SAND REPORT

SAND2008-xxxx
Unlimited Release
August 2008

Charon Parallel Semiconductor Device Simulator

User Manual, Version 1.0

Gary Hennigan, Deborah Fixel, Joseph Castro and Paul Lin

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of
Energy under Contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: <http://www.doe.gov/bridge>

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: <http://www.ntis.gov/ordering.htm>

Charon Parallel Semiconductor Device Simulator

User Manual, Version 1.0

Gary Hennigan, Deborah Fixel, Joseph Castro and Paul Lin
Electrical and Microsystems Modeling

Sandia National Laboratories
P.O. Box 5800
Mail Stop 0316
Albuquerque, NM 87185-0316

June 11, 2010

Abstract

This manual gives usage information for the **Charon** semiconductor device simulator. **Charon** was developed to meet the modeling needs of Sandia National Laboratories and to improve on the capabilities of the commercial *TCAD* simulators; in particular, the additional capabilities are running very large simulations on parallel computers and modeling displacement damage and other radiation effects in significant detail.

The parallel capabilities are based around the *MPI* interface which allows the code to be ported to a large number of parallel systems, including linux clusters and proprietary "big iron" systems found at the national laboratories and in large industrial settings.

Acknowledgements

The authors would like to thank the entire Sandia *QASPR* team and specifically the expertise of Ken Kambour, Ralph Young, Sam Myers, Chuck Hembree and Bill Wampler has been invaluable during the development of **Charon**. Additionally the *Nevada* toolkit developers provided, and continue to provide, great support, specifically Rich Drake. Finally we are grateful to the Xyce developers for providing us with a working format for this user manual.

Contacts

Gary Hennigan	glhenni@sandia.gov
Bug Reports ¹	http://charleston.sandia.gov/bugzilla_Charon
User Mailing List	charon-users@sandia.gov

DRAFT

¹ Accessible only on the Sandia National Laboratories internal restricted network.

This page is left intentionally blank

Table of Contents

1. Introduction	15
1.1 Drift Diffusion Equations.....	15
1.2 Nonlinear Poisson Equation	16
1.3 Defect Equations.....	16
1.4 Discretizations	19
2. Installing and Running	21
2.1 Building Charon	21
General Build Instructions	23
SNL System NWCC Spirit	27
SNL System Thunderbird	27
SNL System TLCC	27
SNL System Red Storm	28
2.2 Running Charon	28
Running on SNL system NWCC Spirit	31
Running on SNL system Thunderbird	31
Running on SNL system TLCC.....	31
Running on SNL system Red Storm	32
3. The Input File	35
3.1 General description	35
3.2 Physics-Specific Input File Reference Page for Charon Semiconductor Model and Material Properties.....	35
MATERIAL Description	37
MODEL Description	38
METAL Description	39
MPARAM Description	40
3.3 Physics-Specific Reference for Charon Semiconductor Physics.....	51
PHYSICS Description	51
SPECIES SOURCE Description	52
SCALING PARAMETERS Description	53
REGION Description	53
MESH REFINEMENT Description	54
CURRENT CONVERGENCE Description	56
STABILIZATION Control	56
BANDGAP NARROWING	58

	CALCULATE VELOCITIES	59
	SURFACE CHARGE	59
3.4	Physics-Specific Input File Reference Page for Charon Nonlinear-Poisson Physics	61
	OPERATOR SCALING Description	61
	DOPING Description	61
	INITIAL SPECIES DENSITY Description	62
3.5	Physics-Specific Input File Reference Page for Charon Semiconductor Drift-Diffusion Physics	62
	OPERATOR SCALING	62
	MOBILITY Description	62
	DIFFUSION Description	64
	TRAPS Description	65
	DEFECTS	68
	DOPING	69
	Initial Species Density	84
	Mole Fraction	94
	TRANSIENT SOURCE	95
	Constant Electric Field BC	100
	Silicon/Oxide Interface BC	101
	FEM Oxide Electric Field BC	102
	SGFVM Oxide Electric Field BC	103
	SRH Surface BC	104
	INTRINSIC CONCENTRATION	105
	Constant Current BC	106
	Contact Series Resistance	107
	INCOMPLETE IONIZATION	109
3.6	<i>Nevada</i> Input Description	109
3.7	Charon subsection of input file	110
	DEBUG Description	110
	STABILIZATION, DISCONTINUITY CAPTURE, and LUMPING Description	111
	OPERATOR SCALING Description	113
	JACOBIAN and FILL STATUS Description	113
	CODE TIMING Description	114
	BOUNDARY CONDITIONS Description	114
	VARIABLE MANAGER DATA OUTPUT	117
	TIME INTEGRATION Description	119
	SOLVER Description	122
	ANALYTIC SOLUTION Description	124
3.8	Default MPARAM Values	125
3.9	Future Capabilities	125

4. Examples	137
4.1 IV sweep for a simple one-dimensional diode	137
4.2 A two-dimensional Bipolar Junction Transistor	142
<i>BJT</i> under normal operating conditions	142
<i>BJT</i> with transient defects	148
A. Input Files for Example Problems	153
A.1 Input file for nonlinear Poisson simulation for PN-diode example problem.....	153
A.2 Case file for IV-sweep simulation of PN-diode example problem.....	156
A.3 Input-command file for nonlinear Poisson simulation for the example BJT problem.....	160
A.4 Templated input-command file for the example BJT problem	164
A.5 Doping input-command file for example BJT problem	167
A.6 Example input file for defects induced by pulse radiation	169
A.7 Example include file for doping, doping_analytic_species.inp	177
A.8 Example include file for defect properties, charon_prototype_defects.inp	179
A.9 Example include file for transient source rate, iv_time_profile.txt	181
A.10 Example include file for source depth distribution, iv_depth_profile.txt	182
A.11 Example chemkin input file, chemkin_prototype.inp	182
A.12 Input file for nonlinear energy simulation for PN-diode problem.....	192
A.13 Input file for energy transport simulation for PN-diode problem	196
B. Defect Reactions Implemented in Charon	203
B.1 Electron Capture	203
B.2 Electron Emission	203
B.3 Hole Capture	204
B.4 Hole Emission	205
B.5 Complex Recombination Charge-Charge	205
B.6 Complex Dissociation Charge-Charge	206
B.7 Complex Recombination Charge-Neutral	207
B.8 Complex Dissociation Charge-Neutral	208
B.9 Complex Recombination Neutral-Neutral	208
B.10 Complex Dissociation Neutral-Neutral	208
B.11 Annihilation Charge-Charge	208
B.12 Annihilation Neutral-Neutral	209
B.13 Annihilation Charge-Charge Create Electron.....	209
B.14 Annihilation Charge-Neutral Create Electron.....	210
B.15 Annihilation Charge-Charge Create Hole	210
B.16 Annihilation Charge-Neutral Create Hole	211

This page is left intentionally blank

List of Figures

2.1	Directory tree after unpacking the Charon distribution.	22
2.2	Relevant compiler specification from <i>gnu4.xml</i>	24
4.1	A simple pseudo-1D Silicon diode geometry.	137
4.2	Mesh for the problem geometry in Figure 4.1.	138
4.3	Cubit journal file used to generate the mesh illustrated in Figure 4.2.	139
4.4	Linux command for executing Charon run for example diode problem in Section 4.1.	140
4.5	Plot of the quantity $\log_{10}(\text{electron density})$ at bias $0.3V$ for the simple diode example problem in Section 4.1.	142
4.6	Electrical current at the contacts of the example diode problem in Section 4.1.	143
4.7	The geometry of a two-dimensional bipolar junction transistor.	143
4.8	Cubit journal file used to generate the mesh for the small <i>BJT</i> example problem geometry of Figure 4.7.	144
4.9	Doping concentrations starting at the top of the device under the emitter contact.	146
4.10	Circuit diagram for <i>BJT</i> problem.	147
4.11	Unix command for executing Charon run for example <i>BJT</i> problem in Section 4.2.	148
4.12	Gummel plot for the <i>BJT</i> illustrated in Figure 4.7 under normal operating conditions.	149
4.13	Current gain for the <i>BJT</i> illustrated in Figure 4.7 under normal operating conditions.	150

This page is left intentionally blank

List of Tables

2.1	Prerequisites for the installation of Charon via source build.	21
2.2	Table of compile options regularly used by Charon developers.	25
2.3	Typical file naming convention.	29
2.4	List of options for the <i>runCharon</i>	34
3.2	Notation used in this chapter	36
3.4	Table of Material Properties	126
3.5	Arora Mobility Model Parameters	127
3.6	Analytic Mobility Model Parameters	128
3.7	III-V Compound Semiconductor Analytic Mobility	129
3.8	Carrier-Carrier Scattering Mobility Model Parameters	130
3.10	Lombardi Surface Mobility Model Parameters	131
3.11	Philips Unified Mobility Model Parameters	132
3.12	Lucent Mobility Model Parameters	133
3.13	Intrinsic Concentration Calculation Parameters.	134
3.14	Incomplete Ionization Model Parameters	135
3.15	Impact Ionization Model Parameters	136
4.1	Summary of the input files names for example <i>PN</i> -diode problem in Section 4.1.	140
4.2	Options for <i>iv-generate.pl</i>	141
4.3	Summary of the input files names for the <i>BJT</i> example problem in Section 4.2.	145

This page is left intentionally blank

1. Introduction

The **Charon** semiconductor device simulator is based on a multi-physics code for simulating general transport-reaction phenomena. The focus of this manual is on **Charon**'s capabilities for modeling semiconductor devices. The semiconductor modeling capability in **Charon** was developed to work in a manner similar to other *TCAD* codes such as Medici[™] and DaVinci[™]. Additionally **Charon** supports massively parallel execution and the explicit treatment of defect physics in order to accurately model radiation effects.

Generally the current release is restricted to junction-type devices, such as diodes and bipolar junction devices. There is some preliminary functionality for modeling MOSFET devices, but it should be viewed as alpha-level.

Another important note is that as of this writing **Charon** has *not* been approved for external distribution. As such users must treat the **Charon** code as *Export Controlled* material.

1.1 Drift Diffusion Equations

Charon was developed to solve the drift-diffusion equations for semiconductor devices. These equations are

$$-\nabla \cdot \epsilon \nabla \psi = q(p - n + C), \quad (1.1)$$

$$\nabla \cdot \mathbf{J}_n - qR = q \frac{\partial n}{\partial t}, \quad (1.2)$$

$$-\nabla \cdot \mathbf{J}_p - qR = q \frac{\partial p}{\partial t}, \quad (1.3)$$

with

$$\mathbf{J}_n = -qn\mu_n \nabla \psi + qD_n \nabla n, \quad (1.4)$$

$$\mathbf{J}_p = -qp\mu_p \nabla \psi - qD_p \nabla p, \quad (1.5)$$

$$C = N_d - N_a, \quad (1.6)$$

where

- ψ - Electric potential,
- n, p - Electron and hole concentrations, respectively,

- $N_{d,a}$ - The donor and acceptor doping concentrations, respectively,
- q - The fundamental electron charge (1.602×10^{-19}),
- R - Generation term,
- $\mu_{n,p}$ - Carrier mobilities,
- $D_{n,p}$ - Carrier diffusion coefficients, and
- ϵ - Electric permittivity of semiconductor.

1.2 Nonlinear Poisson Equation

The nonlinear Poisson equation uses an equilibrium approximation to the drift-diffusion equations to eliminate the flux equations in (1.2) and (1.3). For the Poisson equation, (1.1), the carrier densities, p and n , are approximated by

$$n = n_i e^{\frac{\psi}{kT}}, \quad (1.7)$$

$$p = n_i e^{-\frac{\psi}{kT}}. \quad (1.8)$$

Now (1.1) becomes

$$-\nabla \cdot \epsilon \nabla \psi = q \left[n_i \left(e^{-\frac{\psi}{kT}} - e^{\frac{\psi}{kT}} \right) + C \right]. \quad (1.9)$$

1.3 Defect Equations

Charon can treat defects explicitly in a drift-diffusion framework to accurately simulate the effects of such defects. In addition to the drift-diffusion equations in (1.1)–(1.3) there is an additional equation added for each defect of the form

$$-\nabla \cdot \mathbf{J}_i - q_i R_i = q_i \frac{\partial X_i}{\partial t}, \quad (1.10)$$

where

$$\mathbf{J}_i = -q_i \mu_i X_i \nabla \psi - q_i D_i \nabla X_i \quad (1.11)$$

where

- J_i - the current density of defect species i ,
- R_i - the summation of recombination/generation rate terms of defect species i ,
- X_i - the concentration of defect species i ,
- D_i - the diffusion coefficient of defect species i ,
- μ_i - the mobility of defect species i , and
- $q_i \equiv Z_i q$ - where Z_i is the integer charge number of defect species i .

The defect mobility and diffusivity are related by Einstein's relationship via

$$D_i = \frac{kT}{q_i} \mu_i \quad (1.12)$$

where

- k - Boltzmann's constant ($1.381 \times 10^{-23} J/K = 8.617 \times 10^{-5} eV/K$) and
- T - lattice (material) temperature.

The diffusion of interstitials and vacancies is assumed to be independent of concentration and thus with Fick's Law the diffusivity may be expressed as

$$D_i = D_{0i} \exp\left(-\frac{E_{Ai}}{kT}\right) \quad (1.13)$$

where

- D_{0i} - diffusion coefficient extrapolated to infinite temperature (diffusion pre-factor) of defect species i , and
- E_{Ai} - activation energy of defect species i .

Within **Charon**, the diffusivity of a defect is calculated with (1.13) and then the mobility is calculated via Einstein's relation within (1.12).

Additionally, if the defects are charged then a modification of Poisson's equation, (1.1), is necessary. Poisson's equation becomes

$$-\nabla \cdot \epsilon \nabla \psi = q(p - n + C) + \sum_{i=1}^n q_i X_i \quad (1.14)$$

Note that if the dopants are specified as defect species, then C within (1.14) may be set to zero. Specifying the dopants as defect species is a dynamic treatment of the dopants while specifying C is a static treatment of the dopants.

The source terms, R_i , in (1.10) consist of a set of reactions associated with the capture and emission of the holes and electrons by the specific defect species. **Charon** separates the emission and capture into two reactions. For example consider the capture of an electron by a defect species

The corresponding source term representation would be

$$R(X_j^{m+1}, e^-) = \sigma(X_j^{m+1}, e^-) \nu_n n X_j^{m+1}, \quad (1.16)$$

where

- $\sigma(X_j^{m+1}, e^-)$ – electron capture cross-section for species X_j^{m+1} ,
- ν_n – electron thermal velocity within material (for Si = 2.3e7 cm/s @ 300K),
- n – the electron density,
- X_j^{m+1} – the density of defect species X_j^{m+1} with charge state $m + 1$,

and the actual source terms going into the drift diffusion equations, (1.10), would be

$$R(X_j^{m+1}) = -R(X_j^{m+1}, e^-), \quad (1.17)$$

$$R(e^-) = -R(X_j^{m+1}, e^-), \quad (1.18)$$

$$R(X_i^m) = R(X_j^{m+1}, e^-). \quad (1.19)$$

The emission of an electron is represented by the reaction

The source term associated with the reaction in (1.20) is

$$R(X_j^{m+1} \mapsto e^-) = \sigma(X_j^{m+1}, e^-) \nu_n N_c X_j^m \frac{\gamma(X_j^{m+1})}{\gamma(X_i^m)} \exp\left(\frac{E_{X_i^m} - E_c}{kT}\right) \quad (1.21)$$

where

- ▶ N_c – effective density of states in the conduction band (for Si = $2.86 \times 10^{19} \text{ cm}^{-3}$ @ 300K)
- ▶ $\gamma(X_j^{m+1}), \gamma(X_i^m)$ – state degeneracies for X_j^{m+1} and X_i^m , respectively,
- ▶ $E_{X_i^m}$ – energy of the electron state associated with X_i^m ,
- ▶ E_c – conduction band minimum energy,
- ▶ $E_{X_i^m} - E_c \equiv \Delta E_e^{X_i^m}$ – activation energy for electron emission of species X_i^m ,

and the actual source terms going into the drift diffusion equations, (1.10), would be

$$R(X_j^{m+1}) = -R(X_j^{m+1} \mapsto e^-) \quad (1.22)$$

$$R(e^-) = R(X_j^{m+1} \mapsto e^-) \quad (1.23)$$

$$R(X_i^m) = R(X_j^{m+1} \mapsto e^-) \quad (1.24)$$

Note that each different charge state for a species makes for a new species within **Charon**. For example X_i^m and X_j^{m+1} are represented as two separate defect species within **Charon** and so each would have it's own drift-diffusion equation of the form in (1.10).

A listing of additional reactions that are implemented in **Charon** can be found in Appendix B.

1.4 Discretizations

Charon uses two common discretizations for solving the PDEs associated with the semiconductor drift-diffusion model given above. The finite-element method, or *FEM*, and the “box” method used by many of the commercial *TCAD* simulators. In general the finite-element method offers a richer mathematical context in which to solve partial-differential equations such as the equations in the drift-diffusion model for semiconductor problems, but the *FEM* is much less mature than the box method. In terms of capabilities the box method is the preferred method for solving semiconductor problems in **Charon**. The box method can typically use a coarser discretization for an equivalent problem and thus runs to completion quicker. A good introduction to the box method can be found in [1].

A future technical report will detail the finite-element discretization that is implemented in **Charon**.

This page is left intentionally blank

2. Installing and Running

Charon has been installed and executed on numerous systems at Sandia National Laboratories (*SNL*), including Linux workstations, Linux clusters, and large-scale, proprietary systems available at *SNL* and other national laboratories. The prerequisites for installation of **Charon** are given in Table 2.1.

Requirement	Version	Information
Python	2.4	Used by numerous scripts in the Nevada toolkit during and after installation. Can be obtained via http://www.python.org/download .
Modern compilers	<i>various</i>	You'll need C++, FORTRAN and C compilers. A modern C++ compiler is required with support for templating. Compiler suites known to work: GNU 3.4, GNU 4.2, Intel 10.x
GNUMake	3.81	The GNU make utility. Can be obtained via http://www.gnu.org/software/make .
MPI	1.27	Charon requires an <i>MPI</i> installation in order to install and execute successfully, even if running on a single processor. Both <i>MPICH</i> and <i>Open MPI</i> are known to work, but <i>Open MPI</i> is the preferred solution and can be obtained via http://www.open-mpi.org .

Table 2.1. Prerequisites for the installation of **Charon** via source build. Note that version numbers are minimum. Newer versions of the prerequisites may also work.

2.1 Building Charon

In order to install **Charon** it is assumed that you have a general working knowledge of Unix-like platforms and have a relatively modern system on which to install **Charon**.

The **Charon** distribution consists of the necessary source code and tools to build the exe-

cutable from source. When building from source ensure that your system has the prerequisites as listed in Table 2.1.

The actual distribution of **Charon** consists of a compressed *tarball* that contains all the necessary source code, installation tools and regression tests. The tarball is named `charon_distrib_VERSION.tar.gz`, where `VERSION` is either a numeric version number, such as `1.0`, or a date such as `23Oct2008`. In the former case this is an officially released distribution of **Charon**. In the latter case this is a development distribution and users should be aware that such distributions are not tested and in fact may not even build. Development distributions are intended for users that need the latest capabilities and are willing to risk an untested and unverified version of **Charon**.

Once you have obtained the tarball you can unpack it via the command:

```
tar xzf charon_distrib_1.0.tar.gz
```

for example. Note that the command above relies on the fact that you're using GNU *tar* and have the GNU *gzip* utility installed as well. Figure 2.1 illustrates the first three levels of the directory tree that is created once the extraction is complete.

Figure 2.1. Directory tree after unpacking the **Charon** distribution.

The *src* subdirectory contains both the **Charon** and *Nevada* source code. The *tpl* subdirectory contains third-party libraries necessary for building **Charon**. The *doc* subdirectory should contain the LaTeX source and *PDF* copy of this user manual. The *tests* subdirectory contains a set of tests run regularly as a part of **Charon**'s regression testing. This is available to test the **Charon** installation upon completion of the install process. Finally,

the *toolset* subdirectory contains tools necessary to build **Charon** as well as *XML* files that define the build environment.

General Build Instructions

The process to build **Charon** is mostly independent of the platform on which it is being built. This process relies on *Python* scripts and associated *XML* files to specify things like compilers and the *MPI* environment specific to a particular platform. All of this is contained within the *toolset* subdirectory. Within the *toolset* directory there are two subdirectories where *XML* files that define your environment reside. The first is the *platforms* subdirectory. In this directory are “plugin” files that define parameters for the operating system on which to install **Charon**. The other directory is *compilers*. Within that directory are files that define the compilers and *MPI* environment for your platform. Occasionally you may need to modify a particular file in this directory to specify, for example, the path to your compilers. If you’re installing **Charon** on a 32-bit Linux workstation with the GNU compilers then the appropriate *XML* file would be *gnu4.xml*. Within that file you’ll find a section similar to the one shown in Figure 2.2.

As previously stated, all the required tools are contained in the *toolset* subdirectory of the distribution. The first step in the installation process is to add this subdirectory to your *PATH* environment variable. Under a Bourne shell this can be accomplished via the command

```
export PATH=$PATH:/home/user/Charon_distrib_1.0/toolset
```

for example. For a C-shell environment the command would be

```
set path = ($path /home/user/Charon_distrib_1.0/toolset).
```

To verify that your path is correct you can use

```
which build
```

The output should be

```
/home/user/Charon_distrib_1.0/toolset/build.
```

```

<if_option value="not serial">

  <CPPFLAGS> -DCODE_MP </CPPFLAGS>

  <if_option value="not gm and not lam and not lamgm and not sgi">
    <requires set="MPICHBIN">
      /usr/local/mpich-1.2.7/bin
      /usr/bin
    </requires>
    <CC> $(MPICHBIN)/mpicc </CC>
    <CXX> $(MPICHBIN)/mpicc </CXX>
    <F77> $(MPICHBIN)/mpif77 </F77>
    <CPP> $(MPICHBIN)/mpicc -E </CPP>
    <FPP> $(MPICHBIN)/mpif77 -E </FPP>
    <CPPFLAGS> -DMPICH_SKIP_MPICXX </CPPFLAGS>
    <CPPFLAGS> -DOMPI_SKIP_MPICXX </CPPFLAGS>
  </if_option>

</if_option>

```

Figure 2.2. Relevant compiler specification from *gnu4.xml*.

The next step is to build the third-party libraries required for **Charon** and *Nevada*. To begin this process do

```

cd /home/user/Charon_distrib_1.0/tpl/nevada
buildTPL -t /home/user/Charon_distrib_1.0/src/nevada/nevada.xml \
-o gnu4+opt

```

where `<version>` refers to the version of Trilinos distributed with **Charon**. There is only one version distributed with **Charon** and the version is just a subdirectory within the directory shown above. `buildTPL` is the *Nevada* build utility for the associated third-party libraries. The `-t` option tells `buildTPL` where to look for the file which controls what libraries and what versions of those libraries are needed in order to build **Charon**. This information is contained within *XML* files distributed throughout the source code for **Charon** and *Nevada*. The `-o` option controls which specific build options you wish to apply. In this case, assuming the install is being performed on a generic Linux workstation, the libraries will be built using the GNU 4.x compilers with optimization options turned on. A list of supported

compiler options regularly used by the **Charon** team is shown in Table 2.2. By default the parallel versions of these libraries will be built using the *MPI* compiler wrappers, e.g., `mpiCC`. Note that many of the *Nevada* tools accept the `-h` option which outputs help and available options for that particular tool.

Option	Typical Environment
<i>gnu4</i>	The GNU 4.x compiler suite on Linux
<i>intel10</i>	The Intel 10.x compiler suite on Linux
<i>64BITgnu4</i>	The GNU 4.x compiler suite on 64-bit (x86_64) Linux
<i>nwcc</i>	The SNL NWCC spirit parallel system
<i>tbird</i>	The SNL thunderbird parallel system
<i>tlccintel</i>	The SNL TLCC parallel system (Open MPI)
<i>tlccmvapichintel</i>	The SNL TLCC parallel system (MVAPICH)
<i>storm</i>	The SNL Red Storm parallel system
<i>purple</i>	The LLNL Purple parallel system

Table 2.2. Table of compile options regularly used by **Charon** developers.

The next step is to build some additional tools and install them in a standard location. This is accomplished with the *Nevada* build script `buildTPE`. The invocation of the script is

```
buildTPE -v /home/user/Charon_distrib_1.0/src/nevada/etc \
--prefix=/home/user/Charon_distrib_1.0 -o gnu4+opt+serial
```

The `-v` option is used to specify the location of the `ExecutableVersions.py` file. This file specifies which utilities and associated versions of those utilities to build. The `--prefix` option specifies where to install the resulting utilities. The `-o` option again specifies the specific build options to use. Note that here the *serial* option has been added. While **Charon** does not support the *serial* option the utilities will typically be used on a serial workstation or the login node of a parallel system and should be built as serial utilities. Upon completion `buildTPE` will install the resulting utilities in an installation-specific subdirectory. On a Linux workstation, for example, the utilities are installed in the directory

```
/home/user/Charon_distrib_1.0/bin/Linux
```

After installation the `PATH` environment variable should be modified to include this location.

