CLRC

Technical Report
RAL-TR-1999-006

The EVEREST Doping Profile Module: Version 4

J V Ashby R F Fowler and C Greenough

28th January 1999

COUNCIL FOR THE CENTRAL LABORATORY OF THE RESEARCH COUNCILS

Enquiries about copyright, reproduction and requests for additional copies of this report should be addressed to:

The Central Laboratory of the Research Councils
Library and Information Services
Rutherford Appleton Laboratory
Chilton
Didcot
Oxfordshire
OX11 0QX
Tel: 01235 445384 Fax: 01235 446403
E-mail library@rl.ac.uk

ISSN 1358-6254

Neither the Council nor the Laboratory accept any responsibility for loss or damage arising from the use of information contained in any of their reports or in any communication about their tests or investigations.

The EVEREST Doping Profile Module: Version 4

JV Ashby, RF Fowler and C Greenough

December 1998

Abstract

In this report we describe the EVEREST Doping Module which forms part of the EVEREST suite of programs.

The doping module is responsible for generating a neutral file which gives the acceptor and donor densities at each node the device mesh. The neutral file also contains details of the functions used to generate the doping as mesh refinement in the solver requires this. Commands allow the definition of background doping, regions of uniform doping, non-uniform doping relating to windows found in the Geometry neutral file and a user-programmable FORTRAN subroutine which computes doping concentrations as a function of position.

The EVEREST suite is one of the products of the ESPRIT project EVEREST (ESPRIT 962E-17, *Three-Dimensional Algorithms for a Robust and Efficient Semiconductor Simulator with Parameter Extraction*). EVEREST was a four-year project supported by the European Community under the European Strategic Program for Research in Information Technology (ESPRIT) which is investigating suitable algorithms for the analysis of semiconductor devices in three dimensions, and developing software implementing the most effective of those algorithms.

The original authors of the Doping Module were G.A. Duffett and M.S. Towers of University College, Swansea.

A copy of this report can be found at the Department's web site (*http://www.dci.clrc.ac.uk/*) under page *Group.asp?DCICSEMSW* or anonymous ftp server *www.inf.rl.ac.uk* under the directory *pub/mathsoft/publications*

Mathematical Software Group
Department for Computation and Information
Rutherford Appleton Laboratory
Chilton, DIDCOT
Oxfordshire OX11 0QX

Contents

1 Introduction

The doping module is responsible for generating a neutral file which gives the acceptor and donor densities at each node the device mesh. The neutral file also contains details of the functions used to generate the doping as mesh refinement in the solver requires this. Commands are used to set up a data base for the doping profile of the device. When everything is correctly set up the commands are executed creating the doping profile for the device. This information is then written to the Doping neutral file. The doping profile is computed by summing the contributions from the various parts of the doping definitions described by the user.

During the initial interactive set up phase for the data base, commands are provided for the input. The basic commands are:

TITLE - to specify a title for the problem

GEOMETRY - to specify input geometry neutral file

MESH - to specify input mesh neutral file

RECAP - to list the volumes and windows

BACKGROUND - to specify the uniformly doped background

UNIFORM - to specify a uniformly doped volume

PROCESS - to specify a doping process

PROFILE - to specify a doping profile

FUNCTION - to specify use of a user doping function

OUTPUT - to specify a doping neutral file

CHECK - to check and list the commands input

RUN - to begin execution of the program

END - to end program

The module is designed to be flexible so that entries can be checked and corrected at all stages before any calculations are made. Also, more than one neutral file may be generated in a single session by making the necessary changes to the data base of parameters before re-running. Some logic is included to prevent running of the doping calculations without a minimum of information being input. Essential items include specification of the Geometry and Mesh neutral files, and of the Output doping neutral file.

2 User Interface

The command decoder is initialised with all the syntax information for the command set of the module from a command file and then takes input from the terminal, or even a data file invoking individual commands, with their parameters.The modules use an interactive command decoder, developed at the Rutherford Appleton Laboratory, with an on-line help facility. This command environment allows you to input the doping information either interactively or by means of prepared data files which contain this data. The second mode of use is strongly recommended, especially if you are experimenting with doping levels.

