

# **S** SCRIPT FILE EXAMPLE

The function of a script file is to control the operation of the simulator and to collect and analyze data without direct user interaction. A record/playback feature is provided to assist in creating the script file. A simple example is described in which simulator commands are written to a script file and then played back. “Batch mode operation” is introduced along with “Preprocessor Variables”.

The term “Batch Mode Operation” describes the ability of script files to run many different cases with minimal or no user interaction. For example, a script file can be created to run many cases overnight. For each case, output data can be stored or plotted without any user interaction. This type of operation is particularly useful for protective relay testing where many different types of faults may have to be applied at different locations along a line.

The term “Preprocessor Variable” is used to describe special variables which can be altered from the RunTime Operators Console. Whenever one of these variables is changed, the preprocessor compiler automatically performs a recompile without the necessity of returning to Draft.

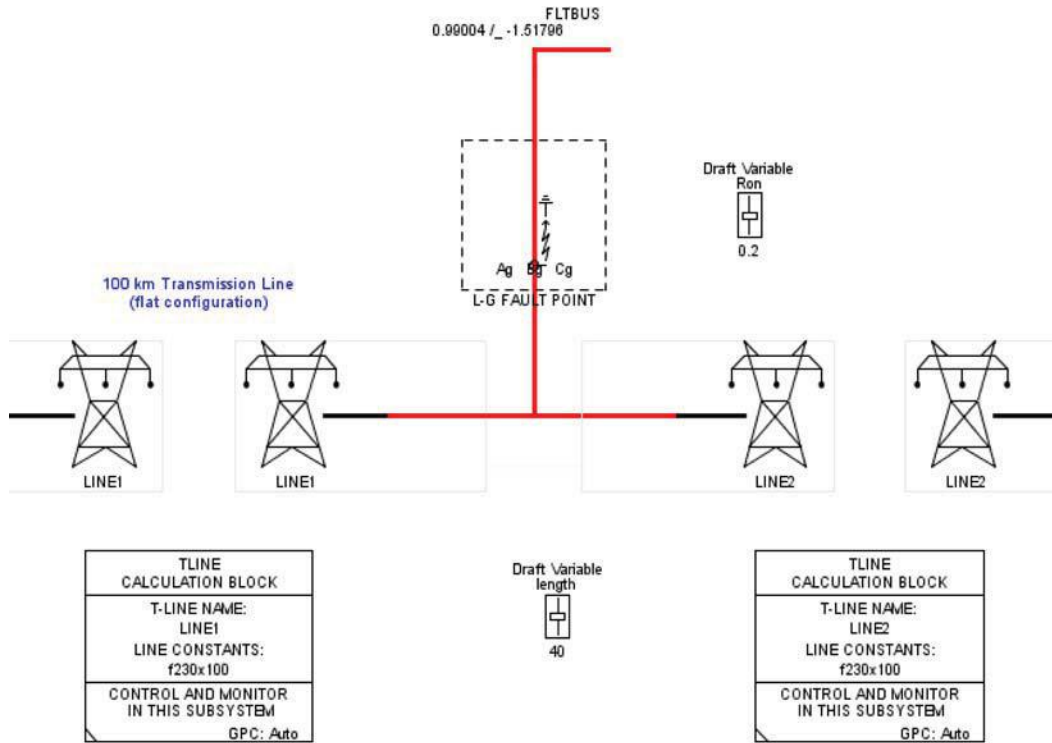
## **7.1 PREPARING THE CIRCUIT IN DRAFT**

To illustrate batchmode operation, the circuit from Chapter 2 consisting of a source, a travelling wave transmission line and a load is used. However, the line model is changed, the total line length is altered to 100 km, and the position of the fault bus is made a variable. The fault resistance is also made a variable.

Changing fault resistance and fault bus location would normally require stopping the case, recompiling in Draft and then returning to RunTime to restart the case. With preprocessor variables, a recompile can be invoked and performed directly in Run-Time whenever a preprocessor variable is changed and the case is stopped.

The new line model is named *lf\_rtds\_sharc\_sld\_TLINE*. The line currently in the Draft circuit is split using two lines and a fault bus is placed between them. With this configuration, the fault bus can be moved anywhere along the line. For the source component and transmission line parameters, refer to Chapter 2.

