Development of Data-Editor for Electro-Magnetic Transients Program

Naoto NAGAOKA [1], Daisuke TATSUDA, Akihiro AMETANI [1]

(1) Dept. of Electrical Engineering, Doshisha University, Kyotanabe, Kyoto, 610-0321, Japan
(e-mail: nnagaoka@mail.doshisha.ac.jp, aametani@mail.doshisha.ac.jp)

***Abstract* – This paper describes an editor for data of the Electro-Magnetic Transients Program (EMTP). The editor can run on Microsoft Windows in addition to Unix operating system, which support the Java Virtual Machine. An interactive syntax check enables to know error while inputting data. The function is realized by a multiple threads feature of the Java language. The data-structure is easily recognized by colored keywords. The type of the recognized circuit element is expressed in the status bar, and also as a tool tip. The editor has another window, which displays an electric manual or a theory of the model for enhancing the convenience of users.**

Keywords – EMTP, Editor, Java, Thread, Electric Rulebook

I. Introduction

Electro-Magnetic Transients Program (EMTP) is widely used for numerical simulations not only of power systems but also of electronic circuits, because the EMTP adopts a generalized solution scheme and has many circuit models.[1],[2] The high capability makes its data structure complicate. Some data input tools using Graphical User Interface (GUI) have been developed for an easy simulation. There is no doubt about the usefulness of the GUI, a character-based editor is still informative and helpful to make use of data accumulated in the past years.

This paper presents a data-editor, which is useful both for beginners and for specialists of the EMTP. The editor is written in Java to achieve "Write Once, Run Anywhere™"[3],[4]. The editor can run not only on Microsoft Windows, but also on Mac OSX and Linux.

The editor adopts the Java Foundation Classes (JFC) and the Swing[5],[6], which are a set of Java class libraries provided to support building GUI for Java applications. The Swing provides many standard GUI components such as buttons, lists, menus, and text areas. It also includes containers such as windows and tool bars

All of the JFC technologies are fully internationalized, so developers can easily build applications that can interact with users around the world using the user's own language and cultural conventions.

II. Structure of Emtp Editor

The EMTP has been widely used in many countries and it runs both on Windows and on Unix operating system. This is a thorny matter for a developer of supporting routines of the EMTP. If a developer takes the Windows Operating System (OS), technical supports to Unix users become difficult. The developed editor is written in Java for clearing the issues. Because the Java language is "platform-neutral," the programmer can develop without regard for operating systems and also computer manufacturers. In addition to the OS matter, the Java automatically tailors locale-specific matter according to the conventions of the end user's language and region. A text within a user interface is the most obvious example of the locale-specific data. For example, a button should expressed in different languages. The Java gives the correct label according to the OS language without any code.

Fig. 1 illustrates a structure of the EMTP-Editor. The editor has two windows, which are an Editor Desktop and a Help Window. A dialog box will be displayed when the user invokes a command such as Open, Save or Print File, or Search & Replace.

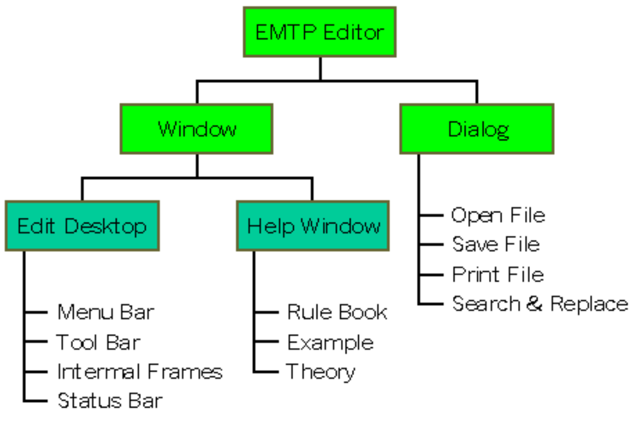


Fig. 1 Structure of "EMTP-Editor"

Fig. 2 shows the Editor Desktop, which is a main window of the editor. The desktop consists of a menu, a tool and a status bar, and some internal frames, which display data decks of the EMTP.