The initial commands available to the user are given by the HELP command, as shown below. Full details of all the commands available in *doping* are given in the Appendixes.

```
Model: Help

The commands currently defined are:

Applications Commands

 TITLE       -  to specify a title for the problem
 GEOMETRY    -  to specify input geometry neutral file
 MESH        -  to specify input mesh neutral file
 RECAP       -  to list the volumes and windows
 BACKGROUND  -  to specify the uniformly doped background
 UNIFORM     -  to specify a uniformly doped volume
 PROCESS     -  to specify a doping process
 PROFILE     -  to specify a doping profile
 FUNCTION    -  to specify use of a user doping function
 OUTPUT      -  to specify a doping neutral file
 CHECK       -  to check and list the commands input
 RUN         -  to begin execution of the program
 END         -  to end program

Internal Commands

 MORE     -  to display the contents a file
 CHANGE   -  to change working directory
 RENAME   -  to rename a file
 COPY     -  to copy a file
 ERASE    -  to delete (remove) a file
 LIST     -  to provide directory listing
 WRITE    -  to provide monitoring of a session
 READ     -  to redirect the input stream to read from a file
 SYNTAX   -  to provide the syntax of a command
 HELP     -  to access HELP system

For further information type: HELP <command name> [<option>],
where <option> is BRIEF or FULL
```

All the commands can be typed in upper or lower case. The syntax of each command can be obtained by using the `syntax` command. For example

```
Geometry: syntax write

WRIte STAte=<choice> [,FIle=<string>] [,PROmpt=<choice>]

Geometry:
```

To get full details on a command and its parameters, such as the `WRITE` command, you can use `HELP WRITE`.

```
Model: help write

 Name     : WRITE
 Purpose  : to provide monitoring of a session
 Syntax   : WRIte STAte=<choice> [,FIle=<string>] [,PROmpt=<choice>]
```

```
Keyword     Type            Status        Current Value
-----------------------------------------------------------------

STATE       choice          required       on,off,close
FILE        string          retained       MONITOR
PROMPT      choice          reset          on,OFF
```

A command can be abbreviated, the shortest value being indicated by the uppercase letters in the syntax section, e.g. the `Write` command may be shortened to just `wri`. The system is reasonably simple to use and working through one or two of the examples below should enable one to get to grips with it.

Doping module provides a number of *internal* commands. These commands, such as HELP and COPY, provide standard information and file handling from within *doping*. As with all commands, details of their usage can be obtained through the HELP command. A summary of these commands is given in Appendix A.

Those commands that access the file store do so by invoking the appropriate system command of the operating system being used. This means that in general if an error report is produced it will be that of the host operating system.

For example, on UNIX systems, the RENAME command will use the UNIX command `mv`. Similarly `LIST` uses the UNIX command `ls`. Although the parameter types for these commands is *string*, the appropriate host systems file expression and options can be used provide any expression that contains spaces is contained within quotation marks. An example of this is:

```
    LIST "-l *.MSH"
```

On a UNIX system this command will list all files with extension `.MSH` in the current working directory.

3 Recommended Procedure for Using the Doping Module

The doping generation module commands are relatively easy to use. However, you must know in detail which named volumes correspond to which uniform doping values and which windows correspond to which implantation parameters. This is complicated by the fact that one physical window may comprise several named windows in the geometry, all of which have to be specified individually.

The *doping* program allows you to build up an impurity distribution for the device you wish to simulate. This distribution is part of the physical data, which together with a description of the geometry and finite element mesh, are required for simulation.

Two types of distribution are permitted:

- Constant doping (usually epitaxial or bulk semiconductor layers)
- Non-uniform doping (for example, diffused p or n-type wells).

The doping density is always in units of cm^{-3}. All geometric lengths and standard deviations (diffusion lengths) must be given in *micron* units. The non-uniform distributions are formed by implanting impurities through a window in a mask on the semiconductor surface and then diffusing the impurities into the semiconductor.

3.1 Specifying input and output files

Doping reads two input files and generates a single output file. The input files are the standard geometry and mesh neutral files produced by the pre-processor. The commands GEOMETRY, MESH and OUTPUT are those used. A typical example is:

```
GEOMETRY PN.GEO
MESH PN.MSH
OUTPUT PN.DOP
```

3.2 Background and Uniform Doping

The underlying impurity type of the substrate can be specified using the BACKGROUND command. BACKGROUND should be used to specify the acceptor and donor background doping.

```
BACKGROUND ACC=1.0D18 DON=1.0D12
```

The UNIFORM command is very similar to BACKGROUND save that it only applies to specific volumes. To specify the acceptor and donor uniform doping in a named volume the command

```
UNIFORM VOL=V1 ACC=1.0D18 DON=1.0D12
```

could be used. If several volumes have the same dopings, these can be set in the same command by including the names in a list inside round brackets and separated by commas.

```
UNIFORM VOL=(V1,V2,V4) ACC=1.0D18 DON=1.0D12
```

If a volume is referred to more than once a warning is issued and the previous values overwritten. This feature allows you to edit the parameters before committal to what may be a long execution time when the RUN command is used.

