Prophet User Reference Guide

Mixed Technology Associates
Palo Alto, California

July 1, 2002

1 Introduction

This document describes the use of Prophet for device simulation applications. Section 2 gives an overview of the Prophet input file structure, and lists the main systems of semiconductor equations that are currently available. Section 3 describes the models and model parameters available. Section 4 is a reference for the commands and keywords used in Prophet input files. Section 5 lists simulation options that are commonly used. Section 6 gives detailed explanations of the available pre-defined systems of equations. Section 7 describes Prophet's built-in operators (for users interested in developing their own equation systems).

2 Prophet Overview

2.1 Prophet Input File

Prophet is quite flexible, and for specialized applications the input files may be quite varied. However, for typical mainstream device simulation applications, the input file usually consists of the following main sections:

- **Include files:** Include files are typically used to access pre-defined, pre-tested systems of equations. Problems that do not need specialized physics should use these pre-defined systems for convenience.
- **Substrate grid and deposited layers:** The extent of the semiconductor substrate is defined, along with the vertical and horizontal grid spacing. Following that, layers above the substrate (e.g., oxide or poly) are defined with "deposit" commands.

- **Doping profile:** The doping profile is defined either by analytic expressions (e.g., Gaussian or constant) or by numerical profiles read in from a file.
- **Electrodes:** The location of the device electrodes are defined with “boundary” commands.
- **Initial solution:** The initial solution at zero applied bias is obtained using a “bias” command.
- **I-V curves:** The applied bias on the device electrodes are swept over the desired range using “bias” commands.
- **Save output:** The results of the simulation can be be displayed (using “graph” commands) or saved for later analysis (using “graph” or “save” commands).

The Prophet input file on the next two pages illustrates these main components listed above.

Example Prophet input file: MOSFET with polygate (continued on next page):

```
#
# Include statements define systems of equations to solve
#
include(silicon_poisson)
include(silicon_dd_lombardi)
#
# Define substrate grid, deposited layers
#
grid dim=2
+ xloc=0.00000,0.0010,0.010,0.10,0.4;
+ xdel=0.00005,0.0005,0.001,0.01,0.05;
+ yloc=-0.1,-0.065,-0.06,-0.030,-0.020,0.0,0.020,0.030,0.06,0.065,0.1
+ ydel=0.01,0.005,0.0025,0.0025,0.005,0.005,0.005,0.0025,0.0025,0.005,0.01
deposit mat=oxide thick=0.002 xdel=0.0005
deposit mat=poly start=-0.045 end=0.045 thick=0.0,0.001,0.01,0.1
+ xdel=.0001,.0005,0.001,0.02
deposit mat=oxide start=-0.060 end=-0.045 thick=0.0,0.001,0.01,0.1
+ xdel=.0001,.0005,0.001,0.02
deposit mat=oxide start=0.045 end=0.060 thick=0.0,0.001,0.01,0.1
+ xdel=.0001,.0005,0.001,0.02 zipper
#
# Define doping profile
#
field set=nstip val=1e20*gbox(X,0,0.01,0.003)*gbox(Y,-0.1,-0.042,0.003) mat=silicon
field set=ndtip val=1e20*gbox(X,0.0,0.01,0.003)*gbox(Y,0.042,0.1,0.003) mat=silicon
field set=ns val=1e20*gbox(X,0.0,0.018,0.008)*gbox(Y,-0.1,-0.08,0.008) mat=silicon
field set=nd val=1e20*gbox(X,0.0,0.018,0.008)*gbox(Y,0.08,0.1,0.008) mat=silicon
field set=npoly val=1e20 material=poly
field set=nsubstrate val=-1e18 material=silicon
field set=netdope val=npoly+nsubstrate+ns+nd+nstip+ndtip
#
# Define electrodes
#
boundary xmin=0.01 xmax=0.03 ymin=-0.1 ymax=-0.1 name=source
boundary xmin=0.01 xmax=0.03 ymin=0.1 ymax=0.1 name=drain
boundary xmin=-0.102 xmax=-0.102 ymin=-0.045 ymax=0.045 name=gate
boundary xmin=0.4 xmax=0.4 ymin=-0.1 ymax=0.1 name=back
```

Example Prophet input file: MOSFET with polygate (continued):

```
#
# Initial solution at zero bias
#
bias initial system=silicon_poisson
bias system=silicon_dd_lombardi elec=gate voltage=0.0
#
# I-V curves
#
bias system=silicon_dd_lombardi elec=gate vstep=0.1 nstep=6
bias system=silicon_dd_lombardi elec=drain vstep=0.1 nstep=12
bias system=silicon_dd_lombardi elec=gate vstep=0.1 nstep=6
bias system=silicon_dd_lombardi elec=drain vstep=-0.1 nstep=12
#
# Save I-V data to file in column format
#
graph iv outfile=mos.iv
```

2.2 List of Prophet Systems

The basic pre-defined model systems currently supported in Prophet are listed below; more detailed explanations are available in the Prophet Systems section. The first three systems cover most mainstream applications (classical drift-diffusion):

- **silicon_poisson:** Solve Poisson equation only (no current flow). Used for initial solutions.
- **silicon_dd:** Solve Poisson equation and two current continuity equations (electrons and holes). Uses Arora dopant dependent mobility model and longitudinal field mobility reduction by default.
- **silicon_dd_lombardi:** Same as silicon_dd, but with Lombardi transverse field mobility model.
- **silicon_dg5:** Five equation density gradient system (quantum corrections, current flow). Uses Arora dopant dependent mobility model and longitudinal field mobility reduction by default.
- **silicon_dg_lombardi:** Same as above, but with Lombardi transverse field mobility model.
- **silicon_poisson_ex:** Same as silicon_poisson, but also calculates vertical E-field. Useful for 1-D MOS capacitors, for example.
- **silicon_dd_ex:** Same as silicon_dd, but with calculation of transverse electric field.
- **silicon_dg3:** Three equation density gradient system (quantum corrections, no current flow).
- **silicon_dg3_ex:** Three equation density gradient system (no current flow). Useful for 1-D MOS capacitors, for example.
- **silicon_dg5_ex:** Same as silicon_dg5, but with calculation of transverse electric field.

3 Prophet Models

3.1 Doping Dependent Mobility Model (Arora)

An empirical low field mobility model based on measurement data for silicon at various temperatures. The model has the following general form:

$$\mu_0(N,T_L) = \mu_{min} + \frac{\mu_{dlt}}{1+(N/N_0)^{\alpha}} \tag{1}$$

where N is the net doping concentration. All the parameters are functions of T_L in the form of $a(T_L/300)^p$ (T_L in K), for example:

$$\mu_{min}(T_L) = \mu_{min0}(T_L/300)^p = \texttt{umin.0} * (\texttt{tl}/300)^{\texttt{umin.p}} \tag{2}$$

with the last expression on the right using the Prophet database parameter names.

Conventional usage expresses mobility in units of $cm^2/(V-sec)$; however, Prophet uses units of $um^2/(V-sec)$, so the numerical values of the mobility parameters appear to be 10^8 larger than usual.

The following table gives the parameter names and default values (note that the ".p" parameters are the temperature exponents):

Table 1: Doping Dependent Mobility Parameters and Default Values

	Prophet parameter name	units	silicon	
			electrons	holes
$\mu_{min}(T_L = 300K)$	umin.0	$um^2/(V \cdot sec)$	88.0*1e8	54.3*1e8
	umin.p		-0.57	-0.57
$\mu_{dlt}(T_L = 300K)$	udlt.0	$um^2/(V \cdot sec)$	1252.0*1e8	407.0*1e8
	udlt.p		-2.33	-2.23
$N_0(T_L = 300K)$	nref.0	cm^{-3}	1.25e17	2.35e17
	nref.p		2.4	2.4
$\alpha(T_L = 300K)$	alpha.0		0.88	0.88
	alpha.p		-0.146	-0.146

3.2 Longitudinal Field Mobility Reduction Model (Vsat)

For high fields which are parallel to the current flow, the effects of velocity saturation must be taken into account. A general approach is to multiply the low-field mobility (e.g., Arora and/or Lombardi model) by a reduction factor which takes into the consideration the high longitudinal (parallel) fields:

$$\mu(N, T_L, E_\perp, E_\parallel) = r_\parallel(E_\parallel)\mu_0(N, T_L, E_\perp) \tag{3}$$

for which μ_0 is the low field mobility, and $r_\parallel$ is the high field longitudinal reduction factor. The form of $r_\parallel$ used for silicon is:

$$r_\parallel = \left[1 + \left(\frac{\mu_0(N, T_L, E_\perp)E_\parallel}{v_{sat}}\right)^\beta\right]^{-1/\beta} \tag{4}$$

The temperature dependence of v_{sat} is given by:

$$v_{sat}(T_L) = v_{sat}(T_L = 0K) + c_1 * T_L \tag{5}$$

Table 2 summarizes the Prophet parameter names and default values for the longitudinal mobility reduction model.

Table 2: Longitudinal Field Mobility Reduction Model Parameters and Default Values

	Prophet parameter name	units	silicon	
			electrons	holes
$beta$	beta		1.395	1.215
$v_{sat}(T_L = 0K)$	vsat.c0	cm/sec	8.64e6	10.65e6
c_1	vsat.c1	$cm/(sec \cdot T(K))$	-2.68e3	-2.68e3

3.3 MOSFET Lombardi Low Field Mobility Model

Lombardi's mobility model accounts for the effect of the transverse electric on mobility in a MOSFET device. Lombardi's formulation adds terms for surface acoustic phonon (μ_{ac}) and surface roughness scattering (μ_{sr}) to the bulk mobility (μ_b) using a Matthiesen-like rule:

$$\frac{1}{\mu_{lomb}} = \frac{1}{\mu_{ac}} + \frac{1}{\mu_b} + \frac{1}{\mu_{sr}} \tag{6}$$

The bulk mobility (μ_b) can be from a variety of formulations, typically the Arora model in Prophet. Lombardi gives the following forms for μ_{ac} and μ_{sr}:

$$\mu_{ac}(N, T_L, E_\perp) = \frac{B}{E_\perp} + \frac{\alpha N^\beta}{T_L E_\perp^{1/3}} \tag{7}$$

$$\mu_{sr}(E_\perp) = \frac{\delta}{E_\perp^2} \tag{8}$$

Table 3 summarizes the Prophet parameter names and default values for the Lombardi mobility model. Conventional usage expresses mobility in units of $cm^2/(V-sec)$; however, Prophet uses units of $um^2/(V-sec)$, so the numerical values of the mobility parameters appear to be 10^8 larger than usual.