A. Menu Bar

The editor supports commands shown in Table 1. These commands are common in a standard text editor except "Exec". The menu accepts a shortcut that activates a menu item from the keyboard. Each shortcut is expressed by a keystroke combination, which is a modifier key and a mnemonic character, like Alt-F. The mnemonic is expressed as an underlined alphanumeric character in a menu title.

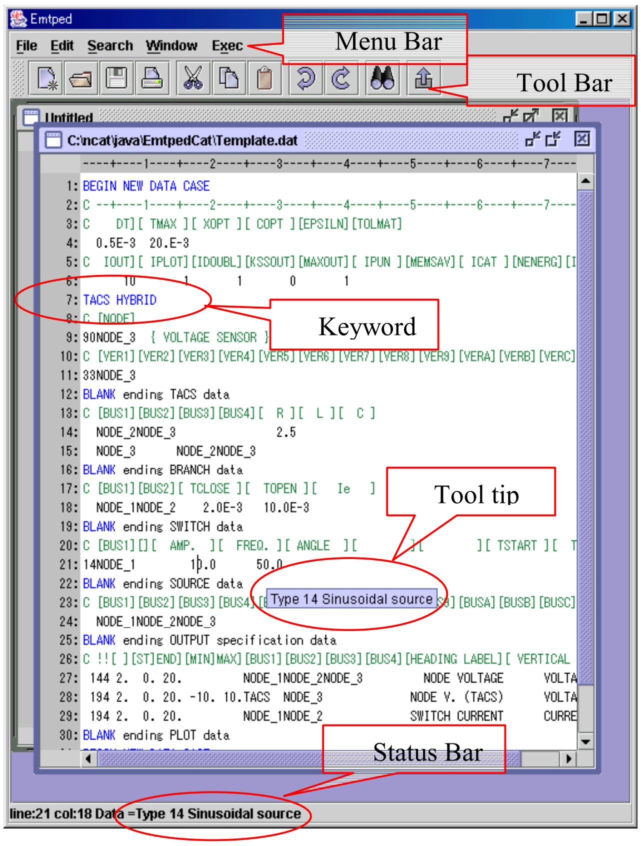
The Exec command invokes the EMTP and directly sends a data deck, which is displayed in an editing internal frame, to the EMTP. Because the editor just internally invokes a command as shown in Table 2, the user can run an appropriate version of the EMTP by creating the command file. Some sample command files for major versions of the EMTP will be distributed with the editor.

Table 1 Menu and commands

File	Edit	Search & Replace	Window	Exec
New^{1}	Cut^{5}	Search^{10}	Cascade	Invoke EMTP^{11}
Open^{2}	Copy^{6}		Tile	
Save^{3}	Paste^{7}		file 1	
Print^{4}	Undo^{8}		:	
Exit	Redo^{9}		file n	

Table 2 Exec command

OS	Command
Windows 9x	command.com /c start edRunTp
Windows NT, 2000, XP	cmd.exe/c edRunTp
Others (Unix)	edRunTp


Emtped
File Edit Search Window Exec
Menu Bar
Tool Bar
Untitled
C:\ncat\java\EmtpedCat\Template.dat
Keyword
Tool tip
Type 14 Sinusoidal source
Status Bar