The modified portion of the Draft circuit with the fault bus placed between two lines is shown below.

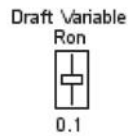


To define a preprocessor variable in Draft, the parameter must be given a variable name rather than a numerical value. A \$ character must precede the variable name entered in the component. In the fault component menus, enter variable name \$Ron as the A,B and C Phase to Ground Fault Resistance parameter. For example;

rtds_sharc_slid_FAULT					
C Phase - Ground Fault Branch Data					
B Phase - Ground Fault Branch Data					
A Phase - Ground Fault Branch Data					
CONFIGURATION			L-G PARAMETERS		
Name	Description	Value	Unit	Min	Max
Agnam	A Phase - Ground Fault Name	AG			
AgRon	A Phase - Ground Fault Resistance	\$Ron	ohm	1E-9	
Agholdi	Extinguish Arc for abs(!) at or below:	0.0	kA	0.0	10.0
Asig	Signal Name to control fault	LGFLT			
Abit	Active bit number in Asig to trigger fault	1		1	21

Update Cancel Cancel All

An initial value is assigned to the variable Ron through the use of a Draft variable slider. The Draft variable slider component also creates a crossreference between Draft and RunTime. The Draft variable slider component icon (*rtds\_draft\_var*) appears as shown below.



Draft Variable Slider (rtds\_draft\_var)

The Slider component requires the following input parameters;

rtds_draft_var					
Parameters					
Name	Description	Value	Unit	Min	Max
Name	Variable Name	Ron			
Type	Type	REAL		0	2
Value	Variable Value	0.2			
Max	Maximum Value	100.0			
Min	Minimum Value	0.0			
Units	Units (eg: kA, KV, P.U.)	ohms			

Update Cancel Cancel All

Note: The Name of the slider must be the same as the variable name entered for the fault resistance. The \$ character is **not** required in the slider description. The fault resistance will initially be set to 0.2 ohms.

The 100km transmission line is modelled as two separate lines, each with its own icon. One icon represents the sending end of the 100km line while the other represents the receiving end. The length of each end can be modified using the preprocessor. This operation moves the position of the fault bus along the length of the line. In the tline calculation block (*lf\_rtds\_sharc\_sld\_TL16CAL*), the preprocessor data can be entered as shown below;

If_rtds_sharc_sld_TL16CAL					
OPTIONS WHEN USING BERGERON DATA					
CONFIGURATION			PROCESSOR ASSIGNMENT		
Name	Description	Value	Unit	Min	Max
rdData	Read line constants from:	tlo/clo		0	1
pp_var	Variable Name or Number for % length	\$length	%	0.0	100.0
hmnpp	To calculate line length use:	(pp_var)%			
frpci	Force use of PI Section model ?	No			
alwpi	If Travel T < T Step, allow PI model ?	Yes			
raistt	If Travel T < T Step and No PI, then:	ERROR			

Update Cancel Cancel All


Enter variable name *\$length* as the preprocessor variable in the tline calculation block. If the icon represents the sending end of the line, set the parameter 'hmnpp' to (pp\_var%). Similarly in the other tline calculation block for the receiving end of

the line, set the preprocessor variable name as  $\$length$  and set the 'hmnpp' parameter to (100ipp\_var%). Now for example, if the preprocessor variable (pp\_var) length is set to 40%, the length of the sending end line will be set to 40% of the total line length and the length of the receiving end line will be set to 60% of the total line length.

An initial value must be assigned to the variable length using the Draft variable slider. The Slider component requires the following input parameters.

rtds_draft_var					
Parameters					
Name	Description	Value	Unit	Min	Max
Variable Name		length			
Type	Type	REAL		0	2
Value	Variable Value	40			
Max	Maximum Value	100.0			
Min	Minimum Value	0.0			
Units	Units (eg: kA, kV, P.U.)	%			

Update Cancel Cancel All

 Note: The Name of the slider must be the same as the variable name entered for the line length. The \$ character is *not* required in the slider description. The initial line length of the sending end line will be 40km and the receiving end line length will be 60km.

