STEPS – SPICE TEXT EDITING PACKAGE UTILIZING SCHEMATICS

by

Denis S. Yip

Memorandum No. UCB/ERL M91/43

15 April 1991

COVER PAGE

STEPS – SPICE TEXT EDITING PACKAGE UTILIZING SCHEMATICS

by

Denis S. Yip

Memorandum No. UCB/ERL M91/43

15 April 1991

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

TITLE PAGE

TITLE:
STEPS -- Spice Text Editing Package utilizing Schematics

ABSTRACT

Circuit designers have been designing circuits on paper and simulating them with SPICE for two decades. With the current techology, the designer now can have all the advantages of workstations and windows, including its graphical interface.

Simulation plays an important role in the design of present day integrated circuits, devices, and processes. STEPS - Spice Text Editing Package utilizing Schematics - is a CAD tool to allow designing in the schematic background and simulating at the same time. STEPS uses the structure of the Berkeley CAD Framework -- OCT, VEM, and RPC.

In addition on allowing the user to design and simulate the circuit in the same graphical schematics interface, STEPS also contains an internal electrical-rule checker to check for errors which can be detected in the initial phrase of the design. With the integration of STEPS and SPICE, circuit designers now can utilize the graphical interface to design the schematics on their workstations.

The motivation for this project is the lack of a standard public-domain schematic-capture package. Although commercial schematic-capture packages are available, those packages tend to provide their own libraries of software to suit their particular needs. STEPS, on the other hand, tries to be as flexible as possible to allow researchers to obtain a schematic-capture package to suit their own needs.

ACKNOWLEDGEMENTS

This project has been funded by a grant from the Semiconductor Research Corporation, and their support is greatly appreciated.

My research advisor, Professor Donald O. Pederson, has given me the initiative, motivation, and both technical and personal advices to me through my graduate career. Without his guidance, my project will not be at all successful. In addition, I would also like to thank him for his careful reading and editing of this report.

I would also like to express my gratitude to Professor Robert Meyer for three of his distinguished classes on analog integrated circuits. He has shown continued interest in the progress of this research, and I thank him for his timely advice and for reading this report.

Several Berkeley graduate students have also helped me while I pursued this research.

Special thanks is given to Dr. Theologus Kelessoglou for his help on expert system. Dr. Andrea Casotto has also given me tremendous help on supporting STEPS onto OCT, the Berkeley CAD environment.

The idea and format of STEPS are partially adapted from the RPC-SPLICE program from the University of Illnois at Urbana under Professor Resve Saleh. Professor Saleh and his students have given me numerous ideas on how the program can be improved.

TABLE OF CONTENTS

1 INTRODUCTION

STEPS, a schematic-capture package, is intended to enable SPICE users to design circuits and simulate them concurrently with a fast and robust package. Almost no text editing is needed since all SPICE information can be entered through schematic interface or dialog boxes in STEPS. All SPICE analysis can be done directly in STEPS, and various postprocessors are integrated inside STEPS to take full advantages of the available graphical postprocessors. For example, NUTMEG, a postprocessor of SPICE3, is available. Furthermore, an internal filter is constructed to allow conversion from SPICE2 output files to XGRAPH. Therefore, by pointing at the appropriate nodes, the user can directly obtain a graphical output of dc, ac, distortion, noise, fourier, or transient analyses. This interface emulates the functionality of an oscilloscope.

At present, most SPICE designers at Berkeley use text editing for entering data into SPICE input files. Although commercial schematic programs, e.g., ORCAD, support schematic entry for personal computers, schematic-entry packages are unavailable in the public domain to support the UNIX and X-Window systems. The limitation that most designers have to convert their schematic design diagrams into SPICE input files by naming all the subsequent nodes and entering parameters and values from a text editor is inconvenient. Nodes are easily misnamed, and corrections may be very cumbersome if the circuit is large. Moreover, no schematic printout is available.

In Chapter 2, the functionality of STEPS is presented. This chapter provides a general overview of all the features available in STEPS, and how these features are designed to suit the circuit designers' needs in the whole design process. Specific commands are not included, but special features of STEPS are studied in the separate sections of the chapter. The specific algorithm and coding of each feature, which are available through the OCT/VEM/RPC Release 8.0, are not presented in this report, but the purpose and the functionality of each specific feature which is useful for the circuit designers are addressed.

STEPS operates within the Berkeley CAD framework -- OCT/VEM/RPC. Chapter 3 gives a brief description on how the whole framework ties together, and which role STEPS plays in the structure. References are given for more details of the OCT/VEM/RPC framework. The major goal of this chapter is to understand that even STEPS is only a part of a larger framework. Fortunately, the remote procedural calling package (RPC) allows STEPS to edit the schematics and execute individual commands of the OCT/VEM/RPC structure.

In Chapter 4, the electrical-rule checker of STEPS is presented. This electrical-rule checker is introduced in STEPS as to check for design errors in an early stage of the design cycle. Problems introduced during the schematic capture are spotted early in the design before the simulation stage; therefore, corrections can be made quickly. The background of the electrical design-rule checker is introduced in the beginning of the chapter. Then, a rule-by-rule presentation analyzes the rules that have already been implemented in the electrical design-rule checker in STEPS. Finally, rules which are not practical to be developed in the STEPS environment are described. The reasons for this are brought out independently.

In Chapter 5, the performance of STEPS is evaluated. Although STEPS includes a significant amount of code (approximately 23,000 lines of C code), the run-time is very fast in the OCT environment. Users therefore do not find the execution time of STEPS to be a burden. Individual run-time performance of the electrical-rule checker is analyzed in Section 5.2 independently.

Chapter 6 compares STEPS with two commercial schematic capture packages - Tektronix's QuicKic and Viewlogic's Workview. These two packages are available and particularly designed for workstation purposes, although a PC interface is also possible. Strikingly similarities are found between the two packages and STEPS. The major difference found is that neither package supports a fully developed electrical-rule checker.

2 FUNCTIONALITY

2.1 Summary

STEPS has been developed to include the complete circuit design process. Many features are introduced not only to make the schematic-capture editing process easier, but also to allow as much convenience and flexibility as possible on the entire circuit design process.

2.2 Structure

STEPS utilizes the structure of the Berkeley CAD Framework -- Oct, VEM, and RPC (Remote Procedure Call Package) [1]. A framework has been developed to allow the integration of CAD tools across many technologies, levels of abstraction, and styles of design. Oct is a centralized data manager for VLSI design data; VEM is a graphical/textual browser and editor for Oct; RPC is a Remote Procedural Calling package for controlling remotely executing CAD tools from VEM. STEPS can utilize all the components and programs in the Oct/VEM/RPC framework. The basic schematic technology of VEM has been altered in STEPS to provide an easier user interface.

The palette in VEM is also modified so that all devices which can be represented in SPICE are included in the symbol palette. Resistors, inductors, capacitors, dependent sources, transformers, voltage sources, current sources, voltage sources connected to ground, ground, voltage supplies, diodes, JFET's, BJT's, and MOSFET's are displayed by windows. Selection and connection of one or more of these various devices are very convenient.

2.3 Editing

Schematic editing is a major feature of STEPS. To allow compatibility of STEPS with the Berkeley Oct/VEM/RPC framework, STEPS utilizes all the convenient editing commands used in VEM. Dialog windows are popped up when information is required.

Due to the fact that the number of transistors may be very large in a practical circuit, several features are included to hasten the circuit editing process. For example, all model parameters of a device, (i.e. BTJ, diode, JFET or MOSFET), are saved globally. The user can change any global parameter as often as he wishes. On the other hand, if there is a specific device which has different parameters than those saved globally, the user can change the specific local parameters of a device through another command. A large amount of time can be saved to change all the parameters on a unit basis.

2.4 SPICE netlist generation

STEPS allows the conversion of schematic diagrams into a SPICE netlist file automatically at any point in the design process. Naming of nodes is automatic in STEPS. All the node names, node values, model names, and model values are named accordingly and then displayed. Users are free to change the model names and model values as often as necessary.

By invoking the command "spice-file", users are allowed to generate an automatic SPICE netlist in ASCII file format. Editing and viewing of this netlist is allowed both under STEPS or under text text editor. The SPICE input file is allowed to be simulated concurrently in STEPS.

Another function of STEPS is to allow easy monitoring of the circuit. Very often in circuit design, an input device, most commonly a voltage source or a current source, is included in the circuit for monitoring means. To allow easy temporary attachments for monitoring purposes, STEPS allows users to use the command 'add-volt-source' or 'add-curr-source' to attach an input device anywhere in the circuit. The user needs to point to the node where the input device is to be attached, and invoke the specific command. A voltage source or a current source is introduced in the SPICE netlist, without any permanent attachment in the actual schematic circuit.

Figure 2.4.1 presents a schematic diagram and Figure 2.4.2 is a SPICE input circuit generated from the schematic diagrams by STEPS.

2.5 Display

For the user's convenience, STEPS displays several parameters for the user's information on his input. Please refer to Figure 2.5.1 for an example of the display. The displayed parameters are divided into two sections. One section provides model values, which includes model names and the model values. For example, a resistor may display r1 and 100 which means the resistor is named r1 and has the value of 100 ohms in the Spice input file. NMOS transistor may display M3 and 4/2 which means the NMOS transistor is named M3 and has the Width/Length parameter of 4/2. Similarly, a npn device may display Q2 and x3 which means the npn is named Q2 and has the area value of 3 times the nominal values. The node numbers are also displayed with a different color (black versus blue and red).

The display of STEPS is designed such that different features of the schematic capture have different colors representing them in a multi-colored workstation. The color blue represents a device model; red represents the model value; green represents the model name; black represents the node name; brown represents the wire connection. All the above representations are displayed in the workstation or in the hardcopy output.

The electrical-rule checker of STEPS changes the display of STEPS by highlighting any rule violation found internally. This feature is designed to allow easy search of the location which has violated specific rules under the electrical-rule checker.

STEPS allows the user to turn off the highlight from the electrical-rule checker and also turn off any portion of the model names, model values, and node names. The user is therefore the controller of the whole display. He can choose the specific parts that he wants to see to be displayed.

Fig 2.5.2 is a STEPS palette of all the models availalbe.

2.6 Printouts

Printouts of STEPS displays are available in both the OCT database format and the PostScript format. Users can select to print part or all of the circuits as well as to print any portion of the model names, model values, and node names. Text-writing is also allowed using the OCT database structure so that hardcopies of the circuits as well as textual information are readily available for presentation purposes.

2.7 SPICE Control Cards

The SPICE-Control commands are designed for the convenience of SPICE users to edit the SPICE files without going through text editing. Therefore, the users may edit all the SPICE options through STEPS' dialog windows. The commands and parameters needed are requested automatically by dialog windows. The SPICE editing is done with a higher speed since the users do not need to type in the whole line of options. Also, many values are automatically generated by schematic arguments to further fasten the SPICE editing process. If the user wants to delete an option (not using that option on subsequent SPICE files), he can invoke the option command and press the CANCEL button on the dialog window. For example, if the user has initially chosen the .temp option but wishes to cancel the .temp option in subsequent runs, he can simply invoke the ".temp" command and then press the CANCEL button when the dialog window is opened. STEPS automatically drops the option from the SPICE input file. The SPICE-Control commands currently implemented in STEPS are as follows: .options, .temp, .width-in, .op, .dc, .ac, .tf , .sens, .disto, .noise, .tran, .four, .print, .plot, .ic, .nodeset, and user-options.

The 'user-options' feature of STEPS allows the user to support his own modified version of SPICE, since individually developed SPICE versions may have different commands than Berkeley standard SPICE. The user has two options to enter his own text entry. He can simply prepare a text file called ~/.steps-options in his home directory or enter the commands through dialog windows. Once this command is invoked, the user is asked to select whether to 'activate' or 'deactivate' the 'user-options' command. Once it is activated, the user is asked to select whether to enter his own text entry in the include file or in the dialog window.

2.8 Simulation

STEPS are integrated with different simulators to allow the user to choose his favorite simulators in conjunction with STEPS. Currently, Berkeley SPICE2G.6 and SPICE3d4 are integrated with STEPS. While running STEPS, the user can invoke SPICE from the schematic diagram. Therefore, the user can look at the output promptly and directly from schematics. The user is then allowed to edit the schematics until the correct result is obtained.

For SPICE2 users, STEPS allows internal conversion so that a graphics plot instead of an ASCII plot of SPICE2 can be displayed. This feature is also convenient for SPICE3 users if a hardcopy is needed since some printers do not support SPICE3 Nutmeg hardcopy plots.

Another feature offered by STEPS allows the user to inspect previous circuits. Therefore, the user who has changed a schematic but is unsure about the change can test out the previous circuit that he has designed.

Users can introduce their own executable programs to complement other CAD tools into STEPS. Many SPICE versions (called "alphabet spices") are available with special models suitable for individual needs. For example, the program SSPICE is used for implementation of algorithms for periodic steady-state analysis in SPICE3. To accomodate this need, a command is introduced in STEPS to allow users to integrate CAD tools with STEPS with great ease. One requirement of such program is that the SPICE input file is the input to the program. When this command is invoked, a dialog window is opened to ask users to enter two fields. The first field is the path and name of the executable program and the second field is solely the name of the program. For example, if the user wants to integrate a program into STEPS called hspice, which is contained in the directory /ic3/user/bin/hspice, the user should enter "/ic3/user/bin/hspice" for the first field and "hspice" for the second. This feature allows the user to integrate his own SPICE-compatible simulator in STEPS.

2.9 Help

For the users' convenience, an on-line help manual is available. All schematic-capture editing commands and STEPS commands are presented in menu format for novice users who may need assistance on certain structures and functionalities of STEPS.

3 Structure

3.1 Summary

Because STEPS is part of the OCT framework, it is important to know the way STEPS communicates with the outside world. This chapter introduces the various structures that STEPS utilizes, and how the whole structure ties together. Although the interface is somewhat transparent to the user, the basic structure of STEPS is presented to give an overview of the integration of the various interfaces.

3.2 Introduction to OCT

OCT is a data manager for VLSI CAD applications. It is a major component of the Berkeley CAD framework and has been used at Berkeley and other sites for the several years. [2,3] OCT offers a simple interface for storing information about the various aspects of an evolving chip design.

The basic unit in a design is the cell. This can be as small as a transistor or NAND gate, or as large as the entire floorplan of a CPU. A cell can consist of instances of other cells, such as a NAND gate consisting of several transistors or the floorplan consisting of an ALU, register file, etc.

A cell can have many aspects or views. For example, there can be a schematic view, showing in an abstract way what sub-cells the cell consists of and how they are connected. There can be the symbolic view, where additional information of rough relative placement, sub-cell size and shape, and initial implementation of interconnect might be kept. Furthermore, there is a physical view, where the implementation is fully defined with exact placement and specific geometry. In addition, there are quite different views, such as the simulation view, which might contain the description of the cell in a format that a particular simulator could understand. [3]

OCT provides a mechanism for representing data but places no meaning on the data. Policy is used for assigning meaning to the data represented using OCT. For example, OCT has objects that represent layers and geometry, but it does not specify how the objects are related to the meaning of a geometry implemented on a given layer. The policy states that a geometry that is contained by a layer is implemented on that layer. As another example, OCT has objects that represent nets and terminals, but it does not specify how connectivity is represented. The policy describes how terminals and nets are used to represent connectivity. [3,4]

3.3 STEPS as an OCTtool

STEPS is introduced to the Berkeley CAD framework OCT as one of the so-called Octtools. Since STEPS is a specific tool to enhance schematic capture for circuit designers, STEPS does not support all views of OCT. Only the schematic view is supported by STEPS. Each circuit in STEPS is a cell in OCT. The different 'colors' in STEPS are represented by different 'layers' in OCT. After the schematic capture is finished in STEPS, the ASCII SPICE netlist is generated internally by a program to read OCT database node by node, connection by connection. Finally, the represented information in schematic form is converted into a SPICE input file. Other commands operate similarly, using the OCT database. Fortunately, the users do not need to understand these internal details presented here to use STEPS. Although STEPS is only one out of many programs in OCT, the user can treat STEPS as an independent program working under the OCT environment.

3.4 Remote Procedure Call Package for OCT/VEM

OCT works in a distributed environment of multiple machines and multiple languages. In order for this environment to be successful, OCT uses a remote file system to access the database, allowing the database to be spread out over many machines. This Remote Procedure Call (RPC) package allows user applications to run as separate processes outside of the OCT/VEM address space. The applications make subroutine calls to VEM and OCT as if they were in the same process and address space, similar to tightly bounded VEM commands. The RPC client, that is the RPC code linked with the application program, interprets these calls and passes them to VEM. The RPC server, linked with VEM, calls the appropriate VEM and OCT routine, and returns the results to the RPC client. Having user applications as separate process communicating with VEM via a local-area-network using RPC has the important advantages for different languages, asynchronous, separate processes, portability, and invisibility of the specific applications. [5]

STEPS is one of the applications which uses RPC. VEM argument lists are built in STEPS, and a remote function can then be executed. However, control is immediately returned to STEPS without waiting for the remote command to finish. Therefore, no time is wasted to wait for the procedural calls. STEPS is started using an operating system remote execution function. The application then connects back to VEM via a reliable byte stream, and a list of commands and remote function bindings.

4 ELECTRICAL-RULE CHECKER

4.1 Summary

Electrical design-rule checking can generate constraints that are satisfied by integrated circuits. Since synthesis tools for all but a few types of circuits are still in experimental stage, circuit design, especially analog design, remains an iterative process, with manual intervention by the designer an essential part of each iteration. Violating a design rule does not necessarily mean a circuit is unusable, but careful analysis can reduce error with great amount of CPU time saved in the simulation process. An electrical-rule checker has been included within STEPS in the Berkeley OCT/RPC CAD environment. Integrated circuit designed in the OCT/VEM environment can be automatically checked for design errors by STEPS' electrical-rule checker.

4.2 Background

Circuit simulation, the task of modeling and numerically analyzing the performance of electrical circuits using computers, plays an essential role in the design of present-day integrated circuits. Unfortunately, many input-file errors are very hard to detect once in the simulation stage. To overcome this disadvantage, electrical design rules are designed to test for design errors in the early stage of the design cycle. Problems are spotted early in the design before the simulation stage. Simulation CPU time can then be saved. By pointing out the places where the implicit assumptions of the rules are violated, the rules can focus attention on those parts of the circuit that need more detailed modeling.

Nowadays, schematic-capture with various CAD tools has been designed to allow designers to initialize, analyze, and design their circuits on a workstation instead of on paper. Design rules can be of great help with the utilizing of CAD tools. Electrical circuit simulators are the main CAD tools used by analog designers and the designers of critical digital building blocks. Therefore, electrical design-rule checking is a very important process to reduce the design time by pointing out errors quickly.

OCT/VEM/RPC is the main CAD tool package in the Berkeley CAD environment. The schematic-capture package, STEPS, is available for initial schematic design of integrated circuits.

The rules are summarized below:

1) Loops of Voltage Sources
2) Loops of Inductors

3) Series Connection of Current Sources
4) Shorting of Power Supplies
5) Shorting of Voltage Sources
6) Extremely Small Capacitance
7) Substrate nodes connected to most positive or negative nodes
8) Length of FET's too large
9) No Path from Supplies to Ground
10) Base Resistance must not be zero
11) Connectivity
12) Values on various parameters

4.3 Electrical Design Rules

4.3.1) Loops of Voltage Sources

Loops of voltage sources cause simulation problems. Unfortunately, there is no easy way to go through every loop schematically to check for loops of voltage sources. Figure 4.1.1 is a loop of voltages. Figure 4.1.2 is a loop with two voltage sources and two resistors. Figure 4.1.3 is a loop of five voltages. The most basic algorithm is to check for the positive terminal and negative terminal of every voltage source and see if any equivalent pair is found. If so, a voltage loop is detected and the loop is highlighted through the RPC application of STEPS. Unfortunately, this method does not work in Figure 4.1.3, where a loop of more than three voltage sources is presented. Therefore, an algorithm is proposed to merge all the back-to-back voltage sources until only two are left. Then, the basic algorithm can be used to check whether the voltage sources are in a loop.

Below is a description of the algorithm:

a) Identify all voltage sources one by one (in any order). Identify the positive node and negative node by OCT objectid. If the symbol is a voltage supply or a voltage-to-ground source, give the ground terminal the objectid of zero. Then, every voltage source in the circuit has a positive node and a negative node, with the objectids representing them.

b) Two lists of node identifications are presented. List 1 is used for merging the voltage sources and List 2 is used to check for loop using the basic algorithm described above.

c) For each voltage source, check to see if the pair of positive and negative nodes are equivalent to the pairs saved in List 2. If so, a loop is detected; otherwise, save the pair to List 2. Note that positive and negative terminals are just arbitrary identifications. Therefore, a pair is saved with no pointer to determine whether the terminals are positive or negative. Also, the check is done in pairs to allow negative voltage sources to be detected.

d) Merging of loops of voltage sources is done through List 1. The algorithm is best illustrated through an example. Refer to Figure 4.1.3. The first voltage source to be checked is the source between node 1 and 2. It is saved in List 1. List 1 now has nodes [1 2]. Another voltage source is added from node 2 to ground. List 1 now has nodes [1 2 2 0]. Repeated nodes are deleted in List 1. Therefore, List 1 now has [1 0]. Another source is added from node 3 to ground. Thus, List 1 has [1 0 3 0] and node 0's are deleted since it is repeated. Henceforth, List 1 has [1 3]. Finally, the last voltage source is added from node 1 to 3. [1 3] matches with List 1. The loop is then detected and highlighted.

4.3.2) Loops of Inductors

Loops of inductors also cause problems in simulating electrical circuits. The algorithm used is the same as that of detecting loops of voltage sources. Figure 4.2.1, 4.2.2, and 4.2.3 illustrate three examples, similar to those in Section 4.3.1.

4.3.3) Series Connection of Current Sources

Series connection of current sources are another problem in electrical circuits. The algorithm for detection of series of current sources is quite difficult, since detection is needed for current sources that are in series, but not in parallel. Fortunately, with the algorithm for detection of loops readily available from Section 4.3.1 and 4.3.2, the algorithm to detect series circuits is not that complicated. First of all, if there are terminals of current sources next to each other, a flag is raised. Then, when all the instances in the circuit are checked, the flag is checked again. If the flag is raised, we check to see if the current sources are in a loop. If the current sources are not in a loop, they are in series. Then, the series of current sources is highlighted and warning messages are presented. Figure 4.3.1 is a series connection of two current sources. When the electrical-rule checker of STEPS is run, the series connection of current source is detected and highlighted. On the other hand, even though the terminals of the current sources are next to each other in Figure 4.3.2, the current sources are detected to be in a parallel connection by the algorithm from part 1. Henceforth, we know that the current sources are not in a series connection and therefore no highlighting is made.

4.3.4) Shorting of Power Supplies

Shorting of power supplies creates a series problem. In analog circuits, shorting of power creates unforseen error which is hard to detect in simulation output. In digital circuits, every path in the switch graph from VDD to ground must have at least one open switch in it. This rule generates constraints mainly for CMOS circuits and pre-charged circuits. For CMOS, the power-short constraints are the primary criteria for the well-formedness of logic gates. For pre-charged circuits (NMOS or CMOS), the power-short constraints are usually satisfied by using a clocking discipline. [6, 7, 8]

The algorithm is to inspect every power supply in the circuit and check to see if there exists any direct connection from power supply to ground. For example, in Figure 4.4.1, there is a connection from the power supply to ground. A warning message 'Shorting of Power Detected' is presented if the electrical-rule checker is ran.

4.3.5) Shorting of Voltage Sources

Similar to Rule 4, the shorting of voltage source creates the same problems. The basic explanation and basic algorithm are similar to Rule 4. Figure 4.5.1 is a voltage source with a direct connection to ground. If this example is run in the electrical-rule checker , a warning message is presented.

4.3.6) Extremely Small Capacitance

In analog circuitry, capacitance in the femtofarads unit causes nonconvergence in transient analysis. Unfortunately, convergence errors are the hardest to detect in simulators like SPICE. Therefore, it is desirable to test the circuit schematically before going to the simulators. Warning signals are given in STEPS, and users can choose to change the errors or ignore the messages. In the electrical-rule checker, every capacitance value is checked. If the capacitance is less than 5 femtofarads, a warning message is presented. Figure 4.6.1 is a circuit with the capacitance of 1 femtofarad. If the electrical-rule checker is run, the warning message 'Capacitance value too small -- may cause convergence errors' is popped up with the corresponding capacitor highlighted.

4.3.7) Substrate nodes connected to most positive or negative nodes

In a MOSFET or a JFET, substrate nodes should be connected to the most positive node for n-type channels and the most negative for p-type. The algorithm is as follows: First, all the instances of the circuit are tested and the most positive and most negative nodes are determined. Then, another loop is run to test each MOSFET and each JFET of the circuit. If the n-type substrate is not connected to the most positive node and the p-type substrate is not connected to the most negative one, warning messages are presented. In Figure 4.7.1, the most positive node is node 1, which is the power supply, and the most negative node is node 0, which is the ground. In this specific example, if the NMOS substrate is not connected to node 1, and/or the PMOS substrate is not connected to node 0, warning message will be presented. Consider the enhancement-load inverter circuit in Figure 4.7.2. Without transistor M1 and capacitor C1, the maximum value for the output voltage Vo would be VDD-VT. By adding M1 and C1, the designer ingeniously raises the gate voltage of M2, so that Vo can reach VDD. This dynamic design technique, known as bootstrapping, eliminates the need for an additional supply [9]. However, in the schematic shown, the substrate of M1 is tied to its source. As a result, the substrate-drain junction of M1 becomes forward biased limiting the voltage rise of Node 3 and, hence, the output swing, as shown in Figure 4.7.3a. From the rule defined above, one can check that the substrate node is not connected to the most negative node and a warning message is presented. The user can then connect the substrate of M1 to ground and the desired output swing is obtained, as illustrated in Figure 4.7.3b.

4.3.8) Length of FET's too large

If the length of MOSFET or JFET is too large, nonconvergence may result in the simulators. Lengths of JFET and MOSFET are tested in the electrical-rule checker, and values of more than 100 lambda's (with a 5 micron/lambda process) are detected and warned. Figure 4.8.1 is an example where the W/L of the PMOS is 132/131. If the electrical-rule checker is run, a warning message 'MOSFET length too large - may cause nonconvergence' is presented. Also, the corresponding MOSFET or JFET is highlighted.

4.3.9) No Path from Supplies to Ground

Path from supplies to ground must exist. Therefore, a test is made on whether a path exists schematically. First of all, the circuit is tested to see if the voltage supply and the ground are both presented. If not, a warning message is given. Then, the node number of the supply is identified. For example, in Figure 4.9.1, node 1 is the supply node. All the nodes of the circuit are identified by instances. Since there are three instances (excluding supplies and grounds) in this circuit, the nodes are identifed as [1,2] (resistor), [2,3,0] (npn), and [3,0] (voltage source). The algorithm is as follows: At the top, the supply node (1) is detected. Then, the node is cancelled to see if node 0 is found. If not, a path from supplies to ground is missing. In Example 9.1, the resistor has nodes [1,2]. With the supply node of [1], node 1 is repeated and cancelled in the global list. So, node

[2] is in our global list. Then, nodes [2,3,0] are in the individual node list. Henceforth, node 2 is cancelled, and node [3,0] are in the global list. Once the node 0 is in our global list, a path is detected from supply to ground and no warning message is presented.

4.3.10) Base Resistance must not be zero

The base resistance (rb) of a bipolar transistor in the circuit is checked against zero since a zero rb causes problems in analog circuits. For example, Figure 4.10.1 is a bipolar blocking relaxation oscillator. When SPICE is run, an error in transient analysis occurs. Figure 4.10.2a shows the partial simulation results up to the time-step error at t=1.87us. This error is due to the zero internal rb of the bipolar transistors. If an rb of 100ohms is added, the problem is corrected. The electrical-rule checker in STEPS detects every bipolor transistor with rb=0 and an error message is included. The algorithm is as follows: First we identify the internal bag and properties of OCT where the parametes are located. Then, we check for the parameter string. If rb=0 is detected or simply no rb is present, a warning message pops up in STEPS. [11]

4.3.11) Connectivity The connectivity of the schematic circuits is a very important factor to be checked since it is quite hard to see if a circuit is actually connected from the display without the recognization of the computer program. The connectivity should be checked by the design-rule checking. The connectivity is checked as follows: Every instance of the circuit is checked one by one. If there is any isolated node, (i.e. node that is not connected with any other instances), a warning message is displayed. Note that the connectivity verification algorithm designed by Spickelmier [10] cannot be used here, since a schematic is compared to the layout to make sure that they are the same in connectivity verification, and the layout is unavailable. The only algorithm that one can use is to check that every node in the circuit is not an isolated node. Figure 4.11.1 is a circuit which looks perfectly valid in both the paper display and under the X-Window. Unfortunately, one of the terminals is actually unconnected.