3.3 Process and Profile simulations

The PROCESS or PROFILE commands are used to specify the doping parameters to be applied through a named window. The PROCESS command uses predefined functions to compute the vertical and lateral doping distributions, while the PROFILE command makes use of a user-defined vertical profile with a choice of a predefined lateral distributions.

The PROCESS command provides three standard processes:

- predeposition,
- drive-in diffusion
- ion implantation.

Each process is described by the user in terms of constants that are used in the defined lateral and vertical functions used to compute the doping distribution. Densities are all in units of cm^{-3} and geometric lengths and deviations (diffusion lengths) are in *micron* units.

For a predeposition process, the vertical profile is computed using a complementary error function, and the lateral profile using a rotated ellipse. The drive-in diffusion profile is computed using a vertical Gaussian distribution with a rotated ellipse lateral profile. The ion implantation profile is computed using a vertical Gaussian distribution but with the peak below the surface; the lateral profile is computed using a complementary error function.

The windows through which the implant are made must be rectangular, although more complex shaped windows can be made up as a series of abutting rectangles.

An example of the PROCESS command is:

```
PROCESS TYPE=PREDEPOSITION, CONSTANTS=(2.6D18,3.D0,0.5),
        DOPTYPE=ACCEPTOR, WINDOW=WINDOW3, ACTION=ADD
```

which applies a predeposition of acceptor through a window WINDOW3.

The PROFILE command enables a user-defined vertical doping profile to be used in the computation of the doping distribution. The depths and concentrations of the one dimensional vertical profile are obtained from the neutral file which is named in the command. Concentrations must have units of cm^{-3} and depths must have *micron* units. The extension .PRO must not be included in the parameter. There is a choice of using either a rotated ellipse or a complementary error function to describe the lateral distribution. The lateral standard deviation constant, used with the complementary error function, must have *micron* units. No doping concentrations are computed outside the range of depths given in this vertical profile.

An example of the PROFILE command is:

```
PROFILE FILE=vertic, LATYPE=rotate, CONSTANT=0.5,
        DOPTPE=ACCEPTOR, WINDOW=wndw3, ACTION=ADD
```

An example of the contents of a *profile* file is given below:

```
$HEAD
 VER_03_87
$CASE
  Profile data on channel
$PROF
 23
-1.0000000E-04 2.9899778E+16
 1.5779767E-02 3.0109827E+16
 4.2018358E-02 3.0482244E+16
 6.3974053E-02 3.1023148E+16
 9.2864849E-02 3.2090907E+16
 0.1274796     3.3449031E+16
 0.1584477     3.4070101E+16
 0.1962833     3.3943749E+16
 0.2271500     3.3510052E+16
 0.2646408     3.2311666E+16
 0.3134241     3.1011025E+16
 0.3559420     2.9788237E+16
 0.3978915     2.9021569E+16
 0.4290017     2.8692063E+16
 0.4676223     2.8385366E+16
 0.4969734     2.8098602E+16
 0.5405805     2.7929309E+16
 0.5929564     2.7743608E+16
 0.6290532     2.7558050E+16
 0.6673896     2.7457054E+16
 0.7120115     2.7216817E+16
 0.7403207     2.7169882E+16
 1.5000000      2.7117668E+16
$END-PROF
$END-CASE
$END-HEAD
```

It should be noted that this is a fixed format file in the style of the general neutral files.

In both the PROCESS and PROFILE commands if several windows have the same parameters, these can be set in the same command by including the names in a list. If the ACTION parameter is REPLACE then the parameters of the last occurrence of that window are replaced. This feature allows the last entry to be corrected. If the ACTION parameter is ADD then a new entry is made to the data structure, allowing multiple implants to be made through the same window.

If several named windows make up one physical window then the parameters will only be updated if the newly input physical window corresponds with that already present in the data structure. The one restriction on the windows is that all named window must be rectangular. A physical window can thus be of any shape provided that its component named windows are all rectangular. If a window is referred to more than once a warning is issued.

3.4 User provided functions

If a user-programmed subroutine is to be used to compute additional impurity contributions the command FUNCTION must be used to indicate this to the controlling program. It is the responsibility of the user to ensure that no errors are present in this routine (called DOPEFN) and that the routine is correctly compiled and linked into the controlling main program. The Solver Module also requires any new doping functions linked into it.

3.5 Specification Validation

You are strongly recommended to use the CHECK command to confirm the correctness and completeness of the parameters set. The commands used and not used are listed, together with tables for background, uniform and non-uniform doping. The input files (geometry and mesh) and the output file (doping) are also displayed.