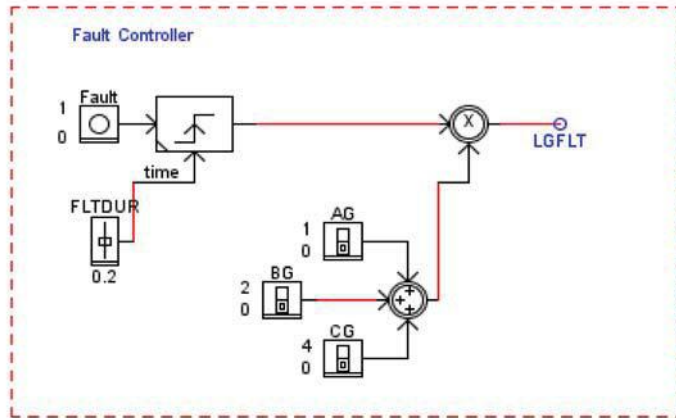
For protective relay testing, fault position may have to be located between 1% and 99% of the line length. The travelling wave line model cannot be used to represent lines whose travel time is less than one simulation time step. For a 50<sup>3</sup>sec time step this is 15km. If the fault bus position is set at 1% of a 100km line, it cannot be modelled with a travelling wave line (ie. the sending end line length is 1km). A useful option is available in the tline calculation block to allow lines shorter than one time step to be modelled as pi sections.

If_rtds_sharc_slid_TL16CAL					
OPTIONS WHEN USING BERGERON DATA					
CONFIGURATION			PROCESSOR ASSIGNMENT		
Name	Description	Value	Unit	Min	Max
rdData	Read line constants from:	tlo/clo		0	1
pp_var	Variable Name or Number for % length	\$length	%	0.0	100.0
hmnpp	To calculate line length use:	(pp var)%			
frdpi	Force use of PI Section model ?	No			
alwpi	If Travel T < T Step, allow PI model ?	Yes			
raistt	If Travel T < T Step and No PI, then:	ERROR			

Update Cancel Cancel All

With the “If Travel T < T Step, allow PI model” option set to YES, the entire range of fault positions can be tested using one line model.

A simple fault controller can be added to the Draft circuit as shown.



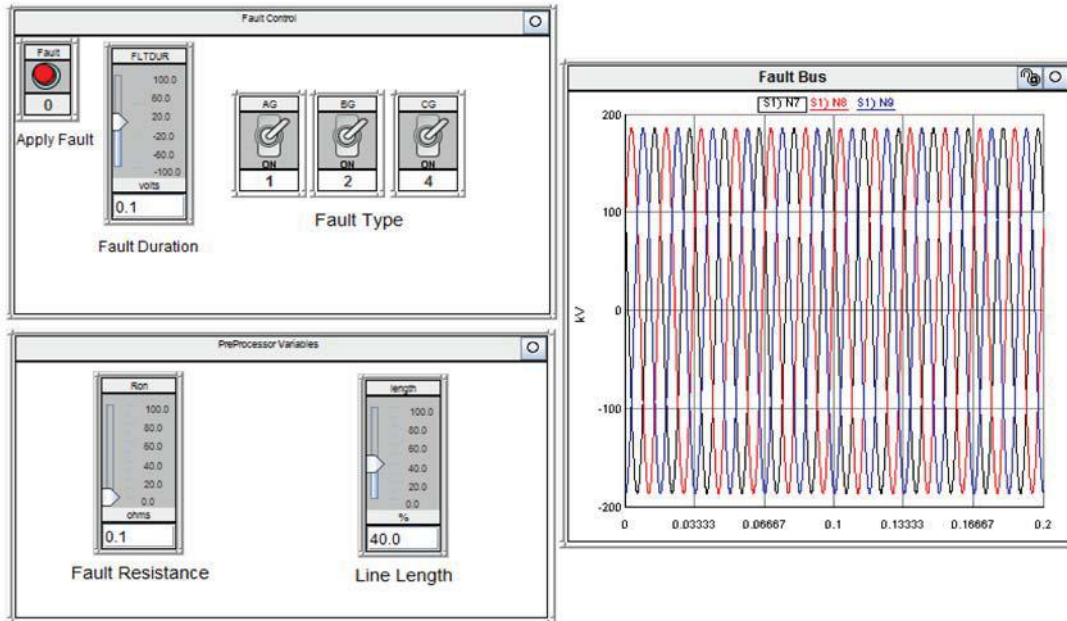
The signal LGFLT should be entered in the fault component as the signal name to control the fault. Set phase A to ground fault to operate on bit 1 of the LGFLT signal, set phase B to ground fault to operate on bit 2 and phase C to ground fault to operate on bit 3. The fault controller will allow the type of fault to be selected in RunTime using the switches AG,BG and CG. Note that the ON value of each switch is set as shown above. The fault duration is set using the FLTDUR slider and the fault is initiated by pressing the Fault pushbutton during the simulation.