4.3.12) Values on various parameters

All values of the schematics are checked against illegal values. For example, the length and width of MOSFET, values of resistors, capacitors, and inductors are not allowed to be negative. Other values are checked and details are omitted here.

4.4 Rules that are not implemented in the Electrical-Rule Checker

4.4.1) Avoid parallel pullups

No signal vertex may be simultaneously connected to two different pulled-up nodes and to ground. This rule is intended to make the usual simple pullup over pulldown ratio calculations accurate. The constraints generated by this rule for an array of the two-port dynamic RAM cells can be satisfied by plaining some restrictions on RAM usage:

A) Simultaneous reads and writes on the same bus are prohibited.
B) Writing into a cell from both buses simultaneously is prohibited.
C) Reading the same cell from both buses simultaneously is prohibited when the storage transistor is on.

Although the last restraint seems an unnatural restriction, the storage pull-down must be wider than usual if both buses read from it at the same time, since the saturation currents of the two pullups are added. [6]

Unfortunately, a signal vertex cannot be detected in OCT. Furthermore, parallel pullups are allowed in analog circuits. Henceforth, this rule is not implemented in the electrical-rule checker.

4.4.2) Avoid gates in the middle of pulldown chains

If the gate of a transistor is connected to a pulldown tree, the connection should be through the pulled-up node, not through a lower node in the tree. For CMOS, a gate connected to either an NMOS pulldown tree or a PMOS pullup tree should be connected through the node where the two trees meet.

Unfortunately, the PASS and CLEAR and pulldown chains are impossible to be detected alone. If user input is required, the purpose of the electrical- rule checker is defeated, since the user has to know that it is actually a pull-down chain. [6]

4.4.3) Avoid charge-sharing

A signal that is used on the gate of a transistor must be either isolated from all other nodes (storage charge) or connected to VDD, ground, or a pulled-up node. When two nodes isolated from power become connected to each other, the charge on the nodes is shared between them. If the nodes initially have different values, the result may be an illegal intermediate voltage. Static storage nodes (nodes on even cycles in the inverter graph) also must be isolated to avoid charge-sharing.

In a charge-sharing circuit, the input is stored at node STORE when LOAD is high during phase 1. LOAD may go high before Theta 1, sharing the charge between STORE and X. If the LOAD and READ signals are simultaneously high, the illegal value can be propagated to the output.

Again, due to the impossibility of determining out what the STORE, LOAD, and the clock Theta 1 is, this rule cannot be proposed to the electrical-rule checker.

5 PERFORMANCE

5.1 Summary A brief analysis of the performance of STEPS is presented in this chapter. Both the speed and the code of the program are inspected. The time performance of the electrical-rule checker is also analyzed independently. The time performance of the electrical-rule checker is analyzed rule-by-rule. The result is that the time performance is very efficient in the electrical-rule checker.

5.2 Performance of the Electrical-Rule Checker The CPU time is individually calculated. Each rule discussed in Section 4 has different CPU time. The CPU runtime were obtained on a DECStation 3100 with 24 Megabytes of physical memory running ULTRIX 4.1. The CPU time is estimated by using the number of objects and dividing by the time range.

Rule 1:	2,200 Objects/Second
Rule 2:	2,200 Objects/Second
Rule 3:	1,850 Objects/Second
Rule 4:	4,330 Objects/Second
Rule 5:	4,330 Objects/Second
Rule 6:	32,680 Objects/Second
Rule 7:	7,500 Objects/Second
Rule 8:	30,890 Objects/Second
Rule 9:	5,260 Objects/Second
Rule 10:	24,680 Objects/Second
Rule 11:	2,850 Objects/Second
Rule 12:	Not evaluated

The electrical-rule checker procedure in STEPS currently has 2614 lines of C code, with RPC applications.

5.3 Performance of STEPS STEPS is written in the language C under the OCT/RPC environment. There are approximately 23,000 lines of C code. The run-time performance of STEPS is directly dependent on the performance on the OCT/VEM/RPC performance, with the independent STEPS runtime less than 10% of the runtime on the VEM graphics environment.

6 OVERVIEW OF COMMERCIAL SCHEMATICS CAD TOOLS

6.1 Summary

Because of the availability of graphics window systems in advanced workstations, many circuit designers design their circuits using schematic-entry programs. Several are now available commercially. A brief comparision of two popular commercial schematic-entry programs, TekTronix's QuicKic, and Viewlogic's Workview with STEPS is described below:

6.2 Tektronix's QuicKic

6.2.1 Background

QuicKic is a package produced by Tektronix, Inc. [12] QuicKic is a menu-driven schematic and layout editor. From schematics, QuicKic can generate Tekspice and Spice netlists. In this section, only the schematic editor is included.

6.2.2 Display

An example display of a circuit in QuicKic is given in Figure 6.2.1 and Figure 6.2.2. Differences between a QuicKic display and a STEPS display are not obvious. All model values, model names, and node values are presented in both QuicKic and STEPS. Several differences can be observed, though. First, QuicKic does not display the area size of the transistor, but STEPS offers this option for the user. With STEPS, the user can choose whether or not to display the area size. Furthermore, with STEPS, the user can choose to display only the area size with an area different from the nominal value. Secondly, in QuicKic, the VCC and VEE power supplies are displayed. This information is not crucial to the circuit since one can easily look at the nodes to determine such information. Furthermore, STEPS allows the user to control the display. This means that the user can select to turn off one or more of the model names, model values, or node names. QuicKic does not allow this feature.

6.2.3 Placing Elements in a Schematic

All the basic commands of a schematic editor (place elements, remove elements, etc) are presented in both STEPS and QuicKic. QuicKic has the advantage of containing a large database of all devices. As a result, the users can select a device model with model values displayed as part of the model names. For example, when a user wants to select a resistor with 4k ohms, he can choose a model called R4000 and place it in the display. In STEPS, the user has to choose a generic resistor and change the value to 4k ohms afterwards. On the other hand, this feature also imposes inconveniences to the user. If the user has a circuit which has only one model value, he can hasten his schematic entry. If he needs various models with different values on his circuit, the searching of database plus the placement from the database will delay the process of schematic entry. One good advantage of QuicKic over STEPS is that a node which is opened is represented as a cross in QuicKic. Once the node is wired, the node becomes a dot in QuicKic. An example is given on Figure 6.2.2 for reference. This gives an easy way for the user to distinguish between connected node and disconnected nodes. In STEPS, disconnected nodes have to be detected by the electrical-rule checker.

Another difference is that QuicKic requires the user to enter a node for the substrate. For example, if a bipolar transistor is requested, the user is asked to enter the node for the substrate. This is inconvenient for several reasons. First of all, it is very generic that this node is the most positive or the most negative node. It is automatically placed this way in STEPS unless changes are requested from the user. Secondly, the user has to know the assigned node name from the model. For example, the user needs to enter the VEE node in order to connect the node to VEE. This may lead to problems such as incorrect entry of node names. In STEPS, the substrate nodes are automatically assigned. If the user wants to use the substrate nodes at other locations, he can do that graphically by connecting a wire rather than entering the node names. This may lead to less error when the user has the chance to check his circuit through the schematic diagram graphically.

6.2.4 Wiring the Schematic

The wiring procedure is quite different between STEPS and QuicKic. Using STEPS, the user has to select one node and then move the mouse to another when a connection is designed. QuicKic uses a different procedure. The user has to select the first node, then the second. After selecting both nodes, the user has to select the second node again to terminate the wire. As the wire is placed, a diamond-shaped marker is displayed at the most recent vertex (the second node). If a connection is made to another wire, a connect point is drawn. A wire is terminated by entering the last node twice. In contrast, STEPS connects two nodes wire-by-wire, which means that the user connects two points in two procedures. In QuicKic, the user can actually connect the wire for one node in three procedures. Which editor presents a better solution is a matter of taste. STEPS appears simplier to use, but QuicKic is more efficient for experienced users.

6.2.5 Subcircuit

Global connection of nodes is allowed in both STEPS and QuicKic. For example, the VCC and VEE nodes can be selected to become global connections no matter what kind of hierarchy is involved. QuicKic presents a better display of these global nodes since all global connections are represented by arrows in QuicKic. STEPS does not have this feature. The definition of subcircuit is very similar in STEPS and QuicKic. All the input and output terminals have to be labelled in both editors. The procedures to create a circuit from a subcircuit are also very similar. In QuicKic, the subcircuit has to be defined as a subcircuit so that one may place it as an element in the circuit. In STEPS, the hierarchy is much less strict due to the OCT structure. This means that one does not need to define a circuit as a subcircuit in STEPS. The hierarchy structure can be switched and defined as the user desires.

6.2.6 SPICE Netlist Generation

In QuicKic, a 'TSPICE' command in the simulation menu is used to generate a spice netlist. All the models are generated, but all the commands have to be entered from a text editor. For example, if a '.print' command is requested, it has to be entered through a text editor by the user. One of the goals of STEPS is to allow immediate simulation at the ending of the schematic-capture process, which means that no command needs to be entered through the text editor. Obviously, QuicKic has a different goal. In STEPS, all the commands (.print is one of them) can be entered through menus in the schematic-capture process. One obvious advantage is that a user can design the circuit through schematic diagrams and simulate it without any intermediate process. In QuicKic, if many runs of simulation are needed, the user needs to use the text editor to enter commands between the initial schematic-capture process and the simulation process. This may be very cumbersome and redundant.

6.2.7 Other Differences

1) STEPS allows key bindings which QuicKic does not allow. This feature is an OCT/VEM function to allow a user to select short keys to represent a command. Experienced user will find key bindings to be very useful since selecting from menu may be a very cumbersome process for a large circuit.

2) Instead of fully automatic menu-driving, the user of QuicKic needs to enter most of the commands through text-entry when asked. In STEPS, most commands can be entered in one key stroke or by menu selection. This may create a faster design process for the user.

3) STEPS allows the user to present his own commands for a different SPICE version, and QuicKic allows only TSPICE to be used. Obviously, as a commercial product, the allowance on only one SPICE is an understandable feature, but as a public domain product like STEPS, the more flexibity is presented, the better the product is.

4) As a commercial product, QuicKic has a very large database to select from, which means that many commercial products can be picked directly without going through the schematic-editor process. STEPS obviously lacks this feature.

5) Unlike STEPS, QuicKic does not interact with the simulation process. This means that the schematic circuit is not checked against any rule violations. STEPS, as mentioned earlier, presents an electrical-rule checker to check for schematic rules violation.

6) STEPS has two commands, "chng-local-parm" and "chng-global-parm", to allow the users to change the model parameters locallly or globally. In QuicKic, the user has to change the model parameters model-by-model.

7) In STEPS, the user can generate a SPICE input file with a voltage or current source added at the selected node, usually in the input, to test the circuit. These are the "add-curr-source" and "add-volt-source" commands in STEPS. QuicKic does not offer a similar presentation.

8) Dialog boxes are not presented in QuicKic when pertinent information is needed. STEPS allows dialog boxes so that required inputs which the user forgets to enter is directed by the dialog boxes.

6.2.8 Conclusion

As a conclusion, both schematic editors are very similar in structure. As a commercial product, QuicKic emphasizes the whole integrity of the software. This disallows certain features which STEPS is able to present. Speedwise, the processing time of QuicKic is less than that of STEPS, although not very significantly.

6.3 Viewlogic's Workview

6.3.1 Background

Workview 4.0 with Viewdraw from Viewlogic is examined and compared with STEPS in this section. Viewlogic has a simulation facility, called Viewsim, for simulation purposes. Furthermore, the tutorials of Workview and Viewsim have different sections for analog and digital designs, which made them easy to use for a particular design purpose.

6.3.2 Display

Workview supports a very easy-to-use display with all the menu options on the left side of the display (Figure 6.3.1). The display structure is very similar to that of STEPS, since all model names, model values, etc are supported in the display. Zooming and panning on the screen are very similar between Workview and OCT/VEM/STEPS. The ViewIn and ViewOut function keys on the display of Workview, which STEPS does not support, are useful when the object to be magnified is in the middle of the screen. The labeling function of STEPS and Workview is very similar. Workview allows a better automatic labelling structure. For example, the user can type in 'A[1:10]' to name the ten nodes selected to become A1, A2, A3, ... A10. This feature is not supported by STEPS. Similar to STEPS, Workview also presents dialog boxes to ask the users for various information. For example, the symbol value and the component value (model name and model value respectively in STEPS) are requested by the dialog box structure, as in STEPS.

6.3.3 Placing Elements in a Schematic

When a user intends to add a component in Workview, a prompt appears at the bottom of the screen saying "Specify component position". The user can then select a position using the mouse of the workstation. Similar to STEPS, the user is also allowed to add a title or text on the screen. Furthermore, the text font or text size can also be entered, as in the OCT/VEM/STEPS structure. Different from STEPS and similar to QuicKic, Workview supports a huge library of models and components. This is very advantageous compared to STEPS, since the user can easily find a commercial product without having to redraw it. The 'copy' function of Workview is very similar to that of STEPS, with the three procedures to select the original component, select the location, and copy the component. Workview has the command 'Reflect' to reflect the component right away. On the other hand, STEPS allows relection through the 'translate 90 degrees' command. Henceforth, if many reflections are needed, Workview provides a more efficient routine here. Fortunately, it is very rare that many reflections are needed.

6.3.4 Wiring the Schematic

Similar to QuicKic and different from STEPS, Workview presents a small square (a cross is presented in QuicKic) for a "dangling net." The circuit is much easier to read with the "dangling net" distinguished from a connected node. Both STEPS and Workview allow the user to copy the whole net instead of component-by-component. A disadvantage in Workview is that when a wire is added, the user needs to type in the command 'Add Net'. STEPS bypasses this by key bindings, which is more efficient and convenient. Moving a component without disrupting the wiring is a major advantage in Workview versus STEPS. Unfortunately, it is difficult to change the OCT/VEM/STEPS structure to present this offering to the user.

6.3.5 Subcircuit

Viewdraw, similar to OCT/VEM/STEPS, supports an unlimited number of hierarchical levels. Subcircuits are defined the same as in OCT/VEM/STEPS. The two hierarchical structures (Workview's and OCT's) are strikingly similar, with no significant discrepancies found.

6.3.6 SPICE Netlist Generation

A wirelist program in Workview interprets the design levels for input to the simulator. A composite-type component indicates to the wirelist program that it should trace down the block hierarchy. Wirelisting links multiple sheets and brings together the hierarchy of a schematic. Viewdraw supports a number of different SPICE netlist formats - the Viewsim (.vsm files) format, the SPICE (.cir files) format, the PSPICE (.cir files) format, and the HSPICE (.cir files) format. An "Export Wirelist" command in Workview takes all the wirelist description files produced by the "ExportCheck" command and produces a connectivity description file. Henceforth, Workview also has the connectivity checks which the electrical-rule checker in STEPS provides.

There are two types of wirelists structures in Workview -- flattened wirelists and hierarchical wirelists. Flattened wirelists (connectivity descriptions) are created from a hierarchical design by expanding and removing the intermediate levels in the design. A flattened wirelist contains only instances of primitive components and their interconnections. The values of global signals, parameterized attributes, and expression evaluation are usually resolved during the generation of the wirelist. This structure is very similar to the STEPS netlist generation command.

A hierarchical wirelist creates a connectivity description file that retains the hierarchical structure of the design. A hierarchical wirelist typically contains a number of composite blocks that correspond to the original design tree. This structure is very similar to the OCT datatrace design manager - VOV - designed and implemented by Casotto at U.C. Berkeley.

Similar to QuicKic and other commercial schematic-capture design packages, Workview does not allow command like ".print" to be entered in the schematic-capture process. Workview allows such commands to be entered only in the simulation package which they present.

6.3.7 Other Differences

1) STEPS allows the user to present his own commands for a different SPICE version which Workview does not allow.

2) As a commercial product, Workview has a very large database to select from, which means that many commercial products can be picked directly without going through the schematic editor process. STEPS obviously lacks this feature.

3) In STEPS, the user can generate a SPICE input file with a voltage or current source added at the selected node, usually in the input, to test the circuit. These are the "add-curr-source" and "add-volt-source" commands in STEPS. Workview, similar to QuicKic, does not offer similar presentation.

6.3.8 Conclusion

Workview and OCT/VEM/STEPS are very similar in structure. Workview provides many of the same commands which STEPS presents. As a schematic editor, Workview is an excellent package as a front end for simulation.

7 CONCLUSION

STEPS is now released through the Berkeley CAD framework - OCTTOOLS. Circuit designers can design the schematic diagrams in the STEPS graphics environment. Simulation can also be done using various versions of SPICE. Almost no text editing is needed since all SPICE information can be entered through schematic or dialog boxes in STEPS. All SPICE analysis can be done directly in STEPS, and various postprocessors are integrated inside STEPS to take full advantages of the graphical postprocessors available. Furthermore, an internal filter is constructed to allow conversion from SPICE2 output data format to XGRAPH. Therefore, by pointing at the appropriate nodes, the user can directly obtain a graphical output of dc, ac, distortion, noise, fourier, or transient analyses. This interface emulates the functionality of an oscilloscope.

In this report, the functionality of STEPS is presented. The features mentioned include the structure, editing power, netlist generation, display, printouts, control cards and simulation within STEPS. As in all the softwore programs, the full power of a tool can only be recognized by experiences. This report can only serve as an introduction to the program itself.

Chapter 3 is devoted to emphasize the structure of STEPS. Since STEPS is actually a part of the OCT/VEM/RPC framework developed in Berkeley, the structure is quite complicated and a full explanation is needed. It is suggested that the OCT manuals are fully explored for users who have interest about the structure of STEPS.

An internal on-line electrical-rule checker is also included in STEPS. It is written to check for errors in schematic diagrams so that users can directly be informed schematically if any circuit rules are violated while using STEPS. This topology checker for schematic circuits is introduced to check for certain degeneracies in a circuit. For example, if only one branch is connected to a node, or if there is no DC path from a node to ground, or if there are loops of voltage sources and/or inductors, the electrical-rule checker in STEPS will point out the errors immediately through highlighting in the schematic diagrams. CPU time and user's design time are thus minimized by these early checks on the errors.

STEPS has been compared with two commercial schematic-capture packages, Tektronix's QuicKic and Viewlogic's Workview. The advantages and disadvantages of the three products are discussed.

REFERENCES

[1] D. Harrison, P. Moore, Rick. L. Spickelmier, and A. R. Newton, "Data Management and Graphics Editing in the Berkeley Design Environment," IEEE Trans. ICCAD '86, Santa Clara, CA.

[2] Rick Spickelmier, editor. "OctTools Distribution 3.0," Memorandum No. UCB/ERL M89/56, UC Berkeley Electronics Research Laboratory, Berkeley, California, 1989.

[3] Rick Spickelmier, Peter Moore, A. Richard Newton, "A Progammer's Guide to Oct," Technical Report UCB/ERL M90, UC Berkeley Electronics Research Laboratory, Berkeley, California, August 15, 1990.

[4] Rick Spickelmier, Jeff L. Burns, and A. Richard Newton. Policy Guides for Oct. Technical Report UCB/ERL M90, UC Berkeley Electronics Research Laboratory, Berkeley, California, January 1990.

[5] Rick Spickelmier, A. Richard Newton, "A User's and Programmer's Guide to RPC: Remote Procedure Package for OCT/VEM," Technical Report UCB/ERL M90, UC Berkeley Electronics Research Laboratory, Berkeley, California, February 28, 1990.

[6] Kevin Karplus. "Exclusion constraints for digital mos circuits: A new set of electrical design rules" IC-CAD, IEEE Trans. ICCAD 1985, 244-246

[7] Kevin Karplus. "A Formal Model for MOS Clocking Disciplines", Cornell Computer Science Technical Report 84-632, Cornell University, August 1984.

[8] David Cooke Noice. "A Clocking Discipline for Two-phase Digital Integrated Circuits", Stanford PhD Thesis, January 1983. University Microfilms 8314482

[9] D. Hodges and H. Jackson, **Analysis and Design of Digital Integrated Circuits**, Second Edition, McGraw-Hill, 1988.

[10] R L Spickelmier and A.R. Newton, "Connectivity Verification Using a Rule-Based Approach", IEEE Trans. ICCAD 1985, 190-192

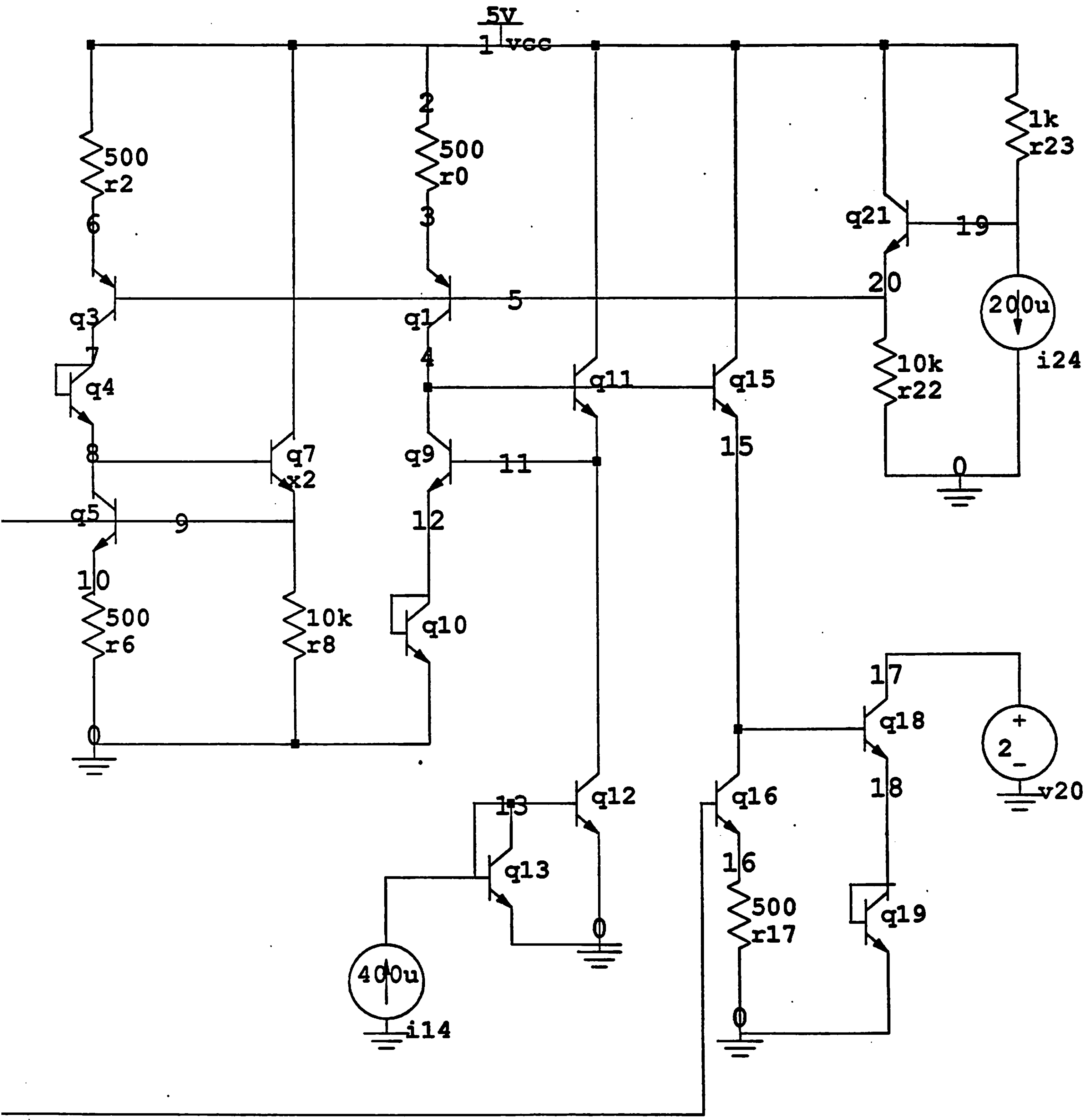
[11] Theologos Kelessoglou, "NECTAR, A Knowledge-Based Framework for Analog Circuit Verification", Memorandum No. UCB/ERL M89/40, UC Berkeley Electronics Research Laboratory, Berkeley, California, 1989.

[12] "QuicKic Graphic Editor Reference Manual", Version 7B, Tektronix, Inc., Beaverton, OR, March 1990.

[13] "WorkView 4.0 Reference Manual", Viewlogic Systems, Inc., Marlboro, Massachusetts.

