PC based Simulation Software Package for an Uninterruptable Power Supply

A thesis submitted to the faculty of

Virginia Polytechnic Institute and State University

in candidacy for the Degree of Master of Science

by

Rajiv D. Parikh

in

Electrical Engineering

Dr. Krishnan Ramu, chair

Dr. Peter Athanas

Dr. F. G. Gray

Blacksburg, Virginia

May, 1993

Abstract

The uninterruptible power supply (hereafter referred to as UPS) has become an integral part of modern computer and communication systems. Various topologies have come to the fore in recent times with the advent of high frequency switching techniques. This project considers such a high frequency operated UPS system. It is modeled, and simulated to provide insight and information for design studies. A framework for the software simulation of a class of UPS is proposed in this document and its implementation results are given. The simulation and the user interface software structures and their implementations are also included. It is intended that the software can be modified on a modular basis to enable the simulation of various topologies of UPS that are in vogue and that may emerge in the future.

Acknowledgement

The author would like to thank Dr. Krishnan Ramu for providing him with the opportunity to do this work. The author would also like to thank Dr. F. G. Gray and Dr. Peter Athanas for taking time out of their busy schedules to be on the defense committee.

The author dedicates this work to his parents Dilip and Jyotsana Parikh, and brother Sanjiv for putting up with all the long discussions about continuing education and supporting every decision that the author has made. Special thanks goes to all the author's friends at Virginia Tech.

Additionally, the author thanks the members of the Motion Controls and Systems Research Group for proof checking the equations used in the work.

Table of Contents

List of Figures

List of Symbols

Components and Parameters

L_i	Inductor at AC input.
L_f, C_f	Inductor and capacitor used in LC filter in AC-DC stage.
L_{lp}, R_p	Primary side inductance and resistance of the transformers.
L_m	Magnetizing inductance of the transformer.
L_{ls}, R_s	Secondary side inductance and resistance of the transformers.
C_b	Battery model capacitance.
R	Load for AC-DC stage simulation (in parallel with battery).
C	Filter capacitor at output of transformer in DC-AC stage.
L_l	Load inductance in DC-AC stage.
C_l	Load capacitance in DC-AC stage.
R_l	Load resistance in DC-AC stage.
K_v	Voltage scale factor in feedback.
K_c	Current scale factor in feedback.
K_{p1},K_{i1}	Constants for the voltage error PI block in both control blocks.
K_{p2},K_{i2}	Constants for the current error PI block in the AC-DC control block.
K_{p2}	Scaler for current control command in DC-AC control block.
K_{p3}	Scaler for voltage control command in DC-AC control block.
β	PWM scaler in AC-DC control.
α	PWM scaler in DC-AC control.
v_{dd}	Diode drop voltage used in rectifiers and inverters (set by the user).

Signals

v_{ac}, i_{ac}	AC input voltage and current.
v_i	Input voltage of rectifier at AC input in AC-DC stage.
v_d, i_f	Input voltage and current of LC filter in the AC-DC stage.
i_{cf}	Current through the C in the LC filter in the AC-DC stage.
v_{dc}	Input voltage to the inverter in the AC-DC stage.
v_b	Voltage of the battery, in between the AC-DC stage and DC-AC stage.
i_{inv}	Input current to the inverter.
v_p, i_p	Input voltage and current of the transformers.
v_m, i_m	Voltage and current through L_m.
v_1, i_s	Voltage and current at the output of the transformers.
i_b	Current through the battery.
i_r	Current through load R in the AC-DC simulation.
i_c	Current through the filter capacitor C in DC-AC stage.
i_{l1}, i_{l2}	Input and output current of the rectifier at load in the DC-AC stage.
v_{rs}	Output voltage of the rectifier at the load in the DC-AC stage.
i_{cL}	Current through the load capacitor C_l.
i_l	Current through the load resistor R_l.
v_s	Voltage at the load resistor R_l.
v_b*	Commanded battery voltage.
v_{b1}	Fedback battery voltage.
v_{er}	Error in fedback voltage signal in both control blocks.
i_b*	Calculated (expected) battery current fedback.
i_{b1}	Actual fedback battery current.
i_{er}	Error in fedback current.

v_c* Control signal for AC-DC stage.

v_s* Commanded output load voltage.

v_{s1} Fedback load voltage.

v_{s2} Rectified and filtered vs1.

v_1 Output of voltage PI block in DC-AC control.

v_2 Sinusoid modulated by v1.

i_1 Calculated load current fedback.

i_{c1} Actual load current fedback.

v_3 Voltage control signal due to error in current fedback.

v* Voltage control signal (sum of v3 and v1).

T1-T4 Inverter control switch signals.

Note:

- Those components and signals with a prime (') apostrophe mark have been brought over to the primary side of the transformer from the secondary side.
- Initially, all signal have the value of zero except v_b, which the user sets.

CHAPTER 1

Introduction - The UPS

Various low power (less than 1 kVA) high frequency topologies for the uninterruptable power supply (UPS) are reviewed in [1]. These topologies are of recent origin and many are in the research stages. The key to successful realization of a new UPS topology and system are software tools. Such software tools for the modeling, simulation and analysis of UPS systems are rare. This work addresses such a need for an important segment of the power electronics industry.

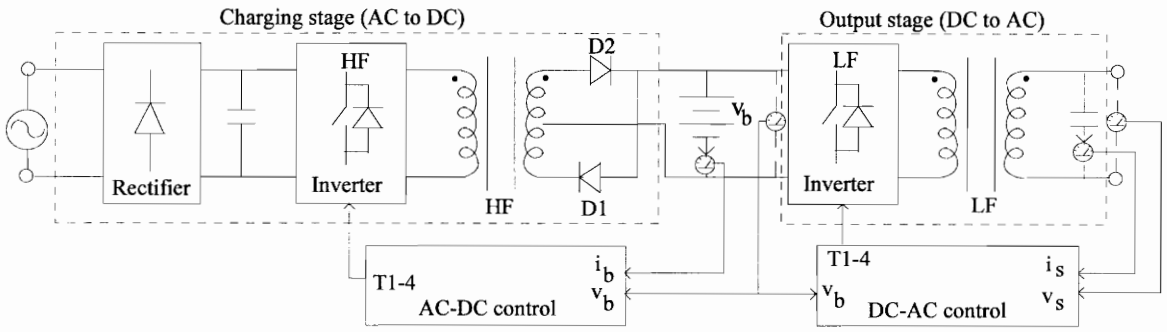

Figure 1. UPS Topology

Figure 1 shows the UPS system chosen for study. Since it has more than one or two power stages, this would provide an ideal choice for developing a generalized but modular software structure. Note that both at the front-end and at the output, high frequency switching provides light weight isolation and achieves high quality output waveform control. The charging stage is separately shown in figure 2 with all the

subsystem components such as filters, rectifier bridge, inverter, high frequency transformer, battery, etc. The output stage is shown in figure 3 separately.

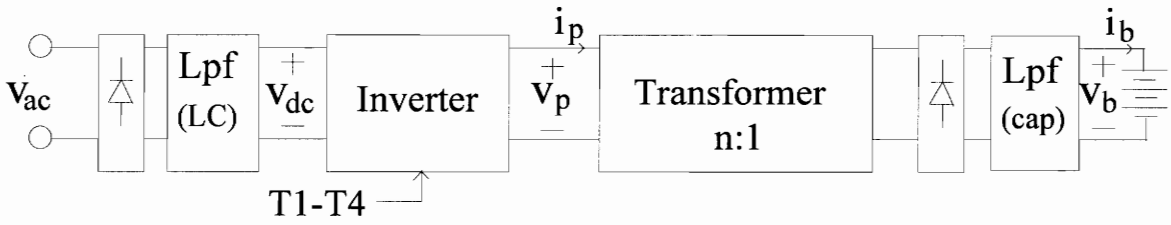

Figure 2. Block diagram for AC to DC conversion

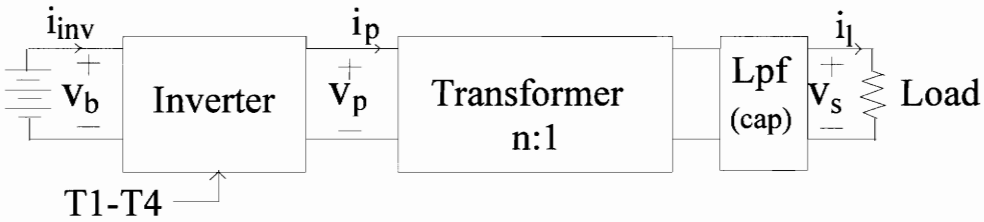

Figure 3. Block diagram for DC to AC conversion

The description of the function of each of these blocks can be found in most modern power or control texts and thus, will be omitted from this discussion.

This document is organized as follows. Chapter 2 gives the model derivation for the UPS. The control configurations for both the charger and the output inverter are given in chapter 3. A generalized but modular software structure for the simulation of the UPS is presented in chapter 4. Implementation details of the software are given in chapters 5 and 6. Chapter 7 describes the test simulation results. Chapter 8 contains the conclusions of the study.

CHAPTER 2

Model Derivation

The block diagrams of the AC to DC and DC to AC conversion stages shown in figures 2 and 3, are used to develop the circuit models of various standard components and are shown in figures 4 and 5, respectively. The subsystem models are given below. A slightly modified schematic of the AC-DC stage as implemented in the simulation software is given in figure 6. Also, the DC-AC stage with a more realistic load is in figure 7.

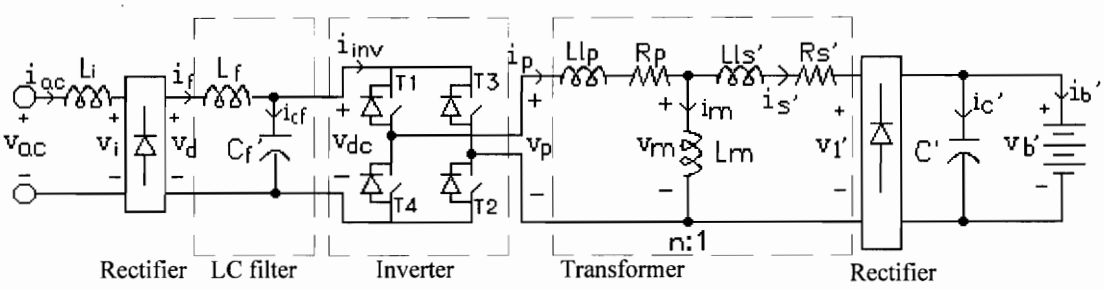

Figure 4. Circuit model for AC to DC conversion

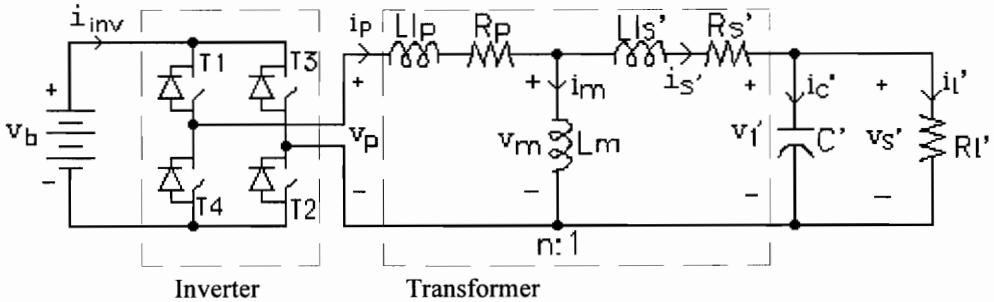

Figure 5. Circuit model for DC to AC conversion

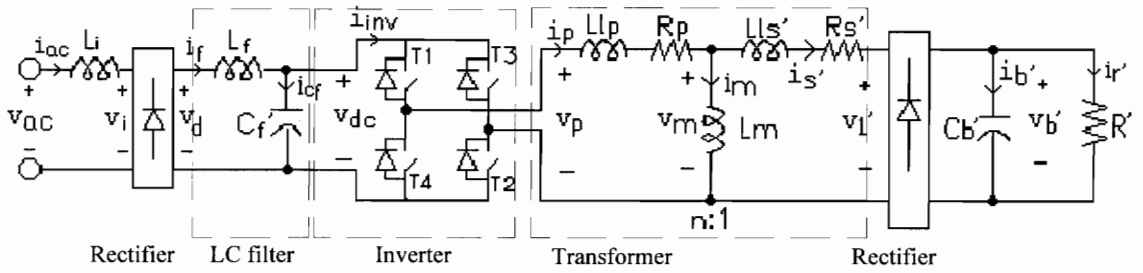

Figure 6. AC-DC stage model for simulation.

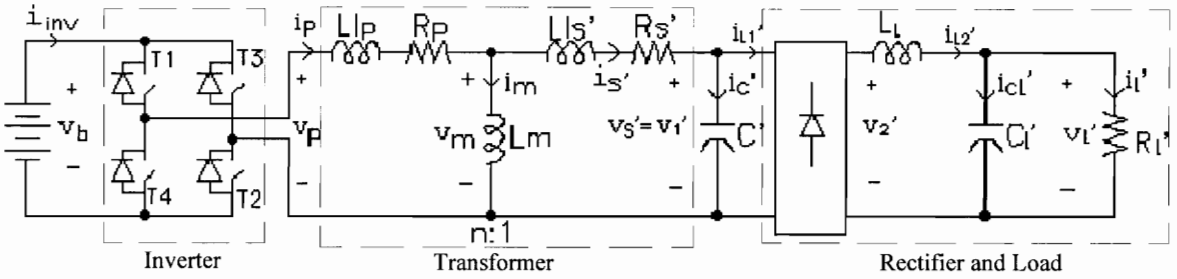

Figure 7. DC-AC stage model with rectified load for simulation.