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----
BEGIN NEW DATA CASE
C --+----1----+----2----+----3----+----4----+----5----+----6----+----7----
C    DT][ TMAX ][ XOPT ][ COPT ][EPSILN][TOLMAT]
  0.5E-3  20.E-3
C  IOUT][ IPLOT][IDOUBL][KSSOUT][MAXOUT][ IPUN ][MEMSAV][ ICAT ][NENERG][I
      10       1       1       0       1
TACS HYBRID
C [NODE]
90NODE_3  { VOLTAGE SENSOR }
C [VER1][VER2][VER3][VER4][VER5][VER6][VER7][VER8][VER9][VERA][VERB][VERC]
33NODE_3
BLANK ending TACS data
C [BUS1][BUS2][BUS3][BUS4][  R ][  L ][  C ]
  NODE_2NODE_3              2.5
  NODE_3      NODE_2NODE_3
BLANK ending BRANCH data
C [BUS1][BUS2][ TCLOSE ][  TOPEN ][   Ie   ]
  NODE_1NODE_2    2.0E-3   10.0E-3
BLANK ending SWITCH data
C [BUS1][][  AMP.  ][  FREQ. ][ ANGLE  ][        ][        ][ TSTART ][  T
14NODE_1        10.0      50.0
BLANK ending SOURCE data
C [BUS1][BUS2][BUS3][BUS4][                    ][BUSA][BUSB][BUSC]
  NODE_1NODE_2NODE_3
BLANK ending OUTPUT specification data
C !![ ][ST]END][MIN]MAX][BUS1][BUS2][BUS3][BUS4][HEADING LABEL][ VERTICAL
 144 2.  0. 20.          NODE_1NODE_2NODE_3         NODE VOLTAGE      VOLTA
 194 2.  0. 20. -10. 10.TACS  NODE_3               NODE V. (TACS)     VOLTA
 194 2.  0. 20.          NODE_1NODE_2              SWITCH CURRENT     CURRE
BLANK ending PLOT data
```