FIGURES

Lists of Figures

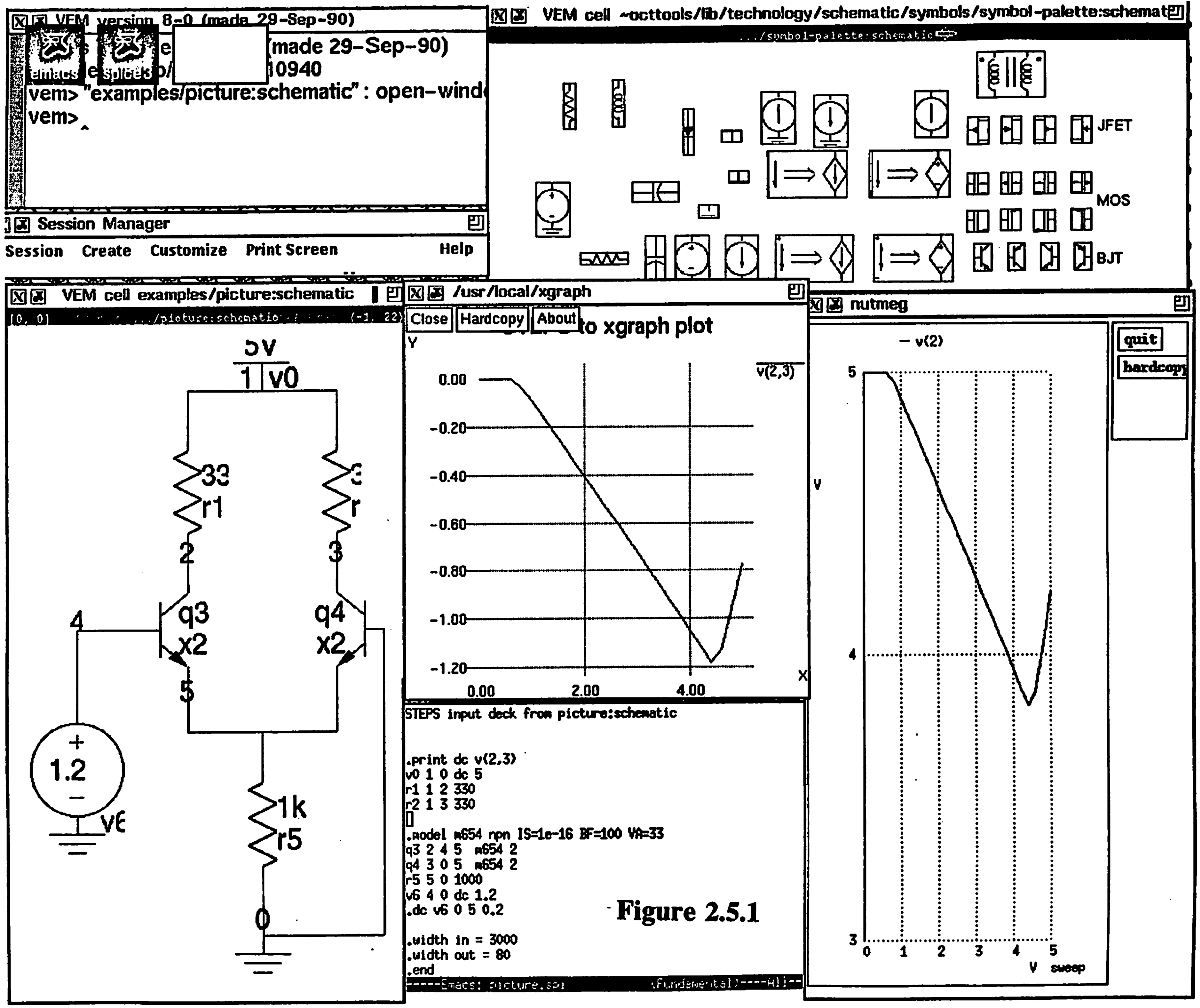


5V
1 vcc
2
500
r0
3
500
r2
6
1k
r23
q21
19
20
q3
q1
5
200u
4
q4
q11
q15
10k
r22
i24
8
q7
x2
q9
11
15
0
q5
9
12
10
500
r6
10k
r8
q10
17
q18
0
q12
13
q16
2
v20
18
16
q13
500
r17
q19
0
400u
i14
0

Figure 2.4.1

```
STEPS input deck from demo:schematic

vcc 1 0 dc 5
r0 2 3 500

.model m1218 pnp IS=1e-16 BF=100 VA=34
q1 4 5 3 m1218 1
r2 1 6 500
q3 7 5 6 m1218 1

.model m988 npn IS=1e-16 BF=100 VA=33
q4 7 7 8 m988 1
q5 8 9 10 m988 1
r6 10 0 500
q7 1 8 9  m988 2
r8 9 0 10000
q9 4 11 12 m988 1
q10 12 12 0 m988 1
q11 1 4 11 m988 1
q12 11 13 0 m988 1
q13 13 13 0 m988 1
q14 1 4 14 m988 1
q15 14 9 15 m988 1
r16 15 0 500
q17 16 14 17 m988 1
q18 17 17 0 m988 1
v19 16 0 dc 2
.dc v19 1 10. .5
q20 1 18 19 m988 1
r21 19 0 10000
r22 1 18 1000
i23 18 0 dc 0.0002
.tf v(8) i23
i24 0 13 dc 0.0004
.tf v(8) i24

.width in = 3000
.width out = 80
.end
```

Figure 2.4.2



```
vem> "examples/picture:schematic" : open-wind
vem>
```

```
STEPS input deck from picture:schematic

.print dc v(2,3)
v0 1 0 dc 5
r1 1 2 330
r2 1 3 330

.model m654 npn IS=1e-16 BF=100 VA=33
q3 2 4 5  m654 2
q4 3 0 5  m654 2
r5 5 0 1000
v6 4 0 dc 1.2
.dc v6 0 5 0.2

.width in = 3000
.width out = 80
.end
----Emacs: picture.spi          (Fundamental)----All--
```

