structure, it releases space on the drum occupied by that structure and looks to see whether there is another. Since PLOT has all the instructions for plotting stored on the drum, its execution is independent of whether a structure exists in the synthetic tree at the time the structure is actually plotted—the structure may even have been deleted by the chemist.

Data Structures. The choice of representation for information within a computer can have a profound effect on the ease of manipulation of that information. We chose the connection table described in reference 1 as the fundamental representation of a chemical structure. This representation consists of the basic data inherent in the structural diagram—connectivity, atom types, and bond types. The stereochemical information which is represented is not presently used in the analysis. All information for structure display and chemical analysis is obtained from the connection table of the current structures *via* a set of simple access routines. The connection table (CT) used is not unique or canonical, is not concise, and is not convenient for human consumption but it does allow the computer direct and rapid access to structural information. An important feature of the CT is that cyclic molecules are handled as easily as acyclic ones and do not have to be treated as a special case.[5]

Other notation systems[6] gain compactness by using a single symbol to represent a group of atoms, but then demand explicit treatment in many special cases when information is requested which is only implicit in the notation. The CT differs from Dendral[7] in that Dendral is based on a tree structure and is a linear notation, whereas the CT is based on a generalized connected graph permitting cycles and is not a linear notation, although it can be made both canonical and linear if desired.[8]



LENGTH OF BLOCK = (No. DELETED) + (No. ADDED x 2) + (No. MADE) + (No. BROKEN) + (No. CHARGES CHANGED) + 16

Figure 6. Structure data block. Each structure in the tree is stored in virtual memory using this format. Only the structure which is currently being displayed on scope 1 and the original target have complete atom and bond tables (see text). The space for this block is dynamically allocated when the structure is first generated.

LHASA uses only two CT's, one for the target structure, which never changes during analysis and which is not available to the access routines, and one for the current structure. The latter is the active CT. Information about other structures is permanently stored in blocks, each corresponding to a node within the synthetic tree representation (see Figure 6). This block contains all key information about the structure including the changes that must be made to its parent in order to generate the structure. The first words of this block are pointers to the structure's immediate relations, its parent, immediate right brother, and immediate leftmost

(5) D. J. Gluck, *J. Chem. Doc.*, **5**, 43 (1965).

(6) For a survey of topological representations see A. Opler, *Proc. Amer. Doc. Inst.*, 499 (1964); F. A. Tate, *Annu. Rev. Inform. Sci. Technol.*, **2**, 285 (1967).

(7) J. Lederberg, G. L. Sutherland, G. G. Buchanan, E. A. Feigenbaum, A. V. Robertson, A. M. Duffield, and C. Djerassi, *J. Amer. Chem. Soc.*, **91**, 2973 (1969).

(8) H. L. Morgan, *J. Chem. Doc.*, **5**, 107 (1965).

son. These words are set to −1 if there is no such relation. The next word contains the number to be displayed in the tree as the name of this structure. Word 5 stores flags such as whether this structure was designated as GOOD or has been plotted yet; word 6 is a pointer to the canonical name for this structure. The next word carries the mnemonic which indicates the transform[3b] used to generate this intermediate from its parent. The set of bonds designated strategic by the chemist are stored in the next two words, while the following word points to a block giving the new coordinates of any atoms reoriented by the chemist. If such an action was not performed, or if no bonds were named strategic, then the corresponding word(s) in the structure block contains zero. Following the move block pointer is a space for storage of the transform rating.

The remainder of the block contains the changes to be made to the parent to generate this intermediate. There are five types of changes: deleting atoms (and bonds to them), adding atoms (involves computing reasonable coordinates for them), breaking bonds, making bonds, and manipulating electronic charges (either moving, deleting, or creating charge). The changes are grouped by type, each group being introduced by the number of changes of that type (zero if none). An *atom deleted* entry simply contains the name of the atom. Each *atom added* entry of two words contains the atom type to be created and the coordinates of the new atom. The operation of attaching a new atom to one already in a structure also makes an entry in the *bond made* section. An entry in the *bond made* or *broken* section simply contains the names of the two atoms involved in the bond.