Once the data has been entered the case can be compiled in the usual manner.

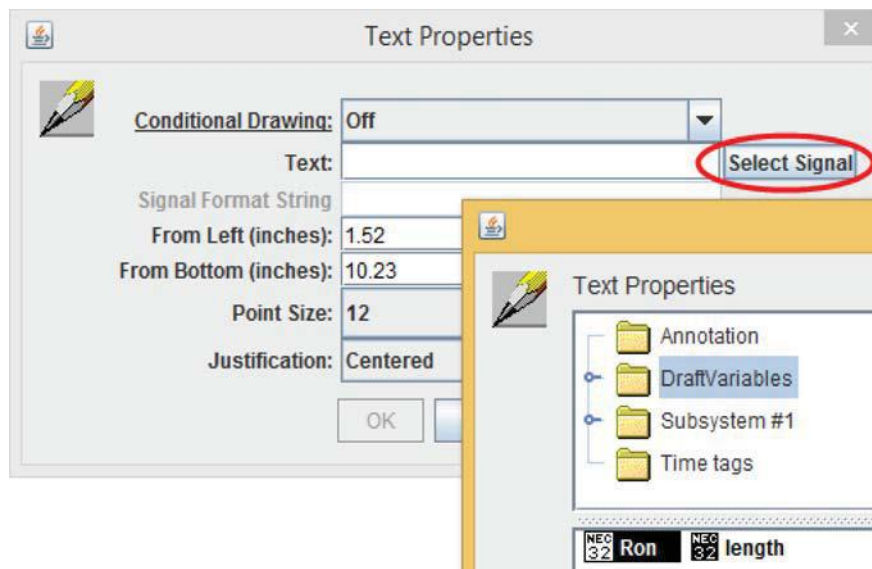
## 7.2 PREPARING THE SIMULATION USING RUNTIME

To begin the simulation, first load the correct RunTime batch file corresponding to the compiled Draft case name. Draft variable sliders can then be created to modify the fault resistance and the line length. As long as the case is stopped, modifying the draft variable slider in RunTime will change its' value. If the case is running, the draft variable sliders will appear greyed out (disabled) in RunTime and changes will not be accepted. On clicking the Run button, RunTime will detect the change and recompile the circuit prior to actually starting the case.

The RunTime operators console can be set up similar to that shown below.

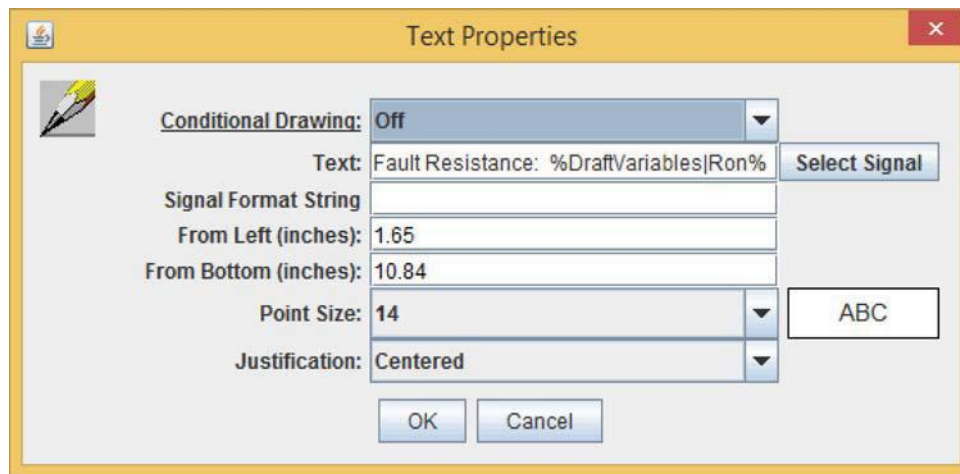


If many plots are being saved and printed during the batch mode operation, text displaying simulation parameters is useful. For example, the fault resistance and line length can be displayed on a plot. To add text, right click on the background of a Run-Time plot to get the Add Text menu. Select *Add Text*, select *Select Signal* from the Add Text menu and finally select the draft variable slider name Ron. For example;