Figure 2.5.1

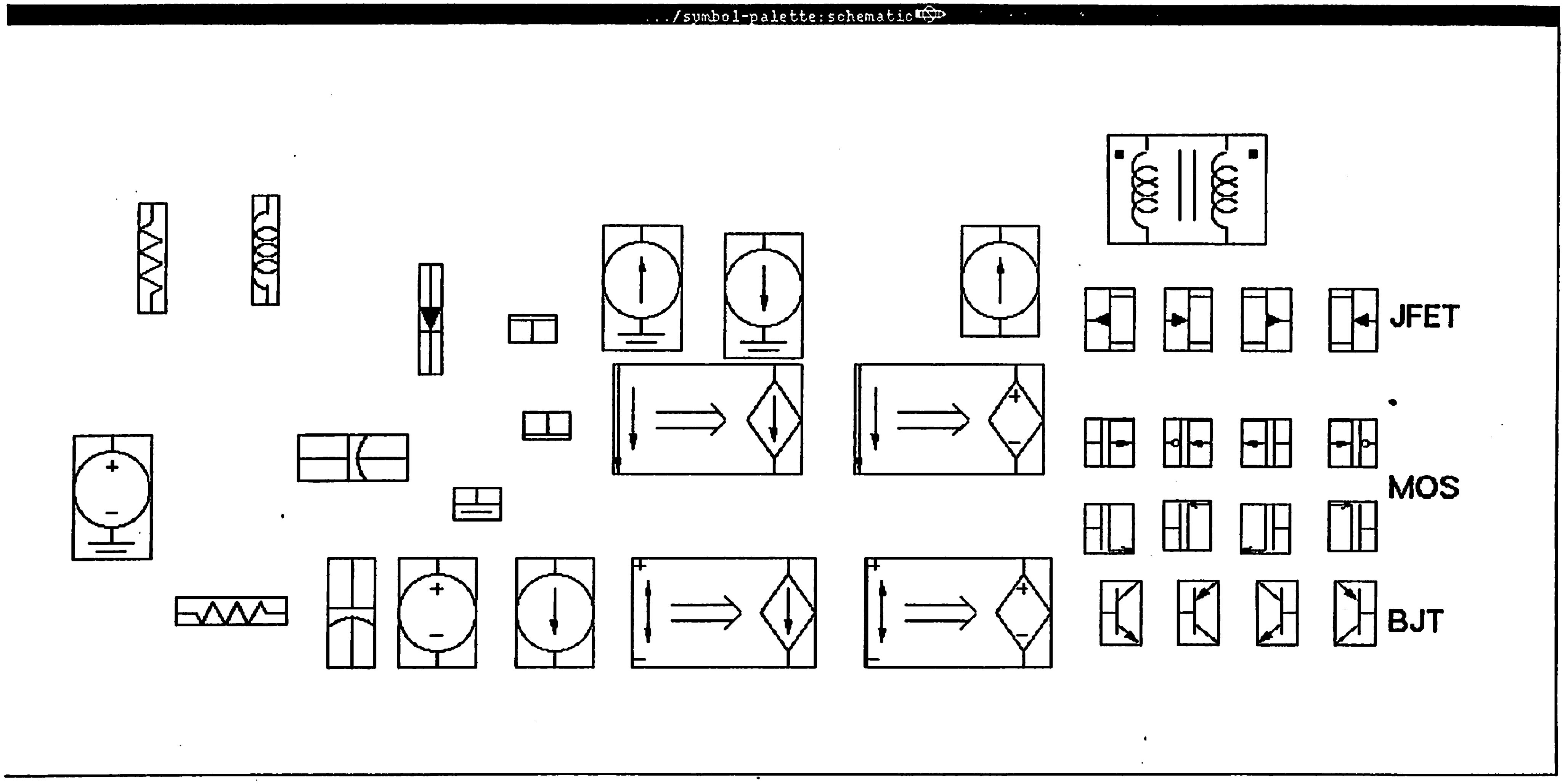


Figure 2.5.2

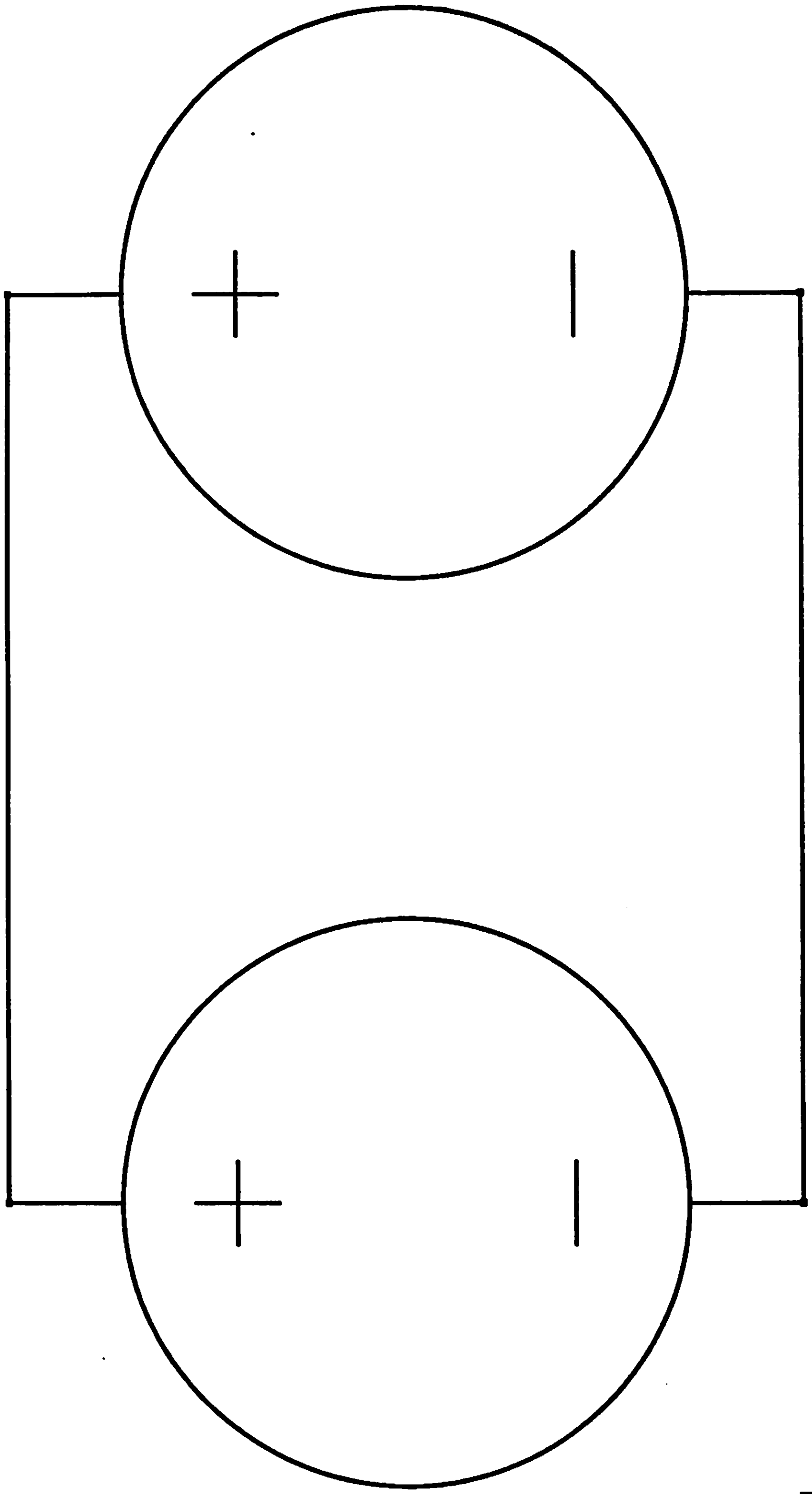


Figure 4.1.1

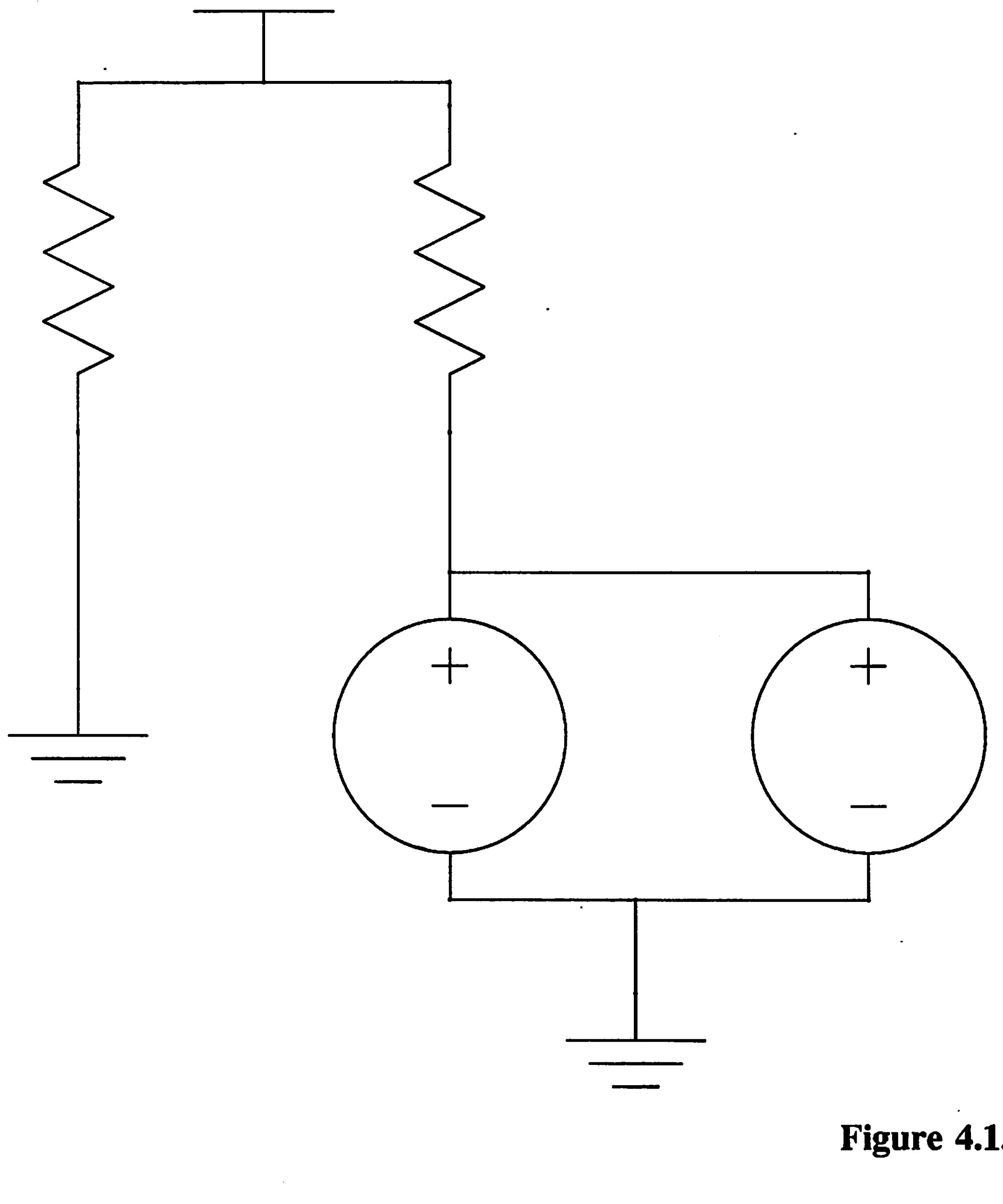


Figure 4.1.2

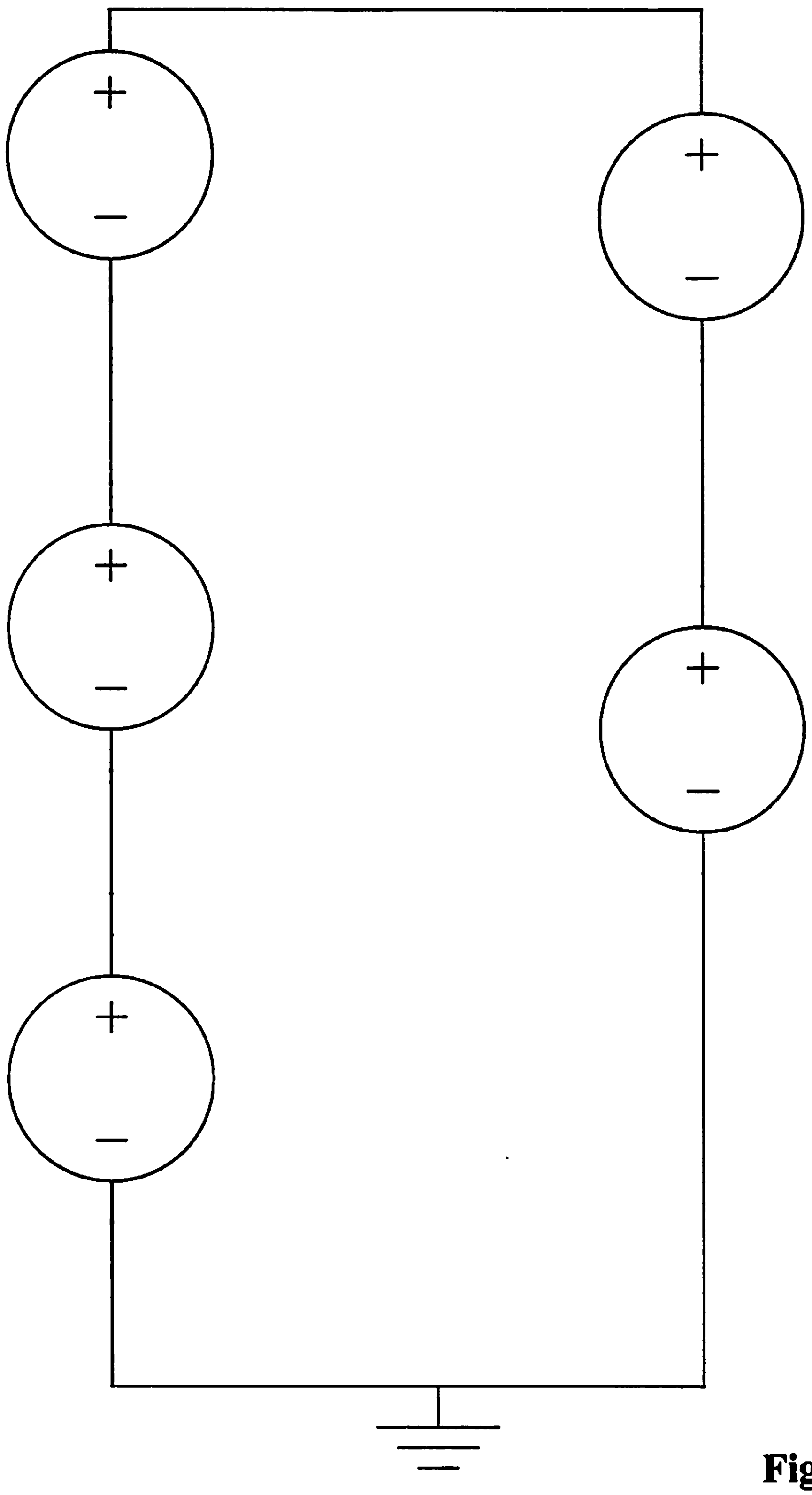


Figure 4.1.3

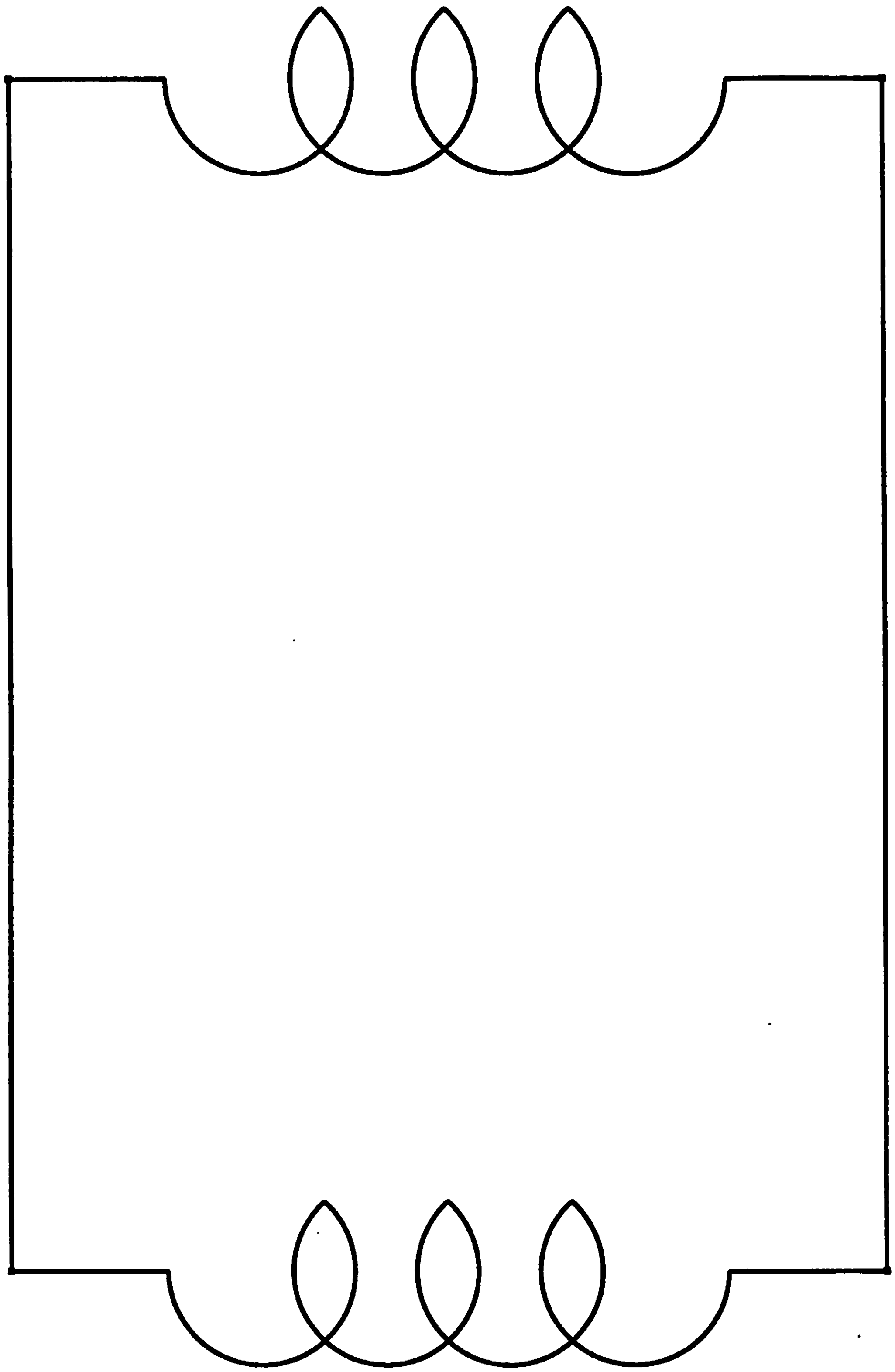


Figure 4.2.1

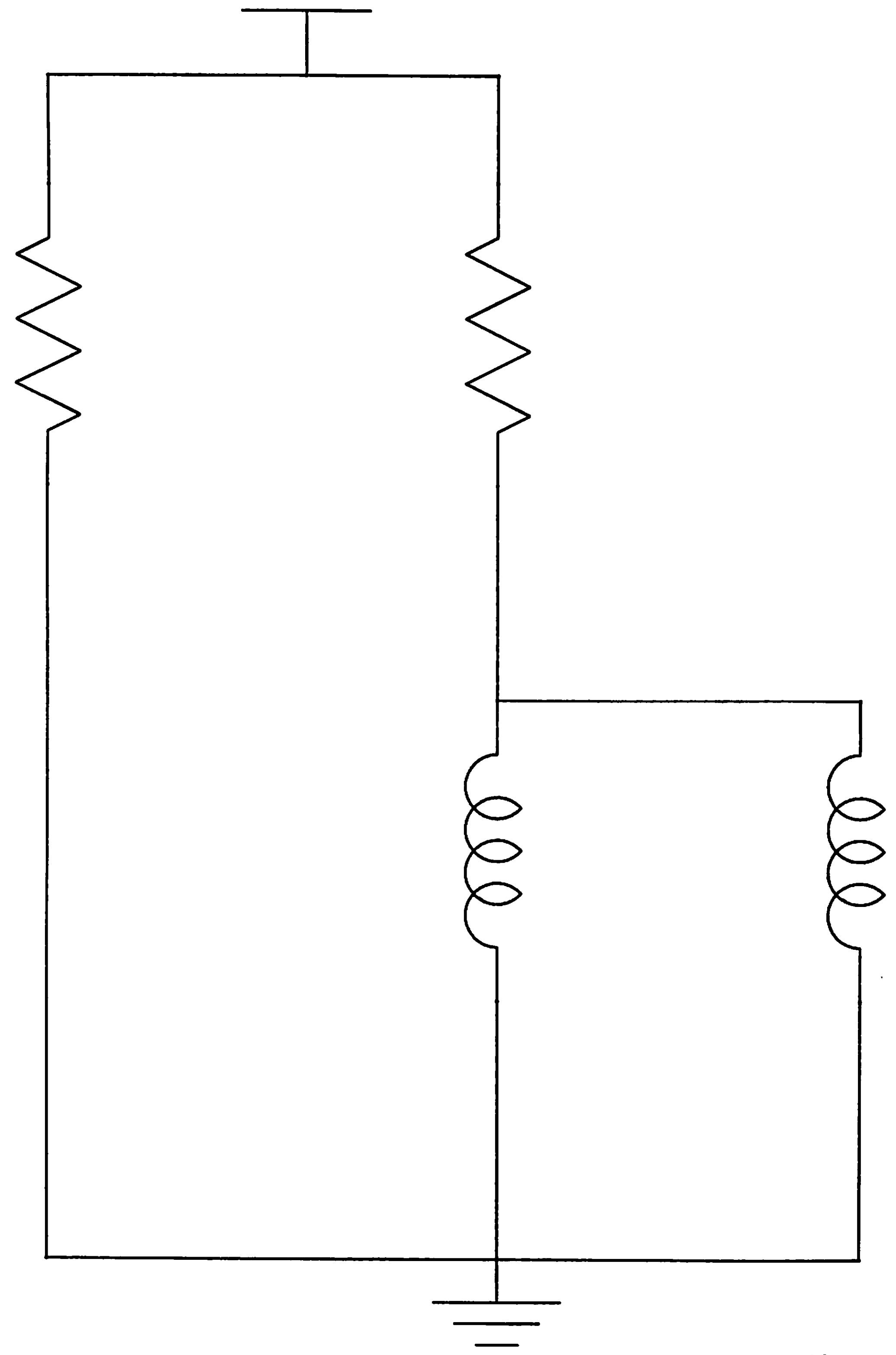


Figure 4.2.2

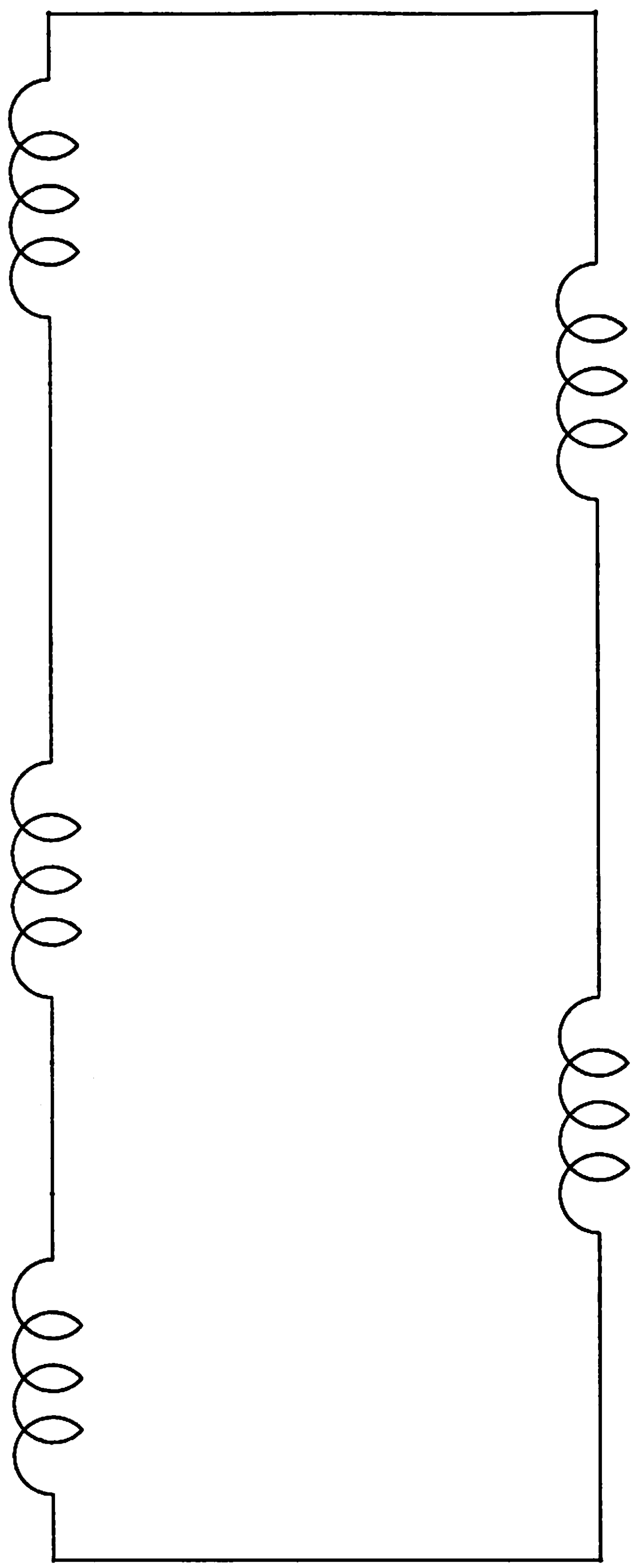


Figure 4.2.3

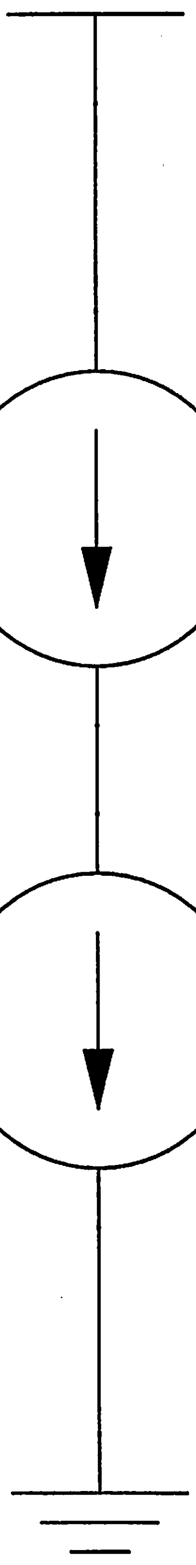


Figure 4.3.1

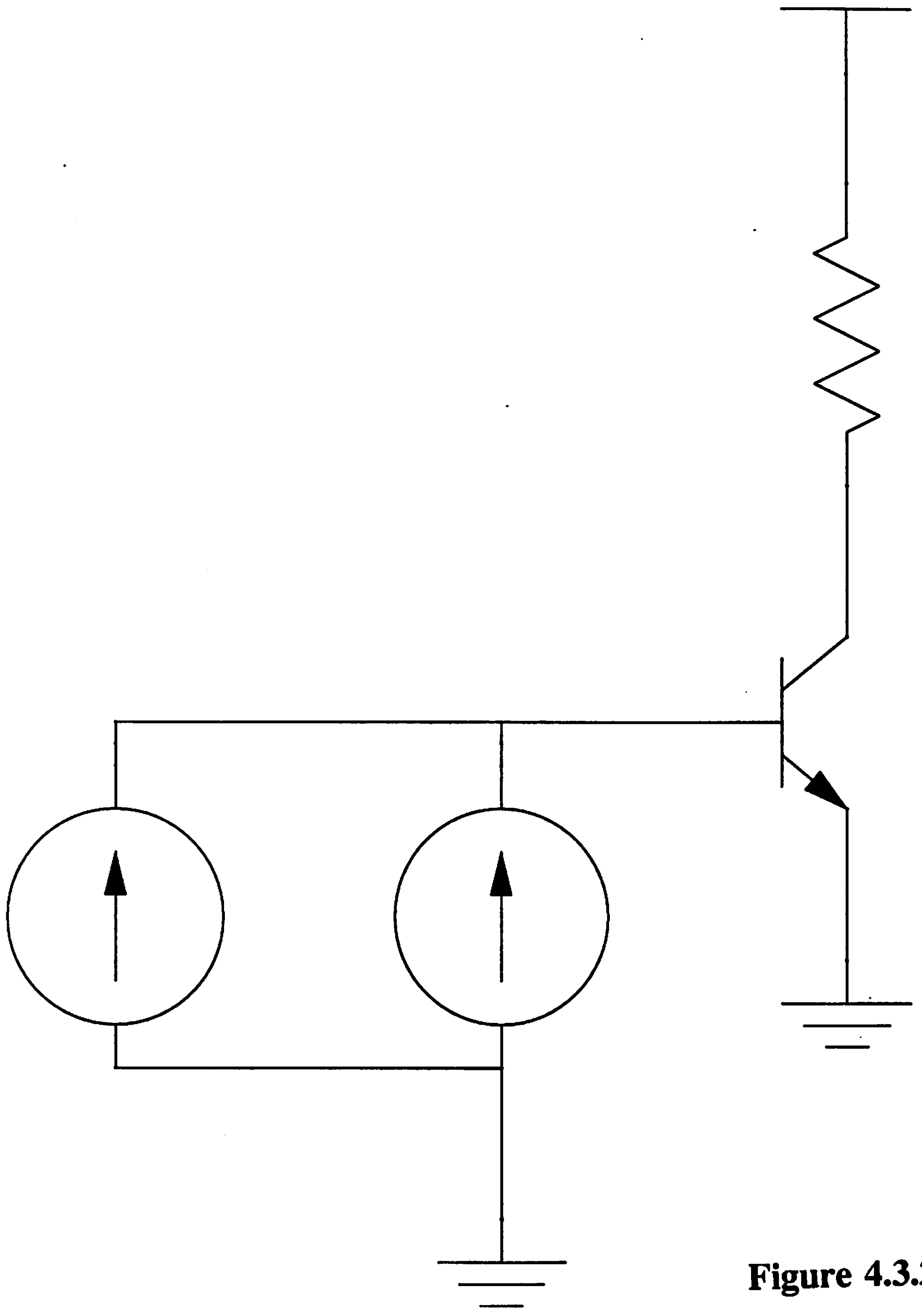


Figure 4.3.2

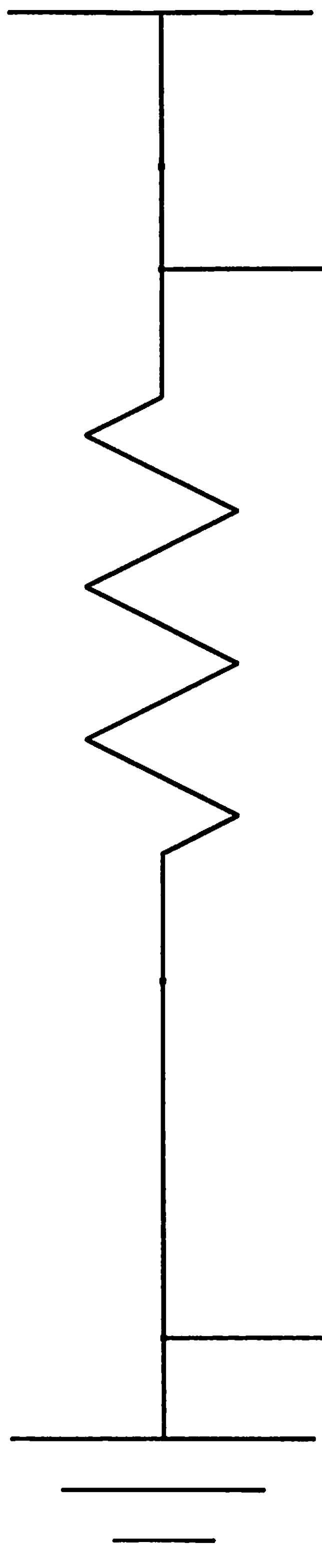


Figure 4.4.1

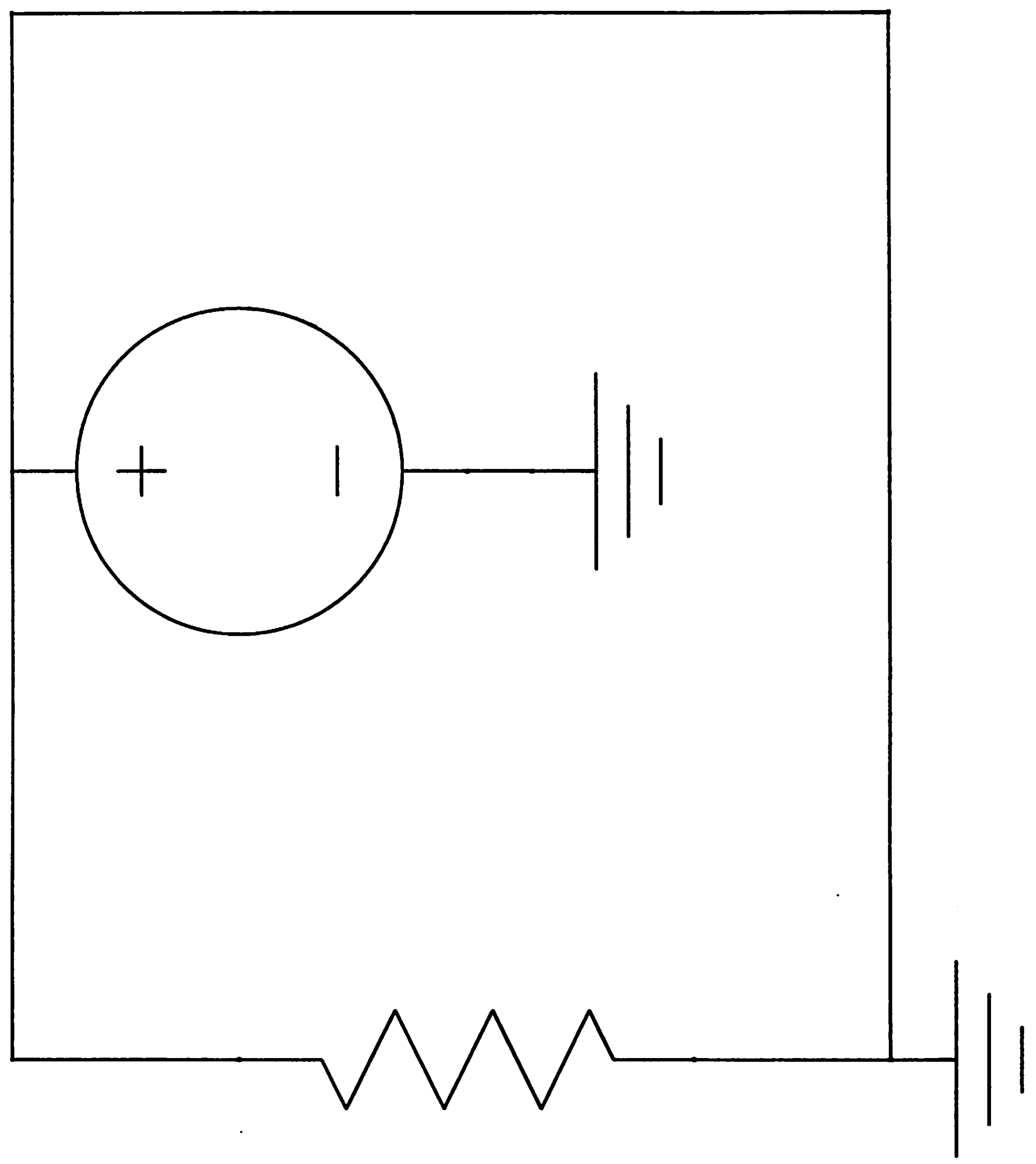


Figure 4.5.1

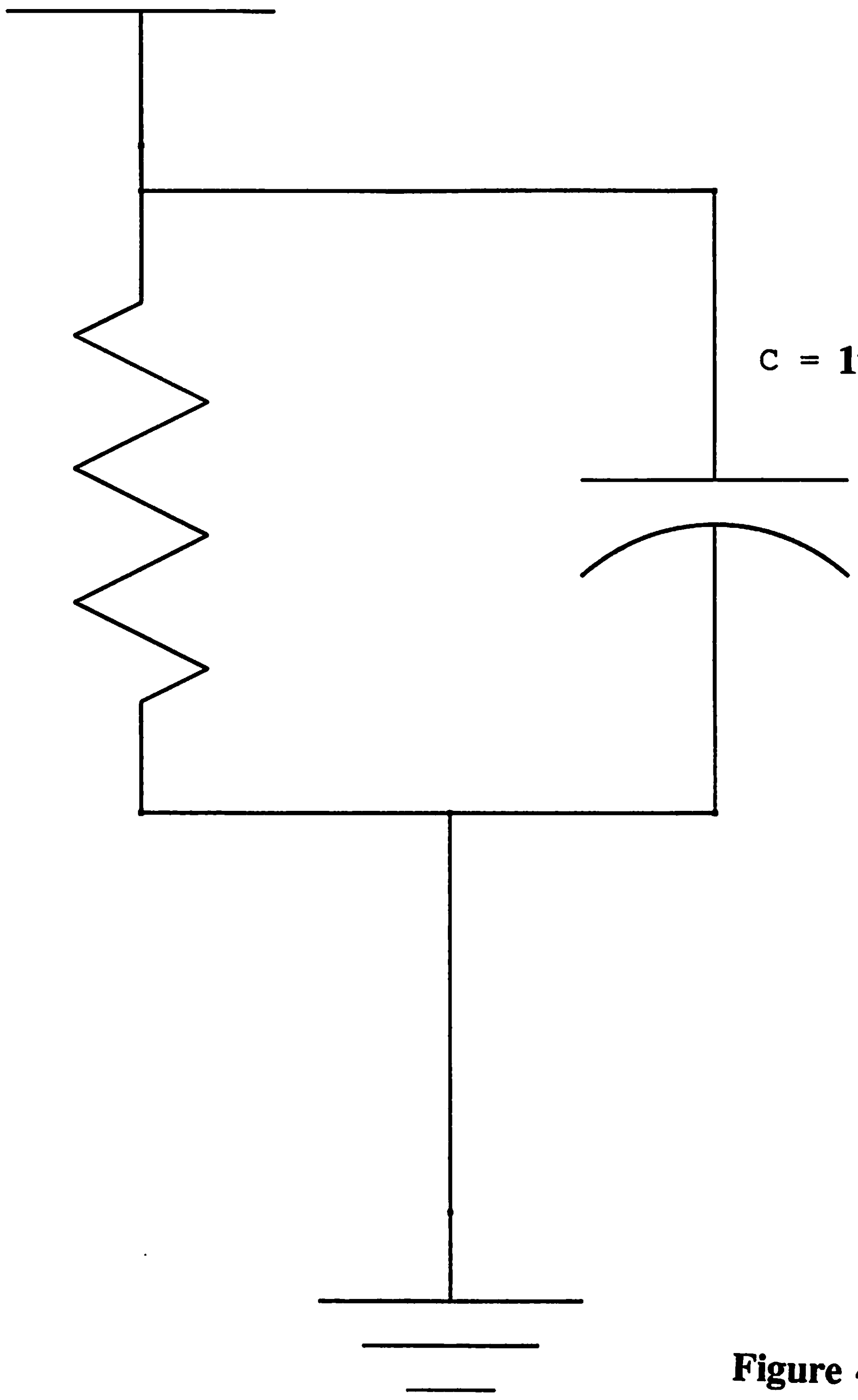


Figure 4.6.1

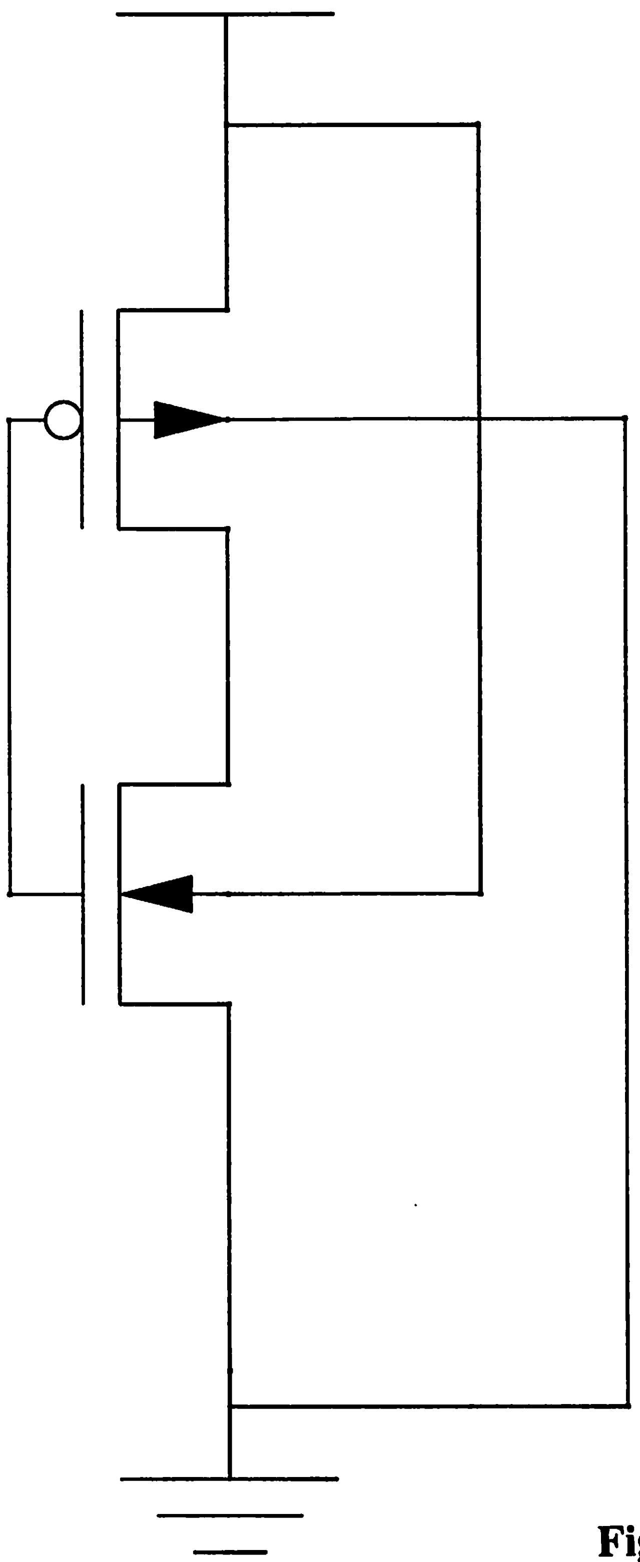


Figure 4.7.1

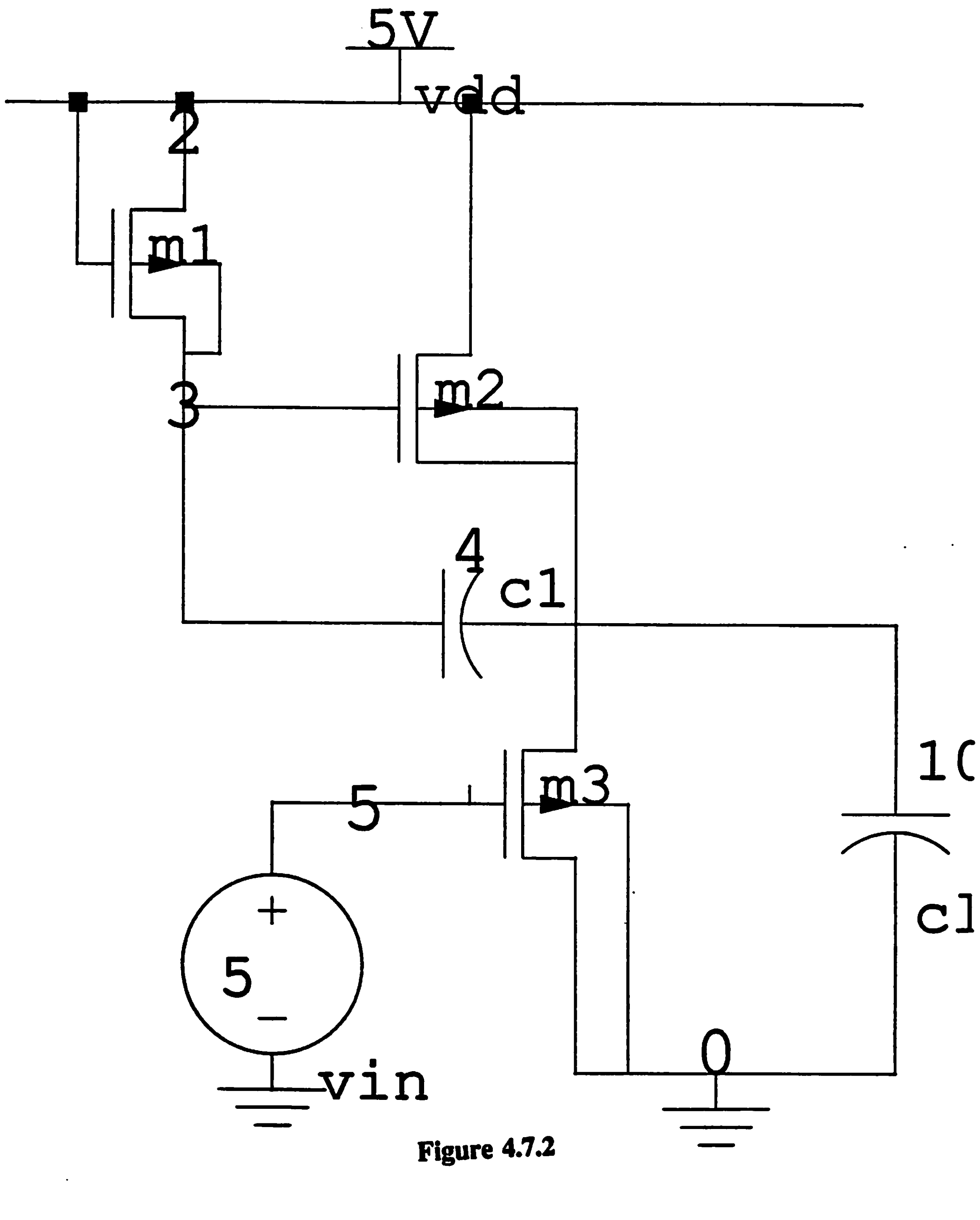
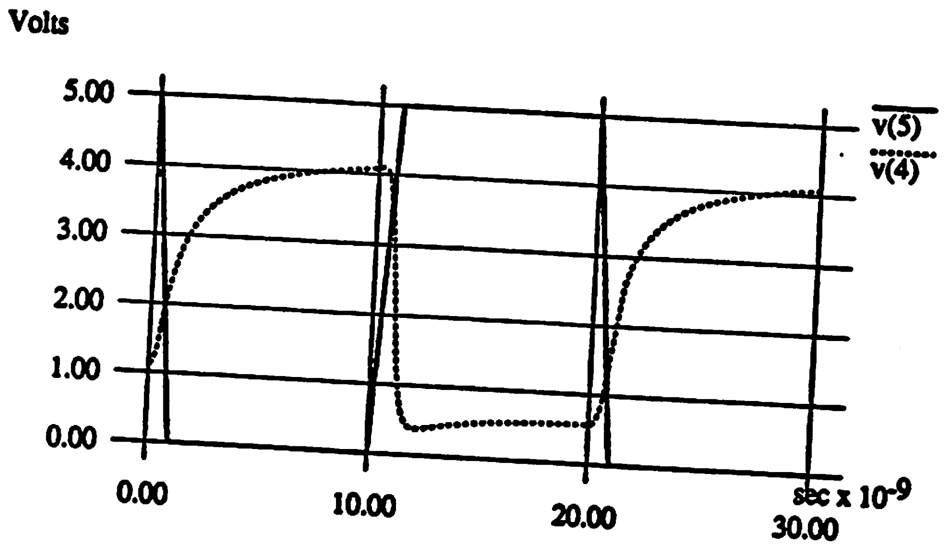
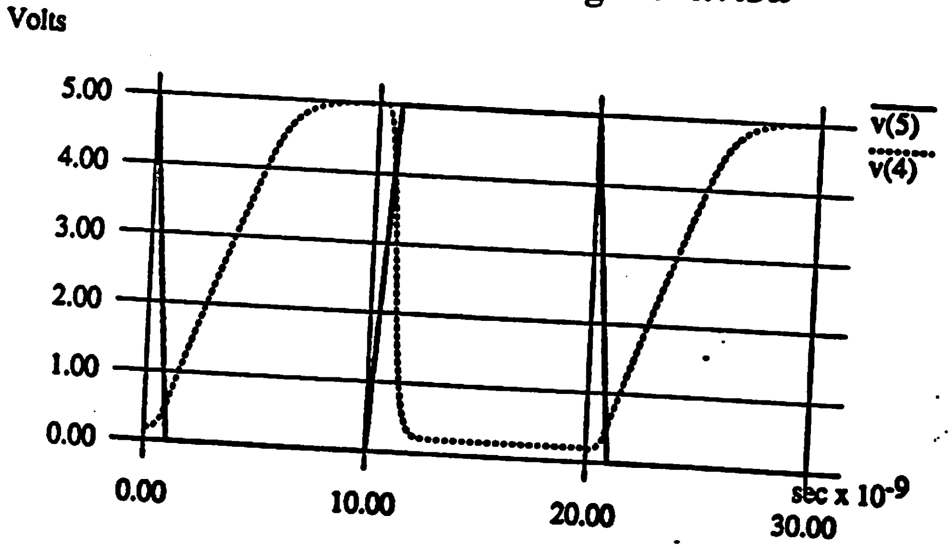


Figure 4.7.2



(a)

Figure 4.7.3a



(b)

