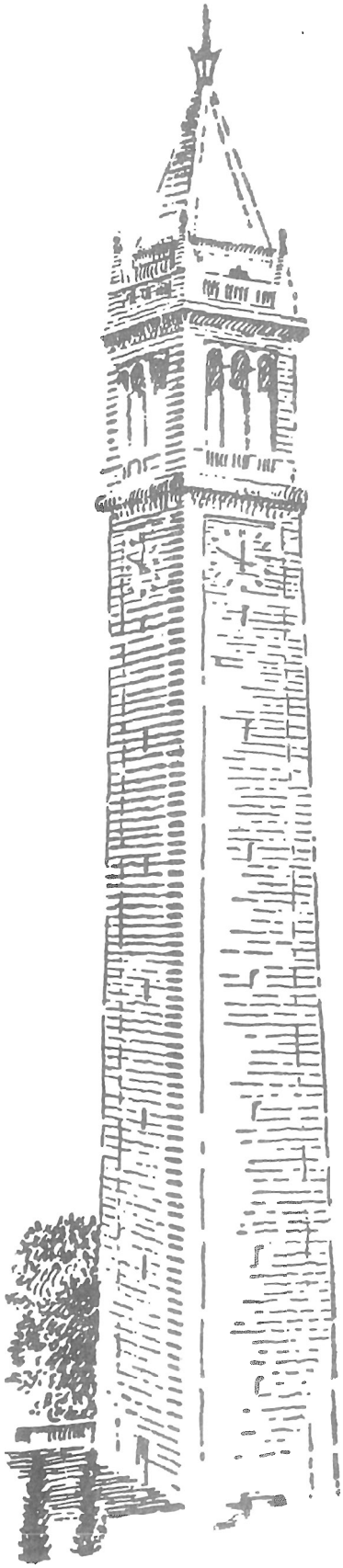

PROGRAM REFERENCE FOR SPICE2

by

Ellis Cohen

Memorandum No. UCB/ERL M592

14 June 1976

ELECTRONICS RESEARCH LABORATORY
College of Engineering
University of California, Berkeley, CA 94720

PROGRAM REFERENCE FOR SPICE2

by

Ellis Cohen

Memorandum No. ERL-M592

14 June 1976

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

ACKNOWLEDGEMENTS

The contributions of many people helped to make SPICE2 what it is today. The research of L. W. Nagel into circuit simulation algorithms and techniques paved the way for most of the analysis methods used in SPICE2. Discussions with S. P. Fan and G. R. Boyle helped in the formulation of the matrix manipulation and nonlinear source implementations. The MOSFET model implementation in SPICE2 is due largely to the efforts of R. Newton and S. Liu. The continuing encouragement and support of my research advisor, Professor D. O. Pederson, is greatly appreciated.

The SPICE2 program could not have been developed without the use of the computer resources provided by the Computer Center of the University of California, Berkeley. The support of ARO Grant DAHC04-74-G0151 and NSF Grant ENG-75-04986 is also acknowledged.

1 Introduction 1

2 Data Management 2

2.1 Description of Data 2
2.2 Previous Method 2
2.3 Design Considerations 2

2.4 Management System Description 4

2.4.1 Definitions 4
2.4.2 Memory Manager Calling Sequences 4
2.4.3 Memory Allocation Algorithms 5

3 Program Structure 7

4 Overlay Root 8

5 Readin Overlay 11

5.1 Scanning 11

5.1.1 Readin 11
5.1.2 Card 12
5.1.3 Runcon 13
5.1.4 Find 13

5.2 Storage 13

5.2.1 Elements 13
5.2.2 Analyses and Outputs 14

5.3 Subcircuit Structures 15
5.4 Forward References 16

6 Errchk Overlay 17

6.1 Forward Reference Check 17
6.2 Compact Element Node Set 18
6.3 Subcircuit Expansion 18
6.4 IUNSAT Resolution 21
6.5 Circuit Summary 21

6.6 Node Table / Topology Check 21
6.7 Breakpoint Table 22

7 Setup Overlay 23

7.1 Matrix Structure 23

7.2 Row-Swap 25
7.3 Reordr 26
7.4 Element Matrix Pointers 26
7.5 Machine Code Generation and Execution 27

8 Analysis 29

8.1 Element Load 29

8.2 Matrix Equation Solution 29

8.2.1 LU Decomposition 30
8.2.2 Forward/Backward Substitution 30

8.3 Devices 31

8.3.1 MOSFET Model 31
8.3.2 Junction Limiting 31

9 Dctran Overlay 33

9.1 Dc Operating Point 33
9.2 Transient Initial Conditions 34
9.3 Dc Transfer Curves 35

9.4 Transient Analysis 35

9.4.1 Numerical Integration Methods 36
9.4.2 Automatic Timestep 36

10 Acan Overlay 38

10.1 Small-signal Analysis 38
10.2 Noise and Distortion Analyses 38

11 Dcop Overlay 39

11.1 Small-signal Transfer Function 39
11.2 Dc Sensitivities 39

12 Ovtpvt Overlay 40

13 Appendices 41

13.1 User's Guide 41

13.2 Linked List Specifications 42

13.2.1 Resistor 43
13.2.2 Capacitor 45
13.2.3 Inductor 46
13.2.4 Mutual Inductance 47
13.2.5 Voltage-controlled Current Source 48

13.2.6 Voltage-controlled Voltage Source 49
13.2.7 Current-controlled Current Source 50
13.2.8 Current-controlled Voltage Source 51
13.2.9 Independent Voltage Source 52
13.2.10 Independent Current Source 53

13.2.11 Diode 54
13.2.12 BJT 55
13.2.13 JFET 56
13.2.14 MOSFET 57
13.2.15 Transmission Line 59

13.2.16 Subcircuit call ('X'-element) 60
13.2.17 Subcircuit Definition 61
13.2.18 Diode Model 63
13.2.19 BJT Model 65
13.2.20 JFET Model 67

13.2.21 MOSFET Model 68
13.2.22 .PRINT dc 70
13.2.23 .PLOT dc 72
13.2.24 Dc Analysis Output Variable 73

13.3 Table Specifications 74

13.3.1 IELMNT 74
13.3.2 ISBCKT 74
13.3.3 IUNSAT 74
13.3.4 ITEMPS 74
13.3.5 IFOUR 75

13.3.6 ISENS 75
13.3.7 IFIELD, ICODE, IDELIM, and ICOLUM 75
13.3.8 JUNODE 75
13.3.9 LSBCKT 76
13.3.10 IORDER, IUR, ITABLE, and ITABID 76

13.3.11 ISR 77
13.3.12 ISEQ 77
13.3.13 ISEQ1 77
13.3.14 NEQN 77
13.3.15 NODEVS 78

13.3.16 NDIAG 78
13.3.17 NMOFFC 78
13.3.18 NUMOFF 78
13.3.19 ISWAP 79
13.3.20 IEQUA 79

13.3.21 IORDER 79
13.3.22 JMNODE 79
13.3.23 IUR 79
13.3.24 IUC 80

13.3.25 ILC 80

13.3.26 ILR 80
13.3.27 MACINS 80
13.3.28 LVNIM1 80
13.3.29 LVN 81
13.3.30 LX0 81

13.3.31 LX1 - LX7 81
13.3.32 LTD 81
13.3.33 LOUTPT 82
13.3.34 NDIAGC 82
13.3.35 LVNC 82

13.3.36 LD0 and LD1 82
13.3.37 LVNCT 83
13.3.38 LOCX 83
13.3.39 LOCY 83

13.4 Labeled-Common Variable Descriptions 84

13.4.1 MEMRY 84
13.4.2 TABINF 85
13.4.3 MISCEL 87
13.4.4 LINE 88
13.4.5 CIRDAT 89

13.4.6 MOSARG 90
13.4.7 STATUS 91
13.4.8 FLAGS 92
13.4.9 KNSTNT 93
13.4.10 DC 94

13.4.11 AC 95
13.4.12 TRAN 96
13.4.13 OUTINF 97
13.4.14 CJE 98
13.4.15 BLANK 99

13.5 MOSFET Equations 100

13.6 Element Load Templates 103

13.6.1 Resistor 103
13.6.2 Capacitor 103
13.6.3 Inductor 103
13.6.4 Voltage-controlled Current Source 104
13.6.5 Voltage-controlled Voltage Source 104

13.6.6 Current-controlled Current Source 104
13.6.7 Current-controlled Voltage Source 104
13.6.8 Independent Voltage Source 105
13.6.9 Independent Current Source 105

13.7 Potential Conversion Problems 106

13.7.1 Call-by-Address 106
13.7.2 Dynamic Region Size 106
13.7.3 Characters-per-Word 107
13.7.4 Words-per-real-value 107
13.7.5 Array Subscripts 107

13.8 Program Listing 109

14 References 110

1 Introduction

The design of high-performance integrated circuits requires an accurate, inexpensive way in which to assess circuit performance. Discrete "breadboard" techniques are inadequate because they do not accurately model the effects of parasitics, thermal coupling, and device matching, which are critical to integrated-circuit performance. Prototype circuit construction must also be ruled out because of the high cost - typically, six man-months and $25000.

As a result of these considerations, the semiconductor industry has turned to the integrated-circuit (IC) simulator to evaluate the performance of circuits during the design stage. The SPICE program was developed to meet the needs of the IC group at Berkeley. The analyses which it performs include dc operating point, nonlinear transient, small-signal frequency-domain, noise, harmonic distortion, and dc sensitivities.

This report describes the internal design of the SPICE2 program (the fundamental theory and algorithms are described in [1]). Chapter 2 describes the dynamic memory allocation technique which is used throughout the program. Chapters 3-12 detail the program structure. Finally, the Appendices (Chapter 13) contain a listing of the User's Guide, specifications for all the internal data structures, descriptions of COMMON variables, and a program listing.

2 Data Management

2.1 Description of Data

As execution proceeds, SPICE requires storage for many different data blocks. Memory must be allocated to hold the circuit description, the linearized circuit equations, operating-point information, charge-storage information for transient (large-signal) analysis, and output variables. In general, both the number and the size of these blocks is a function of the circuit to be analyzed.

2.2 Previous Method

SPICE1 solved this storage problem by allocating a fixed amount of storage in FORTRAN COMMON-blocks for each of a fixed number of data blocks; for example, space for 400 nodes and 25 device models was reserved. This approach had the advantages of simplicity and efficiency, but it suffered from one drawback: storage reserved for one type of data could not be used for another data type. Thus, a 75-node circuit whose description required 26 device models would "overflow memory", even though the total memory available was sufficient.

2.3 Design Considerations

There were four principal considerations in the design of the memory allocation package in SPICE2. First of all, the implementation language had to be FORTRAN - no other language suitable for scientific computation was (or is) as widely available on as many different computers. This consideration required that

the memory to be managed had to consist of a single FORTRAN array; this array is called "VALUE" in SPICE2.

Secondly, both the number and the size of managed data blocks had to be dynamically variable. This consideration eliminated the following type of "dynamic" memory allocation (assuming some fixed number of blocks):

```
DIMENSION A(1000)
 ...
 ... determine sizes of blocks, and store
 ...   in variables I, J, and K
 ...
CALL DOIT(A(1),I,A(1+I),J,A(1+I+J),K)
 ...
END

SUBROUTINE DOIT(A,I,B,J,C,K)
DIMENSION A(I),B(J),C(K)
 ...
 ... all real computation performed nere
 ...
RETURN
END
```

The third consideration was that more than two blocks could be varying in size at the same time. At first, the pattern of memory usage in SPICE2 permitted memory to be managed as a last-in-first-out (LIFO) stack, in which the (only) block whose size was varying at any given moment could be positioned at the top (end) of allocated memory. Subsequent enhancements to SPICE2 eliminated this possibility.

The fourth consideration was the dynamic region-size capability which some computer operating systems support - for example, the SCOPE [2] operating system on Control Data Corporation 6000-series computers. Such a capability offers potential reduc-

tions in both response time and analysis cost sufficient to justify its inclusion in the memory manager.

2.4 Management System Description

2.4.1 Definitions

The memory manager in SPICE controls "tables" using "table pointers". A "table" is a contiguous block of memory; a "table pointer" is a variable which serves both to identify a block (to serve as the block "name") and to indicate the block origin in memory. As an example, consider a table of 100 words with table pointer "LTAB". In SPICE2, the global allocation array is named "VALUE"; the contents of table "LTAB" would be accessed as VALUE(LTAB+1) through VALUE(LTAB+100), regardless of where in VALUE the table happened to be at any given moment.

2.4.2 Memory Manager Calling Sequences

There are 8 entry points in the memory management package: SETMEM, GETMEM, RELMEM, EXTMEM, CLRMEM, SIZMEM, PTRMEM, and CRUNCH. The rest of the SPICE2 program uses only these entry points for all memory allocation and manipulation. A brief description of each entry follows:

entry(arguments)	description
SETMEM(IFAMWA)	initializes the memory manager; the dynamically-managed memory starts at (absolute) address IFAMWA
GETMEM(P,S)	makes a new managed table. P is the table pointer for this new table; the size of the new table is S words.
RELMEM(P,S)	reduces by S words the size of the

table pointed to by table pointer P.

EXTMEM(P,S) extends by S words the size of the table pointed to by table pointer P.

SIZMEM(P,S) sets S to the size (in words) of the table pointed to by table pointer P.

CLRMEM(P) destroys the table pointed to by table pointer P.

PTRMEM(P1,P2) changes the table pointer for the table pointed to by table pointer P1 to be P2.

CRUNCH forces a compaction of the dynamically-managed memory

2.4.3 Memory Allocation Algorithms

The memory manager maintains a "table entry" table at the high-address end of managed memory. This table contains a four-word entry for each allocated table of the form:

word	contents
1	table origin (array subscript)
2	allocated table size (in words)
3	requested block size (in words)
4	address of table pointer

The memory manager uses the redundant information in the "table entry" table for error-checking (verifying, for example, that the value of a table pointer is the same as the appropriate entry in the "table entry" table). Note that any allocated table may in fact be larger than required; that fact would be indicated by an "allocated size" which was greater than the "requested size". The memory manager assumes that arguments are passed "by address" since it uses the address of the table pointer to update its value should the table origin change.

There are two basic algorithms used in the memory manager - one to allocate a new table, the other to extend an existing table. The allocation algorithm can be described by the following:

```
call CRUNCH;
if (memory insufficient) get more memory;
add entry at end of "table entry" table for new table;
store table origin into table pointer variable;
```

The variable MEMAVL (memory available) in the memory manager always contains the number of words of memory which are free (allocated to some block, but more than requested). Therefore the "insufficient memory" test is a simple check of MEMAVL.

The algorithm to extend a table is the following:

```
if (required size > allocated size)
    {   if (insufficient memory) get more memory;
        call COMPRS to move tables below the table
          being extended towards the origin of
          the dynamically-managed memory;
        call COMPRS to move tables above the table
          being extended towards the end of the
          dynamically-managed memory;
    }
store new required size in "table entry" table;
```

The table extension algorithm takes advantage of the fact that when a given table is extended, most of the time the next call on the memory manager will be to extend the same table again. Note that neither algorithm changes the origin of the first table allocated (assuming that it is not subsequently destroyed).

3 Program Structure

The SPICE2 program consists of 15000 FORTRAN and COMPASS statements divided into 7 major overlays together with an overlay root.

A composite memory map for SPICE execution is shown below. At any one moment, only the root and one of the indicated first-level overlays is resident in memory. The dynamically-managed memory begins after the longest overlay (as indicated below).

```
.............................................................
:                                                           :
:                    SPICE overlay root                     :
:                                                           :
:...........................................................:
:        :        :       :        :      :       :        :
: READIN : ERRCHK : SETUP : DCTRAN : DCOP : ACAN : OVTPVT  :
:        :        :       :        :      :       :        :
:        :        :.......:        :......:       :........:
:........:........:       :        :      :......:         :
:                         :        :                       :
:.........................:........:.......................:
:                                                           :<-IFAMWA
:    DYNAMICALLY-MANAGED MEMORY (COMMON-block /BLANK/)      :
:                                                           :
:...........................................................:<-ICORE
```

The variables IFAMWA and ICORE are explained in Sections 2.4.2 and 13.4.1.

Each section of the program is discussed separately in the following chapters of this report.

4 Overlay Root

The SPICE overlay root drives the rest of the program, calling the first-level overlays to process the circuit description, analyze the circuit, and output the simulation results. The root consists of the main program together with routines TITLE, SETMEM, COMPRS, MEMPTR, DMPMEM, TMPUPD, MAGPHS, XOR, OUTNAM, ALFNUM, FIND, MEMORY, DCDCMP, DCSOL, MOVE, COPY, and ZERO. The main loop is described in the following figure:

```
initialize;
while (.not. end-of-input)
{   call READIN;   call ERRCHK;   call SETUP;
    repeat
        {   TEMP = next analysis temperature;   call TMPUPD;
            if (dc transfer curves requested)
                {   MODE = 1;   MODEDC = 3;
                    call DCTRAN;   call OVTPVT;   }
            if (dc operating point or ac analysis requested)
                {   MODE = 1;   MODEDC = 1;
                    call DCTRAN;   call DCOP;   }
            if (ac analysis requested)
                {   MODE = 3;   call ACAN;   call OVTPVT;   }
            if (transient analysis requested)
                {   MODE = 1;   MODEDC = 2;
                    call DCTRAN;   call DCOP;
                    MODE = 2;
                    call DCTRAN;   call OVTPVT;   }
        }
    until (no more analysis temperatures);
}
stop
```

The READIN overlay reads input until either a line containing '.END' is read or end-of-file is found. As the overlay executes, it builds the linked-lists which SPICE uses to store the circuit description, and it sets certain flags in COMMON to indicate what analyses were requested. In the ERRCHK overlay, SPICE expands the subcircuit calls, prints a circuit summary, and

checks for common input errors. The SETUP overlay constructs the pointers which are used to manipulate the sparse equation coefficient matrix in both the DCTRAN and ACAN analysis overlays. This overlay also generates the machine code which is executed to solve the circuit equations.

The main analysis loop consists of the "repeat" loop in the above figure. For each analysis temperature, it checks successively for a dc transfer curve analysis, an dc operating-point analysis, an ac (small-signal) analysis, and a transient (large-signal) analysis. The dc operating-point analysis is always performed if an ac small-signal analysis is requested because SPICE requires the small-signal equivalent models for the nonlinear elements in order to perform the ac analysis.

The DCTRAN overlay performs the dc transfer curve, dc operating-point, transient initial-condition, and transient analyses. Exactly which of these analyses it performs is determined by the values of the COMMON variables MODE and MODEDC. The ACAN overlay performs the ac analysis. Finally, the OVTPVT overlay generates the tabular output listings and line-printer plots from the results of the multi-point analyses.

The overlay root contains several utility subroutines which are used throughout the rest of the program. A brief description of those routines follows (A and B are array names):

name(arguments)	description
MOVE(A,I,B,J,N)	moves N characters from B to A, starting at the Jth and Ith character positions respectively; eight characters/word is assumed
COPY4(A,B,N)	moves N integers from A to B; if A and B overlap in memory - for example, if B is really A(10), to move data within a single array - COPY4 copies either A(1)-to-A(N) or A(N)-to-A(1) to prevent the transfer from destroying the data
COPY8(A,B,N)	COPY4(), but for real variables
COPY16(A,B,N)	COPY4(), but for complex variables
ZERO4(A,N)	zeroes A(1) - A(N) integers
ZERO8(A,N)	ZERO4(), but for real variables
ZERO16(A,N)	ZERO4(), but for complex variables

5 Readin Overlay

The READIN overlay performs the initial input-processing for SPICE. This overlay consists of the routines READIN, KEYSRC, EXTNAM, RUNCON, OUTDEF, CARD, GETLIN, and NXTCHR. It builds the linked-lists which describe the circuit to be analyzed, and it sets certain flags in COMMON to indicate which analyses have been requested. A flowchart for the overlay follows:

```
read title;  if (end-of-file found) exit;
initialize;
call CARD;   if (end-of-file found) exit;
repeat
{   switch (type of input line)
    {   element description:    process in READIN;
                    ".ENDS":    process in READIN;
                  ".MODEL":     process in READIN;
                    ".END":     exit;
                 ".SUBCKT":     process in READIN;
            other "." line:     call RUNCON;
    }
    call CARD;
}   until (end-of-file found);
exit;
```

5.1 Scanning

5.1.1 Readin

The READIN subroutine processes element and device-model definitions. The routine calls CARD to scan the physical input lines. Subroutine FIND is called to preset storage for each input element. Analysis and output directives are processed by RUNCON.

5.1.2 Card

Subroutine CARD is called by READIN to scan the next logical input line. Since physical lines may be logically continued (by putting a '+' in column one of the next physical line), CARD assumes on entry that the next line has already been read from input, and always leaves the next input line (unscanned except for the initial '+' check) in the array AFIELD (in COMMON-block LINE).

Each logical input line is parsed into a set of "numbers" and "names". A field is assumed to be a "name" if it begins with either an alphabetic character or a period (".") followed by an alphabetic character. A field is a "number" if it begins with either a digit (0-9) or a period (".") followed by a digit. Fields are delimited by any character in the set {" ", ",", "=", "(", ")"}; in the present version, all delimiters are equivalent except with processing output variable definitions (see subroutine OUTDEF). The results of the scan are stored in the tables IFIELD, ICODE, IDELIM, and ICOLUM, all of which track with each field of the line (the first word of each of the tables refers to the first field, the second word for the second field, etc.). The entry in ICODE is +1, 0, or -1 to indicate "name", "number", and end-of-line. The IFIELD entries are either a floating-point number or a name stored in 8H format (left-justified, blank-filled to 8 characters). The IDELIM table contains the field delimiter character. Finally, the ICOLUM table contains the column number at which the field began in the input line.

5.1.3 Runcon

The RUNCON subroutine processes run control lines - for example, a ".TRAN" card indicating that a transient analysis is to be done. It sets the pertinent flags in COMMON to indicate which analyses SPICE is to perform (see Section 13.4 for a description of the COMMON flags). Some error-checking is performed - for example, that required analysis parameters have not been omitted.

5.1.4 Find

The FIND subroutine is used to locate a particular list element or to allocate space if the element cannot be found. It has four parameters, described in the following:

parameter	description
ANAME	name of list element
ID	number of list to search
LOC	set by FIND to location of element
IFORCE	1 => element must not already exist

If NSBCKT is nonzero, a subcircuit definition is being read. In that event, FIND searches the current subcircuit definition list (pointed to by the subcircuit whose definition is pointed to by NODPLC(ISBCKT+NSBCKT)) instead of the nominal element list (pointed to by LOCATE(ID)).

5.2 Storage

5.2.1 Elements

All input data to SPICE2 is maintained in a series of linked lists stored in table IELMNT. The LOCATE array (in COMMON-block CIRDAT) contains pointers (subscripts to the NODPLC array) to the

beginning of each list. By convention, the integer- and real-valued data are accessed using separate subscripts for the (equivalenced) arrays NODPLC and VALUE. Each list entry contains a "next-pointer" which is the NODPLC-subscript of the next list element (zero if none). Each entry also contains a VALUE-subscript (usually called LOCV), which points to the first word of real-valued storage for the list element.

A detailed description of every linked-list element structure is contained in Section 13.2.

5.2.2 Analyses and Outputs

SPICE2 performs three principal analyses: dc, ac, and transient. For each of these analyses, a separate COMMON-block is used to store pertinent information. For example, COMMON-block /TRAN/ contains variables TSTEP, TSTOP, and TSTART, which are set to the print-step, final simulation time, and starting simulation time, respectively. A detailed description of the COMMON-variables is contained in Section 13.4.

Analysis output variables are stored in table LOUTPT as analysis proceeds. After each sweep point (of some source (dc), frequence (ac), or time (transient)) is solved, the LOUTPT table is extended and the values of the output variables at that point stored away. The variable ISEQ (see Section 13.2.24) contains the offset for the particular output variable within each extension of the LOUTPT table. Since the results of a given analysis are printed before another analysis is begun, only one output table is necessary. The output variable definitions themselves

are also stored in linked-list form in table IELMNT (with ID=41 through 45).

5.3 Subcircuit Structures

During READIN, each subcircuit definition is stored as a linked list of elements and device models. The "subckt info" field in each element of a subcircuit definition list contains the element ID. Subcircuits are completely local entities in SPICE2; elements or device models introduced within a subcircuit definition are not known outside that definition. As a result, if the line

```
Q1 1 2 3 MODA
```

is read as part of a subcircuit definition, SPICE does not in general know where device model MODA is defined (it may be defined later within this subcircuit definition, or it may be defined in some enclosing subcircuit definition, or it may be defined in the nominal circuit definition). To solve this problem correctly, an "unsatisfied name" table, IUNSAT, was added; this table contains all the names (such as "MODA"). During READIN, the various element definitions contain pointers into the IUNSAT table. Proper resolution of name references is made in subroutine LNKREF (in the ERRCHK overlay) and is discussed in Section 6.4.

5.4 Forward References

For efficiency reasons, the input file is read only once by SPICE. As a result, a reference to some element may be read before the element definition itself. An example of this situation is shown below:

```
...
.DC VIN 0 5 .5
... (other input)
VIN 4 0 PULSE(0 5)
...
```

When the ".DC" line is read, FIND is called with the "force-action" parameter IFORCE set to 0. The effect of this call is to add an "undefined but referenced" element with name "VIN" to the circuit. For such a case, subroutine FIND sets the (LOC + 2)th word of the list entry to ".UN" (for "undefined"). (Properly speaking, the IUNSAT table should also be used for this purpose: at present, elements within subcircuits are not treated in the same manner as elements in the nominal circuit description.)

6 Errchk Overlay

The ERRCHK overlay finishes the processing of input data and performs miscellaneous error checking. The overlay consists of the routines ERRCHK, SHLSRT, PUTNOD, GETNOD, SUBCKT, FNDNAM, NEWNOD, ADDELT, CPYTAB, LNKREF, SUBNAM, ELPRNT, MODCHK, and TOPCHK. A flowchart describing the actions of this overlay follows:

```
check forward references;
generate compact node set;
call SUBCKT to expand subcircuit calls;
call LNKREF to link-up external names;
set source function defaults and limits;
call ELPRNT to print circuit element summary;
call MODCHK to print and preprocess device models;
call TOPCHK to check topology and print node table;
invert resistance values;
change K to M for mutual inductance;
generate breakpoint table;
check analysis limits;
sequence output variables;
```

6.1 Forward Reference Check

The forward-reference problem is explained in Section 5.4. In ERRCHK, all that need be done is to sequence through all the element lists checking whether the (LOC + 2)th word of any entry has the value ".UN". Since all element lists store either a node number or other small integer in the (LOC + 2)th position, the presence of that value indicates that the element was never actually read in, but only referenced.

6.2 Compact Element Node Set

After the READIN overlay has completed, the circuit element list entries contain the input node numbers. Since the analysis portion of SPICE2 requires a compact, consecutive set of node numbers, ERRCHK renumbers all element nodes and stores the new node numbers in place of the user node numbers. A translation table, JUNODE, is constructed and used when SPICE has to print the original (user) node number; if NODPLC(LOC+2) contains the renumbered node number, NODPLC(JUNODE+NODPLC(LOC+2)) contains the input node number.

6.3 Subcircuit Expansion

At the end of execution of the READIN overlay, all subcircuit definitions have been read in and stored on the ID=20 linked list, and all subcircuit calls have been read in and stored on the ID=19 list. Furthermore, before subcircuit expansion is performed, the initial compact node set has been constructed, so that all user node numbers (and all internal node numbers) are known. This last point is important since additional (nonconflicting) "user" node numbers may have to be generated during subcircuit expansion.