2.1 Inverters

The inverter switches are modeled as ideal in terms of their switching characteristics. However, the voltage drop during (reverse current) conduction is accounted for. The input-output relationship of the inverter depends on the switching gate drive signals and the polarity of the currents in the load. A complete input-output relationship is given in Table 1.

This table is converted to a conditional structure in software. Subconditions to account for the diode drop (v_{dd}) during reverse current flow are also included in the table. The inverters in both stages share the conditional construct, only the variables are different.The input parameters are the switch control signals (T1, T2, T3, and T4), the input voltage v_{in} (v_{dc} for AC-DC stage, v_b for DC-AC stage), and the current i_p. The output is voltage v_p and current load to the input block (i_{inv}). This allows the simulation software to 'disconnect' the inverter in case the diode drop condition is not satisfied.

Table 1. Inverter input-output relationship

Switch state				Input		Output	
T1	T2	T3	T4	i_p	v_{in}	v_p	i_{inv}
ON	ON	OFF	OFF	>=0	---	v_{in}	i_p
				<0	>0	v_m	0
					$< -2v_{dd}$	$v_{in}+2v_{dd}$	i_p
					$>= -2v_{dd}$	v_m	0
OFF	OFF	ON	ON	<0	---	$-v_{in}$	$-i_p$
				>=0	>0	v_m	0
					$< -2v_{dd}$	$-v_{in} -2v_{dd}$	$-i_p$
					$>= -2v_{dd}$	v_m	0
OFF	OFF	OFF	OFF	---	>=0	v_m	0
				>=0	$< -2v_{dd}$	$-v_{in} -2v_{dd}$	$-i_p$
					$>= -2v_{dd}$	v_m	0
				<0	$< -2v_{dd}$	$-v_{in} -2v_{dd}$	i_p
					$>= -2v_{dd}$	v_m	0

2.2 Transformers

Figures 4 and 5 (and 6 and 7) show the equivalent circuits for the transformers. The following equations describe the operation.

The two loop equations are:

$$R_p i_p + L_{lp} \frac{di_p}{dt} + L_m \frac{di_m}{d} = v_p \quad (1)$$

and

$$R_s' i_s' + L_{ls}' \frac{di_s'}{dt} - L_m \frac{di_m}{dt} = -v_1' \quad (2)$$

and the node equation,

$$i_p = i_m + i_s' \quad (3)$$

2.3 Rectifiers

The rectifier bridges are modeled as absolute value circuits and their operation can be described by the following equations.

The battery voltage in AC to DC conversion stage is given by,

$$\text{if } |v_1| > 2 \cdot v_{dd}, \text{ then } v_b = |v_1| - 2 \cdot v_{dd}, \text{ else } v_b \text{ disconnected.} \quad (4)$$

and the DC input voltage to the inverter is

$$\text{if } |v_i| > 2 \cdot v_{dd}, \text{ then } v_d = |v_i| - 2 \cdot v_{dd}, \text{ else } v_d \text{ disconnected.} \quad (5)$$

where v_{dd} is the diode drop, v_1 is output of the transformer, and v_i is the input of the rectifier at the AC side of the stage. Note that the voltages used in the comparisons are not brought to the primary side. A similar equation is used for the load in figure 7.

2.4 Filters

The filters are modeled by their differential form and are added to the rest of the equations to obtain the system equations. The LC filter in the AC to DC stage is described by,

$$v_d = L_f \frac{di_f}{dt} + v_{dc} \quad (6)$$

and

$$i_f = C_f \frac{dv_{dc}}{dt} + i_{inv} \quad (7)$$

where i_f and v_d are the input current and voltage, and i_{inv} and v_{dc} are the output current and voltage.

The following three equations describe the load and capacitor in the DC to AC circuit of figure 7:

$$i_c' = i_c' - i_{l1}' = C' \frac{dv_1'}{dt} \quad (8)$$

$$i'_{cL} = C'_1 \frac{dv'_s}{dt} = i'_{l2} - \frac{v'_s}{R'_1} \tag{9}$$

$$L'_1 \frac{di'_{l2}}{dt} = v'_2 - v'_s \tag{10}$$

2.5 Battery

The battery is modeled simply as a capacitor for overall system evaluation. This simple model is used to illustrate the charging and discharging characteristics without dealing with the complications of deriving a more complete model. As the results show, this model is sufficient for this simulation software.

The equation is,

$$i_b = C_b \frac{dv_b}{dt}. \tag{11}$$

It should be noted that this equation holds true when primed (ie.v_b') variables are used. The capacitor C' in parallel with the battery is combined with the battery capacitance to simplify the equations.

For AC-DC stage simulation, a slightly different schematic is used to provide a load to the stage for testing purposes. This configuration is shown in figure 6.

This introduces a new equation for the loop including the battery and the resistor as shown below.

$$v_b = R \cdot i_r \tag{12}$$

2.6 Combined Equations

To make the model easier to simulate, the transformers of both stages are combined with the circuits at their outputs. That is, in the AC-DC stage, the equations for the transformer are combined with the equations for the rectifer (and eventually the capacitive model of the battery).

At the input of the AC to DC stage, a conditional set of equations is created to include the inductor L_i as part of the filter with different rectifier states (on, off). For example, if the rectifier's output is non-zero then L_i and L_f are in series and the sum L_i + L_f is used instead of L_f in equations 6 and 7. If however, the output of the rectifier is zero (i.e. the input voltage is not greater than $2v_{dd}$) then the filter and hence, the rest of the circuit is disconnected from the AC input.

At the output of the AC to DC conversion stage, equation 4 is combined (software) with

$$i_c' + i_b' = \left| i_s' \right| \tag{13}$$

So, for AC to DC stage, equations 1 through 7, 11 through 13 describe the analog part of the circuit. In the case of the DC to AC conversion stage, equations 1 through 3 and 8 through 10 describe the analog circuit. The switches in the inverter can be considered "pseudo-digital" for simplicity, ie., either on or off.

CHAPTER 3

Control Configuration and Algorithms

The feedback control circuits for both power stages are given in figures 8 and 9. Notice the triangular signals for the two circuits. For the AC to DC stage, it is from zero to maximum, while for the DC to AC circuit, it is from negative maximum to positive maximum. Both are of the same frequency in this simulator, however, they can easily be made different by modifying the parameters file. A constant voltage is compared with the feedback in each circuit. The following sections describe the control circuits. The derivation of the circuits can be found in the references.

3.1 AC to DC

The AC to DC control circuit is given in figure 8. The commanded battery voltage, v_b* is compared with the feedback from the battery and this error signal is passed through a PI block with a limiter. This signal is now compared with the feedback current signal

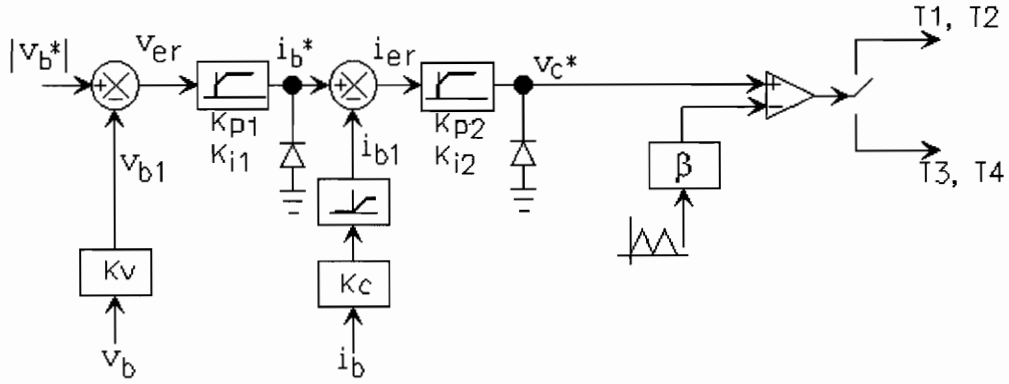

Figure 8. Control circuit for AC to DC conversion

for current limiting and a similar action is performed with the error signal. The emerging control signal is converted to a pulse width modulated (PWM) signal via a comparator and every alternate pulse is passed to T1,T2 and T3,T4 inverter switch gate drives.

3.2 DC to AC

The DC to AC control circuit in figure 9 is considered from [2]. The operation is as follows. The feedback voltage is compared to commanded (required) voltage and the error is amplified through a PI block with a limiter. This signal, v_1 is modulated by a sinusoid (60Hz) to give v_2 and compared to the feedback signal from the load, v_s. Next, it is scaled and added to the difference between the product of v_1 and a cosinusoid, representing the current feedback from the filter capacitor. The cosine term is used to cancel the effect of the filter current. The resulting sum is scaled and added to v_2 to give the command signal for the inverter inputs. Here, the scaled battery voltage is compared to v* for the inverter signals.

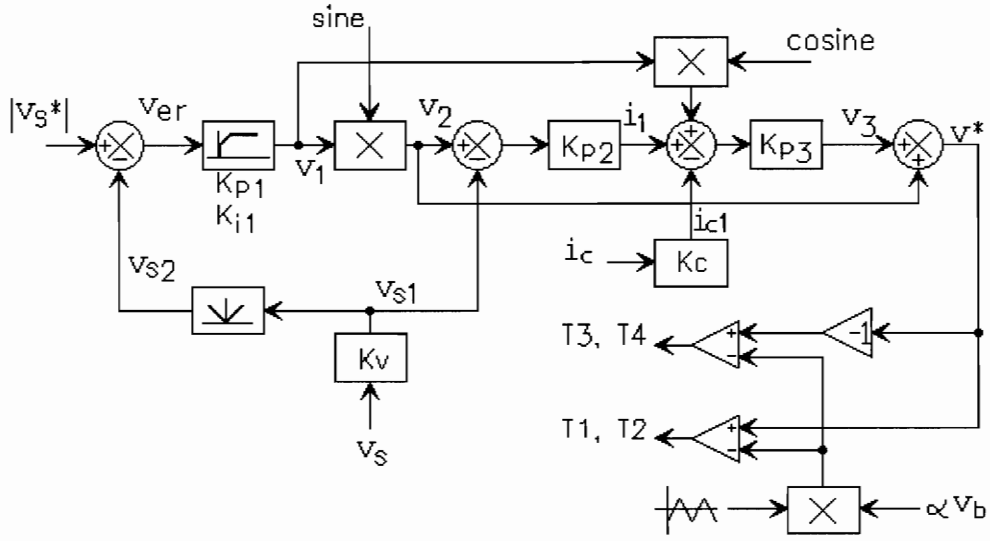

Figure 9. Control circuit for DC to AC conversion

In the proportional-integral (PI) blocks, Simpson's rule for integration is utilized for speed of computation. For example, for PI block with v_{er} as input, the implementation would look like the following:

$$v_{er\_int} = v_{er\_int} + v_{er}/dt$$

$$v_{out} = K_{p1} * v_{er} + K_{i1} * v_{er\_int}$$

Here dt is the time for each iteration, and v_{er_int} is the running integral of v_{er}. A model similar to that described in section 2.3 is used here for the rectifier as well.

CHAPTER 4

Software Description

The software is broken in to two separate sections-- the simulation engine, and the user interface (UI) as illustrated in figure 10. The simulation engine is the main focus of the first subsection. The user interface is developed after the simulation module is completely coded. The edges indicate the flow of the information. The two modules are shown as the two large boxes. The hooks are represented by small rectangles, as are the data structures since both hooks and data structures allow information sharing. Hooks will be explained in Chapter 6. Both, parameter (prm) and output files are indicated by the drums. The two 'screens' are mutually exclusive, that is, only one can be used in any given execution. The figure shows the relationship of the two modules with each other and all the other entities such as files and the user.

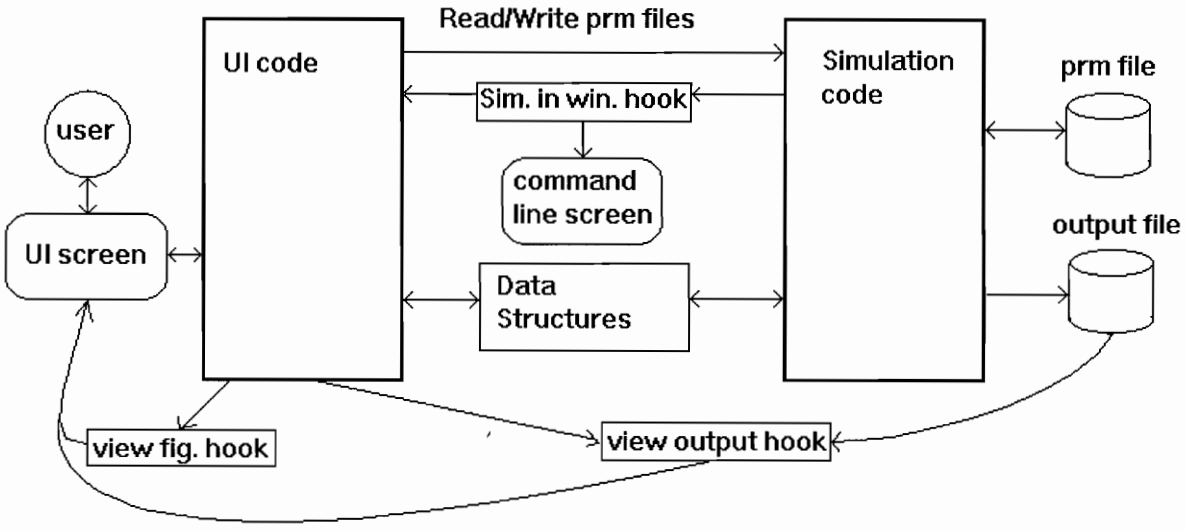

Figure 10. Software flow.