```
line:21 col:18 Data =Type 14 Sinusoidal source
```

Fig. 2 Editor desktop

B. Tool Bar

Fig. 3 illustrates the tool bar. The numbers illustrated below the figure denote commands expressed in Table 1. The tool bar can be dragged out into a separate window by the user (Undocked Tool Bar). When the user of the program pauses with the cursor over the buttons, the tool tip for the button comes up.

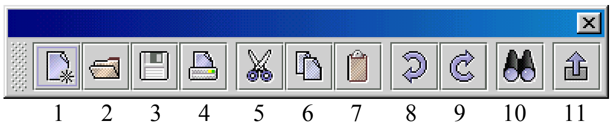


Fig. 3 Tool bar

C. Internal Frames

The editor supports a multiple document interface (MDI) which provides multiple internal frames (data decks). The user can easily compare two or more files. Each internal frame has a title bar and standard window controls (minimize, maximize and close controls). A file name is displayed in the title bar as shown in Fig. 2.

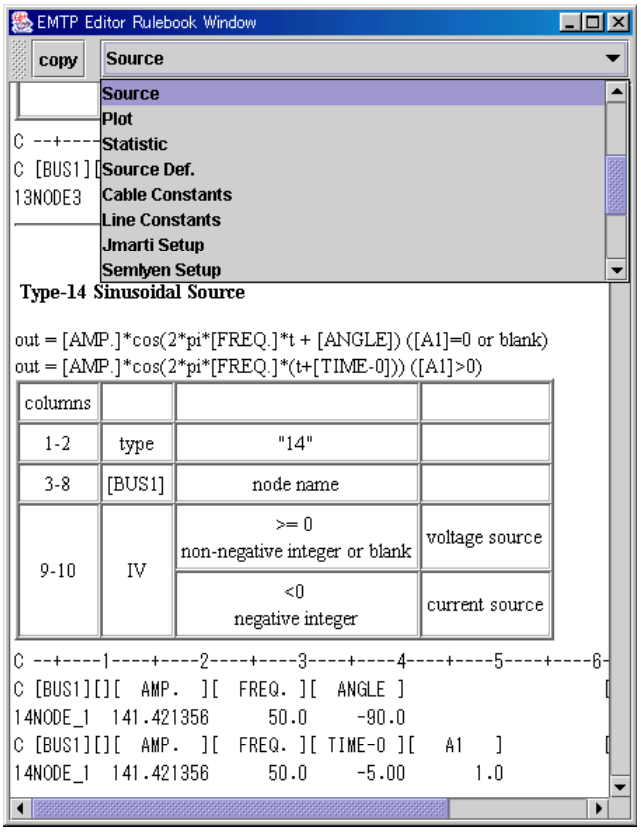
The data structure is easily recognized by colored keywords and a text in a status bar and in a tool tip. This feature is achieved by an interactive syntax check, which is described in the next chapter. The present version executes a shallow syntax check for speeding-up the execution. The supported data of the present version are listed in Appendix.

D. Status Bar

Status bar displays the caret position and the type of the circuit element, which is pointed by the caret.

E. Help Window

When the user presses a right button of a mouse on a line, a rule book windows comes up as shown in Fig. 4. Because the editor knows the type of the EMTP element, a correct page is displayed. This feature is very useful not only for a beginner but also for a professional expert.



Type-14 Sinusoidal Source

out = [AMP.]*cos(2*pi*[FREQ.]*t + [ANGLE]) ([A1]=0 or blank)
out = [AMP.]*cos(2*pi*[FREQ.]*(t+[TIME-0])) ([A1]>0)

columns			
1-2	type	"14"	
3-8	[BUS1]	node name	
9-10	IV	>= 0 non-negative integer or blank	voltage source
		<0 negative integer	current source

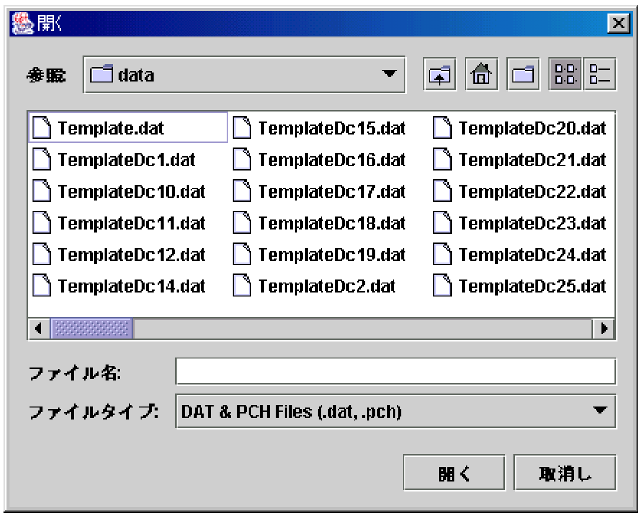
```
C --+----1----+----2----+----3----+----4----+----5----+----6-
C [BUS1][][  AMP.  ][  FREQ. ][  ANGLE ]
14NODE_1  141.421356      50.0      -90.0
C [BUS1][][  AMP.  ][  FREQ. ][ TIME-0 ][   A1   ]
14NODE_1  141.421356      50.0      -5.00        1.0
```

Fig. 4 Rulebook window

The “Copy Button” within the tool bar of the Help Window copies the selected area into a system clipboard. Thus, the example in the rulebook can be paste to the editor window. The user creates a data deck without the thick and heavy printed rulebook. Because the electrical rulebook is written in Hyper Text Markup Language (HTML), the user can customize and translate to any language.

F. Dialog Boxes

Fig. 5 illustrates Open Dialog Box, which comes up when the user enters “Open” command. The dialog box has a typical user interface for open a file. The labels, for example “File Name”, are expressed by Japanese characters in the figure, because the editor knows the locale data of the OS. If the editor runs on an OS of English version, these messages are displayed in English.



参照: data

Template.dat	TemplateDc15.dat	TemplateDc20.dat
TemplateDc1.dat	TemplateDc16.dat	TemplateDc21.dat
TemplateDc10.dat	TemplateDc17.dat	TemplateDc22.dat
TemplateDc11.dat	TemplateDc18.dat	TemplateDc23.dat
TemplateDc12.dat	TemplateDc19.dat	TemplateDc24.dat
TemplateDc14.dat	TemplateDc2.dat	TemplateDc25.dat

ファイル名:

ファイルタイプ: DAT & PCH Files (.dat, .pch)

Fig. 5 Open dialog box

G. Distribution

Installation is tedious work for beginners. The developed program consists from more than 100 files which are some class files, HTML files and so on. However, the distribution files is only an executable Java Archive (JAR) file with a short “Read_me” file.