The next step is to build the third-party libraries specific to **Charon**. These are stored in the directory `Charon_distrib_1.0/tpl/charon`. The necessary steps are

```
cd /home/user/Charon_distrib_1.0/tpl/charon
buildTPL -v Resources \
-a /home/user/Charon_distrib_1.0/tpl/nevada/TPL.xml -o gnu4+opt
```

The `-a` options specifies the location of the *Nevada*-specific third-party libraries on which some of the **Charon**-specific third-party libraries depend. The `-v` options specifies the location of the `LibraryVersions.py` file which contains specific version information for each of the third-party libraries.

The last step in the build process is to compile the **Charon** code itself. This code is contained in the `src` directory. The commands to build the **Charon** executable are

```
cd /home/user/Charon_distrib_1.0/src
build -a /home/user/Charon_distrib_1.0/tpl/nevada/TPL.xml \
-a /home/user/Charon_distrib_1.0/tpl/charon/TPL.xml \
-o gnu4+opt+2D
```

Note the addition of the 2D build option. **Charon** doesn't currently support anything but two-dimensional operation and so this build option is always required. Upon successful completion a **Charon** executable will be saved in the directory

```
/home/user/Charon_distrib_1.0/src/bin
```

The executable has a name which reflects the build options used to create it. Using the above process the executable would be `charon_2D_gnu4_opt.x`, for example.

The following sections concern building **Charon** on the large SNL machines: NWCC Spirit, Thunderbird, TLCC, and Red Storm. Building **Charon** on these platforms is almost the same as described under the general build instructions. The difference is due to the fact that multiple build environments are available on these platforms. One needs to make sure that the same build environment is used to build the third-party libraries and the charon executable, and when running the **Charon** executable. Changing build environments is accomplished by using the *module* command. Issue *module list* to see the currently loaded modules. Using NWCC Spirit as an example, as of November 27, 2008, the default MPI module is `mpi/openmpi-1.2.2_mx_intel-9.1-f040-c045`. It is recommended

to build **Charon** with the `mpi/openmpi-1.2.7_mx_intel-10.1-f015-c015` module, so one would change to this module by using the following command:

```
module swap mpi/openmpi-1.2.2_mx_intel-9.1-f040-c045 \  
            mpi/openmpi-1.2.7_mx_intel-10.1-f015-c015
```

If `mpi/openmpi-1.2.2_mx_intel-9.1-f040-c045` is not the default MPI module then this will have to be replaced by the default MPI module in the `module swap` command. Issue `module list` to see the list of available modules. Also useful are `module load <module>` and `module unload <module>` to load or unload specific modules. Note that the following listings of specific versions of software for building **Charon** is likely to be obsolete by the time this manual reaches the users. The user should always check the available and default build environments.

SNL System NWCC Spirit

It is recommended to use Open MPI with the Intel compiler, for example Open MPI 1.2.7 with Intel 10.1 compiler:

```
module swap mpi/openmpi-1.2.2_mx_intel-9.1-f040-c045 \  
            mpi/openmpi-1.2.7_mx_intel-10.1-f015-c015
```

Build **Charon** as described in the general build instructions but using the `nwcc` option rather than `gnu4`.

SNL System Thunderbird

It is recommended to use Open MPI with the Intel compiler, for example Open MPI 1.2.7 with Intel 10.1 compiler. As of November 2008, this is the default environment so one would not have to do anything. The user should check to see if this is still the case. Build **Charon** as described in the general build instructions, but using the `tbird` option rather than `gnu4`.

SNL System TLCC

As TLCC is still in testing and is not yet in production (scheduled to go into production in January 2009), the information in this section should be verified before building on TLCC.

As of November 2008, the default build environment is Open MPI with the PGI 7 compiler. The PGI 7 compiler has bugs in it that make it unusable for **Charon**, so it is recommended to use the Intel compiler instead. There is a scaling issue with Open MPI, so for users who need to run more than 1500 processors, it is recommended to use MVAPICH instead. For jobs with 512 or fewer processors, Open MPI is fine. There are two options available for TLCC, *tlccintel* and *tlccmvapichintel*, the former for Open MPI and the latter for MVAPICH. Both are for the Intel compiler. Build **Charon** as described in the general build instructions but using the *tlccintel* or *tlccmvapichintel* option rather than *gnu4*.

SNL System Red Storm

One has a choice between building on a Red Storm login node or cross-compiling on reddish. As of November 2008, the default compiler on a Red Storm login node is PGI 7.1.4. But it is recommended that one use the PGI 6.2.5 compiler, so one would need to downgrade the compiler, i.e. `module swap pgi/7.1.4 pgi/6.2.5`. The PGI 7 compiler has bugs in it that will crash an optimized build of **Charon** at run time. It is necessary to load the `gmalloc` and `iobuf` modules, i.e. `module load gmalloc iobuf`. Build **Charon** as described in the general build instructions but using the *storm* option rather than *gnu4*. The only caveat is that the `buildTPE` command needs to be run with the compilers for the service node and not for the compute nodes. The `nevada` and `charon` third-party libraries as well as the `charon` executable need to be built for the compute nodes, which is the default build environment on a Red Storm login node.

2.2 Running Charon

For a **Charon** semiconductor run you generally need four input files. An input mesh and command file for the nonlinear Poisson problem which is used as an initial guess for the full drift-diffusion simulation, and the input mesh and command file for the drift-diffusion run itself. The input mesh file needed by the nonlinear Poisson (*NLP*) and the drift-diffusion simulations are identical and in fact you can simply make one a logical link to the other.

If you're performing the simulation(s) on a single processor of a workstation you can run **Charon** using the native *MPI* command `mpirun`. A typical sequence would be:

```
mpirun -np 1 ~/Charon_distrib_1.0/src/bin/charon_2D_gnu4_opt.x diode.nlp
mpirun -np 1 ~/Charon_distrib_1.0/src/bin/charon_2D_gnu4_opt.x diode.dd
```

This assumes a specific naming convention for the input files. The names of the files should conform to that shown in Table 2.3. Also in Table 2.3 you can see the names of the output Exodus II mesh files. Both the input *.gen and output *.exo files are in the Exodus II format [2]. The input mesh files are typically generated with a mesh-generation package such as *Cubit* [3]. The output mesh files contain the geometry as well as results calculated by **Charon**. These results can be visualized and manipulated by any tool that can read Exodus II files. At Sandia National Laboratories, commonly used applications for this purpose are *Ensign* [4] and *ParaView* [5].

File Name	Purpose
diode.nlp.gen	Input Exodus II mesh file for the nonlinear Poisson simulation
diode.nlp.inp	Input command file for the nonlinear Poisson simulation.
diode.dd.gen	Input Exodus II mesh file for the drift-diffusion simulation. Note this is identical to the input mesh file for the nonlinear-Poisson simulation.
diode.dd.inp	Input command file for the drift-diffusion simulation.
diode.nlp.exo	Output Exodus II mesh file with nonlinear-Poisson simulation results.
diode.dd.exo	Output Exodus II mesh file with drift-diffusion simulation results.

Table 2.3. Typical file naming convention.

Another option for running the simulations is to use the `runCharon` script provided with the **Charon** distribution in the *tool/set* subdirectory. It can optionally run both the *NLP* simulation and drift-diffusion simulation for you. Additionally you can use it to break the problem into pieces for execution on multiple processors. For the example given above the corresponding `runCharon` invocation would be

```
runCharon -p 1 -o gnu4+opt+2D \
-j /home/user/Charon_distrib_1.0/src diode
```

Again there will be the two output Exodus II files referenced in Table 2.3. Additionally there are two output files that capture the screen output. `text_output-nlp` contains the screen output for the *NLP* portion of the run and `text_output-dd` contains the screen output for the drift-diffusion portion of the run. In order to run on more than one processor the only

modification necessary to the above command is to change the `-p 1` argument to `-p <n>` where `<n>` is the number of processors on which you wish to run.

For parallel execution you will need to insure that you have

```
/home/user/Charon_distribution_1.0/bin/Linux,
```

for example, in your `PATH` environment variable. The only difference when running in parallel is the automatic creation of a directory which contains the pieces of the problem for each processor. Typically these pieces are stored in a subdirectory named “01” off the directory from which you invoked the `runCharon` script. Files in there have extensions specific to the run. For example, a file named `diode.dd.par.2.1` is the input Exodus II mesh file for processor ID 1 in a 2-processor run. A file named `diode.dd.exo.2.0` is the output Exodus II file for processor ID 0 in a 2-processor run. These files are all self-contained Exodus II files and can be viewed via the aforementioned methods. Upon completion of a parallel run `runCharon` will automatically combine all the parallel result files into a single file in the run directory with the expected file name, i.e., `diode.dd.exo`. Information on the “01” subdirectory is informational only, as generally the files contained in it are created automatically via *Nevada*. If mesh refinement is turned on, rather than having output files such as `diode.nlp.exo` and `diode.dd.exo`, the output files will be `diode.nlp.hat.<time step>` and `diode.dd.hat.<time step>`, where each file will store the mesh and corresponding solution for one time step. The `runCharon` script is self-documenting and help can be obtained by invoking the script with the `-h` option. The available options are also summarized in Table 2.4.

The last common mode of operation for **Charon**-semiconductor is generating current-voltage, or *I/V*, sweeps. For an in-depth treatment of this mode of operation see the example in Section 4.1.

The following concern running **Charon** on the large Sandia machines. The main difference is that these machines use a batch system, so one needs to submit jobs via a script to the batch queue. In order to run on the *SNL* NWCC Spirit, Thunderbird, or TLCC capacity machines, one needs an HPC Estimations & Requirements Tool (HERT) ID for the capacity machines. For Red Storm, one needs a HERT ID for the capability machine. This can be obtained from <https://computing.sandia.gov/hert>.

Running on *SNL* system NWCC Spirit

The main mode for running on NWCC Spirit is through the batch queue. You'll need to use a script such as the following (it is assumed that `/home/user/Charon_distrib_1.0/toolset` and `/home/user/Charon_distrib_1.0/bin/Linux` are in your `PATH` from the build process):

```
#!/bin/bash
#PBS -l nodes=1:ppn=2:compute,walltime=1:00:00
#PBS -N charon_\#1
#PBS -A <HERT ID>
#PBS -joe

source /opt/modules/default/init/bash
module swap mpi/openmpi-1.2.2_mx_intel-9.1-f040-c045 \
           mpi/openmpi-1.2.7_mx_intel-10.1-f015-c015

runCharon -p 2 -o charon+2D+opt+nwcc \
          -j /home/user/Charon_distrib_1.0/src diode
```

The above script runs a two processor job for a maximum of one hour. The HERT ID should look something like "FY091111", so the line would look like `#PBS -A FY091111`. NWCC Spirit uses the PBS batch system, so this script will need to be submitted using `qsub`, i.e. if the above script is named `nwcc.pbs`, submit this using `qsub nwcc.pbs`. The `module swap` command in the script should be the same as the one you used to build Charon.

Running on *SNL* system Thunderbird

Thunderbird also uses the PBS batch system, so the batch script is almost identical to the one for NWCC Spirit.

Running on *SNL* system TLCC

TLCC uses SLURM, so the batch script will be different. Another difference is that each TLCC compute node has 16 processor cores, so when one requests and gets allocated a node, one has access to 16 processor cores.

```
#SBATCH --account=<HERT ID>
#SBATCH --job-name charon_\#1
#SBATCH --time=1:00:00
#SBATCH --nodes=1 # number of compute nodes
#SBATCH --tasks-per-node=16

module swap mpi/openmpi-1.2.7_ofed_pgi-7.2-3 \
           mpi/openmpi-1.2.7_ofed_intel-10.1-f015-c015

let size=$SLURM_JOB_NUM_NODES*$SLURM_NTASKS_PER_NODE

export NUMA_OPT=/projects/charon/tlcc/numa_wrapper-16ppn

runCharon -p $size -o charon+2D+opt+tlccintel \
          -j /home/user/Charon_distrib_1.0/src diode
```

This example requests one compute node, but then invokes runCharon with 16 MPI processes (using all the processor cores on a node). Because TLCC uses SLURM, this script will need to be submitted using sbatch, i.e. if the above script is named tlcc.slurm, submit this using sbatch tlcc.slurm. The module swap command in the script should be the same as the one you used to build Charon.

Running on SNL system Red Storm

Red Storm batch scripts can use PBS batch script syntax. However, in contrast to NWCC Spirit and Thunderbird, size is used instead of node to specify the number of compute nodes. A second issue is that Red Storm has multicore processors, and the number of cores per node is specified with the -C flag. So one can specify up to 2 for the dual core nodes and up to 4 for the quad core nodes.

```
#!/bin/bash
#PBS -l size=2,walltime=1:00:00
#PBS -N charon_\#1
#PBS -A <HERT ID>
#PBS -joe

let ppn=2
let size=$PBS_NNODES*$ppn
```



```
runCharon -p $size -C $ppn -o charon+2D+opt+storm \  
          -j /home/user/Charon_distrib_1.0/src diode
```

For this example, two compute nodes are requested, and two cores per compute nodes will be used so there will be four MPI processes. Note that Red Storm compute nodes do not support dynamic linking so no `module swap` command is in the script.

Option	Purpose
-h	Outputs help for the script, including this list of options.
-m <i>machine</i>	Parallel platform to run the job on.
-n <i>executable name</i>	Product executable name.
-p <i>number of processors</i>	The number of processors to run on.
-P	(Red storm specific) Specify -VN on Red storm.
-f <i>number of concurrent I/O files</i>	Number of output files that can be written concurrently. Default is the number of processors.
-j <i>path to Charon/Nevada source</i>	Always required.
-o <i>build options</i>	The list of build options for the executable to run.
-r lbw nlp dd explicit_runid	Causes the script to run only the specified simulation. Normally the script first runs a <i>NLP</i> simulation then a drift-diffusion. This overrides the default behavior.
-w read=<...>	If the load-balance weights file was generated by a previous <i>runCharon</i> run, use it. (see the output of <i>runCharon -h</i> for more information.
-b alegrabal noalegrabal	Default behavior is to run the <i>alegrabal</i> script to generate the parallel input files for a parallel run. You can turn this off with the <i>noalegrabal</i> option here if, for example, the files already exist.
-C concat noconcat	Default behavior is to combine all the parallel results files into a single scalar file for post-processing. You can override this default behavior by specifying the <i>noconcat</i> option here.

Table 2.4. List of options for the *runCharon*.

3. The Input File

3.1 General description

The input file consists of two broad categories of commands or parameters, those specific to **Charon** and those used for any application that is based on the *Nevada* toolkit. Unless otherwise noted, or the parameter is shown in quotes, the input parameters are case insensitive. The convention for this manual is to show parameters in uppercase unless that parameter is case sensitive. Also, lines in the input file are often too long to show on a single line in this manual. To facilitate their inclusion in the manual long lines are broken into multiple lines with a “\”. Any input-file line ending with a “\” in this manual should be a single line in the actual input-command file.

Most of the documentation of the input-command file given in this section is available to Sandians online via the URL <http://charleston.sandia.gov/Charon/Charon/Doxygen/inputfile.html>.

Before continuing with this section it is important to become familiar with the notation used in documenting the input-command file. The notation is summarized in Table 3.2.

In general it is easier to take an existing input-command file and modify it for your particular problem than it is to create a new input-command file. To facilitate this the **Charon** distribution includes a selection of the regression tests used by the **Charon** developers. You can find an input file among these tests that as closely as possible reflects your problem and modify to fit your specific needs.

3.2 Physics-Specific Input File Reference Page for **Charon** Semiconductor Model and Material Properties

Notation	Example	Description
VERBATIM TEXT	PERMITTIVITY	Commands, or parameters, in the input file. Although most parameters are case insensitive, unless otherwise noted, the convention in this manual is to show them in upper-case verbatim text.
[VERBATIM TEXT]	[HOLE LIFETIME]	An input file parameter enclosed in square brackets indicates an optional parameter. Either the parameter is not needed for a particular run or the code will use a default value for such parameters unless such parameters are specified by the user in the input file.
<text in angle brackets>	<OPTION 1 option 2>	The angle brackets denote the actual arguments to the input-file parameters. In some cases this may be a number and in others it may be a reference to a <i>METHOD FUNCTION</i> . A “ ” symbol means that the argument can take on one of several possible values.
METHOD FUNCTION <n>	DOPING = METHOD FUNCTION 10	A <i>METHOD FUNCTION</i> is a generic interface to functions implemented within Charon . They are used frequently for values associated with parameters that have some type of variation in the problem.

Table 3.2. Notation used in this chapter

MATERIAL Description

The Material command allows the user to specify a specific material and model. Typically a material is defined within a given element block within the input.

```
MATERIAL <material_number> <material_name>  
    MODEL <model_number>  
END
```

where:

- ▶ **material_number** - an integer identifying this material, used as a reference in other input specifications.
- ▶ **material_name**:
 - ▷ SI - Silicon
 - ▷ GAAS - Gallium Arsenide
 - ▷ POLYSILI - Polysilicon
 - ▷ S.OXIDE - Silicon Dioxide
 - ▷ GERMANIU - Germanium
 - ▷ NITRIDE - Silicon Nitride
 - ▷ SAPPHIRE - Sapphire
 - ▷ ALGAAS - Aluminum Gallium Arsenide
 - ▷ INGAAS - Indium Gallium Arsenide
 - ▷ ALINAS - Aluminum Indium Arsenide
 - ▷ GAASP - Gallium Arsenide Phosphide
 - ▷ INGAP - Indium Gallium Phosphide
 - ▷ INASP - Indium Arsenide Phosphide
- ▶ **model_number**: an integer identifying a model, used as a reference in the MODEL specification

MODEL Description

The MODEL command allows the user to specify properties for a given model. This model is related to a particular material with the Material command. The only parameters that are required are DOPING and MATERIAL NAME. All other parameters are optional and set to the appropriate default values.

```
MODEL <model_number> CHARON SEMICONDUCTOR
  MATERIAL NAME = "<string>"
  [ RELATIVE PERMITTIVITY      = <method_function_ref|constant> ]
  [ ELECTRON MOBILITY          = <method_function_ref|constant> ]
  [ HOLE MOBILITY              = <method_function_ref|constant> ]
  [ ELECTRON DIFFUSION COEFFICIENT = <method_function_ref|constant> ]
  [ HOLE DIFFUSION COEFFICIENT = <method_function_ref|constant> ]
  [ ELECTRON LIFETIME          = <method_function_ref|constant> ]
  [ HOLE LIFETIME              = <method_function_ref|constant> ]
  [ ELECTRON RELAXATION        = <method_function_ref|constant> ]
  [ HOLE RELAXATION            = <method_function_ref|constant> ]
  DOPING                      = <method_function_ref|constant>
  [ TRAPS                     = <method_function_ref|constant> ]
  [ DEFECTS                   = <method_function_ref> ]
  [ INITIAL SPECIES DENSITY    = <method_function_ref> ]
  [ TRANSIENT SOURCE          = <method_function_ref> ]
  [ INTRINSIC CONCENTRATION    = <method_function_ref|constant> ]
  [ BANDGAP                   = <method_function_ref|constant> ]
  [ MOLE FRACTION              = <method_function_ref|constant> ]
  [ PRIORITY                   = <integer> ]
END
```

where:

- ▶ `model_number` is an integer identifying a model, used as a reference in the MODEL specification,
- ▶ `method_function_ref` references a particular method function (e.g., method function 3) and the number references the method id. The method function definition is dependent on the parameter. See section 3.5 for DOPING specification and section 3.5 for INITIAL SPECIES DENSITY specification.

- ▶ `constant` is a real physical value with units specified below
- ▶ MATERIAL NAME - [character string]
- ▶ RELATIVE PERMITTIVITY - [unitless]
- ▶ ELECTRON MOBILITY - [$cm^2/(Vs)$]
- ▶ HOLE MOBILITY - [$cm^2/(Vs)$]
- ▶ ELECTRON DIFFUSION COEFFICIENT - [cm^2/s]
- ▶ HOLE DIFFUSION COEFFICIENT - [cm^2/s]
- ▶ ELECTRON LIFETIME - [s]
- ▶ HOLE LIFETIME - [s]
- ▶ ELECTRON RELAXATION - [s]
- ▶ HOLE RELAXATION - [s]
- ▶ DOPING - if < 0 then donor set to 0.0 and acceptor set to $-1.0 * \text{value}$: if > 0 then donor and acceptor set to value [$\#/cm^3$]
- ▶ TRAPS - will set the value of the recombination rate (will set the poisson contribution to 0.0) [$\#/cm^3/s$]
- ▶ INTRINSIC CONCENTRATION - will set the value of n_i [$\#/cm^3$] (see section 3.5)
- ▶ BANDGAP - will set the value of the bandgap [eV]
- ▶ MOLE FRACTION - must be a value between 0.0 and 1.0 [unitless] (see section 3.5)
- ▶ PRIORITY - This helps resolve conflicts when, for example, a node shared by two material models because it falls on an interface must use the model for some calculation. The most obvious calculation is for built-in potential. The model with the higher priority takes precedence. (Default 1)

METAL Description

The metal command allows the user to specify a metal on a contact to either a semiconductor (for a Schottky contact) or an insulator (for running a MOSFET, e.g.).

```
CONTACT <n>
  SIDESET ID = <m>
  TYPE = <metal semiconductor|metal oxide>
  METAL = <metal_name> | WORKFUNCTION = <real value>
END
```

where:

► metal_name:

- ▷ AL - Aluminum
- ▷ PPOLY - P-type polysilicon
- ▷ NPOLY - N-type polysilicon
- ▷ MO - Molybdenum
- ▷ W - Tungsten
- ▷ MODI - Molybdenum disilicide
- ▷ WDI - Tungsten disilicide
- ▷ CU - Copper
- ▷ PT - Platinum
- ▷ AU - Gold
- ▷ NEUTRAL - Neutral

MPARAM Description

The MPARAM statement allows the user to change the default semiconductor material dependent properties. Note that for those parameters that may be set within the MODEL command described above this command will override the default values for the given material associated with the MODEL. MPARAM may be set with the following input line (independent of case):

```
MPARAM, <material_name>, <parameter_name> = <real value>
```

where:

► **real value:** is a real physical value with the appropriate units specified above

► **material_name:**

- ▷ SI - Silicon
- ▷ GAAS - Gallium Arsenide
- ▷ POLYSILI - Polysilicon
- ▷ S_OXIDE - Silicon Oxide
- ▷ GERMANIU - Germanium
- ▷ NITRIDE - Silicon nitride
- ▷ SAPPHIRE - Sapphire
- ▷ ALGAAS - Aluminum Gallium Arsenide
- ▷ INGAAS - Indium Gallium Arsenide
- ▷ ALINAS - Aluminum Indium Arsenide
- ▷ GAASP - Gallium Arsenide Phosphide
- ▷ INGAP - Indium Gallium Phosphide
- ▷ INASP - Indium Arsenide Phosphide

► **parameter_name:**

- ▷ PERMITTI - Relative Permittivity [unitless]
- ▷ TAUN0 - Electron Lifetime [s]
- ▷ TAUP0 - Hole Lifetime [s]
- ▷ NSRHN - Electron SRH concentration parameter [$\#/cm^3$]
- ▷ NSRHP - Hole SRH concentration parameter [$\#/cm^3$]
- ▷ EG300 - Reference value of the bandgap at 300K [eV]
- ▷ LAT_TEMP - Lattice Temperature [K]
- ▷ ELE_TAUW - Electron Energy Relaxation Time [s]
- ▷ HOL_TAUW - Hole Energy Relaxation Time [s]
- ▷ MUN0 - Electron Mobility [$cm^2/(Vs)$]
- ▷ MUP0 - Hole Mobility [$cm^2/(Vs)$]
- ▷ DIFFN0 - Electron Diffusion Coefficient [cm^2/s]
- ▷ DIFFP0 - Hole Diffusion Coefficient [cm^2/s]
- ▷ AUGN - Electron Parameter for Auger Source Term [cm^6/s]