4.1 Simulation Engine

The simulation code is further divided in to six separate sections as shown in figure 11.

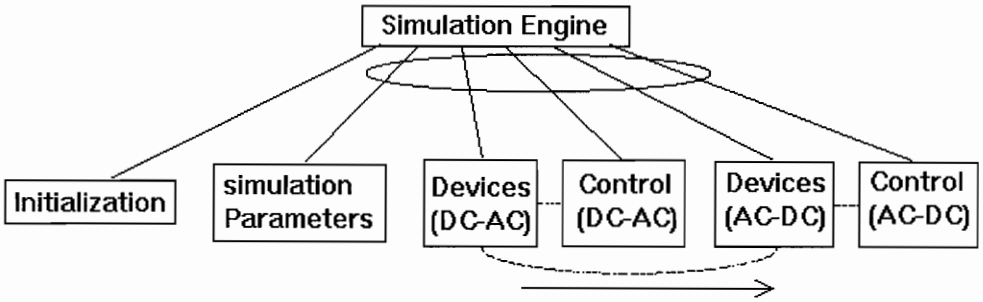

Figure 11. Simulation Engine structure diagram

The arrow at the bottom indicates the order of occurrence in the simulation (i.e. from left to right). The loop around the edges means that those submodules are executed repeatedly. The dotted edges connecting the submodules indicate an existence of an interface between the two modules. From this, a flow chart (figure 12) can be easily derived.

Next, each of the elements in the flow chart are implemented in code. First, for each simulation module, a data structure is constructed, which consist of two subdivisions - parameters that are fixed throughout the simulation and variables (signals) that are dynamic. Also, for each of the "dotted" edges in the structure chart, a data structure is constructed to act as an interface between the two submodules. Following the derivation of the data structures, each of the submodules can easily be converted to code. In some cases, the interface structure and code is combined with that of one of the modules.

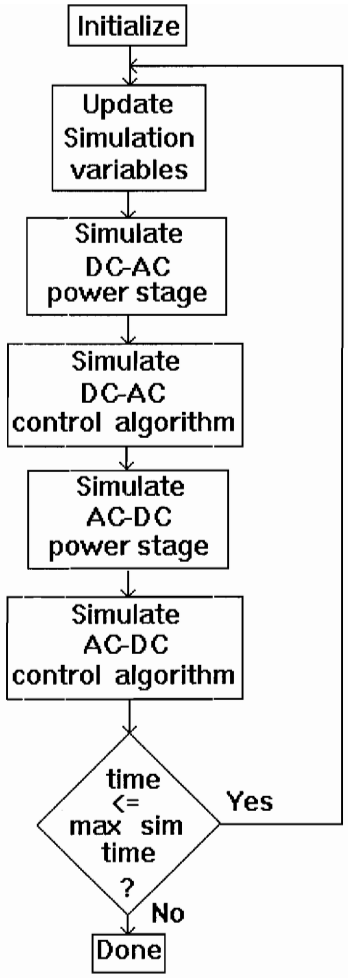

Figure 12. Simulation Engine high level flow chart.

4.2 User Interface

A decision was made to allow the users to execute the simulation without being in the user interface(UI) when a series of simulations were required. This would allow the user to create all the parameter files first and then write a batch file to execute the command line version of the simulator for each parameter file. The user's presence is not necessary after starting the batch process.

This decision also forced the implementation of the simulation code to be written completely independent of the user interface code. Some requirements were established, so the interaction between the two large modules could easily be accomplished.

The first requirement was that all the output that the command line version produces for the screen (messages, etc..) must be done using a special procedure. This procedure can then be made to print to the window in the UI. This procedure is the 'hook' discussed in section 6.4.

Second, all the parameter file i/o is to be located in the command line code, so any other module (UI) can access the same routines that the simulator itself uses to read-in (write) parameter files. However, for this to be transparent, the UI also needs to see the same data structures that the simulator uses. A simple solution is implemented. That is the entire code for the command line simulator is put in an include file and is included in the UI. This allows the UI to use data structures and routines from the simulator. The main program from the simulator is conditionally compiled depending on whether the UI is used or not.

The interface itself has to be straight-forward and user-friendly. The standard and methodic menus are required to allow the user to navigate easily in this environment. Some quick keys are to be implemented such as exit (Alt-X) to allow the user to quickly perform the action.

An additional hook is provided to view the output of the simulator using an in-house viewing software. However, this software has its own requirements. First, only eight signals can be viewed, so only eight signals should be selected in monitor menus. Next, in each line a maximum of five numbers can exist. This means that for eight signals, the first line will have five and the second line will have four (the very first value for each iteration is the time in simulation) variable values. Lastly, the output file should

not have more than 1600 points (due to the memory requirements for the viewing software). These are taken care of in the implementation.

CHAPTER 5

Implementation of Software - Simulation Engine

This section contains the description of various coding and conversion steps that finally result in a simulation loop. Use is made of C language and structured programming with Borland C [4]. Actual implementations are explained with code samples.

5.1 Data Structures

The first step in coding was to create data structures for all the submodules shown in figure 11. There are six main data structures, four for the submodules, and two for the horizontal edges between device and control block. The curved dotted edge between the two device blocks is the battery status, which is made a part of both the device blocks.

The four large structures are similar in implementation. That is, each is a C language structure (record) with a floating point field for each of its various signals and parameters. The two connection structures are basically the control signals for the inverters and thus are identical except for the names. They are also implemented using records with four fields for T1, T2, T3 and T4. Also all the other global variables such as time and sinewave are declared. More can be declared later as the need arises.

5.2 Special Routines

The next step is to code all the necessary math and wave generator routines. Most required routines are provided in standard C libraries. However, for solving differential equations (in state equation format) a Runge-Kutta 4th (RK4) order routine is implemented. The routine needs to take in the number of state equations and the locations (arrays) where the values are stored. The arrays are passed as pointers to floating point numbers. One other vital information the RK4 routine needs is how to compute the derivatives of the state variables. Instead of encoding this information directly in the RK4 routine, a pointer to the function which takes as input the arrays and computes the derivatives is passed to the RK4 routine. The syntax for the RK4 and the derivative (called Prime) functions declaration is as follows:

```
void RK4(       void (*Prime)(float *X, float *XPrime),
                int NumEqn,
                float *X,
                float *XOld)
{
...
}

void Prime1(float *X, float *XP)
{
...
}
```

The call is made as follows:

RK4(Prime1, N, X, Xo);

The main advantage of passing a pointer to Prime in to RK4 is that RK4 is used more than once in the program and this way you would not have to duplicate the code for RK4 in the program.

The routines for function generation such as triangle wave are made using the current time in the simulation. Sine and other functions are from the C math library.

5.3 Input/Output

File and i/o routines are implemented next. These include opening the output data file and writing the header information. The header information consist of the number of points in the file and the names of the signals that are output to the file. Also, all the simulation parameters and other settings need to be saved and read in, as discussed below. A setting file format (called parameter file, extension .prm) is created which first has the title of the program that created it (this simulator) followed by the general simulation settings such as the output file name and maximum time to simulate as well as what signals are monitored (output to the data file during the simulation). Next follow four blocks of data. Each of these correspond to the parameters in circuit modules in figure 11. For example, the first block contains all the parameters in the AC-DC physical circuit in figure 6. A commented sample setting file is provided at the end of this report. All of this is straight forward C standard i/o function calls for text files. One important

part is how to indicate which signals are monitored and then to output them during the simulation easily.

Each signal is given a number starting at 0 for the current time. This number is the index to an array of pointers to floating point numbers. So, to access time during simulation, simply do the following (varlist is the array):

```
printf("%f", *varlist[0]);
```

This was made even easier since the indices were stored in the parameter file and they are read in into an array of integers (called varInd). Hence the above statement now can be extended to cover all signals that are monitored as follows:

```
for (i=0; i< NumSignalsMonitored; i++)
        printf("%f ", *varlist[ varInd[ i ] ]);
```

5.4 Initialization

In this step all the initialization routines are set in place. The requested parameter file is loaded into the memory using the routines coded in the above step. The output file is also initialized. All the global simulation values such as time are set to their appropriate values (time = 0). All of the four modules are also initialized. For this four separate initialization procedures are made to keep the code modular.

In each one, all the signals are set to their initial values, for example, all currents are zero.

5.5 Simulation

This is the last major coding step. In this step, each of the four circuit modules are coded and the main loop is created. For the device circuit modules, proceed from left to write in the circuits (figures 6 and 7) converting each block to code using the model derived in section 2 of this paper. The state equations are put in separate derivative functions so they can be used by the RK4 routine. The routine for the circuit will call RK4 and pass that derivative function to it. There are three sets of state equations, one for the LC filter in the AC to DC stage, and two for the two transformer circuits. All the floating point arrays are declared in these two routines for the state equations.

Similarly, the routines for the control circuit blocks are coded in. Here, the trickiest part is the PI blocks. A standard algorithm for the PI as mentioned in section 3.2 is used in this software. Once the modules are coded, the last item is to code the main loop. Since the simulation is to stop when simulation time reaches maxtime as specified by the user in the parameter file, the main function will resemble the following:

```
void main()
{
        initialize_everything();
        while ( time < maxtime )
        {
                update_globals();          /* update time, sin, cos,
                                              trangle, etc.... */
                sim_AC_DC_device();        /* figure 4 */
                sim_AC_DC_control();       /* figure 8 */
                sim_DC_AC_device();        /* figure 5 */
```

```
            sim_DC_AC_control();        /* figure 9 */
            print_results();            /* output monitored
                                        signals values to file */
        }
        Clean_Up();                     /* close files, etc */
}
```

5.6 General Comments

As each step is completed, the code is thoroughly tested and any customization is added such as the format of the output file. The output file may be viewed in a spreadsheet where the data can be manipulated and exported to a report. The other approach is to write a data viewer or use an existing one as in this case. The interface between simulation and UI is discussed in the following chapter.

CHAPTER 6

Software Implementation - User Interface

The user interface (UI) for this software for DOS is compiled with Borland C++ 3.1 using Turbo Vision toolbox [3, 4]. Turbo Vision provided enough control over the menus and the dialog screens. First subsection discusses the overall flow of the interface. The next two subsections describe the two regular dialog boxes. The next three describe the three hook procedures mentioned in Section 4.2.

6.1 Event Driven Model

The flow of the interface software is described best by figure 13. The figure shows the action within the large UI code box in figure 10. The hot keys are always active, ie., if you ask for help in any screen, the command will open a help window with appropriate help topic. The exit key is implemented as a partial hot key, since the user should not be able to exit from any of the parameter entry or monitor selection screens.

Event driven means the user initiates an event, using the keyboard or mouse, and the software waits for the user or the current action to halt. More details can be found in reference manuals for Borland C++ [4].

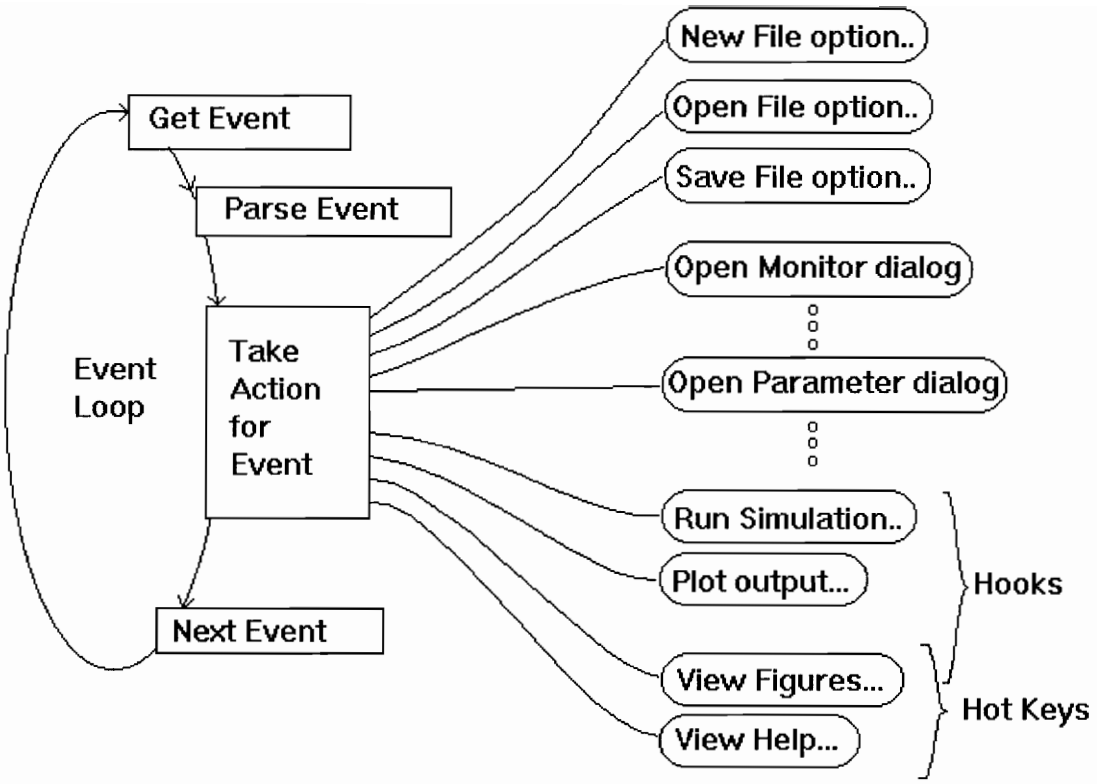

Figure 13. Event Loop

6.2 Parameter Entry and Monitor Selection Dialogs

For this software there are ten dialog screens. Five are for selecting the signals to be monitored and the remaining for entering parameter values and output file name. Each dialog box contains an OK and a Cancel button for user action. For all the parameter entry boxes, the Tab key will move the cursor from one field to the next. In the monitor boxes, the arrow keys will move the cursor from one signal to the next within a column and Tab will jump the cursor from one column to another. Samples screens are shown in figures 14 and 15. As shown in the figures, the program is also Microsoft Windows compatible (i.e. it runs in a DOS window, and allows mouse operations). All of the code is written using standard toolbox objects such as TDialogBox and TInputLine.