Figure 4.7.3b

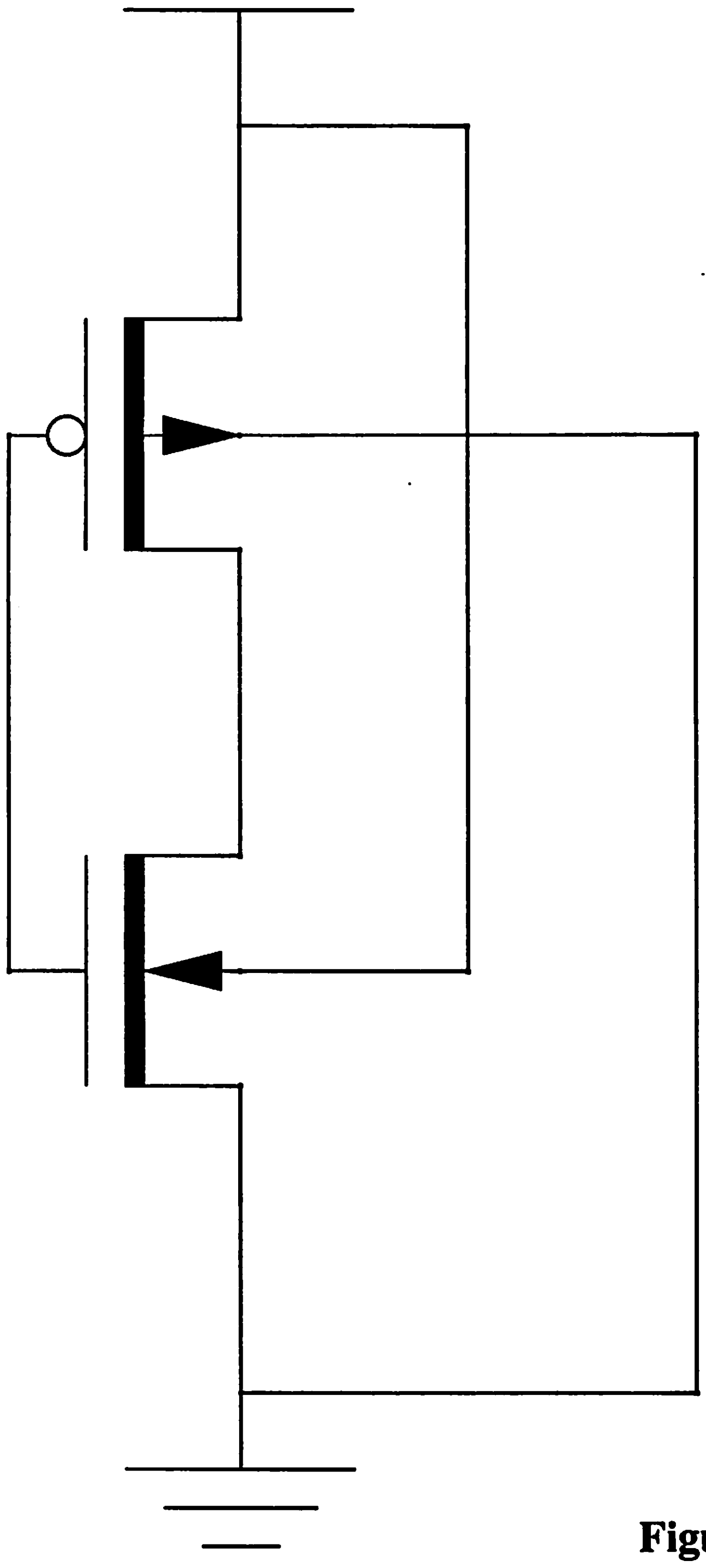


Figure 4.8.1

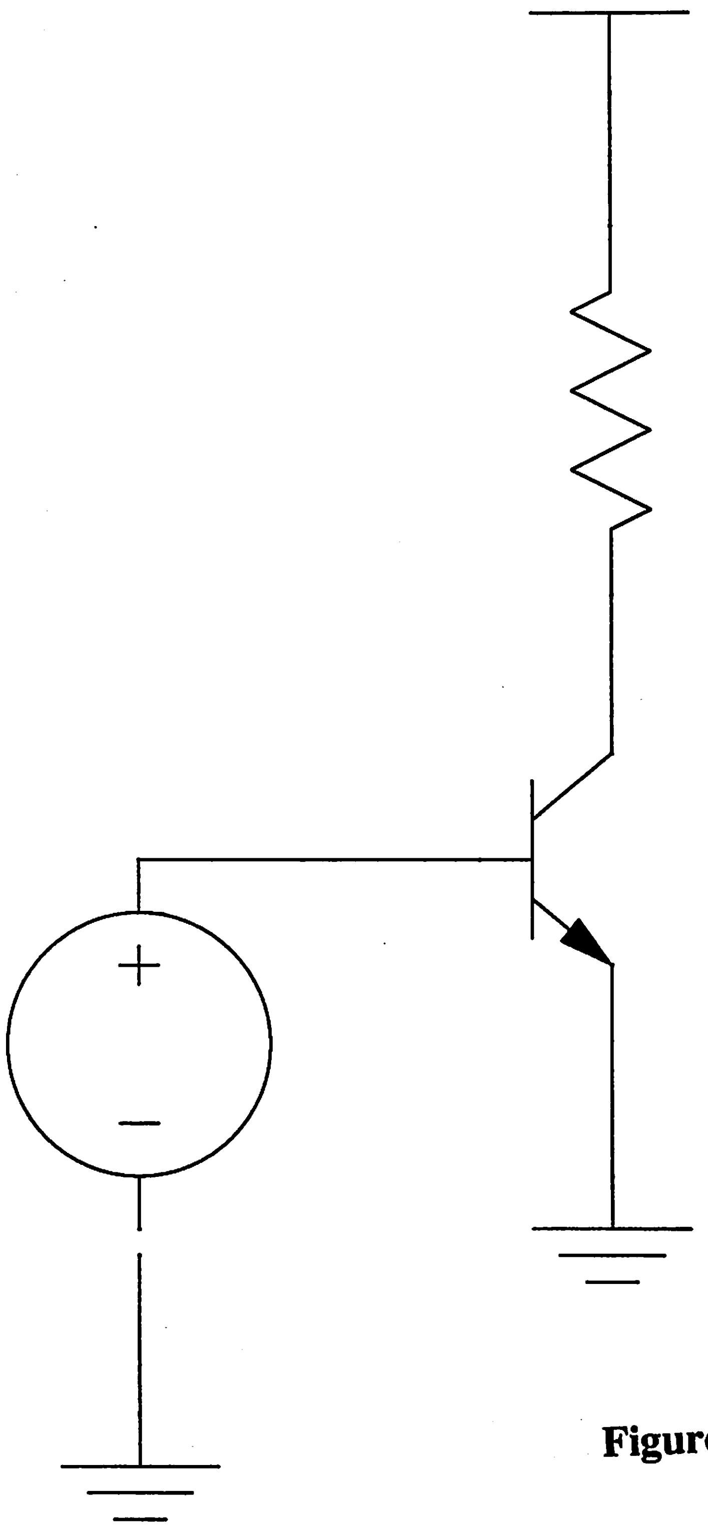


Figure 4.9.1

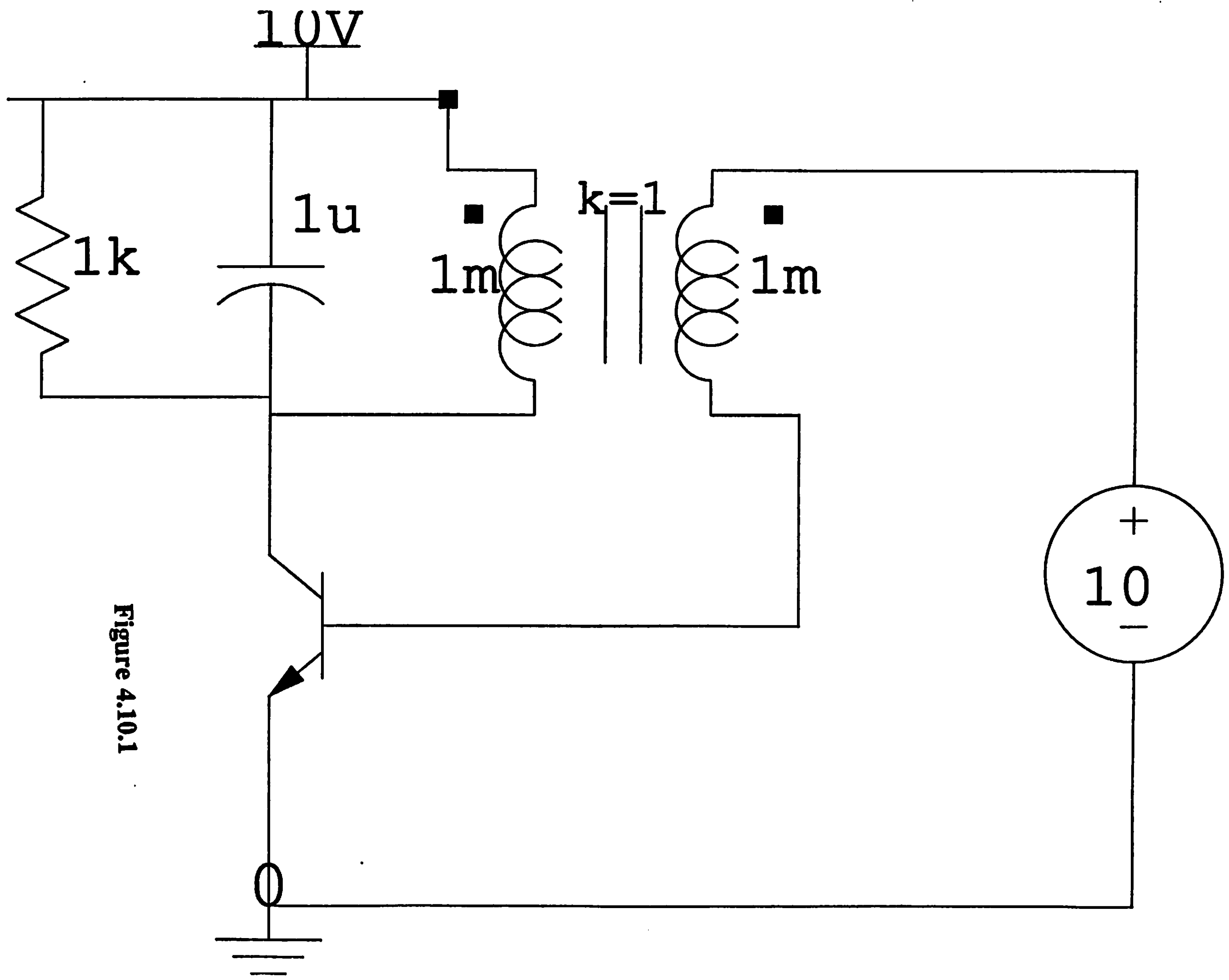


Figure 4.10.1

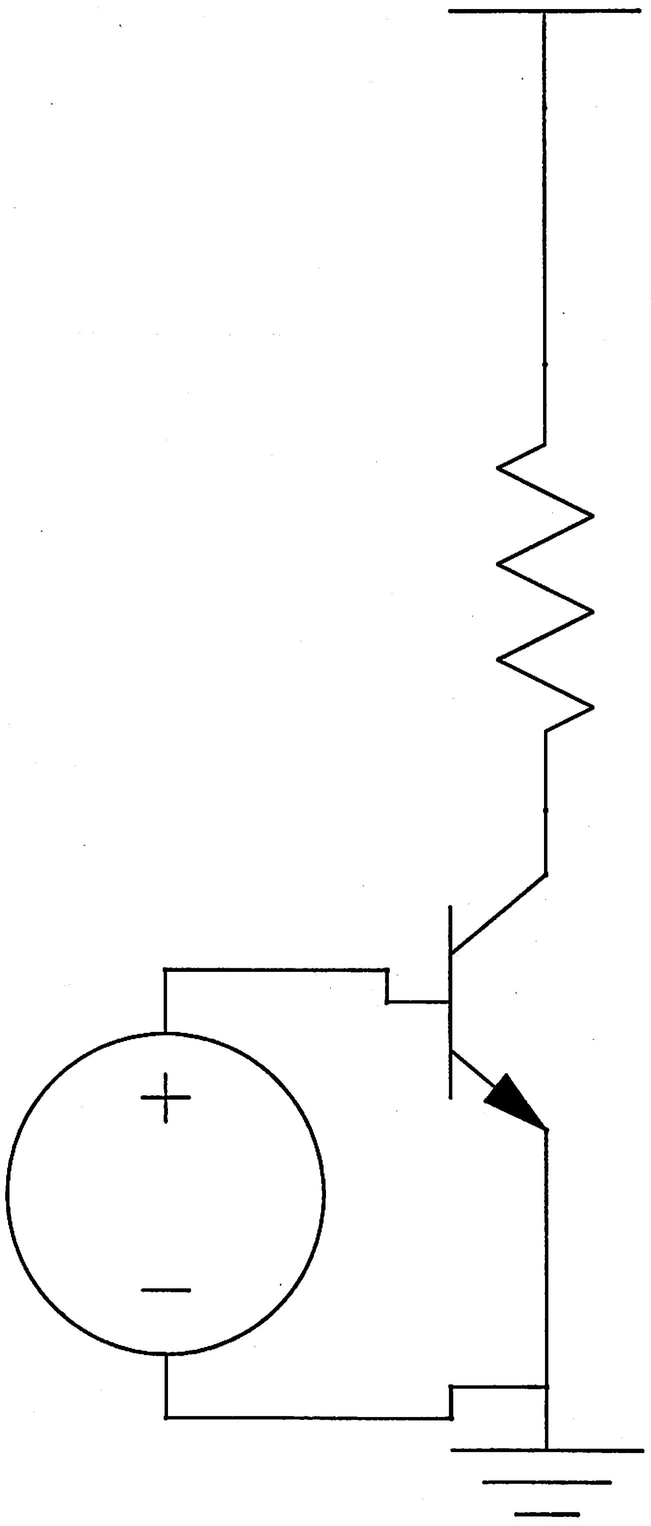


Figure 4.11.1

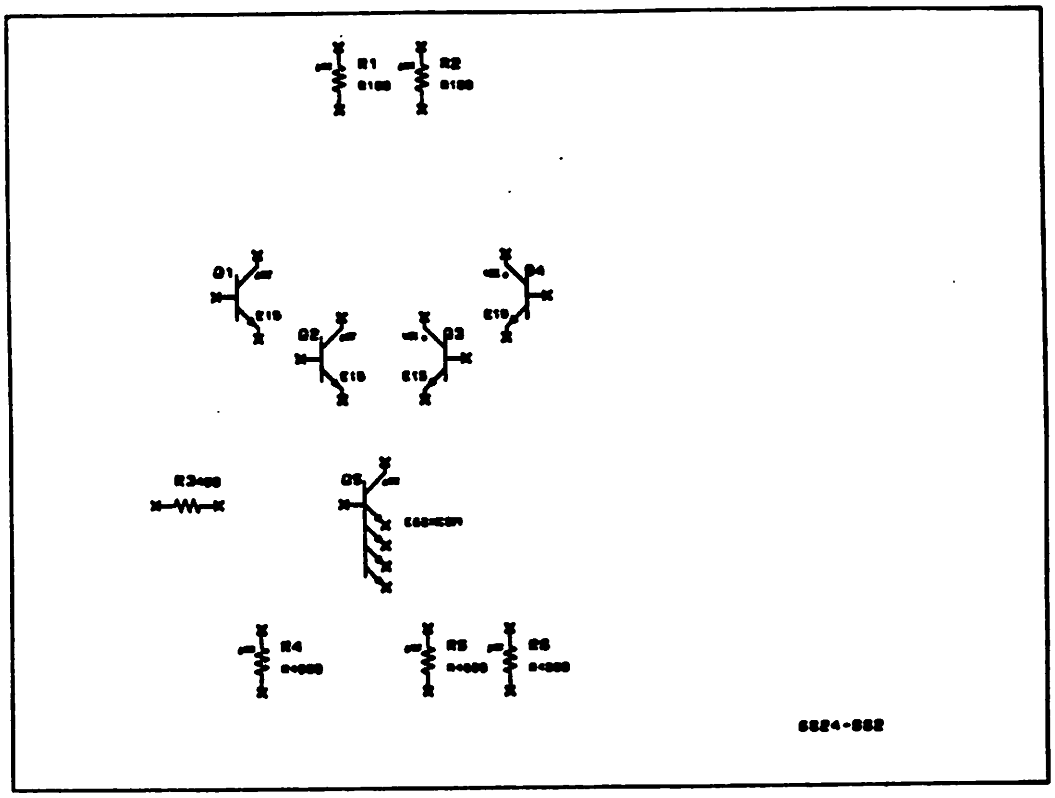


Figure 6.2.1

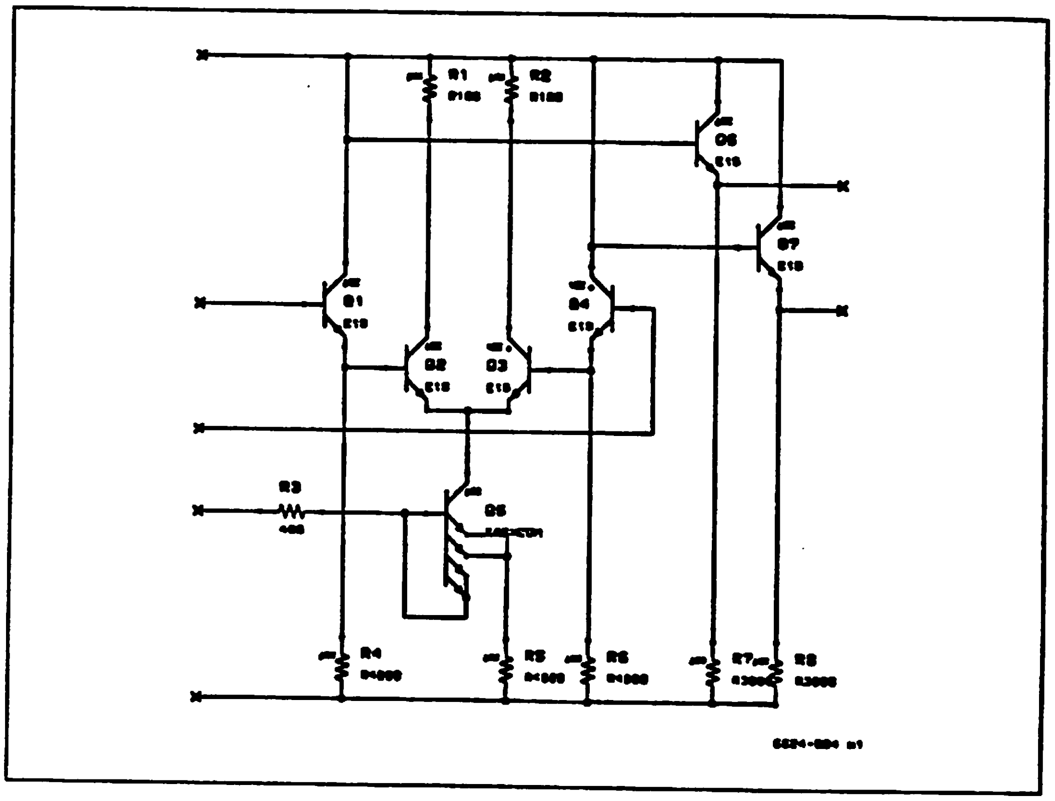

R1
R100
R2
R100
Q6
E10
Q7
E10
Q1
E10
Q4
E10
Q2
E10
Q3
E10
R3
400
Q5
R4
R4000
R5
R4000
R6
R4000
R7
R3000
R8
R3000

Figure 6.2.2

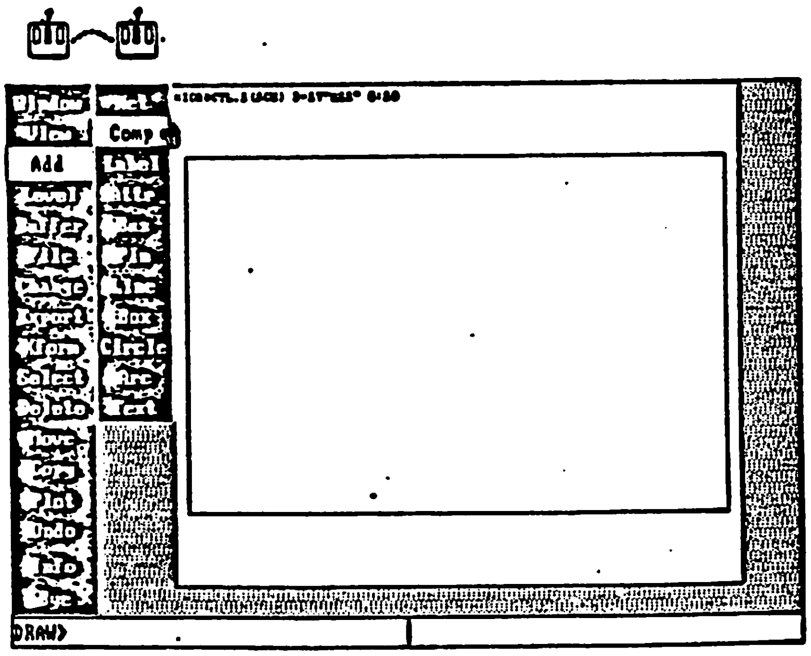
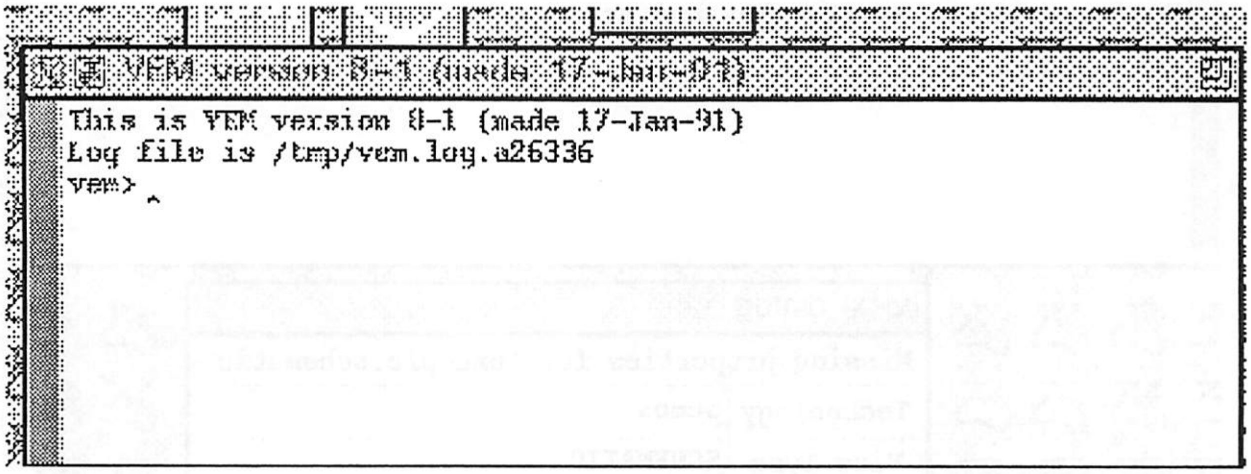
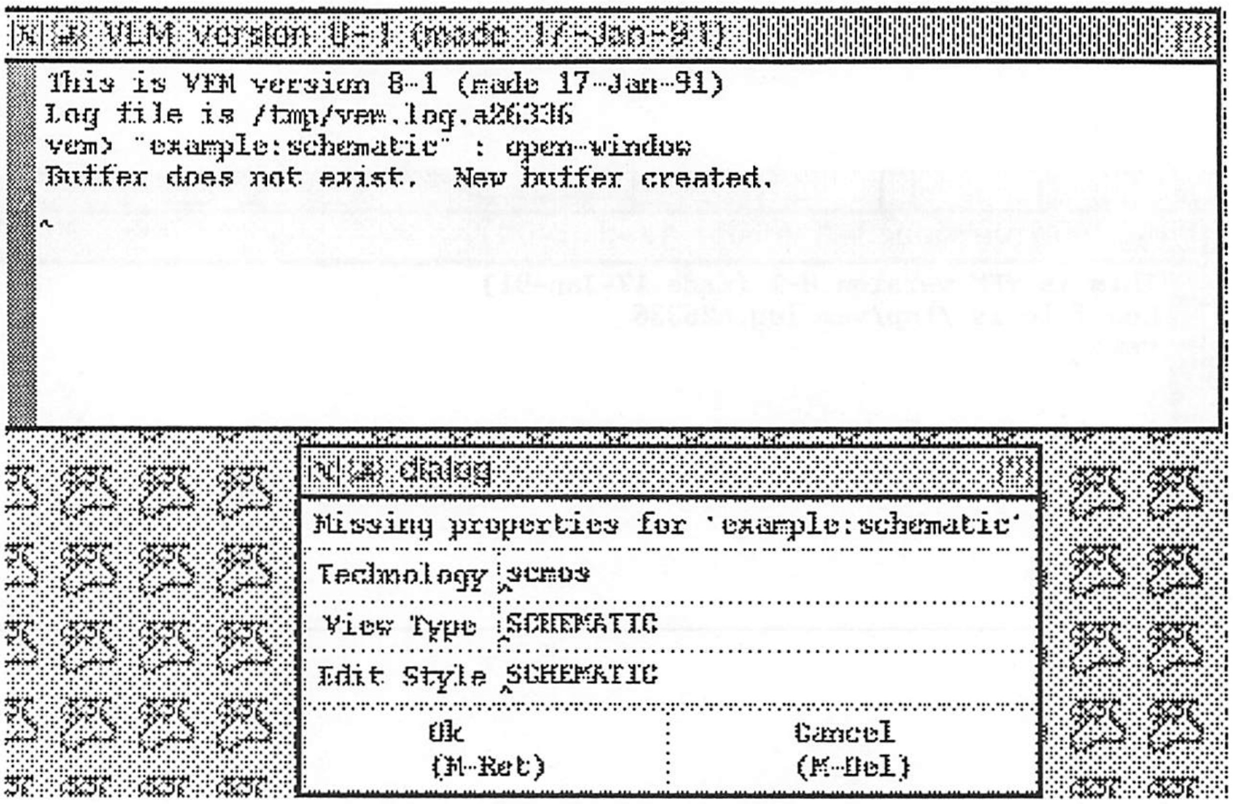


Figure 6.3.1



```
This is VEM version 8-1 (made 17-Jan-91)
Log file is /tmp/vem.log.a26336
vem>
```

Figure U1

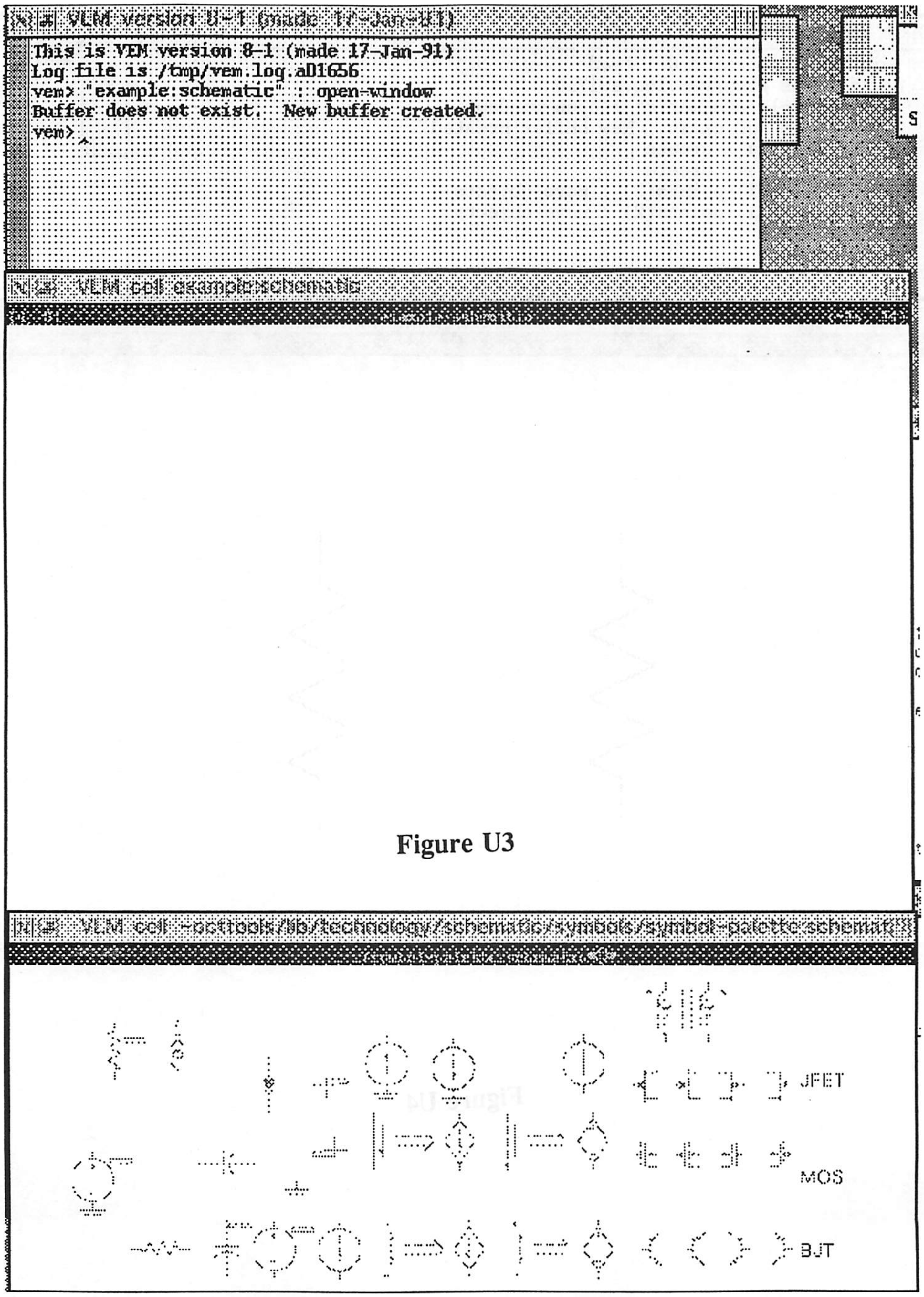


```
This is VEM version 8-1 (made 17-Jan-91)
Log file is /tmp/vem.log.a26336
vem> "example:schematic" : open-window
Buffer does not exist.  New buffer created.
^
```

Missing properties for 'example:schematic'

Technology	scmos
View Type	SCHEMATIC
Edit Style	SCHEMATIC

Figure U2


VEM version 8-1 (made 17-Jan-91)
VEM cell example:schematic
VEM cell ~octtools/lib/technology/schematic/symbols/symbol-palette:schematic
JFET
MOS
BJT

```
This is VEM version 8-1 (made 17-Jan-91)
Log file is /tmp/vem.log.a01656
vem> "example:schematic" : open-window
Buffer does not exist.  New buffer created.
vem>
```