Table 3: Lombardi Mobility Model Parameters and Default Values

	Prophet parameter name	units	silicon	
			electrons	holes
B	lombardi_B	$um^2/(V \cdot sec)$	4.75e7*1e8	9.93e7*1e8
α	lombardi_alpha	$um^2/(V \cdot sec)$	1.74e5*1e8	8.84e5*1e8
β	lombardi_beta		0.125	0.0317
δ	lombardi_delta	$um^2/(V \cdot sec)$	5.82e14*1e8	2.05e14*1e8

4 Prophet Commands

4.1 Prophet Command Syntax

The basic syntax of a Prophet command is:

```
commandname keyword=value keyword=value keyword=value
```

Examples:

```
bias system=silicon_poisson initial
bias system=silicon_dd nstep=5 vstep=0.1 elec=anode
```

Some characteristics of the input syntax:

- No spaces are allowed in the values unless the value is quoted.
- Spaces are allowed between the "=" sign and the keyword and value.
- A keyword may be abbreviated up to its shortest unique abbreviation (unique for its command).
- A logical keyword by itself (with no "=" and no value given) specifies a logical "true."
- A command is extended to two or more lines by placing a "+" as the first non-blank character in the second and subsequent lines.
- A comment is preceded by a "#"; following a "#", everything on the line is ignored.
- Prophet pre-processes the input file with M4 macro processor before execution, so M4 commands may optionally be used in the input file.

The keyword value types are defined below:

logical

> A logical value is either true or false. A "true" value may be specified by using the keyword by itself (e.g., `initial`), or equating the keyword to "t" or "1" (`initial=t` or `initial=1`). "False" is specified by setting the keyword equal to "f" or "0" (`initial=f` or `initial=0`).

real

> A real floating point value. Formats such as `140.0`, `140`, and `1.4e2` are all accepted. Expressions are also allowed (see section on vexpr).

real array

A real array is one or more real values separated by commas, e.g., `vstep=0.1,0.2,0.1`. No spaces are allowed, unless the array is quoted.

integer

An integer value, e.g., `nstep=5`. Expressions are allowed.

integer array

An integer array is one or more integer values separated by commas. No spaces are allowed unless the array is quoted.

string

A string of characters, e.g., `system=silicon_dd`.

string array

A string array is one or more strings separated by commas. No spaces are allowed between strings. Quotes are not needed. Example: `electrode=gate,drain`.

4.2 BIAS

The bias command carries out steady state device analysis.

- SYSTEM = (s)
- INITIAL = (l)
- ELECTRODE = (s)
- NSTEPS=(i)
- VOLTAGE=(r)
- VSTEP=(r)
- TIME=(r)
- TEMPERATURE=(r)

The following list itemizes the valid keywords, their units and their meaning.

SYSTEM: [string]

What system of equations to use. One set must be chosen. Currently supported: newton0, newton1e, newton1h, newton2, and any user-defined system of equations

INITIAL: [logical]

If the INITIAL solution is chosen, all biases are reset to zero, the potential is set to the charge

neutral solution and the carriers adjusted accordingly. Then the system of equations is solved. Other bias parameters are ignored.

ELECTRODE: [string or string array]

The BIAS command allows one applied bias at a time to be modified. The ELECTRODE parameter defines which electrode is being modified. The string refers to a name defined either on a boundary command or read from a previous grid. More than one electrode can be specified using a string array, e.g.,

```
electrode=gate,drain
```

NSTEPS: [integer]

The number of bias increments to be taken. May be omitted if there is just one.

VSTEP: [real or real array (volts)]

The voltage step to be taken for each bias step. If the bias values of multiple electrodes are being changed, use a real array, e.g.,

```
vstep=0.5,1.0
```

where there is one value for each electrode value being changed.

TIME: [real (seconds)]

The time interval for a transient solution.

VOLTAGE: [real or real array (volts)]

If a single step is desired, specify VOLTAGE instead of VSTEP. If the bias values of multiple electrodes are being changed, use a real array, e.g.,

```
voltage=0.5,1.0
```

where there is one value for each electrode being changed.

TCELSIUS: [real (degrees Celsius)]

The temperature at which the simulation is performed. Default is 26.85C (300K).

TKELVIN: [real (degrees Kelvin)]

The temperature at which the simulation is performed. Default is 300K.

VINIT: [real or real array (volts)]

Initial voltage of a voltage ramp. Used in one of these combinations: vinit, vfinal, nstep; or vinit, vstep, nstep. If the bias values of multiple electrodes are being changed, use a real array, e.g.,

```
vinit=0.5,1.0
```

where there is one value for each electrode being changed.

VFINAL: [real or real array (volts)]

Final voltage of a voltage ramp. Used in one of these combinations: vinit, vfinal, nstep; or vinit, vstep, nstep.

If the bias values of multiple electrodes are being changed, use a real array, e.g.,

```
vinit=0.5,1.0
```

where there is one value for each electrode being changed.

Examples:
First bias statement needs the "initial" keyword:
```
bias system=silicon_poisson initial
```
To solve for a particular bias value:
```
bias system=silicon_dd voltage=0.5 elec=anode
```
To solve for a series of bias values:
```
bias system=silicon_dd vstep=0.1 nstep=20 elec=gate
```
To step biases on two electrodes at once:
```
bias system=silicon_dd vinit=0.1,0.1 vfinal=0.5,0.5 nstep=4 elec=gate,drain
```

4.2.1 Less Used BIAS Keywords

MIN.BACKSTEP

If a given bias step does not converge, the size of the voltage step will be reduced until it does. However, "MIN.BACKSTEP" is the minimum fraction of the original voltage step that is allowed before the BIAS command terminates.

TEMPERATURE: [real (degrees celsius)]

The temperature at which the simulation is performed. Default is 26.85C (300K).

VOLT.EXPR: [string or string array]

Voltage expression.

FSTHARMONICS

Harmonic balance analysis parameter.

SNDHARMONICS

Harmonic balance analysis parameter.

FSTFREQUENCY

Harmonic balance analysis parameter.

SNDFREQUENCY

Harmonic balance analysis parameter.

FSTAMPLITUDE

Harmonic balance analysis parameter.

SNDAMPLITUDE

Harmonic balance analysis parameter.

4.3 BOUNDARY

The boundary command allows sections of the structure surface to be redefined. This is convenient for creating device contacts.

- NAME=(s)
- [XMIN=(r)]
- [XMAX=(r)]
- [YMIN=(r)]
- [YMAX=(r)]
- [ZMIN=(r)]
- [ZMAX=(r)]

The following list itemizes the valid keywords, their units and their meaning.

NAME: [string]

This is the name for the new section of the boundary.

XMIN: [real (microns)]
XMAX: [real (microns)]
YMIN: [real (microns)]
YMAX: [real (microns)]
ZMIN: [real (microns)]
ZMAX: [real (microns)]

The parts of the surface lying between XMIN:XMAX, YMIN:YMAX, ZMIN:ZMAX are converted to the requested name. Any omitted bounds default to the device limits. If all bounds are omitted, the entire surface is converted.

RENAME [logical]

Used to change the name of a boundary (e.g., may be useful when reading in a structure from a .TIF or .PAS files). This keyword requires specifying the old and new names with the "old" and "new" keywords.

OLD [string]

Old name for boundary (used with the "rename" keyword).

NEW [string]

New name for boundary (used with the "rename" keyword).

WORKFUNC [real (eV)]

Value of the workfunction of the metal at this contact.

CONTACT [logical]

Denotes this boundary as an electrical contact (default = true).

INTERNAL [logical]

This keyword allows the boundary to be placed on an internal interface (default = false).

MATERIAL [string]

Boundary condition will be applied only on surfaces or interfaces which include the specified material type.

MATERIAL1 [string]

Boundary condition will be restricted to interfaces of type MATERIAL1 and MATERIAL2. Both keywords must be specified.

MATERIAL2 [string]

Boundary condition will be restricted to interfaces of type MATERIAL1 and MATERIAL2. Both keywords must be specified.

Examples:

```
boundary xmin=0 xmax=0 name=anode contact
boundary xmin=1 xmax=1 name=cathode contact
boundary xmin=-0.21 xmax=-0.19 material=poly name=gate contact
```

4.3.1 Less Used BOUNDARY Keywords

R [real]

Lumped element parameter.

L [real]

Lumped element parameter.

C [real]

Lumped element parameter.

DEBUG [logical]

Print some debugging information.

4.4 DBASE

This card allows values in the program database to be examined or modified. Values stored in the database are of several kinds: physical coefficients for the models, parameters controlling the level of output from the program, parameters governing the numerical convergence, and even the detailed specification of what equations are to be solved and what methods used to solve them. A simulation user is typically concerned with only the first two kinds of database objects.

- [PRINT]
- [PRINTVAL]
- [PRINTLIST]
- [PRINTALL]
- [CHECK]
- [DUMP]
- [MODIFY]
- [CREATE]
- [DELETE]
- [DELETELIST]
- [PREFIX=(s)]
- [NAME=(s)]
- [IVAL=(i)]
- [RVAL=(r)]
- [SVAL=(s)]

The following list itemizes the valid keywords, their units, and their meaning. There is also a table of parameters. See also the database document.

PRINT, PRINTVAL: [logical]

Used to print the value of a database parameter given by the NAME keyword.

PRINT will print the value of a scalar property. Use PRINTVAL to print the values of an array. If PRINT is used for an array property, it will give general information only, not the values.

PRINTLIST: [logical]

Print the members and values of a property list. Must also include the NAME of the property list.

PRINTALL: [logical]

Recursively print a property list and all of its contents. Must also include the NAME of the property list.

PREFIX: [string]

Define a prefix which will be used in subsequent names. Each name will have the prefix prepended; e.g. if the prefix is ”library/physics/silicon” and the name is ”interstitial/Dix” then the full name used to print or modify a property will be
”library/physics/silicon/interstitial/Dix”. To cancel the prefix, use prefix=””.

CREATE: [logical]

Create a new property on an existing property list. It will be placed at the head of the list, and override existing properties of the same name. If it is desired to change an exisiting property, use either “dbase modify” or “dbase create”. Must also include the NAME and value of the new property.

`dbase create name=myname ival=3` creates an integer value
`dbase create name=myname rval=3.0` creates a real value
`dbase create name=myname sval=three` creates a string value

CREATELIST: [logical]

Creates a new property list. NAME must be specified.

DELETE: [logical]

Deletes a property from a list. Must also specify the NAME of the property.