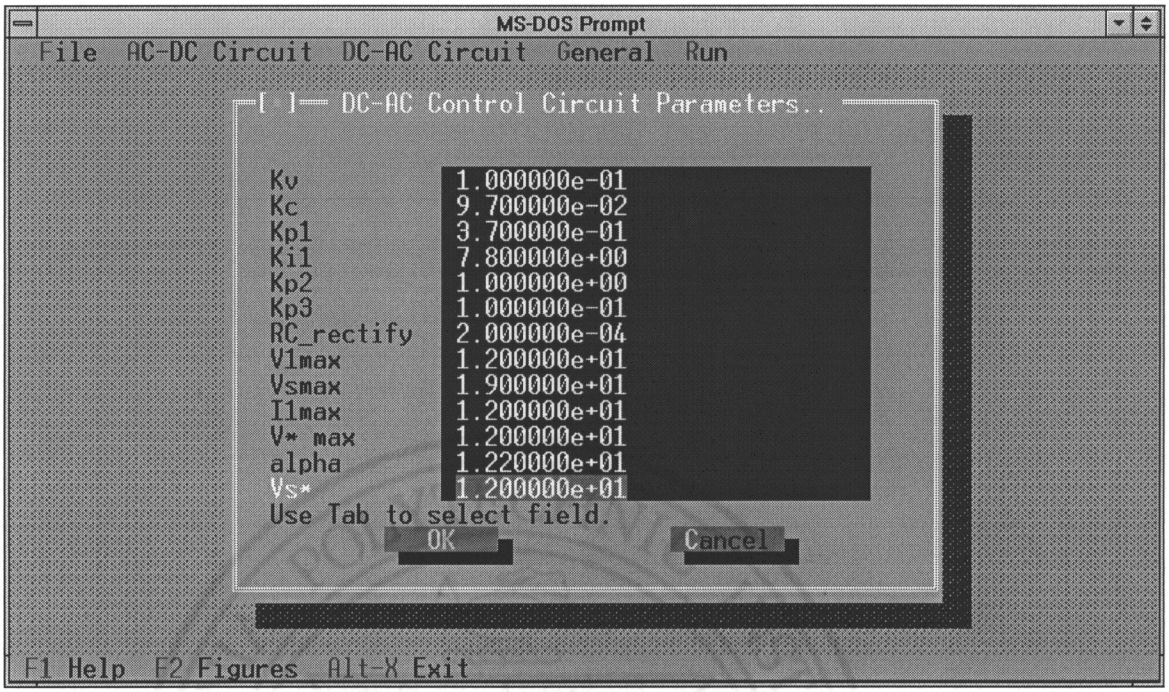

Figure 14. Sample entry dialog box.

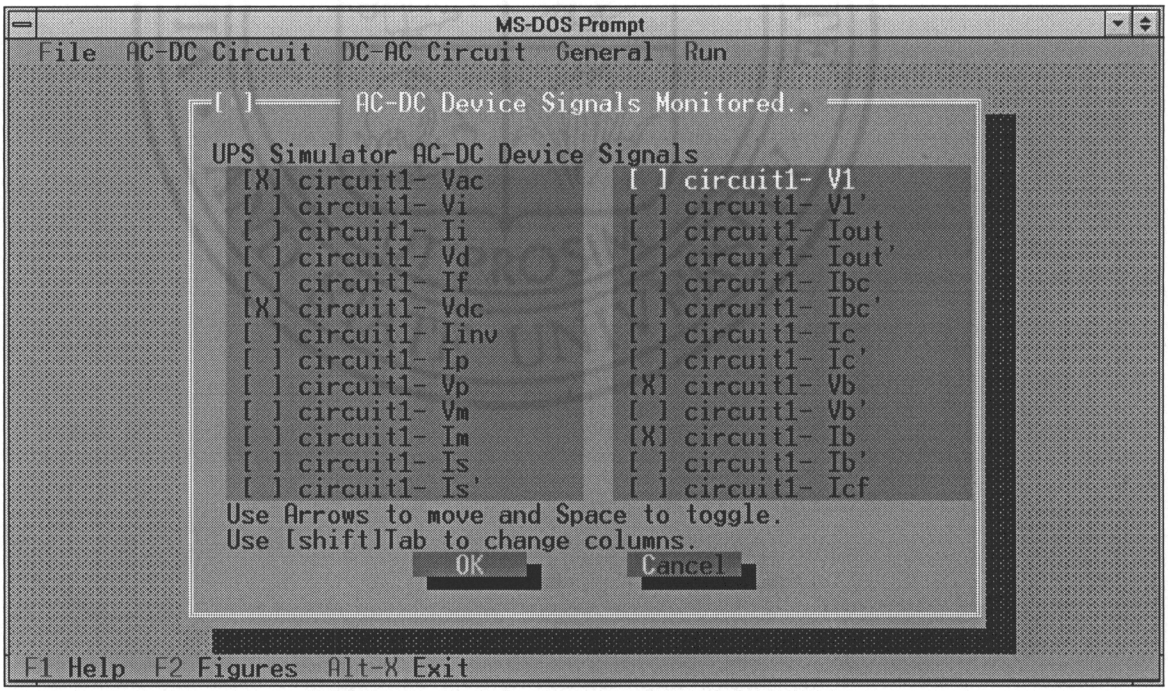

Figure 15. Sample selection dialog box.

6.3 Simulation in a Window

The user interface allows the user to run a simulation from within. However, when the simulation is running, the only key stroke that will have an affect is Control-C, to abort the simulation and return control to the UI code. The simulation in a UI window is accomplished by opening a Turbo Vision "Terminal" window and having an output stream directed to its interior. Then the pointer to this stream is passed to the simulation routine and this routine will pass the pointer to all the routines that do any screen output. This way all the screen output is redirected to the Terminal window in the UI. When the command line version is compiled, the main function for the simulator is also compiled. The main function has an alternate definition of the 'print to screen' function, which simply prints to the DOS screen (stdout, see figures 16 and 17). A key (Alt-F3) is provided to close the simulation window. A sample execution in a window is shown in figure 16 along with the menu which contains the simulation option.

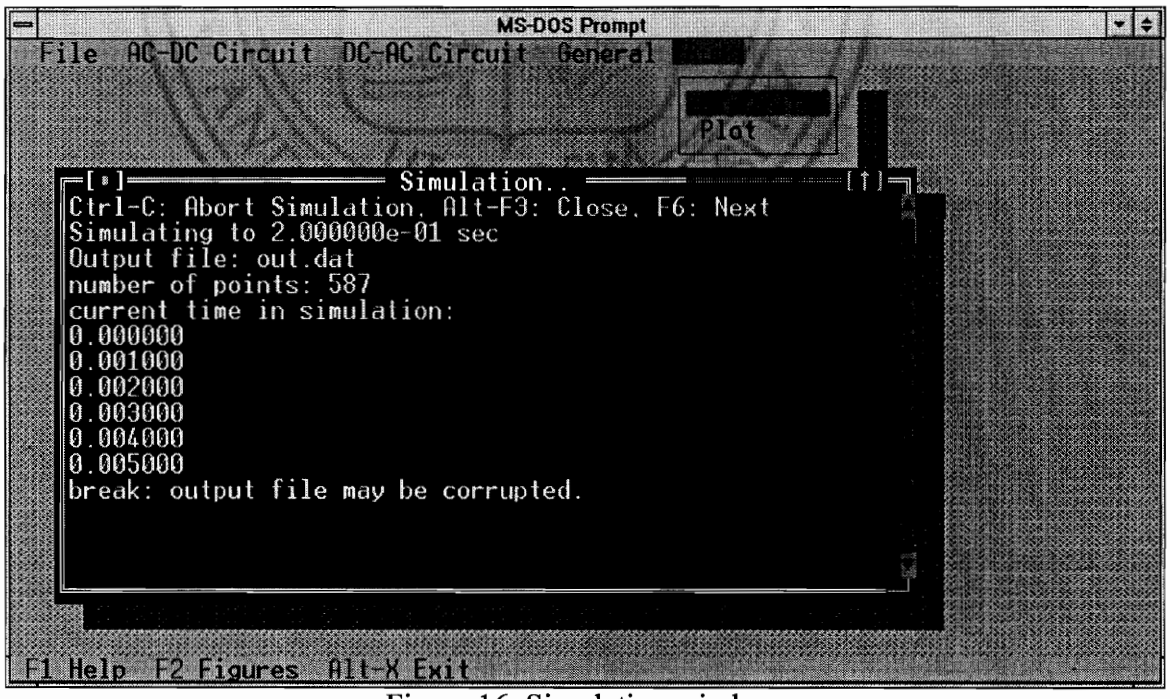

Figure 16. Simulation window.

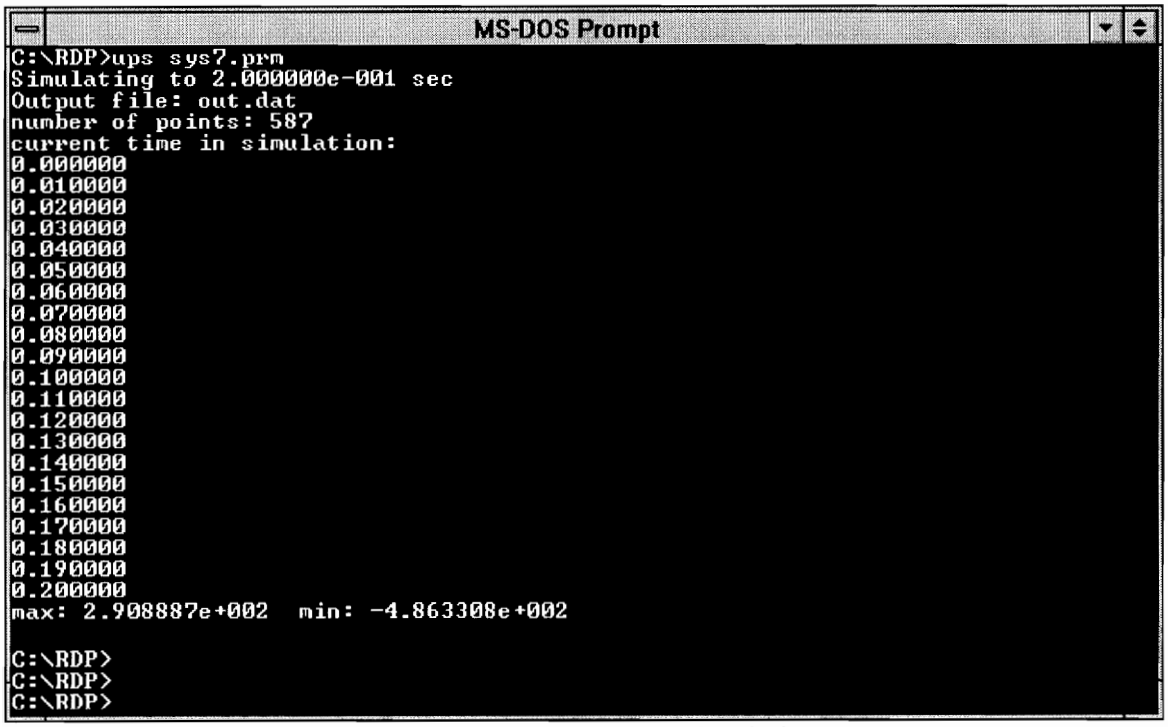

Figure 17. Command line simulation.

The simulation window does not need to be closed after a particular simulation is done. A new window is opened for each simulation however, and so it is a good practice to close those that are not needed anymore. An important feature of running the simulation from within the UI is that everytime the simulation run command is selected, a temporary parameters file (temp$$$$.prm) is created (updated) and that file is loaded by the simulator module and used for simulation. This is useful because the original parameters file is never touched and also if the program crashes (nobody is perfect) the last changes that were made to the parameters are not lost. It is also recommended that the program be restarted after 15 or more simulations have been run from within the UI. This is due to the fact that the memory used by the simulation windows is fragmented once it is allocated and so cannot be reused without time consuming memory defragmentation. In time this lost memory can amount to more than just a few kilobytes.

Restarting the program eliminates this problem since the operating system takes all of the memory back.

6.4 Hook for Viewing Figures

A hot key (F2) is provided for viewing the figures at anytime during the program except while the simulation is running. This is accomplished by checking for the key in main event loop and calling the view figure dialog box (see figure 18). The dialog box will allow the user to select a figure to be viewed and then call an external program to actually load the picture file and display it on the screen (see figure 19). When the user presses ESC key, the control returns to the UI and the display is quickly redrawn to the state it was before the user pressed F2. This scheme also allows the schematics as well as the viewing program itself to change without the code being affected.