Figure U3

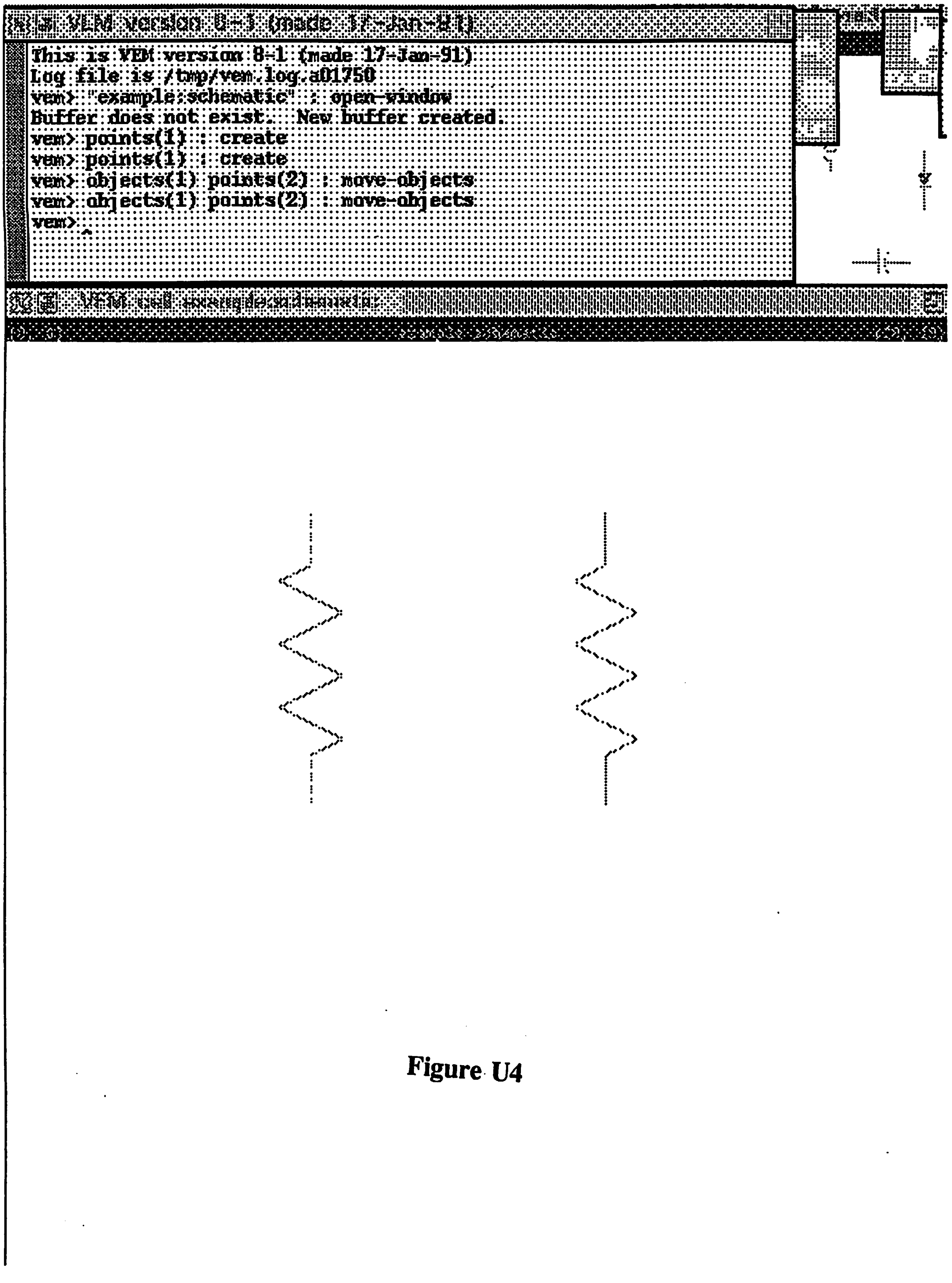
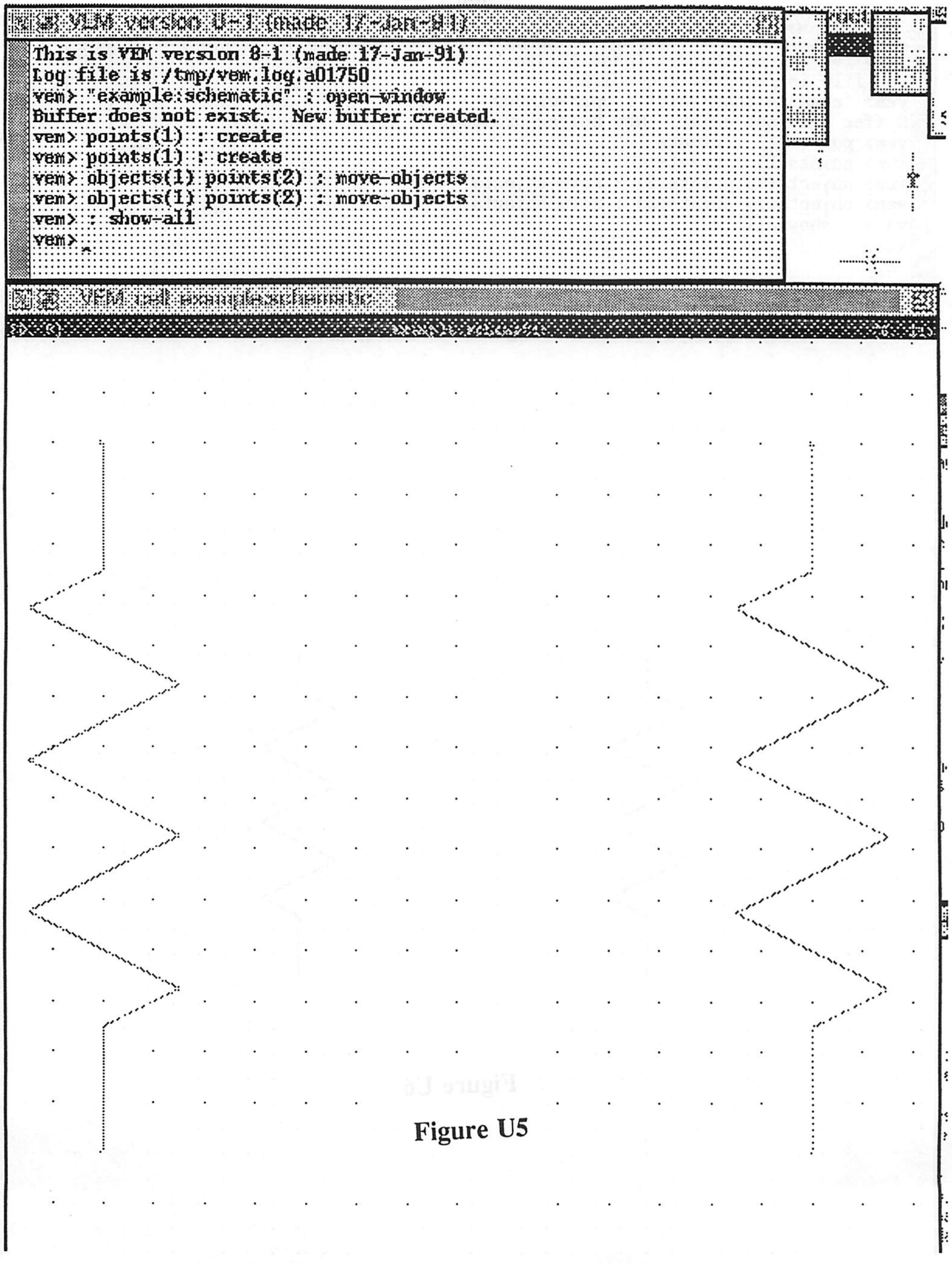
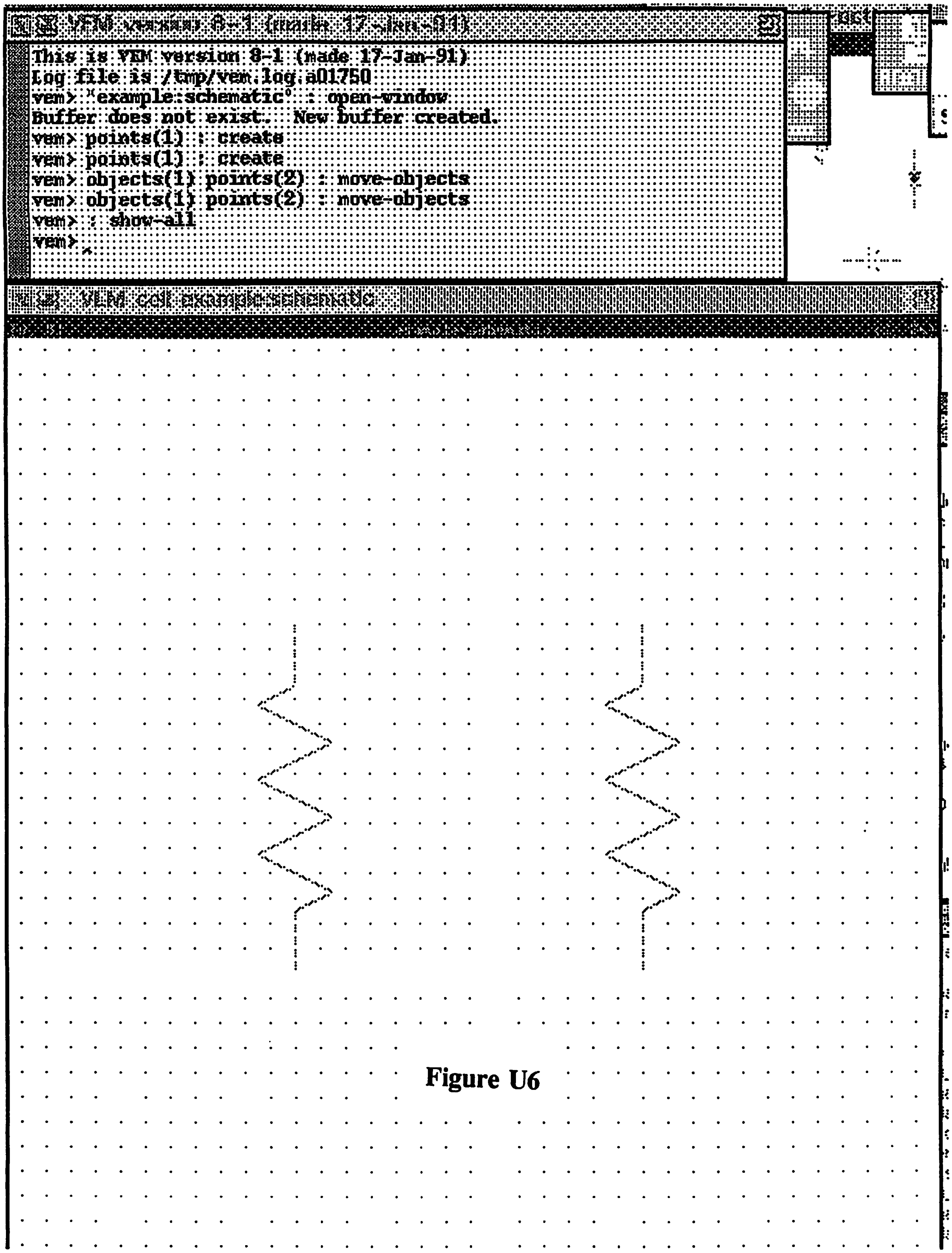

```
This is VEM version 8-1 (made 17-Jan-91)
Log file is /tmp/vem.log.a01750
vem> "example:schematic" : open-window
Buffer does not exist.  New buffer created.
vem> points(1) : create
vem> points(1) : create
vem> objects(1) points(2) : move-objects
vem> objects(1) points(2) : move-objects
vem>
```

Figure U4



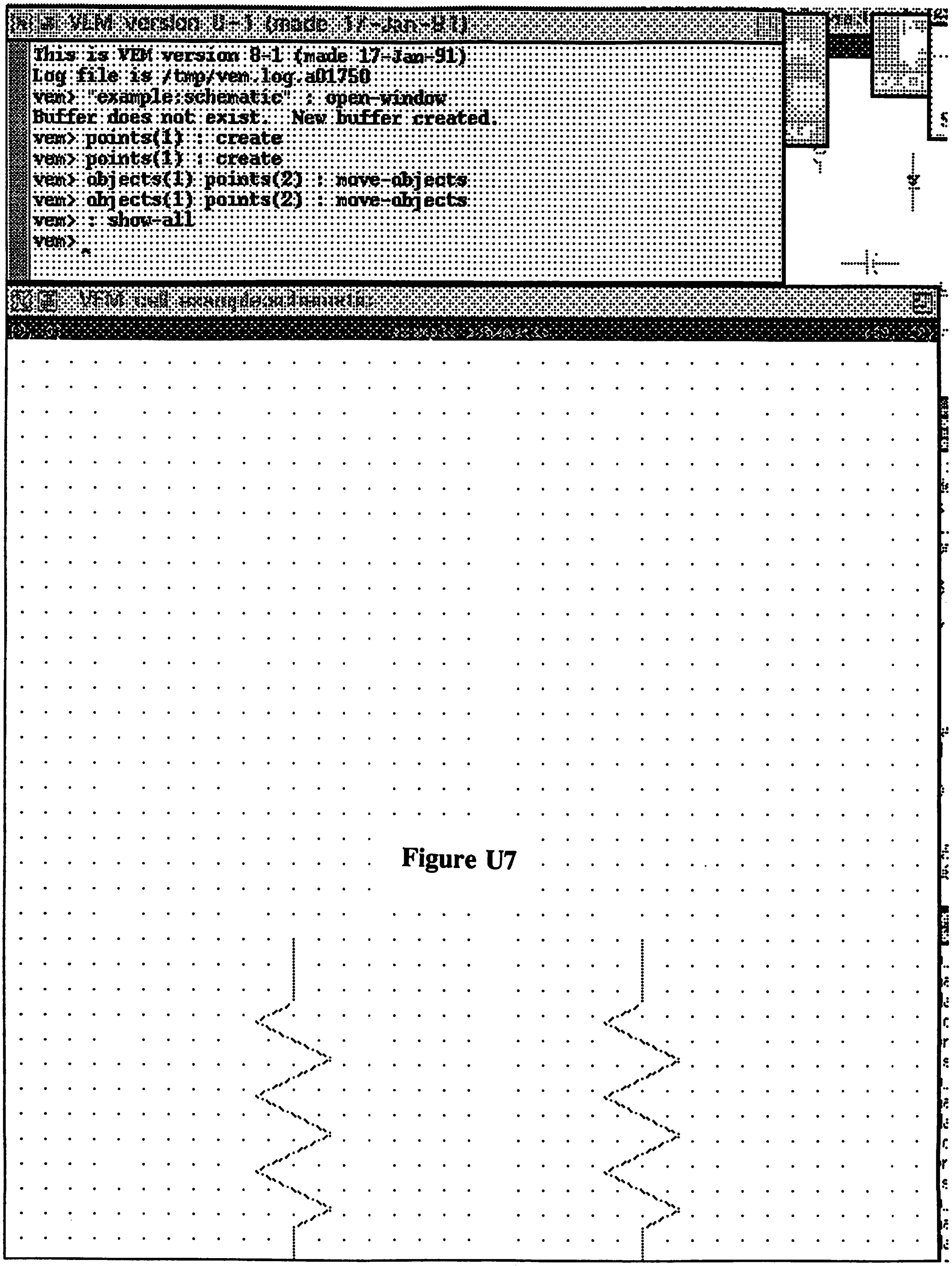
```
This is VEM version 8-1 (made 17-Jan-91)
Log file is /tmp/vem.log.a01750
vem> "example:schematic" : open-window
Buffer does not exist.  New buffer created.
vem> points(1) : create
vem> points(1) : create
vem> objects(1) points(2) : move-objects
vem> objects(1) points(2) : move-objects
vem> : show-all
vem> _
```

Figure U5



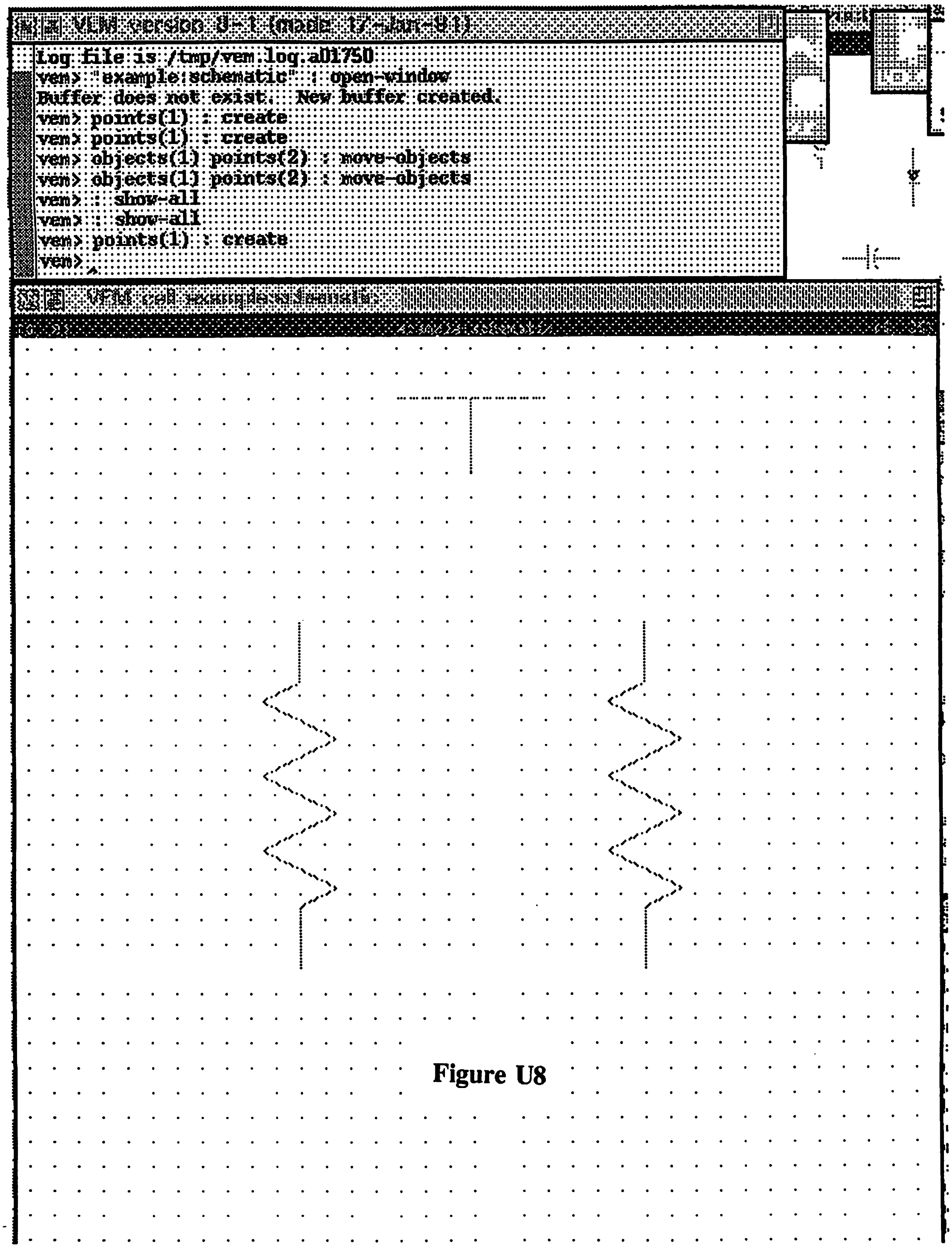
```
This is VEM version 8-1 (made 17-Jan-91)
Log file is /tmp/vem.log.a01750
vem> "example:schematic" : open-window
Buffer does not exist.  New buffer created.
vem> points(1) : create
vem> points(1) : create
vem> objects(1) points(2) : move-objects
vem> objects(1) points(2) : move-objects
vem> : show-all
vem>
```

Figure U6



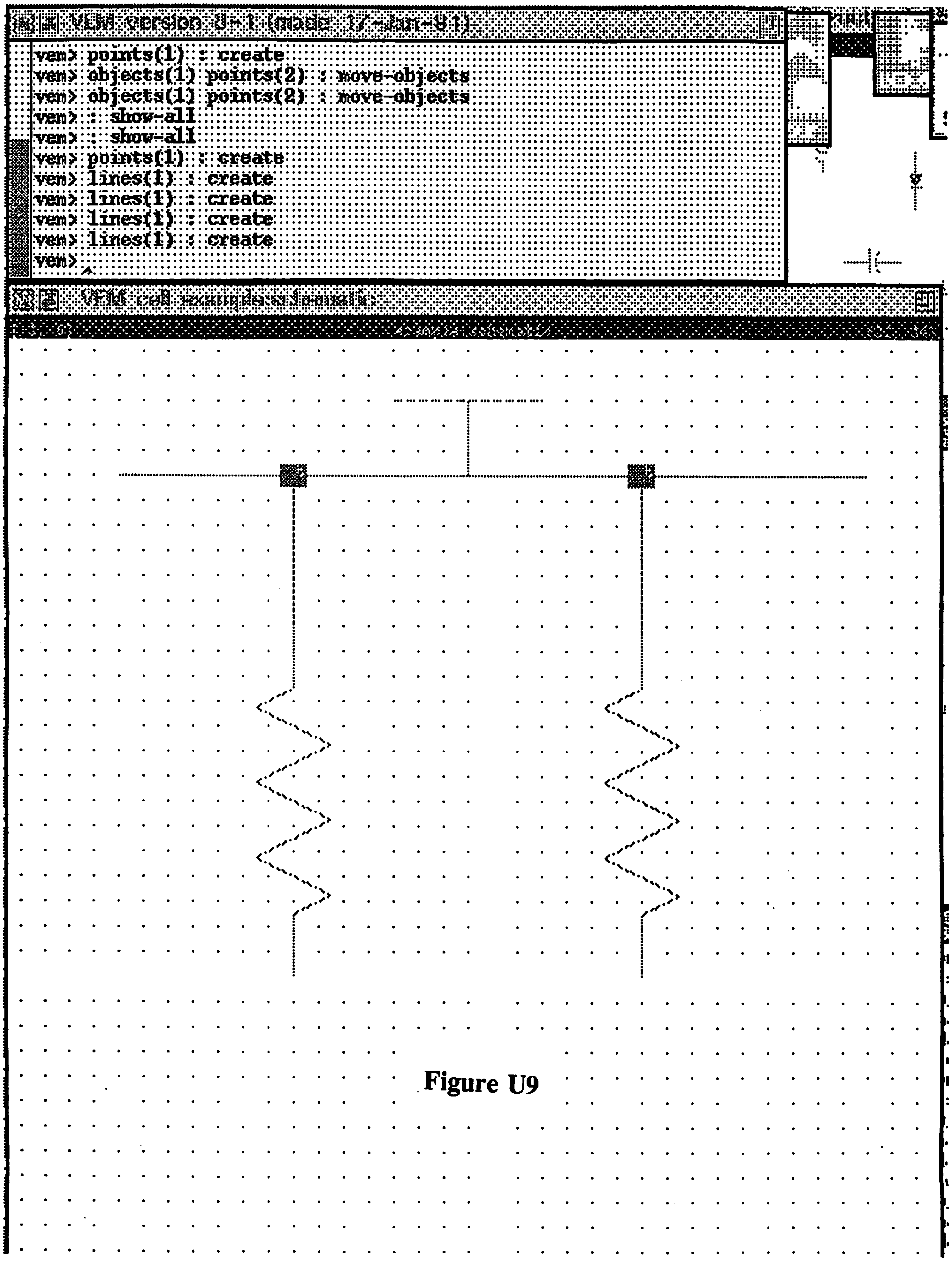
```
This is VEM version 8-1 (made 17-Jan-91)
Log file is /tmp/vem.log.a01750
vem> "example:schematic" : open-window
Buffer does not exist.  New buffer created.
vem> points(1) : create
vem> points(1) : create
vem> objects(1) points(2) : move-objects
vem> objects(1) points(2) : move-objects
vem> : show-all
vem>
```

Figure U7



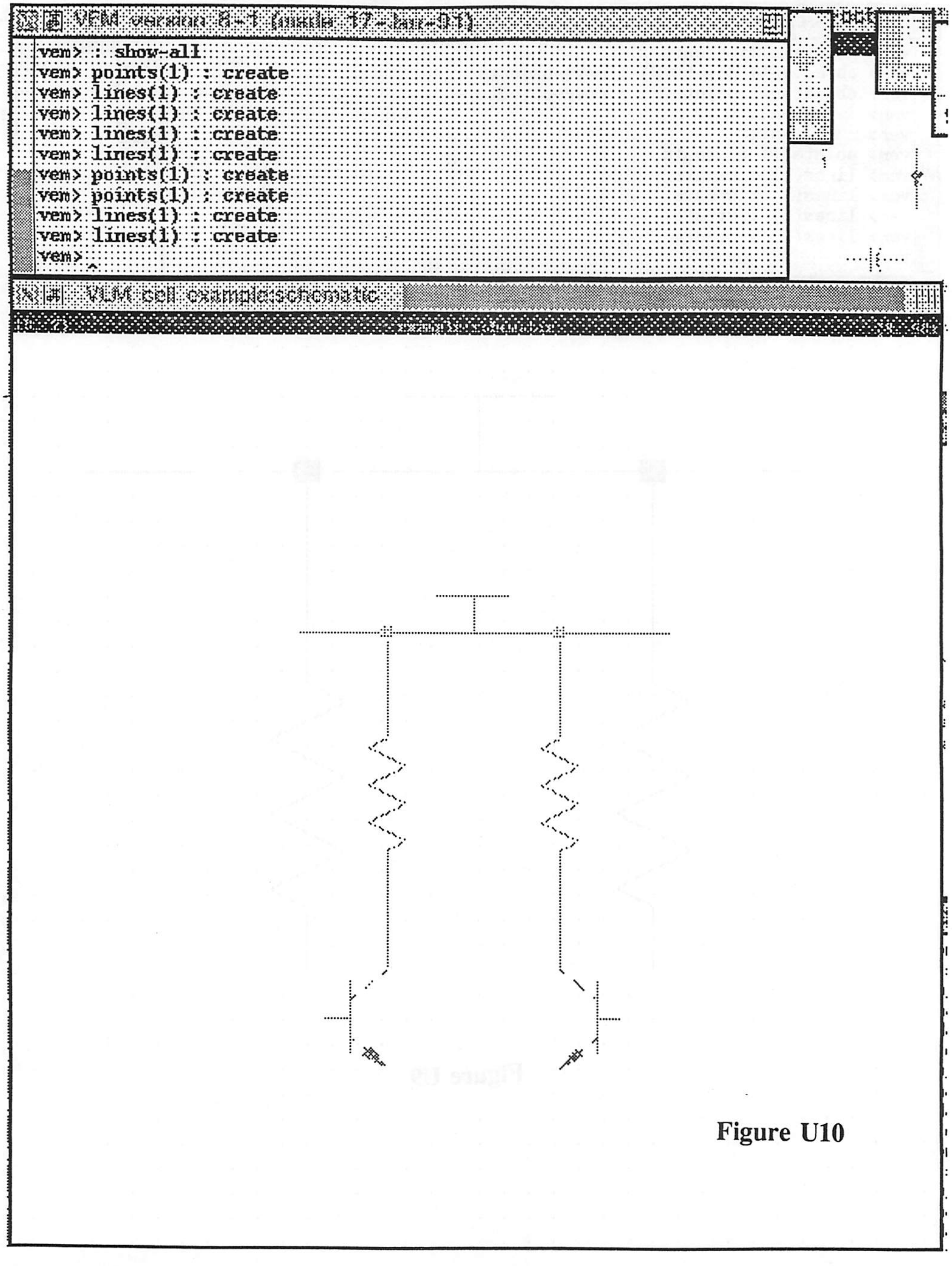
```
Log file is /tmp/vem.log.a01750
vem> "example:schematic" : open-window
Buffer does not exist.  New buffer created.
vem> points(1) : create
vem> points(1) : create
vem> objects(1) points(2) : move-objects
vem> objects(1) points(2) : move-objects
vem> : show-all
vem> : show-all
vem> points(1) : create
vem>
```