DELETELIST: [logical]

Deletes a property list and its contents recursively. No promises on what happens if you self-lobotimize by deleting from the top down.

MODIFY: [logical]

Modifies an existing property. Must also specify the NAME of the property.

`dbase modify name=myname ival=4` modifies an integer value
`dbase modify name=myname rval=4.0` modifies a real value

`dbase modify name=myname sval=four` modifies a string value

CHECK: [logical]

Checks the integrity of the property database.

DUMP: [logical]

Saves a copy of the current in-memory database to file. This includes any changes made with dbase commands. One drawback relative to copying the original database and editing it is that comments are not preserved. Otherwise the new database can be used to replace the standard one by defining the environment variable PROPHET_DBASE or using the -h flag at startup.

NAME: [string]

The name of the property being modified, created, deleted or printed.

IVAL: [integer]

The integer value stored in a created or modified property.

RVAL: [real]

The real value stored in a created or modified property.

SVAL: [string]

The string value stored in a created or modified property.

Examples:
To print the value of silicon electron longitudinal mobility parameter vsat.c0:
`dbase print name=library/physics/silicon/electrons/vsat.c0`
To print all the properties of electrons (silicon) in the database:
`dbase printlist name=library/physics/silicon/electrons`
To modify the maximum number of allowed Newton iterations:
`dbase modify name=options/math/systems/default_numerical_parameters/maxNewton ival=20`

Table of dbase parameters

Tables 1 and 2 have a selection of commonly used dbase parameters. Note that this document is almost always out of date with respect to the actual database; check that for latest values.

Table 4: Control Parameters

options/junctions	Print junction summary information
options/cpu	Print diffusion cpu timings
options/timestep	Print diffusion time steps
options/loops	Print diffusion inner loops
options/outline.only	Run the simulation in outline mode, no diffusion
options/dump	Save the solution after each timestep
options/dumpi	Save the solution after each inner loop
options/matrix.dump	Print the matrix (VERY verbose)
options/dump.system	List Prophet's idea of the system of equations to solve
options/movie	Plot an xgraph of the time evolution of an impurity - 1D domains only. The string value is equal to the name of the impurity or a comma-separated list of impurities
library/math/plot/mincolor	The color in contour plots wrap around from library/math/plot/mincolor to maxcolor. By playing with these parameters, different color spectrums may be found.
library/math/plot/color.%s (string)	For each material, its color in pictures can be set by specifying a string. Colors are strings like "red", "green", etc. The valid values depend on your Xserver and can often be found in /usr/lib/X11/rgb.txt

Table 5: Math and Grid Parameters

Name	Meaning	Units
library/physics/M/default.tdel	Default vertical grid spacing	Microns
library/physics/M/default.tdel	Default lateral grid spacing	Microns
library/math/grid/interval.ratio	Maximum grid interval ratio	
library/math/maxGSloop	Maximum outer loops	
library/math/LTE	Maximum target difference between coarse and fine timesteps	
library/math/LTE.toler	Maximum tolerated difference between coarse and fine timesteps	
library/math/color-elements	Color elements for vectorization	

4.4.1 Less Used DBASE Keywords

ADD

Same as CREATE, except that if the property already exists, it is not replaced. Rather, a property with a duplicate name is created. Probably not commonly used.

TEMPERATURE [real]

Used for setting temperature (Celsius) in a PRINT, PRINTVAL or PRINTLIST dbase command.

PRESSURE [real]

Used for setting pressure in a PRINT, PRINTVAL or PRINTLIST dbase command.

4.5 DEPOSIT

In this step a material layer is deposited onto an existing structure. The amount of material deposited is equal to the value specified by the THICKNESS key. The deposition can be an overall deposition or a selective one restricted to the rectangular window specified by the START and END keywords. Selective deposition can also be specified by naming the material onto which the layer will be deposited with the HETERO key. The layer can be doped with the specified ELEMENT either uniformly by specifying the CONCENTRATION, or with a exponentially (or linearly) varying profile by also specifying SURFACE (and LINEAR).

- MATERIAL=(s)
- THICKNESS=(r)
- [XDEL=(s)]
- [START=(r[,r])]
- [END=(r[,r])]
- [HETERO=(s)]
- [ELEMENT=(s)]
- [CONCENTRATION=(r)]
- [SURFACE=(s)]
- [LINEAR=(i)]
- [TYPE=(s)]
- [MASK=(s)]
- [XMOLE=(s)]
- [YMOLE=(s)]
- [ZIPPER=(l)]

The following list itemizes the valid keywords, their units, and their meaning.

MATERIAL: [string]

The material to be deposited. Possible materials are found in the database in library/physics. Commonly used materials are ”nitride”, ”oxide”, ”poly”, ”resist”, and ”shipley”. Note that ”resist” is opaque to all implants, whereas ion implant penetration through ”shipley” photoresist is a function of ion species, ion energy and Shipley material thickness.

THICKNESS: [real or real array (microns)]

The thickness of the layer to be deposited. Several thickness values can be specified in real array format (e.g., “0.0,0.1”) in order to have greater control over the vertical grid spacing. In this sense, the THICKNESS keyword plays a similar role as the XLOC keyword for the GRID command (see the section on the GRID command for more detailed information).

XDEL: [real or real array (microns)]

The grid spacing in the layer to be deposited. It is possible to specify the thickness and the spacing as arrays; the thickness should start at zero and increase to the total thickness while the spacing should have the same number of values and indicate the spacing at each thickness value. A spacing of zero indicates a “don’t care” condition. This XDEL plays a similar role as the XDEL of the GRID command, but with an important difference in convention. For the GRID command, the first XDEL grid spacing value refers to the top of the initial structure; however, for the DEPOSIT command, the first XDEL refers to the bottom of the deposited layer.

If XDEL is not specified, the deposited layer will have a spacing given by the database parameter library/physics/M/default.tdel, where M is the material deposited. If that parameter does not exist, the layer will have one grid spacing. For insulators, it does not exist; for semiconductors, the parameter is currently 0.02 microns.

START: [real or real array (microns)]

The coordinates of one corner of the rectangular window in which to deposit. If only one coordinate is given, a stripe parallel to the z-axis will be deposited. If START is not set, the beginning of the simulation domained is assumed.

END: [real or real array (microns)]

The coordinates of the opposite corner of the rectangular window in which to deposit. If only one coordinate is given, a stripe parallel to the z-axis will be deposited. If END is not set, the end of the simulation domained is assumed.

HETERO: [string]

The name of the material to selectively deposit on.

ELEMENT: [string]

The impurity element in the layer. If ELEMENT is not set, the layer is assumed to be intrinsic. CONCENTRATION must also be set if ELEMENT is specified.

CONCENTRATION: [real (cm^{-3})]

The concentration of ELEMENT in the layer. ELEMENT must also be specified. If SURFACE is specified, CONCENTRATION refers to the concentration in the bulk.

SURFACE: [real (cm^{-3})]

For a nonuniform doping profile, this is the concentration at the surface of the layer. ELEMENT and CONCENTRATION (referring to the concentration at the bulk side) must also be specified. The profile will vary exponentially between SURFACE and CONCENTRATION unless LINEAR is set.

LINEAR: [logical]

If set to 1, the doping profile will vary linearly between SURFACE and CONCENTRATION. If set to 0 or not set, the doping profile will vary exponentially. ELEMENT, CONCENTRATION and SURFACE must also be specified.

MASK: [string]

The parameter is a file in which to find mask data. The deposit will occur only inside the rectangles described in the mask datafile, unless the INVERT parameter is specified. At present the maskfile format is one line per rectangle, each line formatted as (y-start) (y-end) (z-start) (z-end) box

TYPE: [string]

The model to be used for deposition. Currently there are two models: default and smooth. The default model deposits exactly THICK microns of material above each point on the exposed surface. The smooth model uses an isotropic deposition model to calculate the thickness above each point. In the case of a planar surface, the two models give the same result. When there are steps on the surface, the smooth model causes thicker oxide to be deposit in the shadows of the steps. This provides a simple way to define sidewall spacers.

ZIPPER [logical]

When depositing materials which are laterally adjacent, the last of the deposit commands must contain this zipper option. The zipper algorithm searches for nodes in adjacent materials which have the same location, and adds this information to the geometry description. Note that the deposition of two laterally adjacent materials must be done with identical vertical grid spacing. (See poly spacer example below.)

XMOLE [real]

Defines “xmole” (stochiometry) for deposited layer. Value must be between 0.0 and 1.0.

YMOLE [real]

Defines “ymole” (stochiometry) for deposited layer. Value must be between 0.0 and 1.0.

Examples:

Blanket deposition

```
deposit material=nitride thick=0.05 xdel=0.005
```

Local deposition in 2D

```
deposit material=poly start=0 end=1.0 thick=0.5 xdel=0.05
```

Local deposition in 3D

```
deposit material=poly start=0,0 end=1.0,0.5 thick=0,0.5 xdel=0.01,0.1
+ elem=arsenic concen=1e16
```

Poly gate with oxide spacer:

```
# Gate Oxide:
deposit mat=oxide thick=0.002 xdel=0.002
# Poly Gate:
deposit mat=poly  start=-0.03 end=0.03  thick=0.0,0.1 xdel=0.002,0.05
# Left Oxide Spacer:
deposit mat=oxide start=-0.05 end=-0.03 thick=0.0,0.1 xdel=0.002,0.05
# Right Oxide Spacer:
deposit mat=oxide start=0.03  end=0.05  thick=0.0,0.1 xdel=0.002,0.05 zipper
```

4.5.1 Less Used DEPOSIT Keywords

NO-EPI-COMPLAINT [logical]

Turn off warning message that suitable deposition surface could not be found.

CMODEL

Obsolete - should be removed.

4.6 DUMP

The dump command provides a simple human-readable printout of various quantities.

- SOLUTION=(l)

- DATABASE=(l)
- GRID=(l)
- INTERFACE=(l)
- OUTFILE=(s)
- UCD=(l)
- DX=(l)
- MATLAB=(l)
- FIELD=(s)

The following list itemizes the valid keywords, their units and their meaning.

SOLUTION: [logical]

Dump just the fields from the current grid structure.
The format is ”Node nnn reg nnn at (X Y Z) S1 S2 S3 S4 ...”

DATABASE: [logical]

Dump the database in the same format as it is read. This command allows a modified memory copy of the database to be stored and reused in a subsequent simulation. Note that all of the database is dumped; however only the part under ”library” should be used as a library in subsequent simulations.
The format is ”Node NNN reg NNN at (X Y Z) S1 S2 S3 S4 ...”