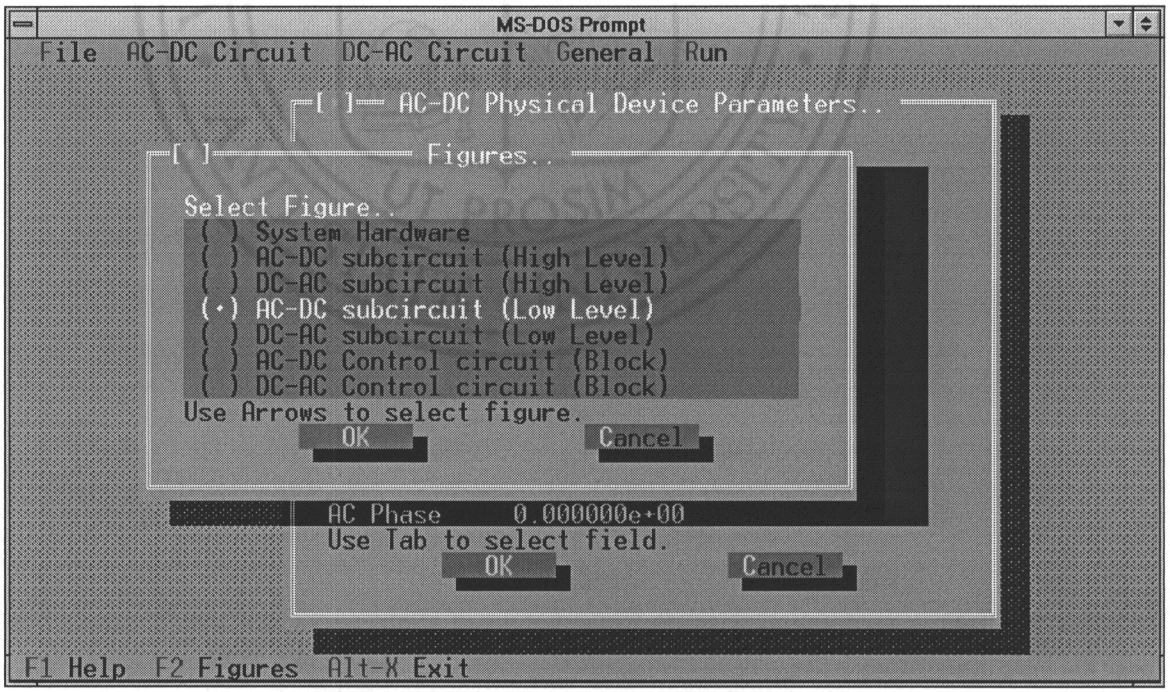

Figure 18 Viewing figures- hot key menu

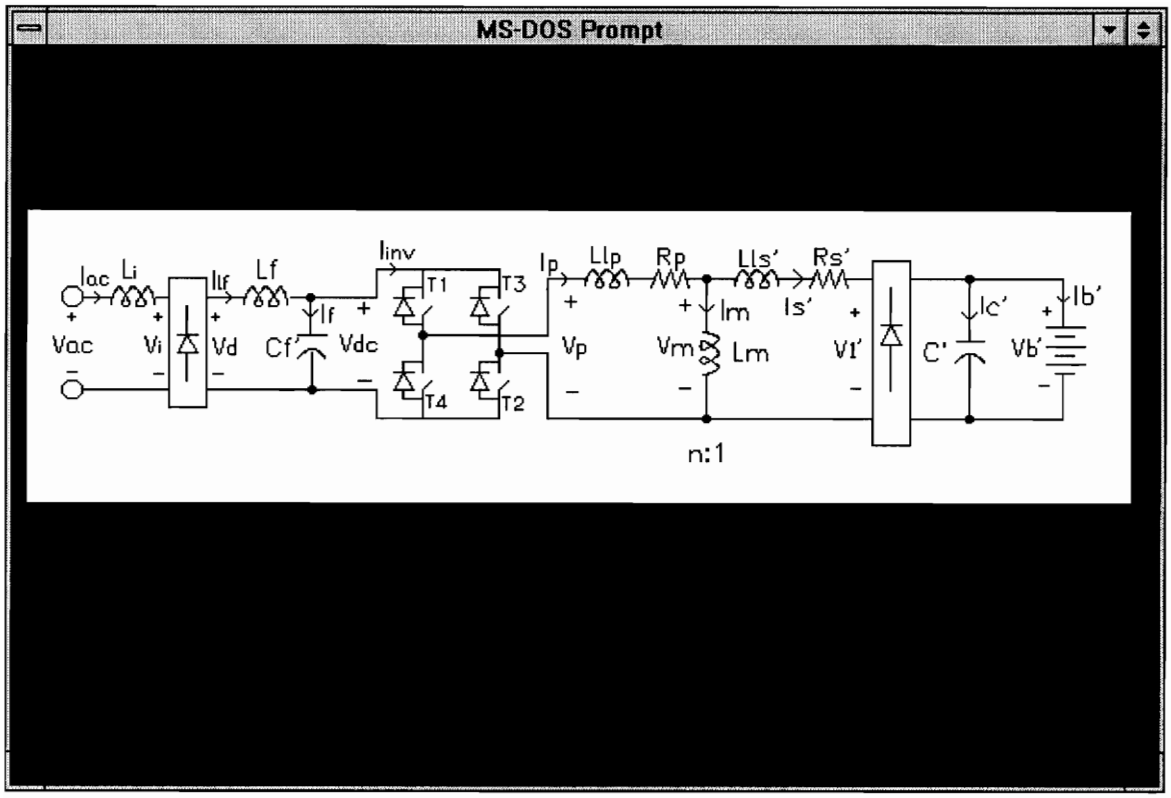

Figure 19. Sample figure as viewed

6.5 Hook for Viewing Output

Viewing the output while in the UI works similar to viewing the figures. When the user selects the plot command, a file dialog box appears as shown in figure 20. Once the user has selected an output data file, the program will call the external "Plot" program with the argument of the data file, which in turn calls the in house plotting program (see figure 21). The details of these two programs are not known nor necessary. The latter program has its own set of keys to change the way the graphs appear on the screen. Enough information is provided on the screen to allow the user to navigate through the program. After the user exits the plotting program, the UI redraws the screen as it was before the plot command.

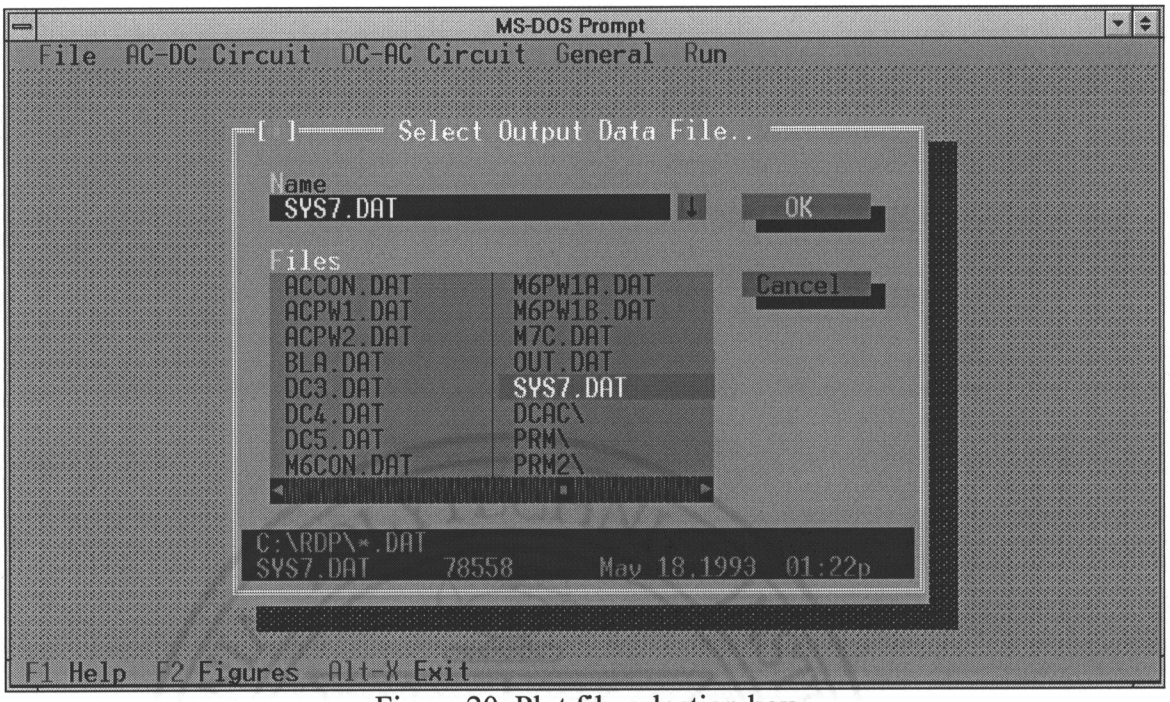

Figure 20. Plot file selection box

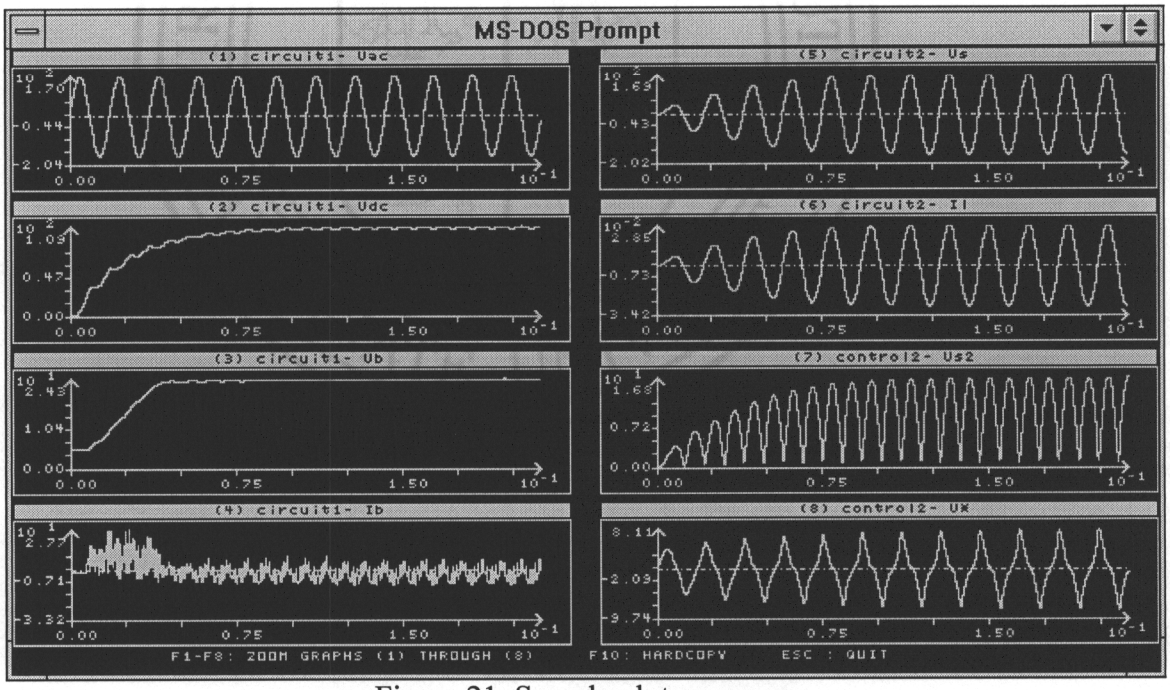

Figure 21. Sample plot on screen

6.6 Context Sensitive Help

This is probably the most useful feature to any new user. Help is activated by pressing the F1 hot key and exited by pressing ESC key. The help window opens containing information about the topic that is currently on the screen. Several highlighted item in the help window allow the user to navigate to related topics as well as to the main help screen. A sample help screen is shown in figure 22, where F1 was pressed while in the AC-DC control parameter entry dialog box.

The help is made context sensitive by changing the "topic" that the UI goes to as the screens change and as various commands are executed. The help file can also be modified without affecting the code as long as no new major topics are added. The file format is directly derived from the sample Turbo Vision Demo help file [4].

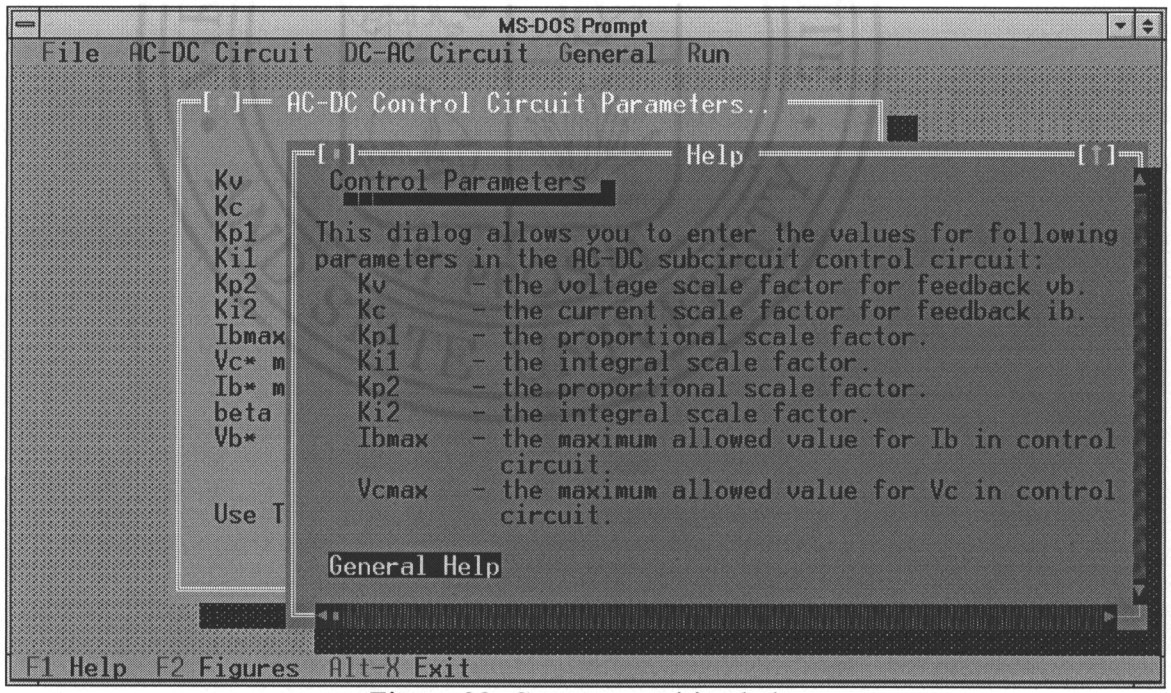

Figure 22. Context sensitive help.