All elements and device models, contained within any subcircuit definition are treated as purely local to that definition by SPICE2. Consider the following input:

```
...
X1 1 2 3 SUB1
.SUBCKT SUB1 17 15 23
Q1 17 3 15 QMOD
.MODEL QMOD37 NPN()
Q2 23 17 6 QMOD37
R123 3 6 1K
X1 17 23 67 SUB2
.ENDS SUB1
...
...
...
.MODEL QMOD37 PNP()
Q5 1 17 45 QMOD37
...
```

Note that QMOD, the device model for subcircuit element Q1, is not defined within the subcircuit. SPICE2 will search enclosing subcircuit definitions (if any), and then the nominal circuit description, for that device model definition. Note that although QMOD37 is defined both within the subcircuit definition and outside that definition, the transistor Q2 within the subcircuit will use the device model definition contained within the subcircuit, and transistor Q5 will not. (For transistor Q5, SPICE2 will try to find a device model QMOD37 which is defined in an outer, enclosing subcircuit definition (if any) or within the nominal circuit description.)

The actual expansion algorithm used in subroutine SUBCKT is as follows:

```
LOCX = LOCATE(19); (pointer to first subckt call)
while (LOCX is nonzero)
{   call FNDNAM to determine subcircuit;
    check for recursive definition;
    LOC = first subcircuit element;
    while (LOC is nonzero)
    {   call FIND to add element space;
        set "subckt info" appropriately;
        call ADDELT to add element data;
        LOC = pointer to next subckt element;
    }
    LOCX = pointer to next X-element;
}
```

Subcircuits are expanded "top-down" since any "X-element" within a subcircuit definition being expanded will be added to the end of the subcircuit call list. The test for recursion checks that the X-element which caused a particular subcircuit to be expanded was not itself added as a result of a previous expansion of that subcircuit. The test is simplified by the fact that "subckt info" for the "X-element" causing the expansion points to the subcircuit definition of which it was a part (if any).

The call to subroutine FIND is somewhat modified since no naming conflict is possible at this point in execution. Therefore, a unique "element name" is constructed from the number of elements of the type being added.

The FNDNAM subroutine determines what element is meant by some name in the IUNSAT table (see Section 5.3). The routine first searches the subcircuit expansion list (which is obtained from the "subckt info" word of each list entry) in reverse order. If no match is found, FNDNAM then searches the nominal circuit description list.

6.4 IUNSAT Resolution

After subcircuit expansion, "unsatisfied" name references must be resolved before the complete circuit description is available (see Section 5.3). This resolution is performed by subroutine LNKREF, which sequences through the linked lists, calling FNDNAM to resolve the reference(s) for each circuit element.

6.5 Circuit Summary

Before printing the circuit summary, the ERRCHK subroutine first makes a pass through the independent sources, setting default function values. Then subroutine ELPRNT is called, which prints out a summary of the circuit elements.

To print out the device model summary, subroutine MODCHK is called. It first assigns default model parameters from the default parameter list DEFVAL. The device models are then printed. After printing the parameter values, MODCHK performs one-time model parameter computations, such as replacing resistances by conductances (details regarding the pre-processing are contained in Sections 13.2.18 - 13.2.21). Finally, MODCHK generates additional node numbers for those nodes added to accommodate nonzero device model resistors.

6.6 Node Table / Topology check

Subroutine TOPCHK is called by ERRCHK to construct the circuit node table and to check that three topological constraints are satisfied by the circuit to be analyzed: no voltage source and/or inductor loop, no current source and/or capacitor cutset, and no node with less than two branches connected exists.

The node table is constructed using tables IUR, ITABLE, and ITABID. The entries in the IUR table are offsets within the ITABLE and ITABID tables. The entries in the ITABLE table are pointers to circuit elements; entries in ITABID are the element ID for the corresponding pointer in ITABLE. After the tables have been constructed, the elements connected to node number N are pointed to by

```
NODPLC(ITABLE+NODPLC(IUR+N))
        through
NODPLC(ITABLE+NODPLC(IUR+N+1)-1)
```

6.7 Breakpoint Table

In transient analysis, SPICE2 always uses a program-calculated timestep, regardless of the user-specified print interval. However, the independent source waveforms frequently have sharp transitions which could cause an unnecessary reduction in the timestep in order to find the exact transition time. To overcome this problem, ERRCHK generates a "breakpoint" table, LSBKPT, which contains a sorted list of all the transition points of the independent sources. During transient analysis, whenever the next timepoint is sufficiently close to one of the breakpoints, the timestep is adjusted so that the program lands exactly on the breakpoint.

7 Setup Overlay

The SETUP overlay generates the pointer arrays necessary to manipulate the sparse equation matrix used in all of the analyses. The overlay consists of routines SETUP, MATPTR, RESERV, REORDR, MATLOC, INDEX, CODGEN, and MINS. A flowchart of this overlay follows:

```
NSTOP = number of equations;
call MATPTR to establish matrix structure;
call REORDR to reorder equations for sparsity;
call MATLOC to store matrix locations;
if (machine code requested)
    call CODGEN to perform code generation;
```

The subroutines MATPTR, REORDR, and MATLOC respectively establish the equation matrix structure, reorder the equations to maximize sparsity, and store the matrix pointers for each circuit element. The CODGEN subroutine generates machine instructions to solve the set of equations.

7.1 Matrix Structure

The equation matrix is very sparse - typically, over 85% of its entries are zero. The matrix structure is stored internally in two different forms during the setup process. Initially, the nonzero matrix entries are recorded using a set of linked lists, one for each row of the matrix, together with some auxiliary tables. The i'th entry of the ISR table is a pointer to the beginning of a linked list, each of whose entries records the number of some nonzero column in the i'th row. The linked list entries themselves are stored at the end of the NUMOFF table.

The auxiliary tables used are NDIAG, NUMOFF, and NMOFFC. The NDIAG table records whether or not the diagonal matrix elements are nonzero, the first N words of the NUMOFF table record the number of nonzero columns in each row of the matrix, and the NMOFFC table contains the number of nonzero rows in each column of the matrix.

The linked-list structure used for matrix setup has the advantage that only a constant (small) overhead is required to modify the matrix structure. However, the linked list method is not desirable for analysis (given that SPICE2 does not use pivoting) since such a structure increases both memory and cpu requirements. Because of these considerations, after the final matrix structure has been established, SPICE generates a compact matrix pointer system.

The compact pointer system divides the equation matrix into three parts: the matrix diagonal, an upper-, and a lower-triangular section. The diagonal is stored separately since it is not sparse. For the upper triangle, the tables IUR and IUC are used. Consider the i'th row of the equation matrix; let I1 = NODPLC(IUR+i) and let I2 = NODPLC(IUR+i+1)-1. Then the numbers of the nonzero columns in the upper-triangular part of the i'th row are stored in NODPLC(IUC+I1) through NODPLC(IUC+I2). (If I1>I2 then there are no such columns.) The actual matrix entries are stored in table LVN "in parallel" to the IUC table. In other words, having found the offset within the IUC table for some (i,j) matrix element, than the actual location within the compressed matrix storage of the element value is immediately

known. The lower-triangular part of the matrix is stored using a dual set of matrix pointers, in tables ILC (the ILR index) and ILR.

7.2 Row-Swap

SPICE2 uses the Modified Nodal formulation [3], [4] to construct the set of equations describing the input circuit. Although that set of equations is nonsingular for any circuit satisfying the topological constraints mentioned in Section 6.6, problems may still arise when no provision for pivoting is incorporated in the program. These problems arise because the subset of equations added to solve for currents flowing in "voltage-defined" branches may be singular [1, pp. 67-69].

Basically, the solution to this singularity problem is to swap the matrix row which solves for a branch current with the corresponding row which solves for the voltage at the positive node of that branch. However, the order in which rows are swapped is not arbitrary if there are trees of voltage-defined elements. The algorithm used in SPICE to decide on the order of row-swaps is the following:

Tables already constructed:

NODEVS:	# of voltage-defined elements connected to node
NDIAG:	nonzero if conductance connected to node
ISEQ:	vector of pointers to all voltage-defined elements

let NUMVDE = # of voltage-defined elements

```
for (I = 1 to NUMVDE)
{   FOUND = .false.;
    for (J = I to NUMVDE)
    {   (examine element pointed to by ISEQ(J))
        for (NODE = each of the 2 element nodes)
        {   if (NODE = ground) continue;
            if (NODEVS(NODE) > 1) continue;
            if (NDIAG(NODE) = 0) go to swap;
            FOUND = .true.; save J;
        }
    }
    if (FOUND = .false.)
        {   abort - voltage-defined element loop exists   }
swap:
    exchange rows for element indicated by J;
    delete element from ISEQ table;
}
```

7.3 Reorder

As previously noted, the typical equation matrix is very sparse. Howver, the order in which unknowns are solved can have a considerable effect on sparsity due to the generation and propagation of "fill-in" terms (zero matrix entries which become nonzero during the LU decomposition). Various algorithms to decide on the order of variable elimination have been proposed (Barry [5], Markowitz [6], Nakhla [7]). SPICE2 uses the Markowitz algorithm; the number of matrix terms per row and column, which the method requires, is obtained from the NUMOFF and NMOFFC tables, respectively.

7.4 Element Matrix Pointers

After the final matrix structure has been established, subroutine MATLOC is called. To minimize element load time, the offsets within the LVN table to each matrix term which a given circuit element increments are stored with each circuit element. In this way, the overhead of mapping an (i,j) matrix location to

its proper offset within the LVN table is incurred only once. For example, a resistor between nodes N1 and N2 causes an addition (or subtraction) to matrix terms (N1,N1), (N1,N2), (N2,N1), and (N2,N2). The locations of each of these terms is stored with each resistor.

7.5 Machine Code Generation and Execution

Subroutines CODGEN and MINS are used to generate the loopless machine instructions which are executed to perform the LU decomposition and forward/backward substitution steps which solve the linearized circuit equations. The code generation process is simplified by noting that two forms of assignment statements can be used to accomplish the entire matrix solution; these forms are:

A = A/B and A = A - B*C

for different values of A, B, and C. The technique used in SPICE2 is to perform a mock equation solution, generating the appropriate machine instructions in place of performing the actual arithmetic. Subroutine CODEXC is used as an interface to the generated machine code, since standard FORTRAN contains no provision for calling subroutines which have not been previously compiled.

Timing comparisons indicate that the code generated for the CDC 6400 computer at the University of California at Berkeley executes between 3 and 4 times faster than the equivalent FORTRAN subroutine (which must contain additional code to again map (i,j) matrix entries to their location within the LVN table as the

equations are solved). However, the code can occupy a considerable amount of memory: for the UA741, approximately 2000 words. Whether or not to use the generated machine code depends on memory availability and computer resource charging algorithms.

8 Analysis

This section briefly describes the element load, equation solution, and junction limiting algorithms in SPICE2.

8.1 Element Load

The construction of the circuit equation coefficient matrix can be performed incrementally, one element at a time (see, for example, [8]). A concise description of where each element type adds to the matrix can be found in Section 13.6.

8.2 Matrix Equation Solution

The solution technique for the system of linearized circuit equations is to first perform an LU factorization of the coefficient matrix, followed by forward and backward substitution steps to solve for a particular right-hand side forcing vector. This technique has two main advantages over simple Gaussian elimination. First of all, multiple right-hand sides can be solved with much less effort, since the L and U factors need to be computed only once. Secondly, SPICE2 uses the adjoint analysis technique [9] to efficiently perform small-signal noise and distortion, and dc sensitivity analyses. The equation matrix of the adjoint circuit is the transpose of the matrix for the original circuit; conveniently, the L and U factors of the transposed matrix are the transpose of the original L and U factors. Thus, the adjoint analyses may be performed for just the cost of a forward and backward substitution step using the transpose of the already-factored equation matrix.

8.2.1 LU Decompositon

As noted, SPICE2 factors the coefficient matrix into lower- and upper-triangular matrices L and U. The actual algorithm used for the factorization is so arranged that the L and U factors occupy the same memory locations as the original matrix. The algorithm is given below:

```
Transform "Ax = b" into "LUx = b"

NSTOP = number of circuit equations;  note that the ground
        node is included in the matrix but not solved.

for (I = 2 to NSTOP)
{   for (J = I+1 to NSTOP)
    {   A(J,I) = A(J,I)/A(I,I);
        for (K = I+1 to NSTOP)
        {   A(J,K) = A(J,K) - A(J,I)*A(I,K);    }
    }
}

Result:  A(I,J) = U(I,J) for J > I
                = L(I,J) for J < I
```

8.2.2 Forward/Backward Substitution

The algorithm used in SPICE2 to perform the forward and backward substitution steps follows:

Forward substitution:

evaluate: $y = Ux = L^{-1} b$

```
for (I = 2 to NSTOP)
{   for (J = I+1 to NSTOP)
    {   B(J) = B(J) - L(J,I)*B(I);    }
}
```

Backward substitution:

$$\text{evaluate:} \quad x = U^{-1} y = U^{-1} L^{-1} b$$

```
for (I = NSTOP to 2 by -1)
{   for (J = I+1 to NSTOP)
    {   B(I) = B(I) - U(I,J);   }
    B(I) = B(I)/U(I,I);
}
```

8.3 Devices

SPICE2 contains built-in models for the p-n junction diode, the bipolar junction transistor (BJT), the junction field-effect transistor (JFET), and the metal-oxide-semiconductor field-effect transistor (MOSFET). The model equations for the diode, BJT, and JFET are described in [1].

8.3.1 MOSFET Model

The MOSFET model used in SPICE2 is derived from the Frohman-Grove model [10], with the addition of some subthreshold [11] and short-channel [12] effects. The nonlinear oxide capacitance formulation of J. Meyer [13] is included. The actual equations used are given in Section 13.5.

8.3.2 Junction Limiting

The Newton-Raphson algorithm used to aid convergence can run into numerical problems due to the exponential nonlinearities in the semiconductor device models. To solve this difficulty, additional constraints are placed on the per-iteration change in device voltages. Subroutine PNJLIM contains (and most concisely describes) the limiting algorithm used for p-n junction voltages;

subroutine FETLIM is used to limit per-iteration changes in FET voltages.

9 Dctran Overlay

The DCTRAN overlay is the largest one in SPICE2; it performs the dc transfer curve, dc operating point, initial transient operating point, and transient analyses. The overlay consists of routines DCTRAN, COMCOF, TRUNC, TERR, SORUPD, ITER8, CODEXC, LOAD, NLCSRC, UPDATE, EVPOLY, EVTERM, NXTPWR, INTGR8, PNJLIM, DIODE, BJT, FETLIM, JFET, MOSFET, MOSEQN, and MOSCAP.

9.1 DC Operating Point

The dc operating point computation is indicated when variables MODE and MODEDC have values 1 and 1, respectively. A flowchart for this analysis follows:

```
initialize;
TIME = 0;
call SORUPD to set sources to time-zero values;
INITF = 2;
call ITER8;
if (converged)
    {   INITF = 4;
        call DIODE; call BJT; call JFET; call MOSFET;
        print operating-point solution;
    }
```

The actual Newton-Raphson iteration is controlled by subroutine ITER8. A flowchart for that subroutine follows:

```
IGOOF = ITERNO = NDRFLO = NONCON = 0;
DONE = .false.;
while (not DONE)
    {   call LOAD;
        if ((NOSOLV is nonzero) and
            (analysis = initial transient)) exit;
        ITERNO = ITERNO + 1;
        switch (INITF) of
            { "1":  if (NONCON = 0) exit; goto solve;
              "2":  INITF = 3; goto solve;
              "3":  if (NONCON = 0) INITF = 1; goto solve;
```

```
        "4","5","6":  INITF = 1;
              }
    solve:
        if (ITERNO > iteration limit) exit;
        call DCDCMP; call DCSOL;
        if (IGOOF > 0)
            {   IGOOF = 0; NDRFLO = NDRFLO + 1;    }
        if (NONCON = 0)
            {   check node voltages for convergence;
                if (voltages converged) DONE = .true.;    }
    }
```

At the conclusion of the dc operating-point analysis (as a result of the calls to the device modeling routines with INITF set to 4), the LX0 table contains all the small-signal device parameters, which are subsequently printed out by subroutine DCOP (in the DCOP overlay).

9.2 Transient Initial-Conditions

The transient initial-condition computation establishes a consistent set of circuit conditions prior to the start of the transient analysis. (By "consistent" is meant conditions which satisfy Kirchhoff's laws.) The DCTRAN overlay performs this analysis when MODE and MODEDC have values of 1 and 2, respectively. A flowchart for the initial transient solution algorithm follows:

```
initialize;
TIME = 0;
call SORUPD to set sources to time-zero values;
INITF = 2;
call ITER8;
if (converged)
    {   print solution;    }
```

9.3 DC Transfer Curves

The dc transfer curve computation is simply a repetitive dc operating point computation performed for a range of values for one independent voltage or current source in the circuit. The analysis is performed when MODE and MODEDC have values of 1 and 3, respectively. A flowchart for this analysis follows:

```
initialize;
TIME = 0;
call SORUPD to set sources to time-zero values;
INITF = 2;
for (each source value)
    {   call ITER8;
        if (not converged) stop analysis;
        store outputs;
        INITF = 6;
    }
(the OVTPVT overlay is called from the overlay root
    to output any requested plots)
```

The value of 6 for INITF causes a linear extrapolation from the previous solutions to obtain the initial guess used for the subsequent iteration; that is, the initial guess for sweep point n+1 is the linear extrapolation of the solutions at sweep points n-1 and n.

9.4 Transient analysis

The transient analysis is performed when MODE has a value of 2. A flowchart of the analysis steps follows:

```
        initialize; TIME = 0; DELTA = TSTEP; INITF = 5;
savout: store outputs in LOUTPT table;
        if (TIME > TSTOP) exit;
        adjust DELTA for breakpoint table values;
newtim: TIME = TIME + DELTA;
        call SORUPD;
        if (INITF not equal to 5) INITF = 6;
        call ITER8;
```

```
        if (converged) goto tsterr;
        TIME = TIME - DELTA; DELTA = DELTA/8; goto tstdel;
tsterr: call TRUNC;
        if (error acceptable) goto savout;
        TIME = TIME - DELTA;
        DELTA = DELNEW (computed in TRUNC);
tstdel: if (DELTA < DELMIN) stop analysis;
        goto newtim;
```

9.4.1 Numerical Integration Methods

SPICE2 uses either Trapezoidal [14] or variable-order Gear [15] to perform the numerical integration necessary for the transient analysis. The choice is based on the value of the COMMON-variable METHOD.

9.4.2 Automatic Timestep

There are two methods available to the user to control the computation of the appropriate transient analysis timestep: iteration count and truncation error estimation. The iteration count method uses the number of Newton-Raphson iterations required to converge at a timepoint as a measure of the rate of change of the circuit. If the number of iterations is less than ITL3, the timestep is doubled; if greater than ITL4, the timestep is divided by 8. This method has the advantage of minimal computation overhead; however, it suffers in that it does not take into account the true rate-of-change of circuit variables.

The second method of timestep determination is based on the estimation of local truncation error. The appropriate function derivative is approximated by a divided difference - hence the "estimation." Although requiring somewhat more cpu time than

iteration count, the truncation error estimation method allows a meaningful error-bound to be set on the computed output values.

10 Acan Overlay

The ACAN overlay performs the small-signal, noise, and distortion The overlay contains routines ACAN, ACDCMP, ACSOL, ACLOAD, NOISE, ACASOL, DINIT, and DISTO. All analyses performed use the linearized models for all the nonlinear devices. A flowchart of this overlay follows:

```
initialize;
for (FREQ = FSTART to FSTOP)
    {   call ACLOAD;
        call ACDCMP;
        call ACSOL;
        save small-signal outputs;
        if (noise analysis requested) call NOISE;
        if (distortion analysis requested) call DISTO;
    }
(the OVTPVT overlay is called from the overlay root
    to output any requested plots)
```

10.1 Small-Signal Analysis

The small-signal analysis is performed at each frequency point by first calling subroutine ACLOAD. This subroutine zeroes and then loads the complex-valued equation matrix using the results of the dc operating point analysis stored in the LX0 table. Then subroutines ACDCMP and ACSOL are called, which perform an LU decomposition and forward/backward substitution on the set of circuit equations to obtain its solution.

10.2 Noise and Distortion Analyses

The noise analysis used in SPICE2 is described by references [16] and [17]. The distortion analysis algorithm is described in [18, 19, 20].

11 Dcop Overlay

The DCOP overlay performs three functions: it prints out the device operating point information, computes the small-signal transfer function, and computes dc sensitivities of output variables with respect to circuit parameters. The overlay contains routines DCOP, SSTF, SENCAL, and ASOL.

11.1 Small-Signal Transfer Function

The small-signal transfer function analysis is performed by subroutine SSTF; it evaluates the gain, input resistance, and output resistance of a specified two-port using the linearized device models computed as a part of the dc operating point analysis.

11.2 DC Sensitivities

The dc sensitivities of output variables with respect to circuit parameters are computed by subroutine SENCAL. The analysis is performed using the adjoint technique [9], thus requiring only one equation solution per output variable (rather than per circuit parameter).

12 OVTPVT overlay

The OVTPVT overlay generates the tabular listings and line-printer plots displaying the results of the multi-point analyses. It contains routines OVTPVT, NTRPL8, SETPRN, SETPLT, PLOT, SCALE, and FOURAN. The overlay is called after each multi-point analysis, to avoid the necessity of storing unnecessary output values in memory.

Subroutine FOURAN is used to calculate the Fourier coefficients of transient analysis output waveforms. The coefficients are computed recursively using a "truncated infinite series" algorithm [1, pp. 254-259].

13 Appendices

13.1 User's Guide

UNIVERSITY OF CALIFORNIA
COLLEGE OF ENGINEERING
DEPARTMENT OF ELECTRICAL ENGINEERING
AND COMPUTER SCIENCES

E. COHEN
D. O. PEDERSON

VERSION 2D.0 (31MAY76)

USER*S GUIDE FOR SPICE

SPICE IS A GENERAL-PURPOSE CIRCUIT SIMULATION PROGRAM FOR NONLINEAR DC, NONLINEAR TRANSIENT, AND LINEAR AC ANALYSES. CIRCUITS MAY CONTAIN RESISTORS, CAPACITORS, INDUCTORS, MUTUAL INDUCTORS, INDEPENDENT VOLTAGE AND CURRENT SOURCES, FOUR TYPES OF DEPENDENT SOURCES, TRANSMISSION LINES, AND THE FOUR MOST COMMON SEMICONDUCTOR DEVICES: DIODES, BJTS, JFETS, AND MOSFETS.

SPICE HAS BUILT-IN MODELS FOR THE SEMICONDUCTOR DEVICES, AND THE USER NEED SPECIFY ONLY THE PERTINENT MODEL PARAMETER VALUES. THE MODEL FOR THE BJT IS BASED ON THE INTEGRAL CHARGE MODEL OF GUMMEL AND POON; HOWEVER, IF THE GUMMEL-POON PARAMETERS ARE NOT SPECIFIED, THE MODEL REDUCES TO THE SIMPLER EBERS-MOLL MODEL. IN EITHER CASE, CHARGE STORAGE EFFECTS, OHMIC RESISTANCES, AND A CURRENT-DEPENDENT OUTPUT CONDUCTANCE MAY BE INCLUDED. THE DIODE MODEL CAN BE USED FOR EITHER JUNCTION DIODES OR SCHOTTKY BARRIER DIODES. THE JFET MODEL IS BASED ON THE FET MODEL OF SHICHMAN AND HODGES. THE MODEL FOR THE MOSFET IS BASED ON THE FROHMAN-GROVE MODEL; HOWEVER, CHANNEL-LENGTH MODULATION, SUB-THRESHOLD CONDUCTION, AND SOME SHORT-CHANNEL EFFECTS ARE INCLUDED.

PROGRAM LIMITATIONS

SPICE USES DYNAMIC MEMORY MANAGEMENT TO STORE ELEMENTS, MODELS, AND OUTPUT VALUES. THUS, THE ONLY LIMITATION IMPOSED BY THE PROGRAM ON THE SIZE OR THE COMPLEXITY OF THE CIRCUIT TO BE SIMULATED IS THAT ALL NECESSARY DATA FIT IN MEMORY. FOR EXAMPLE, A 100-POINT TRANSIENT ANALYSIS OF THE UA741 OPERATIONAL AMPLIFIER REQUIRES APPROXIMATELY 50000 (OCTAL) WORDS OF MEMORY ON THE CDC 6400 AT THE UNIVERSITY OF CALIFORNIA AT BERKELEY.

---- TRANSIENT ANALYSIS

THE TRANSIENT ANALYSIS PORTION OF SPICE COMPUTES THE TRANSIENT OUTPUT VARIABLES AS A FUNCTION OF TIME OVER A USER-SPECIFIED TIME INTERVAL. THE INITIAL CONDITIONS ARE AUTOMATICALLY DETERMINED BY A DC ANALYSIS. ALL SOURCES WHICH ARE NOT TIME DEPENDENT (FOR EXAMPLE, POWER SUPPLIES) ARE SET TO THEIR DC VALUE. FOR LARGE-SIGNAL SINUSOIDAL SIMULATIONS, A FOURIER ANALYSIS OF THE OUTPUT WAVEFORM CAN BE SPECIFIED TO OBTAIN THE FREQUENCY DOMAIN FOURIER COEFFICIENTS. THE TRANSIENT TIME INTERVAL AND THE FOURIER ANALYSIS OPTIONS ARE SPECIFIED ON THE .TRAN AND .FOURIER CONTROL CARDS.

---- ANALYSIS AT DIFFERENT TEMPERATURES

ALL INPUT DATA FOR SPICE IS ASSUMED TO HAVE BEEN MEASURED AT 27 DEG C (300 DEG K). THE SIMULATION ALSO ASSUMES A NOMINAL TERMPERATURE OF 27 DEG C. THE CIRCUIT CAN BE SIMULATED AT OTHER TEMPERATURES BY USING A .TEMP CONTROL CARD.

TEMPERATURE APPEARS EXPLICITLY IN THE EXPONENTIAL TERMS OF THE BJT AND DIODE MODEL EQUATIONS. IN ADDITION, SATURATION CURRENTS HAVE A BUILT-IN TEMPERATURE DEPENDENCE. THE TEMPERATURE DEPENDENCE OF THE SATURATION CURRENT IN THE BJT MODELS IS DETERMINED BY:

```
IS(TEMP) = IO*(TEMP*PT)*EXP(-Q*EG/(K*TEMP))
```

WHERE K IS BOLTZMANS CONSTANT, Q IS THE ELECTRONIC CHARGE, IO IS A CONSTANT, EG IS THE ENERGY GAP WHICH IS A MODEL PARAMETER, AND PT IS THE SATURATION CURRENT TEMPERATURE EXPONENT (ALSO A MODEL PARAMETER, AND USUALLY EQUAL TO 3). THE TEMPERATURE DEPENDENCE OF THE SATURATION CURRENT IN THE JUNCTION DIODE MODEL IS DETERMINED BY:

```
IS(TEMP) = IO*(TEMP*(PT/N))*EXP(-Q*EG/(K*N*TEMP))
```

WHERE N IS THE EMISSION COEFFICIENT, WHICH IS A MODEL PARAMETER, AND THE OTHER SYMBOLS HAVE THE SAME MEANING AS ABOVE. NOTE THAT FOR SCHOTTKY BARRIER DIODES, THE VALUE OF THE SATURATION CURRENT TEMPERATURE EXPONENT, PT, IS USUALLY 2.