The charge changing entries contain the charge type (−0 for neutral, −1 for anion, −2 for radical, and −3 for cation) and the atom on which the charge is to be placed.

The advantage of representing precursors in this manner is that the space required to store an intermediate structure depends only on the changes caused by a given transform and not on the number of atoms in the structure. Thus the structural part of a block necessary to describe the precursor of an aldol reaction requires 9 words of memory, regardless of the size of the molecule. In contrast, to represent this precursor by a complete connection and coordinate table would take up 160 words of memory for a medium-sized structure (20 atoms, 20 bonds).

The disadvantage is that extra time is required to "recall" a structure to "current" status in re-forming the proper CT. To do this, RECALL starts with the desired structure block and traces its lineage to the target. The target CT is copied into the current CT. The lineage of the desired structure is then followed in the reverse order traced—from the target down to the desired intermediate, the current CT being modified according to the changes indicated by each structure block in the lineage. The extra processing required for this is small, since although the tree may be very wide, it rarely attains a depth of greater than 12. With the current program and operating system the space saved is crucial, since the real limitation is available memory.

The canonical representation of the structure is simply a linear bit string constructed from the type of unique compacted connection table generated by the Morgan algorithm.[8] The length of the string used in LHASA is $2 + (14 \times nb)/18$ words where *nb* is the number of bonds between nonhydrogen atoms, and the words are 18 bits long. Thus the name for cubane is 11 words long. Duplication between structures is tested by comparison of names starting at the first word. Since the first word contains the number of atoms and bonds, the matching need proceed further only if the structures being compared have the same number of atoms and bonds. The Morgan format has been adequately documented elsewhere[8] and will not be discussed here.

The philosophy in the development of LHASA has been to let the data and the uses of the data determine subsequent data structures. This has resulted in a variety of different types of data structures and routines to handle them:[2d] arrays—the normal FORTRAN type of subscripted array, except the memory for them is dynamically allocated; *n*-component blocks—a block of contiguous dynamic storage accessed by special routines (a "dynamic" operation is one that is carried out as the program is running, in contrast to an operation such as allocation of array space by FORTRAN, which is done before program execution); set—a block of three contiguous words in which the *i*th bit is a *one* if the *i*th thing is a member of this set; otherwise the *i*th bit is *zero;* list—series of two-component "cells," the first component usually contains data, and the second addresses (points to) the next "cell" in the list; combinations—arrays of sets, lists, or *n*-component blocks; lists of *n*-component blocks; etc.

Each format was used where it was most efficient and convenient. Thus the set operations were found to be very powerful for parallel processing of structural information. The sets take advantage of the computer's ability to perform Boolean operations on 18 bits simultaneously. The use of sets has been described previously[1a] and is further discussed in the following paper.[3a]

Similarly, lists have been found convenient for storing variable length information quantities such as the atoms in a ring or the atoms in a functional group. Finally, conversion routines have been used which convert a list of items into a set of items for logical operations with other sets.

Resource Management. The use of diverse data structures in this work has been aided by a general dynamic storage allocation package[9a] which uses a simple algorithm developed by Dr. J. Goodenough for the Harvard PDP-1. Requests for blocks of storage can be of any size and are filled on a first fit basis. Free storage, originally one large block, becomes fragmented as storage is allocated and released. All blocks are then linked together by ascending core address, and a tag in each link is used to indicate whether the block is free or used. Allocation then requires a simple scan down the chain for the first free block large enough to fill a request. Release of a block changes the tag to indicate "free," and adjacent free blocks are merged to form one large contiguous block. This algorithm is not particularly efficient when a large percentage of dynamic storage is filled with small

(9) (a) D. E. Knuth, "The Art of Computer Programming," Vol. 1, "Fundamental Algorithms," Addison-Wesley, Reading, Mass., 1968, p 435; (b) *ibid.*, p 251.

blocks because the search time to find a free one is lengthy.