An example of the output from CHECK is given below.

```
Dope: CHECK
  Geometry filename is  CORNER.GEO
  Mesh     filename is  CORNER.MSH
  Output   filename is  CORNER.DOP

  The following COMMANDS have been used :

  *** No Title has been set ***

  BACKGROUND DOPING :   A=     0.     , D=  0.150E+18

  *** No Uniform doping has been set ***

  NON-UNIFORM DOPING, number of doping windows =   6
  PROCESS/PROFILE commands used   2 times
```

```
PROCESS command :
Window   Process   Dopant Constants
W1       Ion impl   Acc    0.1000E+19 0.1445E+01 0.1445E+01 0.0000E+00
W2       Ion impl   Acc    0.1000E+19 0.1445E+01 0.1445E+01 0.0000E+00
W3       Ion impl   Acc    0.1000E+19 0.1445E+01 0.1445E+01 0.0000E+00
W1       Ion impl   Acc    0.1000E+18 0.1200E+01 0.7850E+01 0.0000E+00
W2       Ion impl   Acc    0.1000E+18 0.1200E+01 0.7850E+01 0.0000E+00
W3       Ion impl   Acc    0.1000E+18 0.1200E+01 0.7850E+01 0.0000E+00

*** No PROFILE command used ***
```

The RUN command will not execute unless the CHECK command has been used at least once in the current session.

3.6 Profile generation

When all the parameters have been satisfactorily specified the RUN command is invoked. Only at this stage are the possibly lengthy calculations commenced and, if all goes well, the results will be written to the output doping neutral file. Control returns to the command decoder so that, at this stage, further neutral files can be created without ending and re-running the program. It may only be necessary to make minimal changes in the data base of the parameters. If a new geometry file is specified, however, the uniform and non-uniform doping data structures will be erased.

3.7 Program termination

When the task is finished the module is exited with the END command. This cannot normally take place if the RUN command has not been used. The action can be forced, however, by use of the CONFIRM parameter, set to Y.

4 Algorithms

All the commands, except RUN and END, either set up the doping data structures, read input from the terminal or a data file, or list information to the terminal for reference and checking purposes. A system of flags is used to cause reminders to be given if you have not set certain parameters, and also to prevent execution failures if the run command is invoked before essential filenames are supplied. As much information as possible, which is independent of the mesh nodes, is computed and stored when setting up the data structures. This allows the main doping calculations to proceed as efficiently as possible.

Inside the main loop over the mesh nodes, each node is taken in turn. Initially acceptor and donor doping are zero. Then, if set, background doping is added in. Next, if uniform doping is set, the volumes containing the node are identified. To ensure that interface nodes are included in all the relevant volumes, tolerances are added to the volume dimensions This expands them slightly so that such nodes will lie inside them all. Amongst the various uniform doping values available, the interface node is assigned the most significant acceptor/donor pair, according to the following rule:

> *If the current values of acceptor and donor concentration exceed both the new uniform-plus-background values then the current values are retained; otherwise the new values for donor and acceptor concentration are kept.*

Next each contribution is added in from the doped windows. In the present implementation there are three predefined doping distributions corresponding to the following doping processes:

- predeposition,
- drive-in diffusion,
- ion implantation.

Each of these processes uses the parameters supplied through the PROCESS command.

Often windows are to be built up using several panes. This can arise when a non-rectangular window is required and the procedure is to dissect the shape into adjacent rectangular components. Also the need can arise when a physical window spans more than one volume. Since the geometric description restricts windows to lie entirely within a given surface, these types must be decomposed into segments unique to each surface coincident with the original window. The geometric representation does not distinguish between a window and a window pane, and consequently the difference is communicated to the doping module by ensuring that all panes of the same window are included in a list when the appropriate command is used. Windows containing common edges are put into groups so that the non-common window edges become the boundaries of the physical window. When computing the doping at a node, the nodal position with respect to the physical window is determined (i.e. inside the window, external to a boundary or external to a corner) and hence the correct doping for that position can be found.

Finally, the doping contributions computed by the user-programmed routine as a function of nodal co-ordinates are included into the total impurity profile.