- ▷ AUGP - Hole Parameter for Auger Source Term [cm^6/s]
- ▷ NI - Intrinsic Concentration of Material [$\#/cm^3$]
- ▷ AFFINITY - Electron affinity [V]
- ▷ EL_EMAS - Electron effective mass [kg]
- ▷ HO_EMAS - Hole effective mass [kg]
- ▷ EQUIV_MIN - Number of equivalent minima in conduction band [unitless]
- ▷ ATOMIC_DENSITY - the atomic density [$atoms/cm^3$]
- ▷ EFF_DENS_COND - effective density of states in conduction band [$\#/cm^3$]
- ▷ EFF_DENS_VALE - effective density of states in valence band [$\#/cm^3$]
- ▷ ELEC_THERM_VEL - electron thermal velocity [cm/s]
- ▷ HOLE_THERM_VEL - hole thermal velocity [cm/s]
- ▷ VO_BGN - voltage parameter used in Slotboom's bandgap narrowing model (V) (Default for Si: 9.0e-3)
- ▷ NO_BGN - concentration parameter used in Slotboom's bandgap narrowing model ($\#/cm^3$) (Default for Si: 1.0e17)
- ▷ CON_BGN - constant parameter used in Slotboom's bandgap narrowing model (Default for Si: 0.5)
- ▷ GB_N - band degeneracy factor for n-type [unitless] (Default for Si: 2.0)
- ▷ GB_P - band degeneracy factor for p-type [unitless] (Default for Si: 4.0)
- ▷ EBO_N - constant term used in calculation of band ionization energy for n-type [eV] (Default for Si: 0.045)
- ▷ EBO_P - constant term used in calculation of band ionization energy for p-type [eV] (Default for Si: 0.045)
- ▷ ALPHA_N - prefactor for the doping dependent term used in the calculation of the band ionization energy for n-type [eV/cm] (Default for Si: 3.1e-08)
- ▷ ALPHA_P - prefactor for the doping dependent term used in the calculation of the band ionization energy for p-type [eV/cm] (Default for Si: 3.037e-08)
- ▷ BETA_N - prefactor used by temperature dependent term in the calculation of the band ionization energy for n-type [unitless] (Default for Si: 200.0)
- ▷ BETA_P - prefactor used by temperature dependent term in the calculation of the band ionization energy for p-type [unitless] (Default for Si: 200.0)
- ▷ GAMMA_N - The exponent of temperature used in the calculation of the band ionization energy for n-type [unitless] (Default for Si: 1.0)

- ▷ GAMMA_P - The exponent of temperature used in the calculation of the band ionization energy for p-type [unitless] (Default for Si: 0.950)
- ▷ EG_ALPH - The value of alpha used in calculating the energy bandgap as a function of temperature $\left[\frac{eV}{K}\right]$
- ▷ EG_BETA - The value of beta used in calculating the energy bandgap as a function of temperature $\left[\frac{eV}{K}\right]$
- ▷ NC300 - The effective density of states in the conduction band of the semiconductor at 300 K $\left[\frac{\#}{cm^3}\right]$
- ▷ NC_F - The exponent of temperature for describing the effective density of states in the conduction band
- ▷ NV300 - The effective density of states in the valence band of the semiconductor at 300 K $\left[\frac{\#}{cm^3}\right]$
- ▷ NV_F - The exponent of temperature for describing the effective density of states in the valence band
- ▷ HDT_MIN - The value of impurity concentration at which the linear interpolation starts for the incomplete ionization model $\left[\#/cm^3\right]$
- ▷ HDT_MAX - The value of impurity concentration above which complete ionization is assumed $\left[\#/cm^3\right]$

Arora mobility model parameters

- ▷ MUN1_ARO - The minimum electron mobility $\left[\frac{cm^2}{Vs}\right]$
- ▷ MUN2_ARO - The maximum electron mobility $\left[\frac{cm^2}{Vs}\right]$
- ▷ MUP1_ARO - The minimum hole mobility $\left[\frac{cm^2}{Vs}\right]$
- ▷ MUP2_ARO - The maximum hole mobility $\left[\frac{cm^2}{Vs}\right]$
- ▷ CN_ARORA - The reference impurity concentration $\left[\frac{\#}{cm^3}\right]$
- ▷ AN_ARORA - Parameters used in the exponent of normalized impurity concentration
- ▷ EXN1_ARO - Exponent of normalized temperature
- ▷ EXN2_ARO - Exponent of normalized temperature
- ▷ EXN3_ARO - Exponent of normalized temperature
- ▷ EXN4_ARO - Exponent of normalized temperature

- ▷ EXP1_AR0 - Exponent of normalized temperature
- ▷ EXP2_AR0 - Exponent of normalized temperature
- ▷ EXP3_AR0 - Exponent of normalized temperature
- ▷ EXP4_AR0 - Exponent of normalized temperature

Analytic mobility model parameters

- ▷ MUN_MIN - The minimum electron mobility in the analytic mobility model $\left[\frac{cm^2}{V-s} \right]$.
- ▷ MUN_MAX - The maximum electron mobility in the analytic mobility model $\left[\frac{cm^2}{V-s} \right]$.
- ▷ NREFN - The reference impurity concentration used in the analytic mobility model for electrons $\left[\frac{\#}{cm^3} \right]$.
- ▷ NUN - The exponent of normalized temperature used in the numerator of the analytic model for electrons.
- ▷ XIN - The exponent of normalized temperature used in the denominator of the analytic mobility model for electrons.
- ▷ ALPHAN - The exponent of the ratio of the total impurity concentration to NREFN used in the analytic mobility model for electrons.
- ▷ MUP_MIN - The minimum hole mobility in the analytic mobility model $\left[\frac{cm^2}{V-s} \right]$.
- ▷ MUP_MAX - The maximum hole mobility in the analytic mobility model $\left[\frac{cm^2}{V-s} \right]$.
- ▷ NREFP - The reference impurity concentration used in the analytic mobility model for holes $\left[\frac{\#}{cm^3} \right]$.
- ▷ NUP - The exponent of normalized temperature used in the numerator of the analytic mobility model for holes.
- ▷ XIP - The exponent of normalized temperature used in the denominator of the analytic mobility model for holes.
- ▷ ALPHAP - The exponent of the ratio of the total impurity concentration of NREFP used in the analytic mobility model for holes.

III-V Compound Semiconductor Analytic Mobility Model Parameters

- ▷ MIN_X1 - The linear term appearing in the expression of $\mu_n^{Min}(X)$ used in the analytic mobility model.

- ▷ MIN_X2 - The quadratic term appearing in the expression of $\mu_n^{Min}(X)$ used in the analytic mobility model.
- ▷ MAN_X1 - The linear term appearing in the expression of $\mu_n^{Max}(X)$ used in the analytic mobility model.
- ▷ MAN_X2 - The quadratic term appearing in the expression of $\mu_n^{Max}(X)$ used in the analytic mobility model.
- ▷ NREFN2 - The reference impurity concentration used in the quadratic term of the analytic mobility model for electrons $\left[\frac{\#}{cm^3}\right]$.
- ▷ MIP_X1 - The linear term appearing in the expression of $\mu_p^{Min}(X)$ used in the analytic mobility model.
- ▷ MIP_X2 - The quadratic term appearing in the expression of $\mu_p^{Min}(X)$ used in the analytic mobility model.
- ▷ MAP_X1 - The linear term appearing in the expression of $\mu_p^{Max}(X)$ used in the analytic mobility model.
- ▷ MAP_X2 - The quadratic term appearing in the expression of $\mu_p^{Max}(X)$ used in the analytic mobility model.
- ▷ NREFP2 - The reference impurity concentration used in the quadratic term of the analytic mobility model for holes $\left[\frac{\#}{cm^3}\right]$.

Carrier-Carrier Scattering Mobility Model Parameters

- ▷ A_CCS - Parameter used in the carrier-carrier scattering term $\left[\frac{1}{cm-V-s}\right]$.
- ▷ B_CCS - Parameter used in the carrier-carrier scattering term $\left[\frac{1}{cm^2}\right]$.
- ▷ A_LIC - Parameter used in the carrier-carrier scattering mobility model [unitless]
- ▷ B_LIC - Parameter used in the carrier-carrier scattering mobility model [unitless]
- ▷ C_LIC - Parameter used in the carrier-carrier scattering mobility model [unitless]
- ▷ EX_LIC - Exponent used in the carrier-carrier scattering mobility model [unitless]
- ▷ MUNO_LAT - The room temperature mobility for electrons used in the lattice-scattering term of the model $\left[\frac{cm^2}{V-s}\right]$
- ▷ EXN_LAT - The exponent used in the lattice scattering term for electrons in the carrier-carrier scattering mobility model [unitless]
- ▷ AN_IIS - Parameter used in the ionized impurity scattering term for electrons of the carrier-carrier scattering mobility model $\left[\frac{1}{cm-V-s}\right]$.

- ▷ BN_IIS - Parameter used in the ionized impurity scattering term for electrons of the carrier-carrier scattering mobility model $\left[\frac{1}{cm^3}\right]$.
- ▷ MUPO_LAT - The room temperature mobility for holes used in the lattice scattering term of the model $\left[\frac{cm^2}{V-s}\right]$.
- ▷ EXP_LAT - The exponent used in the lattice scattering term for holes in the mobility model [unitless]
- ▷ AP_IIS - Parameter used in the ionized impurity scattering term for holes of the carrier-carrier scattering mobility model $\left[\frac{1}{cm-V-s}\right]$.
- ▷ BP_IIS - Parameter used in the ionized impurity scattering term for holes of the carrier-carrier scattering model $\left[\frac{1}{cm^3}\right]$.

Inversion and Accumulation Layer Mobility Model Parameters

- ▷ BN_IAL - Parameter used in the two dimensional acoustic phonon scattering part of the model for electrons $\left[\frac{cm}{s}\right]$
- ▷ BP_IAL - Parameter used in the two dimensional acoustic phonon scattering part of the model for holes $\left[\frac{cm}{s}\right]$
- ▷ CN_IAL - Parameter used in the two dimensional acoustic phonon scattering part of the model for electrons $\left[\frac{K}{sV^{-2/3}}cm^{3/2}cm^{3exp4.ial}\right]$
- ▷ CP_IAL - Parameter used in the two dimensional acoustic phonon scattering part of the model for holes $\left[\frac{K}{sV^{-2/3}}cm^{3/2}cm^{3exp4.ial}\right]$
- ▷ DN_IAL - Parameter used in the surface roughness term for electrons $\left[\frac{V}{s}cm^{3exp4.ial}\right]$
- ▷ DP_IAL - Parameter used in the surface roughness term for holes $\left[\frac{V}{s}cm^{3exp4.ial}\right]$
- ▷ D1N_IAL - Parameter used in the carrier concentration independent term of the two dimensional Coulomb impurity scattering for electrons $\left[\frac{cm^x}{V-s}\right]$
- ▷ D1P_IAL - Parameter used in the carrier concentration independent term of the two dimensional Coulomb impurity scattering for holes $\left[\frac{cm^x}{V-s}\right]$
- ▷ D2N_IAL - Parameter used in the carrier dependent term of the two dimensional Coulomb impurity scattering term for electrons $\left[\frac{cm^x}{V-s}\right]$
- ▷ D2P_IAL - Parameter used in the carrier dependent term of the two dimensional Coulomb impurity scattering for holes $\left[\frac{cm^x}{V-s}\right]$
- ▷ MASSN_IAL - Electron mass entering the screening parameter P $[m_0]$

- ▷ MASSP_IAL - Hole mass entering the screening parameter P [m_0]
- ▷ EXN4_IAL - Exponent of total impurity concentration used in the acoustic phonon scattering and surface roughness for electrons [unitless]
- ▷ EXP4_IAL - Exponent of total impurity concentration used in the acoustic phonon scattering and surface roughness for holes [unitless]
- ▷ EXN5_IAL - Exponent of electron density in the two dimensional Coulomb impurity scattering term for electrons [unitless]
- ▷ EXP5_IAL - Exponent of hole density in the two dimensional Coulomb impurity scattering term for holes [unitless]
- ▷ EXN6_IAL - Exponent of donor/acceptor concentration in the carrier dependent two dimensional Coulomb impurity scattering term for electrons [unitless]
- ▷ EXP6_IAL - Exponent of donor/acceptor concentration in the carrier dependent two dimensional Coulomb impurity scattering term for holes [unitless]
- ▷ EXN7_IAL - Exponent of donor/acceptor concentration in the carrier independent two dimensional Coulomb impurity scattering term for electrons [unitless]
- ▷ EXP7_IAL - Exponent used in the donor/acceptor concentration in the carrier independent two dimensional Coulomb impurity scattering term for holes [unitless]

Lombardi surface mobility model parameters

- ▷ MUN0_LSM - Mobility parameter used in the bulk term of the Lombardi surface mobility model for electrons $\left[\frac{cm^2}{V-s} \right]$.
- ▷ MUN1_LSM - Mobility parameter used in the bulk term of the Lombardi surface mobility model for electrons $\left[\frac{cm^2}{V-s} \right]$.
- ▷ MUN2_LSM - Mobility parameter used in the bulk term of the Lombardi surface mobility model for electrons $\left[\frac{cm^2}{V-s} \right]$.
- ▷ CRN_LSM - Mobility parameter used in the bulk term of the Lombardi surface mobility model for electrons $\left[\frac{\#}{cm^3} \right]$.
- ▷ CSN_LSM - Mobility parameter used in the bulk term of the Lombardi surface mobility model for electrons $\left[\frac{\#}{cm^3} \right]$.
- ▷ BN_LSM - Mobility parameter used in the acoustic term of the Lombardi surface mobility model for electrons $\left[\frac{cm}{s} \right]$.

- ▷ CN_LSM - Mobility parameter used in the acoustic term of the Lombardi surface mobility model for electrons $[(K - cm/s)(V/cm)^{-2/3}]$.
- ▷ DN_LSM - Mobility parameter used in the acoustic term of the Lombardi surface mobility model for electrons $[(cm^2/V - s)(V/cm)^{exp8.lsm}]$.
- ▷ EXN1_LSM - Exponent used in the bulk term of the Lombardi surface mobility model for electrons [unitless].
- ▷ EXN2_LSM - Exponent used in the bulk term of the Lombardi surface mobility model for electrons [unitless].
- ▷ EXN3_LSM - Exponent used in the bulk term of the Lombardi surface mobility model for electrons [unitless].
- ▷ EXN4_LSM - Exponent used in the acoustic term of the Lombardi surface mobility model for electrons [unitless].
- ▷ EXN8_LSM - Exponent used in the surface roughness term of the Lombardi surface mobility model for electrons [unitless].
- ▷ MUPO_LSM - Mobility parameter used in the bulk term of the Lombardi surface mobility model for holes $[\frac{cm^2}{V-s}]$.
- ▷ MUP1_LSM - Mobility parameter used in the bulk term of the Lombardi surface mobility model for holes $[\frac{cm^2}{V-s}]$.
- ▷ MUP2_LSM - Mobility parameter used in the bulk term of the Lombardi surface mobility model for holes $[\frac{cm^2}{V-s}]$.
- ▷ CRP_LSM - Mobility parameter used in the bulk term of the Lombardi surface mobility model for holes $[\frac{\#}{cm^3}]$.
- ▷ CSP_LSM - Mobility parameter used in the bulk term of the Lombardi surface mobility model for holes $[\frac{\#}{cm^3}]$.
- ▷ BP_LSM - Mobility parameter used in the acoustic term of the Lombardi surface mobility model for holes $[cm/s]$.
- ▷ CP_LSM - Mobility parameter used in the acoustic term of the Lombardi surface mobility model for holes $[(K - cm/s)(V/cm)^{-2/3}]$.
- ▷ DP_LSM - Mobility parameter used in the acoustic term of the Lombardi surface mobility model for holes $[(cm^2/V - s)(V/cm)^{exp8.lsm}]$.
- ▷ EXP1_LSM - Exponent used in the bulk term of the Lombardi surface mobility model for holes [unitless].
- ▷ EXP2_LSM - Exponent used in the bulk term of the Lombardi surface mobility model for holes [unitless].

- ▷ EXP3_LSM - Exponent used in the bulk term of the Lombardi surface mobility model for holes [unitless].
- ▷ EXP4_LSM - Exponent used in the acoustic term of the Lombardi surface mobility model for holes [unitless].
- ▷ EXP8_LSM - Exponent used in the surface roughness term of the Lombardi surface mobility model for holes [unitless].
- ▷ PC_LSM - Mobility parameter used in the bulk term of the Lombardi surface mobility model for holes $\left[\frac{\#}{\text{cm}^3} \right]$.

Philips mobility model parameters

- ▷ MMNN_UM - Mobility parameter used in the Philips Unified mobility model to determine electron mobility at high dopant and/or carrier levels $\left[\frac{\text{cm}^2}{V_s} \right]$.
- ▷ MMXN_UM - Maximum electron mobility in the Philips Unified mobility model $\left[\frac{\text{cm}^2}{V_s} \right]$.
- ▷ NRFN_UM - Reference impurity concentration for electron mobility, used with the Philips Unified mobility model $\left[\frac{\#}{\text{cm}^3} \right]$.
- ▷ ALPN_UM - Exponent used for electron mobility in the Philips Unified mobility model.
- ▷ TETN_UM - Exponent used for temperature dependence of lattice scattering for electrons in the Philips Unified mobility model.
- ▷ NRFD_UM - Reference impurity concentration for donors to model ultra-high doping effects in the Philips Unified mobility model $\left[\frac{\#}{\text{cm}^3} \right]$.
- ▷ CRFD_UM - Factor determining the ultra-high doping effects for donors in the Philips Unified mobility model.
- ▷ MMNP_UM - Mobility parameter used in the Philips Unified mobility model to determine hole mobility at high dopant and/or carrier levels $\left[\frac{\text{cm}^2}{V_s} \right]$.
- ▷ MMXP_UM - Maximum hole mobility in the Philips Unified mobility model $\left[\frac{\text{cm}^2}{V_s} \right]$.
- ▷ NRFP_UM - Reference impurity concentration for hole mobility, used with the Philips Unified mobility model $\left[\frac{\#}{\text{cm}^3} \right]$.
- ▷ ALPP_UM - Exponent used for hole mobility in the Philips Unified mobility model.
- ▷ TETP_UM - Exponent used for temperature dependence of lattice scattering for holes in the Philips Unified mobility model.

- ▷ NRFA_UM - Reference impurity concentration for acceptors to model ultra-high doping effects in the Philips Unified mobility model $\left[\frac{\#}{\text{cm}^3}\right]$.
- ▷ CRFA_UM - Factor determining the ultra-high doping effects for acceptors in the Philips Unified mobility model.

Lucent mobility model parameters

- ▷ AN_LUC - The constant part of the exponent used in the surface roughness term model for electrons.
- ▷ AP_LUC - The constant part of the exponent used in the surface roughness term for holes.
- ▷ BN_LUC - Mobility parameter used in the acoustic term for electrons [cm/s].
- ▷ BP_LUC - Mobility parameter used in the acoustic term for holes [cm/s].
- ▷ CN_LUC - Mobility parameter used in the acoustic term for electrons $\left[\left(\frac{\text{cm}^2}{V-s}\right) (V/\text{cm})^{1/3} \text{cm}^{3EXN4.LUC}\right]$.
- ▷ CP_LUC - Mobility parameter used in the acoustic term for holes $\left[\left(\frac{\text{cm}^2}{V-s}\right) (V/\text{cm})^{1/3} \text{cm}^{3EXP4.LUC}\right]$.
- ▷ DN_LUC - Mobility parameter used in the acoustic term for electrons $[(\text{cm}^2/V-s) (V/\text{cm})^{\gamma_n}]$.
- ▷ DP_LUC - Mobility parameter used in the acoustic term for holes $[(\text{cm}^2/V-s) (V/\text{cm})^{\gamma_p}]$.
- ▷ FN_LUC - The factor multiplying total carrier concentration in the exponent used in the surface roughness term for electrons $[\text{cm}^{3(1-EXN9.LUC)}]$.
- ▷ FP_LUC - The factor multiplying total carrier concentration in the exponent used in the surface roughness term for holes $[\text{cm}^{3(1-EXP9.LUC)}]$.
- ▷ KN_LUC - The exponent of temperature used in the acoustic term for electrons.
- ▷ KP_LUC - The exponent of temperature used in the acoustic term for holes.
- ▷ EXN4_LUC - The exponent of total impurity concentration used in the acoustic term for electrons.
- ▷ EXP4_LUC - The exponent of total impurity concentration used in the acoustic term for holes.
- ▷ EXN9_LUC - The exponent of total impurity concentration used in the surface roughness term for electrons.
- ▷ EXP9_LUC - The exponent of total impurity concentration used in the surface roughness term model for holes.

Impact ionization model parameters

- ▷ N_IONIZATION - parameter for calculating $\alpha_{n,ii}^{\infty} \left[\frac{1}{cm} \right]$
- ▷ N_ION_1 - parameter for calculating $\alpha_{n,ii}^{\infty} \left[\frac{1}{cm-K} \right]$
- ▷ N_ION_2 - parameter for calculating $\alpha_{n,ii}^{\infty} \left[\frac{1}{cm-K^2} \right]$
- ▷ P_IONIZATION - parameter for calculating $\alpha_{p,ii}^{\infty} \left[\frac{1}{cm} \right]$
- ▷ P_ION_1 - parameter for calculating $\alpha_{p,ii}^{\infty} \left[\frac{1}{cm-K} \right]$
- ▷ P_ION_2 - parameter for calculating $\alpha_{p,ii}^{\infty} \left[\frac{1}{cm-K^2} \right]$
- ▷ EXN_II - exponent for ratio of critical electric field to parallel electric field for electrons [unitless]
- ▷ EXP_II - exponent for ratio of critical electric field to parallel electric field for holes [unitless]
- ▷ OP_PH_EN - value of optical phonon energy [eV]
- ▷ LAN300 - optical phonon mean free path for electrons at 300 K [cm]
- ▷ LAP300 - optical phonon mean free path for holes at 300 K [cm]

3.3 Physics-Specific Reference for **Charon** Semiconductor Physics

PHYSICS Description

The charon-semiconductor-specific cards are basically any option defined within the tables below.

```
physics = <n>, <physics name>, <formulation name>
.
<charon-semiconductor-specific cards>
.
END
```

where:

- ▶ *n* - An integer ID to associate with each physics. If you're only running a single physics this is 1.
- ▶ *physics name* - For semiconductor physics this should be SEMICONDUCTOR.
- ▶ *formulation name* - For semiconductor physics this is one of:
 - ▷ NONLINEARPOISSON — NLP
 - ▷ DRIFTDIFFUSION — DD
 - ▷ SGFVM — SG
 - ▷ NONLINEARENERGY — NLE
 - ▷ ENERGYTRANSPORT — ET
 - ▷ SGENERGY — SGET
 - ▷ POISSONLATTICE — NLT (nonlinear poisson for dd plus heat equation)
 - ▷ TLATTICE — LATTICEDD (solves dd plus heat equation)
 - ▷ SGLATTICE — LATTICESG (sgfvm dd plus heat equation)

All subsequent documentation is specific to the physics subsection of the input file and should be contained within that subsection, that is between *physics=semiconductor* and it's corresponding *END* as shown above.

SPECIES SOURCE Description

Source Terms:

The species source command allows the user to enable/disable specific semiconductor source terms (when source terms are turned on). Currently SRH and DEFECT are the only terms turned on by default. To set source specific variables use MPARAM to set for a given material (see section 3.2).

```
species source = <source_name>  
  <off|on>  
end
```

where:

- ▶ **source_name:**
 - ▷ **SRH** - Shockley-Read-Hall recombination term which by default uses the formulation $U_{SRH} = \frac{pn - n_{ie}^2}{\tau_p(n + n_{ie}) + \tau_n(p + n_{ie})}$. If traps are specified see the Traps specification within section 3.5 for SRH formulation.
 - ▷ **AUGER** - Auger recombination which uses the formulation $U_{Auger} = AUGN (pn^2 - nn_{ie}^2) + AUGP (np^2 - pn_{ie}^2)$, where AUGN and AUGP are defined within the MPARAM specification (see section 3.2).
 - ▷ **IMPACT_IONIZATION** - Impact ionization which uses the model found on pages 2-103 to 2-104 of the DaVinci (DV 2002.4) manual. Specific parameters for this model are defined within the MPARAM specification (see section 3.2).
- ▶ **off** - disables the specified source term
- ▶ **on** - enables the specified source term

SCALING PARAMETERS Description

Scaling Parameter:

Allows you to override the scaling parameters used in **Charon** semiconductor analysis.

```
SCALING PARAMETER, "<parameter name>" = <real value>
```

Limitations:

- ▶ Currently only *"C0"* is implemented as an overridable scaling parameter.
- ▶ Be sure to use quotes around the name of the scaling parameter as they are case sensitive. *T0* is NOT the same as *t0*

REGION Description

Defect Species Terms:

To set specific defect species values within a given material see the defects section 3.5. Currently the defect species terms are defined via a Chemkin input file within the region specification as shown below.

```

region = <region_name>
[material = <material_name>]
species information = chemkin file, <chemkin_file_name>
end

```

where:

- ▶ **region_name** : currently any name you please (may be specific regions defined in the future)
- ▶ **material_name** : a specified semiconductor material within the region where valid material name values are in section 3.2, and the default is Si.
- ▶ **chemkin_file_name**: name of the chemkin file containing the list of species (including electron and hole species)

MESH REFINEMENT Description

Initial Mesh Refinement:

This allows the user to specify that adaptive mesh refine should be performed based on the doping. You must activate *Nevada's* mesh refinement for this to be activated.

```

CHARON MESH REFINEMENT
  INITIAL REFINEMENT
    PARAM, COMP=<LOG|SLOG>
    PARAM, THRESH=<real value>
    PARAM, LAYERS=<integer number of layers>
  END

  SIDESSET REFINEMENT
    ID = <integer sideset ID 1>
    ID = <integer sideset ID 2>
    .
    .
    .
    ID = <integer sideset ID N>

```

```

END
END

```

where:

- ▶ **COMP** - The function used to compute the refinement threshold. Supported functions are:
 - ▷ **LOG** - $\text{MAX}|\log(|C_i|) - \log(|C_j|)|$
 - ▷ **SLOG** - $\text{MAX}|\text{sign}(C_i) \cdot \log(1 + |C_i|) - \text{sign}(C_j) \cdot \log(1 + |C_j|)|$
 - ▷ where $i = 1 \dots \text{number of element nodes}$ and $j = i \dots \text{number of element nodes}$
- ▶ **THRESH** - any element whose value computed by **COMP** exceeds this value will be marked as needing refinement.
- ▶ **LAYERS** - this number of layers around any elements that meet the functional refinement criteria will also be refined.