TEMPERATURE APPEARS EXPLICITLY IN THE VALUE OF JUNCTION POTENTIAL, PHI, FOR ALL THE DEVICE MODELS. THE TEMPERATURE DEPENDENCE IS DETERMINED BY:

```
PHI(TEMP) = K*TEMP/Q*LOG(NA*ND/NI(TEMP)*2)
```

WHERE K IS BOLTZMANS CONSTANT, Q IS THE ELECTRONIC CHARGE, NA IS THE ACCEPTOR IMPURITY DENSITY, ND IS THE DONOR IMPURITY DENSITY, NI IS THE INTRINSIC CONCENTRATION, AND EG IS THE ENERGY GAP.

TEMPERATURE APPEARS EXPLICITLY IN THE VALUE OF SURFACE MOBILITY, UO, FOR THE MOSFET MODEL. THE TEMPERATURE DEPENDENCE IS DETERMINED BY:

```
UO(TEMP) = UO(TNOM)/(TEMP/TNOM)*(1.5)
```

TEMPERATURE APPEARS EXPLICITLY FOR RESISTOR VALUES ACCORDING TO THE FOLLOWING EXPRESSION:

```
VALUE(TEMP) = VALUE(TNOM)*(1+TC1*(TEMP-TNOM)+TC2*(TEMP-TNOM)*2))
```

WHERE TEMP IS THE CIRCUIT TEMPERATURE, TNOM IS THE NOMINAL TEMPERATURE, AND TC1 AND TC2 ARE THE FIRST- AND SECOND-ORDER TEMPERATURE COEFFICIENTS.

TYPES OF ANALYSIS

---- DC ANALYSIS

THE DC ANALYSIS PORTION OF SPICE DETERMINES THE DC OPERATING POINT OF THE CIRCUIT WITH INDUCTORS SHORTED AND CAPACITORS OPENED. A DC ANALYSIS IS AUTOMATICALLY PERFORMED PRIOR TO A TRANSIENT ANALYSIS TO DETERMINE THE TRANSIENT INITIAL CONDITIONS, AND PRIOR TO AN AC SMALL-SIGNAL ANALYSIS TO DETERMINE THE LINEARIZED, SMALL-SIGNAL MODELS FOR NONLINEAR DEVICES. IF REQUESTED, THE DC SMALL-SIGNAL VALUE OF A TRANSFER FUNCTION (RATIO OF OUTPUT VARIABLE TO INPUT SOURCE), INPUT RESISTANCE, AND OUTPUT RESISTANCE WILL ALSO BE COMPUTED AS A PART OF THE DC SOLUTION. THE DC ANALYSIS CAN ALSO BE USED TO GENERATE DC TRANSFER CURVES: A SPECIFIED INDEPENDENT VOLTAGE OR CURRENT SOURCE IS STEPPED OVER A USER-SPECIFIED RANGE AND THE DC OUTPUT VARIABLES ARE STORED FOR EACH SEQUENTIAL SOURCE VALUE. IF REQUESTED, SPICE ALSO WILL DETERMINE THE DC SMALL-SIGNAL SENSITIVITIES OF SPECIFIED OUTPUT VARIABLES WITH RESPECT TO CIRCUIT PARAMETERS. THE DC ANALYSIS OPTIONS ARE SPECIFIED ON THE .DC, .TF, .OP, AND .SENS CONTROL CARDS.

---- AC SMALL-SIGNAL ANALYSIS

THE AC SMALL-SIGNAL PORTION OF SPICE COMPUTES THE AC OUTPUT VARIABLES AS A FUNCTION OF FREQUENCY. THE PROGRAM FIRST COMPUTES THE DC OPERATING POINT OF THE CIRCUIT AND DETERMINES LINEARIZED, SMALL-SIGNAL MODELS FOR ALL OF THE NONLINEAR DEVICES IN THE CIRCUIT. THE RESULTANT LINEAR CIRCUIT IS THEN ANALYZED OVER A USER-SPECIFIED RANGE OF FREQUENCIES. THE DESIRED OUTPUT OF AN AC SMALL-SIGNAL ANALYSIS IS USUALLY A TRANSFER FUNCTION (VOLTAGE GAIN, TRANSIMPEDANCE, ETC). IF THE CIRCUIT HAS ONLY ONE AC INPUT, IT IS CONVENIENT TO SET THAT INPUT TO UNITY AND ZERO PHASE, SO THAT OUTPUT VARIABLES HAVE THE SAME VALUE AS THE TRANSFER FUNCTION OF THE OUTPUT VARIABLE WITH RESPECT TO THE INPUT.

THE GENERATION OF WHITE NOISE BY RESISTORS AND SEMICONDUCTOR DEVICES CAN ALSO BE SIMULATED WITH THE AC SMALL-SIGNAL PORTION OF SPICE. EQUIVALENT NOISE SOURCE VALUES ARE DETERMINED AUTOMATICALLY FROM THE SMALL-SIGNAL OPERATING POINT OF THE CIRCUIT, AND THE CONTRIBUTION OF EACH NOISE SOURCE IS ADDED AT A GIVEN SUMMING POINT. THE TOTAL OUTPUT NOISE LEVEL AND THE EQUIVALENT INPUT NOISE LEVEL ARE DETERMINED AT EACH FREQUENCY POINT. THE OUTPUT AND INPUT NOISE LEVELS ARE NORMALIZED WITH RESPECT TO THE SQUARE ROOT OF THE NOISE BANDWIDTH AND HAVE THE UNITS VOLTS/RT HZ OR AMPS/RT HZ. THE OUTPUT NOISE AND EQUIVALENT INPUT NOISE CAN BE PRINTED OR PLOTTED IN THE SAME FASHION AS OTHER OUTPUT VARIABLES. NO ADDITIONAL INPUT DATA IS NECESSARY FOR THIS ANALYSIS.

FLICKER NOISE SOURCES CAN BE SIMULATED IN THE NOISE ANALYSIS BY INCLUDING VALUES FOR THE PARAMETERS KF AND AF ON THE APPROPRIATE DEVICE MODEL CARDS.

THE DISTORTION CHARACTERISTICS OF A CIRCUIT IN THE SMALL-SIGNAL MODE CAN BE SIMULATED AS A PART OF THE AC SMALL-SIGNAL ANALYSIS. THE ANALYSIS IS PERFORMED ASSUMING THAT ONE OR TWO SIGNAL FREQUENCIES ARE IMPOSED AT THE INPUT.

THE FREQUENCY RANGE AND THE NOISE AND DISTORTION ANALYSIS PARAMETERS ARE SPECIFIED ON THE .AC, .NOISE, AND .DISTORTION CONTROL CARDS.

CONVERGENCE

BOTH DC AND TRANSIENT SOLUTIONS ARE OBTAINED BY AN ITERATIVE PROCESS WHICH IS TERMINATED WHEN BOTH OF THE FOLLOWING CONDITIONS HOLD:

1) THE NONLINEAR BRANCH CURRENTS CONVERGE TO WITHIN A TOLERANCE OF 0.1 PERCENT OR 1 PICOAMP (1.0E-12 AMP), WHICHEVER IS LARGER

2) THE NODE VOLTAGES CONVERGE TO WITHIN A TOLERANCE OF 0.1 PERCENT OR 1 MICROVOLT (1.0E-6 VOLT), WHICHEVER IS LARGER

ALTHOUGH THE ALGORITHM USED IN SPICE HAS BEEN FOUND TO BE VERY RELIABLE, IN SOME CASES IT WILL FAIL TO CONVERGE TO A SOLUTION. WHEN THIS FAILURE OCCURS, THE PROGRAM WILL PRINT THE NODE VOLTAGES AT THE LAST ITERATION AND TERMINATE THE JOB. IN SUCH CASES, THE NODE VOLTAGES THAT ARE PRINTED ARE NOT NECESSARILY CORRECT OR EVEN CLOSE TO THE CORRECT SOLUTION.

FAILURE TO CONVERGE IN THE DC ANALYSIS IS USUALLY DUE TO AN ERROR IN SPECIFYING CIRCUIT CONNECTIONS, ELEMENT VALUES, OR MODEL PARAMETER VALUES. REGENERATIVE SWITCHING CIRCUITS OR CIRCUITS WITH POSITIVE FEEDBACK PROBABLY WILL NOT CONVERGE IN THE DC ANALYSIS UNLESS THE ±OFF± OPTION IS USED FOR SOME OF THE DEVICES IN THE FEEDBACK PATH.

INPUT FORMAT

THE INPUT FORMAT FOR SPICE IS OF THE FREE FORMAT TYPE. FIELDS ON A CARD ARE SEPARATED BY ONE OR MORE BLANKS, A COMMA, AN EQUAL (=) SIGN, OR A LEFT OR RIGHT PARENTHESIS; EXTRA SPACES ARE IGNORED. A CARD MAY BE CONTINUED BY PUNCHING A + (PLUS) IN COLUMN 1 OF THE FOLLOWING CARD; SPICE CONTINUES READING BEGINNING WITH COLUMN 2.

A NAME FIELD MUST BEGIN WITH A LETTER (A THROUGH Z) AND CANNOT CONTAIN ANY DELIMITERS. ONLY THE FIRST EIGHT CHARACTERS OF THE NAME ARE USED.

A NUMBER FIELD MAY BE AN INTEGER FIELD (12, -44), A FLOATING POINT FIELD (3.14159), EITHER AN INTEGER OR FLOATING POINT NUMBER FOLLOWED BY AN INTEGER EXPONENT (1E-14, 2.65E3), OR EITHER AN INTEGER OR A FLOATING POINT NUMBER FOLLOWED BY ONE OF THE FOLLOWING SCALE FACTORS:

G=1E9 MEG=1E6 K=1E3 MIL=25.4E-4 M=1E-3 U=1E-6 N=1E-9 P=1E-12

LETTERS IMMEDIATELY FOLLOWING A NUMBER THAT ARE NOT SCALE FACTORS ARE IGNORED, AND LETTERS IMMEDIATELY FOLLOWING A SCALE FACTOR ARE IGNORED. HENCE, 10, 10V, 10VOLTS, AND 10HZ ALL REPRESENT THE SAME NUMBER, AND M, MA, MSEC, AND MMHOS ALL REPRESENT THE SAME SCALE FACTOR. NOTE THAT 1000, 1000.0, 1000HZ, 1E3, 1.0E3, 1KHZ, AND 1K ALL REPRESENT THE SAME NUMBER.

CIRCUIT DESCRIPTION

THE CIRCUIT TO BE ANALYZED IS DESCRIBED TO SPICE BY A SET OF ELEMENT CARDS, WHICH DEFINE THE CIRCUIT TOPOLOGY AND ELEMENT VALUES, AND A SET OF CONTROL CARDS, WHICH DEFINE THE MODEL PARAMETERS AND THE RUN CONTROLS. THE FIRST CARD IN THE INPUT DECK MUST BE A TITLE CARD, AND THE LAST CARD MUST BE A .END CARD. THE ORDER OF THE REMAINING CARDS IS ARBITRARY (EXCEPT, OF COURSE, THAT CONTINUATION CARDS MUST IMMEDIATELY FOLLOW THE CARD BEING CONTINUED).

EACH ELEMENT IN THE CIRCUIT IS SPECIFIED BY AN ELEMENT CARD THAT CONTAINS THE ELEMENT NAME, THE CIRCUIT NODES TO WHICH THE ELEMENT IS CONNECTED, AND THE VALUES OF THE PARAMETERS THAT DETERMINE THE ELECTRICAL CHARACTERISTICS OF THE ELEMENT. THE FIRST LETTER OF THE ELEMENT NAME SPECIFIES THE ELEMENT TYPE. THE FORMAT FOR THE SPICE ELEMENT TYPES IS GIVEN IN WHAT FOLLOWS. THE STRINGS *XXXXXXX*, *YYYYYYY*, AND *ZZZZZZZ* DENOTE ARBITRARY ALPHANUMERIC STRINGS. FOR EXAMPLE, A RESISTOR NAME MUST BEGIN WITH THE LETTER R AND CAN CONTAIN FROM ONE TO EIGHT CHARACTERS. HENCE, R, R1, RSE, ROUT, AND R3AC2ZY ARE VALID RESISTOR NAMES.

DATA FIELDS THAT ARE ENCLOSED IN SQUARE BRACKETS ([ ]) ARE OPTIONAL. ALL INDICATED PUNCTUATION (PARENTHESES, EQUAL SIGNS, ETC.) ARE REQUIRED. WITH RESPECT TO BRANCH VOLTAGES AND CURRENTS, SPICE UNIFORMLY USES THE ASSOCIATED REFERENCE CONVENTION INDICATED IN THE DRAWING BELOW:

```
                          >
                          >>
                I    >>>>>>>>>>>
                          >>
                          >
            ....................
            .                  .
 ...........                    ...........
            .                  .
            ....................

        +           V           -
```

NODES MUST BE NONNEGATIVE INTEGERS BUT NEED NOT BE NUMBERED SEQUENTIALLY. THE DATUM (GROUND) NODE MUST BE NUMBERED ZERO. THE CIRCUIT CANNOT CONTAIN A LOOP OF VOLTAGE SOURCES AND/OR INDUCTORS AND CANNOT CONTAIN A CUTSET OF CURRENT SOURCES AND/OR CAPACITORS. EACH NODE IN THE CIRCUIT MUST HAVE A DC PATH TO GROUND. EVERY NODE MUST HAVE AT LEAST TWO CONNECTIONS EXCEPT FOR TRANSMISSION LINE NODES (TO PERMIT UNTERMINATED TRANSMISSION LINES).

ELEMENT CARDS

**** RESISTORS

GENERAL FORM RXXXXXXX N1 N2 VALUE [TC=TC1[,TC2]]

EXAMPLE RC1 12 17 1K TC=0.001,0.015

N1 AND N2 ARE THE TWO ELEMENT NODES. VALUE IS THE RESISTANCE (IN OHMS) AND MAY BE POSITIVE OR NEGATIVE BUT NOT ZERO. TC1 AND TC2 ARE THE (OPTIONAL) TEMPERATURE COEFFICIENTS; IF NOT SPECIFIED, ZERO IS ASSUMED FOR BOTH. THE VALUE OF THE RESISTOR AS A FUNCTION OF TEMPERATURE IS GIVEN BY:

```
VALUE(TEMP) = VALUE(TNOM)*(1+TC1*(TEMP-TNOM)+TC2*(TEMP-TNOM)*2))
```

**** CAPACITORS AND INDUCTORS

GENERAL FORM CXXXXXXX N+ N- P0 [P1 P2 ...] [IC=INCOND]
LYYYYYYY N+ N- P0 [P1 P2 ...] [IC=INCOND]

EXAMPLES CBYP 13 0 1UF
COSC 17 23 10U 5U IC=3V
LLINK 42 69 1UH
LSHUNT 23 51 10U 32U IC=15.7A

N+ AND N- ARE THE POSITIVE AND NEGATIVE ELEMENT NODES, RESPECTIVELY. P0, (P1, P2, ETC.) ARE THE COEFFICIENTS OF A POLYNOMIAL DESCRIBING THE ELEMENT VALUE. FOR CAPACITORS, THE CAPACITANCE (IN FARADS) IS EXPRESSED AS A FUNCTION OF THE VOLTAGE ACROSS THE ELEMENT. FOR INDUCTORS, THE INDUCTANCE (IN HENRIES) IS EXPRESSED AS A FUNCTION OF THE CURRENT THROUGH THE ELEMENT. TO ILLUSTRATE, THE SECOND EXAMPLE ABOVE DESCRIBES A CAPACITOR WITH A VALUE DEFINED BY

```
C = 10E-6 + 5E-6*V
```

WHERE V IS THE VOLTAGE ACROSS THE CAPACITOR.

FOR THE CAPACITOR, THE (OPTIONAL) INITIAL CONDITION IS THE INITIAL (TIME-ZERO) VALUE OF CAPACITOR VOLTAGE (IN VOLTS). FOR THE INDUCTOR, THE (OPTIONAL) INITIAL CONDITION IS THE INITIAL (TIME-ZERO) VALUE OF INDUCTOR CURRENT (IN AMPS) THAT FLOWS FROM N+, THROUGH THE INDUCTOR, TO N-. NOTE THAT THE INITIAL CONDITIONS (IF ANY) APPLY *ONLY* IF THE UIC OPTION IS SPECIFIED ON THE .TRAN CARD.

**** COUPLED (MUTUAL) INDUCTORS

GENERAL FORM KXXXXXXX LYYYYYYY LZZZZZZZ VALUE

EXAMPLES K43 LAA LBB 0.999
KXFRMR L1 L2 0.87

LYYYYYYY AND LZZZZZZZ ARE THE NAMES OF THE TWO COUPLED INDUCTORS, AND VALUE IS THE COEFFICIENT OF COUPLING, K, WHICH MUST BE GREATER THAN 0 AND LESS THAN OR EQUAL TO 1. THE COUPLED INDUCTORS CAN BE ENTERED IN EITHER ORDER AND MUST BE CONSTANT (ONLY PO CAN BE SPECIFIED ON THE INDUCTOR CARDS). IN TERMS OF THE DOT CONVENTION, THIS COUPLING PUTS A DOT BY THE FIRST NODE OF EACH OF THE COUPLED INDUCTORS.

**** TRANSMISSION LINES (LOSSLESS)

GENERAL FORM TXXXXXXX N1 N2 N3 N4 Z0=VALUE [TD=VALUE] [F=FREQ [NL=NRMLEN]]
+ [IC=V1,I1,V2,I2]

EXAMPLE T1 1 0 2 0 Z0=50 TD=10NS

N1 AND N2 ARE THE NODES AT PORT 1; N3 AND N4 ARE THE NODES AT PORT 2. Z0 IS THE CHARACTERISTIC IMPEDANCE. THE LENGTH OF THE LINE MAY BE EXPRESSED IN EITHER OF TWO FORMS. THE TRANSMISSION DELAY, TD, MAY BE SPECIFIED DIRECTLY (AS TD=10NS, FOR EXAMPLE). ALTERNATIVELY, A FREQUENCY F MAY BE GIVEN, TOGETHER WITH NL, THE NORMALIZED ELECTRICAL LENGTH OF THE TRANSMISSION LINE WITH RESPECT TO THE WAVELENGTH IN THE LINE AT THE FREQUENCY F. IF A FREQUENCY IS SPECIFIED BUT NL IS OMITTED, 0.25 IS ASSUMED (THAT IS, THE FREQUENCY IS ASSUMED TO BE THE QUARTER-WAVE FREQUENCY). NOTE THAT ALTHOUGH BOTH FORMS FOR EXPRESSING THE LINE LENGTH ARE INDICATED AS OPTIONAL, ONE OF THE TWO *MUST* BE SPECIFIED.

NOTE THAT THIS ELEMENT MODELS ONLY ONE PROPAGATING MODE. IF ALL FOUR NODES ARE DISTINCT IN THE ACTUAL CIRCUIT, THEN TWO MODES MAY BE EXCITED. TO SIMULATE SUCH A SITUATION, TWO TRANSMISSION-LINE ELEMENTS ARE REQUIRED. (SEE THE EXAMPLE IN APPENDIX A FOR FURTHER CLARIFICATION.)

THE (OPTIONAL) INITIAL CONDITION SPECIFICATION CONSISTS OF THE VOLTAGE AND CURRENT AT EACH OF THE TRANSMISSION LINE PORTS. NOTE THAT THE INITIAL CONDITIONS (IF ANY) APPLY *ONLY* IF THE UIC OPTION IS SPECIFIED ON THE .TRAN CARD.

**** LINEAR DEPENDENT SOURCES

SPICE ALLOWS CIRCUITS TO CONTAIN LINEAR DEPENDENT SOURCES CHARACTERIZED BY ANY OF THE FOUR EQUATIONS

I=G*V V=E*V I=F*I V=H*I

WHERE G, E, F, AND H ARE CONSTANTS REPRESENTING TRANSCONDUCTANCE, VOLTAGE GAIN, CURRENT GAIN, AND TRANSRESISTANCE, RESPECTIVELY. NOTE: A MORE COMPLETE DESCRIPTION OF DEPENDENT SOURCES AS IMPLEMENTED IN SPICE IS GIVEN IN APPENDIX B.

**** LINEAR VOLTAGE-CONTROLLED CURRENT SOURCES

GENERAL FORM GXXXXXXX N+ N- NC+ NC- VALUE

EXAMPLE G1 2 0 5 0 0.1MMHO

N+ AND N- ARE THE POSITIVE AND NEGATIVE NODES, RESPECTIVELY. CURRENT FLOW IS FROM THE POSITIVE NODE, THROUGH THE SOURCE, TO THE NEGATIVE NODE. NC+ AND NC- ARE THE POSITIVE AND NEGATIVE CONTROLLING NODES, RESPECTIVELY. VALUE IS THE TRANSCONDUCTANCE (IN MHOS).

**** LINEAR VOLTAGE-CONTROLLED VOLTAGE SOURCES

GENERAL FORM EXXXXXXX N+ N- NC+ NC- VALUE

EXAMPLE E1 2 3 14 1 2.0

N+ IS THE POSITIVE NODE, AND N- IS THE NEGATIVE NODE. NC+ AND NC- ARE THE POSITIVE AND NEGATIVE CONTROLLING NODES, RESPECTIVELY. VALUE IS THE VOLTAGE GAIN.

**** LINEAR CURRENT-CONTROLLED CURRENT SOURCES

GENERAL FORM FXXXXXXX N+ N- VNAM VALUE

EXAMPLE F1 13 5 VSENS 5

N+ AND N- ARE THE POSITIVE AND NEGATIVE NODES, RESPECTIVELY. CURRENT FLOW IS FROM THE POSITIVE NODE, THROUGH THE SOURCE, TO THE NEGATIVE NODE. VNAM IS THE NAME OF A VOLTAGE SOURCE THROUGH WHICH THE CONTROLLING CURRENT FLOWS. THE DIRECTION OF POSITIVE CONTROLLING CURRENT FLOW IS FROM THE POSITIVE NODE, THROUGH THE SOURCE, TO THE NEGATIVE NODE OF VNAM. VALUE IS THE CURRENT GAIN.

**** LINEAR CURRENT-CONTROLLED VOLTAGE SOURCES

GENERAL FORM HXXXXXXX N+ N- VNAM VALUE

EXAMPLE HX 5 17 VZ 0.5K

N+ AND N- ARE THE POSITIVE AND NEGATIVE NODES, RESPECTIVELY. VNAM IS THE NAME OF A VOLTAGE SOURCE THROUGH WHICH THE CONTROLLING CURRENT FLOWS. THE DIRECTION OF POSITIVE CONTROLLING CURRENT FLOW IS FROM THE POSITIVE NODE, THROUGH THE SOURCE, TO THE NEGATIVE NODE OF VNAM. VALUE IS THE TRANSRESISTANCE (IN OHMS).

**** INDEPENDENT SOURCES

GENERAL FORM

```
VXXXXXXX N+ N- [[DC] DC/TRAN VALUE] [AC [ACMAG [ACPHASE]]]
IYYYYYYY N+ N- [[DC] DC/TRAN VALUE] [AC [ACMAG [ACPHASE]]]
```

EXAMPLES

```
VCC 10 0 DC 6
VIN 13 2 0.001 AC 1 SIN(0 1 1MEG)
ISRC 23 21 AC 0.333 45.0 SFFM(0 1 10K 5 1K)
VMEAS 12 9
```

N+ AND N- ARE THE POSITIVE AND NEGATIVE NODES, RESPECTIVELY. NOTE THAT VOLTAGE SOURCES NEED NOT BE GROUNDED. CURRENT IS ASSUMED TO FLOW FROM THE POSITIVE NODE, THROUGH THE SOURCE, TO THE NEGATIVE NODE.

DC/TRAN IS THE DC AND TRANSIENT ANALYSIS VALUE OF THE SOURCE. IF THE SOURCE VALUE IS ZERO BOTH FOR DC AND TRANSIENT ANALYSES, THIS VALUE MAY BE OMITTED. IF THE SOURCE VALUE IS TIME-INVARIANT (E.G., A POWER SUPPLY), THEN THE VALUE MAY OPTIONALLY BE PRECEDED BY THE LETTERS DC.

ACMAG IS THE AC MAGNITUDE AND ACPHASE IS THE AC PHASE. THE SOURCE IS SET TO THIS VALUE IN THE AC ANALYSIS. IF ACMAG IS OMITTED FOLLOWING THE KEYWORD AC, A VALUE OF UNITY IS ASSUMED. IF ACPHASE IS OMITTED, A VALUE OF ZERO IS ASSUMED. IF THE SOURCE IS NOT AN AC SMALL-SIGNAL INPUT, THE KEYWORD AC AND THE AC VALUES ARE OMITTED.

ANY INDEPENDENT SOURCE CAN BE ASSIGNED A TIME-DEPENDENT VALUE FOR TRANSIENT ANALYSIS. IF A SOURCE IS ASSIGNED A TIME-DEPENDENT VALUE, THE TIME-ZERO VALUE IS USED FOR DC ANALYSIS. THERE ARE FIVE INDEPENDENT SOURCE FUNCTIONS: PULSE, EXPONENTIAL, SINUSOIDAL, PIECE-WISE LINEAR, AND SINGLE-FREQUENCY FM. IF PARAMETERS OTHER THAN SOURCE VALUES ARE OMITTED OR SET TO ZERO, THE DEFAULT VALUES SHOWN WILL BE ASSUMED. (TSTEP IS THE PRINTING INCREMENT AND TSTOP IS THE FINAL TIME (SEE THE .TRAN CARD FOR EXPLANATION)).

1. PULSE

```
PULSE(V1 V2 TD TR TF PW PER)
```

EXAMPLE

```
VIN 3 0 PULSE(-1 1 2NS 2NS 2NS 50NS 100NS)
```

PARAMETERS AND DEFAULT VALUES			UNITS
V1	INITIAL VALUE	---	VOLTS OR AMPS
V2	PULSED VALUE	---	VOLTS OR AMPS
TD	DELAY TIME	0.0	SECONDS
TR	RISE TIME	TSTEP	SECONDS
TF	FALL TIME	TSTEP	SECONDS
PW	PULSE WIDTH	TSTOP	SECONDS
PER	PERIOD	TSTOP	SECONDS

A SINGLE PULSE SO SPECIFIED IS DESCRIBED BY THE FOLLOWING TABLE:

TIME	VALUE
0	V1
TD	V1
TD+TR	V2
TD+TR+PW	V2
TD+TR+PW+TF	V1
TSTOP	V1

INTERMEDIATE POINTS ARE DETERMINED BY LINEAR INTERPOLATION.