GRID: [logical]

Dump the grid data structures. This may generate a lot of output for large structures.

INTERFACE: [logical]

Dump the subset of grid data structures relating to the interfaces.

OUTFILE: [string]

Save the output to file OUTFILE instead of to the screen.

UCD [logical]

Dump the output in UCD format.

DX [logical]

Dump the output in DX (Data Explorer) format. Use the FIELD keyword to specify which field to dump.

MATLAB [logical]

Dump the output in MATLAB format. Use the FIELD keyword to specify which field to dump.

FIELD [string]

Used to choose which field to dump if dumping to DX or MATLAB format.

Examples:

```
dump solution
dump database outfile=dbase.new
```

4.7 FIELD

This card allows fields to be defined over the existing grid.

- SET(s)
- VALUE(s)

The following list itemizes the valid keywords and their meaning.

SET: [string]

This defines the name of the quantity to be assigned.

VALUE: [string]

The value is an expression to be evaluated across the grid. It can include constants, other solution quantities, and the coordinates X, Y and Z (uppercase).

XRANGE [string]

XRANGE limits the application of the field statement to range specified. For example, "`[0.3:0.5]`" specifies a closed range from x=0.3 to x=0.5, which includes the endpoints of the range. "`(0.3:0.5)`" specifies an open range from x=0.3 to x=0.5, which excludes the endpoints. Open and closed endpoints can be mixed. A fuzzy comparison is used to make the function more robust.

YRANGE

Same as XRANGE, but for y-direction.

ZRANGE

Same as XRANGE, but for y-direction.

STATS [string]

Prints out some statistics for the specified field, such as min, max, average and standard deviation. Typical usage:

```
field stats=electrons
```

MATERIAL [string]

The MATERIAL keyword limits the application of the FIELD command to the specified material.

SOLVAR [string]

The user can request that fluxes be calculated by setting "`set=flux`". The SOLVAR keyword specifies which flux is to be calculated. Typical usage (e.g., for electron currents) is:

```
field set=flux solvar=electrons type=edge
```

This command, issued before the solutions is calculated with a "BIAS" command, causes the currents to be stored in the field "fluxelectrons." Note that this is an edge field, not a nodal field.

PROFILE3D [string]

Read 3-D field from file name specified. See PROFILE2D for format information.

PROFILE2D [string]

Read 2-D field from file name specified. The 2D and 3D file formats support either uniform spacing or rectilinear spacing. The format is

```
<nx> <ny> [<nz>]
'u' | 'r'                ;;; uniform or rectilinear spacing
<tensor mesh coordinates>
<ix> <iy> [<iz>] <value at (ix,iy[,iz])>
...
```

where $<$ nx $>$, $<$ ny $>$, and $<$ nz $>$ are the number of grid lines in each dimension (minimum is 2) indicies run from 0 to n-1, inclusive

The indices and value need not be given in any specific order, but the set should be complete.

For a uniform mesh, the <tensor mesh coordinates> consists of a pair of two real numbers representing the bounds for that dimension ($<$ L $>$ $<$ U $>$). Mesh spacing is (U - L) / ($<$ n[xyz] $>$-1).

For a rectilinear mesh, the <tensor mesh coordinates> consists of the two or three lists of real numbers of length $<$ n[xyz] $>$ (you can have one number per line). These numbers are the locations of mesh lines, and they are assumed to be sorted in increasing order.

The locations of the mesh lines in these files need not correlate to your prophet mesh. Linear interpolation is used to map file values onto the prophet mesh.

For 1d, the values at the endpoints extend to infinity. If you want to truncate assignment outside of the extent, just specify 0 for the value at the start and end of the range.

For 2d and 3d, assignment is truncated outside of the extent specified in the file.

DELETE [string]

Delete the specified field and free its memory.

FILENAME [string]

Read in 1-D data from file. The format is two reals per line, the first being the x-coordinate, the second being the value.

LIST [logical]

List the names and types of all the fields found in the domain.

Examples:

```
field set=boron value=1e18*exp(-(X-0.1)*(X-0.1)/(2*0.01*0.01))
field set=phosphorus value=1e19*gbox(X,0,0.1,0.01)*gbox(Y,0.5,0.8,0.01)
```

The function gbox has a constant value in the interval between its second and third arguments, with gaussian fall-off behavior outside that interval with a characteristic length equal to the fourth argument. The first argument is the axis to substitute. This example specifies a flat profile of 1e19 inside the interval (X,Y) = (0-0.1 , 0.5-0.8) with Gaussian decay of characteristic length 0.01 outside.

4.7.1 Less Used FIELD Keywords

TYPE [string]

Used to specify one of two special field types, either "edge" or "scalar". An edge field is defined for each of the edges in the structure, e.g., for currents. A scalar field has one value defined for the entire geometry.

CLASS [string]

Set the class type of this field (default = "permanent").

AXIS [string]

When reading in 1-D data using the FILENAME command, the AXIS keyword determines whether the data is stored in the x, y, or z direction. The axis is specified by either "`axis=X`", "`axis=Y`", or "`axis=Z`". Default is X.

NEGATE [logical]

When reading in 1-D data from a file using FILENAME, multiply the data values by -1. (Not implemented for 2-D or 3-D data).

DISTANCE [string]

For each node in the field, find the minimum distance to the interface of the material specified. Typical usage:

```
field set=fromOxide distance=oxide
```

finds minimum distance to the oxide interface for each node.

4.8 GRAPH

The GRAPH command yields plots of impurity concentrations, and may appear at any point in the input file. If the solution domain is 1D, one-dimensional plots of concentration versus depth are created. If the solution domain is 2D or 3D, either 2D contour plots, or one-dimensional cross sectional plots (horizontal or vertical or along interfaces) at a specified position can be created. The GRAPH command forks an instance of xgraph to do its work. Thus, it only works while running the X window system. The X window which is created responds to mouse commands. By drawing out a box on the graph, that box will be expanded to fill the entire window. The original perspective can be recovered with the unzoom button. The hardcopy button generates hardcopy in a variety of formats. The popup window allows the selection of a printer or an output file, in b/w or color postscript, HPGL or idraw format. Figures saved in idraw format can have legends, arrows and other graphics added with idraw. The help button is broken. The close button shuts down the xgraph window. Any graph commands issued after an xgraph window has been shut down will cause the simulator to abort.

- [QUANTITY=(x)]
- [ABSLOG=(l)]
- [SIGNEDLOG=(l)]
- [AXIS=(l)]
- [PRINT=(l)]
- [OUTFILE=(s)]
- [XPOSITION=(r)]
- [YPOSITION=(r)]
- [XMIN=(r)]
- [XMAX=(r)]
- [REGION=(s)]
- [CONTOUR=(l)]
- [COLOR=(l)]
- [CMINIMUM=(r)]
- [CMAXIMUM=(r)]
- [CDEL=(r)]
- [NCONTOURS=(i)]
- [BOUNDARY=(l)]
- [ELECTRODES=(l)]
- [SURFACES=(l)]
- [MATERIALS=(l)]
- [GRIDLINE=(l)]

- [ITF=(s)]

The following list itemizes the valid keywords, their units, and their meaning. The commands are divided approximately into General, 1-D, 2-D and Miscellaneous commands.

General Graph Commands:

QUANTITY: [string]

Quantity refers to the quantity to be plotted, which may be virtually any field value. Examples: psi, netdope, electrons, holes, boron, phosphorus, etc. For diffusion simulations, quantities with an asterisk appended (e.g., boron*) refer to the active concentration; otherwise, a name without an "*" refers to the chemical concentration.

ABSLOG: [logical]

Plot data as the log of the absolute value. Default is generally false, but it is set to true for 1-D plots of netdope, electrons, or holes.

SIGNEDLOG: [logical]

Plot positive data as $\log(data)$, negative data as the $-\log(|data|)$. Default is generally false, but it is set to true for contour plots of netdope, electrons, or holes).

AXIS: [logical]

This determines whether an axis is drawn. It is true by default, but if set false the graph will be superimposed on the previous graph.

1-D Graph Commands (including 1-D slice of 2-D domain):

PRINT [logical]

The data should be printed out in two columns, rather than being graphed. If OUTFILE is not specified, the data is printed to the console. For 1-D plots only.

OUTFILE: [string]

Specify name of file in which to print 1-D. Used with the PRINT option.

XPOSITION: [real (microns)]

The depth at which a lateral one-dimensional profile is desired. For example, XPOSITION=0.1 gives the profile at an absolute x value of 0.1 microns.

YPOSITION: [real (microns)]

The lateral position at which a one-dimensional profile is desired if the domain is 2D.

XMIN: [real (microns)]

The minimum depth for a plot. Default is the minimum depth in the domain. Works only for 1-D plots or 1-D cross sections of 2-D plots.

XMAX: [real (microns)]

The maximum depth for a plot. Default is the maximum depth in the domain. Works only for 1-D plots or 1-D cross sections of 2-D plots.

REGION [string]

Applies only to 1-D plots or 1-D cross-sections of 2-D plots. Restrict plot to region with the specified material name.

2-D Graph Commands:

CONTOUR: [logical]

This determines whether or not to draw contours of the impurity specified by QUANTITY.

COLOR: [logical]

This determines whether or not to color the contours of the impurity specified by QUANTITY (must be used with CONTOUR).

CMINIMUM: [real]

The minimum contour plotted (log scale). The default is 14 (not 1e14).

CMAXIMUM: [real]

The maximum contour plotted. The default is 21 (not 1e21).

CDEL: [real]

The interval between contours. If CDEL is not specified, CINT is used to calculate the contour values.

CINT [integer]

Number of contour levels to use in a contour plot. Note that the user can specify CDEL or CINT, but not both. Default value is 10.

BOUNDARY: [logical]

This determines whether or not the lines describing the outlines of the regions are drawn (2d plots). By default boundaries are turned off.

ELECTRODES [logical]

This determines whether or not the lines describing the electrodes are drawn (2d plots). By default, electrodes are drawn for 2-D plots.

SURFACES [logical]

This determines whether or not the lines describing the external outlines of the entire simulation domain are drawn (2d plots). By default, surfaces are turned off, but are turned on by default if no plot quantity is specified.

MATERIALS: [logical]

This determines whether or not the polygons describing the outlines of the regions are colored in (2d plots). By default materials are turned on for 2-D plots, but if axis is turned off, materials defaults to off.

GRIDLINE: [logical]

This determines whether or not the lines of the grid are drawn (2d plots only). The default is to draw no gridlines.