In addition to the CHARON MESH REFINEMENT section you will also need the following *Nevada* adaptivity specification. For readability it is suggested that this section immediately precede the CHARON MESH REFINEMENT section.

```

DOMAIN
  MAXIMUM LEVELS 2
END

ADAPTIVITY SPECIFICATION
  ENABLE ADAPTIVITY
  UNREFINEMENT CONTROL
  LOCK INITIAL REFINEMENT
END
END

```

where:

- ▷ **MAXIMUM LEVELS** - Determines how many refinements of an element marked as needing refinement will be performed. For example, with 2 levels a single quad element will be subdivided into 16 new elements. Note that **Charon** restricts refinement so that at most there is a difference of 1 level of refinement between adjacent elements. For the aforementioned example of 2 levels of refinement this means that elements adjacent to elements needing refinement

will also be refined with 1 level of refinement. In effect this adds 1 extra layer of refinement for every level of refinement above 1 that you specify.

- Limitations: **SIDASET REFINEMENT** is currently only valid if you are also doing **INITIAL REFINEMENT**. The number of layers used when doing sideset refinement is taken from the **LAYERS** parameter in the **INITIAL REFINEMENT** section.

CURRENT CONVERGENCE Description

Current Convergence Test

This allows you to add a convergence test to the nonlinear solver based on the scalar electrical current at the contacts and control its parameters.

```
USE CURRENT CONVERGENCE IN NOX = <TRUE|FALSE>
CURRENT WRMS RELATIVE TOLERANCE = <real value>
CURRENT WRMS ABSOLUTE TOLERANCE = <real value>
```

where:

- the formula used to determine convergence is:

- ▷
$$||I|| = \sum_i \frac{|I_i^{k+1} - I_i^k|}{\epsilon_r |I_i^k| + \epsilon_a} \quad i = 1, \dots, \text{number of contacts}$$
- ▷ ϵ_r - relative tolerance (default: 1.0e-8)
- ▷ ϵ_a - absolute tolerance (default: 1.0e-18)
- ▷ $||I||$ must be less than 1.0 in order to indicate convergence.

STABILIZATION Control

PDE Residual Terms used for stabilization schemes

Specify optional terms to be included in the calculation of the PDE residual for stabilization schemes.


```

SUPG RESIDUAL TERMS
  L2 [= <TRUE|FALSE>]
  POISSON [= <TRUE|FALSE>]
END

```

```

DC RESIDUAL TERMS
  L2 [= <TRUE|FALSE>]
  POISSON [= <TRUE|FALSE>]
END

```

Note that just listing the term, without an "= <TRUE|FALSE>", implies that the corresponding term is on within the residual calculation.

Calculation of Tau in SUPG

This allows you to specify the name of the algorithm to use for the calculation of tau in SUPG stabilization.

```
SUPG TAU CALCULATION = <LINEAR|TANH>
```

if not specified the default is LINEAR.

Calculation of the length scale used in SUPG and DC

This allows you to specify which length-scale to use in SUPG and DC.

```
SUPG LENGTH SCALE CALCULATION = <STREAM|SHAKIB>
```

if not specified the default is STREAM.

► SHAKIB - $h = 2\sqrt{\frac{\mathcal{D}v \cdot g_c \cdot v^T}{|g_c|^2 |v|^2}}$

► STREAM - $h = 2\sqrt{\frac{v \cdot g^c \cdot v^T}{|v|^2}}$

where:

- ▶ $\mathcal{D}$ - The physical dimension
- ▶ g^c - Element covariant tensor
- ▶ g_c - Element contravariant tensor
- ▶ v - Velocity vector

Calculation of Tau for Poisson equation

This allows you to specify the name of the algorithm to use for the calculation of tau as used for stabilization in the Poisson equation.

```
POISSON TAU CALCULATION = <L1|L2|FRANCA1|LP|HARARI1|SHADID1>
```

if not specified the default is L2. The variations are

- ▶ L1: $\tau = \left[\frac{3\lambda^2 \epsilon_r}{\|g_c\|} + |S_\psi| \right]^{-1}$
- ▶ L2: $\tau = \left[\frac{3(\lambda^2 \epsilon_r)^2}{\|g_c\|^2} + (S_\psi)^2 \right]^{-\frac{1}{2}}$

BANDGAP NARROWING

Turn on the bandgap narrowing (BGN) model for heavy doping

```
BANDGAP NARROWING [= <TRUE|ON|FALSE|OFF|CONSISTENT>]
```

Notes:

- ▶ If you turn this on you also need to specify a functional variation for the *INTRINSIC CONCENTRATION* see section 3.5 for an example.
- ▶ CONSISTENT should NOT be used at present.

CALCULATE VELOCITIES

Calculate velocities, and thus Peclet numbers, even for Galerkin runs.

```
CALCULATE VELOCITIES = <TRUE|FALSE>
```

Notes:

- ▶ This causes the calculation of velocities, even for Galerkin runs which don't usually require specific calculation of velocities.
- ▶ You'll need to add the variables to the *VARIABLE MANAGER DATA OUTPUT* so that the velocities will be calculated by the variable manager. Here's an example:

```
VARIABLE MANAGER DATA OUTPUT
  "charon:variable:ELECTRON_DENSITY:peclet", q, n_peclet
  "charon:variable:HOLE_DENSITY:peclet", q, p_peclet
  "charon:stab:variable:electron:velocity:x", q, n_v_x
  "charon:stab:variable:electron:velocity:y", q, n_v_y
  "charon:stab:variable:hole:velocity:x", q, p_v_x
  "charon:stab:variable:hole:velocity:y", q, p_v_y
END
```

SURFACE CHARGE

Set a surface charge on the specified sideset

```
BC, NEUMANN, ELECTRIC_POTENTIAL, SIDESSET <n>, ELEMENT BLOCK <m>, \
METHOD FUNCTION <p>
```

Notes:

- ▶ *n* The integer sideset ID.
- ▶ *m* The integer element block ID. Usually this is 1 for semiconductor problems.
- ▶ *p* The method function ID.

The METHOD FUNCTION specification should look like:

```
METHOD FUNCTION <p>
  KEY = SURFACE_CHARGE
  TYPE = [CONSTANT|LINEAR TIME|LOG TIME]
  VALUE = [qf]
  <INITIAL VALUE = [r]>
  <START TIME = [r]>
  <STOP TIME = [r]>
END
```

where:

► Currently supported types are:

- ▷ *CONSTANT* - Constant value
- ▷ *LINEAR TIME* - A linear increase in charge until *STOP TIME* and constant *VALUE* thereafter. The formula used is:
 - * If $t < \text{START TIME}$
 - charge = INITIAL VALUE
 - * If $\text{START TIME} < t \leq \text{STOP TIME}$
 - charge = $\frac{\text{VALUE} - \text{INITIAL VALUE}}{\text{STOP TIME} - \text{START TIME}} (t - \text{START TIME}) + \text{INITIAL VALUE}$
 - * If $t > \text{STOP TIME}$
 - charge = VALUE
- ▷ *LOG TIME* - Logarithmic variation in charge over linear time. This will allow a more steady buildup of charge for large values. This is linear in the logarithmic value of the surface charge. The formula used is:
 - * If $t < \text{START TIME}$
 - charge = INITIAL VALUE
 - * If $\text{START TIME} < t \leq \text{STOP TIME}$
 - $s = \text{sign}(\text{VALUE})$
 - $v = \frac{\log(|\text{VALUE}|) - \log(|\text{INITIAL VALUE}|)}{\text{STOP TIME} - \text{START TIME}} (t - \text{START TIME}) + \log(|\text{INITIAL VALUE}|)$
 - charge = $s \times 10^v$
 - * If $t > \text{STOP TIME}$
 - charge = VALUE

- ▷ qf is a real constant value, analogous to QF in Medici (units: $\# / cm^2$).
- ▷ r is a real constant value.

Notes:

- ▶ When using a transient variation in the surface charge, and variable time-stepping in the time integrator, you will likely want to specify a set of *BREAK POINTS* in the time integrator at the *START TIME* and *STOP TIME* to insure that the functional variation in surface charge is correctly captured.
- ▶ In the case of *LOG TIME* the *INITIAL VALUE* must be non-zero. If unspecified the default value is 10^{-10} .

3.4 Physics-Specific Input File Reference Page for **Charon** Nonlinear-Poisson Physics

OPERATOR SCALING Description

Operator scaling. Used to scale named operators by a real, scalar value. The names are physics-dependent and there is no error checking on that name. Read the documentation for the specific physics to get a list of those names. As an example you can change the sign on a term by setting it's scale factor to -1.0.

```
OPERATOR SCALING, <term name> = <real value>
```

DOPING Description

Doping Specification:

The Doping Specification is specified within the following section: 3.5.

INITIAL SPECIES DENSITY Description

Initial Defect Density Specification:

The Initial Defect Density Specification is specified within the following section: 3.5.

3.5 Physics-Specific Input File Reference Page for **Charon** Semiconductor Drift-Diffusion Physics

OPERATOR SCALING

Operator scaling. Used to scale named operators by a real, scalar value. The names are physics-dependent and there is no error checking on that name. Read the documentation for the specific physics to get a list of those names. As an example you can change the sign on a term by setting its scale factor to -1.0.

```
OPERATOR SCALING, <term name> = <real value>
```

MOBILITY Description

Mobility is a material property and must be specified as such from within the input file.

```
METHOD FUNCTION <n>  
  KEY = <H_MOBILITY|E_MOBILITY>  
  MODEL NAME = <ARORA|ANALYTIC|IIIV_ANALYTIC|CARR|IALMOB|LUCENT  
                |PHILIPS|PHILIPS_NEW|SURFACE|TEMPMOB>  
END
```

where:

► KEY

- ▷ H_MOBILITY - hole mobility
- ▷ E_MOBILITY - electron mobility

In the case of the Philips mobility, there are two versions. The version called philips uses the magnitude of the electric field parallel to the interface (x-direction). The version called philips_new uses the component of the electric field (not the magnitude) parallel to the direction of electron current flow for calculating electron mobility and the component of electric field parallel to the direction of hole current flow for calculating the hole mobility.

```
METHOD FUNCTION <n>
  KEY = <H_MOBILITY|E_MOBILITY>
  MODEL NAME = <PHILIPS|PHILIPS_NEW>
END
```

► MODEL NAME

- ▷ ARORA - Arora mobility model – reference by Arora, Hauser, and Roulston, "Electron and Hole Mobilities in Silicon as a Function of Concentration and Temperature," IEEE Transactions on Electron Devices, Vol. ED-29, pp.292-295, 1967.
- ▷ ANALYTIC - Analytic mobility model – This model is from the reference by D.M. Caughey and R.E. Thomas "Carrier Mobilities in Silicon Empirically Related to Doping and Field", Proc. IEEE, Vol 55, pp. 2192-2193, 1967.
- ▷ IIIV_ANALYTIC - ** WARNING ** LIMITED TESTING This model for Low-Field Mobility is taken from the Medici User Manual, page 2-163, equations 2-466 through 2-468. This is a modified version of the Analytic mobility model that is mole-fraction dependent. The high-field portion of this model is found in the Medici User Manual, page 2-164, equations 2-469 through 2-471.
- ▷ CARR - Carrier-carrier scattering mobility – References Dorkel and Leturq, Solid-State Electronics, vol. 24, no. 9, pp. 821-825, 1981.
- ▷ IALMOB - ** WARNING ** LIMITED TESTING Inversion and accumulation layer mobility model – References S.A. Mujtaba, "Advanced Mobility Models for Design and Simulation of Deep Submicrometer MOSFETs", Technical Report NO. ICL 90-053, Stanford University (1995).
- ▷ LUCENT - This model is from the reference by M.N. Darwish, J.L. Lentz, M.R. Pinto, P.M. Zeitzoff, T.J. Krutsick and H.H. Vuong, "An Improved Electron and Hole Mobility Model for General Purpose Device Simulation", IEEE Transactions on Electron Devices, vol. 44, no. 9, pp. 1529-1538, Sept. 1997.

- ▷ PHILIPS - A Unified Model of dopant and carrier density dependent scattering. Includes the Caughey-Thomas E-field dependent mobility as well. References: Klaassen, "A Unified Mobility Model for Device Simulation - I", IEEE Solid State Electronics, Vol. 35, pp. 953-959, 1992. Klaassen, "A Unified Mobility Model for Device Simulation - II", IEEE Solid State Electronics, Vol. 35, pp. 961-967, 1992.
- ▷ SURFACE - Lombardi Surface Mobility Model - combines mobility of the semiconductor-insulator interface with bulk mobility. Reference: Lombardi, Manzini, Saporito, and Vanzi, "A physically based mobility model for numerical simulation of nonplanar devices", IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Nov. 1988, vol. 7, no. 11, p.1164-71.
- ▷ TEMPMOB - This model for Carrier Temperature-Dependent mobility is taken from the DaVinci Manual, DV 4.1, eqns. 2-287 and 2-288 on pg. 2-96 of that version. This default model assumes that the velocity is saturated throughout the device, and the carrier energy relaxation times are also assumed to be constant.

DIFFUSION Description

Diffusion is a material property and must be specified as such from within the input file. Currently there is only a single diffusion model and no optional properties. The current diffusion model is simply equivalent to $(\text{mobility} * V_{th})$.

```
METHOD FUNCTION <n>  
  KEY = <H_DIFF_COEF|E_DIFF_COEF>  
END
```

where:

► KEY

- ▷ H_DIFF_COEF - hole diffusion coefficient
- ▷ E_DIFF_COEF - electron diffusion coefficient

TRAPS Description

Traps are specified as a material property so that the trap properties may be set for a given material and/or model. When a TRAPS model is not specified the SRH source term is specified by:

$$U_{SRH} = \frac{pn - n_{ie}^2}{\tau_p (n + n_{ie}) + \tau_n (p + n_{ie})}. \quad (3.1)$$

When a TRAPS model is specified the SRH source term is specified by;

$$U_{SRH} = \sum_i \frac{pn - n_{ie}^2}{\tau_{p_i} (n + n_{t_i}) + \tau_{n_i} (p + p_{t_i})} \quad (3.2)$$

where:

$$n_{t_i} = n_{ie} \exp(E_{t_i}/kT) \quad (3.3)$$

and

$$p_{t_i} = n_{ie} \exp(-E_{t_i}/kT) \quad (3.4)$$

and E_{t_i} is the trap energy level specified relative to the intrinsic Fermi level (eV). Poisson's equation is modified by the inclusion of the number of electrons trapped with addition:

$$-q \sum_i N_{t_i} f_{i_n}. \quad (3.5)$$

Where N_{t_i} is the total number of traps (in $\#/cm^3/eV$) for the i th energy level, and f_{i_n} is the trap occupation function for electron traps where:

$$f_{i_n} = \frac{\tau_{p_i} n + \tau_{n_i} p_{t_i}}{\tau_{p_i} (n + n_{t_i}) + \tau_{n_i} (p + p_{t_i})} \quad (3.6)$$

The contribution for the number of holes trapped on the RHS of Poisson's equation is:

$$-q \sum_i N_{t_i} f_{i_p} \quad (3.7)$$

where the trap occupation function for holes is:

$$f_{i_p} = \frac{\tau_{n_i} p + \tau_{p_i} n_{t_i}}{\tau_{p_i} (n + n_{t_i}) + \tau_{n_i} (p + p_{t_i})} \quad (3.8)$$

These methods are defined within the DaVinci manual [6] on pp.2-150–2-152.

The method fuction may be set to up to 50 trap values. If $E_{<i>i</i>}$ is set then $N_TOTAL_{<i>i</i>}$ must also be set. The TRAPS option requires at least one E and N_TOTAL value. If the TAUP or TAUN values aren't set to a real value, then they are calculated by **Charon**.

```
METHOD FUNCTION <n>
  KEY = TRAPS
  MODEL NAME = <NO TRAPS|TRAPS>
  E0 = <real value> [TAUP0 = <real value>] [TAUN0 = <real value>] \
    N_TOTAL0 = <real value>
  .
  .
  .
  E49 = <real value> [TAUP49 = <real value>] [TAUN49 = <real value>] \
    N_TOTAL49 = <real value>
END
```

or

```
METHOD FUNCTION <n>
  KEY = TRAPS
  MODEL NAME = MGAUSS
  E0 = <real value> [TAUP0 = <real value>] [TAUN0 = <real value>] \
    N_TOTAL0 = <real value>
  .
  .
  .
  E49 = <real value> [TAUP49 = <real value>] [TAUN49 = <real value>] \
    N_TOTAL49 = <real value>
  XMIN = <real value>
  XMAX = <real value>
  XCHAR = <real value>
  XERFC = <TRUE|FALSE>
  YMIN = <real value>
  YMAX = <real value>
  YCHAR = <real value>
END
```

where:

► **MODEL NAME**

- ▷ NO_TRAPS - no traps are set (the default value)
- ▷ TRAPS - setting traps via the equations above
- ▷ MGAUSS - setting traps via the equations above and falls off as a gaussian distribution outside a specified range

► **REAL PARAMETERS FOR TRAP <i> (<i> = 0 - 49)**

- ▷ E<i> - the trap energy level relative to the intrinsic Fermi level (eV)
- ▷ TAUP<i> - the hole lifetime (s) [if not set uses the value calculated within **Charon**]
- ▷ TAUN<i> - the electron lifetime (s) [if not set uses the value calculated within **Charon**]
- ▷ N_TOTAL<i> - the total number of traps ($\#/cm^3/eV$), when > 0.0 this is a neutral electron trap, when < 0.0 this is a neutral hole trap, when $= 0.0$ there is no Poisson contribution
- ▷ XMIN - The value of the X dimension where the uniform profile starts (when using Gaussian traps).
- ▷ XMAX - The value of the X dimension where the uniform profile ends (when using Gaussian traps).
- ▷ XCHAR - The characteristic length of the gaussian profile outside of the uniform region in the x-direction.
- ▷ XERFC - Specifies that the x variation of the profile is described by a complementary error function.
- ▷ YMIN - The value of the Y dimension where the uniform profile starts (for gaussian traps option).
- ▷ YMAX - The value of the Y dimension where the uniform profile ends (for gaussian traps option).
- ▷ YCHAR - The characteristic length of the gaussian profile outside of the uniform region in the y-direction.

Notes:

- If you are using nonzero N_TOTAL values, and the value is of the same order as the doping concentration, then the boundary conditions are going to be modified to account for this. This means if you then wish to turn traps off, in addition to commenting out the **TRAPS = METHODFUNCTION <n>** in the material properties section of the input file you also need to comment out the *METHOD FUNCTION* specification itself to avoid this modification of the boundary conditions.

DEFECTS

NOTE: The defect species must be defined within the physics specification via the region specification.

Defects are specified within a particular physics region and also as a material property so that the defect properties may be set for a given material and/or model.

```
METHOD FUNCTION <n>
  KEY = DEFECTS
  MODEL NAME = NO DEFECTS
END

METHOD FUNCTION <n>
  KEY = DEFECTS
  MODEL NAME = CHEMKIN CONSTANT
  CHEMKIN FILENAME = <chemkin_file_name>
  DIFF_COEF_<species_name_1> = <real_value>
  .
  .
  DIFF_COEF_<species_name_n> = <real_value>
  SOURCE_RATE_E = <real_value>
  SOURCE_RATE_H = <real_value>
  SOURCE_RATE_<species_name_1> = <real_value>
  .
  .
  SOURCE_RATE_<species_name_n> = <real_value>
END

METHOD FUNCTION <n>
  KEY = DEFECTS
  MODEL NAME = CHEMKIN REACTION
  CHEMKIN FILENAME = <chemkin_file_name>
  DIFF_PRE_<species_name_1> = <real_value>
  .
  .
  DIFF_PRE_<species_name_n> = <real_value>
  ENERGY_A_<species_name_1> = <real_value>
  .
  .
```

```
ENERGY_A_<species_name_n> = <real_value>
END
```

where:

► MODEL NAME

- ▷ NO DEFECTS - no defects are set (the default value)
- ▷ CHEMKIN CONSTANT - use chemkin file to set up the defect species and all parameters (diffusion and source rate) are constant values. NOTE, that the electron and hole source rates must be specified
- ▷ CHEMKIN REACTION - use chemkin file to set up the defect species and reaction rates, the diffusion term has the following form $D = D_o \exp\left(\frac{E_D}{kT}\right)$ where D_o is the diffusion coefficient prefactor, E_D is the activation energy, and T is the lattice temperature

► std::string parameters:

- ▷ CHEMKIN_FILENAME = set chemkin_file_name to string name of the chemkin file within quotes
- ▷ species_name_i = the species name defined within the CHEMKIN file

► REAL parameters for defect species_name = i

- ▷ DIFF_COEF_i = the diffusion coefficient for species i [cm^2/s]
- ▷ SOURCE_RATE_i = the source rate for species i [$\#/cm^3/s$]
- ▷ DIFF_PRE_i = the diffusion prefactor (D_o) [cm^2/s]
- ▷ ENERGY_A_i = the defect activation energy (E_D) [eV]

DOPING

Doping is a material property and must be specified as such from within the input file. If the doping material property is of type "METHOD FUNCTION" within the MODEL section (see section 3.2) then the following information is pertinent

```
METHOD FUNCTION <n>
KEY = DOPING
```

```

MODEL NAME = STEP JUNCTION
NA = <real value>
ND = <real value>
[CONFIGURATION = <NP|PN>]
[DIRECTION = <X|Y|Z>]
[JUNCTION LOCATION = <real value>]
END

```

or

```

METHOD FUNCTION <n>
KEY = DOPING
MODEL NAME = FILE
DOPE TO MESH = <ONETOTWO|TWOOTOWO>
FIT METHOD = <INTERPOLATE|CUBIC SPLINE|INVERSE WEIGHTED AVERAGE>
[NA FILE NAME = "<file name>"]
[ND FILE NAME = "<file name>"]
[FILE NAME = "<file name>"]
[IWA NUMBER OF POINTS = <n>]
[SPATIAL CGS SCALE FACTOR = <real value>]
[DOPING CGS SCALE FACTOR = <real value>]
[INTERPOLATE COORD = <X|Y|Z>]
[REVERSEY = <TRUE|FALSE>]
[PARTITION DATA = <TRUE|FALSE>]
[IONIZATION = <TRUE|FALSE>]
END

```

where:

- ▶ DOPE TO MESH - is the method used to place the doping information onto a **Charon** FEM mesh.
 - ▷ ONETOTWO - take 1-dimensional doping data in the file(s) and place it onto the 2-dimensional **Charon** mesh.
 - ▷ TWOOTOWO - take 2-dimensional doping data in the file(s) and place it onto the 2-dimensional **Charon** mesh.
- ▶ FIT METHOD - The method used to fit the doping data to the **Charon** FEM mesh.