2. SINUSOIDAL SIN(VO VA FREQ TD THETA)

EXAMPLE VIN 3 0 SIN(0 1 100MEG 1NS 1E10)

PARAMETERS AND DEFAULT VALUES			UNITS
VO	OFFSET	---	VOLTS OR AMPS
VA	AMPLITUDE	---	VOLTS OR AMPS
FREQ	FREQUENCY	1/TSTOP	HZ
TD	DELAY	0.0	SECONDS
THETA	DAMPING FACTOR	0.0	1/SECONDS

THE SHAPE OF THE WAVEFORM IS DESCRIBED BY THE FOLLOWING TABLE:

TIME	VALUE
0 TO TD	VO
TD TO TSTOP	VO + VA*EXP(-(TIME-TD)*THETA)*SINE(TWOPI*FREQ*(TIME-TD))

3. EXPONENTIAL EXP(V1 V2 TD1 TAU1 TD2 TAU2)

EXAMPLE VIN 3 0 EXP(-4 -1 2NS 30NS 60NS 40NS)

PARAMETERS AND DEFAULT VALUES			UNITS
V1	INITIAL VALUE	---	VOLTS OR AMPS
V2	PULSED VALUE	---	VOLTS OR AMPS
TD1	RISE DELAY TIME	0.0	SECONDS
TAU1	RISE TIME CONSTANT	TSTEP	SECONDS
TD2	FALL DELAY TIME	TD1+TSTEP	SECONDS
TAU2	FALL TIME CONSTANT	TSTEP	SECONDS

THE SHAPE OF THE WAVEFORM IS DESCRIBED BY THE FOLLOWING TABLE:

TIME	VALUE
0 TO TD1	V1
TD1 TO TD2	V1+(V2-V1)*(1-EXP(-(TIME-TD1)/TAU1))
TD2 TO TSTOP	V1+(V2-V1)*(1-EXP(-(TIME-TD1)/TAU1)) +(V1-V2)*(1-EXP(-(TIME-TD2)/TAU2))

4. PIECE-WISE LINEAR

PWL(T1 V1 [T2 V2 T3 V3 T4 V4 ...])

EXAMPLE VCLOCK 7 5 PWL(0 -7 10NS -7 11NS -3 17NS -3 18NS -7 50NS -7)

PARAMETERS AND DEFAULT VALUES

EACH PAIR OF VALUES (TI, VI) SPECIFIES THAT THE VALUE OF THE SOURCE IS VI (IN VOLTS OR AMPS) AT TIME=TI. THE VALUE OF THE SOURCE AT INTERMEDIATE VALUES OF TIME IS DETERMINED BY USING LINEAR INTERPOLATION ON THE INPUT VALUES.

5. SINGLE-FREQUENCY FM

SFFM(VO VA FC MDI FS)

EXAMPLE V1 12 0 SFFM(0 1M 20K 5 1K)

PARAMETERS AND DEFAULT VALUES			UNITS
VO	OFFSET	---	VOLTS OR AMPS
VA	AMPLITUDE	---	VOLTS OR AMPS
FC	CARRIER FREQUENCY	1/TSTOP	HZ
MDI	MODULATION INDEX	---	---
FS	SIGNAL FREQUENCY	1/TSTOP	HZ

THE SHAPE OF THE WAVEFORM IS DESCRIBED BY THE FOLLOWING EQUATION:

VALUE = VO + VA*SINE((TWOPI*FC*TIME) + MDI*SINE(TWOPI*FS*TIME))

**** SEMICONDUCTOR DEVICES

THE ELEMENTS THAT HAVE BEEN DESCRIBED TO THIS POINT TYPICALLY REQUIRE ONLY A FEW PARAMETER VALUES TO SPECIFY COMPLETELY THE ELECTRICAL CHARACTERISTICS OF THE ELEMENT. HOWEVER, THE MODELS FOR THE FOUR SEMICONDUCTOR DEVICES THAT ARE INCLUDED IN THE SPICE PROGRAM REQUIRE MANY PARAMETER VALUES. MOREOVER, MANY DEVICES IN A CIRCUIT OFTEN ARE DEFINED BY THE SAME SET OF DEVICE MODEL PARAMETERS. FOR THESE REASONS, A SET OF DEVICE MODEL PARAMETERS IS DEFINED ON A SEPARATE .MODEL CARD AND ASSIGNED A UNIQUE MODEL NAME. THE DEVICE ELEMENT CARDS IN SPICE THEN REFERENCE THE MODEL NAME. THIS SCHEME ALLEVIATES THE NEED TO SPECIFY ALL OF THE MODEL PARAMETERS ON EACH DEVICE ELEMENT CARD.

EACH DEVICE ELEMENT CARD CONTAINS THE DEVICE NAME, THE NODES TO WHICH THE DEVICE IS CONNECTED, AND THE DEVICE MODEL NAME. IN ADDITION, TWO OPTIONAL PARAMETERS MAY BE SPECIFIED FOR EACH DEVICE: AN AREA FACTOR, AND AN INITIAL CONDITION.

THE AREA FACTOR DETERMINES THE NUMBER OF EQUIVALENT PARALLEL DEVICES OF A SPECIFIED MODEL. THE AFFECTED PARAMETERS ARE MARKED WITH AN ASTERISK UNDER THE HEADING *AREA* IN THE MODEL DESCRIPTIONS BELOW.

TWO DIFFERENT FORMS OF INITIAL CONDITIONS MAY BE SPECIFIED FOR DEVICES. THE FIRST FORM IS INCLUDED TO IMPROVE THE DC CONVERGENCE FOR CIRCUITS THAT CONTAIN MORE THAN ONE STABLE STATE. IF A DEVICE IS SPECIFIED OFF, THE DC OPERATING POINT IS DETERMINED WITH THE TERMINAL VOLTAGES FOR THAT DEVICE SET TO ZERO. AFTER CONVERGENCE IS OBTAINED, THE PROGRAM CONTINUES TO ITERATE TO OBTAIN THE EXACT VALUE FOR THE TERMINAL VOLTAGES. IF A CIRCUIT HAS MORE THAN ONE DC STABLE STATE, THE OFF OPTION CAN BE USED TO FORCE THE SOLUTION TO CORRESPOND TO A DESIRED STATE. IF A DEVICE IS SPECIFIED OFF WHEN IN REALITY THE DEVICE IS CONDUCTING, THE PROGRAM WILL STILL OBTAIN THE CORRECT SOLUTION (ASSUMING THE SOLUTIONS CONVERGE) BUT MORE ITERATIONS WILL BE REQUIRED SINCE THE PROGRAM MUST INDEPENDENTLY CONVERGE TO TWO SEPARATE SOLUTIONS.

THE SECOND FORM OF INITIAL CONDITION SPECIFICATION (USING IC=...) IS PROVIDED TO ALLOW THE USER TO BYPASS THE DC OPERATING POINT CALCULATION NORMALLY MADE BEFORE THE START OF TRANSIENT ANALYSIS. THESE INITIAL CONDITIONS ARE USED BY SPICE *ONLY* IF THE UIC OPTION IS GIVEN ON THE .TRAN CARD.

**** JUNCTION DIODES

GENERAL FORM DXXXXXXX N+ N- MNAME [AREA] [OFF] [IC=VD]

EXAMPLE DBRIDGE 2 10 DIODE1
DCLMP 3 7 DMOD 3.0 IC=0.2

N+ AND N- ARE THE POSITIVE AND NEGATIVE NODES, RESPECTIVELY. MNAME IS THE MODEL NAME, AREA IS THE AREA FACTOR, AND OFF INDICATES AN (OPTIONAL) INITIAL CONDITION ON THE DEVICE FOR DC ANALYSIS. IF THE AREA FACTOR IS OMITTED, A VALUE OF 1.0 IS ASSUMED. THE (OPTIONAL) INITIAL CONDITION SPECIFICATION USING IC=VD IS INTENDED FOR USE WITH THE UIC OPTION ON THE .TRAN CARD, WHEN A TRANSIENT ANALYSIS IS DESIRED STARTING FROM OTHER THAN THE QUIESCENT OPERATING POINT.

**** BIPOLAR JUNCTION TRANSISTORS

GENERAL FORM QXXXXXXX NC NB NE MNAME [AREA] [OFF] [IC=VBE,VCE]

EXAMPLE Q23 10 24 13 QMOD IC=0.6,5.0

NC, NB, AND NE ARE THE COLLECTOR, BASE, AND EMITTER NODES, RESPECTIVELY. MNAME IS THE MODEL NAME, AREA IS THE AREA FACTOR, AND OFF INDICATES AN (OPTIONAL) INITIAL CONDITION ON THE DEVICE FOR THE DC ANALYSIS. IF THE AREA FACTOR IS OMITTED, A VALUE OF 1.0 IS ASSUMED. THE (OPTIONAL) INITIAL CONDITION SPECIFICATION USING IC=VBE,VCE IS INTENDED FOR USE WITH THE UIC OPTION ON THE .TRAN CARD, WHEN A TRANSIENT ANALYSIS IS DESIRED STARTING FROM OTHER THAN THE QUIESCENT OPERATING POINT.

**** JUNCTION FIELD-EFFECT TRANSISTORS

GENERAL FORM JXXXXXXX ND NG NS MNAME [AREA] [OFF] [IC=VDS,VGS]

EXAMPLE J1 7 2 3 JM1 OFF

ND, NG, AND NS ARE THE DRAIN, GATE, AND SOURCE NODES, RESPECTIVELY. MNAME IS THE MODEL NAME, AREA IS THE AREA FACTOR, AND OFF INDICATES AN (OPTIONAL) INITIAL CONDITION ON THE DEVICE FOR DC ANALYSIS. IF THE AREA FACTOR IS OMITTED, A VALUE OF 1.0 IS ASSUMED. THE (OPTIONAL) INITIAL CONDITION SPECIFICATION USING IC=VDS,VGS IS INTENDED FOR USE WITH THE UIC OPTION ON THE .TRAN CARD, WHEN A TRANSIENT ANALYSIS IS DESIRED STARTING FROM OTHER THAN THE QUIESCENT OPERATING POINT.

**** MOSFETS

GENERAL FORM MXXXXXXX ND NG NS NB MNAME [W=VAL] [L=VAL] [AD=VAL] [AS=VAL]
+ [OFF] [IC=VDS,VGS,VBS]

EXAMPLE M31 2 17 6 10 MODM L=2MIL W=0.5MIL

ND, NG, NS, AND NB ARE THE DRAIN, GATE, SOURCE, AND BULK (SUBSTRATE) NODES, RESPECTIVELY. MNAME IS THE MODEL NAME. W AND L ARE THE CHANNEL WIDTH AND LENGTH, IN CENTIMETERS; IF OMITTED, BOTH W AND L ARE ASSUMED TO BE 1.0. AD AND AS ARE THE AREAS OF THE DRAIN AND SOURCE DIFFUSIONS, IN CM↑2; IF NOT SPECIFIED, BOTH AREAS ARE ASSUMED TO BE 1.0E-6. OFF INDICATES AN (OPTIONAL) INITIAL CONDITION ON THE DEVICE FOR DC ANALYSIS. THE (OPTIONAL) INITIAL CONDITION SPECIFICATION USING IC=VDS,VGS,VBS IS INTENDED FOR USE WITH THE UIC OPTION ON THE .TRAN CARD, WHEN A TRANSIENT ANALYSIS IS DESIRED STARTING FROM OTHER THAN THE QUIESCENT OPERATING POINT.

**** .MODEL CARD

GENERAL FORM .MODEL MNAME TYPE(PNAME1=PVAL1 PNAME2=PVAL2 ...)

EXAMPLE .MODEL MOD1 NPN(BF=50 IS=1E-13 VA=50)

THE .MODEL CARD SPECIFIES A SET OF MODEL PARAMETERS THAT WILL BE USED BY ONE OR MORE DEVICES. MNAME IS THE MODEL NAME, AND TYPE IS ONE OF THE FOLLOWING SEVEN TYPES:

NPN	NPN BJT MODEL
PNP	PNP BJT MODEL
D	DIODE MODEL
NJF	N-CHANNEL JFET MODEL
PJF	P-CHANNEL JFET MODEL
NMOS	N-CHANNEL MOSFET MODEL
PMOS	P-CHANNEL MOSFET MODEL

PARAMETER VALUES ARE DEFINED BY APPENDING THE PARAMETER NAME, AS GIVEN BELOW FOR EACH MODEL TYPE, FOLLOWED BY AN EQUAL SIGN AND THE PARAMETER VALUE. MODEL PARAMETERS THAT ARE NOT GIVEN A VALUE ARE ASSIGNED THE DEFAULT VALUES GIVEN BELOW FOR EACH MODEL TYPE.

---- DIODE MODEL

THE DC CHARACTERISTICS OF THE DIODE ARE DETERMINED BY THE PARAMETERS IS AND N. AN OHMIC RESISTANCE, RS, IS INCLUDED. CHARGE STORAGE EFFECTS ARE MODELED BY A TRANSIT TIME, TT, AND A NONLINEAR DEPLETION LAYER CAPACITANCE WHICH IS DETERMINED BY THE PARAMETERS CJO, PB, AND M. THE TEMPERATURE DEPENDENCE OF THE SATURATION CURRENT IS DEFINED BY THE PARAMETERS EG, THE ENERGY GAP, AND PT, THE SATURATION CURRENT TEMPERATURE EXPONENT. REVERSE BREAKDOWN IS MODELED BY AN EXPONENTIAL INCREASE IN THE REVERSE DIODE CURRENT AND IS DETERMINED BY THE PARAMETERS BV AND IBV (BOTH OF WHICH ARE POSITIVE NUMBERS).

	AREA	NAME	PARAMETER	DEFAULT	TYPICAL
1	*	IS	SATURATION CURRENT	1.0E-14	1.0E-14
2	*	RS	OHMIC RESISTANCE	0	10
3		N	EMISSION COEFFICIENT	1	1.0
4		TT	TRANSIT TIME	0	0.1NS
5	*	CJO	ZERO-BIAS JUNCTION CAPACITANCE	0	2PF
6		PB	JUNCTION POTENTIAL	1	0.6
7		M	GRADING COEFFICIENT	0.5	0.5
8		EG	ENERGY GAP	1.11	1.11 SI
					0.69 SBD
					0.67 GE
9		PT	SATURATION CURRENT TEMP. EXPONENT	3.0	3.0 JN
					2.0 SBD
10		KF	FLICKER NOISE COEFFICIENT	0	
11		AF	FLICKER NOISE EXPONENT	1	
12		FC	FORWARD-BIAS NONIDEAL JUNCTION CAPACITANCE COEFFICIENT	0.5	
13		BV	REVERSE BREAKDOWN VOLTAGE	100.0	40.0
14		IBV	CURRENT AT BREAKDOWN VOLTAGE	1.0E-3	

---- BJT MODELS (BOTH NPN AND PNP)

THE BIPOLAR JUNCTION TRANSISTOR MODEL IN SPICE IS AN ADAPTATION OF THE INTEGRAL CHARGE CONTROL MODEL OF GUMMEL AND POON; HOWEVER, IF THE ADDITIONAL GUMMEL-POON PARAMETERS ARE NOT SPECIFIED, THE SIMPLER EBERS-MOLL MODEL OF SPICE, VERSION 1, IS USED. THE DC MODEL IS DEFINED BY THE PARAMETERS BF, C2, IK, AND NE, WHICH DETERMINE THE FORWARD CURRENT GAIN CHARACTERISTICS, BR, C4, IKR, AND NC, WHICH DETERMINE THE REVERSE CURRENT GAIN CHARACTERISTICS, VA AND VB, WHICH DETERMINE THE OUTPUT CONDUCTANCE FOR FORWARD AND REVERSE REGIONS, AND THE SATURATION CURRENT, IS. THREE OHMIC RESISTANCES, RB, RC, AND RE, ARE INCLUDED. BASE CHARGE STORAGE IS MODELED BY FORWARD AND REVERSE TRANSIT TIMES, TF AND TR, AND NONLINEAR DEPLETION LAYER CAPACITANCES WHICH ARE DETERMINED BY CJE, PE, AND ME FOR THE B-E JUNCTION, AND CJC, PC, AND MC FOR THE B-C JUNCTION. A CONSTANT COLLECTOR-SUBSTRATE CAPACITANCE, CCS, IS ALSO INCLUDED. THE TEMPERATURE DEPENDENCE OF THE SATURATION CURRENT IS DETERMINED BY THE ENERGY GAP, EG, AND THE SATURATION CURRENT TEMPERATURE EXPONENT, PT.

	AREA	NAME	PARAMETER	DEFAULT	TYPICAL	
1		BF	IDEAL FORWARD CURRENT GAIN	100	100	
2		BR	IDEAL REVERSE CURRENT GAIN	1	0.1	
3	*	IS	SATURATION CURRENT	1.0E-14	1.0E-16	
4	*	RB	BASE OHMIC RESISTANCE	0	100	
5	*	RC	COLLECTOR OHMIC RESISTANCE	0	10	
6	*	RE	EMITTER OHMIC RESISTANCE	0	1	
7		VA	FORWARD EARLY VOLTAGE	INFINITE	200	
8		VB	REVERSE EARLY VOLTAGE	INFINITE	200	
9	*	IK	FORWARD HIGH-CURRENT KNEE CURRENT	INFINITE	10MA	
10		C2	FORWARD LOW-CURRENT NONIDEAL BASE CURRENT COEFFICIENT	0	1000	
11		NE	NONIDEAL LOW-CURRENT BASE-EMITTER EMISSION COEFFICIENT	2.0	2.0	
12	*	IKR	REVERSE HIGH-CURRENT KNEE CURRENT	INFINITE	100MA	
13		C4	REVERSE LOW-CURRENT NONIDEAL BASE CURRENT COEFFICIENT	0	1.0	
14		NC	NONIDEAL LOW-CURRENT BASE-COLLECTOR EMISSION COEFFICIENT	2.0	2.0	
15		TF	FORWARD TRANSIT TIME	0	0.1NS	
16		TR	REVERSE TRANSIT TIME	0	10NS	
17	*	CCS	COLLECTOR-SUBSTRATE CAPACITANCE	0	2PF	
18	*	CJE	ZERO-BIAS B-E JUNCTION CAPACITANCE	0	2PF	
19		PE	B-E JUNCTION POTENTIAL	1.0	0.7	
20		ME	B-E JUNCTION GRADING COEFFICIENT	0.5	0.33	
21	*	CJC	ZERO BIAS B-C JUNCTION CAPACITANCE	0	1PF	
22		PC	B-C JUNCTION POTENTIAL	1.0	0.5	
23		MC	B-C GRADING COEFFICIENT	0.5	0.33	
24		EG	ENERGY GAP	1.11	1.11	SI
					0.67	GE
25		PT	SATURATION CURRENT TEMP. EXPONENT	3.0		
26		KF	FLICKER NOISE COEFFICIENT	0	6.6E-16	NPN
					5.3E-13	PNP
27		AF	FLICKER NOISE EXPONENT	1	1.0	NPN
					1.5	PNP
28		FC	FORWARD-BIAS NONIDEAL JUNCTION CAPACITANCE COEFFICIENT	0.5		

---- JFET MODELS (BOTH N AND P CHANNEL)

THE JFET MODEL IS DERIVED FROM THE FET MODEL OF SHICHMAN AND HODGES. THE DC CHARACTERISTICS ARE DEFINED BY THE PARAMETERS VTO AND BETA, WHICH DETERMINE THE VARIATION OF DRAIN CURRENT WITH GATE VOLTAGE, LAMBDA, WHICH DETERMINES THE OUTPUT CONDUCTANCE, AND IS, THE SATURATION CURRENT OF THE TWO GATE JUNCTIONS. TWO OHMIC RESISTANCES, RD AND RS, ARE INCLUDED. CHARGE STORAGE IS MODELED BY NONLINEAR DEPLETION LAYER CAPACITANCES FOR BOTH GATE JUNCTIONS WHICH VARY AS THE -1/2 POWER OF JUNCTION VOLTAGE AND ARE DEFINED BY THE PARAMETERS CGS, CGD, AND PB.

	AREA	NAME	PARAMETER	DEFAULT	TYPICAL
1		VTO	THRESHOLD VOLTAGE	-2.0	-2.0
2	*	BETA	TRANSCONDUCTANCE PARAMETER	1.0E-4	1.0E-3
3		LAMBDA	CHANNEL LENGTH MODULATION PARAMETER	0	1.0E-4
4	*	RD	DRAIN OHMIC RESISTANCE	0	100
5	*	RS	SOURCE OHMIC RESISTANCE	0	100
6	*	CGS	ZERO-BIAS G-S JUNCTION CAPACITANCE	0	5PF
7	*	CGD	ZERO-BIAS G-D JUNCTION CAPACITANCE	0	1PF
8		PB	GATE JUNCTION POTENTIAL	1	0.6
9	*	IS	GATE JUNCTION SATURATION CURRENT	1.0E-14	1.0E-14
10		KF	FLICKER NOISE COEFFICIENT	0	
11		AF	FLICKER NOISE EXPONENT	1	
12		FC	FORWARD-BIAS NONIDEAL JUNCTION CAPACITANCE COEFFICIENT	0.5	

---- MOSFET MODELS (BOTH N AND P CHANNEL)

THE MOSFET MODEL IS DERIVED FROM THE FRCHMAN-GROVE MODEL. THE DC CHARACTERISTICS OF THE MOSFET ARE DEFINED BY THE PARAMETERS VTO, BETA, LAMBDA, PHI, AND GAMMA. VTO IS POSITIVE (NEGATIVE) FOR ENHANCEMENT MODE AND NEGATIVE (POSITIVE) FOR DEPLETION MODE N-CHANNEL (P-CHANNEL) DEVICES. CHARGE STORAGE IS MODELED BY THREE CONSTANT CAPACITORS, CGS, CGD, AND CGB, BY THE NONLINEAR OXIDE CAPACITANCE WHICH IS DISTRIBUTED AMONG THE GATE-SOURCE, GATE-DRAIN, AND GATE-BULK REGIONS USING THE FORMULATION OF J. E. MEYER, AND BY THE NONLINEAR DEPLETION-LAYER CAPACITANCES FOR BOTH SUBSTRATE JUNCTIONS WHICH VARY AS THE -1/2 POWER OF JUNCTION VOLTAGE AND ARE DETERMINED BY THE PARAMETERS CBD, CBS, AND PB.

	NAME	PARAMETER	DEFAULT	TYPICAL	UNITS
1	VTO	ZERO-BIAS THRESHOLD VOLTAGE	0.0	-0.1	V
2	KP	INTRINSIC TRANSCONDUCTANCE PARAMETER	1.0E-5	3.1E-5	A/V*2
3	GAMMA	BULK THRESHOLD PARAMETER	0.0	0.37	V*(1/2)
4	PHI	SURFACE POTENTIAL AT STRONG INVERSION	0.6	0.65	V
5	LAMBDA	CHANNEL-LENGTH MODULATION PARAMETER	0.0	0.02	/V
6	RD	DRAIN OHMIC RESISTANCE	0.0	1.0	OHMS
7	RS	SOURCE OHMIC RESISTANCE	0.0	1.0	OHMS
8	CGS	GATE-SOURCE OVERLAP CAPACITANCE PER CM CHANNEL WIDTH	0.0	4.0E-13	F/CM
9	CGD	GATE-DRAIN OVERLAP CAPACITANCE PER CM CHANNEL WIDTH	0.0	4.0E-13	F/CM
10	CGB	GATE-BULK OVERLAP CAPACITANCE PER CM CHANNEL LENGTH	0.0	2.0E-12	F/CM
11	CBD	ZERO-BIAS B-D JUNCTION CAPACITANCE PER CM*2 OF JUNCTION AREA	0.0	2.0E-8	F/CM*2
12	CBS	ZERO-BIAS B-S JUNCTION CAPACITANCE PER CM*2 OF JUNCTION AREA	0.0	2.0E-8	F/CM*2
13	TOX	OXIDE THICKNESS	INFINITY	1.0E-5	CM
		IF NSUB SPECIFIED:	1.0E-5		
14	PB	BULK JUNCTION POTENTIAL	0.8	0.87	V
15	JS	BULK JUNCTION REVERSE SATURATION CURRENT PER CM*2 OF JUNCTION AREA	1.0E-8	1.0E-8	A/CM*2
16	NSUB	EFFECTIVE SUBSTRATE DOPING	0.0	4.0E15	/CM*3
17	NSS	EFFECTIVE SURFACE STATE DENSITY	0.0	1.0E10	/CM*2
18	NFS	EFFECTIVE FAST SURFACE STATE DENSITY	0.0	1.0E10	/CM*2
19	XJ	METALURGICAL JUNCTION DEPTH	0.0	1.0E-4	CM
20	LD	LATERAL DIFFUSION COEFFICIENT	0.8	0.8	
21	NGATE	POLYSILICON GATE DOPING	AL GATE	1.0E20	/CM*3
22	TPS	TYPE OF POLYSILICON: +1 (-1) FOR OPPOSITE (SAME) AS SUBSTRATE	+1.0		
23	UO	SURFACE MOBILITY	700	600	CM*2/V-S
24	UCRIT	CRITICAL FIELD FOR MOBILITY DEGRADATION	1.0E+4	1.0E+4	V/CM
25	UEXP	CRITICAL FIELD EXPONENT (MOBILITY)	0.0	0.1	
26	UTRA	TRANSVERSE FIELD COEFFICIENT (MOBILITY)	0.0	0.3	
27	KF	FLICKER NOISE COEFFICIENT	0.0		
28	AF	FLICKER NOISE EXPONENT	1.0		
29	FC	FORWARD-BIAS NONIDEAL JUNCTION CAPACITANCE COEFFICIENT	0.5		

**** SUBCIRCUITS

A SUBCIRCUIT THAT CONSISTS OF SPICE ELEMENTS CAN BE DEFINED AND REFERENCED IN A FASHION SIMILAR TO DEVICE MODELS. THE SUBCIRCUIT IS DEFINED IN THE INPUT DECK BY A GROUPING OF ELEMENT CARDS; THE PROGRAM THEN AUTOMATICALLY INSERTS THE GROUP OF ELEMENTS WHEREVER THE SUBCIRCUIT IS REFERENCED. THERE IS NO LIMIT ON THE SIZE OR COMPLEXITY OF SUBCIRCUITS, AND SUBCIRCUITS MAY CONTAIN OTHER SUBCIRCUITS. AN EXAMPLE OF SUBCIRCUIT USAGE IS GIVEN IN APPENDIX A.

**** .SUBCKT CARD

GENERAL FORM .SUBCKT SUBNAM N1 [N2 N3 ...]

EXAMPLE .SUBCKT OPAMP 1 2 3 4

A SUBCIRCUIT DEFINITION IS BEGUN WITH A .SUBCKT CARD. SUBNAM IS THE SUBCIRCUIT NAME, AND N1, N2, ... ARE THE EXTERNAL NODES, WHICH CANNOT BE ZERO. THE GROUP OF ELEMENT CARDS WHICH IMMEDIATELY FOLLOW THE .SUBCKT CARD DEFINE THE SUBCIRCUIT. THE LAST CARD IN A SUBCIRCUIT DEFINITION IS THE .ENDS CARD (SEE BELOW). CONTROL CARDS MAY NOT APPEAR WITHIN A SUBCIRCUIT DEFINITION; HOWEVER, SUBCIRCUIT DEFINITIONS MAY CONTAIN ANYTHING ELSE, INCLUDING OTHER SUBCIRCUIT DEFINITIONS, DEVICE MODELS, AND SUBCIRCUIT CALLS (SEE BELOW). NOTE THAT ANY DEVICE MODELS OR SUBCIRCUIT DEFINITIONS INCLUDED AS PART OF A SUBCIRCUIT DEFINITION ARE STRICTLY LOCAL (I.E., SUCH MODELS AND DEFINITIONS ARE NOT KNOWN OUTSIDE THE SUBCIRCUIT DEFINITION). ALSO, ANY ELEMENT NODES NOT INCLUDED ON THE .SUBCKT CARD ARE STRICTLY LOCAL, WITH THE EXCEPTION OF 0 (GROUND) WHICH IS ALWAYS GLOBAL.