ITF: [string]

The interface along which to plot a concentration, expressed as material1/material2. The value in material1 is plotted along the interface with material2 (so a plot at material1/material2 is not necessarily the same as material2/material1). Usage:

```
graph quantity=netdope itf=silicon/oxide
```

Miscellaneous Graph Commands:

IV [logical]

Request a file with accumulated I-V data from this run. Used after all BIAS commands of interest. Typical usage:

```
graph iv outfile=mos.iv
```

Examples:
To make a 2-D color contour plot of netdope:
```
graph quantity=netdope contour color
```

Same as above, but with user specified contours:
```
graph quantity=netdope contour color cmin=15 cmax=22
```

To make a vertical 1-D cross section of potential in a 2-D simulation:
```
graph quantity=psi yposition=0.0
```

To make a horizontal 1-D cross section of electron density in a 2-D simulation, and print results to a file:
```
graph quantity=electrons xposition=0.01 print outfile=electrons.dat
```

To examine materials and gridlines of a 2-D structure:
```
graph gridline
```

To plot the hole density in silicon at the silicon/oxide interface:

```
graph quantity=holes itf=silicon/oxide
```

4.8.1 Less Used GRAPH Keywords

ELEMENT: [string]

ELEMENT refers to the quantity to be plotted, which may be virtually any field value. Examples: psi, netdope, boron, phosphorus, etc. For diffusion simulations, quantities with an asterisk appended (e.g., boron*) refer to the active concentration; otherwise, a name without an "*" refers to the chemical concentration. It is preferred to use the synonym QUANTITY.

CMAT [string]

Determine upper and lower bounds for contour plots automatically checking only the material CMAT. This command is used only if the user wants Prophet to determine these values automatically. Otherwise, cmin, cmax, and cdel can be explicitly specified with the keywords CMIN, CMAX, and CDEL.

NEW.WINDOW: [logical]

Currently not working. Start sending graph data to a new window, leaving picture in previous frame untouched.

GRIDPOINT [logical]

Attempts to print gridpoints, but currently not working for xgraph.

ROUNDTO [real]

When Prophet automatically determines CMIN (CMAX) for contour plots, it rounds down (up) to the nearest fraction specified by ROUNDTO.

XCLIP [logical]

Currently not needed.

LINE [string]

Sets line color for contour plot (default is black).

LOG [integer]

If LOG=1, graph $\log(data)$. If LOG=-1, plot positive data as $\log(data)$, negative data as the $-\log(|data|)$. If LOG=0, graph actual data unchanged. It is preferred to use ABSLOG and SIGNEDLOG keywords instead.

XGRAPH [logical]

Choose XGRAPH as plotting tool (default is true for 2-D plots).

XMGR [logical]

Choose XMGR as plotting tool (default is true for 1-D plots).

FILL: [logical]

Currently not working. Should be re-named SAMESCALE. This determines whether the same scale is used in the x and y directions for 2d plots. By default the picture is scaled to fill the plotting window, but if fill=f, the picture will be reduced until both axis can be drawn at the same scale.

XYFORMAT [string]

Alternative output format for 1-D PRINT output (printf syntax).

YMIN: [real (microns)]

Currently not working. The left hand side of the plot (2d plot) or the bottom of the y axis (1d plot). Default is the leftmost point in the domain (2d) or the smallest y value (1d).

YMAX: [real (microns)]

Currently not working. The right hand side of the plot (2d plot) or the top of the y axis (1d plot). Default is the rightmost point in the domain (2d) or the largest y value (1d).

4.9 GRID

The GRID command defines a simple 1D,2D or 3D tensor product grid. The GRID (or LOAD) command must appear before any commands such as field, graph or bias.

The algorithm for creating non-uniform tensor product grid operates segment-by-segment, where the segments are defined by the values specified in the XLOC (or YLOC, ZLOC) keyword. Each XLOC value has an associated XDEL value, which is the grid spacing request that Prophet will try to accomodate. When a segment has the same XDEL values at both endpoints, the grid will be uniformly spaced at that value. When the XDEL values are different, Prophet will grade the grid spacing as smoothly as possible, minimizing the ratio of adjacent grid spacing. However, within a segment the ratio will never be greater than "/library/math/grid/interval.ratio" (set to 1.5 in the Prophet database currently). If the change in requested grid spacing within a segment is too extreme to meet this ratio, the smaller grid XDEL will be satisfied, and the larger ignored. To ensure that the requested spacing is possible, the user can check his/her input to see if the following is satisfied:

$$interval.ratio \geq \frac{(\text{segmentLength} - \text{smallerXDEL})}{(\text{segmentLength} - \text{largerXDEL})} \tag{9}$$

If the above inequality is not satisfied, it is possible that the grid spacings in adjoining segments will

have a larger ratio than that specified by "/library/math/grid/interval.ratio". However, the maximum ratio is always met within each individual segment.

- MATERIAL=(s)
- ELEMENT=(s)
- [RESISTIVITY=(r)]
- [CONCENTRATION=(r)]
- [ORIENTATION=(i)]
- [DIMENSION=(i)]
- [XLOC=(r)]
- [XDEL=(r)]
- [YLOC=(r)]
- [YDEL=(r)]
- [ZLOC=(r)]
- [ZDEL=(r)]
- [MAP.MOSFET=(s)]
- [XMOLE=(r)]
- [YMOLE=(r)]

The following list itemizes the valid keywords, their units, and their meaning:

MATERIAL: [string]

> The type of material the substrate is made of. Possible values are found in the library in library/physics.

ELEMENT: [string]

> The impurity element initially in the substrate. If ELEMENT is not set, the substrate is assumed to be intrinsic. Either RESISTIVITY or CONCENTRATION must also be set.

RESISTIVITY: [real (ohms)]

> The substrate resistivity. Specify either RESISTIVITY or CONCENTRATION.

CONCENTRATION: [real (cm^{-3})]

> The uniform concentration of ELEMENT in the substrate. Specify either CONCENTRATION or RESISTIVITY.

DIMENSION: [integer]

> The spatial dimension of the original grid of the simulation domain. Value may be 1, 2, or 3.

XLOC: [real (microns)]

> An array of grid locations in the vertical direction, separated by commas. The first and last

values specify the beginning and end of the initial simulation domain. Intermediate values are used to specify a varying grid spacing.

XDELTA: [real (microns)]

The grid spacing in the vertical direction. An array of the same length as XLOC can be given to specify the grid spacing at each XLOC. The grid is varied gradually between each XLOC.

YLOC: [real (microns)]

An array of grid locations in the lateral Y direction, separated by commas. The first and last values specify the beginning and end of the initial simulation domain. Intermediate values are used to specify a varying grid spacing.

YDELTA: [real (microns)]

The grid spacing in the Y direction. An array of the same length as YLOC can be given to specify the grid spacing at each YLOC. The grid is varied gradually between each YLOC.

ZLOC: [real (microns)]

An array of grid locations in the Z direction, separated by commas. The first and last values specify the beginning and end of the initial simulation domain. Intermediate values are used to specify a varying grid spacing.

ZDELTA: [real (microns)]

The grid spacing in the Z direction. An array of the same length as ZLOC can be given to specify the grid spacing at each ZLOC. The grid is varied gradually between each ZLOC.

MAP.MOSFET [string]

Map the fields from a MOSFET's pas file onto a new grid, flattening the silicon/gate-oxide interface in the process. THe basic purpose is to create a rectlinear grid, but using the fields from a MOS device with surface curvature created by process simulation This command assumes the silicon/gate-oxide interface is located at x=0. Fields in the silicon substrate are shifted vertically, if necessary. The new grid must be previously defined, and must not lie outside the domain of the mapped fields. The new grid must have its top at x=0. Typical usage:

```
grid dim=2 xloc=0,0.01,0.2 xdel=0.001,0.01,0.05
+ yloc=-0.2,0,0.2 ydel=0.01,0.01,0.01 mat=silicon
grid map.mosfet=myfile.pas
```

XMOLE [real]

Defines "xmole" (stochiometry) for region described by the grid statement. Value must be between 0.0 and 1.0.

YMOLE [real]

Defines “ymole” (stochiometry) for region described by the grid statement. Value must be between 0.0 and 1.0.

Examples::

1-D MOS Capacitor:

```
grid dim=1 mat=silicon
+ xloc=0.00000,0.0010,0.010,0.10,0.20,0.50
+ xdel=0.00005,0.0002,0.001,0.01,0.05,0.05
+ elem=boron conc=1.0e+18
deposit mat=oxide thickness=0.0,0.002 xdel=0.00005,0.0005
```

Note that in the deposit command above, the first “xdel” value refers to the bottom of the deposited layer. However in the grid command, the first “xdel” refers to the top of the initial structure. In order to have good accuracy, it is necessary to have vertical grid spacing on the order of an Angstrom adjacent to the silicon oxide interface.

2-D MOS:

```
grid dim=2 mat=silicon
+ xloc=0.00000,0.0010,0.010,0.10,0.20,0.50
+ xdel=0.00005,0.0002,0.001,0.01,0.05,0.05
+ yloc=-0.15,-0.07,-0.05,-0.04,0.040,0.050,0.07,0.15
+ ydel=0.02,0.005,0.002,0.005,0.005,0.002,0.005,0.02
+ elem=boron conc=1.0e+18
deposit mat=oxide thickness=0.0,0.002 xdel=0.00005,0.0005
```

For a MOS device, it is important to have sufficient lateral grid density at the source and drain junctions.

4.9.1 Less Used GRID Keywords

MASK.GRID [logical], EDGE.DELTA [real (microns)], SPACE.DELTA [real (microns)]

Grid refinement around mask edges in a process flow can be automated using the MASK.GRID option. The logical MASK.GRID specifies that fine grid should be automatically allocated at the mask edges encountered in the input file, with coarser grid in between. The finest spacing at the edges is given by EDGE.DELTA. The coarsest spacing is given by SPACE.DELTA. The fine spacing is guaranteed to be honored, but depending on location of mask edges, the coarse spacing may be finer than requested. The spacing is automatically graded from coarse to fine areas. A SPACE.DELTA of 0.0 requests that the grid be made as coarse as possible between

mask edges, consistent with not increasing the spacing more than a factor of 1.5 in any interval. The value of 1.5 comes from library/math/grid/interval.ratio.

ORIENTATION: [integer]

The crystalline orientation of the substrate. Valid values are 100, 110 and 111.

TRIANGULATE [logical]

Generate a triangular grid from a rectangular grid, or from a grid composed of rectangles and triangles.

QUALITY [logical]

Calculate and print a number of quality indicators for a 2-D triangular grid.