- ▷ INTERPOLATE - Interpolate the data. Finds the points within the doping data within which a point lies and interpolates between them to find the value. This is only relevant when using "DOPE TO MESH = ONETOTWO"
- ▷ CUBIC SPLINE - Uses a cubic spline interpolation. This is only relevant when using "DOPE TO MESH = ONETOTWO"
- ▷ INVERSE WEIGHTED AVERAGE - Use an inverse weighted average to find the doping value at a specified point in the **Charon** mesh. This is only relevant when using "DOPE TO MESH = TWOTOTWO"
- ▶ IWA NUMBER OF POINTS - The number of points to use for the "INVERSE WEIGHTED AVERAGE". (Default is 3)
- ▶ SPATIAL CGS SCALE FACTOR - **Charon** expects spatial data within the doping file(s) to be in CGS units (cm). If it is not then you can use this scaling factor to convert it to CGS units.
- ▶ DOPING CGS SCALE FACTOR - **Charon** expects doping values to be in CGS units (# per cubic centimeter). If the doping data within the file is not in these units you can apply a scaling factor here to convert it to CGS units.
- ▶ INTERPOLATE COORD - The coordinate direction to use when interpolating 1-dimensional data to a 2-dimensional **Charon** mesh. This is only relevant for "DOPE TO MESH = ONETOTWO" and "FIT METHOD = INTERPOLATE". (Default is "X")
- ▶ REVERSEY - Most other TCAD codes define the upper left corner of the device as 0,0 with y increasing as you go down. **Charon** uses a standard coordinate scheme and so if the doping data is from another TCAD that uses the upper left corner as the origin this option will map it correctly to **Charon**'s coordinate system.
- ▶ PARTITION DATA - This option partitions the doping data into rectangular regions and then uses these "sections" to reduce the search area when using "INVERSE WEIGHTED AVERAGE" on 2D doping data being placed on a 2D **Charon** mesh. By default this option is true if the number of doping data points is greater than 1000.
- ▶ File specification. Generally you can either specify one file, via "FILE NAME = <file name>" or two files, one of which stores the donor profile and one of which stores the acceptor profile via "ND FILE NAME = <file name>" and "NA FILE NAME = <file name>". The file format is ASCII with tab-separated columns. For single file reads the data should be in the format "x Nd Na" for 1D doping data or "x y Nd Na" for 2D doping data.
- ▶ IONIZATION - This is set to true when the incomplete ionization model is needed (see section 3.5).

or

```

METHOD FUNCTION <n>
  KEY = DOPING
  MODEL NAME = ONEDIM PROFILE
  FIT METHOD = <INTERPOLATE|CUBIC SPLINE>
  [IONIZATION = <TRUE|FALSE>]

  FILE_NAME_1 = "<file name>"
  XMIN_1 = <real value>
  XMAX_1 = <real value>
  [XCHAR_1 = <real value> | XYRATIO_1 = <real value>]
  [XERFC_1 = <TRUE|FALSE>]
  [YMIN_1 = <real value>]
  [YMAX_1 = <real value>]
  [SPATIAL_CGS_SCALE_FACTOR_1 = <real value>]
  [DOPING_CGS_SCALE_FACTOR_1 = <real value>]
  [REVERSEY_1 = <TRUE|FALSE>]
END

```

where:

- ▶ **FIT METHOD** - The method used to fit the doping data to the **Charon** FEM mesh.
 - ▷ **INTERPOLATE** - Interpolate the data. Finds the points within the doping data within which a point lies and interpolates between them to find the value.
 - ▷ **CUBIC SPLINE** - Uses a cubic spline interpolation.
- ▶ **IONIZATION** - This is set to true when the incomplete ionization model is needed.
- ▶ **XMIN_1** - The value of the X dimension where the uniform region starts.
- ▶ **XMAX_1** - The value of the X dimension where the uniform region ends.
- ▶ **XCHAR_1** - The characteristic length of the gaussian profile outside of the uniform region in the x-direction. If the user does not specify a value for either XCHAR or XYRATIO, then a default value of 0.0002 is used for XCHAR.
- ▶ **XYRATIO_1** - This is the factor which multiplies the extent of the profile when the profile is rotated to the horizontal direction.

- ▶ XERFC_1 - Specifies that the x variation of the profile is described by a complementary error function.
- ▶ YMIN_1 - The user can optionally specify the minimum value in y of the input profile. If this value is specified, the profile drops to zero below this y-value. If not specified by the user, the doping concentration will be set equal to the concentration at the top and bottom points of the input profile.
- ▶ YMAX_1 - The user can optionally specify the maximum value in y of the input profile. If this value is specified, the profile drops to zero above this y-value. If not specified by the user, the doping concentration will be set equal to the concentration at the top and bottom points of the input profile.
- ▶ SPATIAL_CGS_SCALE_FACTOR_1 - **Charon** expects spatial data within the doping file(s) to be in CGS units (cm). If it is not, then you can use this scaling factor to convert it to CGS units.
- ▶ DOPING_CGS_SCALE_FACTOR_1 - **Charon** expects doping values to be in CGS units (# per cubic centimeter). If the doping data within the file is not in these units you can apply a scaling factor here to convert it to CGS units.
- ▶ REVERSEY_1 - Most other TCAD codes define the upper left corner of the device as 0,0 with y increasing as you go down. **Charon** uses a standard coordinate scheme and so if the doping data is from another TCAD that uses the upper left corner as the origin this option will map it correctly to **Charon**'s coordinate system.

or

```
METHOD FUNCTION <n>
  KEY = DOPING
  MODEL NAME = FUNCTION

  FUNCTION_<m> = <UNIFORM|LERFC|GAUSSIAN|M_GAUSSIAN|POLYGON|Y_JUNCTION|LINEAR GRADING>
  [IONIZATION = <TRUE|FALSE>]
END
```

where the overall doping is determined by calculating all of the functions and summing the result and the parameters are:

- ▶ <m> - is 1,2,3,...,<number of functions>

- ▶ FUNCTION_<m>
 - ▷ UNIFORM
 - ▷ LERFC
 - ▷ GAUSSIAN
 - ▷ M_GAUSSIAN
 - ▷ POLYGON
 - ▷ Y_JUNCTION
 - ▷ LINEAR GRADING
- ▶ IONIZATION - This is set to true when the incomplete ionization model is needed (see section 3.5).

Specification for uniform doping

This section describes the doping specification for doping whose profile is uniform over a specified region.

```
METHOD FUNCTION = <n>
  KEY = DOPING
  MODEL NAME = FUNCTION

  FUNCTION_1 = UNIFORM
  NMAX_1 = <real value>
  TYPE_1 = <P|N>
  XMIN_1 = <real value>
  XMAX_1 = <real value>
  YMIN_1 = <real value>
  YMAX_1 = <real value>
  ZMIN_1 = <real value>
  ZMAX_1 = <real value>
END
```

where:

Note that only one coordinate direction is required. Additional coordinate directions can be used to block out a region.

- ▶ FUNCTION_1 = UNIFORM - specifies that the doping is a uniform variation. The "1" indicates that this is function number one. You can have any number of functions, which will be summed together to form the overall doping profile, but the function numbers must start at "1" and be sequential up to the number of functions.
- ▶ NMAX_1 - The value of the doping within the defined region. Note that "MAX" is used for consistency with other functions. Since this is a uniform doping there is no variation.
- ▶ TYPE_1 - Either "P" or "N" to indicate acceptor or donor doping, respectively.
- ▶ XMIN_1 - The value of the X dimension where the uniform region starts.
- ▶ XMAX_1 - The value of the X dimension where the uniform region ends.
- ▶ YMIN_1 - The value of the Y dimension where the uniform region starts.
- ▶ YMAX_1 - The value of the Y dimension where the uniform region ends.
- ▶ ZMIN_1 - The value of the Y dimension where the uniform region starts.
- ▶ ZMAX_1 - The value of the Y dimension where the uniform region ends.

Specification for log10(erfc()) doping

This section describes the doping specification for doping whose variation can be described by the base 10 logarithm of a complementary error function.

```
METHOD FUNCTION = <n>
KEY = DOPING
MODEL NAME = FUNCTION

FUNCTION_1 = LERFC
SIGN_1 = <-1|1>
NMAX_1 = <real value>
NMIN_1 = <real value>
LOCATION_1 = <real value>
WIDTH_1 = <real value>
TYPE_1 = <P|N>
DIRECTION_1 = <X|Y|Z>
END
```

The equation governing the doping is given by:

$$N = 10^{\left[\frac{\log(NMAX) - \log(NMIN)}{2} \operatorname{erfc}\left(\frac{\operatorname{SIGN}(\operatorname{DIRECTION} - \operatorname{LOCATION})}{\operatorname{WIDTH}}\right) + \log(NMIN) \right]} \quad (3.9)$$

where:

- ▶ FUNCTION_1 = LERFC - specifies that the doping is described by the base 10 logarithm of a complementary error function. The "1" indicates that this is function number one. You can have any number of functions, which will be summed together to form the overall doping profile, but the function numbers must start at "1" and be sequential up to the number of functions.
- ▶ SIGN_1 - The direction of the complementary error function. "-1" means it increases in the positive x direction, "1" means it decreases in the positive x direction.
- ▶ NMAX_1 - The value of the doping at the maximum of the complementary error function.
- ▶ NMIN_1 - The value of the doping at the minimum of the complementary error function.
- ▶ TYPE_1 - Either "P" or "N" to indicate acceptor or donor doping, respectively.
- ▶ LOCATION_1 - The location where the complementary error function goes from min to max.
- ▶ WIDTH_1 - The width of the erfc() region
- ▶ DIRECTION_1 - "X", "Y" or "Z" direction of the variation of the complementary error function.

Specification for gaussian doping

This section describes the doping specification for doping whose variation can be described by a gaussian function.

```
METHOD FUNCTION = <n>
KEY = DOPING
MODEL NAME = FUNCTION

FUNCTION_1 = <GAUSSIAN|GAUSS>
```

```

NMAX_1 = <real value>
NMIN_1 = <real value>
TYPE_1 = <P|N>

XLOC_1 = <real value>
XWIDTH_1 = <real value>
FLATX_1 = <-1|0|1>
XRANGE_MIN_1 = <real value>
XRANGE_MAX_1 = <real value>

YLOC_1 = <real value>
YWIDTH_1 = <real value>
FLATY_1 = <-1|0|1>
YRANGE_MIN_1 = <real value>
YRANGE_MAX_1 = <real value>

ZLOC_1 = <real value>
ZWIDTH_1 = <real value>
FLATZ_1 = <-1|0|1>
ZRANGE_MIN_1 = <real value>
ZRANGE_MAX_1 = <real value>

```

END

You must specify at least one direction completely, and you can specify all three in which case the doping will be a product of all the gaussians.

Be careful when using the ranges, XRANGE_MIN/XRANGE_MAX for example, as you can end up with discontinuities in the doping. Range is used as in $\text{XRANGE_MIN} \leq x < \text{XRANGE_MAX}$.

The equation governing this distribution, in one dimension, is:

$$N = NMAX e^{\left[-\ln\left(\frac{NMAX}{NMIN}\right) \left(\frac{x-XLOC}{XWIDTH}\right)^2 \right]} \quad (3.10)$$

where:

- FUNCTION_1 = GAUSS - specifies that the doping has a gaussian variation. The "1" indicates that this is function number one. You can have any number of functions,

which will be summed together to form the overall doping profile, but the function numbers must start at "1" and be sequential up to the number of functions.

- ▶ NMAX_1 - The maximum value (gauss peak) of the doping.
- ▶ NMIN_1 - The minimum value of the doping at the gauss drop off. This is used to modulate how fast the gaussian distribution falls off.
- ▶ TYPE_1 - Either "P" or "N" to indicate acceptor or donor doping, respectively.
- ▶ For each of X, Y, Z:
 - ▷ XLOC_1 - The location of the gaussian peak.
 - ▷ XWIDTH_1 - The width of the gaussian distribution.
 - ▷ FLATX_1 - Specifies either a single direction gaussian in the negative x direction (-1), the positive x direction (1), or a bidirectional (0) gaussian falling off in both directions.
 - ▷ XRANGE_MIN_1 - The minimum value of X, inclusive, at which to turn on this function (optional, unless XRANGE_MAX is specified)
 - ▷ XRANGE_MAX_1 - The maximum value of X, exclusive, at which to turn off this function (optional, unless XRANGE_MIN is specified).

Notes:

- ▶ Charon's gaussian is similar to the Medici-style gaussian, but uses different parameters. You can convert to Medici style parameters for characteristic length with the following formula:

$$x.char = \frac{XWIDTH}{\sqrt{\ln\left(\frac{NMAX}{NMIN}\right)}} \quad (3.11)$$

Specification for Medici-style gaussian doping

This is the doping specification for doping whose variation can be described by a gaussian function similar to the way Medici does it.

```
METHOD FUNCTION = <n>
KEY = DOPING
MODEL NAME = FUNCTION
```

FUNCTION_1 = <M_GAUSSIAN|M_GAUSS>

NMAX_1 = <real value>

TYPE_1 = <P|N>

XMIN_1 = <real value>

XMAX_1 = <real value>

XCHAR_1 = <real value>

XERFC_1 = <TRUE|FALSE>

YMIN_1 = <real value>

YMAX_1 = <real value>

YCHAR_1 = <real value>

END

You must specify both X and Y parameters.

The equation governing this distribution, is

$$N = NMAX \cdot b(y) \cdot a(x) \quad (3.12)$$

where:

$$\blacktriangleright b(y) = \begin{cases} e^{-\left(\frac{y-YMIN}{YCHAR}\right)^2} & \text{if } y < YMIN \\ 1 & \text{if } YMIN \leq y \leq YMAX \\ e^{-\left(\frac{y-YMAX}{YCHAR}\right)^2} & \text{if } y > YMAX \end{cases}$$

$$\blacktriangleright a(x) = \begin{cases} a_1(x) & \text{XERFC is FALSE} \\ a_2(x) & \text{XERFC is TRUE} \end{cases}$$

where:

$$\blacktriangleright a_1(x) = \begin{cases} e^{-\left(\frac{x-XMIN}{XCHAR}\right)^2} & \text{if } x < XMIN \\ 1 & \text{if } XMIN \leq x \leq XMAX \\ e^{-\left(\frac{x-XMAX}{XCHAR}\right)^2} & \text{if } x > XMAX \end{cases}$$

$$\blacktriangleright a_2(x) = \frac{\operatorname{erfc}\left(\frac{x-XMAX}{XCHAR}\right) - \operatorname{erfc}\left(\frac{x-XMIN}{XCHAR}\right)}{2}$$

If **XERFC** is not specified then it's default value is FALSE.

Specification for polygon doping

This section describes the doping specification for a profile which is a polygon over a specified region. Outside the polygon region and inside the bounding box, the profile falls off as a gaussian.

```
METHOD FUNCTION = <n>
  KEY = DOPING
  MODEL NAME = FUNCTION

  FUNCTION_1 = POLYGON
  NMAX_1 = <real value>
  TYPE_1 = <P|N>
  XMIN_1 = <real value>
  XMAX_1 = <real value>
  YMIN_1 = <real value>
  YMAX_1 = <real value>

  XPOLYMIN_1 = <real value>
  XPOLYMAX_1 = <real value>
  YPOLYMIN_1 = <real value>
  YPOLYMAX_1 = <real value>
  RCHAR_1 = <real value>
END
```

where:

- ▶ **FUNCTION_1 = POLYGON** - specifies that the doping has a uniform variation inside the inner polygon region and falls off as a gaussian outside the inner polygon and inside the bounding box. The "1" indicates that this is function number one. You can have any number of functions, which will be summed together to form the overall doping profile, but the function numbers must start at "1" and be sequential up to the number of functions.
- ▶ **NMAX_1** - The value of the doping within the defined region. Note that "MAX" is used for consistency with other functions.

- ▶ TYPE_1 - Either "P" or "N" to indicate acceptor or donor doping, respectively.
- ▶ XPOLYMIN_1 - The value of the X dimension where the uniform region starts.
- ▶ XPOLYMAX_1 - The value of the X dimension where the uniform region ends.
- ▶ YPOLYMIN_1 - The value of the Y dimension where the uniform region starts.
- ▶ YPOLYMAX_1 - The value of the Y dimension where the uniform region ends.
- ▶ XMIN_1 - The value of the X dimension where the gaussian extension starts.
- ▶ XMAX_1 - The value of the X dimension where the gaussian extension ends.
- ▶ YMIN_1 - The value of the Y dimension where the gaussian extension starts.
- ▶ YMAX_1 - The value of the Y dimension where the gaussian extension ends.
- ▶ RCHAR_1 = The characteristic length of the gaussian profile outside of the polygon region and inside the bounding box.

Specification for yjunction doping

This is the doping specification for a profile that matches the background doping at a specified depth.

```
METHOD FUNCTION = <n>
  KEY = DOPING
  MODEL NAME = FUNCTION

  FUNCTION_1 = Y_JUNCTION
  NMAX_1 = <real value>
  TYPE_1 = <P|N>
  XMIN_1 = <real value>
  XMAX_1 = <real value>
  YMIN_1 = <real value>
  YMAX_1 = <real value>
  [XCHAR_1 = <real value> | XYRATIO_1 = <real value>]
  BACKDOPE_1 = <real value>
  YJUNC_1 = <real value>
  [XERFC_1 = <TRUE|FALSE>]
END
```

where:

- ▶ **FUNCTION_1 = Y_JUNCTION** - specifies that the doping has a uniform variation within a specified region and falls off as a gaussian outside this region, matching the background doping concentration at a specified depth. The "1" indicates that this is function number one. You can have any number of functions, which will be summed together to form the overall doping profile, but the function numbers must start at "1" and be sequential up to the number of functions.
- ▶ **NMAX_1** - The value of the doping within the defined region. Note that "MAX" is used for consistency with other functions.
- ▶ **TYPE_1** - Either "P" or "N" to indicate acceptor or donor doping, respectively.
- ▶ **XMIN_1** - The value of the X dimension where the uniform profile starts.
- ▶ **XMAX_1** - The value of the X dimension where the uniform profile ends.
- ▶ **YMIN_1** - The value of the Y dimension where the uniform profile starts.
- ▶ **YMAX_1** - The value of the Y dimension where the uniform profile ends.
- ▶ **XCHAR_1** - The characteristic length of the gaussian profile outside of the uniform region in the x-direction.
- ▶ **XYRATIO_1** - The ratio of the horizontal characteristic length to the vertical characteristic length.
- ▶ **BACKDOPE_1** - The background doping concentration, which, together with the junction depth, is used to calculate the characteristic length of the profile in the y-direction.
- ▶ **YJUNC_1** - The y-location under the center of the profile where the magnitude of the profile being added equals the magnitude of the background profile.
- ▶ **XERFC_1** - Specifies that the x variation of the profile is described by a complementary error function.

Specification for linearly-graded doping

Doping specification for doping whose profile is linearly graded over the specified region in either the x- or y-direction.

```
METHOD FUNCTION = <n>
  KEY = DOPING
  MODEL NAME = FUNCTION

  FUNCTION_1 = LINEAR_GRADING
  NBEGIN_1 = <real value>
  NEND_1 = <real value>
  TYPE_1 = <P|N>
  [GRADING_1 = X_LINEAR
  XMIN_1 = <real value>
  XMAX_1 = <real value>]
  [GRADING_ = Y_LINEAR
  YMIN_1 = <real value>
  YMAX_1 = <real value>]
END
```

where:

- ▶ **FUNCTION_1 = LINEAR_GRADING** - specifies that the doping is a linearly graded variation. The "1" indicates that this is function number one. You may have any number of functions, which will be summed together to form the overall doping profile, but the function numbers must start at "1" and be sequential up to the number of functions.
- ▶ **NBEGIN_1** - The value of doping at the left or bottom, depending on whether the grading is in the x-direction or y-direction.
- ▶ **NEND_1** - The value of doping at the top or right, depending on whether the grading is in the x-direction or y-direction.
- ▶ **TYPE_1** - Either "P" or "N" to indicate acceptor or donor doping, respectively.
- ▶ **GRADING_1** - Set to X_LINEAR if the grading is in the x-direction or set to Y_LINEAR if the grading is in the y-direction.
- ▶ **XMIN_1** - Left boundary of the graded region for grading in x-direction.
- ▶ **XMAX_1** - Right boundary of the graded region for grading in x-direction.
- ▶ **YMIN_1** - Bottom boundary of the graded region for grading in y-direction.
- ▶ **YMAX_1** - Top boundary of the graded region for grading in y-direction.

Initial Species Density

The initial defect density is a material property and is specified with a METHOD FUNCTION within a MODEL (see section 3.2) that allows the user to specify the initial density distribution for a given defect species. Note that this does NOT include the electron and hole species. This function will eventually be expanded to be as flexible as the doping specification.

```
METHOD FUNCTION <n>
  KEY = INITIAL SPECIES DENSITY
  MODEL NAME = CONSTANT
  INITIAL_DENSITY_<species_name_1> = <real value>
  .
  .
  INITIAL_DENSITY_<species_name_n> = <real value>
END
```

or

```
METHOD FUNCTION <n>
  KEY = INITIAL SPECIES DENSITY
  MODEL NAME = FILE
  FIT METHOD = <INTERPOLATE|CUBIC SPLINE>
  DOPE TO MESH = ONEtoTWO
  <species_name_1> FILE NAME = "<string_file_name>"
  .
  .
  <species_name_n> FILE NAME = "<string_file_name>"
  SPATIAL CGS SCALE FACTOR = <real value>
  [SPATIAL CGS SCALE FACTOR = <real value>]
  [DOPING CGS SCALE FACTOR = <real value>]
  [INTERPOLATE COORD = <X|Y|Z>]
  [IONIZATION = <TRUE|FALSE>]
END
```

where:

- MODEL NAME

- ▷ CONSTANT - A constant density value is set for a given species in units of $[\#/cm^3]$. The species list is specified within the physics specification via the region specification.
- ▷ FILE - A file is read in for each species specified (via `<species_name> FILE NAME`). The file format is ASCII with tab-separated columns. If a species file is not specified the default initial distribution is 0.0 over all space.

► Model Specifications

- ▷ DOPE TO MESH - is the method used to place the doping information onto a **Charon** FEM mesh.
 - * ONETOTWO - take 1-dimensional doping data in the file(s) and place it onto the 2-dimensional **Charon** mesh.
- ▷ FIT METHOD - The method used to fit the doping data to the **Charon** FEM mesh.
 - * INTERPOLATE - Interpolate the data. Finds the points within the doping data within which a point lies and interpolates between them to find the value. This is only relevant when using "DOPE TO MESH = ONETOTWO"
 - * CUBIC SPLINE - Uses a cubic spline interpolation. This is only relevant when using "DOPE TO MESH = ONETOTWO"
- ▷ SPATIAL CGS SCALE FACTOR - **Charon** expects spatial data within the doping file(s) to be in CGS units (cm). If it is not then you can use this scaling factor to convert it to CGS units.
- ▷ DOPING CGS SCALE FACTOR - **Charon** expects doping values to be in CGS units (number per cubic centimeter). If the density data within the file is not in these units you can apply a scaling factor here to convert it to CGS units.
- ▷ INTERPOLATE COORD - The coordinate direction to use when interpolating 1-dimensional data to a 2-dimensional **Charon** mesh. This is only relevant for "DOPE TO MESH = ONETOTWO" and "FIT METHOD = INTERPOLATE". (Default is "X")
- ▷ File specification. A file for a given species is specified via "`<species_name> FILE NAME = <file name>`". The file format is ASCII with tab-separated columns. For single file reads the data should be in the format "x density" for 1D species density.
- ▷ IONIZATION - Set to true when the incomplete ionization model is needed (see section 3.5).

or

```

METHOD FUNCTION <n>
  KEY = INITIAL SPECIES DENSITY
  MODEL NAME = FUNCTION

  FUNCTION_<m> = <UNIFORM|LERFC|GAUSSIAN|M_GAUSSIAN|POLYGON|Y_JUNCTION>
  [IONIZATION = <TRUE|FALSE>]
END

```

where the overall initial species density is determined by calculating all of the functions and summing the result and the parameters are:

- ▶ <m> - is 1,2,3,..., i.e., number of functions
- ▶ IONIZATION - This is set to true when the incomplete ionization model is needed (see section 3.5).

Specification for uniform initial species density

This section describes the initial species density specification for a species whose profile is uniform over a specified region.

```

METHOD FUNCTION = <n>
  KEY = INITIAL SPECIES DENSITY
  MODEL NAME = FUNCTION

  FUNCTION_1 = UNIFORM
  NMAX_1 = <real value>
  TYPE_1 = <species_name_1>
  XMIN_1 = <real value>
  XMAX_1 = <real value>
  YMIN_1 = <real value>
  YMAX_1 = <real value>
  ZMIN_1 = <real value>
  ZMAX_1 = <real value>
END

```

where: Note that only one coordinate direction is required. Additional coordinate directions can be used to block out a region.

- ▶ **FUNCTION_1 = UNIFORM** - specifies that the species has a uniform variation. The "1" indicates that this is function number one. You can have any number of functions, which will be summed together to form the overall species profile, but the function numbers must start at "1" and be sequential up to the number of functions.
- ▶ **NMAX_1** - The value of the initial species density within the defined region. Note that "MAX" is used for consistency with other functions. Since this is a uniform doping there is no variation.
- ▶ **TYPE_1** - The species name goes here.
- ▶ **XMIN_1** - The value of the *X* dimension where the uniform region starts.
- ▶ **XMAX_1** - The value of the *X* dimension where the uniform region ends.
- ▶ **YMIN_1** - The value of the *Y* dimension where the uniform region starts.
- ▶ **YMAX_1** - The value of the *Y* dimension where the uniform region ends.
- ▶ **ZMIN_1** - The value of the *Y* dimension where the uniform region starts.
- ▶ **ZMAX_1** - The value of the *Y* dimension where the uniform region ends.

Specification for $\log_{10}(\text{erfc}())$ initial species density

This section describes the initial species density specification for species whose variation can be described by the base 10 logarithm of a complementary error function.

```

METHOD FUNCTION = <n>
KEY = INITIAL SPECIES DENSITY
MODEL NAME = FUNCTION

FUNCTION_1 = LERFC
SIGN_1 = <-1|1>
NMAX_1 = <real value>
NMIN_1 = <real value>
LOCATION_1 = <real value>
WIDTH_1 = <real value>
TYPE_1 = <species_name_1>
DIRECTION_1 = <X|Y|Z>
END
    
```

The equation governing the species density is given by:

$$N = 10^{\left[\frac{\log(NMAX) - \log(NMIN)}{2} \operatorname{erfc}\left(\frac{\operatorname{SIGN}(\operatorname{DIRECTION} - \operatorname{LOCATION})}{\operatorname{WIDTH}}\right) + \log(NMIN) \right]} \quad (3.13)$$

where:

- ▶ FUNCTION_1 = LERFC - specifies that the species density is described by the base 10 logarithm of a complementary error function. The "1" indicates that this is function number one. You can have any number of functions, which will be summed together to form the overall species profile, but the function numbers must start at "1" and be sequential up to the number of functions.
- ▶ SIGN_1 - The direction of the complementary error function. "-1" means it increases in the positive x direction, "1" means it decreases in the positive x direction.
- ▶ NMAX_1 - The value of the species density at the maximum of the complementary error function.
- ▶ NMIN_1 - The value of the species density at the minimum of the complementary error function.
- ▶ TYPE_1 - The species name.
- ▶ LOCATION_1 - The location where the complementary error function goes from min to max.
- ▶ WIDTH_1 - The width of the erfc() region
- ▶ DIRECTION_1 - "X", "Y" or "Z" direction of the variation of the complementary error function.