CHAPTER 7

Results

This section is divided into two subsections. The first section provides samples of screen outputs from the user interface to give the feel of the program to the reader. The second subsection focuses on the actual simulation outputs.

7.1 User Interface

The following figures are provided below to show the screen from the UI. Figure 23 shows the following important features of the program:

- the status bar at the bottom of the screen with the hot keys.
- the menu bar at the top with the menu keys highlighted.
- a sample dialog box.
- a help window opened to the topic of the dialog box.
- the View Figures dialog box.

Figure 24 has two simulation windows open (one simulation was finished or aborted before the second was opened). Notice that the second simulation was aborted. Also, notice the highlighted topics ("General Help" is selected, "Plot" is not) in the help window.

A list of features and utilities included in the software package is given in Appendix D.

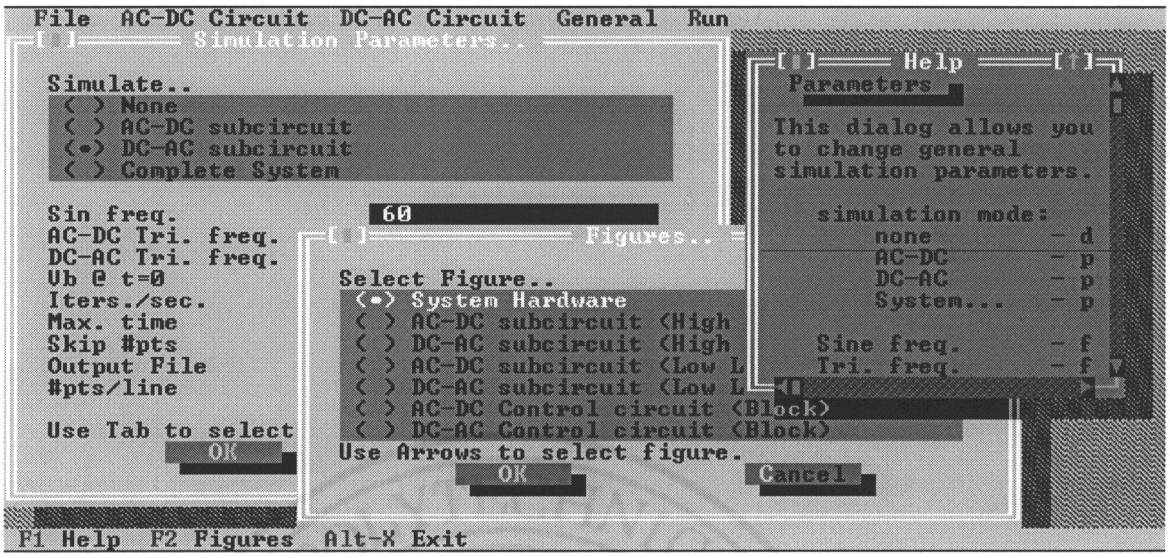

Figure 23. Sample User Interface Screen 1

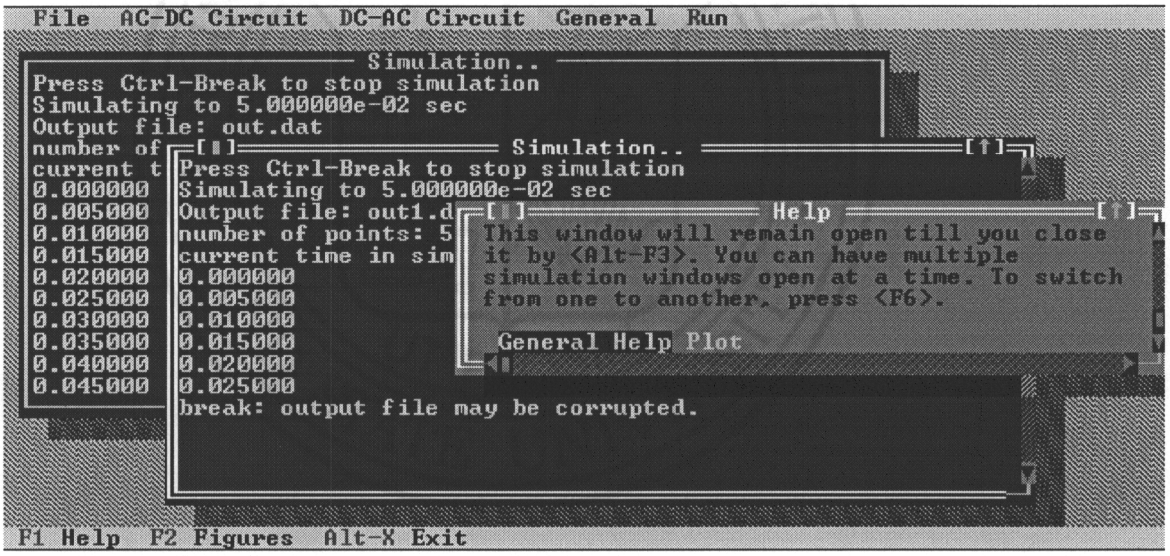

Figure 24. Sample User Interface Screen 2

7.2 Simulation Output

Each stage was simulated separately and then both were simulated together as a system. Sample results are provided for each of the simulations below.

For all the simulations, the simulation parameters files (as produced by the parameter file print program, PPRINT) are included in Appendices A, B and C. Most of the parameters are the same for all three simulations. The general parameters such as the initial battery voltage and the mode of simulation (stage 1, 2 or both) are different between the simulations. The parameters of AC-DC stage simulation are shown in Appendix A. Appendix B lists the parameters for the DC-AC stage simulation and the system simulation parameters are in Appendix C.

In the simulation of the AC-DC stage, an AC voltage of 120 V rms (170 V peak-peak) at 60Hz is applied and is converted to DC via a rectifier-filter block. This voltage is shown in figure 25 as v_{dc} and is fed to the inverter and the transformer blocks. The output of the transformer block, v_1' is fed via a rectifier to the battery and a load (a resistor in parallel with the capacitive model of the battery). The battery voltage and the current are also shown in figure 25. The voltage of the battery, v_b climbs to about 24 volts in the steady-state as specified in the control parameters. The maximum that it could climb is about 35 volts since the transformer is fed 110 V and the ratio n is 3. The load is applied only during the simulation of the AC-DC stage. If the entire system is simulated, there is no load at the battery (except, of course, the DC-AC stage). This data obtained from this simulation verifies the operation of the filters, the inverter, and the transformer as well as the two rectifiers as shown in figure 6.

For the simulation of the DC-AC power stage, the v_b is held constant, while the control circuit switches the inverter (see figure 26). All the signals start out at zero volts and climb up to their steady-state values. The output, v_s is a sinusoid output as shown in

figure 27. The value of v_s was set in the control parameters to 120V rms. The output in figure 27 verifies that the control algorithm is functional. All the intermediate signals are given to verify the system performance. The sinusoid shape of the output verifies the models used for the inverter and the transformer.

The system simulation results are shown in figures 28 and 29. The applied AC input is shown is first converted to DC via the LC filter. This DC voltage is then stepped down by the transformer to v_b, the battery voltage. The battery voltage is input to the DC-AC stage and v_s is the output signal at the load. The battery voltage ramps to about 24V as set in the control and then remains constant. Similarly, the load voltage (figure 29) rises and then comes to a steady-state amplitude of about 170V peak to peak as specified in the control parameters. The above test simulations show that the simulator software performs its function correctly. Figure 30 shows the simulation with the rectified load voltage, v_l . It behaves similar to the filter in the AC-DC stage as expected.

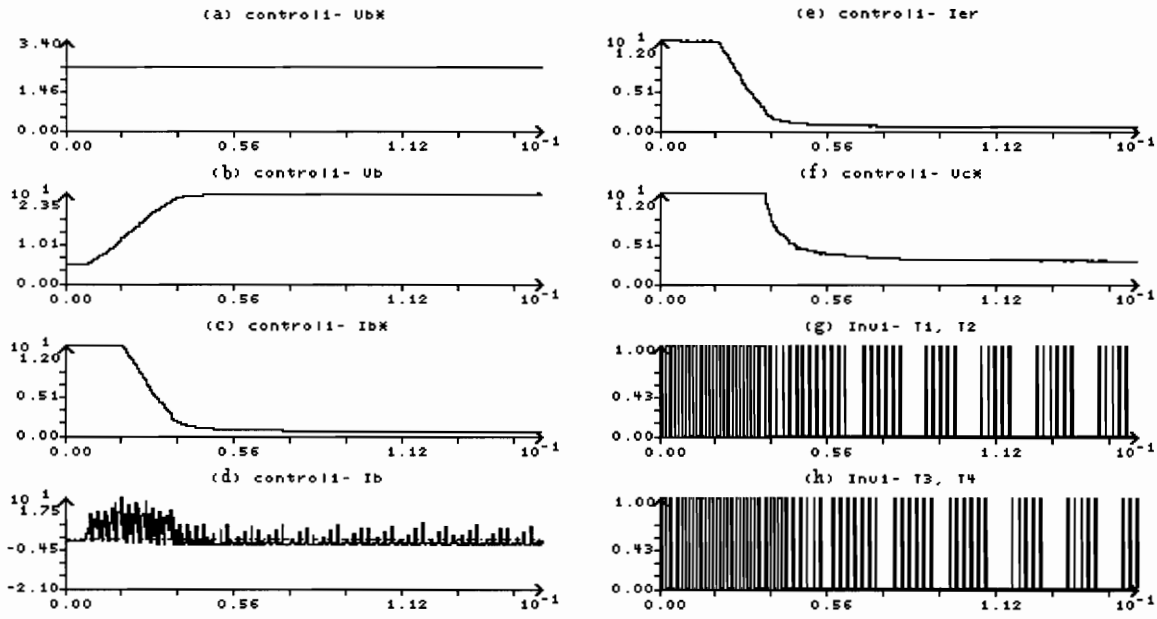

Figure 25. AC-DC simulation output (a-h)

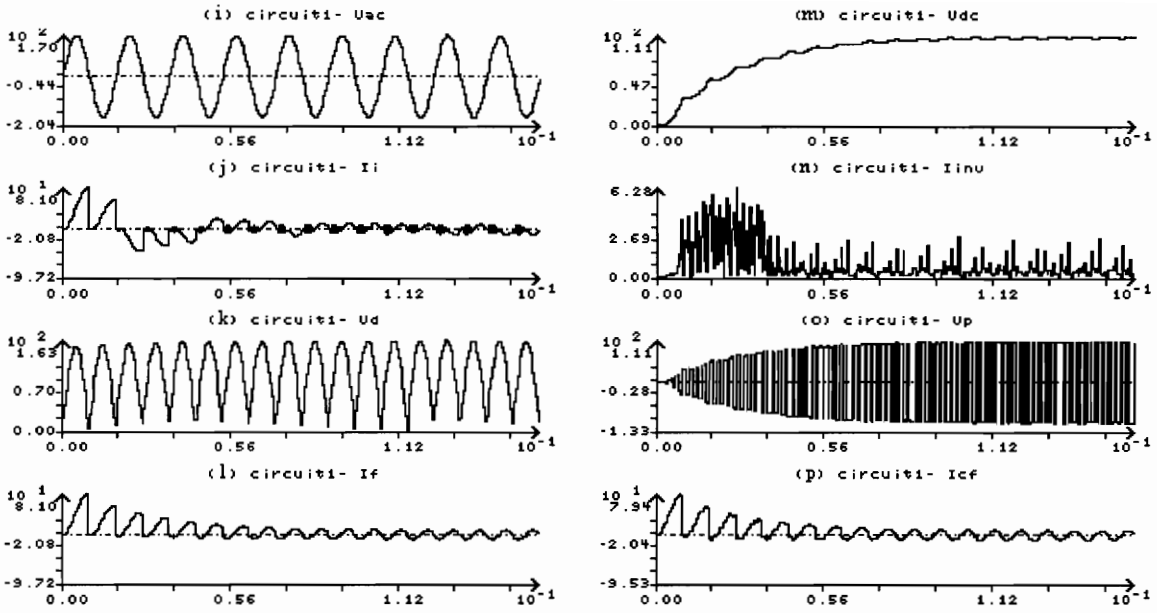

Figure 25. AC-DC simulation output (i-p)