The JAR is a platform-independent file format that aggregates many files into one. Java byte codes (class files) and their requisite components, such as images and HTML files, can be bundled in a JAR file. The whole material can be downloaded by the user in a single transaction. The JAR format also supports compression, which reduces the file size, and improves the download time. The file size of the current version is less than 200 kB and is small comparing with the typical program developed by the other language. Because the program written in Java runs through the Java Runtime Environment (JRE), the executable files only holds the Java byte code and does not have enormous executable code of the library. The user can start the editor just after copying the executable JAR file by clicking the icon of the file. It will take 10 to 15 seconds to display the Edit Desktop for invoking the JRE (Java Virtual Machine).

III. Structure Analysis

A. Data base

The proposed program has a database, which stores information of each element. Table 3 shows the fields of the database. The “Code” is expressed by a binary number and each bit denotes characteristic of the data. For example, some bits express the level of the data (such as branch, switch, source, or plot data) and another bit expresses whether the record includes a keyword or not.

Table 3 Database

Field	Comment	Example
Element Name	Index Key	Pi_Equivalents
Code	32bit Integer	03010000 (Hex)
Tool Tip	Tool Tip Message	“Pi_Equivalents”
Color	Color Table Number	0 (black)
Color Line Flag	If Keyword Then False	true
Help File Name	Chapter of Rule Book	branch__.html
Anchor	Section of Rule Book	Pi_Equivalents

B. Model-View Separation

The Swing architecture, which is adopted by the editor, is rooted in the model-view-controller (MVC) design that dates back to SmallTalk. MVC architecture calls for a visual application to be broken up into three separate parts:

(1) A model that represents the data for the application.
(2) The view that is the visual representation of that data.
(3) A controller that takes user input on the view and translates that to changes in the model.

The architecture is useful for simplifying testing and improving accessibility, in general. It is also useful in a text component because printed views and screen views should be formatted differently when the text is displayed.

The text within the Java can be stored with attributes. The developed editor is used “Font Color” attribute and a user-defined attribute to save the “Code” shown in Table 3. If the attribute is set to the text data, the text is automatically displayed using the stored attribute in the Java. The editor accesses the Code through the stored attribute and retrieves the related data such as name of HTML file for a rulebook.

C. Multiple Threads[6]

The data structure of the EMTP becomes complicate to handle by the new editor when the data case becomes large. This fact force to consume processing times for analyzing the structure of the data and for detecting errors. Otherwise, the response of the editor should be as quick as possible for a comfortable usability of the program. The developed editor adopts a multiple threads scheme to satisfy the contradicting requirements.