Specification for gaussian initial species density

This section describes the initial species density specification for a species whose variation can be described by a gaussian function.

```
METHOD FUNCTION = <n>
KEY = INITIAL SPECIES DENSITY
MODEL NAME = FUNCTION
```



```

FUNCTION_1 = <GAUSSIAN|GAUSS>
NMAX_1 = <real value>
NMIN_1 = <real value>
TYPE_1 = <species_name_1>

XLOC_1 = <real value>
XWIDTH_1 = <real value>
FLATX_1 = <-1|0|1>
XRANGE_MIN_1 = <real value>
XRANGE_MAX_1 = <real value>

YLOC_1 = <real value>
YWIDTH_1 = <real value>
FLATY_1 = <-1|0|1>
YRANGE_MIN_1 = <real value>
YRANGE_MAX_1 = <real value>

ZLOC_1 = <real value>
ZWIDTH_1 = <real value>
FLATZ_1 = <-1|0|1>
ZRANGE_MIN_1 = <real value>
ZRANGE_MAX_1 = <real value>

```

END

You must specify at least one direction completely, and you can specify all three in which case the species density will be a product of all the gaussians.

Be careful when using the ranges, XRANGE_MIN/XRANGE_MAX for example, as you can end up with discontinuities in the species density. Range is used as in $\text{XRANGE_MIN} \leq x < \text{XRANGE_MAX}$.

The equation governing this distribution, in one dimension, is:

$$N = NMAX e^{\left[-\ln\left(\frac{NMAX}{NMIN}\right) \left(\frac{x-XLOC}{XWIDTH}\right)^2 \right]} \quad (3.14)$$

where:

- FUNCTION_1 = GAUSS - specifies that the species density has a gaussian variation. The "1" indicates that this is function number one. You can have any number of

functions, which will be summed together to form the overall species profile, but the function numbers must start at "1" and be sequential up to the number of functions.

- ▶ NMAX_1 - The maximum value (gauss peak) of the species density.
- ▶ NMIN_1 - The minimum value of the species density at the gauss drop off. This is used to modulate how fast the gaussian distribution falls off.
- ▶ TYPE_1 - The species name.
- ▶ For each of X, Y, Z:
 - ▷ XLOC_1 - The location of the gaussian peak.
 - ▷ XWIDTH_1 - The width of the gaussian distribution.
 - ▷ FLATX_1 - Specifies either a single direction gaussian in the negative x direction (-1), the positive x direction (1), or a bidirectional (0) gaussian falling off in both directions.
 - ▷ XRANGE_MIN_1 - The minimum value of X, inclusive, at which to turn on this function (optional, unless XRANGE_MAX is specified)
 - ▷ XRANGE_MAX_1 - The maximum value of X, exclusive, at which to turn off this function (optional, unless XRANGE_MIN is specified).

Specification for Medici-style gaussian initial species density

This section describes the initial species density specification for species whose variation can be described by a gaussian function similar to the way Medici does it.

```
METHOD FUNCTION = <n>
KEY = INITIAL SPECIES DENSITY
MODEL NAME = FUNCTION

FUNCTION_1 = <M_GAUSSIAN|M_GAUSS>
NMAX_1 = <real value>
TYPE_1 = <species_name_1>

XMIN_1 = <real value>
XMAX_1 = <real value>
XCHAR_1 = <real value>
XERFC_1 = <TRUE|FALSE>
```

```
YMIN_1 = <real value>
YMAX_1 = <real value>
YCHAR_1 = <real value>
```

END

You must specify both X and Y parameters.

The equation governing this distribution, is

$$N = NMAX \cdot b(y) \cdot a(x) \quad (3.15)$$

where:

$$\blacktriangleright b(y) = \begin{cases} e^{\left[-\left(\frac{y-YMIN}{YCHAR}\right)^2\right]} & \text{if } y < YMIN \\ 1 & \text{if } YMIN \leq y \leq YMAX \\ e^{\left[-\left(\frac{y-YMAX}{YCHAR}\right)^2\right]} & \text{if } y > YMAX \end{cases}$$

$$\blacktriangleright a(x) = \begin{cases} a_1(x) & \text{XERFC is FALSE} \\ a_2(x) & \text{XERFC is TRUE} \end{cases}$$

where:

$$\blacktriangleright a_1(x) = \begin{cases} e^{\left[-\left(\frac{x-XMIN}{XCHAR}\right)^2\right]} & \text{if } x < XMIN \\ 1 & \text{if } XMIN \leq x \leq XMAX \\ e^{\left[-\left(\frac{x-XMAX}{XCHAR}\right)^2\right]} & \text{if } x > XMAX \end{cases}$$

$$\blacktriangleright a_2(x) = \frac{\operatorname{erfc}\left(\frac{x-XMAX}{XCHAR}\right) - \operatorname{erfc}\left(\frac{x-XMIN}{XCHAR}\right)}{2}$$

If **XERFC** is not specified then it's default value is FALSE.

Specification for polygon initial species density

Initial species density specification for a species whose profile is a polygon over a specified region. Outside the polygon region and inside the bounding box, the profile falls off as a gaussian.

METHOD FUNCTION = <n>

```
KEY = INITIAL SPECIES DENSITY
MODEL NAME = FUNCTION

FUNCTION_1 = POLYGON
NMAX_1 = <real value>
TYPE_1 = <species_name_1>
XMIN_1 = <real value>
XMAX_1 = <real value>
YMIN_1 = <real value>
YMAX_1 = <real value>

XPOLYMIN_1 = <real value>
XPOLYMAX_1 = <real value>
YPOLYMIN_1 = <real value>
YPOLYMAX_1 = <real value>
RCHAR_1 = <real value>
END
```

where:

- ▶ **FUNCTION_1 = POLYGON** - specifies that the species has a uniform variation inside the inner polygon region and falls off as a gaussian outside the inner polygon and inside the bounding box. The "1" indicates that this is function number one. You can have any number of functions, which will be summed together to form the overall species profile, but the function numbers must start at "1" and be sequential up to the number of functions.
- ▶ **NMAX_1** - The value of the initial species density within the defined region. Note that "MAX" is used for consistency with other functions.
- ▶ **TYPE_1** - The species name goes here.
- ▶ **XPOLYMIN_1** - The value of the X dimension where the uniform region starts.
- ▶ **XPOLYMAX_1** - The value of the X dimension where the uniform region ends.
- ▶ **YPOLYMIN_1** - The value of the Y dimension where the uniform region starts.
- ▶ **YPOLYMAX_1** - The value of the Y dimension where the uniform region ends.
- ▶ **XMIN_1** - The value of the X dimension where the gaussian extension starts.

- ▶ XMAX_1 - The value of the X dimension where the gaussian extension ends.
- ▶ YMIN_1 - The value of the Y dimension where the gaussian extension starts.
- ▶ YMAX_1 - The value of the Y dimension where the gaussian extension ends.
- ▶ RCHAR_1 = The characteristic length of the gaussian profile outside of the polygon region and inside the bounding box.

Specification for yjunction initial species density

This section describes the initial species density specification for a profile that matches the background doping at a specified depth.

```
METHOD FUNCTION = <n>
  KEY = INITIAL SPECIES DENSITY
  MODEL NAME = FUNCTION

  FUNCTION_1 = Y_JUNCTION
  NMAX_1 = <real value>
  TYPE_1 = <species_name_1>
  XMIN_1 = <real value>
  XMAX_1 = <real value>
  YMIN_1 = <real value>
  YMAX_1 = <real value>
  [XCHAR_1 = <real value> | XYRATIO_1 = <real value>]
  BACKDOPE_1 = <real value>
  YJUNC_1 = <real value>
  [XERFC_1 = <TRUE|FALSE>]
END
```

where:

- ▶ FUNCTION_1 = Y_JUNCTION - specifies that the species density has a profile that is uniform within a specified region and falls off as a gaussian outside of that region and matches the magnitude of the value of background species density at a specified depth within the device. The "1" indicates that this is function number one. You can have any number of functions, which will be summed together to form the overall doping profile, but the function numbers must start at "1" and be sequential up to the number of functions.

- ▶ NMAX_1 - The value of the initial species density within the defined region. Note that "MAX" is used for consistency with other functions.
- ▶ TYPE_1 - The initial species density name goes here.
- ▶ XMIN_1 - The value of the X dimension where the uniform profile starts.
- ▶ XMAX_1 - The value of the X dimension where the uniform profile ends.
- ▶ YMIN_1 - The value of the Y dimension where the uniform profile starts.
- ▶ YMAX_1 - The value of the Y dimension where the uniform profile ends.
- ▶ XCHAR_1 - The characteristic length of the gaussian profile outside of the uniform region.
- ▶ XYRATIO_1 - The ratio of the horizontal characteristic length to the vertical characteristic length.
- ▶ BACKDOPE_1 - The background initial species density, which is used, together with the junction depth, in calculating the characteristic length in the y -direction.
- ▶ YJUNC_1 - The y -location under the center of the profile where the magnitude of the profile being added equals the magnitude of the background profile.
- ▶ XERFC_1 - Specifies that the x -variation of the profile is described by a complementary error function.

Mole Fraction

This is the specification for MOLE FRACTION when there is some form of functional variation. If a grading on the mole fraction is required, a "METHOD FUNCTION" within the MODEL section should be used, and the following information is pertinent

```

METHOD FUNCTION <n>
  KEY = MOLE FRACTION
  MODEL NAME = LINEAR
  X_MOLE = <real value>
  X_END = <real value> | X_SLOPE = <real value>
  [GRADING = X_LINEAR
  X_MIN = <real value>
  X_MAX = <real value>]
```

```
[GRADING = Y_LINEAR  
Y_MIN = <real value>  
Y_MAX = <real value>]  
END
```

where:

- ▶ **X_MOLE** - The mole fraction to use in the region for compound materials, which must be a value between 0.0 and 1.0.
- ▶ **X_END** - The mole fraction for graded compounds at the right or at the top, depending upon whether the linear grading is in the x-direction or the y-direction.
- ▶ **X_SLOPE** - The slope of the mole fraction for graded compounds. If this parameter is used, the mole fraction has a value of **X_MOLE** at the bottom or left of the region and a value of **X_MOLE + width*X_SLOPE** at the right or top of the region.
- ▶ **GRADING** - Set to **X_LINEAR** if the grading is in the x-direction and set to **Y_LINEAR** if the grading is in the y-direction.
- ▶ **X_MIN** - left boundary of the graded region for grading in x-direction.
- ▶ **X_MAX** - right boundary of the graded region for grading in x-direction.
- ▶ **Y_MIN** - bottom boundary of the graded region for grading in y-direction.
- ▶ **Y_MAX** - top boundary of the graded region for grading in y-direction.

TRANSIENT SOURCE

NOTE: The defect species must be defined within the physics specification via the region specification.

Transient sources are specified within a particular physics region and also as a material property so that the defect properties may be set for a given material and/or model. A transient source term may be set for a given species with this specification. Different models within a method function are used to set the time dependent source rate as described below.

```
METHOD FUNCTION <n>
```

```

KEY = TRANSIENT SOURCE
MODEL NAME = STEP
START_TIME_<species_name_1> = <real_value>
END_TIME_<species_name_1> = <real_value>
STEP_VALUE_<species_name_1> = <real_value>
.
.
START_TIME_<species_name_n> = <real_value>
END_TIME_<species_name_n> = <real_value>
STEP_VALUE_<species_name_n> = <real_value>
END

```

where:

- ▶ MODEL NAME
- ▶ STEP - source rate of value STEP_VALUE over a step function starting at time = START_TIME and ending at time = END_TIME
- ▶ REAL parameters for defect_species_name, electrons (E), or holes (H) = i
- ▶ STEP_VALUE $_i$ = source rate value of step function for species i , where > 0 will result in a source and < 0 will result in a sink $[\#/cm^3/s]$
- ▶ START_TIME $_i$ = start time of step function for species i [s]
- ▶ END_TIME $_i$ = end time of step function for species i [s]

or

```

METHOD FUNCTION <n>
KEY = TRANSIENT SOURCE
MODEL NAME = GAUSSIAN
START_TIME_<species_name_1> = <real_value>
END_TIME_<species_name_1> = <real_value>
MAX_VALUE_<species_name_1> = <real_value>
.
.
START_TIME_<species_name_n> = <real_value>
END_TIME_<species_name_n> = <real_value>

```



```
MAX_VALUE_<species_name_n> = <real_value>
END
```

where:

- ▶ MODEL NAME
- ▶ GAUSSIAN - source rate of value MAX_VALUE at time = (START_TIME+END_TIME)/2.0
- ▶ REAL parameters for defect_species_name, electrons (E), or holes (H)= i
- ▶ MAX_VALUE_ i = source rate peak value of gaussian function for species i , where > 0 will result in a source and < 0 will result in a sink $[\#/cm^3/s]$
- ▶ START_TIME_ i = start time of gaussian function for species i [s]
- ▶ END_TIME_ i = end time of gaussian function for species i [s]

or

```
METHOD FUNCTION <n>
  KEY = TRANSIENT SOURCE
  MODEL NAME = FILE
  FIT METHOD = <INTERPOLATE|CUBIC SPLINE>
  DOPE TO MESH = ONEtoTWO
  FILE_NAME_<species_name_1> = "<string_file_name>"
  .
  .
  FILE_NAME_<species_name_n> = "<string_file_name>"
END
```

where:

- ▶ MODEL NAME
- ▶ FILE - A file is read in for each species specified (via <species_name> FILE NAME). The file format is ASCII with tab-separated columns. If a species file is not specified the default initial distribution is 0.0 over all space.
- ▶ MODEL SPECIFICATIONS

- ▷ DOPE TO MESH - is the method used to place the doping information onto a **Charon** FEM mesh.
- ▷ ONETOTWO - take 1-dimensional source data in the file(s) and place it onto the 2-dimensional **Charon** mesh.
- ▷ FIT METHOD - The method used to fit the doping data to the **Charon** FEM mesh.
 - * INTERPOLATE - Interpolate the data. Finds the points within the doping data within which a point lies and interpolates between them to find the value. This is only relevant when using "DOPE TO MESH = ONETOTWO"
 - * CUBIC SPLINE - Uses a cubic spline interpolation. This is only relevant when using "DOPE TO MESH = ONETOTWO"
- ▷ File specification. A file for a given species is specified via "<species_name> FILE NAME = <file_name>" The file format is ASCII with tab-separated columns. For single file reads the data should be in the format "x density" for 1D species density.

or

```

METHOD FUNCTION <n>
  KEY = TRANSIENT SOURCE
  MODEL NAME = ONEDIM PROFILE
  FIT METHOD = <INTERPOLATE|CUBIC SPLINE>
  FILE_NAME_<species_name_1> = "<string_file_name>"
  TIME_FILE_NAME_<species_name_1> = "<string_file_name>"
  START_TIME_<species_name_1> = <real value>
  END_TIME_<species_name_1> = <real value>
  NMAX_<species_name_1> = <real value>
  [XMIN_<species_name_1> = <real value>]
  [XMAX_<species_name_1> = <real value>]
  [XCHAR_<species_name_1> = <real value>]
  [XERFC_<species_name_1> = <real value>]
  [TCHAR_<species_name_1> = <real value>]
  [TERFC_<species_name_1> = <TRUE|FALSE>]
  [YMIN_<species_name_1> = <real value>]
  [YMAX_<species_name_1> = <real value>]
  [REVERSEY_<species_name_1> = <TRUE|FALSE>]
  .
  .

```

```

FILE_NAME_<species_name_n> = "<string_file_name>"
TIME_FILE_NAME_<species_name_n> = "<string_file_name>"
START_TIME_<species_name_n> = <real value>
END_TIME_<species_name_n> = <real value>
NMAX_<species_name_n> = <real value>
[XMIN_<species_name_n> = <real value>]
[XMAX_<species_name_n> = <real value>]
[XCHAR_<species_name_n> = <real value>]
[XERFC_<species_name_n> = <real value>]
[TCHAR_<species_name_n> = <real value>]
[TERFC_<species_name_n> = <TRUE|FALSE>]
[YMIN_<species_name_n> = <real value>]
[YMAX_<species_name_n> = <real value>]
[REVERSEY_<species_name_n> = <TRUE|FALSE>]
END

```

where:

- ▶ **MODEL NAME**
- ▶ **FIT METHOD** - The method used to fit the doping data to the **Charon** FEM mesh.
 - ▷ **INTERPOLATE** - Interpolate the data. Finds the points within the doping data within which a point lies and interpolates between them to find the value.
 - ▷ **CUBIC SPLINE** - Uses a cubic spline interpolation.
- ▶ **START_TIME_{*i*}** = start time of gaussian function for species *i* [s]
- ▶ **END_TIME_{*i*}** = end time of gaussian function for species *i* [s]
- ▶ **XMIN_{*i*}** - The value of the *X* dimension where the uniform region starts.
- ▶ **XMAX_{*i*}** - The value of the *X* dimension where the uniform region ends.
- ▶ **XCHAR_{*i*}** - The characteristic length of the gaussian profile outside of the uniform region in the *x*-direction.
- ▶ **XERFC_{*i*}** - Specifies that the *x* variation of the profile is described by a complementary error function.
- ▶ **TCHAR_{*i*}** - The characteristic length in time of the gaussian profile outside of the uniform region.

- ▶ TERFC_ - Specifies that the time variation of the profile is described by a complementary error function.
- ▶ YMIN_ - The user can optionally specify the minimum value in y of the input profile. If this value is specified, the profile drops to zero below this y-value. If not specified by the user, the rate will be set equal to the rate at the top and bottom points of the input profile.
- ▶ YMAX_ - The user can optionally specify the maximum value in y of the input profile. If this value is specified, the profile drops to zero above this y-value. If not specified by the user, the rate will be set equal to the rate at the top and bottom points of the input profile.
- ▶ REVERSEY_ - Most other TCAD codes define the upper left corner of the device as 0,0 with y increasing as you go down. **Charon** uses a standard coordinate scheme and so if the doping data is from another TCAD that uses the upper left corner as the origin this option will map it correctly to **Charon**'s coordinate system.

Constant Electric Field BC

To specify a constant E-Field BC requires two entries in the input command file. Both of these entries are in the general section of the input file, NOT the charon-specific section. This boundary condition applies to the finite element formulation.

The first is the specification of the Neumann BC:

```
BC, NEUMANN, ELECTRIC_POTENTIAL, SIDESSET <m>, ELEMENT BLOCK <n>, \
METHOD FUNCTION <p>
```

where:

- m - The integer sideset ID. - n - The integer element block ID. - p - The integer method function ID.

The next entry is the specification of the method function:

```
METHOD FUNCTION <p>
KEY = SC_CONSTANT_EFIELD_BC
EFIELD_VALUE = 200.0
```

```
RELATIVE_PERMITTIVITY = 1.0
END
```

Silicon/Oxide Interface BC

This is an interfacial boundary condition that accounts for the difference in permittivity between two materials. An optional interface charge may be used. This boundary condition is to be used with the finite element formulation.

To specify a silicon/oxide interface boundary condition requires two entries for EACH element block at the interface. These entries are in the general section of the input file, NOT the charon-specific section.

The following example is for an interface between element blocks 1 and 2. First is the specification of the Neumann BCs:

```
BC, NEUMANN, ELECTRIC_POTENTIAL, SIDASET 2, ELEMENT BLOCK 1, \
METHOD FUNCTION 10

BC, NEUMANN, ELECTRIC_POTENTIAL, SIDASET 2, ELEMENT BLOCK 2, \
METHOD FUNCTION 11
```

Next is the specification of the method functions:

```
METHOD FUNCTION 10
  KEY = SILICON_OXIDE_INTERFACE
  INTERFACE_CHARGE = -3.0e10
END

METHOD FUNCTION 11
  KEY = SILICON_OXIDE_INTERFACE
  INTERFACE_CHARGE = -3.0e10
END
```

where:

- INTERFACE_CHARGE has units of $\# / \text{cm}^2$

FEM Oxide Electric Field BC

This boundary condition is for the Finite Element formulation, and it approximates an oxide layer without explicitly modeling it. The bc puts an electric field on the surface that is equal to the potential difference across the oxide divided by its thickness. An optional charge at the surface may be applied as well.

To specify the FEM Oxide Electric Field BC requires two entries in the input command file. Both of these entries are in the general section of the input file, NOT the charon-specific section.

The first is the specification of the Neumann BC:

```
BC, NEUMANN, ELECTRIC_POTENTIAL, SIDESSET <m>, ELEMENT BLOCK <n>, \
METHOD FUNCTION <p>
```

where:

- m - The integer sideset ID. - n - The integer element block ID. - p - The integer method function ID.

The next entry is the specification of the method function:

```
METHOD FUNCTION <p>
  KEY = EFIELDBC_OXIDE_FEM
  SURFACE_CHARGE = <real value>
  GATE_VOLTAGE = <real value>
  OXIDE_THICKNESS = <real value>
  OXIDE_PERMITTIVITY = <real value>
END
```

where:

- ▶ SURFACE_CHARGE has units of $\#/\text{cm}^2$
- ▶ GATE_VOLTAGE has units of volts
- ▶ OXIDE_THICKNESS has units of cm
- ▶ OXIDE_PERMITTIVITY is the relative permittivity (=3.9 for SiO_2)

SGFVM Oxide Electric Field BC

This boundary condition is for the Scharfetter-Gummel formulation, and it approximates an oxide layer without explicitly modeling it. The bc puts an electric field on the surface that is equal to the potential difference across the oxide divided by its thickness.

To specify the SGFVM Oxide Electric Field BC requires two entries in the input command file. Both of these entries are in the general section of the input file, NOT the charon-specific section.

The first is the specification of the Neumann BC:

```
BC, NEUMANN, ELECTRIC_POTENTIAL, SIDESSET <m>, ELEMENT BLOCK <n>, \
METHOD FUNCTION <p>
```

where:

- ▶ m - The integer sideset ID.
- ▶ n - The integer element block ID.
- ▶ p - The integer method function ID.

The next entry is the specification of the method function:

```
METHOD FUNCTION <p>
  KEY = EFIELDBC_OXIDE_SGFVM
  SURFACE_CHARGE = <real value>
  GATE_VOLTAGE = <real value>
  OXIDE_THICKNESS = <real value>
  OXIDE_PERMITTIVITY = <real value>
END
```

where:

- ▶ SURFACE_CHARGE has units of $\# / \text{cm}^2$
- ▶ GATE_VOLTAGE has units of volts

- ▶ OXIDE_THICKNESS has units of cm
- ▶ OXIDE_PERMITTIVITY is the relative permittivity (=3.9 for SiO_2)

SRH Surface BC

This boundary condition specifies a carrier density based upon an SRH calculation at the boundary. This is to represent the surface effects on the carriers due to an oxide interface at the surface of interest. This is basically the recombination term calculated by the TRAP model multiplied by a user supplied oxide width. The boundary condition is set as a Neumann BC as shown below:

The first is the specification of the Neumann BC:

```
BC, NEUMANN, <ELECTRON_DENSITY | HOLE_DENSITY> , SIDESSET <m>, ELEMENT BLOCK <n>,\
METHOD FUNCTION <p>
```

where:

- m - The integer sideset ID. - n - The integer element block ID. - p - The integer method function ID.

The next entry is the specification of the method function:

```
METHOD FUNCTION <p>
  KEY = SC_SRH_SURFACE_BC
  OXIDE_WIDTH = <real_value>
  MODEL NAME = <NO TRAPS | TRAPS>

  < appropriate_trap_model_specification >

  END
```

where:

- ▶ OXIDE_WIDTH - the width of the oxide interface (cm)

► MODEL NAME

- ▷ NO_TRAPS - no traps are set (the default value)
- ▷ TRAPS - setting traps via the equations above
- ▷ MGAUSS - setting traps via the equations above and falls off as a gaussian distribution outside a specified range - appropriate_trap_model_specification - set the appropriate specifications for the given trap model.

INTRINSIC CONCENTRATION

This is the specification for INTRINSIC CONCENTRATION when it has some form of functional variation. At present the only model implemented is based on the work of Slotboom [7],[6].