**** .ENDS CARD

GENERAL FORM .ENDS [SUBNAM]

EXAMPLE .ENDS OPAMP

THIS CARD MUST BE THE LAST ONE FOR ANY SUBCIRCUIT DEFINITION. THE SUBCIRCUIT NAME, IF INCLUDED, INDICATES WHICH SUBCIRCUIT DEFINITION IS BEING TERMINATED: IF OMITTED, ALL SUBCIRCUITS BEING DEFINED ARE TERMINATED. THE NAME IS NEEDED ONLY WHEN NESTED SUBCIRCUIT DEFINITIONS ARE BEING MADE.

**** SUBCIRCUIT CALLS

GENERAL FORM XYYYYYYY N1 [N2 N3 ...] SUBNAM

EXAMPLE X1 2 4 17 3 1 MULTI

SUBCIRCUITS ARE USED IN SPICE BY SPECIFYING PSEUDO-ELEMENTS BEGINNING WITH THE LETTER X, FOLLOWED BY THE CIRCUIT NODES TO BE USED IN EXPANDING THE SUBCIRCUIT, FOLLOWED BY THE SUBCIRCUIT NAME. THE NODES MUST BE IN THE ORDER THAT THEY ARE DEFINED IN ON THE .SUBCKT CARD.

CONTROL CARDS

**** TITLE CARD

EXAMPLE POWER AMPLIFIER CIRCUIT

THIS CARD MUST BE THE FIRST CARD IN THE INPUT DECK. ITS CONTENTS ARE PRINTED VERBATIM AS THE HEADING FOR EACH SECTION OF OUTPUT.

**** .END CARD

EXAMPLE .END

THIS CARD MUST ALWAYS BE THE LAST CARD IN THE INPUT DECK. NOTE THAT THE PERIOD IS AN INTEGRAL PART OF THE NAME.

**** COMMENT CARD

GENERAL FORM * ANY COMMENTS

EXAMPLE * RF=1K GAIN SHOULD BE 100

THIS CARD IS PRINTED OUT IN THE INPUT LISTING BUT IS OTHERWISE IGNORED.

**** .TEMP CARD

GENERAL FORM .TEMP T1 [T2 [T3 ...]]

EXAMPLE .TEMP -55.0 25.0 125.0

THIS CARD SPECIFIES THE TEMPERATURES AT WHICH THE CIRCUIT IS TO BE SIMULATED. T1, T2, ... ARE THE DIFFERENT TEMPERATURES, IN DEGREES C. TEMPERATURES LESS THAN -223.0 DEG C ARE IGNORED. MODEL DATA IS SPECIFIED AT TNOM DEGREES (SEE THE .OPTION CARD FOR TNOM); IF THE .TEMP CARD IS OMITTED, THE SIMULATION ALSO WILL BE PERFORMED AT A TEMPERATURE OF TNOM.

**** .WIDTH CARD

GENERAL FORM .WIDTH IN=COLNUM

EXAMPLE .WIDTH IN=72

COLNUM IS THE LAST COLUMN READ FROM EACH LINE OF INPUT; THE SETTING TAKES EFFECT WITH THE NEXT LINE READ. THE DEFAULT VALUE FOR COLNUM IS 80.

**** .OPTIONS CARD

GENERAL FORM .OPTIONS OPT1 OPT2 ... (OR OPT=OPTVAL ...)

EXAMPLE .OPTIONS ACCT LIST NODE

THIS CARD ALLOWS THE USER TO RESET PROGRAM CONTROL AND USER OPTIONS FOR SPECIFIC SIMULATION PURPOSES. ANY COMBINATION OF THE FOLLOWING OPTIONS MAY BE INCLUDED, IN ANY ORDER. *X* (BELOW) REPRESENTS SOME POSITIVE NUMBER.

OPTION	EFFECT
ACCT	CAUSES THE EXECUTION TIME FOR THE VARIOUS SECTIONS OF THE PROGRAM TO BE PRINTED AS WELL AS OTHER ACCOUNTING INFORMATION TO BE PRINTED.
LIST	CAUSES THE SUMMARY LISTING OF THE INPUT DATA TO BE PRINTED.
NOMOD	SUPPRESSES THE PRINTOUT OF THE MODEL PARAMETERS.
NOPAGE	SUPPRESSES PAGE EJECTS
NODE	CAUSES THE NODE TABLE TO BE PRINTED.
OPTS	CAUSES THE OPTION VALUES TO BE PRINTED.
GMIN=X	RESETS THE VALUE OF GMIN, THE MINIMUM CONDUCTANCE ALLOWED BY THE PROGRAM. THE DEFAULT VALUE IS 1.0E-12.
RELTOL=X	RESETS THE RELATIVE ERROR TOLERANCE OF THE PROGRAM. THE DEFAULT VALUE IS 0.001 (0.1 PERCENT).
ABSTOL=X	RESETS THE ABSOLUTE CURRENT ERROR TOLERANCE OF THE PROGRAM. THE DEFAULT VALUE IS 1 PICOAMP.
VNTOL=X	RESETS THE ABSOLUTE VOLTAGE ERROR TOLERANCE OF THE PROGRAM. THE DEFAULT VALUE IS 1 MICROVOLT.
TRTOL=X	RESETS THE TRANSIENT ERROR TOLERANCE. THE DEFAULT VALUE IS 7.0. THIS PARAMETER IS AN ESTIMATE OF THE FACTOR BY WHICH SPICE OVERESTIMATES THE ACTUAL TRUNCATION ERROR.
CHGTOL=X	RESETS THE CHARGE TOLERANCE OF THE PROGRAM. THE DEFAULT VALUE IS 1.0E-14.
NUMDGT=X	RESETS THE NUMBER OF SIGNIFICANT DIGITS PRINTED FOR OUTPUT VARIABLE VALUES. X MUST SATISFY THE RELATION 0 < X < 8. THE DEFAULT VALUE IS 4. NOTE: THIS OPTION IS INDEPENDENT OF THE ERROR TOLERANCE USED BY SPICE (I.E., IF THE VALUES OF OPTIONS RELTOL, ABSTOL, ETC. ARE NOT CHANGED THEN ONE MAY BE PRINTING NUMERICAL *NOISE* FOR NUMDGT > 4.
TNOM=X	RESETS THE NOMINAL TEMPERATURE. THE DEFAULT VALUE IS 27 DEG C (300 DEG K).

ITL1=X RESETS THE DC ITERATION LIMIT. THE DEFAULT IS 100.

ITL2=X RESETS THE DC TRANSFER CURVE ITERATION LIMIT. THE DEFAULT IS 20.

ITL3=X RESETS THE LOWER TRANSIENT ANALYSIS ITERATION LIMIT. THE DEFAULT VALUE IS 4.

ITL4=X RESETS THE TRANSIENT ANALYSIS TIMEPOINT ITERATION LIMIT. THE DEFAULT IS 10.

ITL5=X RESETS THE TRANSIENT ANALYSIS TOTAL ITERATION LIMIT. THE DEFAULT IS 5000.

LIMTIM=X RESETS THE AMOUNT OF TIME RESERVED BY SPICE FOR GENERATING PLOTS. THE DEFAULT VALUE IS 2 (SECONDS).

LIMPTS=X RESETS THE TOTAL NUMBER OF POINTS THAT CAN BE PRINTED OR PLOTTED IN A DC, AC, OR TRANSIENT ANALYSIS. THE DEFAULT VALUE IS 201.

LVLCOD=X IF X IS 2 (TWO), THEN MACHINE CODE FOR THE MATRIX SOLUTION WILL BE GENERATED. OTHERWISE, NO MACHINE CODE IS GENERATED. THE DEFAULT VALUE IS 2.

LVLTIM=X IF X IS 1 (ONE), THE ITERATION TIMESTEP CONTROL IS USED. IF X IS 2 (TWO), THE TRUNCATION-ERROR TIMESTEP IS USED. THE DEFAULT VALUE IS 2. IF METHOD=GEAR AND MAXORD>2 THEN LVLTIM IS SET TO 2 BY SPICE.

METHOD=NAME SETS THE NUMERICAL INTEGRATION METHOD USED BY SPICE. POSSIBLE NAMES ARE GEAR OR TRAPEZOIDAL. THE DEFAULT IS TRAPEZOIDAL.

MAXORD=X SETS THE MAXIMUM ORDER FOR THE INTEGRATION METHOD IF GEAR*S VARIABLE-ORDER METHOD IS USED. X MUST BE BETWEEN 2 AND 6. THE DEFAULT VALUE IS 2.

**** .OP CARD

GENERAL FORM .OP

THE INCLUSION OF THIS CARD IN AN INPUT DECK WILL FORCE SPICE TO DETERMINE THE DC OPERATING POINT OF THE CIRCUIT WITH INDUCTORS SHORTED AND CAPACITORS OPENED. NOTE: A DC ANALYSIS IS AUTOMATICALLY PERFORMED PRIOR TO A TRANSIENT ANALYSIS TO DETERMINE THE TRANSIENT INITIAL CONDITIONS, AND PRIOR TO AN AC SMALL-SIGNAL ANALYSIS TO DETERMINE THE LINEARIZED, SMALL-SIGNAL MODELS FOR NONLINEAR DEVICES.

SPICE PERFORMS A DC OPERATING POINT ANALYSIS IF NO OTHER ANALYSES ARE REQUESTED.

**** .DC CARD

GENERAL FORM .DC SRCNAM VSTART VSTOP VINCR

EXAMPLE .DC VIN 0.25 5.0 0.25

THIS CARD DEFINES THE DC TRANSFER CURVE SOURCE AND SWEEP LIMITS. SRCNAM IS THE NAME OF AN INDEPENDENT VOLTAGE OR CURRENT SOURCE. VSTART, VSTOP, AND VINCR ARE THE STARTING, FINAL, AND INCREMENTING VALUES RESPECTIVELY. THE ABOVE EXAMPLE WILL CAUSE THE VALUE OF THE VOLTAGE SOURCE VIN TO BE SWEPT FROM 0.25 VOLTS TO 5.0 VOLTS IN INCREMENTS OF 0.25 VOLTS.

**** .TF CARD

GENERAL FORM .TF OUTVAR INSRC

EXAMPLE .TF V(5,3) VIN

THIS CARD DEFINES THE SMALL-SIGNAL OUTPUT AND INPUT FOR THE DC SMALL-SIGNAL ANALYSIS. OUTVAR IS THE SMALL-SIGNAL OUTPUT VARIABLE AND INSRC IS THE SMALL-SIGNAL INPUT SOURCE. IF THIS CARD IS INCLUDED, SPICE WILL COMPUTE THE DC SMALL-SIGNAL VALUE OF THE TRANSFER FUNCTION (OUTPUT/INPUT), INPUT RESISTANCE, AND OUTPUT RESISTANCE. FOR THE ABOVE EXAMPLE, SPICE WOULD COMPUTE THE RATIO OF V(5,3) TO VIN, THE SMALL-SIGNAL INPUT RESISTANCE AT VIN, AND THE SMALL-SIGNAL OUTPUT RESISTANCE MEASURED ACROSS NODES 5 AND 3.

**** .SENS CARD

GENERAL FORM .SENS OV1 [OV2 ...]

EXAMPLE .SENS V(9) V(4,3) V(17)

IF A .SENS CARD IS INCLUDED IN THE INPUT DECK, SPICE WILL DETERMINE THE DC SMALL-SIGNAL SENSITIVITIES OF EACH SPECIFIED OUTPUT VARIABLE WITH RESPECT TO EVERY CIRCUIT PARAMETER. NOTE: FOR LARGE CIRCUITS, LARGE AMOUNTS OF OUTPUT CAN BE GENERATED.

**** .NOISE CARD

GENERAL FORM .NOISE OUTV INSRC NUMS

EXAMPLE .NOISE V(5) VIN 10

THIS CARD CONTROLS THE NOISE ANALYSIS OF THE CIRCUIT. OUTV IS A VOLTAGE OUTPUT VARIABLE WHICH DEFINES THE SUMMING POINT. INSRC IS THE NAME OF THE INDEPENDENT VOLTAGE OR CURRENT SOURCE WHICH IS THE NOISE INPUT REFERENCE. NUMS IS THE SUMMARY INTERVAL. SPICE WILL COMPUTE THE EQUIVALENT OUTPUT NOISE AT THE SPECIFIED OUTPUT AS WELL AS THE EQUIVALENT INPUT NOISE AT THE SPECIFIED INPUT. IN ADDITION, THE CONTRIBUTIONS OF EVERY NOISE GENERATOR IN THE CIRCUIT WILL BE PRINTED AT EVERY NUMS FREQUENCY POINTS (THE SUMMARY INTERVAL). IF NUMS IS ZERO, NO SUMMARY PRINTOUT WILL BE MADE.

THE OUTPUT NOISE AND THE EQUIVALENT INPUT NOISE MAY ALSO BE PRINTED AND/OR PLOTTED (SEE THE DESCRIPTION OF THE .PRINT AND .PLOT CARDS).

**** .TRAN CARD

GENERAL FORM .TRAN TSTEP TSTOP [TSTART [TMAX]] [UIC]

EXAMPLES .TRAN 1NS 100NS
.TRAN 1NS 1000NS 500NS
.TRAN 10NS 1US UIC

TSTEP IS THE PRINTING INCREMENT, TSTOP IS THE FINAL TIME, AND TSTART IS THE INITIAL TIME. IF TSTART IS OMITTED, IT IS ASSUMED TO BE ZERO. THE TRANSIENT ANALYSIS ALWAYS BEGINS AT TIME ZERO. IN THE INTERVAL [ZERO, TSTART), THE CIRCUIT IS ANALYZED (TO REACH A STEADY STATE), BUT NO OUTPUTS ARE STORED. IN THE INTERVAL [TSTART, TSTOP], THE CIRCUIT IS ANALYZED AND OUTPUTS ARE STORED. TMAX IS THE MAXIMUM STEPSIZE THAT SPICE WILL USE (DEFAULT VALUE IS TSTOP/50.0).

UIC (USE INITIAL CONDITIONS) IS AN OPTIONAL KEYWORD WHICH INDICATES THAT THE USER DOES NOT WANT SPICE TO SOLVE FOR THE QUIESCENT OPERATING POINT BEFORE BEGINNING THE TRANSIENT ANALYSIS. IF THIS KEYWORD IS SPECIFIED, SPICE USES THE VALUES SPECIFIED USING IC=... ON THE VARIOUS ELEMENTS AS THE INITIAL TRANSIENT CONDITION AND PROCEEDS WITH THE ANALYSIS.

**** .FOUR CARD

GENERAL FORM .FOUR FREQ OV1 [OV2 OV3 ...]

EXAMPLE .FOUR 100KHZ V(5)

THIS CARD CONTROLS WHETHER SPICE PERFORMS A FOURIER ANALYSIS AS A PART OF THE TRANSIENT ANALYSIS. FREQ IS THE FUNDAMENTAL FREQUENCY, AND OV1, ..., ARE THE OUTPUT VARIABLES FOR WHICH THE ANALYSIS IS DESIRED. THE FOURIER ANALYSIS IS PERFORMED OVER THE INTERVAL [TSTOP-PERIOD, TSTOP], WHERE TSTOP IS THE FINAL TIME SPECIFIED FOR THE TRANSIENT ANALYSIS, AND PERIOD IS ONE PERIOD OF THE FUNDAMENTAL FREQUENCY. THE DC COMPONENT AND THE FIRST NINE COMPONENTS ARE DETERMINED. FOR MAXIMUM ACCURACY, TMAX (SEE THE .TRAN CARD) SHOULD BE SET TO PERIOD/100.0 (OR LESS FOR VERY HIGH-Q CIRCUITS).

**** .AC CARD

GENERAL FORM

```
.AC DEC ND FSTART FSTOP
.AC OCT NO FSTART FSTOP
.AC LIN NP FSTART FSTOP
```

EXAMPLES

```
.AC DEC 10 1 10KHZ
.AC DEC 20 1 100KHZ
.AC LIN 100 1 100HZ
```

DEC STANDS FOR DECADE VARIATION, AND ND IS THE NUMBER OF POINTS PER DECADE. OCT STANDS FOR OCTAVE VARIATION, AND NO IS THE NUMBER OF POINTS PER OCTAVE. LIN STANDS FOR LINEAR VARIATION, AND NP IS THE NUMBER OF POINTS. FSTART IS THE STARTING FREQUENCY, AND FSTOP IS THE FINAL FREQUENCY. IF THIS CARD IS INCLUDED IN THE DECK, SPICE WILL PERFORM AN AC ANALYSIS OF THE CIRCUIT OVER THE SPECIFIED FREQUENCY RANGE. NOTE THAT IN ORDER FOR THIS ANALYSIS TO BE MEANINGFUL, AT LEAST ONE INDEPENDENT SOURCE MUST HAVE BEEN SPECIFIED WITH AN AC VALUE.

**** .DISTO CARD

GENERAL FORM

```
.DISTO RLOAD [INTER [SKW2 [REFPWR [SPW2]]]]
```

EXAMPLE

```
.DISTO RL 2 0.95 1.0E-3 0.75
```

THIS CARD CONTROLS WHETHER SPICE WILL COMPUTE THE DISTORTION CHARACTERISTICS OF THE CIRCUIT IN A SMALL-SIGNAL MODE AS A PART OF THE AC SMALL-SIGNAL SINUSOIDAL STEADY-STATE ANALYSIS. THE ANALYSIS IS PERFORMED ASSUMING THAT ONE OR TWO SIGNAL FREQUENCIES ARE IMPOSED AT THE INPUT; LET THE TWO FREQUENCIES BE F1 (THE NOMINAL ANALYSIS FREQUENCY) AND F2 (=SKW2*F1). THE PROGRAM THEN COMPUTES THE FOLLOWING DISTORTION MEASURES:

HD2 - THE MAGNITUDE OF THE FREQUENCY COMPONENT 2*F1 ASSUMING THAT F2 IS NOT PRESENT.
HD3 - THE MAGNITUDE OF THE FREQUENCY COMPONENT 3*F1 ASSUMING THAT F2 IS NOT PRESENT.
SIM2 - THE MAGNITUDE OF THE FREQUENCY COMPONENT F1 + F2.
DIM2 - THE MAGNITUDE OF THE FREQUENCY COMPONENT F1 - F2.
DIM3 - THE MAGNITUDE OF THE FREQUENCY COMPONENT 2*F1 - F2.

RLOAD IS THE NAME OF THE OUTPUT LOAD RESISTOR INTO WHICH ALL DISTORTION POWER PRODUCTS ARE TO BE COMPUTED. INTER IS THE INTERVAL AT WHICH THE SUMMARY PRINTOUT OF THE CONTRIBUTIONS OF ALL NONLINEAR DEVICES TO THE TOTAL DISTORTION IS TO BE PRINTED. IF OMITTED OR SET TO ZERO, NO SUMMARY PRINTOUT WILL BE MADE. REFPWR IS THE REFERENCE POWER LEVEL USED IN COMPUTING THE DISTORTION PRODUCTS. IF OMITTED, A VALUE OF 1 MW (THAT IS, DBM) IS USED. SKW2 IS THE RATIO OF F2 TO F1. IF OMITTED, A VALUE OF 0.9 IS USED (I.E., F2 = 0.9*F1). SPW2 IS THE AMPLITUDE OF F2. IF OMITTED, A VALUE OF 1.0 IS ASSUMED.

THE DISTORTION MEASURES HD2, HD3, SIM2, DIM2, AND DIM3 MAY ALSO BE BE PRINTED AND/OR PLOTTED (SEE THE DESCRIPTION OF THE .PRINT AND .PLOT CARDS).

**** .PRINT CARD

GENERAL FORM .PRINT PRTYPE OV1 [OV2 ... OV8]

EXAMPLES

```
.PRINT TRAN V(4) I(VIN)
.PRINT AC VM(4,2) VR(7) VP(8,3)
.PRINT DC V(2) I(VSRC) V(23,17)
.PRINT NOISE INOISE
.PRINT DISTO HD3 SIM2(DB)
```

THIS CARD DEFINES THE CONTENTS OF A TABULAR LISTING OF ONE TO EIGHT OUTPUT VARIABLES. PRTYPE IS THE TYPE OF THE ANALYSIS (DC, AC, TRAN, NOISE, OR DISTORTION) FOR WHICH THE SPECIFIED OUTPUTS ARE DESIRED. THE FORM FOR VOLTAGE OR CURRENT OUTPUT VARIABLES IS AS FOLLOWS:

V(N1[,N2]) SPECIFIES THE VOLTAGE DIFFERENCE BETWEEN NODES N1 AND N2. IF N2 (AND THE PRECEDING COMMA) IS OMITTED, GROUND (0) IS ASSUMED. FOR THE AC ANALYSIS, FIVE ADDITIONAL OUTPUTS CAN BE ACCESSED BY REPLACING THE LETTER V BY:

```
VR  -  REAL PART
VI  -  IMAGINARY PART
VM  -  MAGNITUDE
VP  -  PHASE
VDB -  20*LOG10(MAGNITUDE)
```

I(VXXXXXXX) SPECIFIES THE CURRENT FLOWING IN THE INDEPENDENT VOLTAGE SOURCE NAMED VXXXXXXX. POSITIVE CURRENT FLOWS FROM THE POSITIVE NODE, THROUGH THE SOURCE, TO THE NEGATIVE NODE. FOR THE AC ANALYSIS, THE CORRESPONDING REPLACEMENTS FOR THE LETTER I MAY BE MADE IN THE SAME WAY AS DESCRIBED FOR VOLTAGE OUTPUTS.

OUTPUT VARIABLES FOR THE NOISE AND DISTORTION ANALYSES HAVE A DIFFERENT FORM FROM THAT OF THE OTHER ANALYSES. THE GENERAL FORM IS

OV[(X)]

WHERE OV IS ANY OF ONOISE (OUTPUT NOISE), INOISE (EQUIVALENT INPUT NOISE), HD2, HD3, SIM2, DIM2, OR DIM3 (SEE DESCRIPTION OF DISTORTION ANALYSIS), AND X MAY BE ANY OF:

```
R  - REAL PART
I  - IMAGINARY PART
M  - MAGNITUDE (DEFAULT IF NOTHING SPECIFIED)
P  - PHASE
DB - 20*LOG10(MAGNITUDE)
```

THUS, SIM2 (OR SIM2(M)) DESCRIBES THE MAGNITUDE OF THE SIM2 DISTORTION MEASURE, WHILE HD2(R) DESCRIBES THE REAL PART OF THE HD2 DISTORTION MEASURE.

**** .PLOT CARD

GENERAL FORM .PLOT PLTYPE OV1 [(PLO1,PHI1)] [OV2 [(PLO2,PHI2)] ... OV8]

EXAMPLES

```
.PLOT DC V(4) V(5) V(1)
.PLOT TRAN V(17,5) (2,5) I(VIN) V(17) (1,9)
.PLOT AC VM(5) VM(31,24) VDB(5) VP(5)
.PLOT DISTO HD2 HD3(R) SIM2
```

THIS CARD DEFINES THE CONTENTS OF ONE PLOT OF FROM ONE TO EIGHT OUTPUT VARIABLES. PLTYPE IS THE TYPE OF ANALYSIS (DC, AC, TRAN, NOISE, OR DISTORTION) FOR WHICH THE SPECIFIED OUTPUTS ARE DESIRED. THE SYNTAX FOR THE OVI IS IDENTICAL TO THAT FOR THE .PRINT CARD, DESCRIBED ABOVE.

THE OPTIONAL PLOT LIMITS (PLO,PHI) MAY BE SPECIFIED AFTER ANY OF THE OUTPUT VARIABLES. ALL OUTPUT VARIABLES TO THE LEFT OF A PAIR OF PLOT LIMITS (PLO,PHI) WILL BE PLOTTED USING THE SAME LOWER AND UPPER PLOT BOUNDS. IF PLOT LIMITS ARE NOT SPECIFIED, SPICE WILL AUTOMATICALLY DETERMINE THE MINIMUM AND MAXIMUM VALUES OF ALL OUTPUT VARIABLES BEING PLOTTED AND SCALE THE PLOT TO FIT. MORE THAN ONE SCALE WILL BE USED IF THE OUTPUT VARIABLE VALUES WARRANT (I.E., MIXING OUTPUT VARIABLES WITH VALUES WHICH ARE ORDERS-OF-MAGNITUDE DIFFERENT STILL GIVES READABLE PLOTS).

THE OVERLAP OF TWO OR MORE TRACES ON ANY PLOT IS INDICATED BY THE LETTER X.