The percentage of dynamic storage used is displayed on scope 1. Typical storage utilization ranges as high as 90%. Running at such high utilization, however, is dangerous because one request which cannot be filled causes failure of the program. To prevent the program from continuing synthetic analysis when free storage is at a dangerously low level, the executive monitors the level and, when necessary, returns control to the chemist with the message "dynamic storage low." The chemist can then prune the synthesis tree to gain more storage and proceed.

The storage used for lists is separate from general dynamic storage and is maintained by the common method[9b] of having a list of free cells. Allocation of list cells is very efficient, since they are all the same size. Return of lists to the free cell pool is the responsibility of the programmer as with IPL-V,[10] in contrast to SLIP[11] or LISP,[12] which automatically return a list when it is no longer referenced.

Virtual Memory. In the earliest versions of the program no auxiliary storage was used, causing a severe demand for core memory as the number of intermediates generated grew. This occurred not only because the structure representation resided in core but also because the display file for the synthetic tree and other display information which grew with the number of structures also resided in core.

To relieve this problem, the effective size of memory has been expanded by 131 K of *virtual memory*. The virtual memory actually exists only on the drum, but through the use of an access package and some instruction generator macros, it may be referred to by the programmer much as though it were in core memory. Virtual memory is also dynamically allocated and released in a similar manner to core memory. Only minor changes to LHASA were required to move all structure blocks (Figure 6) and canonical names to virtual memory. This move increased the running time of the program only 5%. The virtual memory package, developed in collaboration with Mr. John Newell, will be discussed more fully in a later paper.

Acknowledgment. We are indebted to the National Institutes of Health for financial assistance to this project and to the Advanced Research Projects Agency for their support of the Harvard Center for Research in Computer Sciences.

Appendix I

The DEC-340 display controller obtains its instructions from the PDP-1 memory by "cycle-stealing." It can interpret only seven different kinds of instructions. The *parameter* (PRM) instruction specifies one of eight intensity levels, one of four scales, and provides a way to stop the display. The *point* (PNT) instruction positions the display beam in absolute scope coordinates in the x direction or the y direction. The *slave* (SLV) instruction specifies the particular scopes on which the picture is to appear. The beam on each of the other displays follows the same movements, but is blanked. The *subroutine* (SBR) instruction permits structuring of an image. Typically one positions the beam and then calls a symbol subroutine. The subroutine then draws the symbol using only relative vectoring instructions. In this way the same symbol may appear several places on the display.

The *vector* (VTR) instruction causes a vector (line) to be drawn from the current beam position to a point $(x + dx, y + dy)$. The 340 display can only represent a vector 128 units in x and y in one instruction. Thus the number of instructions required to draw a given line depends upon the length of the line. The *increment* (INC) instructions allow the specification of up to four moves of the beam. They are used to perform incremental plotting (*e.g.*, to generate a vector of 512 units in x and y) and also to define characters and special symbols.

Standard characters are displayed by a hardware character generator using the *character* (CHR) instruction. Because the format of the instructions varies, each instruction must tell the mode of the next instruction. A sample program to display the tracking cross and the title "LHASA" is given in Table II (explanatory comments at right).

Once started, the display runs independently of the PDP-1 central processor and produces a static picture. To change the picture being shown, the 340 display list must be changed. This can be done by changing selected instructions or by regenerating the complete list. In practice the former method is used for minor changes (*e.g.*, intensity or pen coordinates) and the latter for major alterations. LHASA utilizes a set of DECAL subroutines to create the display list dynamically and to handle dynamic storage allocation.