5 Examples Session

5.1 A One-Sided *n-p* Step Junction

This doping definition can be used with the example given in the GEOMETRY program.

```
GEOM NP
MESH INI
RECAP
UNIF N+SI, 3.0D14, 1.0D19
UNIF P_SI, 5.0D15, 1.0D14
OUT NP
CHECK
RUN
END
```

5.2 A *P*-Channel Double-Diffused MOSFET

```
geo file=DMOS
mesh file=DMOS
uniform vol=v1, a=3.0D15, d=2.5D12
process predep cons=(2.6D18,3.0,0.5) doptype=d wind=nimpl
process drive cons=(1.0D19,6.0,0.75) a (pimplant,window2)
CHECK
```

```
OUTPUT FILE=DMOS
RUN
END
```

5.3 Combined user defined profiles

```
geom NP
mesh file=ini
output FILE=NP
check
profile myfile rot 0.5 acc (window1,window2)
FUNCTION
run
END
```

A Internal Commands

A.1	MORE	to display the contents a file
A.2	CHANGE	to change working directory
A.3	RENAME	to rename a file
A.4	COPY	to copy a file
A.5	RM	to delete (remove) a file
A.6	LIST	to provide directory listing
A.7	WRITE	to provide monitoring of a session
A.8	READ	to redirect the input stream to read from a file
A.9	SYNTAX	to provide the syntax of a command
A.10	HELP	to access HELP system

A.1 MORE - to display the contents a file

Syntax

```
MORe FILe=<string>
```

Description

Displays the contents of the specified file in the current window.

Parameters

FILE	Required *string* A *string* giving the name of the file to be displayed.

Examples

```
more file=ANODE
MOR CATHODE
```

A.2 CHANGE - to change working directory

Syntax

```
CHAnge DIRectory=<string>
```

Description

Changes the current working directory.

Parameters

DIRECTORY	Required *string* A *string* giving the name of the new working directory.

Examples

```
change directory=results
CHA MODELS
```

A.3 RENAME - to rename a file

Syntax

```
REName FILe1=<string> FILe2=<string>
```

Description

Renames a given file to a new name.

Parameters

FILE1	Required *string* A *string* giving the current file name.
FILE2	Required *string* A *string* giving the new file name.

Examples

```
RENAME FILE1=RESULT1 FILE2=RESULT.SAVE
ren output1 output2
```

A.4 COPY - to copy a file

Syntax

```
COPy FILe1=<string> FILe2=<string>
```

Description

Copies a given file to a new file.

Parameters

FILE1	Required *string* A *string* giving the source file name.
FILE2	Required *string* A *string* giving the destination file name.

Examples

```
COPY FILE1=RESULT1 FILE2=RESULT.SAVE
cop output1 output2
```

A.5 RM - to delete (remove) a file

Syntax

```
RM FILe=<string>
```

Description

Removes (deletes) the given file from the file system.

Parameters

FILE	Required *string* A *string* giving the name of the file to be removed (deleted).

Examples

```
RM FILE=RESULT
rm output
```

A.6 LIST - to provide directory listing

Syntax

```
LISt [FILe=<string>]
```

Description

Provide a listing of the current or specified directory.

Parameters

FILE	Optional *string* : initial = " " A *string* giving the name of the file.

Examples

```
LIST
lis *.MSH
```

A.7 WRITE - to provide monitoring of a session

Syntax

```
WRIte STAte=<choice> [FIle=<string>]
      [PROmpt=<choice>]
```

Description

Redirects the command decoder echo output to the file specified by the FILE parameter. The information flow is controlled by the STATE parameter. This command can enable the constructions of command files to drive the program in a background mode.

The echoing of the command prompt can be controlled using the PROMPT parameter.

Parameters

STATE

required *choice*
Controls the flow of information to the monitoring file It has values NO, OFF or CLOSE. ON switches on monitoring. OFF suspends it but does not close the file and CLOSE ends monitoring and closes the file.

FILE

retained *string* : initial = MONITOR
Output file name to receive the monitoring stream.

PROMPT

reset *choice* : initial = OFF
Allows you to select whether the command prompt is echoed in the monitoring file. It has values ON or OFF.

Examples

```
WRITE STATE=ON FILE=MONITOR PROMPT=OFF
wri on junk on
```

A.8 READ - to specify a command input file

Syntax

```
REad FIle=<string> [ECHO=<choice>]
```

Description

Redirects the command decoder to take its input from a file specified by the FILE parameter. However, if FILE is given as TERMINAL, input returns to the standard input stream.

The echoing of the commands being read by the decoder can be controlled using the ECHO parameter.

Parameters

FILE	required *string* Input file name containing program commands.
ECHO	reset *choice* : initial = OFF Echo control option. Values can be ON or OFF.