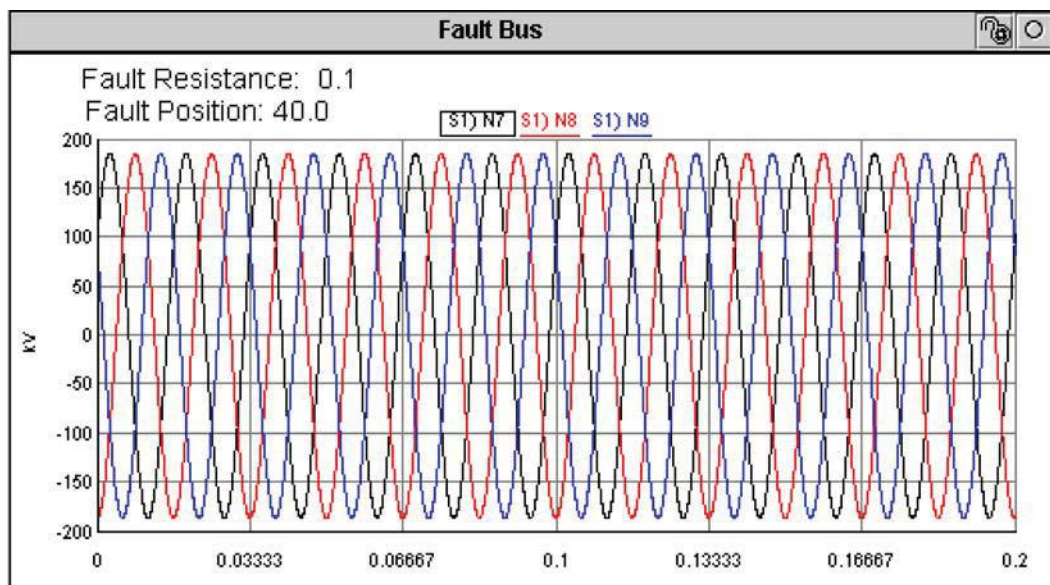


This will display, as text, the current value of the slider Ron. Descriptive text can be added before or after the string as shown below.





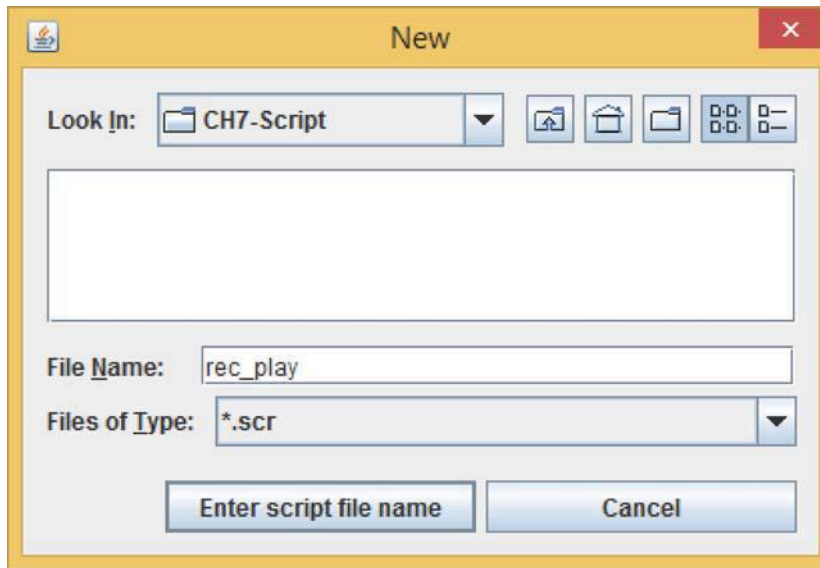
This results in a plot that appears similar to that shown below. Every time the fault resistance or line length draft variable sliders are changed in RunTime, the corresponding values will change accordingly on the plot(s).