Most GUI work naturally occurs in a thread called “event-dispatching thread” in the Java. Once the GUI is visible, most programs are driven by events such as button actions or mouse clicks, which are always handled in the event-dispatching thread. The Java is designed in conformity to the “single-thread rule.” Thus, the events are dispatched in a predictable order from the same event queue as mouse and keyboard events, timer events, and paint requests.

Although there are several advantages in executing all of the user interface code in a single thread, a background thread to perform time-consuming operations without affecting the performance of GUI is required for the editor. It will take a long time for analyzing structure of data of the EMTP, and text is displayed according to the result of the analysis. For example, keyword such as "BEGIN NEW DATA CASE" should be displayed as a colored text.

D. Algorithm

Fig. 6 illustrates a flow chart for the structure analysis of a data deck. If a keyboard event for an insertion or a deletion of a character is detected, the event handler kills a thread, which is previously invoked for an analysis of the structure. Because it is generally impossible to finish the analysis between the keyboard events, the previous analysis should be interrupted by the subsequent keyboard event. A new thread is created and started after a confirmation of the termination of the thread.

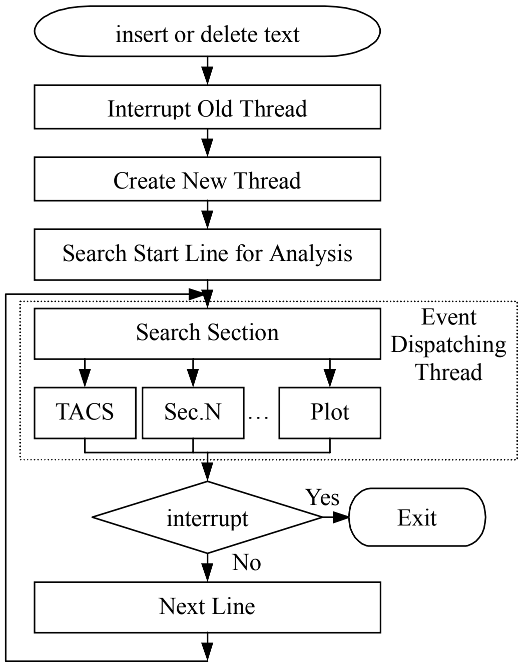


Fig. 6 Flow chart for analyzing structure

Strictly speaking, the structure analysis has to be restarted from a top of the data deck, when a user inputs or deletes a character. To reduce a load of the CPU, the program try to localize the analyzing area.

The following operation related to the GUI is processed within the event-dispatch thread to keep consistency of the GUI. The analysis is carried out on a line-by-line basis. At the first stage, the editor determines a section, which is estimated to be included in the present line by the help of a "BLANK" line. Within the section, the element of the EMTP is mainly determined by the type-code entered at the beginning of the line. If the editor find the element, the program looks in the database and sets its text color and binary code into an attribute field of the text component of the Java. The text color is automatically changed by the Swing libraries.

If the line analysis is finished, the program checks the interrupt flag, which is set by the keyboard event handler. If the continuous editing is detected, the thread for analyzing the old data deck is aborted.

IV. CONCLUSIONS

This paper describes a character-based text editor specialized for the EMTP. The editor is platform independent from viewpoints of hardware and also from an operating system. The feature is achieved by the Java language. Although Java brings no advantage to a numerical processing field, the feature of the language is suitable for developers of supporting routines of the EMTP. The platform-neutral technology releases a developer from tedious conversion works.

Although the developed editor is designed for conservative EMTP users, it is also useful for beginners. The editor has an electric manual written in HTML, and the file can be easily translated into any language. The editor opens appropriate page of the rulebook automatically. In addition, the editor tells the error by changing text color. The recognized circuit element is expressed in the status bar, and also a tool tip for the line comes up when the user pauses with the mouse cursor over any of the lines. These features may be helpful for beginners. The property is achieved by an interactive syntax check with a multiple threads programming for improving the execution speed of the check as high as possible.