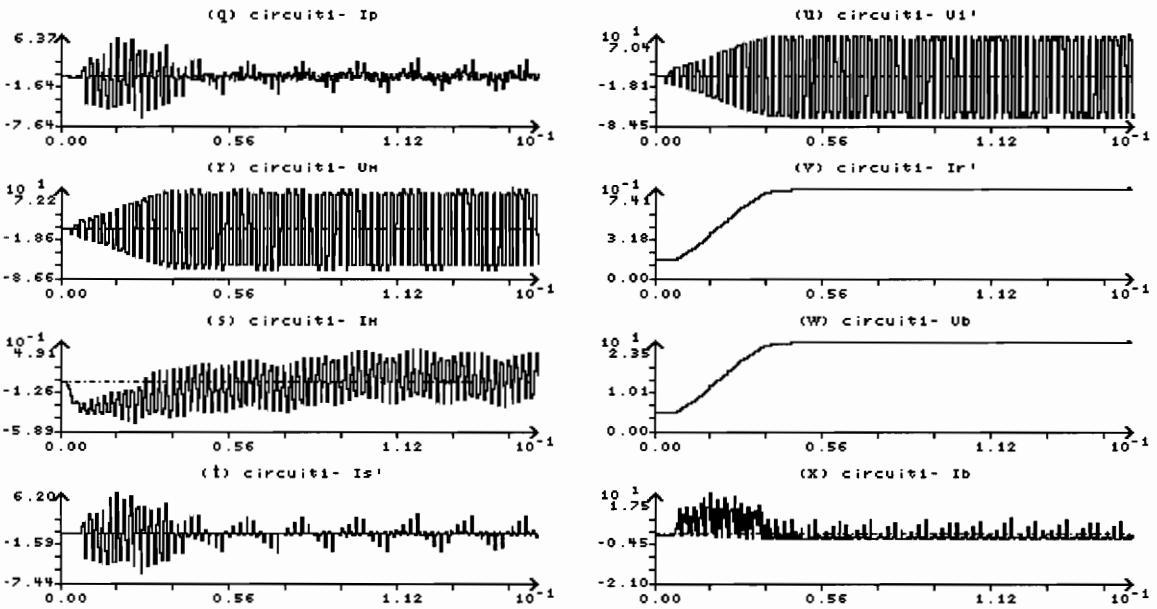

Figure 25. AC-DC simulation output (q-x)

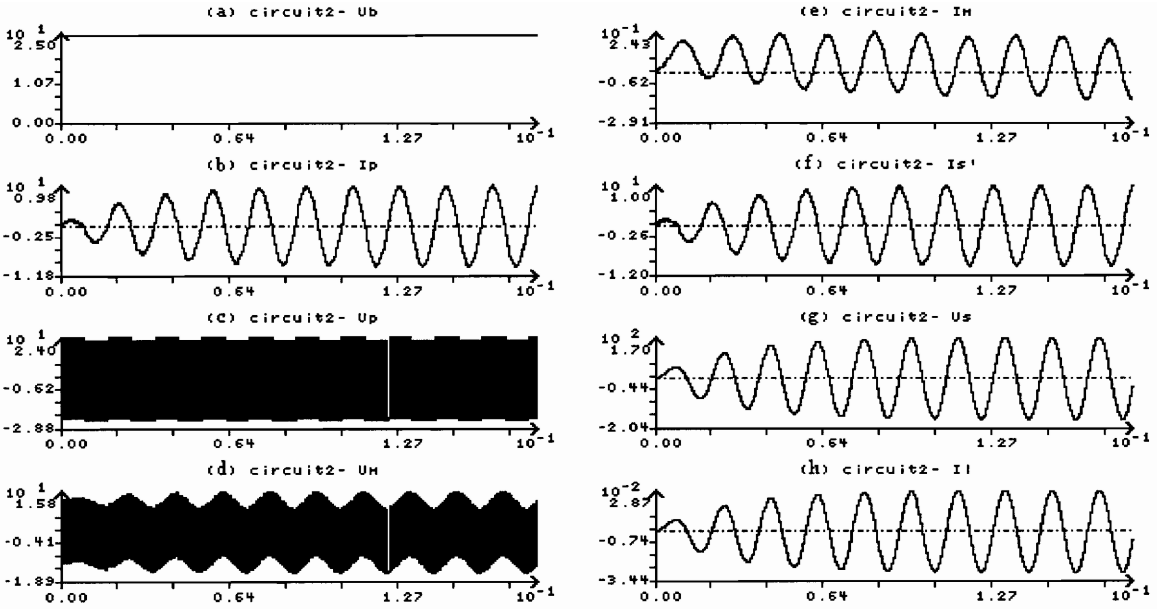

Figure 26. DC-AC simulation output (a-h)

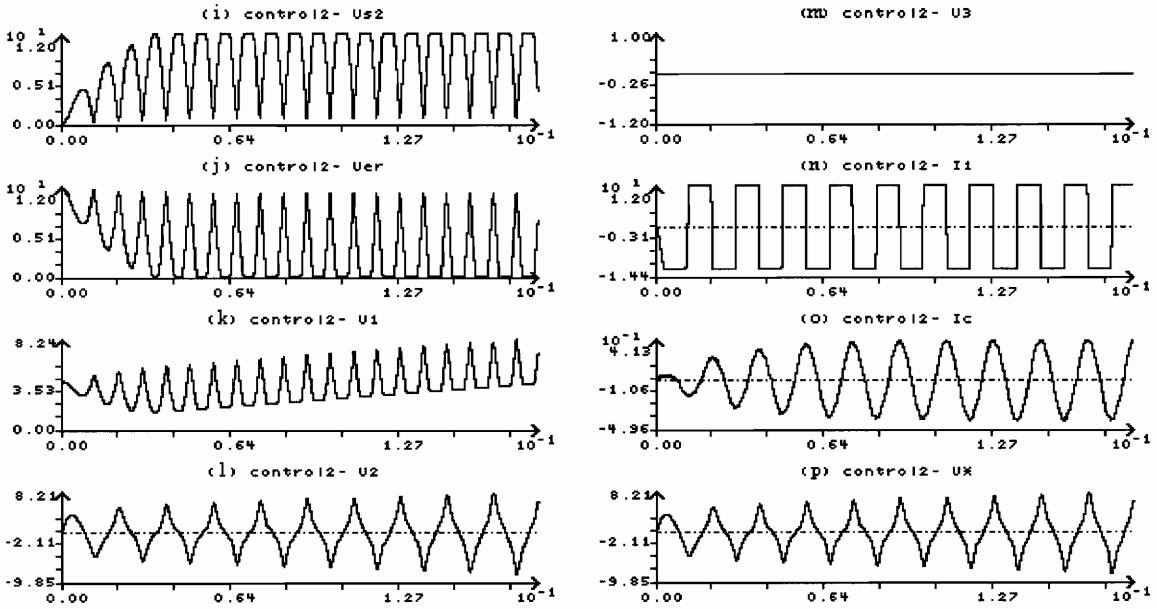

Figure 26. DC-AC simulation output (i-p)

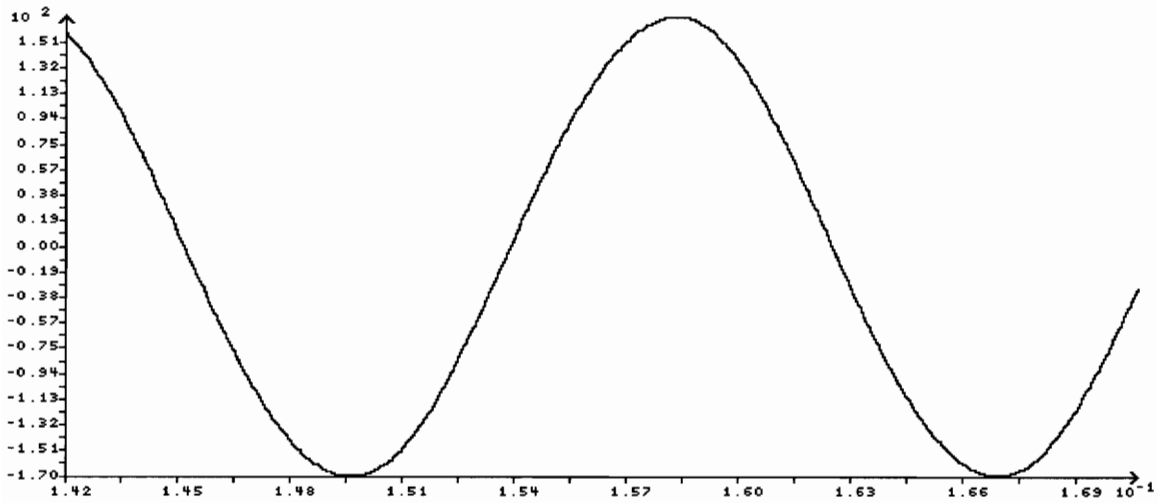

Figure 27. Steady-state output voltage, v_s.

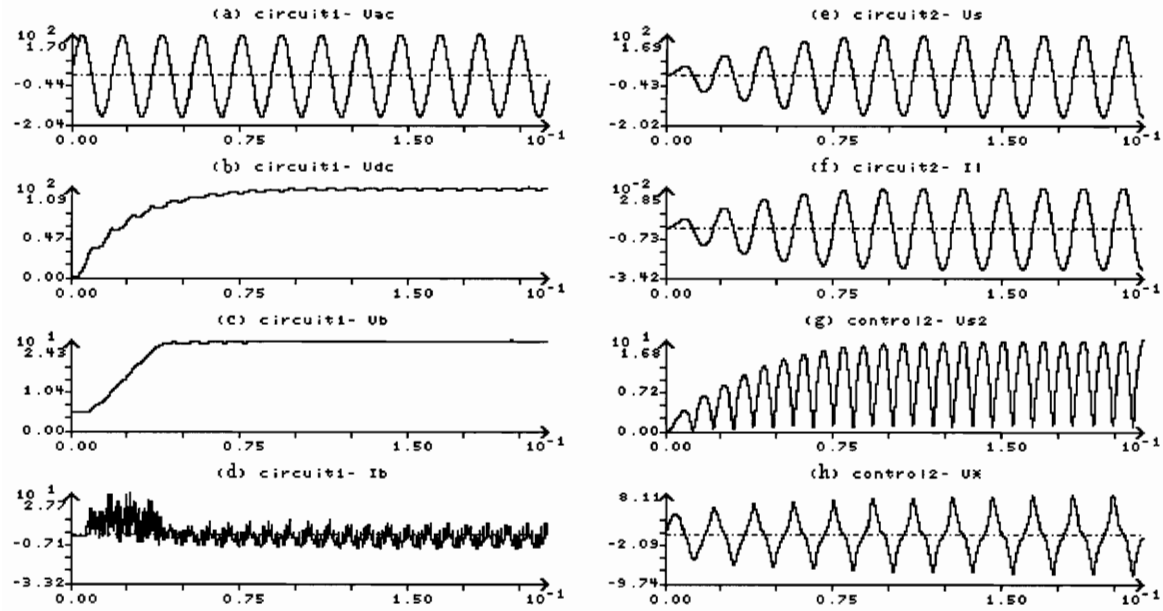

Figure 28. System simulation.

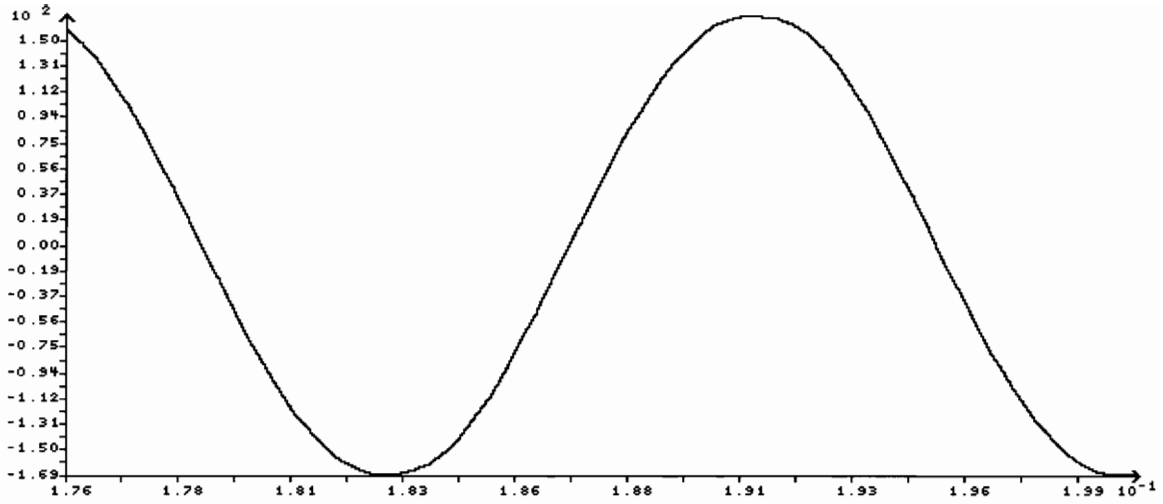

Figure 29. System simulation, load voltage, v_s.