Figure U8



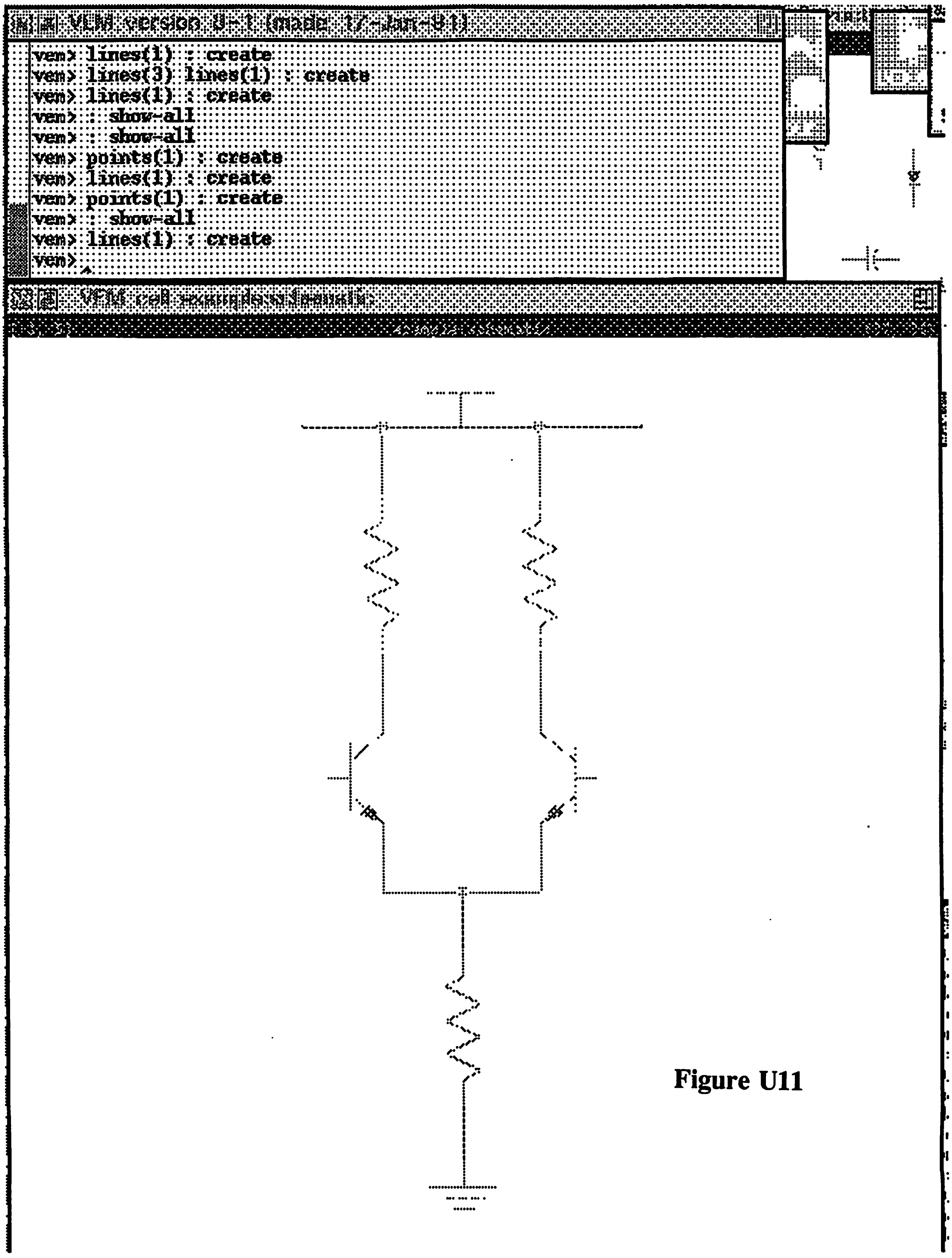
```
vem> points(1) : create
vem> objects(1) points(2) : move-objects
vem> objects(1) points(2) : move-objects
vem> : show-all
vem> : show-all
vem> points(1) : create
vem> lines(1) : create
vem> lines(1) : create
vem> lines(1) : create
vem> lines(1) : create
vem>
```

Figure U9



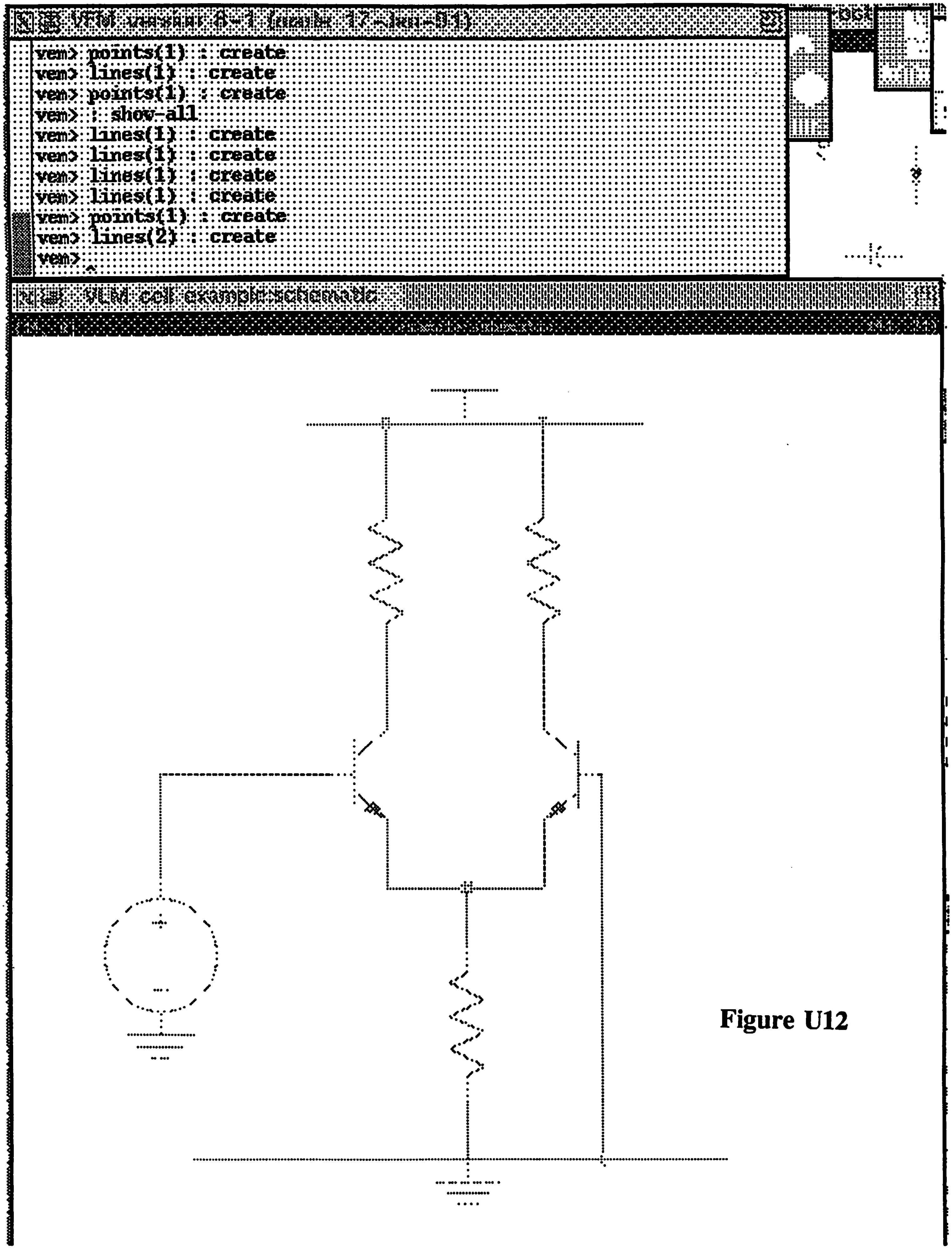
```
vem> : show-all
vem> points(1) : create
vem> lines(1) : create
vem> lines(1) : create
vem> lines(1) : create
vem> lines(1) : create
vem> points(1) : create
vem> points(1) : create
vem> lines(1) : create
vem> lines(1) : create
vem>
    ^
```

Figure U10



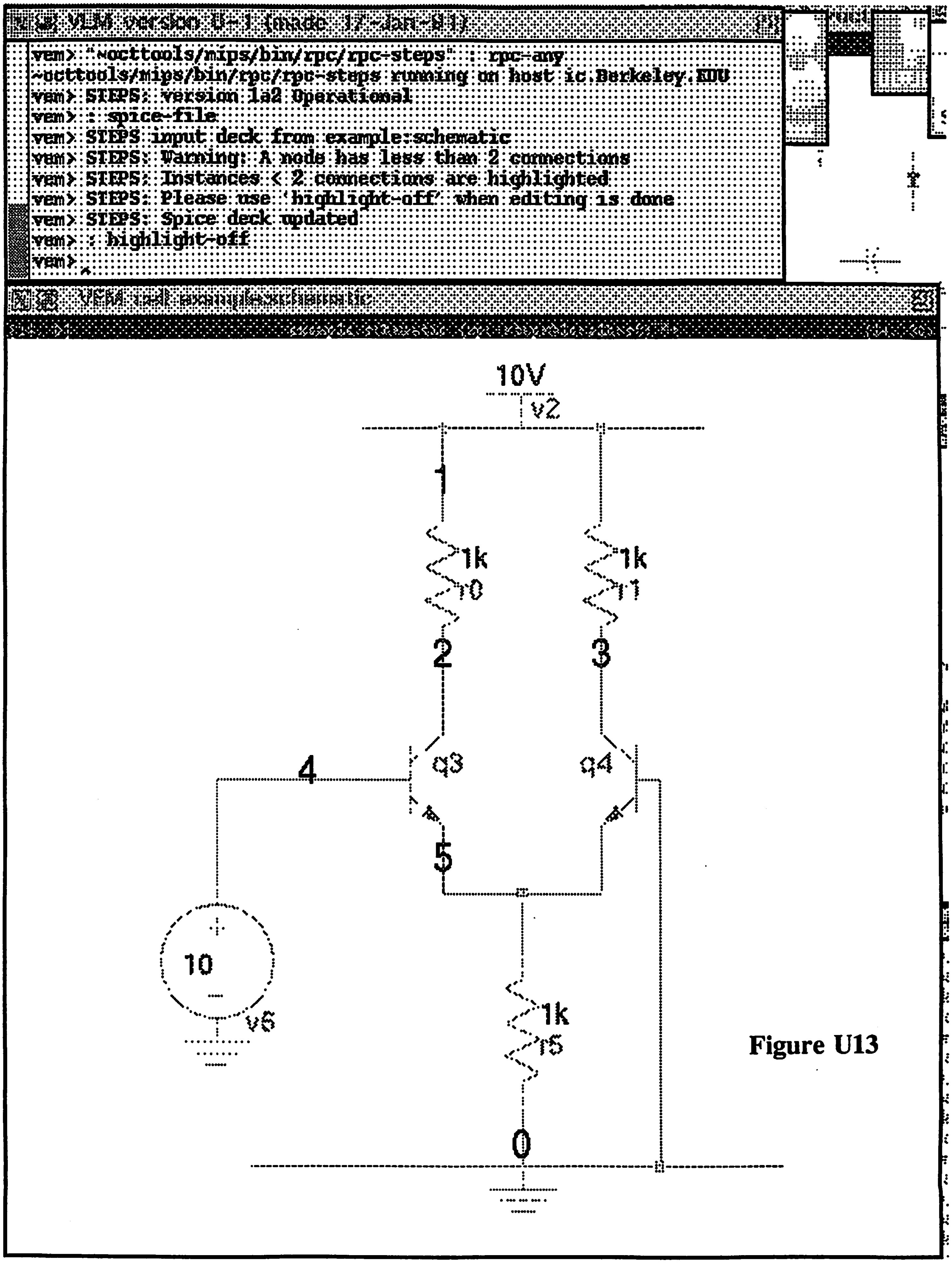
```
VEM version 8-1 (made 17-Jan-91)
vem> lines(1) : create
vem> lines(3) lines(1) : create
vem> lines(1) : create
vem> : show-all
vem> : show-all
vem> points(1) : create
vem> lines(1) : create
vem> points(1) : create
vem> : show-all
vem> lines(1) : create
vem>
```

Figure U11



```
vem> points(1) : create
vem> lines(1) : create
vem> points(1) : create
vem> : show-all
vem> lines(1) : create
vem> lines(1) : create
vem> lines(1) : create
vem> lines(1) : create
vem> points(1) : create
vem> lines(2) : create
vem> ^
```

Figure U12



```
vem> "~octtools/mips/bin/rpc/rpc-steps" : rpc-any
~octtools/mips/bin/rpc/rpc-steps running on host ic.Berkeley.EDU
vem> STEPS: version 1a2 Operational
vem> : spice-file
vem> STEPS input deck from example:schematic
vem> STEPS: Warning: A node has less than 2 connections
vem> STEPS: Instances < 2 connections are highlighted
vem> STEPS: Please use 'highlight-off' when editing is done
vem> STEPS: Spice deck updated
vem> : highlight-off
vem>
```

Figure U13

STEPS (Spice Text Editing Package utilizing Schemat

Denis S. Yip / Prof. D. O. Pederson

Version 1a3 / March 1990

User's Manual

Table of Contents

Section 1: INTRODUCTION

SPICE is an all-purpose electrical circuit simulation program. At present, most SPICE designers in the Berkeley system use ascii text editing for SPICE input file entry. Although there are commercial schematic programs, for example, ORCAD, to support schematic entry of SPICE for personal computers, schematic-entry packages are unavailable in the public domain to support UNIX and the X-Window systems. This is often inconvenient since most designers have to convert their schematic design diagrams into SPICE input files by naming all the subsequent nodes and entering parameters and values from a text editor. Errors frequently occur due to misnaming of nodes and corrections to these errors may be very cumbersome if the circuit is of significant size. Also, a schematic printout is unavailable.

STEPS is designed for local SPICE users to overcome the above difficulties through a public domain schematic-entry package. Users can easily design their circuits on terminals and check their designs with SPICE as often as needed. STEPS is used as a pre-processor for Spice2, Spice3, NECTAR (A Knowledge Based Environment to Enhance SPICE), and xgraph. Therefore, simulation of circuits by Spice2, Spice3, NECTAR, and xgraph can be done while running STEPS. A SPICE2-to-xgraph filter is also integrated to present direct xgraph output from the schematic diagrams through SPICE2. Furthermore, users can also integrate their own favorite simulators into STEPS as long as the simulators recognize SPICE input file.

STEPS utilizes the structure of the Berkeley CAD Framework -- Oct, VEM, and RPC (Remote Procedure Call package). The schematic package of VEM is the pre-processor for STEPS; therefore, STEPS can utilize all the components of VEM. In addition, layouts of the circuit elements can be done in VEM as desired. The RPC package is specific to the local system of remote functions communicating with VEM and OCT; therefore, the protocol and interface are specific to VEM and OCT. STEPS has all the advantages of VEM and OCT and can easily integrate with other programs which utilize the OCT object. On the other hand, the technology of VEM is altered for the convenience of SPICE users. The basis and format of STEPS are partially taken from the RPC-SPLICE program from University of Illnois at Urbana.

Device values and model names are printed in STEPS through the "label" structure in Oct, and the node numbers are automatically named and displayed in STEPS. Since speed is a major concern in STEPS, schematic editings are set up so that minumun key or menu inputs are required. All SPICE information has default values and editing of all SPICE models and devices is made as easy as possible. Since generation of SPICE information from schematics is the main goal of STEPS, SPICE users may find that STEPS has an advantage over other schematic-entry packages since many inputs with other packages for other information are not necessary in STEPS. Also, through the memory structure of OCT, all STEPS parameters are saved. Therefore, if a model or a parameter is changed, it is saved permanently even if the user has exited both STEPS and VEM.

The goal of STEPS is to enable the SPICE users to design circuits and simulate them concurrently with speed and robustness. Therefore, almost no text editings are needed since all SPICE information can be entered through schematics or dialog boxes in STEPS. All SPICE analysis (for example, sensitivity analysis, various types of plots, etc) are allowed to be done directly in STEPS. Furthermore, the postprocessor of SPICE, for example, NUTMEG, is also integrated inside STEPS. Therefore, SPICE outputs will be able to be analyzed in a more detailed format. The present emphasis is on increased functionality, higher-level modeling, and integration with display tools, and physical design tools, such as Integrated Circuit Layout. STEPS has accompanied some of these goals by utilizing the IC Layout Tool -- VEM.

Section 2: OVERVIEW

Familiarity in editing with VEM is a basic requirement for using STEPS effectively. This user-manual is presented in three sections:

1) TUTORIAL

Running through this tutorial provides a general idea on how to run STEPS.

2) STEPS

A detailed description of all current STEPS commands is given here.

3) VEM commands (Appendix A)

All VEM commands are presented here. Users who are familiar with VEM can skip this section. On the other hand, those who want to know more about VEM are advised to look at the VEM users' guide refer to ~octtools/document/PostScript/schematic.ps.Z or refer to ~dyip/steps/schematic.ps@ic.Berkeley.EDU or refer to ~dop/steps/schematic.ps@janus.Berkeley.EDU.

STARTUP:

Instructions to use STEPS in janus.berkeley.edu:

1) In your ~/.rhosts file, type in the following lines:

```
janus.Berkeley.EDU
```

janus

2) Copy ~dop/.Xdefaults into your home directory ~/.Xdefaults.

3) Set path to ~dop/bin as the first option on your path by modifying your ~/.cshrc file or ~/.login file. Your setpath should look something like set path = (. ~dop/bin ~dop $home/bin) ... etc

4) Run STEPS by typing "vem7-3 &" or "vem &" and following the TUTORIAL in the user manual. The user manual is contained in ~dop/steps/manual. You can access it by typing "ditroff -ms -P{Printer} manual" or get a copy by mailing dyip@ic.berkeley.edu or owner-octtools@ic.berkeley.edu

Instructions to use STEPS in ic.berkeley.edu:

1) In your ~/.rhosts file, type in the following:

ic.Berkeley.EDU

ic

2) Copy ~dyip/.Xdefaults into your home directory ~/.Xdefaults.

3) Set path to ~dyip/bin as the first option on your path by modifying your ~/.cshrc file or ~/.login file. Your setpath should look something like set path = (. ~dyip/bin ~dyip $home/bin) ... etc

4) Run steps by typing "vem7-3 &" or "vem &" and following the TUTORIAL in the user's manual. The user's manual is contained in ~dyip/steps/manual. You can access it by typing "ditroff -ms -P{Printer} manual" or get a copy by mailing dyip@ic.berkeley.edu or owner-octtools@ic.berkeley.edu

The online help menu of STEPS gives a detailed description of STEPS and a brief description of all VEM commands. Look through the online menu when problems arise. Currently, only ic.Berkeley.EDU and janus.Berkeley.EDU support STEPS. If you want your machine to be supported, please mail steps@ic.Berkeley.EDU.

Section 3: TUTORIAL

The following is a brief tutorial on how to use STEPS. Follow through the tutorial, and you will gain a basic idea on how STEPS works. For efficient use of STEPS, familiarity and experience are necessary. It should be keep in mind that an emitter-coupled pair is being created in this tutorial.

Setup STEPS by following the instructions in Section 2: Overview's "Startup" section. Then, after setting up the appropriate display (X11 Window System is required), type "vem &" in your UNIX shell. If VEM Version 7 instead of VEM Version 8 is desired, type "vem7-3 &". Now, a white VEM console window is opened, as shown in Figure U1. Type "example:schematic" and then press "o" on the keyboard with the mouse on the console window. "example" is the cellname and a "schematic" tells VEM that you are editing through schematics. Since the "example" cell does not yet exist, a message "Buffer does not exist. New buffer created" appears in the VEM console window. Then, a dialog box is displayed with "Technology=scmos", "View Type=schematic", and "Edit Style=schematic" in it. Move the cursor over the box labeled "OK" and click the left mouse button. Please refer to Figure U2.

A VEM graphics window is now opened. Arguments can be entered in VEM through points, lines, boxes, text, and object. Move the cursor to the VEM graphics window and click the left mouse button. This creates a point in the VEM graphics window. Hit the <delete> key to delete the point or simply press Control-u or Control-w to delete all the arguments. To enter a box, press the left mouse button, and hold it down while you move the mouse. As you move the mouse, a box appears. To draw a line, move the mouse to the place you would like one end of the line to be and click the left mouse button to create a point. With the cursor over the point you just entered, press the left mouse button again and move the mouse around. You have now created a line. Control-u and/or delete keys can be used to delete the above arguments.

While the cursor is in the VEM graphics (not console) window, hit the key "r" to invoke STEPS. You can also invoke STEPS from the VEM menu by clicking the middle mouse button. Under Applications, you will see the option "STEPS". Click the left mouse button over STEPS commands from VEM through key bindings or menu. After you have invoked STEPS, wait for the message "STEPS: operational" in the console window (the white one). The grey window is called the graphics window and the white window console window. Once the operational message is displayed, WHILE HOLDING DOWN THE SHIFT KEY, click the middle mouse button. You now get the STEPS menu. In order to get the regular VEM menu, just click the mouse button WITHOUT holding down the shift key.

Now, invoke the command "spice-devices" either from the STEPS menu (STEPS option) or type ":spice-devices" in the graphics window. A device palette window is now opened which contains all Spice devices. Please refer to Figure U3. Press two points next to each other about three inches apart in the graphics window. Move the cursor to the "spice-devices" palette and then move the cursor on top of the vertical resistor. Press "c" or invoke the command "create-instance" through VEM menu by pressing the middle mouse button. You have now created two resistors in the graphics window. Please refer to Figure U4. Try the command "show-all" by pressing "f" or invoking VEM menu. Figure U5 is a reference. "Show-all" zooms the window such that the devices are at the middle. Try the command "zoom-in" by pressing "z" or invoking the command through VEM menu. Then try the command "zoom-out" by pressing "Z" or invoking the command through the VEM menu. Please refer to Figure U6. Move the cursor to the top of the resistors and invoke the command "pan" by pressing "p" or invoking the command through VEM menu. Please refer to Figure U7. "Pan" gives you more space to work with on the directions desired. Continue to invoke these four commands until you are familiar with them.

Place a point above the resistor after you have "zoomed" the window to the appropriate scale (such that you have enough space to work). Move the cursor to the "spice-devices" palette at the top of a voltage supply (VCC) and then press "c" or invoke the command "create-instance" through menu. Now, you have a voltage supply on top of the two resistors. Please refer to Figure U8. Keep in mind that you are trying to create an emitter-coupled pair in this tutorial. Create a horizontal line right under the voltage supply. Then, create a vertical line between the terminal of the VCC and the horizontal line. Now, create two vertical lines between the horizontal line and the two resistors. If you place the line incorrectly, delete the line by the pressing the <delete> key two times or by pressing Control-u and then redraw the lines.

Press "c" or invoke the command "create" through VEM menu. Note the color of the wire on the node changes from light blue to brown. This means that the lines are connected and they are now wires instead of plain lines. Place a point on top of the wire (node) by pressing the left mouse button and then press Control-n (or select the VEM command "select-net" through the VEM menu). You will see the color changed to light blue and a message "A machine generated net has been selected" in the console window. Please refer to Figure U9. Nodes are called nets in VEM. The node is changed to a different color. This means that the entire node is connected together.

Now, under the left resistor, create a point and then choose an npn BJT into the graphics window. While placing the cursor on top of the BJT in the graphics window (not the "spice-devices" window), press "s" or invoke "select-object" from the VEM menu. You will see the highlighting of the npn. Now type "my:180" and press "t" or invoke the command "transform". This is to transform the object by 180 degrees by the y-axis (mirror). Then press "m" or invoke the command "move-objects" in the VEM menu. Press "u" or invoke the command "unselect-objects". You will see that the npn is mirrored. Move the npn under the right resistor by selecting the npn using "s" and placing two points. The first point should be placed under the left transistor and the second under the right one. Look at how the object is moved by pressing "m". Now create another npn from the spice-palette under the left resistor. "Copy-objects" is similar to "move-objects". Select the left npn by pressing "s" and then place the first point on top of the selected object. Place the second point on the leftmost side and then press "x" or invoke "copy-objects" through menu or key. Unselect the npn. You have now copied an object. Please refer to Figure U10. To remove the object, select the leftmost npn again using "s" and then press "D". The object is now deleted.

Remember that an emitter-coupled-pair is being formed. Create two lines from the bottom of the resistors to the collector of the npn. Press "c". Now connect the two emitters of the npn's by lines and then press "c". Create a resistor under all the objects. Create a line from the top of the resistor to the line which connect the emitters. Press "c" again. Now, create a ground under the resistor and connect the ground to the bottom of the third resistor. Also, create a long line at the ground terminal so that a long grounded node is represented. Please refer to Figure U11. Connect the base of the right npn to the ground. Place a voltage source on the left side and connect the top of the voltage source (positive side) to the base of the left npn and connect the bottom of the voltage source to ground (or use the voltage source with the grounded symbol on the negative end). An emitter-coupled pair is now created, as in Figure U12.

Now, press ":spice-file" and hit RETURN or invoke the command "spice-file" under "text file" in the STEPS menu by pressing SHIFT and clicking the middle mouse button together. Look at how the display is changed, as in Figure U13. You can also edit the models, (e.g., model values), by invoking the STEPS command "model-edit". The model names can also be changed. Other SPICE options can be entered through the "SPICE-Control" commands in STEPS. Try all the commands in STEPS one-by-one by following the last section of this menu (Section 4: STEPS commands).

One advantage of schematic entry is to copy subcircuits directly and quickly. You are now going to copy the whole emitter-coupled-pair (ECP). Make sure you have enough space to copy the whole ECP by panning (pressing "p" or by menu - "pan") and zooming (pressing "z" or by menu - "zoom-in"). Use a box to surround the whole emitter-coupled-pair (ECP). Press "s" or by menu - "select-objects" inside the box. Copy the whole box by putting one point inside the box and another outside it (again, make sure you have enough space), and then press "x" or by menu - "copy-objects". Press "u" (or by menu - "unselect-objects") to unselect. Now you have two ECP's. Connect the two ECP's and then invoke the command "spice-file" again. Look at how STEPS analyzes the whole circuit.

By working with several schematic diagrams and invoking more STEPS commands, you will be familiarized with STEPS. Report any difficulties, bugs, or simply comments to steps@ic.Berkeley.EDU.

PRINTOUT OF SCHEMATICS

You can printout the schematics of the circuit by:

1) Using the option "print-out" in STEPS.

2) In UNIX, type in "oct2ps <cellname>:schematic > <cellname>.ps" to generate a Postscript output file. For example, type in "oct2ps example:schematic > example.ps" to generate a hardcopy of the cell named 'example'.

3) In UNIX, type in "oct2ps <cellname>:schematic | lpr -P{Printer}" to direct output to printer directly.

Section 4: STEPS COMMANDS

4.1) OVERVIEW of STEPS Commands

STEPS - spice-devices -

"Open a window displaying all the Spice devices"

STEPS - print-out -

"Present a PostScript Printout of the schematic diagram"

STEPS - bugs -

"Online bug report, which is mailed directly to the author"

STEPS - quit -

"Quit STEPS"

Edit - model-edit -

"Edit the selected model"

Edit - chng-local-parm -

"Change the model parameters of selected device locally"

Edit - chng-global-parm -

"Change the model parameters of selected device globally"

Text File - spice-file -

"Generate a Spice input file of the schematics"

Text File - add-volt-source -

"Generate a Spice input file with a voltage source added at the selected node (usually the input)"

Text File - add-curr-source -

"Generate a Spice input file with a current source added from ground to the selected node (usually the input)"

Analyses - xgraph -

"Display an xgraph output of the selected node after running .plot on Spice2"

Analyses - edit-after-update -

"Display an editing window after the schematics are changed to a Spice input file"

Analyses - Spice2 -

"Display the Spice2 output after converting from schematics to Spice"

Analyses - Spice3 -

"Display a Spice3 window of the corresponding schematics"

Analyses - nectar -

"Display a nectar window of the corresponding schematics"

Analyses - curr-file-edit -

"Edit the current .spi file without updating from schematics to Spice"

Analyses - curr-file-spice2 -

"Running Spice2 without updating from schematics to Spice"

Analyses - curr-file-spice3 -

"Running Spice3 without updating from schematics to Spice"

Analyses - curr-file-nectar -

"Running nectar without updating from schematics to Spice"

Analyses - choose-editor -

"Choose the desired editor - vi or Emacs"

Analyses - integrate -

"Integrate other simulators or programs into STEPS"

Analyses - execute -

"Execute programs which had been 'integrated' by users.