Examples

In this example a sequence of commands are read from the file NAIL and ECHOed to the standard output device:

```
Everest: read nail echo=on
Everest: GEO * >DATA>NAIL.GEO
Everest: MES * >DATA>OXNAIL.MSH
Everest: DOP * >DATA>NAIL.DOP
Everest: RES NAIL.OUT R
Everest: PHY NAIL.PHY
Everest: BIAS LEFT-CNT 0
Everest:
```

A.9 SYNTAX - to provide the syntax of a command

Syntax

```
SYNtax [COMmand=<string>]
```

Description

Displays the formal syntax of all the currently defined commands. If the syntax of a specific command name is required then that name is given as a parameter to the command.

Parameters

COMMAND	retained *string* : initial = ALL Specifies the commands name for which the syntax is required. If the syntax of all the currently defined commands is required, then the special command name ALL should be used.

Examples

The following example obtains the syntax of all the commands in the DOCUMENT program.

```
Doc: syntax

The commands currently defined are:

SYNtax [COMmand=<string>]
Help [KEY=<string>] [OPTion=<string>]
```

```
FILe INput=<string>, OUTput=<string>
PROcess
Quit
TItle TEXT=<string>
AUthor TEXT=<string>
Date TEXT=<string>
OPtions [SORT=<choice>] [CONtents=<choice>]
        [RUNoff=<choice>] [FRont_page=<choice>]

 For further information type: HELP <command name> [<option>],
 where <option> is BRIEF or FULL

 Doc:
```

A.10 HELP - to access HELP system

Syntax

```
Help [KEY=<string>] [OPTion=<choice>]
```

Description

Accesses to the inbuilt HELP system within the command decoder. HELP is one of the internal commands of the command processor and has a companion command SYNTAX.

HELP has two parameters allowing the selection of help on a specific command and the level of help required (SUMMARY, BRIEF, FULL and SYNTAX). If no command name is given summary help is given on all the commands currently defined.

If an ambiguous or invalid command name is given a warning or error message is given.

BRIEF help gives information on the purpose, syntax and the current state of the selected command. A table of command keywords, their type, status and current value (if applicable) is printed.

When the FULL option is used the Help System uses the inbuilt free text retrieval system to access the help data base. This allows the display of the full command description and the searching for specific keywords. This option is not supported in the current release.

Parameters

KEY — reset *string* : initial =
Either the global command name SUMMARY, or the specific command name on which help is sought.

OPTION — reset *choice* : initial = BRIEF
The level of help required. This can be SUMMARY, BRIEF, FULL or SYNTAX.

Examples

```
Everest: help output

Name     : OUTPUT

Purpose  : to specify results file
Syntax   : OUTput FILE=<string> [REPLACE=<choice>]


Keyword      Type             Status        Current Value
-----------------------------------------------------------------

FILE         string           required       undefined
REPLACE      choice           reset          replace,NOREPLACE
```

B Command Reference Section

B.1	BACKGROUND	to specify the uniformly doped background
B.2	CHECK	to list and check the commands already typed
B.3	END	to end the program
B.4	FUNCTION	to specify the use of a user doping function
B.5	GEOMETRY	to provide the geometric data
B.6	RECAP	to list volumes and windows
B.7	MESH	to provide the mesh data
B.8	OUTPUT	to specify the doping (output) neutral file
B.9	PROCESS	to specify a standard doping process
B.10	PROFILE	to specify a user-defined vertical profile
B.11	RUN	to run the main routine
B.12	TITLE	to specify a title for the problem
B.13	UNIFORM	to specify a uniformly doped volume

B.1 BACKGROUND - to specify the uniformly doped background

Syntax

```
BACKground  Acceptor=<real> Donor=<real>
```

Description

Specifies the acceptor and donor densities of the uniformly doped background region. This command is used to avoid having to type in the same doping values for many volumes.

Parameters

ACCEPTOR

required *real* in cm^{-3} : always positive
Typical values are in the range 1×10^{14} to $1 \times 10^{19} cm^{-3}$.

DONOR

required *real* in cm^{-3} : always positive
Typical values are in the range 1×10^{14} to $1 \times 10^{19} cm^{-3}$.

Examples

```
BACK A=1.0D18, D=1.0D12
back 1.0D12, 1.6D14
```

B.2 CHECK - to list and check the commands already typed

Syntax

```
CHECK
```

Description

Lists the commands input so far. If the number of specified volumes exceeds the number of UNIFORM commands when the BACKGROUND command has not been used, it indicates which volumes have not had doping values assigned to them. If the geometry file, mesh or output file has not been specified, it issues a warning to this effect. It can be used at any stage before the RUN command to verify commands and parameters already typed in. It is recommended that this command is used at least once before running the main routine with the RUN command.

Parameters

This command has no parameters.