ADAPT [logical]

Grid adaptation for 2-D triangular grid.

REFINE [logical]

Grid refinement for 2-D triangular grid based on the field specified with FIELD keyword.

FIELD [string]

This keyword specifies which field to use for refinement criteria when using the REDINE keyword.

RATIO [real]

Unused parameter for REFINE keyword.

CLEANUP [logical]

Clean-up 2-D triangular grid.

SHEAR [real]

Shift x-coordinate of each point by an amount defined by SHEAR times that point's y-coordinate.

XSHIFT [real]

Shift x-location of coordinates by specified amount.

YSHIFT [real]

Shift y-location of coordinates by specified amount.

CURVE.MOS [logical]

Flatten a structure with a curved top surface by moving coordinate points, thus shifting the

fields. Note that this will create a non-rectlinear grid.

4.10 IMPLANT

During an IMPLANT step, the statistics from the library are used to calculate the PearsonIV impurity profile of the implanted ELEMENT. If the range, standard deviation or higher moments of the desired impurity distribution are specified, these values overwrite the library values used to calculate the profile. The user can also specify an exponentially decaying tail, which begins either at BEGINTAIL or the position where the concentration is half the peak value, to account for the effects of channeling.

- ELEMENT=(s)
- DOSE=(r)
- ENERGY=(r)
- [RANGE=(r)]
- [SIGMA=(r)]
- [LAT.SIGMA=(r)]
- [SKEWNESS=(r)]
- [KURTOSIS=(r)]
- [TAIL=(r)]
- [BEGINTAIL=(r)]
- [TILT=(r)]
- [FILE=(s)]
- [DSCALE=(r)]
- [PLUSFACTOR=(r)]

The following list itemizes the valid keywords, their units, and their meaning:

ELEMENT: [string]

The impurity element to be implanted. Possible values are found in the library in library/physics/implant.

DOSE: [real (cm^{-2})]

The dose of the implant. The dose may be trimmed using the command dbase create library/physics/silicon/boron/tom.factor rval=0.83 taking boron as an example. This will modify the boron dose to 83% of the nominal value in every subsequent implant.

ENERGY: [real (keV)]

The energy of the implant.

TILT: [real (degrees)]

The tilt of the beam. The model assumes no rotation, only tilt.

RANGE: [real (microns)]

The range of the distribution.

SIGMA: [real (microns)]

The vertical standard deviation of the distribution.

LAT.SIGMA: [real (microns)]

The lateral standard deviation of the distribution.

SKEWNESS: [real]

Skewness of the profile (3rd moment).

KURTOSIS: [real]

Kurtosis of the profile (4th moment).

TAIL: [real (microns)]

The characteristic length for the exponential tail of the profile.

BEGINTAIL: [real (microns)]

The distance from the surface where the tail of the profile should begin. Default is where the concentration of the implant reaches half the maximum.

FILE: [string]

The name of the file from which the 1D implant statistics is read in. The file must have two columns, separated by tabs or blank characters. The first column should give the depth in microns, and the second column the concentration in cm^{-3}. For a 2D simulation, by default the lateral sigma is set to equal the vertical sigma value which is extracted from the 1D implant statistics given by the file. The user can also set a value for the lateral sigma using the ”lat.sigma” option.

DSCALE: [real]

A scaling factor to apply to a profile read from file. Default value is 1.0. Caution: does not apply to profiles specified with dose and energy, only profiles from file.

PLUSFACTOR: [real]

Instructs the implant code to implant a number of interstitials equal in distribution to the implanted species, and with dose equal to that of the implanted species multiplied by PLUSFACTOR. The implanted dose should not exceed the amorphization threshold of silicon; attempts to do so will be rejected.

Examples:
implant energy=30 dose=1.e16 elem=arsenic
implant element=boron energy=100 dose=1e15 lat.sigma=.021
implant element=arsenic file=my_arsenic lat.sigma=0.1

where file ”my_arsenic” is of the form:

0.010 3.0e20
0.012 3.1e20
0.0135 3.3e20

4.10.1 Less Used IMPLANT Keywords

CONCENTRATION

Concentration resulting from implant. Either DOSE or CONCENTRATION may be specified, but not both.

COLUMN.IMSIL

Read implant data from IMSIL.

CMODEL

Obsolete keyword. Should be removed.

4.11 LOAD

The LOAD command is used to restart a simulation from a stored file.

- PAS=(s)
- TIF=(s)

The following list itemizes the valid keywords, their units, and their meaning.

PAS: [string]

The name of the Prophet Ascii Structure file where the structure is stored.

TIF: [string]

The name of the TIF file where the structure is stored.

Examples:

```
load pas=middle.pas
load tif=fromsuprem4.tif
```

4.11.1 Less Used LOAD Keywords

LIST [logical]

Currently not working. Use “FIELD LIST” instead. The “LOAD LIST” command is meant to list regions and fields in the current domain. This keywork should be removed.

DUMMY [integer]

Obsolete keyword (not referenced). Should be removed.

ROTATE [real]

Obsolete keyword (not referenced). Should be removed.

RAW [string]

Read in an additional field from specified filename into an existing domain. Format is

```
<No. of nodes> <fieldname>
<field values>
```

The field values are given one per line. The number of nodes must match that of the existing domain. It is not allowed to overwrite an existing field.

TRI [string]

Load SUPREM4/IGGI file with specified filename.

4.12 SAVE

The SAVE command is used to save a simulation in several formats.

- [PAS=(s)]
- [TIF=(s)]

The following list itemizes the valid keywords, their units, and their meaning.

TIF: [string]

The name of the file to use for storing a TIF file (usually has a .tif suffix)

Examples:
`save tif=forsuprem4.tif`

4.12.1 Less Used SAVE Keywords

MTV [string)]

Save to MTV format file with specified filename. Currently hard-coded to save “netdope” field.

UCD [string]

Currently not working. Save file in AVS UCD format.

4.13 SOLVE

This command solves transient PDEs associated with the fields in the structure. The description of the PDEs is found on the property lists associated with the fields (see math documentation).

- HOURS
- MINUTES
- SECONDS
- TEMPERATURE
- TEMPFINAL
- SYSTEM

The following list itemizes the valid keywords and their meaning.

HOURS: [real]

MINUTES: [real]

SECONDS: [real]

The total time is 3600*HOURS + 60*MINUTES + SECONDS. Timesteps which are printed out are in seconds.

TEMPERATURE: [real]

Temperature at which solution process is carried out, in degrees Celsius. This causes a variable T (absolute temperature = TEMPERATURE + 273.15) and a variable kT (where k is Boltzmann’s constant normalized by electron charge) to be defined in the parser symbol table, for use in temperature dependent constants.

TEMPFINAL: [real]

If the temperature is not constant, it is linearly ramped from TEMPERATURE to TEMPFINAL.

SYSTEM: [string]

The system of equations to be solved is decided by comparing the name of the system with the fields in the structure. If no system is specified, the system of equations to be solved is taken from

library/physics/material/field/math.default

for each field in the structure. (Normally all fields will have a SeeAlso link to the same system; if not, two or more systems will be solved simultaneously, one for each independent system belonging to the fields present. Any field not having a PDE system defined will be ignored during the solution process). If a SYSTEM=xyz is given, the system is taken from

library/physics/material/field/math.xyz

If the given systemname (”default” if none given) does not solve for any of the fields in the structure, a warning will be printed:

No system has a pdeblock defined? steady_solve: bad setup. Quitting

Some systems are predefined, among them the following System Name Expected Fields Problem description

”default” for dopants standard fermi-level dependent diffusion for silicon doping

”default” for fields c1,c2,...cn simple reaction-diffusion system using reactions in ode.F coupled diffusion for TED ”moment1” for dopants and simulation using one moment interstitials model of clusters nonlinear poisson equation for ”newton0” for psi device simulation with flat fermi levels

”newton1e” for psi,electrons drift-diffusion model for electrons only

”newton1h” for psi,holes drift-diffusion model for holes only

”newton2” for psi,holes,electrons drift-diffusion model for 2 carriers

4.14 SYSTEM Command

- Summary
- Synopsis
- Argument List
- Argument Descriptions

Summary:

This command allows user to specify to Prophet a new system of partial differential equations (PDEs) and associated boundary conditions (BCs), interface constraints (ICs), and auxiliary functions. The PDE system description must be complete, including all necessary operators (PDE terms, BCs, ICs, and functions) needed to completely specify the system. If the geometric and physical operators known to Prophet are insufficient to fully describe the PDE system of interest, one or more new operator routines will have to be created and compiled into Prophet.

Synopsis:

```
system name=system_name
+   sysvars=var1,var2,var3,...
+   term0=R*geo_op.phy_op(in1,...|out1,...)@{region1,...}
+   ...
+   termNT-1=R*geo_op.phy_op(in1,...|out1,...)@{region1,...}
+   tmpvars=tmp1,tmp2,tmp3,...
+   func0=R*phy_op(in1,...|out1,...)@{region1,...}
+   ...
+   funcNF-1=R*phy_op(in1,...|out1,...)@{region1,...}
+   init0=''field:expression''
+   ...
+   initN-1=''field:expression''
```

Argument List:

- NAME
- SYSVARS
- TERM0, TERM1, ...
- TMPVARS
- FUNC0, FUNC1, ...
- INIT0, INIT1, ...

Argument Descriptions:

The following list itemizes the valid arguments, their units, and their meaning.

NAME: [string]

The name of the PDE system to be defined.

SYSVARS: [string]

Comma-separated, ordered list of the solution variables for the PDE system. The PDE terms (see below) will be ordered in the matrix equation according to the order of the associated solution variavles in the SYSVARS list.

TERM0, TERM1, ...: [string]

Description of a single term of the PDE system. Prophet currently limits PDE systems to 78 PDE terms, named term0...term77.

For region-based PDE terms, the general form is:

```
termi=R*geo_op.phy_op(in1,in2,...|out1,out2,...)@{region1,region2,...}
```

For boundary conditions or interface constraints, the general form is:

```
termi=R*geo_op.phy_op(in1,in2,...|out1,out2...)@{reg1/reg2,reg3/reg4,...}
```

- **R**: any real number, with a minus sign if appropriate (minus sign by itself is invalid)
- **geo_op**: one of the geometric operators known to Prophet
- **phy_op**: one of the physical operators (a.k.a., flux routines) known to Prophet
- **in1,in2,...**: comma-separated list of sysvars, tmpvars, and other defined fields needed to compute this PDE term. If there are no input fields, ”0” (the number zero) should precede the vertical bar.
- **out1,out2,...**: comma-separated list of sysvars (PDEs) in which this term appears
- **region1,region2,...**: comma-separated list of regions in which this term is relevant
- **reg1/reg2,reg3/reg4...**: comma-separated list of boundaries or interfaces (written as two region names, or a region name and a boundary name, separated by a slash ”/”) to which this term should be applied

TMPVARS: [string]

Comma-separated, ordered list of the temporary (auxiliary) variables computed and possibly used in the PDE system. These variables can not involve differential operators, or they must be part of the PDE system. The temporary variables are computed in the order given by [the TMPVARS list — the FUNC statements], so a TMPVAR must be listed first if it is used in the computation of another TMPVAR.