The functional variation of the intrinsic concentration due to bandgap narrowing is given by

$$n_{ie}(T) = n_i(T) e^{(\Delta E_g)}, \quad (3.16)$$

where:

$$\Delta E_g = \frac{V0_BGN \cdot q}{2kT} \left[\ln \left(\frac{N_{total}(x, y)}{N0_BGN} \right) + \sqrt{\ln \left(\frac{N_{total}(x, y)}{N0_BGN} \right)^2 + CON_BGN} \right]. \quad (3.17)$$

and you can modify the parameters V0_BGN, N0_BGN and CON_BGN via the MPARAM statement.

```
METHOD FUNCTION 1
  KEY = INTRINSIC CONCENTRATION
  MODEL NAME = MEDICI
END
```

where:

- The Medici model for intrinsic concentration is given by:

$$n_i(T) = \sqrt{N_c N_v} e^{\left(-\frac{E_g}{2kT}\right)} \quad (3.18)$$

Constant Current BC

To specify a constant current BC first requires specification of the objective function in the general section of the input file:

```
OBJECTIVE FUNCTION
  NAME = CURRENT
  CONTACTS = <n>
  TARGETS = <real value>
  INITIAL SCALING = <real value>
END
```

along with a statement for sensitivities:

```
SENSITIVITIES
  PARAM = "CONTACT VOLTAGE <m>"
END
```

and a statement for loca solution scaling:

```
LOCA SOLUTION SCALING
  TYPE = INITIAL AVERAGES
END
```

Additionally, statements are required within the nox parameters section of the input file in order to enable the objective function:

```
PRINT SOLUTION STATISTICS = TRUE
ENABLE OBJECTIVE FUNCTION CONSTRAINT = TRUE
LOCA RESIDUAL SCALING = ROW SUM
```

as well as a sublist for the Constraint parameters:

```
SUBLIST "Constraints"
  "Constraint Parameter <m>" = "CONTACT VOLTAGE <m>"
  "Bordered Solver Method" = "Householder"
END
```

The constant current BC is used together with a dirichlet boundary condition on the electric potential for the same contact as the contact to which the objective function is applied:

```
DIRICHLET BC, ELECTRIC_POTENTIAL, SIDESSET <p> <real value>
```

In order to run the constant current boundary condition using the default **Charon** build (without revad enabled), the following statement must be included in the Physics setup section of the input file:

```
USE FAST CURRENT CALCULATION = TRUE
```

A number of examples which use the constant current boundary condition can be found in /Charon_data/Benchmarks/Regression/Semiconductor/2D/current_constraint.

Contact Series Resistance

To specify a contact series resistance BC first requires specification of the objective function in the general section of the input file:

```
OBJECTIVE FUNCTION
NAME = RESISTOR
CONTACTS = <n>
VOLTAGE = <real value>
RESISTANCE = <real value>
INITIAL SCALING = <real value>
END
```

where the voltage parameter and the resistance parameter correspond to $V_{applied}$ and R , respectively, in the expression $\frac{V_{applied}-\phi}{R} - currents = 0$. Loca solves for the contact voltage ϕ in this expression.

along with a statement for sensitivities:

```
SENSITIVITIES
PARAM = "CONTACT VOLTAGE <m>"
END
```

and a statement for loca solution scaling:

```
LOCA SOLUTION SCALING
  TYPE = INITIAL AVERAGES
END
```

Additionally, statements are required within the nox parameters section of the input file in order to enable the objective function:

```
PRINT SOLUTION STATISTICS = TRUE
ENABLE OBJECTIVE FUNCTION CONSTRAINT = TRUE
LOCA RESIDUAL SCALING = ROW SUM
```

as well as a sublist for the Constraint parameters:

```
SUBLIST "Constraints"
  "Constraint Parameter <m>" = "CONTACT VOLTAGE <m>"
  "Bordered Solver Method" = "Householder"
END
```

The series contact resistance BC is used together with a dirichlet boundary condition on the electric potential for the same contact as the contact to which the objective function is applied:

```
DIRICHLET BC, ELECTRIC_POTENTIAL, SIDESSET <p> <real value>
```

In order to run the series contact resistance boundary condition using the default **Charon** build (without revad enabled), the following statement must be included in the Physics setup section of the input file:

```
USE FAST CURRENT CALCULATION = TRUE
```

An example which uses the contact series resistance boundary condition can be found in `~/Charon_data/Benchmarks/Regression/Semiconductor/2D/current_constraint/lumped_bc.`

INCOMPLETE IONIZATION

The expression for incomplete ionization for donor impurities is given by

$$N_D^+ = \frac{N_D}{1 + \mathbf{GB_N} \exp[(E_{Fn} - E_C + \Delta E_D)/(kT)]}. \quad (3.19)$$

The expression for incomplete ionization for acceptor impurities is given by

$$N_A^+ = \frac{N_A}{1 + \mathbf{GB_P} \exp[(E_V - E_{Fp} + \Delta E_A)/(kT)]}. \quad (3.20)$$

The quantities ΔE_D and ΔE_A are the donor and acceptor activation energies where:

$$\Delta E_D = \mathbf{EB0_N} - \mathbf{ALPHA_N} N_D^{1/3} + \mathbf{BETA_N} (T^{-\mathbf{GAMMA_N}} - 300^{-\mathbf{GAMMA_N}}) \frac{kT}{q} \quad (3.21)$$

and

$$\Delta E_A = \mathbf{EB0_P} - \mathbf{ALPHA_P} N_A^{1/3} + \mathbf{BETA_P} (T^{-\mathbf{GAMMA_P}} - 300^{-\mathbf{GAMMA_P}}) \frac{kT}{q}. \quad (3.22)$$

One can modify the parameters $\mathbf{GB_N}$, $\mathbf{GB_P}$, $\mathbf{EB0_N}$, $\mathbf{EB0_P}$, $\mathbf{ALPHA_N}$, $\mathbf{ALPHA_P}$, $\mathbf{BETA_N}$, $\mathbf{BETA_P}$, $\mathbf{GAMMA_N}$, $\mathbf{GAMMA_P}$, $\mathbf{HDT_MIN}$ and $\mathbf{HDT_MAX}$ via the MPARAM statement (see section 3.2).

3.6 Nevada Input Description

Title card:

```
TITLE
Any single line of text
```

Nevada debugging commands

These commands are used to turn on various levels of screen output to show progression of various stages of simulation:

```
DEBUG MODE, PARSER
DEBUG MODE, LOCATION
DEBUG MODE, TIMESTEP
```

Nevada application-specific section

. For **Charon** this should be the the single word **CHARON** with a corresponding **END**

```
CHARON
.
  <charon-specific cards>
.
END
```

3.7 Charon subsection of input file

All documentation within this section is specific to the **CHARON** subsection of the input file and should be contained within that subsection, that is between *CHARON* and it's corresponding *END* (See above).

DEBUG Description

Default debug level

Many of the member functions within **Charon** are written so that if the code is compiled with `-DCHARONDEBUG` various information is output to the screen. The amount of data that is output is generally tied to the *DEBUG LEVEL*. Generally this ranges from 0 (no debug output) to 10 (all debug output). (See *DEBUG METHOD* for more information.)

```
DEBUG LEVEL=<i> (i = 0 - 9)
```

Charon specific method debugging

This card turns on debugging for a specific method within **Charon**. While the default debug level is set using *DEBUG LEVEL* a *DEBUG METHOD* card allows finer control of what

method, or member function, outputs it's debugging information, and for which processor the output should be performed. Setting the processor ID flag to -1 will cause all processors to output that information. This overrides *DEBUG LEVEL* for the named method. Generally you have to know the name assigned to a particular method in order to utilize this feature. The *method name* is not validated and so any value is allowed there. *method name* is case sensitive.

```
DEBUG METHOD, <method name> <i> <j> (i - debug level, j - processor id)
```

Debugging parameters

This is generally only useful to developers. If, as a developer, you have something you want to temporarily control from the input file, and something which should otherwise not be controllable from the input file, you can use this card. Information here is simply inserted into a Hash_Map within the Charon_Debug object which is a singleton and can be accessed from anywhere within the code. Currently supported value types are real, integer and strings.

```
DEBUG PARAMETER, <parameter name> = <value>
```

STABILIZATION, DISCONTINUITY CAPTURE, and LUMPING Description

Stabilization Options

```
STABILIZATION OPTIONS
  SUPG = <TRUE|FALSE>
  SOURCE = <TRUE|FALSE>
  DC = <TRUE|FALSE>
  SGS = <TRUE|FALSE>
  VMS = <TRUE|FALSE>
END
```

Discontinuity Capturing

```
DISCONTINUITY CAPTURING = <TRUE|FALSE>
```

```

DC ALGORITHM = <SHAKIB1|SHADID1|CODINA1>
[DC PARAMETER, <GAMMA|RELATIVE TOLERANCE|SCALING> = <real value>]
[DC PARAMETER, <COEFFICIENT VARIATION> = <LINEAR|QUADRATIC>]
[DC PARAMETER, <CODINA ZERO TOLERANCE> = <real value>]
[DC PARAMETER, <CHI> = <real value>]

```

Algorithms are defined as:

- ▶ SHAKIB1 - $\frac{|\tilde{R}_v|}{\sqrt{\nabla v G_c \nabla v + \epsilon_r ||v||}}$
- ▶ SHADID1 - $K(\tau \tilde{R}_v)^\gamma \left[\frac{||\frac{1}{v} G_c \nabla v||}{\frac{1}{2} \sqrt{||G_c||}} \right]^{1-\gamma}$
- ▶ CODINA1 - $\int \nu_{\text{codina}} \nabla \hat{v} \cdot \left(\mathbf{I} - \frac{\mathbf{a}_v \otimes \mathbf{a}_v}{||\mathbf{a}_v||^2} \right) \cdot \nabla v \, d\Omega$

where:

- ▶ v - the DOF
- ▶ τ - stabilization parameter from SUPG
- ▶ $\tilde{R}_v$ - the approximate residual for variable v
- ▶ G_c - the element covariant tensor
- ▶ ϵ - the RELATIVE TOLERANCE from DC PARAMETER input (Default: 1.0e-6)
- ▶ γ - the GAMMA from DC PARAMETER input (Default: 0.66667)
- ▶ K - the SCALING from DC PARAMETER input (Default: 1.0)
- ▶ COEFFICIENT VARIATION - Determines how the DC coefficient behaves. (Default: LINEAR)
- ▶ $\nu_{\text{codina}} = \begin{cases} \frac{\alpha h |\tilde{R}_v|}{2 ||\nabla v||} & \text{if } ||\nabla v|| \neq 0 \\ 0 & \text{otherwise} \end{cases}$
- ▶ $\alpha = \max \left(0, \chi - \frac{2v}{h ||\mathbf{a}_v||} \right)$
- ▶ $h \equiv ||G_c||$

- a_v - the velocity associated with DOF v .

Notes:

- For CODINA1 the algorithm can be sensitive to what is considered zero for the $||\nabla v||$. By default this is set to 1.0e-2. This can be adjusted via *DC PARAMETER, CODINA ZERO TOLERANCE*.
- For CODINA1 χ has a default value of 0.7. This can be adjusted via *DC PARAMETER, CHI* and should be between 0.0 and 1.0.

Lumping

Currently supported term names are **SOURCE** and **MASS**

```
LUMP, <term name> = <TRUE|FALSE>
```

OPERATOR SCALING Description

Operator scaling Used to scale named operators by a real, scalar value. The names are physics-dependent and there is no error checking on that name. Read the documentation for the specific physics to get a list of those names. As an example you can change the sign on a term by setting it's scale factor to -1.0.

```
OPERATOR SCALING, <term name> = <real value>
```

JACOBIAN and FILL STATUS Description

Jacobian calculation

Here you specify the method with which to compute the Jacobian. Currently supported values, in order of preference, are:

- AD - Automatic Differentiation (fastest)

- ▶ JFNK with AD Prec - Jacobian-Free Newton-Krylov with AD Jacobian for the Preconditioner
- ▶ FDC - Finite Difference with Coloring
- ▶ FD - Finite Difference
- ▶ NONE - Matrix Free

JACOBIAN = <FD|AD|JFNK WITH AD PREC|FDC|NONE>

Fill Status

If the physics supports the capability this card can be used to toggle the reporting of various fill quantities on and off. Generally fill status includes things like Peclet and CFL numbers.

REPORT FILL STATUS = <TRUE|FALSE>

CODE TIMING Description

Code timings

This controls whether or not you wish to have code timings output to the screen.

OUTPUT TIMINGS = <TRUE|FALSE>

BOUNDARY CONDITIONS Description

Specification of **Dirichlet boundary conditions**

DIRICHLET BC, <variable name>, sideset <n> <constant>|<METHOD FUNCTION <m>>

where:

- ▶ `<variable name>` - The name of the variable to which this BC should be applied
- ▶ `<n>` - The Exodus sideset ID
- ▶ `<constant>` - A constant value
- ▶ `<m>` - The ID of a METHOD FUNCTION describing a possibly time-varying Dirichlet BC. Currently the supported METHOD FUNCTIONS are LINEAR_TIME_VBC and SINUSOID_TIME_VBC.

Specification **Linear-time variable BC** or **Sinusoidal-time variable BC** for Dirichlet BC specification

NOTE: this specification is only valid for `<variable_name> = electric_potential`

```
METHOD FUNCTION <m>
  key = <linear_time_vbc | sinusoid_time_vbc >

  start time = <constant>
  start value = <constant>

  final time = <constant>
  final value = <constant>
END
```

Specification for **Transient Doping Boundary Condition** for Dirichlet BC specification

NOTE: this specification is valid for `<variable_name> = electric_potential — electron_density — hole_density`

Note this BC is primarily used when the dopants are treated as a species and may vary with time due to reactions with other species. This uses the ohmic condition (quasi-neutrality at the boundaries) to calculate the potential, electron density, or hole density at the boundary. It is highly recommended if one variable is used with this BC that all 3 variables are set with the same BC (note they all may reference the same method function described below).

```
METHOD FUNCTION <m>
```

```
key = <transient_doping_vbc>

start time = <constant>
start value = <constant>

final time = <constant>
final value = <constant>
END
```

where:

- ▶ start time = start time of linear voltage ramp
- ▶ start value = starting value of linear voltage ramp
- ▶ final time = final time of linear voltage ramp
- ▶ final value = final value of linear voltage ramp

Specification for **Constant Electric Field Boundary Condition** for Neumann BC specification

NOTE: this specification is valid for <variable_name> = electric_potential

The Constant Electric Field Boundary Condition is specified within the following section:
3.5

Specification for **Silicon/Oxide Interface Boundary Condition** for Neumann BC specification

NOTE: this specification is valid for <variable_name> = electric_potential

The Silicon/Oxide Interface Boundary Condition is specified within the following section:
3.5

Specification for **FEM Oxide Electric Field Boundary Condition** for Neumann BC specification

NOTE: this specification is valid for <variable_name> = electric_potential

The FEM Oxide Electric Field Boundary Condition is specified within the following section: 3.5

Specification for **SGFVM Oxide Electric Field Boundary Condition** for Neumann BC specification

NOTE: this specification is valid for <variable_name> = electric_potential

The SGFVM Oxide Electric Field Boundary Condition is specified within the following section: 3.5

Specification for **SRH Surface Boundary Condition** for Neumann BC specification

NOTE: this specification is valid for <variable_name> = electron_density — hole_density

The SRH Surface Boundary Condition is specified within the following section: 3.5

Specification for **Constant Current Boundary Condition** to be used with Dirichlet BC specification

NOTE: this specification is valid for <variable_name> = electric_potential

The Constant Current Boundary Condition is specified within the following section: 3.5

Specification for **Contact Series Resistance** to be used with Dirichlet BC specification

NOTE: this specification is valid for <variable_name> = electric_potential

The Contact Series Resistance is specified within the following section: 3.5

VARIABLE MANAGER DATA OUTPUT

Variable Manager data output

This is used to output charon::VariableManager data to the exodus file.

```

VARIABLE MANAGER DATA OUTPUT
  [NUMBER OF QUAD POINTS = <n>]
  "<VM variable name 1>", <Q|N|QG|NG>, <output name 1> \
    [, UNSCALE="<scale factor name>"]
    .
    .
    .
  "<VM variable name n>", <Q|N|QG|NG|C>, <output name n> \
    [, UNSCALE="<scale factor name>"]
END

```

where:

- ▶ *NUMBER OF QUAD POINTS* = <n> - This sets the number of values to output if you use the "QE" variable type. The QE type is used to output all the individual values at the quad points. Unfortunately it is difficult to determine the number of quad points when data is first registered so this value must be set to the maximum for all the elements in the mesh. For example, for an all-quad mesh this would typically be set to 4. If you don't set it there is a default internal value of 4.
- ▶ *"VM variable name"* - The name of the variable as known to charon::VariableManager (must be enclosed by quotes and case/spacing IS significant)
- ▶ *VM variable type* - Must be one of:
 - ▷ N - nodal
 - ▷ Q - quadrature point
 - ▷ NG - nodal gradient
 - ▷ QG - quadrature point gradient
 - ▷ QE - output all individual quad-point values
 - ▷ C - a constant value
- ▶ *output name* - The name of the variable in the Exodus file. Note that names should be kept simple here as there are some severe limitations (that aren't checked for) on variable names in Exodus files. Each name must be unique (again, no error check performed)
- ▶ *UNSCALE=<name of scaling factor>* This is an optional argument that allows you to unscale the output variables if you know the name of the scaling factor as stored in

charon::ScalingParameters. Be sure to use quotes around the expression to prevent the parser from upper-casing it and do NOT use spaces. charon::ScalingParameters is case sensitive, so that $t0$ and $T0$ are different scale factors, for example. You can use simple expressions here as well. For example if the variable needs to be unscaled by "C0/X0" that is supported. You can use constants as well. Keep in mind that the expression is VERY limited. You can't use grouping and don't use spaces in the expression. About the most complex expression you could use would be something like: "1.25e10*C0/X0"

Limitations:

- ▶ Keep the Exodus variable names simple. Exodus has some pretty severe limitations on variable names so keep them case insensitive, short and no special characters.
- ▶ If you request a variable that isn't provided by anything you can expect an error from charon::VariableManager. An example of this would be requesting something at the nodes that is only evaluated at the quadrature points. For the most part variables are evaluated at the quadrature points and requesting the output of such a variable at the nodes here will not change that and the VM will output an error about nothing providing that variable.
- ▶ When you request the output of a quadrature point variable you'll get the average value of that variable for a given element, an element variable in Exodus parlance. This can loosely be considered the value of that variable at the center of the element. You can also output all the quad point values using the QE type, but it may be harder to interpret any visualization of such values.
- ▶ You have to know the *name* of the scaling factor in order to have unscaled output of variables. In general the physics should report these at the beginning of the run, but nothing enforces this rule and so you may have to look at the code to find these names.
- ▶ Don't forget to add the Exodus variable names to the *Nevada PLOT VARIABLE* section of the input file. This applies to everything except "constant" vm data. Such data is always output to the Exodus file as a global variable and thus shouldn't be added to the list of plot variables.

TIME INTEGRATION Description

Charon Time Integration

This is used to control the time integrator within **Charon**, Charon.TimeIntegrator.

```

TIME INTEGRATION
  ALGORITHM = GENERALIZED ALPHA
  TIME STEP CONTROL = <CONSTANT|VARIABLE>
  INITIAL TIME STEP SIZE = <real value>
  [CONSISTENT RESTART = <TRUE|FALSE>]
  [START TIME] = <real value|EXODUS INITIAL GUESS>
  END TIME = <real value>
  MAX NUMBER OF TIME STEPS = <int value>
  [CONSTANT TIME STEP COUNT = <integer value>]
  [TIME ERROR INDICATOR LIMIT = <real value>]
  [TIME STEP INCREASE FACTOR = <real value>]
  [TIME STEP DECREASE FACTOR = <real value>]
  [MAX NUMBER OF STEP FAILURES = <integer value>]
  [RELATIVE ERROR = <real value>]
  [ABSOLUTE ERROR = <real value>]
  [PSEUDO NEWTON STEP TARGET = <integer value>]
  [PARAMETER, <parameter name> = <real value>]
  [OUTPUT FREQUENCY TYPE = <STEP COUNT|TIME INTERVAL>]
  [OUTPUT STEP COUNT = <int value>]
  [OUTPUT TIME INTERVAL = <real value>]
  [BREAK POINTS = <real value>,<real value>, ... ,<real value> END>]
  [RESTART TIME INTEGRATOR AT BREAK POINTS = <TRUE|FALSE>]
  [MAX TIME STEP = <positive real value>]
END

```

where:

- The only supported algorithm is currently GENERALIZED ALPHA
- INITIAL TIME STEP SIZE in the case of *VARIABLE* time stepping controls the initial time step size, while for *CONSTANT* stepping it is the size of the time step for the duration of the simulation. Note that to preserve Δt^2 accuracy for constant time stepping the initial step is reduced and gradually increased until the requested step size is obtained.

- ▶ **CONSISTENT RESTART** allows the user to do a full restart. This means the state of the time integrator is stored in its entirety.
- ▶ **START TIME** is the time at which **Charon** should start the simulation (Default: 0.0).
- ▶ **END TIME** is the time at which **Charon** should terminate the simulation. There is no default, this is required.
- ▶ **MAX NUMBER OF TIME STEPS** is the maximum number of time steps after which charon will terminate the simulation. There is no default, this is required.
- ▶ **CONSTANT TIME STEP COUNT** governs how many constant time steps to take when using *VARIABLE* step control. An algorithm generally has a minimum that cannot be overridden, but you can override the maximum here.
- ▶ **TIME ERROR INDICATOR LIMIT** The limit at which a time step is considered to have failed as an L2 error norm between the calculated and predicted value of the solution (Default: 0.5).
- ▶ **TIME STEP INCREASE FACTOR** The maximum increase in time step size (Default: 1.5)
- ▶ **TIME STEP DECREASE FACTOR** the maximum amount that a time step can be decreased. This is only valid for *PSEUDO* time variation (Default: 0.5).
- ▶ **RELATIVE ERROR, ABSOLUTE ERROR** - Used in the calculation of the time step error. The formula is:
 - ▷
$$t_e = \frac{\sum_i (x_i - \text{predictor}_i)^2}{\epsilon_r \text{MAX}(|x_i|) + \epsilon_a}$$
 - ▷ Defaults, relative=1.0e-6, absolute=1.0e-12
- ▶ **PSEUDO NEWTON STEP TARGET COUNT** - The target number of Newton steps for pseudo-transient runs. If the count falls below this target the time stepper will increase the time step size and decrease the time step size if it is above this count. (Default: 5)
- ▶ **PARAMETER A** named general parameter passed directly into the time-integration algorithm. For generalized alpha the supported parameters are:
 - ▷ RHO_INF
 - ▷ ALPHA_F
 - ▷ ALPHA_M
 - ▷ GAMMA

- ▶ **OUTPUT FREQUENCY TYPE** defines how often time steps should be output to the exodus file. In the case of *STEP COUNT*, **Charon** will write to the exodus file every n time steps where n is specified with the *OUTPUT STEP COUNT* keyword. In the case of *TIME INTERVAL* **Charon** will write to the exodus file on the closest step to the specified interval. The time interval is specified with the *OUTPUT TIME INTERVAL* keyword. (Default: STEP COUNT)
- ▶ **OUTPUT STEP COUNT** tells charon how often to print the time steps - Every n steps. (Default: 1)
- ▶ **OUTPUT TIME INTERVAL** tells charon how often to print the time steps - Every x seconds. This does NOT force output on exact numbers. For example, if your interval is set to 0.5 and you last output to the exodus file at time 0.444, then the next output would occur for the first time step on or after 0.944. Using the "time interval" will limit the maximum time step size to the interval! (Default 0.1)
- ▶ **BREAK POINTS** is a list of real number time values that charon should solve for exactly. Users MUST put the "END" key after the list of break points so the parser knows the list is finished. If a break point is outside the valid range (between start and end times) it is just ignored.
- ▶ **RESTART TIME INTEGRATOR AT BREAK POINTS** If set to true, the time integrator will reset at the break point. It will throw away any stored history and start as if you ended the job and resubmitted a new one starting at this point. This is important if there is an occurrence at the break point such as switching on a transient source term. The history could provide a bad initial guess for the next step since new physics has been turned on. (Default: TRUE)
- ▶ **MAX TIME STEP** allows the user to set a maximum allowable time step size. The time integrator will cap its step size at this value if it is reached. (Default: REAL.MAX)

Note: If you specify **RHO_INF** then you cannot specify **ALPHA_F**, **ALPHA_M** or **GAMMA**, and vice versa.

SOLVER Description

Charon Solver

This is used to control the nonlinear and linear solvers within **Charon**. **Charon** uses the Trilinos solvers[8, 9]. The following is an example for the nonlinear and linear solver parameters section:

```
nox parameters

maximum nonlinear iterations = 15
use absolute tolerance test = false
use relative tolerance test = true
absolute tolerance = 1.0e-6
relative tolerance = 1.0e-3
wrms absolute tolerance = 1.0e-6
wrms relative tolerance = 1.0e-4
wrms tolerance = 1.0
wrms alpha = 1.0
wrms beta = 0.5
linear scaling = row sum

print solution statistics = true

sublist "Direction"
  "Method" = "Newton"
  sublist "Newton"
    "Rescue Bad Newton Solve" = "true"
    sublist "Linear Solver"
      "Aztec Solver" = "GMRES"
      "Max Iterations" = 500
      "Size of Krylov Subspace" = 500
      "Tolerance" = 1.0e-9
      "Output Frequency" = 100
      "Preconditioner" = "New Ifpack"
      "Preconditioner Operator" = "Use Jacobian"
      "Overlap" = 1
      sublist "Ifpack"
        "schwarz: reordering type" = "rcm"
        "fact: level-of-fill" = 2
      end
    end
  end
end
end
end
end
```

Charon parses the lines until the line `sublist "Direction"`, and the rest of the lines follow the Trilinos Teuchos parameter list format and are treated as such. For this example,

convergence of the nonlinear solver (Newton) will be determined by the relative tolerance test as well as meeting a certain criterion for the WRMS test. Row sum scaling will be used (in general semiconductor problems are more likely to converge with row sum scaling). The generalized minimal residual (GMRES) Krylov linear solver in Aztec will be used, with Krylov subspace size of 500 and the residual needs to meet a tolerance of 10^{-9} . The preconditioner is a Schwarz domain decomposition preconditioner with an incomplete lower/upper (ILU) factorization on each subdomain with level of fill of two, one level of overlap and RCM reordering. RCM reordering tends to be better for most test cases. The ILU preconditioner tends to be very robust, but may not be the best choice for larger problems on a large number of processors. For this case, a multigrid preconditioner, available through the ML library, may be a better choice.