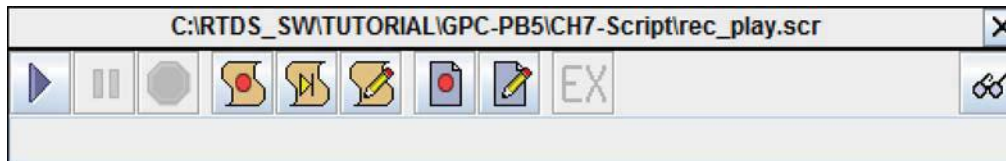
### 7.3 RUNTIME SCRIPT RECORD/PLAYBACK FEATURE

The script record feature writes simulator commands to a script text file. During a record session, the operations performed in RunTime by the user will be written to the script file. For example, if a simulation is started during a record session, the command *Start* will be written to the script file.

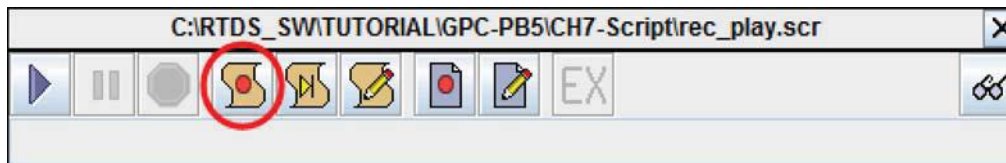
To start a record session, select Scripti>New from the RunTime Toolbar. A file re-questor will appear requesting a script file name.



Enter a script file name and a script tool bar will appear in RunTime as shown.



Select the record script button from the Script toolbar as shown to begin the record session.



A script file labelled rec\_play.scr is now being recorded.

During the record session,

S start the simulation


S set the A phase to ground fault switch (AG) to 1

S set the B and C phase to ground fault switches (BG & CG) to 0


S set the fault duration slider to 0.1 seconds (FLTDUR)

S apply the fault by pressing the Fault pushbutton

S stop the simulation

Select the stop button  from the script tool bar to end the record session.

Once the recording session is complete, a script file will exist in which the recorded actions are stored. In this example, the script file name is rec\_play.scr. To view the

contents of the rec\_play.scr file, select the edit button  from the script toolbar.



The script file should look similar to that shown below.

```
SUSPEND 2.554;
Start;
SUSPEND 6.97;
SetSwitch "Subsystem #1 : CTLs : Inputs : AG" = 1;
SUSPEND 1.011;
SetSwitch "Subsystem #1 : CTLs : Inputs : BG" = 0;
SUSPEND 1.372;
SetSwitch "Subsystem #1 : CTLs : Inputs : CG" = 0;
SUSPEND 10.094;
SetSlider "Subsystem #1 : CTLs : Inputs : FLTDUR" = 0.1;
SUSPEND 1.853;
PushButton "Subsystem #1 : CTLs : Inputs : Fault";
SUSPEND 0.2;
ReleaseButton "Subsystem #1 : CTLs : Inputs : Fault";
SUSPEND 3.715;
Stop;
```

The SUSPEND times recorded in the script are the amount of time in seconds that it took the user to perform the next action and will vary from script to script. SUSPEND times could be removed to increase the speed of the script.


The SetSwitch command was recorded in the script file when the AG,BG and CG switches were set to 1 and 0. This command requires the switch name and position as arguments.

The SetSlider command was recorded in the script file when the FLTDUR slider was set to 0.1. This command requires the slider name and position as arguments.

The PushButton and ReleaseButton commands were recorded in the script file when the FLT pushbutton was pressed and released. These commands require the pushbutton name as an argument.

The Start and Stop commands do not require any arguments and a script file does not need to include these commands. A script file can be started at any time before, during or after a simulation run.

The script file can now be run in its existing form or it can be edited. The section which follows will introduce the script edit facility.

To run the recorded script file, select the play button  from the Script Toolbar.