APPENDIX A: EXAMPLE DATA DECKS

THE FOLLOWING DECK DETERMINES THE DC OPERATING POINT AND SMALL-SIGNAL TRANSFER FUNCTION OF A SIMPLE DIFFERENTIAL PAIR.

```
SIMPLE DIFFERENTIAL PAIR
VCC 7 0 12
VEE 8 0 -12
VIN 1 0
RS1 1 2 1K
RS2 6 0 1K
Q1 3 2 4 MOD1
Q2 5 6 4 MOD1
RC1 7 3 10K
RC2 7 5 10K
RE 4 8 10K
.MODEL MOD1 NPN(BF=50 VA=50 IS=1.0E-12 RB=100)
.TF V(5) VIN
.END
```

THE FOLLOWING DECK DETERMINES THE DC TRANSFER CURVE AND THE TRANSIENT PULSE RESPONSE OF A SIMPLE RTL INVERTER. THE INPUT IS A PULSE FROM 0 TO 5 VOLTS WITH DELAY, RISE, AND FALL TIMES OF 2NS AND A PULSE WIDTH OF 30NS. THE TRANSIENT INTERVAL IS 0 TO 100NS, WITH PRINTING TO BE DONE EVERY NANOSECOND.

```
SIMPLE RTL INVERTER
VCC 4 0 5
VIN 1 0 PULSE(0 5 2NS 2NS 2NS 30NS)
RB 1 2 10K
Q1 3 2 0 Q1
RC 3 4 1K
.PLOT DC V(3)
.PLOT TRAN V(3) (0,5)
.PRINT TRAN V(3)
.MODEL Q1 NPN(BF=20 RB=100 TF=0.1NS CJC=2PF)
.DC VIN 0 5 0.1
.TRAN 1NS 100NS
.END
```

THE FOLLOWING DECK DETERMINES THE AC SMALL-SIGNAL RESPONSE OF A ONE-TRANSISTOR AMPLIFIER OVER THE FREQUENCY RANGE 1HZ TO 100MEGHZ.

```
ONE-TRANSISTOR AMPLIFIER
VCC 5 0 12
VEE 6 0 -12
VIN 1 0 AC 1
RS 1 2 1K
Q1 3 2 4 X33
RC 5 3 500
RE 4 6 1K
CBYPASS 4 0 1UFD
.PLOT AC VM(3) VP(3)
.AC DEC 10 1HZ 100MEGHZ
.MODEL X33 NPN(BF=30 RB=50 VA=20)
.END
```

THE FOLLOWING DECK SIMULATES A FOUR-BIT BINARY ADDER, USING SEVERAL SUB-CIRCUITS TO DESCRIBE VARIOUS PIECES OF THE OVERALL CIRCUIT.

```
ADDER - 4 BIT ALL-NAND-GATE BINARY ADDER

*** SUBCIRCUIT DEFINITIONS

.SUBCKT NAND 1 2 3 4
*   NODES:  INPUT(2), OUTPUT, VCC
Q1 9 5 1 QMOD
D1CLAMP 0 1 DMOD
Q2 9 5 2 QMOD
D2CLAMP 0 2 DMOD
RB 4 5 4K
R1 4 6 1.6K
Q3 6 9 8 QMOD
R2 8 0 1K
RC 4 7 130
Q4 7 6 10 QMOD
DVBEDROP 10 3 DMOD
Q5 3 8 0 QMOD
.ENDS NAND

.SUBCKT ONEBIT 1 2 3 4 5 6
*   NODES:  INPUT(2), CARRY-IN, OUTPUT, CARRY-OUT, VCC
X1 1 2 7 6 NAND
X2 1 7 8 6 NAND
X3 2 7 9 6 NAND
X4 8 9 10 6 NAND
X5 3 10 11 6 NAND
X6 3 11 12 6 NAND
X7 10 11 13 6 NAND
X8 12 13 4 6 NAND
X9 11 7 5 6 NAND
.ENDS ONEBIT

.SUBCKT TWOBIT 1 2 3 4 5 6 7 8 9
*   NODES:  INPUT - BIT0(2) / BIT1(2), OUTPUT - BIT0 / BIT1,
*           CARRY-IN, CARRY-OUT, VCC
X1 1 2 7 5 10 9 ONEBIT
X2 3 4 10 6 8 9 ONEBIT
.ENDS TWOBIT

.SUBCKT FOURBIT 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
*   NODES:  INPUT - BIT0(2) / BIT1(2) / BIT2(2) / BIT3(2),
*           OUTPUT - BIT0 / BIT1 / BIT2 / BIT3, CARRY-IN, CARRY-OUT, VCC
X1 1 2 3 4 9 10 13 16 15 TWOBIT
X2 5 6 7 8 11 12 15 14 15 TWOBIT
.ENDS FOURBIT

*** DEFINE NOMINAL CIRCUIT

.MODEL DMOD D
.MODEL QMOD NPN(BF=75 RB=100 CJE=1PF CJC=3PF)
VCC 99 0 DC 5V
VIN1A 1 0 PULSE(0 3 0 10NS 10NS   10NS   50NS)
```

```
VIN1B 2 0 PULSE(0 3 0 10NS 10NS   20NS  100NS)
VIN2A 3 0 PULSE(0 3 0 10NS 10NS   40NS  200NS)
VIN2B 4 0 PULSE(0 3 0 10NS 10NS   80NS  400NS)
VIN3A 5 0 PULSE(0 3 0 10NS 10NS  160NS  800NS)
VIN3B 6 0 PULSE(0 3 0 10NS 10NS  320NS 1600NS)
VIN4A 7 0 PULSE(0 3 0 10NS 10NS  640NS 3200NS)
VIN4B 8 0 PULSE(0 3 0 10NS 10NS 1280NS 6400NS)
X1 1 2 3 4 5 6 7 8 9 10 11 12 0 13 99 FOURBIT
RBIT0 9 0 1K
RBIT1 10 0 1K
RBIT2 11 0 1K
RBIT3 12 0 1K
RCOUT 13 0 1K
.PLOT TRAN V(1) V(2) V(3) V(4) V(5) V(6) V(7) V(8)
.PLOT TRAN V(9) V(10) V(11) V(12) V(13)
.PRINT TRAN V(1) V(2) V(3) V(4) V(5) V(6) V(7) V(8)
.PRINT TRAN V(9) V(10) V(11) V(12) V(13)

.TRAN 1NS 6400NS
*** (FOR THOSE WITH MONEY (AND MEMORY) TO BURN)

.OPT ACCT LIST NODE LIMPTS=6401
.END
```

THE FOLLOWING DECK SIMULATES A TRANSMISSION-LINE INVERTER. TWO TRANSMISSION-LINE ELEMENTS ARE REQUIRED SINCE TWO PROPAGATION MODES ARE EXCITED. IN THE CASE OF A COAXIAL LINE, THE FIRST LINE (T1) MODELS THE INNER CONDUCTOR WITH RESPECT TO THE SHIELD, AND THE SECOND LINE (T2) MODELS THE SHIELD WITH RESPECT TO THE OUTSIDE WORLD.

```
TRANSMISSION-LINE INVERTER
V1 1 0 PULSE(0 1 0 0.1N)
R1 1 2 50
X1 2 0 0 4 TLINE
R2 4 0 50
.SUBCKT TLINE 1 2 3 4
T1 1 2 3 4 Z0=50 TD=1.5NS
T2 2 0 4 0 Z0=100 TD=1NS
.ENDS TLINE
.TRAN 0.1NS 20NS
.PLOT TRAN V(2) V(4)
.END
```

APPENDIX B: NONLINEAR DEPENDENT SOURCES

SPICE ALLOWS CIRCUITS TO CONTAIN DEPENDENT SOURCES CHARACTERIZED BY ANY OF THE FOUR EQUATIONS

I=F(V) V=F(V) I=F(I) V=F(I)

WHERE THE FUNCTIONS MUST BE POLYNOMIALS, AND THE ARGUMENTS MAY BE MULTI-DIMENSIONAL. THE POLYNOMIAL FUNCTIONS ARE SPECIFIED BY A SET OF COEFFICIENTS P0, P1, ..., PN. BOTH THE NUMBER OF DIMENSIONS AND THE NUMBER OF COEFFICIENTS ARE ARBITRARY. THE MEANING OF THE COEFFICIENTS DEPENDS UPON THE DIMENSION OF THE POLYNOMIAL, AS SHOWN IN THE FOLLOWING EXAMPLES:

SUPPOSE THAT THE FUNCTION IS ONE-DIMENSIONAL (THAT IS, A FUNCTION OF ONE ARGUMENT). THEN THE FUNCTION VALUE FV IS DETERMINED BY THE FOLLOWING EXPRESSION IN FA (THE FUNCTION ARGUMENT):

```
FV = P0 + (P1*FA) + (P2*FA*2) + (P3*FA*3) + (P4*FA*4) + (P5*FA*5) + ...
```

SUPPOSE NOW THAT THE FUNCTION IS TWO-DIMENSIONAL, WITH ARGUMENTS FA AND FB. THEN THE FUNCTION VALUE FV IS DETERMINED BY THE FOLLOWING EXPRESSION:

```
FV = P0 + (P1*FA) + (P2*FB) + (P3*FA*2) + (P4*FA*FB) + (P5*FB*2)
     + (P6*FA*3) + (P7*FA*2*FB) + (P8*FA*FB*2) + (P9*FB*3) + ...
```

CONSIDER NOW THE CASE OF A THREE-DIMENSIONAL POLYNOMIAL FUNCTION WITH ARGUMENTS FA, FB, AND FC. THEN THE FUNCTION VALUE FV IS DETERMINED BY THE FOLLOWING EXPRESSION:

```
FV = P0 + (P1*FA) + (P2*FB) + (P3*FC) + (P4*FA*2) + (P5*FA*FB)
     + (P6*FA*FC) + (P7*FB*2) + (P8*FB*FC) + (P9*FC*2) + (P10*FA*3)
     + (P11*FA*2*FB) + (P12*FA*2*FC) + (P13*FA*FB*2) + (P14*FA*FB*FC)
     + (P15*FA*FC*2) + (P16*FB*3) + (P17*FB*2*FC) + (P18*FB*FC*2)
     + (P19*FC*3) + (P20*FA*4) + ...
```

NOTE: IF THE POLYNOMIAL IS ONE-DIMENSIONAL AND EXACTLY ONE COEFFICIENT IS SPECIFIED, THEN SPICE ASSUMES IT TO BE P1 (AND P0 = 0.0), IN ORDER TO FACILITATE THE INPUT OF LINEAR CONTROLLED SOURCES.

FOR ALL FOUR OF THE DEPENDENT SOURCES DESCRIBED BELOW, THE INITIAL CONDITION PARAMETER IS DESCRIBED AS OPTIONAL. IF NOT SPECIFIED, SPICE ASSUMES 0.0. THE INITIAL CONDITION FOR DEPENDENT SOURCES IS AN INITIAL *GUESS* FOR THE VALUE OF THE CONTROLLING VARIABLE. THE PROGRAM USES THIS INITIAL CONDITION TO OBTAIN THE DC OPERATING POINT OF THE CIRCUIT. AFTER CONVERGENCE HAS BEEN OBTAINED, THE PROGRAM CONTINUES ITERATING TO OBTAIN THE EXACT VALUE FOR THE CONTROLLING VARIABLE. HENCE, TO REDUCE THE COMPUTATIONAL EFFORT FOR THE DC OPERATING POINT (OR IF THE POLYNOMIAL SPECIFIES A STRONG NONLINEARITY), A VALUE FAIRLY CLOSE TO THE ACTUAL CONTROLLING VARIABLE SHOULD BE SPECIFIED FOR THE INITIAL CONDITION.

**** VOLTAGE-CONTROLLED CURRENT SOURCES

GENERAL FORM GXXXXXXX N+ N- [POLY(ND)] NC1+ NC1- ... P0 [P1 ...] [IC=...]

EXAMPLES G1 1 0 5 3 0 0.1MMHO
GR 17 3 17 3 0 1M 1.5M IC=2V
GMLT 23 17 POLY(2) 3 5 1 2 0 1M 17M 3.5U IC=2.5, 1.3

N+ AND N- ARE THE POSITIVE AND NEGATIVE NODES, RESPECTIVELY. CURRENT FLOW IS FROM THE POSITIVE NODE, THROUGH THE SOURCE, TO THE NEGATIVE NODE. POLY(ND) ONLY HAS TO BE SPECIFIED IF THE SOURCE IS MULTI-DIMENSIONAL (ONE-DIMENSIONAL IS THE DEFAULT). IF SPECIFIED, ND IS THE NUMBER OF DIMENSIONS, WHICH MUST BE POSITIVE. NC1+, NC1-, ... ARE THE POSITIVE AND NEGATIVE CONTROLLING NODES, RESPECTIVELY. ONE PAIR OF NODES MUST BE SPECIFIED FOR EACH DIMENSION. P0, P1, P2, ..., PN ARE THE POLYNOMIAL COEFFICIENTS. THE (OPTIONAL) INITIAL CONDITION IS THE INITIAL GUESS AT THE VALUE(S) OF THE CONTROLLING VOLTAGE(S). IF NOT SPECIFIED, 0.0 IS ASSUMED. THE POLYNOMIAL SPECIFIES THE SOURCE CURRENT AS A FUNCTION OF THE CONTROLLING VOLTAGE(S). THE SECOND EXAMPLE ABOVE DESCRIBES A CURRENT SOURCE WITH VALUE

I = 1E-3*V(17,3) + 1.5E-3*V(17,3)*2

NOTE THAT SINCE THE SOURCE NODES ARE THE SAME AS THE CONTROLLING NODES, THIS SOURCE ACTUALLY MODELS A NONLINEAR RESISTOR.

**** VOLTAGE-CONTROLLED VOLTAGE SOURCES

GENERAL FORM EXXXXXXX N+ N- [POLY(ND)] NC1+ NC1- ... P0 [P1 ...] [IC=...]

EXAMPLES E1 3 4 21 17 10.5 2.1 1.75
EX 17 0 POLY(3) 13 0 15 0 17 0 0 1 1 1 IC=1.5,2.0,17.35

N+ AND N- ARE THE POSITIVE AND NEGATIVE NODES, RESPECTIVELY. POLY(ND) ONLY HAS TO BE SPECIFIED IF THE SOURCE IS MULTI-DIMENSIONAL (ONE-DIMENSIONAL IS THE DEFAULT). IF SPECIFIED, ND IS THE NUMBER OF DIMENSIONS, WHICH MUST BE POSITIVE. NC1+, NC1-, ... ARE THE POSITIVE AND NEGATIVE CONTROLLING NODES, RESPECTIVELY. ONE PAIR OF NODES MUST BE SPECIFIED FOR EACH DIMENSION. P0, P1, P2, ..., PN ARE THE POLYNOMIAL COEFFICIENTS. THE (OPTIONAL) INITIAL CONDITION IS THE INITIAL GUESS AT THE VALUE(S) OF THE CONTROLLING VOLTAGE(S). IF NOT SPECIFIED, 0.0 IS ASSUMED. THE POLYNOMIAL SPECIFIES THE SOURCE VOLTAGE AS A FUNCTION OF THE CONTROLLING VOLTAGE(S). THE SECOND EXAMPLE ABOVE DESCRIBES A VOLTAGE SOURCE WITH VALUE

V = V(13,0) + V(15,0) + V(17,0)

(IN OTHER WORDS, AN IDEAL VOLTAGE SUMMER).

**** CURRENT-CONTROLLED CURRENT SOURCES

GENERAL FORM FXXXXXXX N+ N- [POLY(ND)] VN1 [VN2 ...] P0 [P1 ...] [IC=...]

EXAMPLES F1 12 10 VCC 1MA 1.3M
FXFEP 13 20 VSENS 0 1

N+ AND N- ARE THE POSITIVE AND NEGATIVE NODES, RESPECTIVELY. CURRENT FLOW IS FROM THE POSITIVE NODE, THROUGH THE SOURCE, TO THE NEGATIVE NODE. POLY(ND) ONLY HAS TO BE SPECIFIED IF THE SOURCE IS MULTI-DIMENSIONAL (ONE-DIMENSIONAL IS THE DEFAULT). IF SPECIFIED, ND IS THE NUMBER OF DIMENSIONS, WHICH MUST BE POSITIVE. VN1, VN2, ... ARE THE NAMES OF VOLTAGE SOURCES THROUGH WHICH THE CONTROLLING CURRENT FLOWS; ONE NAME MUST BE SPECIFIED FOR EACH DIMENSION. THE DIRECTION OF POSITIVE CONTROLLING CURRENT FLOW IS FROM THE POSITIVE NODE, THROUGH THE SOURCE, TO THE NEGATIVE NODE OF EACH VOLTAGE SOURCE. P0, P1, P2, ..., PN ARE THE POLYNOMIAL COEFFICIENTS. THE (OPTIONAL) INITIAL CONDITION IS THE INITIAL GUESS AT THE VALUE(S) OF THE CONTROLLING CURRENT(S) (IN AMPS). IF NOT SPECIFIED, 0.0 IS ASSUMED. THE POLYNOMIAL SPECIFIES THE SOURCE CURRENT AS A FUNCTION OF THE CONTROLLING CURRENT(S). THE FIRST EXAMPLE ABOVE DESCRIBES A CURRENT SOURCE WITH VALUE

I = 1E-3 + 1.3E-3*I(VCC)

**** CURRENT-CONTROLLED VOLTAGE SOURCES

GENERAL FORM HXXXXXXX N+ N- [POLY(ND)] VN1 [VN2 ...] P0 [P1 ...] [IC=...]

EXAMPLES HXY 13 20 POLY(2) VIN1 VIN2 0 0 0 0 1 IC=0.5 1.3
HR 4 17 VX 0 0 1

N+ AND N- ARE THE POSITIVE AND NEGATIVE NODES, RESPECTIVELY. POLY(ND) ONLY HAS TO BE SPECIFIED IF THE SOURCE IS MULTI-DIMENSIONAL (ONE-DIMENSIONAL IS THE DEFAULT). IF SPECIFIED, ND IS THE NUMBER OF DIMENSIONS, WHICH MUST BE POSITIVE. VN1, VN2, ... ARE THE NAMES OF VOLTAGE SOURCES THROUGH WHICH THE CONTROLLING CURRENT FLOWS; ONE NAME MUST BE SPECIFIED FOR EACH DIMENSION. THE DIRECTION OF POSITIVE CONTROLLING CURRENT FLOW IS FROM THE POSITIVE NODE, THROUGH THE SOURCE, TO THE NEGATIVE NODE OF EACH VOLTAGE SOURCE. P0, P1, P2, ..., PN ARE THE POLYNOMIAL COEFFICIENTS. THE (OPTIONAL) INITIAL CONDITION IS THE INITIAL GUESS AT THE VALUE(S) OF THE CONTROLLING CURRENT(S) (IN AMPS). IF NOT SPECIFIED, 0.0 IS ASSUMED. THE POLYNOMIAL SPECIFIES THE SOURCE VOLTAGE AS A FUNCTION OF THE CONTROLLING CURRENT(S). THE FIRST EXAMPLE ABOVE DESCRIBES A VOLTAGE SOURCE WITH VALUE

V = I(VIN1)*I(VIN2)

13.2 Linked List Specifications

Each list element generally contains both integer and real data. Even though both data types require only one word of memory on the CDC 6400 computer, separate subscripts are used to access the two types. All integer data is referenced using the array NODPLC; all real (and character) data is accessed using the (equivalenced) array VALUE. The VALUE-subscript for the first real value is stored in the integer part of the list element and is called LOCV (LOCM for device models); the NODPLC-subscript is called LOC.

In the detailed list-element structure definitions which follow, notation is defined only on first use.

13.2.1 RESISTOR

ID = 1

```
   - 1: subckt info
LOC+ 0: next-pointer
   + 1: LOCV                          LOCV+ 0: element name
   + 2: n1                                + 1: g(TEMP)
   + 3: n2                                + 2: r(TNOM)
   + 4: (n1,n2)                           + 3: temp. coefficient 1
   + 5: (n2,n1)                           + 4: temp. coefficient 2
   + 6: (n1,n1)
   + 7: (n2,n2)
```

Comments:

1) "subckt info" is used to indicate subcircuit relationships (see Sections 5.3 and 6.3 for details). Briefly,

a) if the element is part of the nominal circuit description, "subckt info" is zero.

b) if the element is contained within a subcircuit definition, "subckt info" is the element ID (1 for resistors).

c) if the element is added to the circuit as a result of subcircuit expansion, "subckt info" is a pointer to the "X" element which caused the expansion.

2) "next-pointer" points to (is the NODPLC-subscript of) the next element of the same ID; if there is no next element, "next-pointer" is zero. However, if this element is a part of a subcircuit definition, then "next-pointer" points to the next element within the definition, regardless of ID.

3) "LOCV" points to the real-valued storage for the element.

4) "ni" is element node number i. During READIN, this is the number read from input; after ERRCHK, this entry is replaced by an index into the JUNODE array (the compact renumbered node list).

5) The notation "(a,b)" means a pointer to matrix location (a,b): the a'th row, b'th column entry.

6) "element name" is the element name, left-justified, with blank fill to 8 characters.

7) "g(TEMP)" is the element conductance, adjusted for the value of TEMP.

8) "r(TNOM)" is the input element resistance (assumed to be at TNOM degrees).

13.2.2 CAPACITOR

```
     ID = 2

        - 1: subckt info
     LOC+ 0: next-pointer
        + 1: LOCV                        LOCV+ 0: element name
        + 2: n1                                + 1: computed element value
        + 3: n2                                + 3: initial condition
        + 4: function code                     + 3: argument vector
        + 5: (n1,n2)
        + 6: (n2,n1)
        + 7: tp(function coefficients)
        + 8: LXi offset                  LXi + 0: q(capacitor)
        + 9: exponent vector                   + 1: i(capacitor)
        +10: (n1,n1)
        +11: (n2,n2)
```

Comments:

1) "function code" is zero for "polynomial", the only function currently implemented.

2) The notation "tp(something)" means "a table pointer to a table which contains 'something'".

3) "LXi offset" is the offset for this element into any of the LXi tables (LX0, LX1, etc.) which are used during analysis to contain intermediate analysis results.

4) "argument vector" and "exponent vector" are used by the function evaluation routines.

5) "q(element)" means the charge stored in element.

6) "i(element)" means the current flowing in element.

13.2.3 INDUCTOR

```
     ID = 3

        - 1: subckt info
     LOC+ 0: next-pointer
        + 1: LOCV                        LOCV+ 0: element name
        + 2: n1                                + 1: computed element value
        + 3: n2                                + 2: initial condition
        + 4: function code                     + 3: argument vector
        + 5: IBR
        + 6: (n1,IBR)
        + 7: (n2,IBR)
        + 8: (IBR,n1)
        + 9: (IBR,n2)
        +10: tp(function coefficients)
        +11: LXi offset                  LXi + 0: phi(inductor)
        +12: exponent vector                   + 1: v(inductor)
        +13: (IBR,IBR)
```

Comments:

1) "IBR" is the equation number for the inductor current.

2) "phi(element)" is the flux in element.

3) "v(element)" is the voltage across element.

13.2.4 MUTUAL INDUCTANCE

```
     ID = 4

        - 1: subckt info
     LOC+ 0: next-pointer
        + 1: LOCV                          LOCV+ 0: element name
        + 2: ptr(L1)                             + 1: value
        + 3: ptr(L2)
        + 4: (L1,L2)
        + 5: (L2,L1)
```

Comments:

1) "ptr(Li)" means a pointer to one of the inductor elements which this element is coupling. During READIN, this word is an index into the IUNSAT table; after ERRCHK, it points directly to the inductor.

2) "(Li,Lj)" means the matrix location (A,B) where A is the equation number for the current in Li, and B is the equation number for the current in Lj.

3) During READIN, "value" is K, the coefficient of coupling; after ERRCHK, "value" is M, the mutual inductance.

13.2.5 VOLTAGE-CONTROLLED CURRENT SOURCE

```
     ID = 5

        - 1: subckt info
     LOC+ 0: next-pointer
        + 1: LOCV                        LOCV+ 0: element name
        + 2: n1
        + 3: n2
        + 4: dimension of function
        + 5: function code
        + 6: tp(controlling nodes)
        + 7: tp(matrix locations)
        + 8: tp(function coefficients)
        + 9: tp(argument vector)
        +10: tp(exponent vector)
        +11: tp(initial conditions)
        +12: LXi offset                  LXi + 0: i(source)
                 2 values/dimension:         + 1: controlling v
                                             + 2: di(source)/dv(control)
```

13.2.6 VOLTAGE-CONTROLLED VOLTAGE SOURCE

```
     ID = 6

        - 1: subckt info
     LOC+ 0: next-pointer
        + 1: LOCV                        LOCV+ 0: element name
        + 2: n+
        + 3: n-
        + 4: dimension of function
        + 5: function code
        + 6: IBR
        + 7: tp(controlling nodes)
        + 8: tp(matrix locations)
        + 9: tp(function coefficients)
        +10: tp(argument vector)
        +11: tp(exponent vector)
        +12: tp(initial conditions)
        +13: LXi offset                  LXi + 0: v(source)
                                             + 1: i(source)
                 2 values/dimension:         + 2: controlling v
                                             + 3: dv(source)/dv(control)
```

Comments:

1) "IBR" is the equation number for the current through this element.

13.2.7 CURRENT-CONTROLLED CURRENT SOURCE

```
     ID = 7

        - 1: subckt info
     LOC+ 0: next-pointer
        + 1: LOCV                        LOCV+ 0: element name
        + 2: n1
        + 3: n2
        + 4: dimension of function
        + 5: function code
        + 6: tp(ptrs to controlling currents)
        + 7: tp(matrix locations)
        + 8: tp(function coefficients)
        + 9: tp(argument vector)
        +10: tp(exponent vector)
        +11: tp(initial conditions)
        +12: LXi offset                  LXi + 0: i(source)
                 2 values/dimension:         + 1: controlling i
                                             + 2: di(source)/di(control)
```

13.2.8 CURRENT-CONTROLLED VOLTAGE SOURCE

```
     ID = 8

        - 1: subckt info
     LOC+ 0: next-pointer
        + 1: LOCV                        LOCV+ 0: element name
        + 2: n+
        + 3: n-
        + 4: dimension of function
        + 5: function code
        + 6: IBR
        + 7: tp(ptrs to controlling currents)
        + 8: tp(matrix locations)
        + 9: tp(function coefficients)
        +10: tp(argument vector)
        +11: tp(exponent vector)
        +12: initial conditions
        +13: LXi offset                  LXi + 0: v(source)
                                             + 1: i(source)
                 2 values/dimension:         + 2: controlling i
                                             + 3: dv(source)/di(control)
```

13.2.9 INDEPENDENT VOLTAGE SOURCE

```
     ID = 9

        - 1: subckt info
     LOC+ 0: next-pointer
        + 1: LOCV                          LOCV+ 0: element name
        + 2: n+                                + 1: dc/transient value
        + 3: n-                                + 2: ac value: magnitude
        + 4: function code                     + 3: ac value: phase
        + 5: tp(function coefficients)
        + 6: IBR
        + 7: (n1,IBR)
        + 8: (n2,IBR)
        + 9: (IBR,n1)
        +10: (IBR,n2)
```

Comments:

1) "function code" indicates which built-in function to use for this source, from the following possibilities:

function code	built-in function
0	<no function specified>
1	pulse
2	sine
3	exponential
4	piece-wise linear (PWL)
5	single-frequency fm (SFFM)

2) "IBR" is the equation number for the current flowing in this element.

13.2.10 INDEPENDENT CURRENT SOURCE

```
     ID = 10

        - 1: subckt info
     LOC+ 0: next-pointer
        + 1: LOCV                        LOCV+ 0: element name
        + 2: n1                                + 1: dc/transient value
        + 3: n2                                + 2: ac value: magnitude
        + 4: function code                     + 3: ac value: phase
        + 5: tp(function coefficients)
```

Comments:

1) "function code" indicates which built-in function to use for this source, from the following possibilities:

function code	built-in function
0	<no function specified>
1	pulse
2	sine
3	exponential
4	piece-wise linear (PWL)
5	single-frequency fm (SFFM)

13.2.11 DIODE

ID = 11

```
   - 1: subckt info
LOC+ 0: next-pointer
   + 1: LOCV                          LOCV+ 0: element name
   + 2: np                                + 1: area factor
   + 3: nn                                + 2: IC:  vd
   + 4: np'
   + 5: mp
   + 6: off
   + 7: (np,np')
   + 8: (nn,np')
   + 9: (np',np)
   +10: (np',nn)
   +11: LXi offset                    LXi + 0: v(diode)
   +12: LD0 offset                        + 1: i(diode)
   +13: (np,np)                           + 2: geq
   +14: (nn,nn)                           + 3: q(diode capacitance)
   +15: (np',np')                         + 4: i(diode capacitance)
```

Comments:

1) If the diode has no extrinsic resistance (RS = 0 in the corresponding device model), then np = np' = the p-doped side of the diode, and nn = the n-doped side. If RS is nonzero, then it is modeled as a resistance between np and np', with np' being the true p-doped side of the diode.

2) "mp" is a pointer to the device model for this element.

3) "off" is zero unless the element was specified as "off" in the circuit description (in which case its value is 1).

4) "IC:" means "initial condition specification".

5) "LD0 offset" is the offset for this element into the distortion analysis working storage table LD0.

13.2.12 BJT

ID = 12

```
   - 1: subckt info
LOC+ 0: next-pointer
   + 1: LOCV
   + 2: nc
   + 3: nb
   + 4: ne
   + 5: nc'
   + 6: nb'
   + 7: ne'
   + 8: mp
   + 9: off
   +10: (nc,nc')
   +11: (nb,nb')
   +12: (ne,ne')
   +13: (nc',nc)
   +14: (nc',nb')
   +15: (nc',ne')
   +16: (nb',nb)
   +17: (nb',nc')
   +18: (nb',ne')
   +19: (ne',ne)
   +20: (ne',nc')
   +21: (ne',nb')
   +22: LXi offset
   +23: LD0 offset
   +24: (nc,nc)
   +25: (nb,nb)
   +26: (ne,ne)
   +27: (nc',nc')
   +28: (nb',nb')
   +29: (ne',ne')
```

```
LOCV+ 0: element name
    + 1: area factor
    + 2: IC: vbe
    + 3: IC: vce
```

```
LXi + 0: vbe
    + 1: vbc
    + 2: ic
    + 3: ib
    + 4: gpi
    + 5: gmu
    + 6: gmo
    + 7: go
    + 8: q(cbe)
    + 9: i(cbe)
    +10: q(cbc)
    +11: i(cbc)
    +12: q(ccs)
    +13: i(ccs)
```

Comments:

1) The element nodes nc, nb, and ne are the collector, base, and emitter nodes, respectively. If any of the extrinsic resistances (RB, RC, or RE) are nonzero, then the corresponding resistor is included between nodes nx and nx' (for the appropriate 'x'). Otherwise, nx' is the same node as nx.