The editor has basic commands for a typical editor and the most commands are invoked from a tool bar by clicking a button. The editor also has an "Exec" command, which invokes the EMTP. The users can simulate a system without exiting the editor program.

ACKNOWLEDGMENTS

The authors are most appreciative of the contributions of Dr. Y. Baba for useful suggestion to this project.

The authors wish to express our gratitude to the students of the Power System Analysis Laboratory of Doshisha University, especially M. Nayel, T. Kawakami and A. Miyamoto for testing the program.

REFERENCES

[1] W. Scott-Meyer, "EMTP Rule Book," B.P.A., 1977
[2] H. W. Dommel, "EMTP Theory Book," B.P.A., 1986
[3] Matt Curtin, "Write Once, Run Anywhere^{TM}: Why It Matters," http://java.sun.com/features/1998/01/wora.html
[4] Sun Microsystems, Inc "Java^{TM}2 SDK, Standard Edition Documentation Version 1.4.1, " http://java.sun.com/j2se/1.4.1/docs/index.html
[5] S. Pantham, "Pure JFC Swing", Sams Publishing, 1999.
[6] Sun Microsystems, Inc "The Swing Connection," http://java.sun.com/products/jfc/tsc/, http://java.sun.com/products/jfc/tsc/articles/threads/threads1.html http://java.sun.com/products/jfc/tsc/articles/threads/threads2.html

APPENDIX

List of Supported Data (Ver.1)

1. $Cards

$CLOSE, $DEBUG, $DEPOSIT, $DISABLE, $ENABLE, $ERASE, $LISTOFF, $LISTON, $OPEN, $PUNCH, $SPY, $SPYEND, $STARTUP, $STOP, $UNITS, $VINTAGE, $WIDTH

2. /Cards

/REQUEST, /TACS, /MODELS, /BRANCH, /SWITCH, /SOURCE, /OUTPUT, /INITIAL, /LOAD FLOW, /PLOT,

3. Special Request Cards

59 keywords (from "ABSOLUTE TACS DIMENSIONS" to "ZO")

4. TACS/MODELS

"TACS STAND ALONE", "TACS HYBRID",
order-zero block, n-th order s-block,
All Devices: Code 50-66, FORTRAN expression,
TACS Sources (Type-11, 13, 14, 23, 24, 26, 90, 91, 92 and 93),
TACS initialize (Type-77), TACS output (Type-33) ,
"MODELS", "MODELS STAND ALONE"

5. Branch Data

5-1. Linear Elements

lumped series R-L-C, "NAME :"
pi-circuits, mutually-coupled R-L,
"CASCADED PI", "USE AR", "USE RL"
multi-phase constant-parameter line, JMARTI line,
recursive-convolution line,
"TRANSFORMER", "IDEAL TRANSFORMER",
"TRANSFORMER THREE PHASE",
"KIZILCAY F-DEPENDENT",

5-2. Nonlinear Elements

Nonlinear R:
Type-91 (TACS controlled and Multiphase Time-Varying),
Type-92 (Exponential and Piecewise-Linear)
Type-97 Time-Varying,
Type-99 Pseudo-Nonlinear

Nonlinear L:
Type-93 True Nonlinear,
Type-96 Pseudo-Nonlinear Hysteretic,
Type-98 Pseudo-Nonlinear,
"TACS CONTROL"

6. Switch

Type-0 Ordinary Switch, Type-11 Diode,
Type-12 Gap and Triac, Type-13 TACS Controlled Switch,
Type-76 STATISTIC switch

7. Sources

Empirical Functions Source,
Static Sources (Type-code from 11 to 18)
Dynamic Sources (Type-19 UM and Type-59 SM),
Connection to TACS Variable (Type-60)

8. Output

All types are supported (Type -5 to 1)

9. Initial conditions

All types are supported (Type 2 to 4)

10. Load Flow

FIX SOURCE power constraint

11. Plot

graph case-title text, graph subheading text, plot data,
"CALCOMP PLOT", "PEN PLOT", "PRINTER PLOT",
"X-Y PLOT" "SCREEN PEN", "SCREEN PLOT",
"BRANCH", "FOURIER OFF", "FOURIER ON",
"SMOOTH", "PRINT HEAD OFF"

12. Others

statistical tabulation, SPY statistical tabulation (FIND-QUIT)

13.Unsuported Data (Ver.1)

$INCLUDE, $ABORT,
continuation request "$$", Free format input
"DATA BASE MODULE", "NODA SETUP",
"LINE MODEL FREQUENCY SCAN",
Parallel Monte Carlo simulation,
Type-93 Nonlinear L "FORTRAN" keyword,
SYSTEMATIC Switch,