During the playback session you should notice that all the actions recorded in the script file are now being performed in RunTime without any interaction.

## 7.4 BATCHMODE OPERATION

As mentioned previously, batchmode operation refers to running many simulations without user interaction. Script files can be modified to achieve this type of operation using 'C' like script functions. This example creates a script which will perform 27 simulations according to the flow chart below.

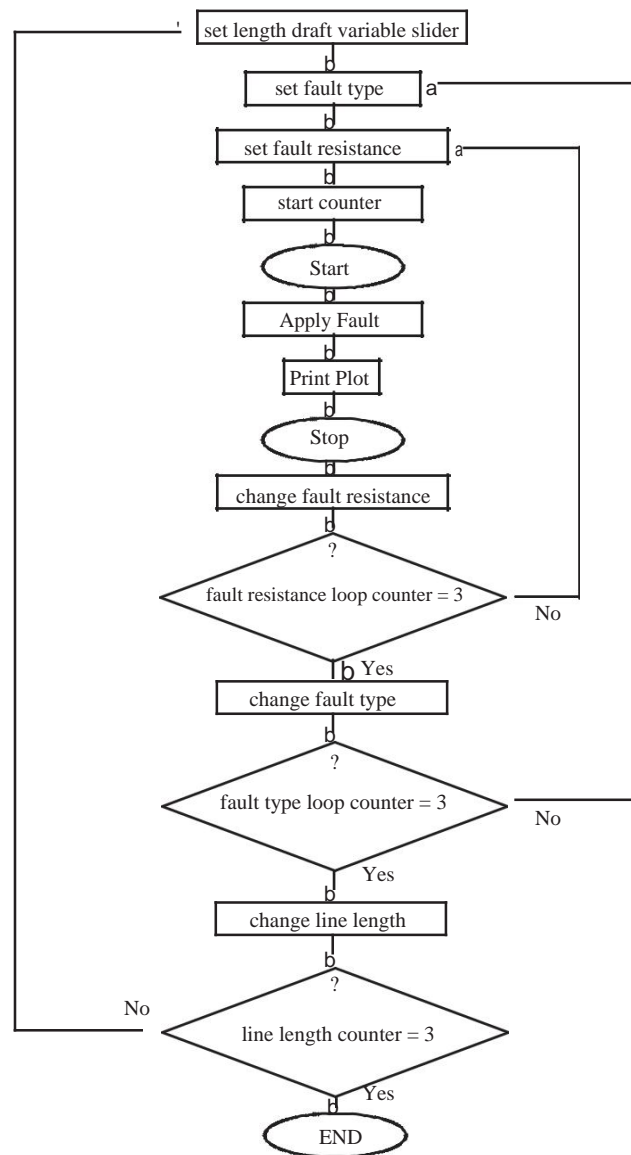


Figure 7.1

Typically when creating a script file, a base script would be created first using the record feature. The base script can then be modified to operate according to the flow

chart. For example, the following RunTime operations could be recorded in a script file.

S set the length draft variable slider

S set the fault resistance draft variable slider

S set the A phase to ground fault switch (AG) to 1


S set the B and C phase to ground fault switches (BG & CG) to 0

S start the simulation

S apply the fault by pressing the Fault pushbutton

S print the plot

S stop the simulation

The base script file can now be edited as shown below. To edit the script file select the  button from the script toolbar. Specific lines have been numbered in this

example, so they can be referenced and discussed later in the chapter.

```

/*****
/*          Variable Declaration          */

int j,k,i,loop_counter;                /* LINE 1 */
float llength[3],res[3];
int fault_typeA[3],fault_typeB[3],fault_typeC[3];

/*****
/*          Initialization          */

llength[0] = 40.0;                      /* LINE 2 */
llength[1] = 50.0;
llength[2] = 80.0;

res[0] = 0.1;
res[1] = 1.0;
res[2] = 5.0;

fault_typeA[0] = 1;
fault_typeA[1] = 0;
fault_typeA[2] = 0;

fault_typeB[0] = 0;
fault_typeB[1] = 2;
fault_typeB[2] = 0;

fault_typeC[0] = 0;
fault_typeC[1] = 0;
fault_typeC[2] = 4;

/*****
for (j=0;j<3;j++)                        /* LINE 3 */