13.2.13 JFET

ID = 13

```
   - 1: subckt info
LOC+ 0: next-pointer
   + 1: LOCV
   + 2: nd
   + 3: ng
   + 4: ns
   + 5: nd'
   + 6: ns'
   + 7: mp
   + 8: off
   + 9: (nd,nd')
  +10: (ng,nd')
  +11: (ng,ns')
  +12: (ns,ns')
  +13: (nd',nd)
  +14: (nd',ng)
  +15: (nd',ns')
  +16: (ns',ng)
  +17: (ns',ns)
  +18: (ns',nd')
  +19: LXi offset
  +20: (nd,nd)
  +21: (ng,ng)
  +22: (ns,ns)
  +23: (nd',nd')
  +24: (ns',ns')
```

```
LOCV+ 0: element name
    + 1: area factor
    + 2: IC: vds
    + 3: IC: vgs
```

```
LXi + 0: vgs
    + 1: vgd
    + 2: ig (gate)
    + 3: id (drain)
    + 4: i(gate-to-drain)
    + 5: gm
    + 6: gds
    + 7: ggs
    + 8: ggd
    + 9: q(cgs)
   +10: i(cgs)
   +11: q(cgd)
   +12: i(cgd)
```

Comments:

1) The element nodes nd, ng, and ns are the drain, gate, and source, respectively.

13.2.14 MOSFET

ID = 14

```
   - 1: subckt info
LOC+ 0: next-pointer
   + 1: LOCV
   + 2: nd
   + 3: ng
   + 4: ns
   + 5: nb
   + 6: nd'
   + 7: ns'
   + 8: mp
   + 9: off
   +10: (nd,nd')
   +11: (ng,nb)
   +12: (ng,nd')
   +13: (ng,ns')
   +14: (ns,ns')
   +15: (nb,ng)
   +16: (nb,nd')
   +17: (nb,ns')
   +18: (nd',nd)
   +19: (nd',ng)
   +20: (nd',nb)
   +21: (nd',ns')
   +22: (ns',ng)
   +23: (ns',ns)
   +24: (ns',nb)
   +25: (ns',nd')
   +26: LXi offset
   +27: (nd,nd)
   +28: (ng,ng)
   +29: (ns,ns)
   +30: (nb,nb)
   +31: (nd',nd')
   +32: (ns',ns')
```

```
LOCV+ 0: element name
    + 1: channel length
    + 2: channel width
    + 3: drain diffusion area
    + 4: source diffusion area
    + 5: IC: vds
    + 6: IC: vgs
    + 7: IC: vbs
    + 8: devmod
    + 9: von
    +10: vdsat
```

```
LXi + 0: vbd
    + 1: vbs
    + 2: vgs
    + 3: vgd
    + 4: id
    + 5: ibs
    + 6: ibd
    + 7: gm
    + 8: gds
    + 9: gmbs
    +10: gbd
    +11: gbs
    +12: q(cbd)
    +13: i(cbd)
    +14: q(cbs)
    +15: i(cbs)
    +16: q(cgs)
    +17: i(cgs)
    +18: q(cgd)
    +19: i(cgd)
    +20: q(cgb)
    +21: i(cgb)
```

Comments:

1) The element nodes nd, ng, ns, and nb are respectively the drain, gate, source, and bulk (substrate).

2) "devmod" is the device mode: +1 (-1) for normal (inverse).

3) "von" is the adjusted threshold voltage for the device.

4) "vdsat" is the saturation voltage for the device.

13.2.15 TRANSMISSION LINE

ID = 17

```
   - 1: subckt info
LOC+ 0: next-pointer
   + 1: LOCV
   + 2: n1
   + 3: n2
   + 4: n3
   + 5: n4
   + 6: ni1
   + 7: ni2
   + 8: IBR1
   + 9: IBR2
   +10: (n1,n1)
   +11: (n1,ni1)
   +12: (n2,IBR1)
   +13: (n3,n3)
   +14: (n4,IBR2)
   +15: (ni1,n1)
   +16: (ni1,ni1)
   +17: (ni1,IBR1)
   +18: (ni2,ni2)
   +19: (ni2,IBR2)
   +20: (IBR1,n2)
   +21: (IBR1,n3)
   +22: (IBR1,n4)
   +23: (IBR1,ni1)
   +24: (IBR1,IBR2)
   +25: (IBR2,n1)
   +26: (IBR2,n2)
   +27: (IBR2,n4)
   +28: (IBR2,ni2)
   +29: (IBR2,IBR1)
   +30: LTD offset
   +31: (n3,ni2)
   +32: (ni2,n3)
```

```
LOCV+ 0: element name
    + 1: z0
    + 2: td
    + 3: excitation: IBR1
    + 4: excitation: IBR2
    + 5: IC: v(port 1)
    + 6: IC: i(port 1)
    + 7: IC: v(port 2)
    + 8: IC: i(port 2)
```

```
LTD + 0: past value (LOCV + 3)
    + 1: past value (LOCV + 4)
```

Comments:

1) The model for this element is described below in terms of SPICE circuit elements (although the program does not allow functional dependencies):

```
R1 n1 ni1 Z0
V1 ni1 n2 delay(v(n3,n4),TD)+delay(i(v2),TD)*Z0
R2 n3 ni2 Z0
V2 ni2 n4 delay(v(n1,n2),TD)+delay(i(v1),TD)*Z0
```

13.2.16 SUBCIRCUIT CALL ('X' element)

```
     ID = 19

        - 1: subckt info
     LOC+ 0: next-pointer
        + 1: LOCV                        LOCV+ 0: element name
        + 2: tp('X' element nodes)
        + 3: ptr to subcircuit description
```

Comments:

1) Subcircuit structures and manipulations are sketched as a part of the description of subcircuit definitions in Section 13.2.17.

13.2.17 SUBCIRCUIT DEFINITION

```
     ID = 20

        - 1: subckt info
     LOC+ 0: next-pointer
        + 1: LOCV                    LOCV+ 0: subcircuit name
        + 2: tp(subckt definition nodes)
        + 3: ptr to definition element list
```

Comments:

1) Consider the following SPICE input (not contained within any enclosing subcircuit definition):

```
     X1 1 2 3 SUB1
     .SUBCKT SUB1 17 13 4
     X1 17 6 4 SUB2
     R1 17 13 1K
     Q1 1 4 17 A
     .MODEL A NPN()
     .ENDS SUB1
```

At the end of READIN (before ERRCHK), the internal data structures representing this input would look something like the following:

```
                    ..........
                    :        :
                    :       0:
                    :........:
                    :        :
     LOCATE(19) -->:       0:
                    :........:
                    :        :
                    : (LOCV) : --> "X1"
                    :........:
                    :        :     ..........
                    :(nodlst): --> :       1:
                    :........:     :       2:
                    :        :     :.......3:
                    :(subckt): -----------------> "SUB1" (in IUNSAT table)
                    :........:
```

```
                  ..........
                  :        :
                  :       0:
                  :........:
                  :        :
LOCATE(20) -->    :       0:
                  :........:
                  :        :
                  : (LOCV) : --> "SUB1"
                  :........:
                  :        :     ..........
                  :(nodlst): --> :      17:
                  :........:     :      13:
                  :        :     :......4:
                  :ptr(def):-.
                  :........: |   ..........     ..........
                             |   :        :     :        :
                             |   :(ID)  19:     :(ID)   1:
                             |   :........:     :........:
                             |   :        :     :        :
                             --> :nxt-pntr: --> :nxt-pntr: --> ...
                                 :........:     :........:
                                 :        :     :        :
                                 : <"X1"> :     : <"R1"> :
                                 :        :     :        :
                                 : (defn) :     : (defn) :
                                 :........:     :........:
```

2) After subcircuit expansion (controlled by subroutine SUBCKT in overlay ERRCHK), copies of the elements on the subcircuit element list (LOC + 3) are added (with appropriately modified node numbers) to the element lists. For these added elements, "subckt info" is a pointer to the "X" element which caused the given element to be added.

13.2.18 DIODE MODEL

```
     ID = 21

        - 1: subckt info
     LOC+ 0: next-pointer
        + 1: LOCV                          LOCV+ 0: model name
        + 2: <unused>                            + 1: IS
                                                 + 2: RS
                                                 + 3: N
                                                 + 4: TT
                                                 + 5: CJO
                                                 + 6: PB
                                                 + 7: M
                                                 + 8: EG
                                                 + 9: PT
                                                 +10: KF
                                                 +11: AF
                                                 +12: FC
                                                 +13: BV
                                                 +14: IBV
                                                 +15: f(FC)
                                                 +16: f(FC)
                                                 +17: f(FC)
                                                 +18: vcrit
```

Comments:

1) The model parameter keywords are defined in the User's Guide (Section 13.1).

2) The notation "f(args)" refers to some value (which is a function of the variable(s) "args") which is evaluated once in subroutine MODCHK.

3) One-time only preprocessing of model parameters is done in subroutine MODCHK in the ERRCHK overlay. For the diode model, the preprocessing is as follows:

location	replaced by
LOCV+ 2	1/RS (0 if RS = 0)
+12	FC*PB
+13	computed start of reverse exponential
+15	PB*(1-(1-FC)^(1-M))/(1-M)
+16	(1-FC)^(1+M)
+17	1-FC*(1+M)
+18	N*VT*ln(N*VT/(sqrt(2)*IS))

4) SPICE does not generate an additional node for the diode resistance if RS is zero; instead, the np′ node is set equal to the np node (see Section 13.2.11). This processing of np, together with setting 1/RS to 0 when RS is 0, allows SPICE to treat RS consistently during analysis regardless of the value of the resistance.

13.2.19 BJT MODEL

ID = 22

```
   - 1: subckt info
LOC+ 0: next-pointer
   + 1: LOCV
   + 2: model type
```

```
LOCV+ 0: model name
    + 1: BF
    + 2: BR
    + 3: IS
    + 4: RB
    + 5: RC
    + 6: RE
    + 7: VA
    + 8: VB
    + 9: IK
    +10: C2
    +11: NE
    +12: IKR
    +13: C4
    +14: NC
    +15: TF
    +16: TR
    +17: CCS
    +18: CJE
    +19: PE
    +20: ME
    +21: CJC
    +22: PC
    +23: MC
    +24: EG
    +25: PT
    +26: KF
    +27: AF
    +28: FC
    +29: f(FC,PE)
    +30: f(FC,PE)
    +31: f(FC,PE)
    +32: f(FC,PC)
    +33: f(FC,PC)
    +34: f(FC,PC)
    +35: f(FC,PC)
    +36: vcrit
```

Comments:

1) "model type" is +1 for "npn" and -1 for "pnp".

2) The preprocessing performed in subroutine MODCHK is as follows:

location	replaced by
LOCV+ 4	1/RB (0 if RB = 0)
+ 5	1/RC (0 if RC = 0)
+ 6	1/RE (0 if RE = 0)
+ 7	1/VA (0 if VA = 0)
+ 8	1/VB (0 if VB = 0)
+ 9	1/IK (0 if IK = 0)
+12	1/IKR (0 if IKR = 0)
+28	FC*PE
+29	PE*(1-(1-FC)^(1-ME))/(1-ME)
+30	(1-FC)^(1+ME)
+31	1-FC*(1+ME)
+32	FC*PC
+33	PC*(1-(1-FC)^(1-MC))/(1-MC)
+34	(1-FC)^(1+MC)
+35	1-FC*(1+MC)
+36	VT*ln(VT/(sqrt(2)*IS))

3) See Comment 4 in Section 13.2.18 regarding the processing of device model resistors.

13.2.20 JFET MODEL

ID = 23

```
   - 1: subckt info
LOC+ 0: next-pointer
   + 1: LOCV
   + 2: model type
```

```
LOCV+ 0: model name
    + 1: VTO
    + 2: BETA
    + 3: LAMBDA
    + 4: RD
    + 5: RS
    + 6: CGS
    + 7: CGD
    + 8: PB
    + 9: IS
   +10: KF
   +11: AF
   +12: FC
   +13: f(FC)
   +14: f(FC)
   +15: f(FC)
   +16: vcrit
```

Comments:

1) "model type" is +1 for "njf" and -1 for "pjf".

2) The preprocessing performed in the MODCHK subroutine is described in what follows:

location	replaced by
LOCV+ 4	1/RD (0 if RD = 0)
+ 5	1/RS (0 if RS = 0)
+12	FC*PB
+13	PB*(1-(1-FC)^(1-M))/(1-M)
+14	(1-FC)^(1+M)
+15	1-FC*(1+M)
+16	VT*ln(VT/(sqrt(2)*IS))

Note: for the JFET, M is not a model parameter and is fixed at 0.5.

3) See Comment 4 in Section 13.2.18 regarding the processing of device model resistors.

13.2.21 MOSFET MODEL

ID = 24

```
   - 1: subckt info
LOC+ 0: next-pointer
   + 1: LOCV
   + 2: model type
```

```
LOCV+ 0: model name
    + 1: VTO
    + 2: KP
    + 3: GAMMA
    + 4: PHI
    + 5: LAMBDA
    + 6: RD
    + 7: RS
    + 8: CGS
    + 9: CGD
   +10: CGB
   +11: CBD
   +12: CBS
   +13: TOX
   +14: PB
   +15: JS
   +16: NSUB
   +17: NSS
   +18: NFS
   +19: XJ
   +20: LD
   +21: NGATE
   +22: TPS
   +23: UO
   +24: UCRIT
   +25: UEXP
   +26: UTRA
   +27: KF
   +28: AF
   +29: FC
   +30: f(FC)
   +31: f(FC)
   +32: f(FC)
   +33: VINIT
   +34: vbi
   +35: xd
```

Comments:

1) "model type" is +1 for "nmos" and -1 for "pmos".

2) "VINIT" is the initial guess used for the MOSFET diode voltages.

2) The preprocessing performed by subroutine MODCHK is described below:

location	replaced by
LOCV+ 2	KP*COX
+ 3	sqrt(2*EPSSIL*CHARGE*NSUB)/COX
+ 4	2*VT*ln(NSUB/NI)
+ 6	1/RD (0 if RD = 0)
+ 7	1/RS (0 if RS = 0)
+13	EPSOX/TOX (COX; 0 if TOX = 0)
+29	FC*PB
+30	PB*(1-(1-FC)^(1-M))/(1-M)
+31	(1-FC)^(1+M)
+32	1-FC*(1+M)
+33	VT*ln(VT/(sqrt(2)*JS))
+34	VTO-GAMMA*sqrt(PHI)
+35	sqrt(EPSSIL^2/(CHARGE*NSUB))

Note: for the MOSFET, M is not a model parameter and is fixed at 0.5. Also, some of the above pre-processing is a function of which model parameters were specified by the user. See the listing of subroutine MODCHK for the exact details.

3) See Comment 4 of Section 13.2.18 regarding the processing of device model resistors.

13.2.22 .PRINT DC

```
     ID = 31

        - 1: subckt info
     LOC+ 0: next-pointer
        + 1: LOCV                          LOCV+ 0: pseudo-name
        + 2: analysis type
        + 3: number of variables
        + 4: ptr(variable 1)
        + 5: typ(variable 1)
        + 6: ptr(variable 2)
        + 7: typ(variable 2)
        + 8: ptr(variable 3)
        + 9: typ(variable 3)
        +10: ptr(variable 4)
        +11: typ(variable 4)
        +12: ptr(variable 5)
        +13: typ(variable 5)
        +14: ptr(variable 6)
        +15: typ(variable 6)
        +16: ptr(variable 7)
        +17: typ(variable 7)
        +18: ptr(variable 8)
        +19: typ(variable 8)
```

Comments:

1) "pseudo-name" is meaningless (simply a unique "name" so that subroutine FIND may be used).

2) "analysis type" is actually redundant information:

analysis type	actual analysis
1	dc transfer curve
2	transient
3	ac
4	noise
5	distortion

3) The number of output variables must be between 1 and 8.

4) ptr(var) means a pointer to the appropriate output description on the appropriate output variable list.

5) typ(var) describes the form in which the variable is to be output. Assuming that the variable is "V(1)" then the following table gives the interpretation of "typ":

typ(var)	actual interpretation of "var"
1	V(1)
2	VM(1)
3	VR(1)
4	VI(1)
5	VP(1)
6	VDB(1)

6) The linked-list entries for ID = 32, 33, 34, and 35 are exactly the same as for ID = 31 except that the analysis types are transient, ac, noise, and distortion, respectively.

13.2.23 .PLOT DC

ID = 36

```
   - 1: subckt info
LOC+ 0: next-pointer
   + 1: LOCV
   + 2: analysis type
   + 3: number of variables
   + 4: ptr(variable 1)
   + 5: typ(variable 1)
   + 6: ptr(variable 2)
   + 7: typ(variable 2)
   + 8: ptr(variable 3)
   + 9: typ(variable 3)
  +10: ptr(variable 4)
  +11: typ(variable 4)
  +12: ptr(variable 5)
  +13: typ(variable 5)
  +14: ptr(variable 6)
  +15: typ(variable 6)
  +16: ptr(variable 7)
  +17: typ(variable 7)
  +18: ptr(variable 8)
  +19: typ(variable 8)
```

```
LOCV+ 0: pseudo-name
    + 1: plo(variable 1)
    + 2: phi(variable 1)
    + 3: plo(variable 2)
    + 4: phi(variable 2)
    + 5: plo(variable 3)
    + 6: phi(variable 3)
    + 7: plo(variable 4)
    + 8: phi(variable 4)
    + 9: plo(variable 5)
   +10: phi(variable 5)
   +11: plo(variable 6)
   +12: phi(variable 6)
   +13: plo(variable 7)
   +14: phi(variable 7)
   +15: plo(variable 8)
   +16: phi(variable 8)
```

Comments:

1) "plo(var)" and "phi(var)" are the lower and upper plot limits for the indicated output variable. A value of 0.0 means that no plot limit was specified.

2) The linked-list entries for ID = 37, 38, 39, and 40 are exactly the same as for ID = 36 except that the analysis types are transient, ac, noise, and distortion, respectively.

13.2.24 DC ANALYSIS OUTPUT VARIABLE

ID = 41

```
   - 1: subckt info
LOC+ 0: next-pointer
   + 1: LOCV                          LOCV+ 0: variable name
   + 2:      n1 | ptr(source) | --
   + 3:      n2 |     --      | --
   + 4: ISEQ    |             |
   + 5: type  0 |      1      | 2 - 8
   + 6: <unused>
```

Comments:

1) "variable name" is an artificially-constructed unique identifier for the output variable. It has one of 3 forms:

output variable	"variable name"
v(n1,n2)	(n1*2^12)*2^18 + (n2*2^12)*2^6
i(vxxxxx)	vxxxxx (the name of the source)
noise/distortion	desired output - for example, "nd3"

2) "ISEQ" is the offset within the memory block for the outputs of a given sweep point at which the value for this output variable is stored.

3) "type" is the type of output variable (and determines the meaning of (LOC + 2) and (LOC + 3) as detailed above):

type	meaning	
0	voltage:	v(n1,n2)
1	current:	i(source)
2	noise:	onoise
3	noise:	inoise
4	distortion:	HD2
5	distortion:	HD3
6	distortion:	DIM2
7	distortion:	SIM2
8	distortion:	DIM3

4) The linked-list entries for ID = 42, 43, 44, and 45 are exactly the same as for ID = 41 except that the output variables are for transient, ac, noise, and distortion analyses, respectively.

13.3 Table Specifications

13.3.1 IELMNT

All linked-list elements are stored in this table. It is always the first table allocated; its origin is never changed by the memory manager. SPICE2 takes advantage of that fact and uses absolute (rather than relative) subscripting for the linked lists (that is, NODPLC(variable) rather than NODPLC(IELMNT+offset)).

13.3.2 ISBCKT

This table is used as a stack of pointers to subcircuit definitions. Each entry is one word, and consists of the NODPLC-subscript of the subcircuit definition currently being read (list ID=20). The table is required since subcircuit definitions may be nested.

13.3.3 IUNSAT

This table contains all names which cannot be resolved until after subcircuit expansion has been performed (in the ERRCHK overlay). Each entry is one word.

13.3.4 ITEMPS

This table contains the temperatures at which analysis has been requested to be performed. The first entry is always TNOM; if the table contains more than one entry, the first analysis temperature used is the second entry in the table.

13.3.5 IFOUR

This table contains pointers to the output variables (list ID=42) for which a Fourier analysis has been requested.

13.3.6 ISENS

This table contains pointers to the output variables (list ID = 41) for which a dc sensitivity analysis has been requested.

13.3.7 IFIELD, ICODE, IDELIM, and ICOLUM

These tables are filled by subroutine CARD. For each field of information scanned from the current logical input line, one entry is made in each of the tables, as follows:

```
IFIELD:        VALUE(IFIELD+n) contains either:
                 1)  a name, stored in 8H format,
                 2)  a (floating-point) number

ICODE:         NODPLC(ICODE+n) contains
                 +1 => corresponding IFIELD is name
                  0 => corresponding IFIELD is number
                 -1 => previous field was last one of line

IDELIM:        VALUE(IDELIM+n) contains the character
                 which delimited the field, stored in
                 1H format

ICOLUM:        NODPLC(ICOLUM+n) contains the column
                 position in the current line at which
                 the field started
```

13.3.8 JUNODE

This table contains the input user node numbers. SPICE2 uses a compact set of element node numbers, numbered sequentially from 1 (for the ground node). In routine ERRCHK, an ordered list of user node numbers is constructed in table JUNODE, and all element nodes are replaced by offsets into the JUNODE table. The

overhead is put on obtaining the user node numbers since they are printed perhaps three times during a simulation run, while the compact element node numbers are needed thousands of times.

13.3.9 LSBKPT

This table contains the transient analysis breakpoints, in order of increasing time. Each entry is one word (see Section 6.7). The breakpoint values are obtained from the independent source waveforms and time-delayed elements (transmission lines).

13.3.10 IORDER, IUR, ITABLE, and ITABID

These tables are constructed by TOPCHK, and destroyed after topological checking has been performed. The IORDER table is used to check that a conductive path to ground from every node in the circuit exists. The other tables are used in the generation of the node table, with the following descriptions:

IUR: NODPLC(IUR+n) is the offset from the origin of the ITABLE and ITABID tables to where the pointers to elements connected to node "n" are stored. The last pointer for node "n" is stored at the location in ITABLE indicated by NODPLC(IUR+n+1)-1.

ITABLE: NODPLC(ITABLE+m) contains a pointer into the IELMNT table to a circuit element.

ITABID: NODPLC(ITABID+m) contains the element ID for the element pointed to by the corresponding entry in the ITABLE table.

13.3.11 ISR

This table, together with part of the NUMOFF table, is used to record the nonzero circuit equation coefficient matrix elements. The ISR table contains NSTOP+1 entries. The i'th entry is a list header which points to a linked-list stored in NUMOFF containing the numbers of the nonzero columns of the i'th row of the matrix. The actual list manipulation is performed in subroutine RESERV.

13.3.12 ISEQ

This table is used to keep track of the "voltage-defined" elements (that is, those elements whose currents are unknowns in the Modified Nodal equations - (controlled) voltage sources, inductors, and the dependent sources within transmission-line elements). Each entry points to the (LOC + 0)'th position of one of those elements.

13.3.13 ISEQ1

This table tracks the ISEQ table. It is used to keep track of where element nodes are stored, since the nodes needed for transmission-line elements are stored in (relative) locations different from those of the other elements.

13.3.14 NEQN

This table also tracks the ISEQ table. It records the number of the equation added to the Modified Nodal matrix to evaluate the element current (see variable IBR in subroutine MATLOC).

13.3.15 NODEVS

This table records the number of "voltage-defined" elements connected to each circuit node. It is used in subroutine REORDR to help decide the order in which to swap matrix rows (to eliminate some singularity problems).

13.3.16 NDIAG

In the SETUP overlay, this table records whether the diagonal elements of the matrix are nonzero (due to the loading of some element, or due to propagation of fill-in from the LU factorization process). During dc analysis, this table is used to hold the right-hand side (of the system of circuit equations) while the reordering required by the row-swap process is performed.

13.3.17 NMOFFC

The j'th entry of this table contains the number of nonzero off-diagonal terms in the j'th column of the equation matrix.

13.3.18 NUMOFF

The first NSTOP words of this table contain the number of nonzero off-diagonal terms in each row of the equation matrix. The rest of the table is used to hold the linked-list elements (pointed to by the entries in the ISR table) which indicate which columns are nonzero for a given row of the matrix. Each list element contains two words: a pointer to the next list element (zero if none exists), and the number of the column containing the nonzero matrix element. Note that absolute subscripts are

used for the linked list (the origin of the NUMOFF table must not change while matrix setup is in progress).

13.3.19 ISWAP

This table records the matrix row-swap performed in subroutine REORDR. It is indexed using the renumbered row number, and contains the corresponding original row number.

13.3.20 IEQUA

This table is the inverse of ISWAP: it maps original matrix row numbers into the set of reordered row positions.

13.3.21 IORDER

This table records the simultaneous swapping of rows and columns (equations) done to maximize the sparsity of the equation matrix. It is indexed using the renumbered equation number, and contains the corresponding original equation number.

13.3.22 JMNODE

This table is the inverse of IORDER: it maps equations into their reordered position.

13.3.23 IUR

This table, in conjunction with the IUC table, records the position of the upper-triangular matrix elements in the one-dimensional storage used to hold the matrix coefficients. The i'th entry of the IUR table indicates the starting offset within the IUC table for the nonzero column numbers of the i'th row. The last entry for the i'th row is indicated by the contents of

the (i'th + 1) entry minus 1.

13.3.24 IUC

This table, indexed through the IUR table, records the numbers of the nonzero columns for the upper-triangular equation matrix terms.

13.3.25 ILC

This table is the dual of the IUR table, for the lower-triangular matrix terms. It is accessed by column number, and indicates the starting position within the ILR table where the nonzero row positions are stored.

13.3.26 ILR

This table is the dual of the IUC table, for the lower-triangular part of the equation matrix. It contains the numbers of the nonzero rows for given columns of the matrix.

13.3.27 MACINS

This table contains the machine code (generated by subroutine CODGEN) which is used to solve the system of linearized circuit equations.

13.3.28 LVNIM1

This table contains the solution to the linearized circuit equations obtained from the previous Newton-Raphson iteration.

13.3.29 LVN

This table contains the system of circuit equations (the equation coefficients and the right-hand side vector). The variables LYNL, LYU, and LYL contain the offsets in the LVN table to the beginning of the matrix diagonal, upper-triangle, and lower-triangle terms, respectively.

13.3.30 LX0

This table contains nonlinear device operating-point information. During transient analysis, the table also contains information about capacitor charges and inductor fluxes. A precise description of the table contents for each circuit element is included in Section 13.2. Note that this table always contains information relative to the current iteration/sweep point.

13.3.31 LX1 - LX7

These tables contain the previous contents of the LX0 table. In particular, LX1 contains the previous contents of LX0, LX2 contains the previous contents of LX1, and so forth.

13.3.32 LTD

This table stores previous values of time-delayed sources within the circuit. The LXi tables are inadequate for this purpose because more than seven delayed values of some source may be necessary.