FUNC0, FUNC1, ...: [string]

Description of a single function of the PDE system. Such auxiliary functions involve only local computations, so that the geo_op, which is always ”nodal”, is omitted from the function description. Prophet currently limits PDE systems to 57 auxiliary functions, named func0...func56.

The general form of a PDE function is:

```
funci=R*phy_op(in1,in2,...|out1,out2,...)@{region1,region2,...}
```

- **R**: any real number, with a minus sign if appropriate (minus sign by itself is invalid)
- **phy_op**: one of the physical operators (a.k.a., flux routines) known to Prophet
- **in1,in2,...**: comma-separated list of sysvars, tmpvars, and other defined fields needed to compute this function. If there are no input fields, ”0” (the number zero) should precede the vertical bar
- **out1,out2,...**: comma-separated list of tmpvars that this function computes
- **region1,region2,...**: comma-separated list of regions in which this function is relevant

INIT0, INIT1, ...: [string]

Optional command to initialize fields. If a field already exists at the time the system is executed, the initialization is not performed. The basic syntax is
`initN=''FieldName:Expression''`
for which N is an integer, “FieldName” is the name of the field to be initialized, and “Expression” evaluates to the initial value of that field. Currently, Prophet is limited to 57 initialization functions. Example of usage:
`init3="mn:0.19"`
The above will check to see if the field mn exists; if it doesn’t, the field will be created and given the initial value of 0.19.

4.14.1 Less Used System Commands

SPAWN: [string]

A list of field names which should be created on the domain when this system is solved, if they are not already present. Another feature of the spawn variable is conditional creation. A variable may be marked as being necessary if another variable exists. For example,
`+ spawn=''psi,boronActive:boron''`
In this example, “psi” is always created, but “boronActive” is created only if “boron” is present.

NFUNC: [integer]

This keyword should no longer be used, as it is unnecessary. Prophet has learned how to count the number of functions. The previous meaning was the specification of the number of function terms in the PDE system.

NTERM: [integer]

This keyword should no longer be used, as it is unnecessary. Prophet has learned to count the number of terms on its own. The previous meaning was the specification of the number of PDE terms in the PDE system.

5 Prophet Options

5.1 Numerical Solution Control Parameters

Parameters controlling convergence tolerances, etc., are defined below. All of the parameters need the prefix `library/math/systems/default_numerical_parameters`. That is, when the table list the parameter as `maxNewton`, the command to modify the parameter is:
`dbase modify name=library/math/systems/default_numerical_parameters/maxNewton ival=20`.

Table 6: Numerical Solution Control Parameters

Parameter Name	Default	Description
maxNewton	10	Maximum number of Newton iterations
NewtonUpd	1e-5	Relative convergence criterion (applies to updates)
NewtonMaxUpd	1e9	Maximum amount of update allowed per Newton iteration

5.1.1 Less Used Numerical Solution Control Parameters

All of the parameters need the prefix `library/math/systems/default_numerical_parameters`.
That is, when the table list the parameter as `method`, the command to modify the parameter is:
`dbase modify name=library/math/systems/default_numerical_parameters/method sval=direct`.

Table 7: Less Used Numerical Solution Control Parameters

Parameter Name	Default	Description
MaxRHSNormIncrease	1e15	Maximum amount of RHS norm increase allowed per iteration
MaxAggregateRHSNormIncrease	1e30	Maximum increase from initial RHS norm
method	direct	(iterative avaliable with petsc).
lsPackage	umfpack	linear solver package Also available: petsc, bsparse

5.2 Debugging Options

Debugging options are used to provide extra information for trouble-shooting.

Table 8: Newton Iteration Control

Parameter Name	Default	Description
options/dump-inner	false	Create a *.pas file for each Newton iteration
options/print.merr	false	Print position and value of max update error

6 Prophet Systems

In this chapter, some standard model systems are described which are available in the Prophet distribution. Each "system" is a set of equations forming a particular type of description of the semiconductor device.

To use the system entitled "silicon_poisson", for example, enter the following line near the beginning of the Prophet input file:

```
include(silicon_poisson)
```

The basic systems currently supported are listed below; more detailed explanations follow in subsequent sections. The first three systems cover most mainstream applications:

- **silicon_poisson:** Solve Poisson equation only (no current flow). Used for initial solutions.
- **silicon_dd:** Solve Poisson equation and two current continuity equations (electrons and holes). Uses Arora dopant dependent mobility model and longitudinal field mobility reduction by default.
- **silicon_dd_lombardi:** Same as silicon_dd, but with Lombardi transverse field mobility model.
- **silicon_dg5:** Five equation density gradient system (quantum corrections, current flow). Uses Arora dopant dependent mobility model and longitudinal field mobility reduction by default.
- **silicon_dg_lombardi:** Same as above, but with Lombardi transverse field mobility model.
- **silicon_poisson_ex:** Same as silicon_poisson, but also calculates vertical E-field. Useful for 1-D MOS capacitors, for example.
- **silicon_dd_ex:** Same as silicon_dd, but with calculation of transverse electric field.
- **silicon_dg3:** Three equation density gradient system (quantum corrections, no current flow).
- **silicon_dg3_ex:** Three equation density gradient system (no current flow). Useful for 1-D MOS capacitors, for example.
- **silicon_dg5_ex:** Same as silicon_dg5, but with calculation of transverse electric field.

All of the above semiconductor systems requre a **netdope** field to be defined in order to specify the doping profile. While the format of the doping specification can very greatly, an example is shown below (and not repeated for the individual systems):

netdope - set doping profile for structure

```
# doping example: 1-D diode (see ‘‘field’’ command defintion for more info)
field set=boron value=1.0e18*gbox(X,-0.3,0.0,0.1) material=silicon
field set=arsenic value=1.0e18*gbox(X,0.5,0.2,0.1) material=silicon
field set=netdope value=arsenic-boron
```

Detailed descriptions of the current built-in systems follow this section. All of the descriptions use the following definitions:

$$
\begin{aligned}
\psi &= \text{electrostatic potential} \\
n, p &= \text{conc. of electrons, holes} \\
\varepsilon(r) &= \text{dielectric constant} \\
q &= \text{electronic charge} \\
N_D, N_A &= \text{conc. of donors, acceptors} \\
J_n, J_p &= \text{current density of electrons, holes} \\
F_n = -J_n/q &= \text{flux density of electrons} \\
F_p = J_p/q &= \text{flux density of holes} \\
r &= \text{recombination rate} \\
\mu_n &= \text{electron mobility} \\
\mu_p &= \text{hole mobility} \\
\phi_n &= \text{electron quasi} - \text{Fermi|level} \\
\phi_p &= \text{hole quasi} - \text{Fermi|level}
\end{aligned}
$$

6.1 silicon_poisson

The “silicon_poisson” system solves for psi.

$$-\nabla \cdot (\varepsilon(r)\nabla\psi) - q(p - n + N_D^+ - N_A^-) = 0 \tag{10}$$

The silicon_poisson system can only be used in situations in which no currents are flowing. It is most often used to obtain the initial solution at zero bias to be used in a subsequent silicon_dd calculation, for example. However, silicon_poisson can also be used with applied bias for certain structures, such

as a 1-D MOS capacitor. In that case, the most convenient approach is use a different material (e.g., poly) for the gate, and then set the following dbase parameter as follows:

```
dbase create name="/options/modulate_qf_with_bias" sval=''poly''
```

This dbase entry will set the fixed quasi-Fermi level of the poly equal to the bias applied to the electrode in the bias statement.

An implementation of silicon_poisson is given below:

```
system name=silicon_poisson
+       sysvars=psi
+       term0=ndiv_fbm.lapflux(psi|psi)@{silicon,poly,oxide}
+       term1=nodal.nscd(electrons,holes,netdope|psi)@{silicon,poly}
+       term2=dirichlet.device_dirichlet(netdope|psi)@{silicon/gate,poly/gate,silicon/ba
+       term3=constraint.continuity(psi|psi)@{silicon/oxide,poly/oxide}
+       tmpvars=electrons,holes
+       func0=quasiFermi(psi|electrons,holes)@{silicon,poly}
```

6.2 silicon_dd

The "silicon_dd" system solves for psi, electrons, and holes.

$$
\begin{aligned}
-\nabla\cdot(\varepsilon(r)\nabla\psi) - q(p - n + N_D^+ - N_A^-) &= 0 && (11)\\
\frac{\partial n}{\partial t} - \nabla\cdot(-F_n) + r_{\text{srh}} &= 0 && (12)\\
\frac{\partial p}{\partial t} - \nabla\cdot(-F_p) + r_{\text{srh}} &= 0 && (13)\\
& && (14)
\end{aligned}
$$

or, with the carrier fluxes written more explicitly:

$$
\begin{aligned}
-\nabla\cdot(\varepsilon(r)\nabla\psi) - q(p - n + N_D^+ - N_A^-) &= 0 && (15)\\
\frac{\partial n}{\partial t} - \nabla\cdot[\mu_n(kT/q)\nabla n - \mu_n n\nabla\psi] + r_{\text{srh}} &= 0 && (16)\\
\frac{\partial p}{\partial t} - \nabla\cdot[\mu_p(kT/q)\nabla p + \mu_p p\nabla\psi] + r_{\text{srh}} &= 0 && (17)
\end{aligned}
$$

where the quantities in brackets are the negatives of the carrier fluxes, implemented by the "ncflux" flux term. For reference, $F_n = -J_n/q$ and $F_p = J_p/q$.

This system requires “netdope” to be defined after the structure is created with grid and deposit statements.