```

```

{
/* Set the preprocessor variable length */
SetSlider "DraftVariables : length" = llength[j]; /* LINE 4 */

for (k=0;k<3;k++)
{
/* Set the fault type */
SetSwitch "Subsystem #1 : CTLs : Inputs : AG" = fault_typeA[k];
SetSwitch "Subsystem #1 : CTLs : Inputs : BG" = fault_typeB[k];
SetSwitch "Subsystem #1 : CTLs : Inputs : CG" = fault_typeC[k];

for (i=0;i<3;i++)
{
loop_counter++; /* LINE 5 */
*

/* Set the fault resistance */
SetSlider "DraftVariables : Ron" = res[i]; fprintf(stdmsg,"Running
Simulation Case Number %d\n",loop_count-
er); /* LINE 6 */ Start;

SUSPEND 2.694;
PushButton "Subsystem #1 : CTLs : Inputs : Fault";
SUSPEND 0.261;
ReleaseButton "Subsystem #1 : CTLs : Inputs : Fault";
PlotPrint "Fault Bus";
Stop;
}
}
}

```

S Line1 ĩ all variables must be declared before they are used within the script.

S Line2 ĩ llength array is initialized so that the fault bus will be moved to 40%,50% and 80% of the line. Arrays named `fault_type` are initialized so the first fault applied is an A phase to ground followed by a B phase to ground and then a C phase to ground fault. Three different fault resistances have been chosen 0.1,1.0 and 5.0 ohms and array `res` is initialized with these values.

S Line3 ĩ three for loops are added. The outer loop will be executed three times according to the parameters entered in the for loop. Each time the loop is executed the length preprocessor variable is changed. The next for loop will be executed three times. Each time the fault type will be changed. The inner for loop will also be executed three times. In each loop, the fault resistance will be changed, simulation started, fault applied, and plot printed.


S Line4 ĩ simulator commands can accept variables as arguments. The SetSlider command has been set to the llength array. The value of "j" determines what value the draft variable length slider will be assigned.

S Line5 ĩfor each simulation case that is run, the loop\_counter is incremented.

S Line6 ĩthe fprintf function has been added so that the statement "Running simulation Case Number #" will be written to the RunTime message window during the script run.

After the script file has been modified, play back the script. Notice the fprintf message in the RunTime message window as the script is running. The script will run all 27 cases according to the flow chart in Figure 7.1.

During the run, it may be required to obtain data from the simulation to be used within the script. Simulation signals must be declared in the script as external type variables. An external variable option is available in the script toolbar to write external variables to a script file. The external variable option is only available during a record session. To add an external variable declaration to an existing batchmode script,

S select the append button  from the script toolbar. The append feature starts a record session but will write the new commands to the bottom of the script file and will not overwrite the existing text.

S select the external variable option  from the script toolbar. A signal selector dialogue box will open.

S add N7 as an external variable

S stop the record session

The script file should now contain an external variable declaration similar to that shown below.

```
external "Subsystem #1 : Node Voltages : S1) N7";
```

In the external variable declaration, the name appearing in quotes is the actual variable name of the signal. This variable is very long and would be inconvenient to use within the script. A new variable name can be assigned by placing the variable following the ending quotes as shown.

```
external "Subsystem #1 : Node Voltages : S1) N7"Volts;
```

If node voltage 7 is referenced from within the script file, the variable name Volts would be used. The external variable declaration can appear anywhere within the script file, but it must appear before the variable is used. Typically variable declarations would appear at the top of a file. Move the external variable declaration to the top of the script file and add the script functions shown in bold type to determine the peak steady state value of node voltage 7 during the simulation.

```
float llength[3],res[3],pk_volts;  
.  
.  
.  
  
Start;  
UpdatePlots;  
pk_volts = arraymax(Volts);  
fprintf(stdout,"Fault Bus Voltage = %f kV\n",pk_volts);  
.
```



## *Script File Example*

.

The fprintf statement is writing the peak voltage to a file named stdout. This file can be viewed in RSCAD/RunTime by selecting the view output button from the script file toolbar.

For further details and descriptions on script operations and functions, please refer to the RSCAD/RunTime online help.