13.3.33 LOUTPT

This table is used to store the values of requested output variables (node voltages and/or voltage-source currents) during analysis. Each entry in the table consists of n+1 words; the first word is the value of the swept variable, and the rest are the corresponding output variable values. For ac analysis, the size of each entry in the table is 2*(n+1) words (for the real and imaginary parts of the complex output variable values).

13.3.34 NDIAGC

This table serves the same purpose for the ac analysis as does the NDIAG table for the dc/transient analyses (see Section 13.3.6), except that this table is twice as long since for the ac analysis the right-hand side is complex-valued.

13.3.35 LVNC

This table is the complex-valued analog to the LVN table (see Section 13.3.29).

13.3.36 LD0 and LD1

These tables are used in the small-signal distortion analysis, which is performed using subroutines DINIT (for initialization) and DISTO. The LD0 table contains sub-computations which need be evaluated only once for a given operating point. The LD1 table holds copies of the solution vector at different frequencies as the distortion analysis is performed.

13.3.37 LVNCT

This table is only allocated if both noise and small-signal distortion analyses have been requested. It is used to store the solution to the current frequency-point because both the noise and distortion analyses destroy the right-hand side vector.

13.3.38 LOCX

This table contains the interpolated x-axis values for output variables.

13.3.39 LOCY

This table contains the interpolated y-axis values of the output variables. For n sweep points and m output variables, it contains m*n words, with all the values for the first output variable first, then the values for the second output variable, and so forth.

13.4 Labeled-Common Variable Descriptions

13.4.1 MEMRY

This COMMON-block contains variables used to manage the dynamically-allocated tables.

name	description
LORG	NODPLC(LORG+n) is word n in memory
ICORE	total number of memory words in use
MAXCOR	maximum value of ICORE for present circuit
MAXUSE	maximum number of words required by circuit
MEMAVL	number of words of available memory (in other words, (words allocated) - (words required))
LDVAL	NODPLC(LDVAL) is last word of dynamically-managed memory
NUMBLK	number of allocated tables (does not include the "table entry" table)
LOCTAB	NODPLC(LOCTAB+1) is the first word of the "table entry" table
LTAB	NODPLC(LTAB+1) is the first word of the entry in the "table entry" table found by function MEMPTR

13.4.2 TABINF

This COMMON-block contains table information (table pointers and some table lengths). (The notation "tp(something)" means "a table pointer to a table containing 'something'.")

name	description
IELMNT	tp(input data linked-list storage)
ISBCKT	tp(subcircuit definition stack)
NSBCKT	size of ISBCKT table
IUNSAT	tp(unsatisfied name list)
NUNSAT	size of IUNSAT table
ITEMPS	tp(analysis temperatures)
NUMTEM	size of ITEMPS table
ISENS	tp(dc sensitivity outputs)
NSENS	size of ISENS table
IFOUR	tp(Fourier analysis outputs)
NFOUR	size of IFOUR table
IFIELD	tp(scanned input line fields)
ICODE	tp(type of data in corresponding IFIELD entry)
IDELIM	tp(delimiter character for IFIELD entry)
ICOLUM	tp(beginning column number of IFIELD entry)
INSIZE	size of IFIELD, ICODE, IDELIM, and ICOLUM tables
JUNODE	tp(nodes used in circuit description)
LSBKPT	tp(breakpoint values for transient analysis)
NUMBKP	size of LSBKPT table
IORDER	tp(record of row renumbering)
JMNODE	tp(inverse(IORDER table))
IUR	tp(IUC indices)
IUC	tp(nonzero columns in upper triangle)
ILC	tp(ILR indices)
ILR	tp(nonzero rows in lower triangle)
NUMOFF	tp(number of nonzero off-diagonal entries in row)
ISR	tp(ptrs to linked-list of matrix structure)
NMOFFC	tp(number of nonzero off-diagonal entries in column)
ISEQ	tp(ptrs to voltage-defined elements (L,E,V,H,T))
ISEQ1	tp(element node flag for corresponding ISEQ entry)
NEQN	tp(equation number for corresponding ISEQ entry)
NODEVS	tp(number of voltage-defined branches incident to node)
NDIAG	tp(in REORDR: whether diagonal is nonzero; in analysis: copy of LVN)
ISWAP	tp(record of equation swaps)
IEQUA	tp(inverse(ISWAP))
MACINS	tp(machine instructions (code))
LVNIM1	tp(previous solution (copy of LVN))
LX0	tp(current circuit state information)
LVN	tp(right-hand-side for set of circuit equations)

LYNL	LVN offset: matrix diagonal terms
LYU	LVN offset: matrix upper-triangle terms
LYL	LVN offset: matrix lower-triangle terms
LX1 - LX7	tp(past values of LX0 table)
LD0	tp(distortion analysis constants)
LD1	tp(distortion analysis work space)
LTD	tp(previous timepoint information for elements which involve delay (transmission lines))
LOUTPT	tp(saved output values to be printed/plotted)

13.4.3 MISCEL

This COMMON-block contains miscellaneous variables containing the program version, execution time, etc.

name	description
APROG()	program version and release date
ATIME	real-time clock (hh:mm:ss)
ADATE	date (dd mmm yy)
ATITLE()	circuit title (first input line)
RSTATS()	program statistics:
	(1) - cpu time: READIN
	(2) - cpu time: SETUP
	(3) - cpu time: dc transfer curves
	(4) - no. iter: dc transfer curves
	(5) - cpu time: dc operating point
	(6) - no. iter: dc operating point
	(7) - cpu time: ac analysis
	(8) - no. iter: ac analysis
	(9) - cpu time: transient analysis
	(10) - no. iter: transient analysis
	(11) - cpu time: print/plot generation
	(20) - NSTOP
	(21) - no. matrix terms before REORDR
	(22) - no. matrix terms after REORDR
	(23) - no. of fillin matrix terms
	(26) - "operation count" for matrix
	(27) - percent sparsity of matrix
IWIDTH	input line width
LWIDTH	output line width
NOPAGE	nonzero => inhibit page ejects for subtitles

13.4.4 LINE

This COMMON-block contains variables pertaining to the next input line to be scanned.

name	description
ACHAR	input character (in 1H format)
AFIELD()	next input line (in A8 format)
OLDLIN()	previous input line (in A8 format)
KNTRC	column number in current input line
KNTLIM	last column to scan in current input line

13.4.5 CIRDAT

This COMMON-block contains miscellaneous circuit data.

name	description
LOCATE()	LOCATE(n) points to the first element with ID = n (in table IELMNT)
JELCNT()	JELCNT(n) is the number of elements with ID = n
NUNODS	number of user nodes (in nominal input circuit)
NCNODS	number of circuit nodes (after subckt expansion)
NUMNOD	total circuit nodes (including nodes added for extrinsic resistances in device models)
NSTOP	number of circuit equations
NUT	number of nonzero terms in upper triangle
NLT	number of nonzero terms in lower triangle
NXTRM	size of LX0 table
NDIST	size of LD0 table
NTLIN	size of LTD table
IBR	temporary variable: next equation number available for element currents
NUMVS	number of voltage-defined elements (inductors, (in)dependent voltage sources, and transmission lines)

13.4.6 MOSARG

This COMMON-block contains the mosfet device parameters used in subroutines MOSEQN and MOSCAP.

name	description
GAMMA	gamma
BETA	k′
VTO	zero-bias threshold voltage
PHI	inversion potential
COX	thin-oxide capacitance
VBI	built-in voltage
XNFS	fast surface state density
XNSUB	effective substrate doping
XD	depletion-layer-width constant
XJ	metallurgical junction depth
XL	channel length
XLAMDA	lambda
UTRA	mobility transfield factor
UEXP	mobility exponent
VBP	voltage at which mobility begins to degrade
VON	threshold voltage
VDSAT	saturation voltage
GM	d(id)/d(vgs)
GMBS	d(id)/d(vbs)
GDS	d(id)/d(vds)
CDRAIN	drain current (id)

13.4.7 STATUS

This COMMON-block contains variables which determine the program analysis status.

name	description
OMEGA	2 * PI * FREQ
TIME	simulation time
DELTA	current timestep
DELOLD()	previous timesteps
AG()	truncation-error estimation coefficients
VT	thermal voltage (=K*T/Q or BOLTZ*TEMP/CHARGE)
XNI	intrinsic carrier concentration
EGFET	energy gap for mosfets
MODE	analysis mode: 1 - dc analysis (subtype MODEDC) 2 - transient analysis 3 - ac analysis
MODEDC	dc analysis type: 1 - dc operating point 2 - initial transient point 3 - dc transfer curve
ICALC	number of output sweep points
INITF	analysis state within iteration for given sweep point: 1 - converge with 'off' devices floating 2 - initialize junction voltages 3 - converge with 'off' devices held off 4 - store small-signal parameters 5 - first timepoint in transient analysis 6 - first iteration of sweep point: predict junction voltages
METHOD	numerical integration method flag: 1 - trapezoidal 2 - Gear
IORD	integration order
MAXORD	maximum integration order
NONCON	number of nonconvergent branches
ITERNO	iteration number for current sweep point
ITEMNO	temperature number
NOSOLV	1 => "use initial conditions" (UIC) specified for transient analysis

13.4.8 FLAGS

This COMMON-block contains miscellaneous output and error flags.

name	description
IPRNTA	nonzero => print accounting information
IPRNTL	nonzero => print circuit summary list
IPRNTM	nonzero => print device model summary
IPRNTN	nonzero => print node table
IPRNTO	nonzero => print options
LIMTIM	seconds of cpu time to reserve for output
LIMPTS	maximum number of sweep points
LVLCOD	machine code flag: 1 - no machine code 2 - generate and execute machine code to solve the circuit equations
LVLTIM	timestep determination algorithm flag: 1 - iteration count 2 - truncation-error estimation
ITL1	iteration limit: dc operating point
ITL2	iteration limit: per point of dc transfer curve
ITL3	lower transient analysis limit (if LVLTIM = 1)
ITL4	upper transient analysis limit
ITL5	iteration limit for entire transient analysis
IGOOF	local error flag
NOGO	global error flag (error if nonzero)
KEOF	nonzero => end-of-file on input

13.4.9 KNSTNT

This COMMON-block contains various execution-time constants.

name	description
TWOPI	2 * PI
XLOG2	ln(2)
XLOG10	ln(10)
ROOT2	sqrt(2)
RAD	1 radian, in degrees (360/TWOPI)
BOLTZ	Boltzmann's constant
CHARGE	electronic charge
CTOK	Kelvin temperature equivalent to 0 Centigrade
GMIN	minimum branch conductance
RELTOL	relative convergence tolerance
ABSTOL	absolute current convergence tolerance
VNTOL	absolute voltage convergence tolerance
TRTOL	truncation-error estimation tolerance
CHGTOL	charge error tolerance
EPS0	permittivity of free space
EPSSIL	permittivity of silicon
EPSOX	permittivity of silicon dioxide

13.4.10 DC

This COMMON-block contains variables pertinent to the dc analyses.

name	description
TCSTAR	start value for dc transfer curve
TCSTOP	final value for dc transfer curve
TCINCR	increment for dc transfer curve
ICVFLG	number of dc transfer curve sweep points (0 => no dc transfer curve requested)
ITCELM	pointer to source to be swept
KSSOP	nonzero => dc operating point must be computed (small-signal transfer function and/or dc sensitivity analyses requested)
KINEL	small-signal transfer function analysis flag: 0 - no small-signal transfer function >0 - pointer to input source
KIDIN	ID of element pointed to by KINEL
KOVAR	pointer to output variable for small-signal transfer function analysis
KIDOUT	<unused>

13.4.11 AC

This COMMON-block contains variables pertinent to the ac analyses.

name	description
FSTART	starting frequency for ac analysis
FSTOP	final frequency for ac analysis
FINCR	if frequency sweep is linear, FINCR is an additive constant. if frequency sweep is logarithmic (octave/decade), FINCR is a multiplicative constant.
SKW2	ac distortion analysis: skw2
REFPRL	ac distortion analysis: reference power level
SPW2	ac distortion analysis: spw2
JACFLG	number of ac sweep points (0 => no ac analysis requested)
IDFREQ	type of frequency sweep: 1 - decade 2 - octave 3 - linear
INOISE	nonzero => perform noise analysis
NOSPRT	noise summary interval: every NOSPRT sweep points, a noise summary is printed
NOSOUT	pointer to noise analysis output variable
NOSIN	pointer to noise analysis input source
IDIST	distortion analysis flag: 0 - no distortion analysis >0 - pointer to distortion load resistor
IDPRT	distortion summary interval: every IDPRT sweep points, a distortion summary is printed

13.4.12 TRAN

This COMMON-block contains variables pertinent to the transient analysis.

name	description
TSTEP	transient analysis print interval
TSTOP	final simulation time for transient analysis
TSTART	time at which to begin saving outputs
DELMAX	maximum value for DELTA (timestep)
TDMAX	maximum delay for delayed-value elements (transmission lines)
FORFRE	Fourier analysis frequency
JTRFLG	number of transient output points (0 => no transient analysis requested)

13.4.13 OUTINF

This COMMON-block contains variables used in the generation of output.

name	description
STRING()	scratch line buffer used in generating output
YVAR()	output values for one line
XSTART	starting x-value for output
XINCR	increment to get next x-value for output
ITAB()	pointers to output variable definitions
ITYPE()	typ() values for output variables
ILOGY()	logarithmic plot flag: 1 - not logarithmic 2 - logarithmic
NPOINT	number of output points
NUMOUT	number of output for current analysis mode
KNTR	temporary output variable counter
NUMDGT	number of significant digits to use for tabular output variable listing

13.4.14 CJE

This COMMON-block contains variables describing the current job environment. Its contents are set by subroutine GETCJE.

name	description
JOBNAM	<unused>
USRID1	<unused>
USRID2	<unused>
MAXTAP	<unused>
ITAPE	<unused>
MAXECS	<unused>
IECS	<unused>
MAXMEM	maximum amount of memory available to SPICE
IMEM	<unused>
MAXLIN	<unused>
ILINES	number of output lines generated
MAXPCH	<unused>
IPUNCH	<unused>
MAXTIM	cpu time limit (in milliseconds)
ITIME	elapsed cpu time (in milliseconds)
MAXPPU	<unused>
IPPU	<unused>
IEFTIM	<unused>
ISPTIM	<unused>
MAXDLR	<unused>
ICOST	cost of job, excluding lines printed (in cents)
XCJEX()	<unused>

13.4.15 BLANK

This COMMON-block contains the array into which all dynamic memory allocation is made.

name	description
VALUE()	global dynamically-managed allocation array
NODPLC()	VALUE-equivalence for integer values
CVALUE()	VALUE-equivalence for complex values

13.5 MOSFET Equations

The MOSFET model in SPICE2 is based on three independent variables: VGS, VDS, and VBS. Variables used in the model equations are defined at the end of this section.

Channel modulation:

```
Leff    = L0*(1-Lambda*VDS)
```

Mobility variation:

```
Ueff    = U0*(Vbp/(Vgst-UTRA*VDS))^UEXP

              for (VGS-VTO-UTRA*VDS) > Vbp
                      (otherwise, Ueff = U0)
```

Two-dimensional body effect:

```
Gammad = Gamma*(1-G1)

G1     = Xj*(sqrt(1+2*W/Xj)-1)/L0

W      = Xd*sqrt(PHIs-VBS)
```

Saturation voltage:

```
Vdsat  = VGS-Vbi+0.5*Gammad^2*V1

V1     = 1-sqrt(1+4*(VGS-Vfb-VBS)/Gammad^2))
```

Drain current:

Subthreshold region

```
Id     = Beta*(F1-2*Gammad*F2/3)*exp(VGS-VON)/VFACT

VON    = Vbi+Gammad*sqrt(PHIs-VBS)
            +k*T/q*(1+q*Nfs/Cox+Gamma/2*1/sqrt(PHIs-VBS))

F2     = (PHIs+VSATON-VBS)^(3/2)-(PHIs-VBS)^(3/2)

VFACT  = VON-Vbi-VSATON/2-(2/3)*Gammad/VSATON*F2

F1     = (VON-Vbi-VSATON/2)*VSATON

VSATON = minimum(VDS, Vdsat at VGS = VON)
```

Linear region

```
Id      = Beta*(F1-2*Gammad*F2/3)

Beta    = Kp*(Ueff/U0)*(L0/Leff)

Kp      = U0*Cox*W/L0

F1      = (VGS-Vbi-VDS/2)*VDS

F2      = (PHIs+VDS-VBS)^(3/2)-(PHIs-VBS)^(3/2)
```

Saturation region

As in linear, with "VDS" replaced by "VDSAT".

Capacitance:

```
CBD     = CBD(0)/(1-VBD/PB)^0.5        for VBD < FC*PB
        = linear extrapolation         for VBD > FC*PB

CBS     = same equation as CBD

CGB     = overlap capacitance (from geometry) + C1(Cox)

C1(Cox)= constant W*L*Cox for VGS < VTO-PHIs
        = linear ramp to 0 for VTO-PHIs < VGS < VTO
        = 0 for VGS > VTO

CGS     = overlap capacitance (from geometry) + C2(Cox)

C2(Cox)= constant 0 for VGS < VTO-PHIs/2
        = linear ramp to (2/3)*Cox for VTO-PHIs/2 < VGS < VTO
        = constant (2/3)*Cox for VTO < VGS < VTO+VDS
        = (2/3)*Cox*(1-(VGS-VTO-VDS)^2/(2*(VGS-VTO)-VDS)^2)
             for VGS > VT+VDS

CGD     = constant 0 for VGS < VT+VDS
        = (2/3)*Cox*(1-(VGS-VTO)^2/(2*(VGS-VTO)-VDS)^2)
             for VGS > VT+VDS
```

Variable definitions:

```
Gamma   = model parameter
             if not specified and NSUB is nonzero,
               Gamma is computed using the expression

                    Gamma = Xd*sqrt(Vx+sqrt(1+Vx^2))/(L0*VDS)
               where
                    Vx    = (VDS-Vdsat)/4
```

Xd	=	depletion constant (sqrt(2*EPSSIL/q*NSUB))
PHIs	=	surface potential = 2*PHIf
Vfb	=	PHIms - Qss/Cox
Vbi	=	Vfb + PHIs
q	=	electronic charge
k	=	Boltzmann's constant
T	=	temperature (in degrees Kelvin)

13.6 Element Load Templates

This section describes the various element "stamps" or locations in the circuit equation coefficient matrix to which different elements add. The notation used below is described in the following figure:

```
ni       the i'th element node
i(elt)   the equation added to the Modified Nodal equations to
         solve for the current in element "elt"
A()      the Modified Nodal coefficient matrix
B()      the right-hand side for the set of circuit equations
```

13.6.1 Resistor

```
A(n1,n1) = A(n1,n1) + 1/R
A(n1,n2) = A(n1,n2) - 1/R
A(n2,n1) = A(n2,n1) - 1/R
A(n2,n2) = A(n2,n2) + 1/R
```

13.6.2 Capacitor

```
A(n1,n1) = A(n1,n1) + C/h        (h = timestep; infinity for dc)
A(n1,n2) = A(n1,n2) - C/h        (note:  conductance depends upon
A(n2,n1) = A(n2,n1) - C/h            the integration method)
A(n2,n2) = A(n2,n2) + C/h

B(n1) = B(n1) + C/h*V(C)         (V(C) = voltage across C)
B(n2) = B(n2) - C/h*V(C)
```

13.6.3 Inductor

```
A(n1  ,i(L)) = +1
A(n2  ,i(L)) = -1
A(i(L),  n1) = +1
A(i(L),  n2) = -1
A(i(l),i(l)) = -L/h              (h = timestep; infinity for dc)

B(i(L)) = B(i(L)) - L/h*I(L)     (I(L) = current through L)
```

13.6.4 Voltage-controlled Current Source

```
A(n1,nc1) = A(n1,nc1) + gm
A(n1,nc2) = A(n1,nc2) - gm
A(n2,nc1) = A(n2,nc1) - gm
A(n2,nc2) = A(n2,nc2) + gm
```

(gm = d(source)/d(controlling nodes (nc1,nc2))

13.6.5 Voltage-controlled Voltage Source

```
A(  n1,i(E)) = +1
A(  n2,i(E)) = -1
A(i(E),  n1) = +1
A(i(E),  n2) = -1
A(i(E), nc1) = -Av
A(i(E), nc2) = +Av
```

(Av = d(source)/d(controlling voltage (nc1,nc2)))

13.6.6 Current-controlled Current Source

```
A(n1,i(Vc)) = +Ai
A(n2,i(Vc)) = -Ai
```

(Vc = voltage source through which controlling
current flows;
Ai = d(source)/d(controlling current i(V)))

13.6.7 Current-controlled Voltage Source

```
A(  n1, i(H)) = +1
A(  n2, i(H)) = -1
A(i(H),   n1) = +1
A(i(H),   n2) = -1
A(i(H),i(Vc)) =  R
```

(R = d(source)/d(current flowing in Vc))

13.6.8 Independent Voltage Source

```
A(  n1,i(V)) = A(  n1,i(V)) + 1
A(  n2,i(V)) = A(  n2,i(V)) - 1
A(i(V),  n1) = A(i(V),  n1) + 1
A(i(V),  n2) = A(i(V),  n2) - 1
A(i(V),i(V)) = A(i(V),i(V)) + 1

B(i(V)) = V
```

13.6.9 Independent Current Source

```
B(n1) = B(n1) - I
B(n2) = B(n2) + I
```

13.7 Potential Conversion Problems

13.7.1 Call-by-address

An implicit assumption in the design of the memory management package in SPICE2 is that it is possible to change the value of some variable given its "address". The (U.C. Computer Center) system function LOCF is used for that purpose: it takes a single argument, and returns the address of the argument. Problems arise if the FORTRAN compiler uses call-by-value instead of call-by-address, since in that event calls to LOCF will always return the address of the local copy of the passed parameter.

13.7.2 Dynamic Region Size

SPICE2 is written so as to take advantage of the possibility of dynamically changing its region size (the number of words of memory allocated to the job by the operating system). This dynamic region variation is achieved by using "illegal" subscripts in the array VALUE to reference memory above (higher-addressed than) the last word of code for the program. Thus, by changing the number of words allocated to the jobstep, the size of VALUE can be effectively varied.

If dynamic region variation is either not desirable or impossible, one may simply declare

```
DIMENSION VALUE(50000)
```

and live with a fixed-size maximum amount of available space, with corresponding waste for small-sized circuits.

13.7.3 Characters-per-word

SPICE2 assumes 8 characters per real-valued variable. This assumption is "wired" into the code in many places; fortunately, it is a reasonable assumption for most large computers.

13.7.4 Words-per-real-value

SPICE2 was originally written for the CDC 6400 computer - which has 60-bit words. As a result, only one word is used both for integer and real values. However, since more than 48 bits of floating-point precision are necessary for accurate solutions of medium-sized circuits, most non-CDC computers require that floating-point variables be typed "DOUBLE PRECISION" (or "REAL*8"). This causes problems in SPICE because the memory manager allocates space in terms of "words" instead of "some number of integers/reals" (which would be more machine independent). Changes in SPICE2 to solve this problem for a specific computer are extensive because modifications must be made throughout the program.

13.7.5 Array subscripts

As mentioned above, SPICE2 achieves its dynamic region size variation, together with dynamic memory allocation, by using "illegal" (in a strict ANSI FORTRAN sense) subscripts. There are two ways to resolve this problem on computers which check array subscripts. One may locate the origin of the /BLANK/ COMMON-block at the end of all executed code for SPICE2, (in which case the dynamic region-size capability may be used). Alternatively, one can simply declare VALUE to be of size 25000 (or some other

large number) and dispense with dynamic region-size variations (and put up with the wasted memory in which this method results).

13.8 Program Listing

Persons who wish to obtain the SPICE2 program should request the updated version from the Electronics Research Laboratory, University of California, Berkeley, California 94720. The Laboratory charges only a nominal handling charge for any of the programs that it has available.

14 References

[1] L. W. Nagel, "SPICE2: A Computer Program to Simulate Semiconductor Circuits," ERL Memo No. ERL-M520, Electronics Research Laboratory, University of California, Berkeley, May, 1975.

[2] SCOPE Reference Manual, Publication No. 60189400, Control Data Corporation, Palo Alto, California, 1971, pp. 3-51 - 3-52.

[3] C. W. Ho, A. E. Ruehli, and P. A. Brennan, "The Modified Nodal Approach to Network Analysis," Proceedings 1974 International Symposium on Circuits and Systems, San Francisco, April 1974, pp. 505-509.

[4] C. W. Ho, A. E. Ruehli, and P. A. Brennan, "The Modified Nodal Approach to Network Analysis," Trans. IEEE, vol. CAS-22, No. 6, June 1975, pp. 504-509.

[5] R. D. Berry, "An Optimal Ordering of Electronic Circuit Equations for a Sparse Matrix Solution," Trans. IEEE, vol. CT-18, January 1971, pp. 40-50.

[6] H. M. Markowitz, "The Elimination Form of the Inverse and Its Application to Linear Programming," Management Science, vol. 3, April 1957, pp. 255-269.

[7] M. Nakhla, K. Singhal, and J. Vlach, "An Optimal Pivoting Order for the Solution of Sparse Systems of Equations," Trans. IEEE, vol. CAS-21, March 1974, pp. 222-225.

[8] C. A. Desoer and E. S. Kuh, Basic Circuit Theory, New York: McGraw-Hill, 1969, pp. 429-430.

[9] S. W. Director and R. A. Rohrer, "The Generalized Adjoint Network and Network Sensitivities," Trans. IEEE, vol. CT-16, August 1969, pp. 318-323.

[10] A. S. Grove, Physics and Technology of Semiconductor Devices, New York: Wiley, 1967.

[11] R. M. Swanson and J. D. Meindl, "Ion-Implanted Complementary MOS Transistors in Low-Voltage Circuits," IEEE Journal of Solid-State Circuits, vol. SC-7, No. 2, April 1972.

[12] H. C. Poon, L. D. Yau, and R. L. Johnson, "DC Model for Short-Channel IGFET's," 1973 IEDM Technical Digest, pp. 156-159.

[13] J. E. Meyer, "MOS Models and Circuit Simulation," RCA Review, vol. 32, March 1971.

[14] A. Ralston, A First Course in Numerical Analysis, New York: McGraw-Hill Book Company, 1965, pp. 408-411.

[15] C. W. Gear, "Numerical Integration of Stiff Ordinary Equations," University of Illinois at Urbana-Champaign, Report 221, January 20, 1967.

[16] R. A. Rohrer, L. W. Nagel, R. G. Meyer, and L. Weber, "Computationally Efficient Electronic Circuit Noise Calculations," IEEE Journal of Solid-State Circuits, vol. SC-6, August 1971, pp. 204-213.

[17] R. G. Meyer, L. W. Nagel, and S. K. Lui, "Computer Simulation of 1/f Noise Performance of Electronic Circuits," IEEE Journal of Solid-State Circuits, vol. SC-8, June 1973, pp. 237-240.

[18] S. H. Chisholm and L. W. Nagel, "Efficient Computer Simulation of Distortion in Electronic Circuits," Trans. IEEE, vol. CT-20, November 1973, pp. 742-745.

[19] Y. L. Kuo, "Distortion Analysis of Bipolar Transistor Circuits," Trans. IEEE, vol CT-20, November 1973, pp. 709-717.

[20] S. Narayanan, "Transistor Distortion Analysis Using Volterra Series Representation," Bell System Technical Journal, May-June 1967.