SPICE-Control - .options -

"Invoke .options in Spice"

SPICE-Control - .temp -

"Invoke .temp in Spice"

SPICE-Control - .width-in -

"Invoke .width in"

SPICE-Control - .op -

"Invoke .op in Spice"

SPICE-Control - .dc -

"Invoke .dc in Spice by selecting the corresponding source"

SPICE-Control - .ac -

"Invoke .ac in Spice"

SPICE-Control - .tf -

"Invoke .tf in Spice by selecting the corresponding source"

SPICE-Control - .sens -

"Invoke .sens in Spice"

SPICE-Control - .disto -

"Invoke .disto in Spice by selecting the corresponding load resistor"

SPICE-Control - .noise -

"Invoke .noise in Spice by selecting the corresponding voltage or current source"

SPICE-Control - .tran -

"Invoke .tran in Spice"

SPICE-Control - .four -

"Invoke .four in Spice"

SPICE-Control - .print -

"Invoke .print in Spice"

SPICE-Control - .plot -

"Invoke .plot in Spice"

SPICE-Control - .ic -

"Invoke .ic in Spice by choosing the initial conditions of selected nodes"

SPICE-Control - .nodeset -

"Invoke .nodeset in Spice by choosing the conditions of selected nodes"

SPICE-Control - user-options -

"User can include their own text in this command"

Display - highlight-off -

"Turn off the highlight of erroneous models"

Display - model-name-off -

"Turn off all model names display"

Display - model-name-on -

"Turn on all model names display"

Display - model-value-off -

"Turn off all model values display"

Display - model-value-on -

"Turn on all model values display"

Display - node-value-off -

"Turn off all node values display"

Display - node-value-on -

"Turn on all node values display"

Display - all-value-off -

"Turn off all Spice displays"

Display - all-value-on -

"Turn on all Spice displays (default is on)"

HELP - user-help -

"Online HELP manual"

4.2) STEPS commands

I) spice-devices

This command displays a window (called a palette) which has the symbols of the Spice devices. Resistors, inductors, capacitors, dependent sources, transformers, voltage sources, current sources, voltage sources connected to ground, ground, voltage supply, diodes, JFET's, BJT's, and MOSFETs are displayed by windows. The user can press the left mouse button in his own console window to select a point and then put the mouse on the selected symbol and then press "c" or invoke the MENU command "create-instance". The device selected is created in the location specified. Note that two types of MOSFETs are present: The lowest version has only three terminals - gate, drain, and source. The substrate is assumed to be connected to ground. If something other than ground is needed to connect to the substrate, the upper version of mosfet, which has four terminals (gate, drain, source, and substrate), should be selected.

II) print-out

Invoking this command automatically generates a PostScript printout of the schematic diagram. There are three options for this command: namely, "Suppress node values", "Suppress model names", and "Suppress model values". The user can select one or more of the above options. Not selecting any options by pressing "OK" will give a PostScript printout of a fully labeled schematic printout. Suppressing one or all of the above options leads to a PostScript printout with no labels on the suppressed options. The PostScript printout is named <cellname>.ps, and the user can obtain a printout with the Unix command "lpr -P{Printer} <cellname>.ps".

III) bugs

Invoking this command opens a dialog window. User is asked to report any bugs, suggestions, or comments promptly. Typing in the bugs will lead this command to mail the information to the author.

IV) quit

Invoking this command quits STEPS permanently. The user can still use VEM, but if STEPS is desired, the user has to either press "r" in his VEM console window, or invoke the STEPS command through menu.

4.3) Edit commands

I) model-edit

This command allows the user to edit every model inside the "spice-devices" palette. All the necessary Spice parameters are requested through dialog windows. Move the mouse on top of the model that needs to be edited and type ":model-edit" or invoke this command through the STEPS menu by pressing the shift key and the middle mouse button together. All of the necessary commands are asked and you can ignore those commands which you do not intend to enter. Once the model (or instance, as called in OCT) is edited, no subsequent modeling is needed. Even if the user exits STEPS and VEM totally, the model parameter is presented provided that the user has saved the window by the vem-command "save-window" or "save-all". Note that inside the dialog windows, EMACS editing is assumed. Ctrl-k kills the line to allow reentering of text data. Also note that it is not necessary to press RETURN after entering the data. Simply press 'OK' after you finished editing.

II) chng-local-parm

"chng-local-parm" changes the local parameters of the model selected. Move the mouse on top of the model that needs to be edited and press ":chng-local-parm" or invoke the command through the STEPS menu. The model parameters of SPICE include only JFET, MOSFET, BJT, and diodes. So, selecting models other than the above presents a warning message. The local model parameters can be changed. For example, if the global npn BF is 100, but you only want to change one npn to 40, you can easily do so by invoking this command. Also, if you have 10 npn transistors with BF equals 100 and another 10 equal 50, you can first set the global parameter to 100, and then change one model through this command to 50. Finally, you can copy the other 9 npn's (with BF = 50) by copying the model with local parameters changed to 50 using the "copy-objects" command on that specific model.

III) chng-global-parm

This command changes the global default parameters of all MOSFETs, JFETs, BJTs, and diodes. Press ":chng-global-parm" or invoke the command using STEPS menu on any spot in the VEM graphics window. The default values or the last changed values are displayed, and the user can change any parameter. The Spice input file uses these parameters on all devices (for example, all npn's) unless the device parameters are changed locally through "chng-local-parm".

4.4) Text File

I) spice-file

This command is the heart of the program. In any position of the VEM graphics window, invoke this command by typing ":spice-file" or through the STEPS menu. Naming the nodes (or nets, as called in VEM and OCT) are not necessary. Once this command is invoked, the Spice file is generated. All the node and model names and values are named accordingly and then displayed. The Spice input file is called <cellname>.spi and is contained in the home directory or in the directory where the cell is contained.

II) add-volt-source

This command is the same as "spice-file" except that a voltage source is connected to ground and added in the Spice input file. On the node where the voltage source is to be added, make a point by placing the mouse on top of the node and press the left mouse button. A point is displayed on top of the wire (or net). Press Control-n to invoke the VEM command "select-net" or choose the command through the VEM menu by pressing the middle mouse button. Then, press ":add-volt-source" or invoke the command through the STEPS menu by pressing shift and middle mouse button. A voltage source is then added in the Spice input file. The positive side is the node selected and the negative side is the ground. Of course, negative values are allowed on the voltage source by using "model-edit". This command is designed since most inputs of an IC circuit requires a source input. This saves time in editing the circuit.

III) add-curr-source

This command is the same as "add-volt-source" except that a current source instead of a voltage source is selected. The current source is pointed from ground to the node selected. Negative values are also permitted. For more details, see Section II above.

4.5) Analyses

I) xgraph

This command is designed for Spice2 users so that a graphics plot instead of an ascii plot from Spice2 can be displayed. Invoke this command through the menu or typing in ":xgraph" in the graphics window. Five options are presented: namely, .dc, .ac, .tran, .disto, and .noise. Dialog windows are opened to ask the users to enter subsequent data for the plot. Then, an xgraph plot is presented after STEPS runs through Spice2 internally. The user can either select a hardcopy by changing the plot to .ps file and printing them as necessary. This command is also convenient for Spice3 users if a hardcopy is needed since some printers do not support Spice3 Nutmeg hardcopy plots. You can also select the nodes through schematics by putting a point or line on top of the node (net) and then press Ctrl-n before invoking this command. Any number of nodes are accepted. These are used as arguments and no manual text entry is necessary.

II) edit-after-update

Instead of using another window to look at the Spice input file, the user can directly edit the Spice input file. Invoke this command through STEPS menu or by typing ":edit-after-update". An xterm window is opened with the newly updated Spice input file. The user can now edit the input file first before going to Spice simulator.

III) Spice2

This command directly invokes Spice2 from the schematic diagram. Invoke this command through menu or by typing ":spice2". A window is opened and the Spice2 output file is displayed. The user can look at the output promptly and directly from schematics. The user can then edit the schematics again until the correct result is obtained.

IV) Spice3

This command is similar to the command ":spice2" described above except that Spice3 is invoked. The user can directly run the circuit through the most updated version of Spice3 and then edit the circuit.

V) NECTAR

NECTAR is a knowledge-based environment of SPICE. NECTAR corrects errors in the SPICE inputs directly. Users who are not familiar with NECTAR are recommended to read the NECTAR menu first before invoking this command. Invoke this command through STEPS menu or by typing ":nectar". a NECTAR window is opened and the user can directly use nectar to simulate the circuit from schematics and edit the schematics until a correct result is obtained.

VI) curr-file-edit

This command is the same as "edit-after-update" except that the Spice input file is not updated. The previous Spice input file therefore is displayed by an editor. This command is presented so that a user who has changed the schematics but is unsure about the change can use this command to try out the previous circuit. Also, the user can refer back to the previous circuit to compare with the present circuit. Thus, the current <cellname>.spi (Spice input file) is edited without updating from schematics to Spice input. If no <cellname>.spi file exists, a warning message is displayed.

VII) curr-file-spice2

This command is the same as "spice2" except that the Spice input file is not updated. The previous Spice input file therefore is simulated by Spice2. This command is presented so that a user who has changed the schematics but is unsure about the change can use this command to try out the previous circuit. Also, the user can refer back to the previous circuit to compare with the present circuit. Thus, the current <cellname>.spi (Spice input file) is analyzed without updating from schematics to Spice input. If no <cellname>.spi file exists, a warning message is displayed.

VIII) curr-file-spice3

This command is the same as "spice3" except that the Spice input file is not updated. The previous Spice input file therefore is simulated by Spice3. This command is presented so that a user who has changed the schematics but is unsure about the change can use this command to try out the previous circuit. Also, the user can refer back to the previous circuit to compare with the present circuit. Thus, the current <cellname>.spi (Spice input file) is analyzed without updating from schematics to Spice input. If no <cellname>.spi file exists, a warning message is displayed.

IX) curr-file-nectar

This command is the same as "nectar" except that the Spice input file is not updated. The previous Spice input file therefore is simulated by nectar. This command is presented so that a user who has changed the schematics but is unsure about the change can use this command to try out the previous circuit. Also, the user can refer back to the previous circuit to compare with the present circuit. Thus, the current <cellname>.spi (Spice input file) is analyzed without updating from schematics to Spice input. If no <cellname>.spi file exists, a warning message is displayed.

X) choose-editor

This command asks the user to choose an editor -- either Emacs or vi.

XI) integrate

This command is introduced for the convenience of the users to introduce their own executable programs to complement other CAD tools to STEPS. Many SPICE versions (called "alphabet spices") are available nowadays with special models suitable for individual needs. For example, the program S-SPICE is used for implementation of algorithms for periodic steady-state analysis in SPICE3. This command therefore is introduced to allow users to integrate CAD tools into STEPS with great ease. One requirement of such program is that SPICE input file is the input to the program. When this command is invoked, a dialog window is opened to ask users to enter two fields. The first field is the path and name of the executable program and the second field is solely the name of the program. For example, if the user wants to integrate a program to STEPS called hspice, which is contained in the directory /ic3/user/bin/hspice, the user should enter "/ic3/user/bin/hspice" for the first field and "hspice" for the second. Rather than entering "~user/bin/hspice", enter the full pathname. Otherwise, errors may occur.

XII) execute

This command is used to execute programs and/or CAD tools which the user has introduced through the command "integrate". The user is asked to enter a program name which he had integrated from the command 'integrate'. Then, a new spice file is created from STEPS and the program can be ran and evaluated inside STEPS. The user is then asked to select the way the program is run. For example, the way spice2 is run is "spice2 < input > output" and the way spice3 is run is "spice3 input". So, the user can choose the approriate way through a selection of menus. Keep in mind that the program should take SPICE input file as the input. Otherwise, unpredictable results may be obtained.

4.6) SPICE-Control

The SPICE-Control commands are designed for the convenience of SPICE users to edit the Spice files without going through text editing. Therefore, the users may edit all the Spice options through STEPS' dialog windows. The commands and parameters needed are requested automatically. The SPICE editing is done at a high speed since the users do not need to type in the whole line of options. Also, some values are automatically generated by schematic arguments to further hasten the SPICE editing process. If the user wants to delete an option (not using that option on subsequent Spice files), he can invoke the option command and press the CANCEL button on the dialog window. For example, if the user has initially chosen the .temp options but wishes to cancel the .temp option in subsequent runs, he can simply invoke the ".temp" command and then press the CANCEL button when the dialog window is opened. Then, STEPS automatically drops the option from the SPICE input file.

I) .options

When this command is invoked by the STEPS menu or by typing ":.options", all the options in Spice can be entered with ease. A first dialog window asks the users to select by mouse one or more of the options below: ACCT, LIST, NOMOD, NOPAGE, NODE, OPTS. These commands have no arguments. Then, a second dialog window is opened to ask the users to choose any .options value in SPICE. The following options are presented: GMIN, RELTOL, ABSTOL, VNTOL, TRTOL, CHGTOL, PIVTOL, PIVREL, NUMDGT, TNOM, ITL1, ITL2, ITL3, ITL4, ITL5, CPTIME, LIMTIM, LIMPTS, LVLCOD, LVLTIM, METHOD, MAXORD, DEFL, DEFW, DEFAD, and DEFADS. All default arguments are also printed in the dialog window. If the user wants to change one of the parameters, he can simply do so through the mouse, and the changed value(s) is printed in the Spice input file when the Spice input file is updated from schematic-entry.

II) .temp

If the user wants to analyze the circuit at different temperatures, this command can be invoked by menu or by typing ":.temp". Enter the values needed. For example, if the user wants to simulate the circuit at three tempeartures, namely: -55, 25, 125 (all in Celcius), he needs to type "-55, 25, 125" in the dialog window.

III) .width-in

The default value of ".width in" in Spice is 80. The user can change this to any appropriate value by this command.

IV) .op

".op" forces SPICE to determine the dc operating point of the circuit with inductors shorted and capacitors opened. Invoke this command and press either YES or NO. "YES" asks Spice for an dc operating point analysis. The default value is "NO".

V) .dc

Since the ".dc" command in SPICE requires a voltage or current source, this command needs a source as the argument. Move the mouse on top of a voltage source, current source, or a voltage supply (vdd, vss, vcc, or vee). Invoke this command by menu or by typing ".dc" while the mouse is on top of a source. Then, a multi-argument dialog window is opened to request for the VSTART, VSTOP, and VINCR values, which are the starting, final, and incrementing values, respectively. Then, a .dc is generated in the Spice input file on the selected voltage or current source.

VI) .ac

When this command is invoked, a dialog window is opened asking the user to choose from one of the three analyses types -- DEC, OCT, and LIN, which represent decade variation, octave variation, and linear variation, respectively. Three values are requested, which are N, FSTART, and FSTOP respectively. N is the number of steps of the analysis, FSTART is the starting frequency and FSTOP is the stopping frequency.

VII) .tf

Since the ".tf" option in SPICE requires an input source, the user needs to supply an argument for this command. Move the mouse on top of a voltage source, current source, or a voltage supply. Invoke this command by menu or by typing ".tf" while the mouse is still on top of the source. When this command is invoked, a dialog window is opened to ask if the user wants to refresh the node values in the schematics. This option is provided because the user may have not had any spice runs or some additional components are added to the circuits. Therefore, the nodes may be out of date if the above situations arise. So, the user can press 'yes' to refresh the node values if those are the cases. Then, a dialog window is opened to ask for the OUTVAR name. OUTVAR is the small-signal output. Examples are V(3), I(5), V(I3), I(VSUPPLY2), etc. The Spice file contains the line <.tf OUTVAR INSRC>, where OUTVAR is the output variable entered through dialog window and INSRC is the source input selected by schematics as argument to this command.

VIII) .sens

When this command is invoked, a dialog window is opened to ask if the user wants to refresh the node values in the schematics. This option is provided because the user may have not had any spice runs or some additional components are added to the circuits. Therefore, the nodes may be out of date if the above situations arise. So, the user can press 'yes' to refresh the node values if those are the cases. Then, a dialog window is presented to ask for the output variables required. SPICE will determine the dc small-signal sensitivities of each specified output variable with respect to every circuit parameter. For example, V(9) V(4,3) V(17) I(VSUPPLY6) may be entered. Do not enter commas between the output variables.

IX) .disto

Since a load resistor is required for this Spice option, the user needs to enter a resistor as a schematic argument for this command. Move the mouse on top of the resistive load that you need to select and invoke this command by menu or by typing ".disto". Then, a dialog window is opened to ask for the <INTER <SKW2 <REFPWR <SPW2>>>>. INTER is the interval at which the summary printout of the contributions of all nonlinear devices to the total distortion is to be printed. The analysis is performed assuming that one or two signal frequencies are imposed at the input; let the two frequencies be f1 (the normal analysis frequency) and f2 (=SKW2*f1). The program then computes the following distortion measures: HD2 (2 * f1), HD3 (3 * f1), SIM2 (f1 + f2), DIM2 (f1 - f2), and DIM3 (2 * f1 - f2). If INTER is omitted or set to zero, no summary printout is made. REFPWR is the reference power level used in computing the distortion products; a default value of 1mW is used. SKW2 is the ratio of f2 to f1. If omitted, a value of 0.9 is used. SPW2 is the amplitude of f2. The default value is 1.0.

X) .noise

Since .noise requires an input source, this command needs a source (voltage source, current source, or voltage supply) as argument. Move the mouse on top of a source and then invoke the command by STEPS menu or by typing ".noise". When this command is invoked, a dialog window is opened to ask if the user wants to refresh the node values in the schematics. This option is provided because the user may have not had any spice runs or some additional components are added to the circuits. Therefore, the nodes may be out of date if the above situations arise. So, the user can press 'yes' to refresh the node values if those are the cases. Then, a dialog window is opened to ask for the OUTVAR and NUMS. OUTVAR is the output voltage which defines the summing point. NUMS is the summary interval. If NUMS is zero, no summary printout is made.

XI) .tran

Once this command is invoked, TSTEP, TSTOP, TSTART, TMAX, and UIC are asked. TSTEP is the printing or plotting increment for output. TSTOP is the final time, and TSTART is the initial time. TMAX is useful when one wishes to guarantee a computing interval which is smaller than TSTEP. UIC is defaulted to N (no). If the user types Y instead, UIC is also presented. TSTART, TMAX, and UIC are optional arguments.

XII) .four

When this command is invoked, a dialog window is opened to ask if the user wants to refresh the node values in the schematics. This option is provided because the user may have not had any spice runs or some additional components are added to the circuits. Therefore, the nodes may be out of date if the above situations arise. So, the user can press 'yes' to refresh the node values if those are the cases. Then, the frequency and the output variable(s) are asked. The Fourier analysis is done through SPICE and/or NUTMEG.

XIII) .print

When this command is invoked, a dialog window is opened to ask if the user wants to refresh the node values in the schematics. This option is provided because the user may have not had any spice runs or some additional components are added to the circuits. Therefore, the nodes may be out of date if the above situations arise. So, the user can press 'yes' to refresh the node values if those are the cases. Then, a dialog window with five options are displayed: namely, DC, AC, TRAN, NOISE, and DISTO. The user can choose one or more of the options and enter the appropriate arguments. Refer to a SPICE manual for the correct form of arguments.

XIV) .plot

When this command is invoked, a dialog window is opened to ask if the user wants to refresh the node values in the schematics. This option is provided because the user may have not had any spice runs or some additional components are added to the circuits. Therefore, the nodes may be out of date if the above situations arise. So, the user can press 'yes' to refresh the node values if those are the cases. Then, a dialog window with five options are displayed: namely, DC, AC, TRAN, NOISE, and DISTO. The user can choose one or more of the options and enter the appropriate arguments. Refer to a SPICE manual for the correct form of arguments.

XV) .ic

This command refers to the initial conditions. When this command is invoked, a dialog window is opened to ask if the user wants to refresh the node values in the schematics. This option is provided because the user may have not had any spice runs or some additional components are added to the circuits. Therefore, the nodes may be out of date if the above situations arise. So, the user can press 'yes' to refresh the node values if those are the cases. Then, another window is opened to ask the user to enter the '.ic' options. For example, if the user wants the initial conditions to be 3V at node 1, 4V at node 2, and 5V at node3, he should enter "V(1)=3 V(2)=4 V(3)=5", with a space between each entry. This is in the same format when using SPICE directly. On subsequent SPICE runs, the .ic that the user has entered is recognized. To cancel individual nodes, simply delete the corresponding entries. To delete .ic totally, simply invoke '.ic' and press 'cancel' when the node values are requested.

XVI) .nodeset

This command is the same as '.ic' except that '.nodeset' is utilized.

XVII) user-options

The user has two options to enter their own text entry. He can simply prepare an include file called ~/.steps-options in their home directory or enter the commands through dialog windows. Once this command is invoked, the user is asked to select whether to 'activate' or 'deactivate' the 'user-options' command. Once it is activated, the user is asked to select whether to enter his own text entry in the include file or in the dialog window.

4.7) Display commands

For the user's convenience, STEPS displays several SPICE values after the user has invoked "spice-file" or after the user has edited the models through "model-edit". The displayed parameters are divided into two sections. One section provides model values, which displays model names and the model values. For example, a resistor may display r1 and 100 which means the resistor is named r1 and has the value of 100 ohms in the Spice input file. An nmos may display M3 and 4/2 which means the nmos is named M3 and has the Width/Length parameter of 4/2. Similarly, an npn device may display Q2 and x3 which means the npn is named Q2 and has the area value of 3 times the normal values. With a different color (black versus blue and red), the node numbers are also displayed after "spice-file" is ran.

Unfortunately, due to the limitation of the X font of the X window, the values may not be printed (a small box is printed instead) if the models are zoomed too small. Also, some users may not want to see such values since they may think that the values are distracting. STEPS allows the users to control the display through the commands below.

I) highlight-off

When the users run "spice-file" or equivalent analysis, STEPS automatically checks whether if the circuit contains nodes that have less than two connections. Then, the respective models are highlighted. This command is used to turn off the highlighted models. If the user runs "spice-file" again, the highlights will be automatically turned off. The user does not need to run this command everytime.

II) model-name-off

Turn off all the model names. The default is on.

III) model-name-on

Turn on all the model names. The default is on.

IV) model-value-off

Turn off all the model values. The default is on.

V) model-value-on

Turn on all the model values. The default is on.

VI) node-value-off

Turn off all the node values. The default is on.

VII) node-value-on

Turn on all the node values. The default is on.

VIII) all-value-off

Turn off all the display values, including the model values, model names, and the node values. The default is on.

IX) all-value-on

Turn on all the display values, including the model values, model names, and the node values. The default is on.

4.8) HELP

I) user-help

This command invokes the online help manual. The online help menu is divided into five sections, namely:

1) Schematic capture - MENU commands

This command lists all of the VEM schematic capture MENU commands and a brief description of each of them.

2) Schematic capture - non-MENU commands

These non-MENU commands are VEM commands which are not uncommonly used and thus are not in the VEM menu system. A brief description of each of them is also presented in the help manual.

3) STEPS

All STEPS commands are given here, with a detailed description for each of them for the convenience of users without users manual of STEPS on hand.

4) STEPS startup

This help menu gives a brief description on how to startup STEPS.

5) STEPS setup

This help menu gives a brief description on how to setup STEPS (how the .Xdefaults file is configured, etc.)

Appendix: VEM COMMANDS

The following are descriptions of all VEM commands for reference purposes. The first symbol is the key-binding. One can access the commands either by invoking the commands through the menu system or just press the key bindings inside the VEM console window.

o: *System - open-window -*

"Creates a new graphics window from an old one or a cell specification"

Ctrl-d: *System - close-window -*

"Closes the window containing the mouse cursor"

?: *System - where -*

"Displays information about the current mouse position"

L: *System - list-palettes -*

"Displays list of available palettes"

P: *System - palette -*

"Opens a window containing a menu of layers or instances"

MENU: *System - re-read -*

"Rereads the contents of the cell from disk"

S: *System - save-window -*

"Saves the contents of the window containing the mouse on disk"

Ctrl-s: System - switch-facet -

"Replaces the cell in this window with one you specify"

b: System - bindings -

"Displays the key, menu, and type-in bindings for a command"

p: Display - pan -

"Pans the window so that the mouse (or point) is the new center"

z: Display - zoom-in -

"Zooms in by a factor of two or to the provided box"

Z: Display - zoom-out -

"Zooms out by a factor of two or in proportion to the provided box"

e: Display - toggle-expansion -

"Toggles between the 'contents' and 'interface' facets of instances"

f: Display - show-all -

"Zooms the window to show all of the cell's contents"

=: Display - same-scale -

"Makes the window containing the mouse the same size as the one with a point" *i:* *Display - push-master -*

"Opens a new window looking at the master of instance under the cursor"

MENU: *Options - window-options -*

"Configure window specific display options"

MENU: *Options - layer-display -*

"Displays a dialog allowing you to change which layers are displayed"

Ctrl-p: *Props&Bags - show-property -*

"Displays the properties attached to object under cursor"

Ctrl-e: *Props&Bags - edit-property -*

"Edits properties attached to object under cursor"

B: *Props&Bags - select-bag -*

"Selects the bag containing the object under the cursor"

Ctrl-b: *Props&Bags - select-bag-contents -*

"Selects the items in the bag containing the object under the cursor"

MENU: *Props&Bags - create-bag -*

"Creates a new empty bag"

MENU: *Props&Bags - attach-to-bag -*

"Attaches selected items to a previously created bag"

MENU: Props&Bags - detach-from-bag -

"Detaches selected items from a previously created bag"

U: Undo - undo -

"undo the last operation"

c: Edit - create -

"Creates new instances, segments, or formal terminals"

D: Edit - delete-objects -

"Deletes existing geometry"

n: Edit - name-objects -

"Changes the name of instances, nets, bags, properties, and terminals"

I: Edit - replace-instance -

"Replaces instances with other instances in symbolic"

E: Edit - edit-label -

"Creates or edits labels"

s: Selection - select-objects -

"Selects items inside boxes, under points, and that intersect lines"

Ctrl-n: Selection - select-net -

"Highlights the net containing the object under the cursor"

Ctrl-t: Selection - select-terms -

"Selects formal terminals"

u: Selection - unselect-objects -

"Unselects items that were previously selected (use control-W to unselect all)"

t: Selection - transform -

"Transforms selected objects in place according to supplied specification"

m: Selection - move-objects -

"Moves selected geometry to a new location or orientation"

MENU: Selection - copy-objects -

"Copies selected geometry to a new location or orientation"

M: Selection - drag-instance -

"Drags selected instances and segments connected to them"

R: Application - rpc-any -

"Starts an rpc application given host and path"

r: Application - RPC-SPICE -

"steps - See steps help manual for details"

V: Version -

"Displays the current VEM version"

Ctrl-l: redraw-window -

"Redraws the window containing the cursor"

(: push-context -

"Pushes the current argument list allowing you to enter new ones"

): pop-context -

"Pops the current argument list replacing it with the pushed one"

CMD: log-bindings -

"Outputs all alias, menu, and key bindings to log file"

Ctrl-c: interrupt -

"Interrupts a graphics window preventing redraw"

W: write-window -

"Saves the contents of the window containing the mouse to an alternate location on disk"

S: save-all -

"Allows you to choose which buffers of all buffers to save on disk"

CMD: rr-nc -

"Rereads the contents of the cell without confirmation (dangerous)"

CMD: *deep-reread* -

"Rereads the contents of the cell recursively"

CMD: *recover-facet* -

"Replaces current cell with an alternate version saved earlier."

CMD: *kill-buffer* -

"Removes all traces of the cell in the window containing the mouse"

CMD: *kill-application* -

"Stops any RPC application running in the window containing the mouse"

c: *create-geometry* -

"Creates new boxes, paths, polygons, and labels"

a: *alter-geometry* -

"Replaces existing geometry with one you specify"

C: *create-circle* -

"Creates new circles"

I: *create-instance* -

"Places instances of other physical cells at a specified spot" *w:* *set-path-width* -

"Sets the default path width on a layer-by-layer basis"

T: *create-terminal* -

"Creates a new formal terminal"

l: *change-layer* -

"Changes the layer of existing geometry to one you specify"

D: *delele-objects* -

"Deletes selected objects"

T: *promote* -

"Promotes a set of actual terminals selected using select-term"

l: *change-segment* -

"Changes the layer and/or the width of selected segments"

l.: *select-layer* -

"Line select-objects but conditionalizes on a layer"

CMD: *alt-buffer* -

"Places another facet in the background of the window containing the cursor"

CMD: *swap-bufs* -

"Swaps the background and foreground facets of a window"

Ctrl-r: *rpc-reset* -

"Resets the RPC state of a window if the application dies unexpectedly"

c: *create* -

"Creates new primitives, wires, and formal terminals"