Examples

```
CHECK
```

B.3 END - to end the program

Syntax

```
END    [Confirm=<choice>]
```

Description

Halts the program. Normally this cannot happen if the RUN command has not been used. If you really want to exit, this can be forced by setting the Confirm parameter to Y.

Parameters

Confirm	an optional *choice* : initial = NO Confirmation of ending. Values are (Yes, No).

Examples

```
END
END Y
```

B.4 FUNCTION - to specify the use of a user doping function

Syntax

```
FUNCtion NUmber=<integer>
```

Description

A number of different doping functions can be selected by the use of the integer value NUMBER. This value is passed to the the embedded routine DOPEFN. Specifies the use of a user-defined (and programmed) routine called via DOPEFN which is used to compute doping contributions as a function of the position co-ordinates (x, y, z). The user may add any FORTRAN coding to the routine DOPEFN in order to compute doping contributions at the nodes in the geometry of the device being modelled. Real-valued functions must however, be provided for both the acceptor and the donor densities (units of cm^{-3}). It must be noted that since the doping is computed node by node this routine must provide a doping contribution for *every* node in the mesh. Obviously these nodal contributions can be set to zero in certain regions of the device as required. Further details can be found in the preamble to the routine DOPEFN.

Note that changes to DOPEFN must also be include in the Solver Module.

The user must ensure that no errors exist in the routine and that it is properly compiled and linked into the main program (see local user instructions for details).

Parameters

NUMBER	required *integer* A *integer* value specifying the which of the available doping functions should be used..

Examples

```
FUNCTION 1
func  2
```

B.5 GEOMETRY - to provide the geometric data

Syntax

```
GEOmetry   FILE=<string>
```

Description

Specifies the neutral file from which the geometry information is to be taken. The information will be read using the neutral file reading routines. The extension .GEO must not be included in the parameter and it must not exceed 20 characters in length.

Parameters

FILE	required *string* A *string* value specifying the neutral file name.

Examples

```
GEO FILE=PN
geo mos
```

B.6 RECAP - to list volumes types and windows

Syntax

```
RECap
```

Description

Lists the volumes and windows that have been defined in the geometry neutral file. One UNIFORM command is expected to be input for each volume unless the BACKGROUND command has been used. There may be more windows than implant-diffusion steps as the contacts can be specified as windows.

Parameters

This command has no parameters.

Examples

```
RECAP
```

B.7 MESH - to provide the mesh data

Syntax

```
MESh FILE=<string>
```

Description

Specifies the neutral file from which the mesh information is to be taken. The information will be read using the neutral file reading routines. The extension .MSH must not be included in the parameter and it must not exceed 20 characters in length.

Parameters

FILE	required *string* A *string* value specifying the neutral file name.

Examples

```
MES FILE=FINE
mesh ini
```

B.8 OUTPUT - to specify the doping (output) neutral file

Syntax

```
OUTput  FILE=<string>
```

Description

Specifies the neutral file to which the nodal values of the acceptor and donor concentrations will be output. The extension .DOP must not be included in the parameter and it must not exceed 20 characters in length. The information will be written using the neutral file writing routines. This command must be used before the RUN command.

Parameters

FILE	required *string* A *string* value specifying the neutral file name.

Examples

```
OUTPUT FILE=PN
output mos
```

B.9 PROCESS - to specify a standard doping process

Syntax

```
PROCess TYPe=<choice> CONstants=<rlist> DOPtype=<choice>
        WINdow=<slist> [ACTion=<choice>]
```

Description

Specifies a standard doping process and the names of the windows for this doping. There are three standard processes:

- predeposition,
- drive-in diffusion
- ion implantation.

Each process is described by the user in terms of constants that are used in the defined lateral and vertical functions used to compute the doping distribution. Densities are all in units of cm^{-3} and geometric lengths and deviations (diffusion lengths) are in *micron* units.

For a predeposition process, the vertical profile is computed using a complementary error function, and the lateral profile using a rotated ellipse. The drive-in diffusion profile is computed using a vertical Gaussian distribution with a rotated ellipse lateral profile. The ion implantation profile is computed using a vertical Gaussian distribution but with the peak below the surface; the lateral profile is computed using a complementary error function.

Windows must be rectangular. Odd shaped windows can be made up as a series of abutting rectangles and must be included in the same string list when the command is used.

It is recommended that no more than three implants share a common window. This is to avoid errors in the calculation of the impurity distribution since the solution assumes non-interacting impurities.

Parameters

TYPE

required *string*
A *string* which gives the name of the process required values are (PREdeposition, DRIvein, IONimplantation).