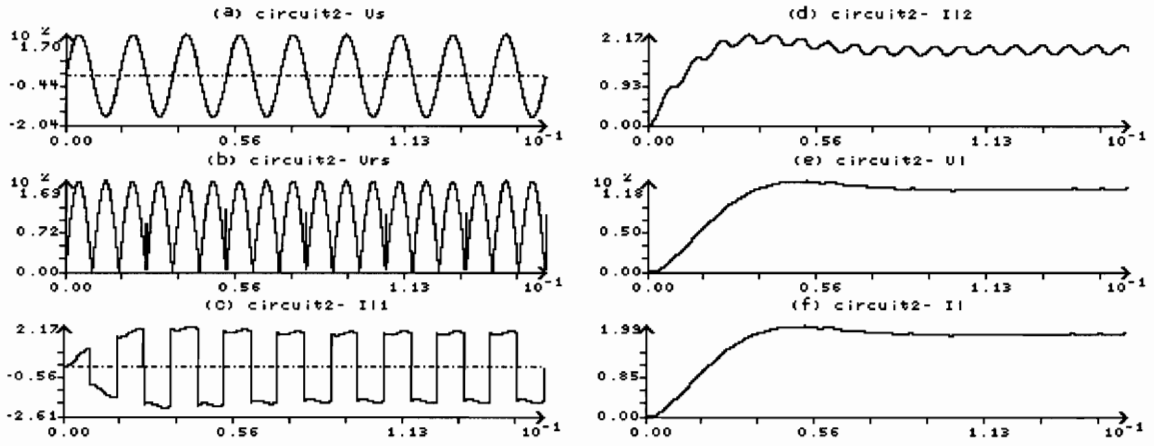

Figure 30. System simulation and dc load voltage, v_l.

CHAPTER 7
Conclusions

The contributions of the present study are:

1. A systematic step by step derivation of the model of a high frequency operated UPS is made.
2. A software simulation structure is proposed for the simulation of the UPS, which could be the kernel for developing a generalized software for all classes of UPS systems.
3. The actual implementation of the software on a PC is presented for the use of the engineers.
4. Simulation results are presented from the study for a specific set of controller configurations. The ease of change of controller configuration in the present software development facilitates system design studies for optimization for subsystems and their control.
5. It is anticipated that software tools such as these would accelerate the prototyping of many emerging UPS topologies and subsystem and system optimization.

BIBLIOGRAPHY

[1] Krishnan, R.; Srinivasan, S., "Topologies for Uninterruptible Power Supplies." A paper accepted for presentation in IEEE Industrial Electronics Symposium in Budapest, 6 pages, June 1-3, 1993.

[2] Skjellnes, A.; Snilsberg, G.; Munchow, E., "Matching a UPS to the computer market." INTELEC' '89. Part 2 (of 2), Oct., 1989.

[3] "Turbo Vision for C++, User's Guide." Borland International. Scotts Valley, California. 1992.

[4] Borland C/C++ 3.1 with Application Frameworks. Borland International. Scotts Valley, California. 1992.

[5] Shultz, R. D.; Smith, R. A., "Introduction to Electric Power Engineering." John Wiley & Sons, Inc. New York. 1988.

Appendix A - Sample Setting File

(Note: This is parameters file used for AC-DC stage simulation)

```
UPS simulator
mode: 1
out file: out.dat
sine freq: 60
AC-DC PWM freq: 10000
DC-AC PWM freq: 10000
iterations per sec: 1000000
time to simulate: 1.500000e-001
print every 441 th point.
number of variables to monitor: 9
number of values per line: 5
signals monitored plus time:
   time
   control1- Vb*
   control1- Vb
   control1- Ib*
   control1- Ib
   control1- Ier
   control1- Vc*
   Inv1- T1, T2
   Inv1- T3, T4
power stage AC-DC settings:
li:        1.000000e-003
lf:        9.000000e-003
cf:        1.000000e-002
Vdd:       7.000000e-001
n:         3.000000e+000
rp:        5.000000e-001
lp:        2.000000e-004
lm:        9.300000e-003
ls':       1.000000e-005
rs':       1.000000e-002
cs':       9.500000e+001
cb':       1.200000e-003
rb':       1.600000e-003
AC_amp:    1.700000e+002
AC_phase:  0.000000e+000
```

```
control algorithm AC-DC settings:
kv:        1.000000e-001
kp1:       1.000000e+001
ki1:       1.500000e+000
kp2:       4.500000e+000
ki2:       8.000000e-001
kc:        5.030000e-003
beta:      1.230000e+001
ibmax:     1.200000e+002
ib*max:    1.200000e+001
vc*max:    1.200000e+001
vb*:       2.400000e+000
power stage DC-AC settings:
vdd:       7.000000e-001
n:         1.000000e-001
rp:        5.000000e-002
lp:        9.000000e-004
lm:        5.000000e-002
ls':       1.000000e-003
rs':       5.000000e-002
cs':       4.420000e-003
ll':       0.000000e+000
cl':       0.000000e+000
rl':       1.000000e+001
vb @ t=0:        5.000000e+000
control algorithm DC-AC settings:
kv:        1.100000e-001
kp1:       2.700000e-001
ki1:       8.800000e+000
kp2:       1.000000e+000
kc:        9.700000e-002
kp3:       1.000000e-001
rc:        2.000000e-004
v1max:     1.200000e+001
vsmax:     1.900000e+001
i1max:     1.200000e+001
v*max:     1.200000e+001
alpha:     1.220000e+001
vs*:       1.200000e+001
description: default AC-DC
UPS simulator
```

Appendix B - Sample Setting File

(Note: This is the parameters file used for DC-AC stage simulation.)

```
UPS simulator
mode: 2
out file: out.dat
sine freq: 60
AC-DC PWM freq: 20000
DC-AC PWM freq: 10000
iterations per sec: 1000000
time to simulate: 1.700000e-001
print every 113 th point.
number of variables to monitor: 9
number of values per line: 5
signals monitored plus time:
   time
   circuit2- Vb
   circuit2- Ip
   circuit2- Vp
   circuit2- Vm
   circuit2- Im
   circuit2- Is'
   circuit2- Vs
   circuit2- Il
power stage AC-DC settings:
li:        1.000000e-003
lf:        9.000000e-003
cf:        1.000000e-002
Vdd:       7.000000e-001
n:         2.000000e+000
rp:        5.000000e-001
lp:        2.000000e-004
lm:        9.300000e-003
ls':       1.000000e-005
rs':       1.000000e-002
cs':       9.500000e+001
cb':       5.000000e-003
rb':       1.600000e-003
AC_amp:    1.700000e+002
AC_phase:  0.000000e+000
```

```
control algorithm AC-DC settings:
kv:         1.000000e-001
kp1:        1.000000e+001
ki1:        1.000000e+000
kp2:        4.000000e+000
ki2:        6.000000e-001
kc:         5.030000e-003
beta:       1.240000e+001
ibmax:      1.200000e+002
ib*max:     1.200000e+001
vc*max:     1.200000e+001
vb*:        2.400000e+000
power stage DC-AC settings:
vdd:        7.000000e-001
n:          4.110000e-002
rp:         5.000000e-002
lp:         9.000000e-004
lm:         5.000000e-002
ls':        1.000000e-003
rs':        5.000000e-002
cs':        3.700000e-003
ll':        0.000000e+000
cl':        0.000000e+000
rl':        1.000000e+001
vb @ t=0:         2.400000e+001
control algorithm DC-AC settings:
kv:         1.000000e-001
kp1:        3.700000e-001
ki1:        6.700000e+000
kp2:        1.000000e+000
kc:         9.700000e-002
kp3:        1.000000e-002
rc:         2.000000e-004
v1max:      1.200000e+001
vsmax:      1.200000e+001
i1max:      1.200000e+001
v*max:      1.200000e+001
alpha:      1.220000e+001
vs*:        1.200000e+001
description: default AC-DC
UPS simulator
```

Appendix C - Sample Setting File

(Note: This is the parameters file used for system simulation.)

```
UPS simulator
mode: 3
out file: out.dat
sine freq: 60
AC-DC PWM freq: 24000
DC-AC PWM freq: 10000
iterations per sec: 1000000
time to simulate: 2.000000e-001
print every 341 th point.
number of variables to monitor: 9
number of values per line: 5
signals monitored plus time:
   time
   circuit1- Vac
   circuit1- Vdc
   circuit1- Vb
   circuit1- Ib
   circuit2- Vs
   circuit2- Il
   control2- Vs2
   control2- V*
power stage AC-DC settings:
li:        1.000000e-003
lf:        9.000000e-003
cf:        1.000000e-002
Vdd:       7.000000e-001
n:         2.900000e+000
rp:        5.000000e-001
lp:        2.000000e-004
lm:        9.300000e-003
ls':       1.000000e-005
rs':       1.000000e-002
cs':       9.500000e+001
cb':       1.900000e-003
rb':       1.600000e-003
AC_amp:    1.700000e+002
AC_phase:  0.000000e+000
```

```
control algorithm AC-DC settings:
kv:         9.340000e-002
kp1:        1.200000e+001
ki1:        2.500000e+000
kp2:        4.500000e+000
ki2:        1.000000e+000
kc:         5.030000e-002
beta:       1.230000e+001
ibmax:      1.200000e+002
ib*max:     1.200000e+001
vc*max:     1.200000e+001
vb*:        2.400000e+000
power stage DC-AC settings:
vdd:        7.000000e-001
n:          4.110000e-002
rp:         5.000000e-002
lp:         9.000000e-004
lm:         5.000000e-002
ls':        1.000000e-003
rs':        5.000000e-002
cs':        3.420000e-003
ll':        0.000000e+000
cl':        0.000000e+000
rl':        1.000000e+001
vb @ t=0:         5.000000e+000
control algorithm DC-AC settings:
kv:         1.000000e-001
kp1:        3.700000e-001
ki1:        7.800000e+000
kp2:        1.000000e+000
kc:         9.700000e-002
kp3:        1.000000e-001
rc:         2.000000e-004
v1max:      1.200000e+001
vsmax:      1.900000e+001
i1max:      1.200000e+001
v*max:      1.200000e+001
alpha:      1.220000e+001
vs*:        1.200000e+001
description: default AC-DC
UPS simulator
```

Appendix D - Features and Utilities

Features

The many features of the software will be highlighted next. They are presented in a bullet form with description where necessary for quick reference.

- Two versions - Command line and user interface applications(UIA).
- Batch mode - Command line version can be executed from a batch multiple times to more than one simulation.
- Standard interface - The UIA features a standard menu system consisting of a menu bar on the top with well accepted File menu and dialog boxes. Also, at the bottom of the screen, a Hot key bar shows the user keys for quick action.
- Standard dialog - The dialog boxes for entry and selection are created using a standard library of selection and entry elements and window objects such as buttons and borders.
- Context sensitive help - The user can quickly obtain help on the topic that he/she is currently working with without searching for it.
- Figures online - As an extension of the help system, any of the circuit diagrams can be viewed from the UIA by pressing F2.
- Mouse friendly - If a mouse is present in the system, it will be automatically activated. Any action except parameter entry can be performed using a mouse.
- Plot option - This option under the Run menu allows the user to select any of the existing data files for viewing on the screen. Another feature is that if the

user has a better viewing software, the UIA can be made to call that software simply by creating a hook routine called Plot.exe which executes in turn the viewing program.

- Flexible output format - The output data file format is controlled by parameters in the General Parameters entry box. The format can be easily changed to generate files which can be imported into spreadsheets or other commercial software for further analysis.
- Error checking - Most parameters are checked for errors such as 0V for input signals, etc. Also, the parameter file (*.prm) is also checked for integrity during loading, thus preventing loading of *.prm files from other software.
- Simulation modes - Three modes of simulation allow the user to simulate just one stage of the UPS system or the entire system.
- Total control - The UIA allows the user to create parameter files for use with a built-in simulator or the command line simulator. All the parameters of the system can be set using the UIA, thus providing total control of the UPS.
- Performance -

  AC-DC simulation - 200 secs/1 million iterations

  DC-AC simulation - 217 secs/1 million iterations

  System simulation - 350 secs/1 million iterations

The performance was measured on a 486 DX/2 - 66 Mhz with a SCSI hard disk controller. The difference of about 67 seconds between the sum of the single stage simulation and the system simulation is mainly due to the fact that only nine values are written to a file in the system simulation, while a total of 18 are written in the stage simulations. Thus the hard disk subsystem of the computer plays a very important role in the performance of the

simulator. It is expected that using some accepted performance measure such as the Norton Speed Index or the Intel iCOMP rating, timing on other systems could also be computed.

Utilities

The following utilities are provided in the software package:

- UIA.exe - This is the main executable software with the user interface and the simulator. The UIA.exe accepts a xxx.prm argument as initial parameter file. It defaults to the default.prm file if no argument is present. It will read, modify and write xxx.prm files and output a data file as well as status of the simulation to the screen if a simulation is performed.
- UPS.exe - This is the command line version of the simulator. The UPS.exe requires a xxx.prm argument and outputs a data file and status information on the screen.
- PPrint.exe - This program is used to print the xxx.prm in a commented form. It requires a xxx.prm argument and outputs the commented parameters to the screen. Standard redirection (>) can be used to capture the output to a file.
- Plotting.exe - This viewing software and its associated files are used in-house for viewing purpose.
- Plot.exe - The UIA.exe calls this routine with the argument of xxx.dat file and this software in turn calls the Plotting.exe to actually view the data. The user can view data files from command line using this software.
- TPicem.exe - This is a public domain picture file viewer. It accepts standard xxx.pic file as an argument and displays them on a EGA style screen. This called by the UIA to view the figures.

Vita

The author was born in Baroda, India on May 10, 1971 and moved with his family to Lynchburg, Virginia in 1982. He went to high school at E. C. Glass High School and also attended the Central Virginia Governors/Magnet School for two years, where he first developed interest in computers. He started his Bachelors at Virginia Tech in August, 1988 and received a BS in Computer Engineering with Electrical Engineering as second major in 1992. He immediately started his Masters and received a MS EE in May of 1993. The author has interests in wide variety of new technologies, enjoys going to movies and playing chess.