In LHASA there is a hierarchy of pictures—a *frame*, *pictures*, *and subpictures*. The frame is the master program made up of calls to pictures and subpictures. The scope control unit must start at the beginning of a frame. The display continuously runs through the frame. If the call to a picture is present in the frame, then that picture is visible. The routines to create these items are given below

frame name ⟸ SFRM (POSTS, *ia*, *fa*)
EFRM()

pict name ⟸ SPICT (*ia*, *fa*)
EPICT()

subpict name ⟸ SSUB (*ia*, *fa*)
ESUB()

where SFRM, SPICT, and SSUB declare the beginning of a frame, picture, and subpicture, respectively. POSTS is an array the length of the number of pictures. The routines use *ia* and *fa* as the initial and final address of a region of core in which the display code will be generated. If $ia = 0$ and $fa \neq 0$, then *fa* is used as an initial estimate for the amount of dynamic storage required for the display code. If $ia = fa = 0$, then a block of free storage is found which is just big enough to hold the display list generated. The routines return the beginning address of the display list.

DDN (decimal number)
DST (address of character string)

Display decimal number or character string.

LINE (dx, dy, intensify)

(10) A. Newell and F. M. Tonge, *Commun. Ass. Computing Mach.*, **3**, 205 (1960).
(11) J. Weizenbaum, *ibid.*, **6**, 524 (1963).
(12) J. McCarthy, *ibid.*, **3**, 184 (1960).

Table II. Sample Graphics Program Using DECAL

START:	SC2	ILV4					Scale 2 intensity 4
	SLV	5555	PNT				Display on all 4 slave scopes
	YBP	767	PNT				Position in Y
	XBP	319	SBR				Position in X
	DJS	CROSS	SBR				Call cross subroutine
	PNT						Enter point mode
	YBP	920	PNT				Position in Y
	XBP	400	PRM				Position in X
	SC8	ILV7	CHR				Scale 8 intensity 7
	··371410						Up shift, L, H[a]
	··012301						A, S, A
	··00						Escape from character mode
	SBR						Enter subroutine mode
	DJP	START					Loop to refresh display
CROSS:	DDS	CSAVE	VTR				Save return address
	VDX	−5	VDY	0			Invisible vector
	VDX	10	VDY	0	IFY		Draw visible line (horizontal)[b]
	VDX	−5	VDY	−5			Invisible vector
	VDX	0	VDY	10	IFY	ESC	Draw line (vertical) and escape
	SBR						Enter subroutine mode
CSAVE:	DJP	..	PRM				Return to calling routine

[a] Each character is represented by a 2-digit octal number (six bit code). [b] VDX n = let the component along X be set to n.

Table III. Program for Display of Movable Line

LOOP1: TEMP ⟸ SPICT (0, 0)	Start generating pict code
SCALE (1); INTENSITY (4)	Set scale and intensity
POINT (X1, Y1)	Move beam to start of line
LINE (PENX, PENY, 1)	Draw line to current position
EPICT ()	End picture code
ARRASE NEWLINE	Release old display code
NEWLINE ⟸ TEMP	Rename the picture
SUBSPICT (3, NEWLINE, SCOPE 1)	Substitute the new picture
IF PENDOWN THEN GOTO LOOP 1	Loop until pen is lifted

Draw line from current position (x, y) to $(x + dx, y + dy)$. If *intensify* = 1, the line is visible; if *intensify* = 0, invisible.

POINT (x, y)

POINT creates instructions to position the beam to (x, y).

SCALE (1, 2, 4, or 8)
INTENSITY (0-7)

SCALE sets the scale of the display to one of four values.
INTENSITY sets the brightness of the beam to one of 8 values.

CALL (subpict name)
POST (pict name, slaves)
INITIATE (frame name, slaves)
CSCOPE (i, slaves)
SUBSPICT (i, pict name, slaves)

CALL inserts a reference to a subpicture in the code being generated.
POST puts a picture in the *frame*, and *slaves* specifies on which scopes the picture should appear.
INITIATE starts the display processor execution of the *frame*.
CSCOPE changes the slave assignment of the ith picture in the *frame*.
SUBSPICT substitues a new picture for the ith picture in the *frame*.

The sample program in Table III dynamically generates a "rubber-band-like" line between point (X1, Y1) and the current pen position (PENX, PENY). The old display list is returned to available memory, and the new display list is substituted for its place in the frame. In this example PENX, PENY, and PENDOWN are dynamically updated by an interrupt handling program every 40 milliseconds (25 Hz).