CONSTANTS

required *reallist*
A list of positive constants required for the computation of the doping profile. The number and meanings of the constants are different for each of the processes. The constants required are:
(1) predeposition (3 constants):
N_0 - solid solubility limit at the surface
σ_A - diffusion length = $\sqrt{Dt}$
α - parameter describing ellipse
(2) drive-in diffusion (3 constants):
N_t - total impurity per unit surface area
σ_A - diffusion length = $\sqrt{Dt}$
α - parameter describing ellipse

(3) ion implantation (4 constants):
N_p - peak impurity concentration (when $\sigma_B = 0$)
x_p - depth of peak below surface
σ_A - vertical standard deviation
σ_B - lateral standard deviation

DOPTYPE

required *choice*
The type of doping. Values are (Acceptor, Donor).

WINDOW

required *stringlist*
The list of window names for which this doping is applied.

ACTION

optional *choice* : initial = ADD
The ACTion, if a window has already been doped is either Add or Replace. Values are (Add, Replace)

Examples

```
PROC TYP=pred, CON=(2.6D18,3.D0,0.5), DOPT=a, WIND=(wndw3,wndw4)
proc ION (2.D16,1.2D0,3.,2.) a (wl,w2,wel4,w5) add
```

B.10 PROFILE - to specify a user-defined vertical profile

Syntax

```
PROFile FILe=<string> LAType=,choice> CONstant=<real>
        DOPtpe=<choice> WINdow=<slist> [ACTion=<choice>]
```

Description

This enables a user-defined vertical doping profile to be used in the computation of the doping distribution. The depths and concentrations of the one dimensional vertical profile are obtained from the neutral file which is named in the command. Concentrations must have units of cm^{-3} and depths must have *micron* units. The extension .PRO must not be included in the parameter and it must not exceed 20 characters in length. There is a choice of using either a rotated ellipse or a complementary error function to describe the lateral distribution. The lateral standard deviation constant, used with the complementary error function, must have *micron* units. No doping concentrations are computed outside the range of depths given in this vertical profile.

Windows must be rectangular. Odd shaped windows can be made up as a series of abutting rectangles and must be included in the same string list when the command is used.

It is recommended that no more than three implants share a common window. This is to avoid errors in the calculation of the impurity distribution since the solution assumes non-interacting impurities.

Parameters

FILE

required *string*
A *string* which gives the name of the file in which the profile is stored (the extension .PRO is assumed).

LATYPE

required *choice*
The type of lateral profile required values are (ROTate, ERFc).

CONSTANT

required *real*
The positive constant required for the computation of the lateral doping profile. The meaning of the constant is different for each of the lateral profile types. The constants required are:
(1) rotated ellipse:
α - parameter describing ellipse
(2) complementary error function:
σ_B - lateral standard deviation

DOPTYPE

required *choice*
The type of doping. Values are (Acceptor, Donor).

WINDOW

required *stringlist*
The list of window names for which this doping is applied.

ACTION

optional *choice* : initial = ADD
The ACTion, if a window has already been doped is either Add or Replace. Values are (Add, Replace).

Examples

```
PROF FIL=vertic LAT=rotate CON=0.5 DOPT=a WIND=wndw3
prof myfile erfc 3.0 a (w01,w02,wnd4,wnd5) add
```

B.11 RUN - to run the main routine

Syntax

```
RUN
```

Description

Causes the main routine to commence calculating the nodal values of doping density, and to output a doping neutral file.

Parameters

This command has no parameters.

Examples

```
RUN
```

B.12 TITLE - to specify a title for the problem

Syntax

```
TITLE   TEXT=<string>
```

Description

Specifies the title to be written into the output doping neutral file for identification purposes. It must not exceed 20 characters in length.

Parameters

TEXT	required *string* A *string* value specifying the title in the output doping neutral file.

Examples

```
TITLE TEXT=Diode
TITLE MOSFET
```

B.13 UNIFORM - to specify a uniformly doped volume

Syntax

```
UNIForm  VOLume=<slist> Acceptor=<real> Donor=<real>
```

Description

Specifies the acceptor and donor densities in each volume. To avoid having to type in the same doping values for many volumes, the BACKGROUND command can often be used to advantage.

Parameters

VOLUME	required *stringlist* A *string* value specifying the volume name or names.
ACCEPTOR	required *real* in cm^{-3} : always positive Typical values are in the range 1×10^{14} to $1 \times 10^{19} cm^{-3}$.
DONOR	required *real* in cm^{-3} : always positive Typical values are in the range 1×10^{14} to $1 \times 10^{19} cm^{-3}$.

Examples

```
UNIF VOL=P+SI, A=1.0D18, D=1.0D12
unif n-si, 1.0D12, 1.6D14
```