An implementation of this system is given below.

```
system name=silicon_dd
+       sysvars=psi,electrons,holes
+       term0=dirichlet.device_dirichlet_h(netdope,ni,nc,nv|psi,electrons,holes)
@{silicon/gate,poly/gate,silicon/back,silicon/source,silicon/drain,oxide/gate,si
licon/substrate}
+       term1=ndiv_fbm.lapflux(psi|psi)@{silicon,poly,oxide}
+       term2=nodal.nscd(electrons,holes,netdope|psi)@{silicon,poly}
+       term3=ndiv_fbm.ncflux(psi,electrons,tl,nmob0,nin,edge|electrons)@{silico
n,poly}
+       term4=ndiv_fbm.ncflux(psi,holes,tl,pmob0,nip,edge|holes)@{silicon,poly}
+       term5=constraint.continuity(psi|psi)@{silicon/oxide,poly/oxide}
+       term6=transient.ddt(electrons,holes|electrons,holes)@{silicon,poly}
+       term7=nodal.srh(electrons,holes,ni,taun,taup|electrons)@{silicon,poly}
+       term8=nodal.srh(electrons,holes,ni,taun,taup|holes)@{silicon,poly}
+       tmpvars=eg,nc,nv,ni,nmob0,pmob0,kappa,nin,nip,taun,taup
+       func0=eg(tl,netdope|eg)@{silicon,poly}
+       func1=dos(electrons,holes,tl|nc,nv)@{silicon,poly}
+       func2=ni(tl,eg,nc,nv|ni)@{silicon,poly}
+       func3=mob0(electrons,holes,tl,netdope|nmob0,pmob0)@{silicon,poly}
+       func4=kappa(tl|kappa)@{silicon,poly,oxide}
+       func5=nic(ni,nc,electrons,tl|nin)@{silicon,poly}
+       func6=nic(ni,nv,holes,tl|nip)@{silicon,poly}
+       func7=tau(electrons,holes,netdope|taun,taup)@{silicon,poly}
```

6.3 silicon_dd_lombardi

The “silicon_dd_lombardi” model is similar to the silicon_dd system except that it adds the Lombardi transverse field mobility reduction model.

6.4 silicon_dg5

The “silicon_dg5” system solves for the five equation density gradient model.

$$\nabla \cdot (\varepsilon \nabla \psi) + q(p - n + N_D^+ - N_A^-) = 0 \quad (18)$$

$$\frac{\partial n}{\partial t} - \nabla \cdot (-\mu_n n \nabla \phi_n) + r = 0 \quad (19)$$

$$\frac{\partial p}{\partial t} - \nabla \cdot (\mu_p p \nabla \phi_p) + r = 0 \quad (20)$$

$$-\nabla \cdot (b_n \nabla \sqrt{n}) - \frac{\sqrt{n}}{2}(\psi - \frac{kT}{q} \log \frac{n}{n_i} - \phi_n) = 0 \quad (21)$$

$$-\nabla \cdot (b_p \nabla \sqrt{p}) + \frac{\sqrt{p}}{2}(\psi + \frac{kT}{q} \log \frac{p}{n_i} - \phi_p) = 0 \quad (22)$$

The quantities in parentheses are negative fluxes. The solution variables for the above five equations are ψ, ϕ_n, ϕ_p, $\sqrt{n}$, and $\sqrt{p}$, respectively. The physical constants b_n and b_p are defined as:

$$b_n = \frac{\hbar^2}{4lqm_n^*} \quad (23)$$

$$b_p = \frac{\hbar^2}{4lqm_p^*} \quad (24)$$

where l can be considered to be either the space dimensionality or a fitting parameter; m_n^* (m_p^*) is the electron (hole) effective mass.

Required Field Definitions

This system requires netdope to be defined after the structure is created with grid and deposit statements.

An implementation of the silicon_dg5 system is given below:

```
system name=silicon_dg5
+   sysvars=psi,sqrt_n,sqrt_p,phi_n,phi_p
+   term0=ndiv_fbm.lapflux(psi|psi)@{silicon,poly,oxide}
+   term1=nodal.nscd(electrons,holes,netdope|psi)@{silicon,poly}
+   term2=constraint.continuity(psi|psi)@{silicon/oxide,poly/oxide,silicon/poly}
+   term3=dirichlet.dg_dirichlet(netdope|psi,phi_n,phi_p)@{silicon/gate,poly/gat
e,silicon/back,silicon/source,silicon/drain,oxide/gate,silicon/substrate}
+   term4=ndiv_fbm.diffusion(bn,sqrt_n|sqrt_n)@{silicon,poly}
+   term5=-0.5*nodal.prod(sqrt_n,dphi_n|sqrt_n)@{silicon,poly}
+   term6=dirichlet.default_dirichlet(0|sqrt_n,sqrt_p)@{silicon/oxide,poly/oxide}
+   term7=ndiv_fbm.diffusion(bp,sqrt_p|sqrt_p)@{silicon,poly}
+   term8=0.5*nodal.prod(sqrt_p,dphi_p|sqrt_p)@{silicon,poly}
+   term9=ndiv_fbm.nqfflux(psi,phi_n,tl,mobn,electrons,edge|phi_n)@{silicon,poly}
+   term10=ndiv_fbm.nqfflux(psi,phi_p,tl,mobp,holes,edge|phi_p)@{silicon,poly}
+   tmpvars=electrons,holes,phin0,phip0,dphi_n,dphi_p,bn,bp,mobn,mobp
+   func0=prod(sqrt_n,sqrt_n|electrons)@{silicon,poly}
+   func1=prod(sqrt_p,sqrt_p|holes)@{silicon,poly}
+   func2=phiMB(electrons,psi,tl|phin0)@{silicon,poly}
```

```
+   func3=phiMB(holes,psi,tl|phip0)@{silicon,poly}
+   func4=sub(phin0,phi_n|dphi_n)@{silicon,poly}
+   func5=sub(phip0,phi_p|dphi_p)@{silicon,poly}
+   func6=6.35e-9*inverse(mn|bn)@{silicon,poly}
+   func7=6.35e-9*inverse(mp|bp)@{silicon,poly}
+   func8=mob0(electrons,holes,tl,netdope|mobn,mobp)@{silicon,poly}
+   init0="tl:${solve/bias/Tcelsius}+${library/physics/celsius}"
+   init1="edge:1"
+   init2="sqrt_n:sqrt(electrons)"
+   init3="sqrt_p:sqrt(holes)"
+   init4="mn:0.19"
+   init5="mp:0.49"
```

6.5 silicon_dg5_lombardi

The "silicon_dg5_lombardi" system is similar to the "silicon_dg5" system except that it adds the Lombardi mobility model.

6.6 silicon_dg3

The "silicon_dg3" system solves for the three equation density gradient model.

$$\nabla \cdot (\varepsilon \nabla \psi) + q(p - n + N_D^+ - N_A^-) = 0 \tag{25}$$

$$-\nabla \cdot (b_n \nabla \sqrt{n}) - \frac{\sqrt{n}}{2}(\psi - \frac{kT}{q} \log \frac{n}{n_i} - \phi_n) = 0 \tag{26}$$

$$-\nabla \cdot (b_p \nabla \sqrt{p}) + \frac{\sqrt{p}}{2}(\psi + \frac{kT}{q} \log \frac{p}{n_i} - \phi_p) = 0 \tag{27}$$

The solution variables for the above three equations are ψ, $\sqrt{n}$, and $\sqrt{p}$, respectively. The physical constants b_n and b_p are defined as:

$$b_n = \frac{\hbar^2}{4lqm_n^*} \tag{28}$$

$$b_p = \frac{\hbar^2}{4lqm_p^*} \tag{29}$$

where l can be considered to be either the space dimensionality or a fitting parameter; m_n^* (m_p^*) is the electron (hole) effective mass.

The silicon_dg3 system can only be used in situations in which no currents are flowing. It is most often used to obtain the initial solution at zero bias to be used in a subsequent silicon_dg5 calculation,

for example. However, silicon_dg3 can also be used with applied bias for certain structures, such as a 1-D MOS capacitor. In that case, the most convenient approach is use a different material (e.g., poly) for the gate, and then set the following dbase parameter as follows:

```
dbase create name="/options/modulate_qf_with_bias" sval=''poly''
```

This dbase entry will set the fixed quasi-Fermi level of the poly equal to the bias applied to the electrode in the bias statement.

Required Field Definitions

This system requires netdope to be defined after the structure is created with grid and deposit statements.

```
system name=silicon_dg3
+       sysvars=psi,sqrt_n,sqrt_p,ex
+       term0=ndiv_fbm.lapflux(psi|psi)@{silicon,poly,oxide}
+       term1=nodal.nscd(electrons,holes,netdope|psi)@{silicon,poly}
+       term2=dirichlet.dg_dirichlet(netdope|psi)@{silicon/gate,poly/gate,silico
n/back,silicon/source,silicon/drain,oxide/gate,silicon/substrate}
+       term3=constraint.continuity(psi|psi)@{silicon/oxide,poly/oxide}
+       term4=ndiv_fbm.diffusion(bn,sqrt_n|sqrt_n)@{silicon,poly}
+       term5=-0.5*nodal.prod(sqrt_n,phin|sqrt_n)@{silicon,poly}
+       term6=ndiv_fbm.diffusion(bp,sqrt_p|sqrt_p)@{silicon,poly}
+       term7=0.5*nodal.prod(sqrt_p,phip|sqrt_p)@{silicon,poly}
+       term8=dirichlet.default_dirichlet(0|sqrt_n,sqrt_p)@{silicon/oxide,poly/o
xide}
+       term9=-1.602e-15*interface.copy(stchg|psi)@{silicon/oxide,poly/oxide}
+       term10=nodal.copy(ex|ex)@{silicon,poly,oxide}
+       term11=dnodal.deriv_x(psi|ex)@{silicon,poly,oxide}
+       tmpvars=electrons,holes,phin,phip,bn,bp
+       func0=prod(sqrt_n,sqrt_n|electrons)@{silicon,poly}
+       func1=prod(sqrt_p,sqrt_p|holes)@{silicon,poly}
+       func2=phiMB(electrons,psi,tl|phin)@{silicon,poly}
+       func3=phiMB(holes,psi,tl|phip)@{silicon,poly}
+       func4=6.35e-9*inverse(mn|bn)@{silicon,poly}
+       func5=6.35e-9*inverse(mp|bp)@{silicon,poly}
```

6.7 silicon_poisson_ex, silicon_dd_ex, silicon_dg3_ex, silicon_dg5_ex

These "ex" systems differ from their non -ex variants only by including the calculation of the x-component of the electric field, Ex. This term may be useful for capacitance calculations, mobility

calculations, etc.

7 Prophet Operators

7.1 Prophet Operators

Prophet has a large variety of built-in operators. These operators are used to construct the pre-defined systems described in the previous section. The same operators can also be used to construct new systems by the end-user.