If the size of a problem is fixed, but the number of processors is increased, more Aztec iterations will be needed per Newton step to meet the linear solver convergence criterion. The number of necessary linear iterations can be reduced by increasing the amount of ILU overlap. However this requires additional memory, and it is not recommended to increase the overlap beyond 2 or 3. The number of linear iterations can also be reduced by increasing the ILU level of fill, but this requires more memory and increases the cost to compute the factors. Increasing the level of fill beyond 2 or 3 is seldom worth the additional cost. If it is necessary to increase the size of the Krylov subspace, one should keep in mind that for a Krylov method such as GMRES, not only will more memory be required, but each successive iteration will take longer than the previous one. For some cases, it may be advantageous to use an iterative technique such as Transpose-free quasi-minimal residual (TFQMR) or Bi-conjugate gradient with stabilization (BiCGStab) instead. However, in our experience GMRES tends to be quite robust. Although the required linear solver tolerance can vary with the problem, one should be careful not to make it too loose as this could cause problems for the nonlinear solver (which will lead to an increase in nonlinear solver steps or even nonconvergence of the nonlinear solver). There are many other options for both the nonlinear and linear solvers and the reader is advised to consult the Trilinos documentation [9] for all the options.

One additional point that can severely impact the performance of **Charon** is that the number of processors needs to be chosen so that no swapping on compute nodes occurs. This is not an issue on platforms with compute nodes that do not have swap space such as Red Storm and TLCC, but it is important to platforms with compute nodes that have swap space such as Thunderbird and NWCC Spirit.

ANALYTIC SOLUTION Description

Analytic/Manufactured Solutions

These lines are relevant when running the code with a manufactured solution and must be within the **CHARON** section of the input file.

```
ANALYTIC COMPARISON = <TRUE|FALSE>
ANALYTIC CASE NAME = "<String name of case within Charon>"
```

where:

- ▶ If you specify *ANALYTIC COMPARISON = TRUE* then you must specify a non-empty name for *ANALYTIC CASE NAME*.
- ▶ The *ANALYTIC CASE NAME* is a unique string used internally by the code to switch between known manufactured solutions, thus this name must be used when writing the code associated with this particular manufactured solution.

3.8 Default MPARAM Values

3.9 Future Capabilities

Charon currently supports mainly diodes and bipolar junction transistors. Several capabilities have already been added and are under development in order to be able to support devices such as metal-oxide-semiconductor field-effect transistors (MOSFETs) and heterojunction bipolar transistors (HBTs). These devices require the use of more than one element block. **Charon** does have a capability to model a MOSFET using a boundary condition that approximates an oxide layer without explicitly modeling it. At the current time we have the capability to use different material models in different element blocks but the same physics (or same set of equations) in each element block. In the future, we expect to have the capability to use different sets of equations in different element blocks as well.

property	input file name	units	Si	GaAs	PolySi	Oxide	Germanium
relative permittivity	PERMITTI	none	11.8	13.1	11.8	3.9	11.8
electron lifetime	TAUN0	sec	1.0e-7	1.0e-9	1.0e-7	2.0e-5	1.0e-7
hole lifetime	TAUP0	sec	1.0e-7	1.0e-9	1.0e-7	2.0e-5	1.0e-7
electron SRH concentration	NSRHN	$\#/cm^3$	5.0e16	1.0	5.0e16	5.0e16	5.0e16
hole SRH concentration	NSRHP	$\#/cm^3$	5.0e16	1.0	5.0e16	5.0e16	5.0e16
lattice temperature	LAT_TEMP	K	300.15	300.15	300.15	300.15	300.15
electron relaxation time	ELE_TAUW	sec	2.0e-13	1.0e-12	2.0e-13	2.0e-13	2.0e-13
hole relaxation time	HOL_TAUW	sec	2.0e-13	1.0e-12	2.0e-13	2.0e-13	2.0e013
electron mobility	MUN0	$\frac{cm^2}{V \cdot s}$	1350.0	8500.0	1350.0	20.0	1350.0
hole mobility	MUP0	$\frac{cm^2}{V \cdot s}$	495.0	400.0	495.0	2.0e-5	495.0
Auger coefficient for electrons	AUGN	$\frac{cm^6}{s}$	2.8e-31	0.0	2.8e-31	2.8e-31	2.8e-31
Auger coefficient for holes	AUGP	$\frac{cm^6}{s}$	9.9e-32	0.0	9.9e-32	9.9e-32	9.9e-32
intrinsic concentration	NI	$\frac{\#}{cm^3}$	1.25e10	1.8e6	1.25e10	1.25e10	1.25e10
electron affinity	AFFINITY	V	4.17	4.07	4.17	0.97	4.0
concentration parameter in Slotboom BGN	N0_BGN	$\frac{1}{cm^3}$	1.0e17	1.0e17	1.0e17	1.0e17	1.0e17
voltage parameter in Slotboom BGN	V0_BGN	V	9.0e-3	0.0	9.0e-3	9.0e-3	9.0e-3
constant parameter in Slotboom BGN	CON_BGN	eV	0.5	0.0	0.5	0.5	0.5

Table 3.4. Table of Material Properties

property	input file name	units	Si	GaAs	Oxide
minimum electron mobility	MUN1_AR0	$\frac{cm^2}{V \cdot s}$	88.0	8500.0	10.0
maximum electron mobility	MUN2_AR0	$\frac{cm^2}{V \cdot s}$	1252.0	0.0	20.0
electron exponent parameter	AN_ARORA	none	0.88	1.0	0.88
reference impurity concentration	CN_ARORA	$\frac{\#}{cm^3}$	1.26e17	1.26 e17	1.26e17
exponent of normalized temperatures	EXN1_AR0	none	-0.57	-0.57	-0.57
exponent of normalized temperatures	EXN2_AR0	none	-2.33	0.0	-2.33
exponent of normalized temperatures	EXN3_AR0	none	2.4	0.0	2.4
exponent of normalized temperatures	EXN4_AR0	none	-0.146	0.0	-0.146
minimum hole mobility	MUP1_AR0	$\frac{cm^2}{V \cdot s}$	54.3	400.0	1.0e-5
maximum hole mobility	MUP2_AR0	$\frac{cm^2}{V \cdot s}$	407.0	0.0	2.0e-5
hole exponent parameter	AP_ARORA	none	0.88	1.0	0.88
reference impurity concentration	CP_ARORA	$\frac{\#}{cm^3}$	2.35e17	2.35e17	2.35e17
exponent of normalized temperatures	EXP1_AR0	none	-0.57	0.0	-0.57
exponent of normalized temperatures	EXP2_AR0	none	-2.23	0.0	-2.23
exponent of normalized temperatures	EXP3_AR0	none	2.4	0.0	2.4
exponent of normalized temperatures	EXP4_AR0	none	-0.146	0.0	-0.146

Table 3.5. Arora Mobility Model Parameters

property	input file name	units	Si	GaAs
minimum electron mobility	MUN_MIN	$\frac{cm^2}{V-s}$	55.24	0.0
maximum electron mobility	MUN_MAX	$\frac{cm^2}{V-s}$	1429.23	8.5e3
reference impurity concentration for electrons	NREFN	$\frac{\#}{cm^3}$	1.07e17	1.69e17
exponent of normalized temperature used in the numerator for electrons	NUN	none	-2.3	-1.0
exponent of normalized temperature used in the denominator for electrons	XIN	none	-3.8	0.0
exponent of the ratio of total impurity concentration to NREFN for electrons	ALPHAN	none	0.733	0.436
minimum hole mobility	MUP_MIN	$\frac{cm^2}{V-s}$	49.705	0.0
maximum hole mobility	MUP_MAX	$\frac{cm^2}{V-s}$	479.37	400.0
reference impurity concentration for holes	NREFP	$\frac{\#}{cm^3}$	1.61e17	2.75e17
exponent of normalized temperature used in the numerator for holes	NUP	none	-2.2	-2.1
exponent of normalized temperature used in the denominator for holes	XIP	none	-3.7	0.0
exponent of the ratio of total impurity concentration to NREFP for holes	ALPHAP	none	0.70	0.395

Table 3.6. Analytic Mobility Model Parameters

property	name	units	AlGaAs	InGaAs	AlInAs	GaAsP	InGaP	InAsP
linear term appearing in $\mu_n^{Min}(X)$	MIN_X1	none	-0.983	-22.3	-0.92	23.9	17.6	2.6
quadratic term appearing in $\mu_n^{Min}(X)$	MIN_X2	none	0.0	-0.186	0.0	0.0	0.0	-0.0045
linear term appearing in $\mu_n^{Max}(X)$	MAN_X1	none	-0.95	-1.76	-0.97	48.5	21.4	-0.931
quadratic term appearing in $\mu_n^{Max}(X)$	MAN_X2	none	0.0	1.12	0.0	0.0	0.0	0.116
reference impurity concentration	NREFN2	$\frac{\#}{cm^3}$	1.75e18	1.75e18	2.51e17	1.75e18	2.51e17	2.51e17
linear term appearing in $\mu_p^{Min}(X)$	MIP_X1	none	0.0	0.0	0.0	0.0	0.0	0.0
quadratic term appearing in $\mu_p^{Min}(X)$	MIP_X2	none	0.0	0.0	0.0	0.0	0.0	0.0
linear term appearing in $\mu_p^{Max}(X)$	MAP_X1	none	0.0	0.0	0.0	0.0	0.0	0.0
quadratic term appearing in $\mu_p^{Max}(X)$	MAP_X2	none	0.0	0.0	0.0	0.0	0.0	0.0
reference impurity concentration	NREFP2	$\frac{\#}{cm^3}$	1.0e30	1.0e30	1.0e30	1.0e30	1.0e30	1.0e30

Table 3.7. III-V Compound Semiconductor Analytic Mobility

property	input file name	units	Si	GaAs
parameter used in the carrier-carrier scattering term	A_CCS	$\frac{1}{cm-V-s}$	1.04e21	1.04e21
parameter used in the carrier-carrier scattering term	B_CCS	$\frac{1}{cm^2}$	7.45e13	7.45e13
parameter used in mobility model	A_LIC	none	1.0	1.0
parameter used in mobility model	B_LIC	none	2.126	1.0
parameter used in the mobility model	C_LIC	none	0.0	0.0
exponent used in the mobility model	EX_LIC	none	0.715	0.0
room temperature mobility for electrons	MUN0_LAT	$\frac{cm^2}{V-s}$	1430.0	8500.0
exponent used in the lattice scattering term for electrons	EXN_LAT	none	2.3	0.0
parameter used in the ionized impurity scattering term	AN_IIS	$\frac{1}{cm-V-s}$	2.4e21	2.4e21
parameter used in the ionized impurity scattering term for electrons	BN_IIS	$\frac{1}{cm^3}$	1.37e20	1.37e20
room temperature mobility for holes	MUP0_LAT	$\frac{cm^2}{V-s}$	495.0	400.0
exponent used in the lattice scattering term for holes	EXP_LAT	none	2.2	0.0
parameter used in the ionized impurity scattering term for holes	AP_IIS	$\frac{1}{cm-V-s}$	5.2e20	5.2e20
parameter used in the ionized impurity scattering term for holes	BP_IIS	$\frac{1}{cm^3}$	5.63e19	5.63e19

Table 3.8. Carrier-Carrier Scattering Mobility Model Parameters

property	input file name	units	Si	GaAs
parameter in bulk term	MUN0_LSM	$\frac{cm^2}{V-s}$	52.2	0.0
parameter in bulk term	MUN1_LSM	$\frac{cm^2}{V-s}$	43.4	0.0
parameter in bulk term	MUN2_LSM	$\frac{cm^2}{V-s}$	1417.0	1.0e6
parameter in bulk term	CRN_LSM	$\frac{\#}{cm^3}$	9.68e16	9.68e16
parameter in bulk term	CSN_LSM	$\frac{\#}{cm^3}$	3.43e20	0.0
parameter in acoustic term	BN_LSM	cm/sec	4.75e7	1.0e10
parameter in acoustic term	CN_LSM	$(K - cm/s)(V/cm)^{-2/3}$	1.74e5	0.0
parameter in acoustic term	DN_LSM	$(cm^2/V - s)(V/cm)^{exp8.Lsm}$	5.82e14	1.0e6
exponent in bulk term	EXN1_LSM	none	0.680	0.0
exponent in bulk term	EXN2_LSM	none	2.0	0.0
exponent in bulk term	EXN3_LSM	none	2.5	0.0
exponent in acoustic term	EXN4_LSM	none	0.125	0.0
exponent in surface roughness term	EXN8_LSM	none	2.0	0.0
parameter in bulk term	MUP0_LSM	$\frac{cm^2}{V-s}$	44.9	0.0
parameter in bulk term	MUP1_LSM	$\frac{cm^2}{V-s}$	29.0	0.0
parameter in bulk term	MUP2_LSM	$\frac{cm^2}{V-s}$	470.5	1.0
parameter in bulk term	CRP_LSM	$\frac{\#}{cm^3}$	2.23e17	2.23e17
parameter in bulk term	CSP_LSM	$\frac{\#}{cm^3}$	6.1e20	0.0
parameter in acoustic term	BP_LSM	$cm\ sec$	9.93e6	1.0e10
parameter in acoustic term	CP_LSM	$(K - cm/s)(V/cm)^{-2/3}$	8.84e5	0.0
parameter in acoustic term	DP_LSM	$(cm^2/V - s)(V/cm)^{exp8.Lsm}$	2.05e14	1.0e6
exponent in bulk term	EXP1_LSM	none	0.719	0.0
exponent in bulk term	EXP2_LSM	none	2.0	0.0
exponent used in the bulk term for holes	EXP3_LSM	none	2.2	0.0
exponent in the acoustic term	EXP4_LSM	none	0.0317	0.0
exponent in surface roughness term	EXP8_LSM	none	2.0	0.0
parameter in bulk term	PC_LSM	$\frac{\#}{cm^3}$	9.23 e16	0.0

Table 3.10. Lombardi Surface Mobility Model Parameters

property	input file name	units	Si	GaAs
parameter at high dopant/carrier levels	MMNN_UM	$\frac{cm^2}{V-s}$	52.2	0.0
maximum electron mobility	MMXN_UM	$\frac{cm^2}{V-s}$	1417.0	8500.0
reference impurity concentration	NRFN_UM	$\frac{\#}{cm^3}$	9.68e16	1.0e30
exponent for electron mobility	ALPN_UM	none	0.68	1.0
exponent for temperature dependence of lattice scattering	TETN_UM	none	2.285	0.0
reference impurity concentration for donors	NRFD_UM	$\frac{\#}{cm^3}$	4.0e20	1.0e30
factor for ultra-high doping effects	CRFD_UM	none	0.21	1.0e30
parameter at high dopant/carrier levels	MMNP_UM	$\frac{cm^2}{V-s}$	44.9	0.0
maximum hole mobility	MMXP_UM	$\frac{cm^2}{V-s}$	470.5	400.0
reference impurity concentration	NRFP_UM	$\frac{\#}{cm^3}$	2.23e17	1.0e30
exponent for hole mobility	ALPP_UM	none	0.719	1.0
exponent for temperature dependence of lattice scattering	TETP_UM	none	2.247	0.0
reference impurity concentration	NRFA_UM	$\frac{\#}{cm^3}$	7.2e20	1.0e30
factor for ultra-high doping effects	CRFA_UM	none	0.5	1.0e30

Table 3.11. Philips Unified Mobility Model Parameters

property	input file name	units	Si	GaAs
part of surface roughness term for electrons	AN_LUC	none	2.58	2.58
part of surface roughness term for holes	AP_LUC	none	2.18	2.18
acoustic term for electrons	BN_LUC	cm/s	3.61e7	3.61e7
parameter in acoustic term for holes	BP_LUC	cm/s	1.51e7	1.51e7
acoustic term for electrons	CN_LUC	$(\text{cm}^2/V - s) (V/\text{cm})^{1/3} \text{cm}^{3EXN4.LUC}$	1.70e4	1.70e4
parameter used in acoustic term for holes	CP_LUC	$(\text{cm}^2/V - s) (V/\text{cm})^{1/3} \text{cm}^{3EXP4.LUC}$	4.18e3	4.18e3
acoustic term for electrons	DN_LUC	$(\text{cm}^2/V - s) (V/\text{cm})^{\gamma_n}$	3.58e18	3.58e18
acoustic term for holes	DP_LUC	$(\text{cm}^2/V - s) (V/\text{cm})^{\gamma_p}$	4.10e15	4.10e15
factor for surface roughness term (electrons)	FN_LUC	$\text{cm}^{3(1-EXN9.LUC)}$	6.85e-21	6.85e-21
factor for surface roughness term (holes)	FP_LUC	$\text{cm}^{3(1-EXP9.LUC)}$	7.82e-21	7.82e-21
exponent of temperature in acoustic term	KN_LUC	none	1.7	1.7
exponent of temperature in acoustic term	KP_LUC	none	0.9	0.9
exponent of total impurity concentration	EXN4_LUC	none	0.0233	0.0233
exponent of total impurity concentration	EXP4_LUC	none	0.0119	0.0119
exponent of total impurity concentration	EXN9_LUC	none	0.0767	0.0767
exponent of total impurity concentration	EXP9_LUC	none	0.123	0.123

Table 3.12. Lucent Mobility Model Parameters

property	input file name	units	Si	GaAs	PolySi	Oxide	Ge
bandgap reference value	EG300	eV	1.08	1.424	1.08	9.0	0.66
N_C at 300K	NC300	$\frac{1}{cm^3}$	2.8e19	4.7e17	2.8e19	2.8e19	1.04e19
N_V at 300K	NV300	$\frac{1}{cm^3}$	1.04e19	7.0e18	1.04e19	1.04e19	6.0e18
α in the bandgap function	EGALPH	$\frac{eV}{K}$	4.73e-4	5.405e-4	4.73e-4	4.73e-4	4.77e-4
β in the bandgap function	EGBETA	K	636.0	204.0	636.0	636.0	235.0
exponent of temperature for describing N_C	NC_F	none	1.5	1.5	1.5	1.5	1.5
exponent of temperature for describing N_V	NV_F	none	1.5	1.5	1.5	1.5	1.5

Table 3.13. Intrinsic Concentration Calculation Parameters

property	input file name	units	Si	GaAs
band degeneracy factor (n-type)	GB_N	none	2.0	2.0
band degeneracy factor (p-type)	GB_P	none	4.0	2.0
constant term used in the calculation of the band ionization energy (n-type)	EB0_N	eV	0.045	0.005
constant term used in the calculation of the band ionization energy (p-type)	EB0_P	eV	0.045	0.005
prefactor for doping dependent term (n-type)	ALPHA_N	$\frac{eV}{cm^3}$	3.1e-8	3.1e-8
prefactor for doping dependent term (p-type)	ALPHA_P	$\frac{eV}{cm^3}$	3.037e-8	3.037e-8
prefactor for temperature dependent term (n-type)	BETA_N	none	200.0	200.0
prefactor for temperature dependent term (p-type)	BETA_P	none	200.0	200.0
exponent of temperature (n-type)	GAMMA_N	none	1.0	1.0
exponent of temperature (p-type)	GAMMA_P	none	0.95	0.95
minimum impurity concentration for which high doping transition applies	HDT_MIN	$\frac{1}{cm^3}$	1.0e18	1.0e18
maximum impurity concentration for which high doping transition applies	HDT_MAX	$\frac{1}{cm^3}$	1.0e19	1.0e19

Table 3.14. Incomplete Ionization Model Parameters

property	input file name	units	Si	GaAs	Oxide	Ge
parameter for calculating $\alpha_{n,ii}^{\infty}$	N_IONIZATION	$\frac{1}{cm}$	7.03e5	2.994e5	7.03e2	1.55e7
parameter for calculating $\alpha_{n,ii}^{\infty}$	N_ION_1	$\frac{1}{cm-K}$	0.0	0.0	0.0	0.0
parameter for calculating $\alpha_{n,ii}^{\infty}$	N_ION_2	$\frac{1}{cm-K^2}$	0.0	0.0	0.0	0.0
parameter for calculating $\alpha_{p,ii}^{\infty}$	P_IONIZATION	$\frac{1}{cm}$	1.528e6	2.215e5	1.528e3	1.0e7
parameter for calculating $\alpha_{p,ii}^{\infty}$	P_ION_1	$\frac{1}{cm-K}$	0.0	0.0	0.0	0.0
parameter for calculating $\alpha_{p,ii}^{\infty}$	P_ION_2	$\frac{1}{cm-K^2}$	0.0	0.0	0.0	0.0
used for ratio of critical electric field to parallel electric field	EXN_II	none	1.0	1.6	1.0	1.0
used for ratio of critical electric field to parallel electric field	EXP_II	none	1.0	1.75	1.0	1.0
optical phonon energy	OP_PH_EN	eV	0.063	0.035	0.063	0.037
optical phonon mean free path for electrons at 300 K	LAN300	cm	10.4542e-7	3.5272e-6	9.66151e-8	6.88825e-7
optical phonon mean free path for holes at 300 K	LAP300	cm	6.32079e-7	3.6765e-6	5.86026e-8	8.39505e-7

Table 3.15. Impact Ionization Model Parameters

4. Examples

A number of simple examples of **Charon** usage are given. If you're using a **Charon** distribution these examples can be found in the directory `Charon_distrib_1.0/doc/examples`.

Note that the following examples are for the purposes of illustration only and do not necessarily represent realistic devices. The examples are purposely simple and were primarily created with the goal of being manageable on a standard workstation.

4.1 IV sweep for a simple one-dimensional diode

The problem to be simulated is shown in Figure 4.1. The doping is an abrupt junction located at the exact center of the diode ($x = 0.5\mu m$). Doping is symmetric with $N_d = N_a = 1.0e16$. Because the electrical contacts run the entire height of the device there is no two-dimensional character to this problem.

Figure 4.1. A simple pseudo-1D Silicon diode geometry.

The first step to solving any problem using **Charon** is to generate the discretization geometry. This is done using an external mesh generation package such as *Cubit* [3]. **Charon**

currently reads and writes files in the *Exodus II* file format commonly used at Sandia National Laboratories [2] so any program that outputs the mesh in *Exodus II* format can be used. A typical mesh for the problem is shown in Figure 4.2. This mesh was generated using the Cubit journal file shown in Figure 4.3.

Figure 4.2. Mesh for the problem geometry in Figure 4.1. This mesh was created in *Cubit* via the journal file shown in Figure 4.3.

Once the mesh has been generated the next step in the process is to define the input command file, or files, necessary to perform the simulation. In this case an *IV-sweep* is to be performed. To accomplish this more easily **Charon** is distributed with a helper script, *iv-generate.pl*, to automate the process. *iv-generate.pl* requires two input command files. As is the case for most **Charon** drift-diffusion simulations a good initial guess must be generated in order to ensure convergence. **Charon** typically accomplishes this by first running a *NLP* simulation (see Section 1.2). For *iv-generate.pl* this means that an input-command file must be created for the *NLP* portion of the simulation. For the problem at hand the necessary file is given in Appendix A.1 (see Chapter 3 for an in-depth explanation of the the parameters in the input file).

The next step when using *iv-generate.pl* is to generate a skeleton input-command file for the drift-diffusion simulations to be performed. This file is identical to an input-command file for a drift-diffusion simulation with the exception that the file name used to specify an initial guess and the boundary condition defining the applied electric potential use parameters that *iv-generate.pl* will recognize and replace. For this problem the skeleton input-command file is given in Appendix A.2 and the parameters recognized by *iv-generate.pl* are highlighted.

```
# The two-dimensional mesh will be defined on a surface of
# this brick
create brick x 1.0 y 0.5 z 0.1

# This will make all coordinates positive. Not necessary
# but can be convenient
move vertex 4 location 0 0 0

# Define the sidesets that will be the electrical contacts
# for the simulations
sideset 1 curve 3
sideset 2 curve 1

# Element blocks for a single material
block 1 surface 1
block 1 element type quad4

## Long side
curve 2 4 interval 100

## Short side (contacts)
curve 3 1 interval 10

# Actually mesh the geometry
mesh surface 1
merge all

# Charon uses CGS units so transform output to
# be in centimeters
transform mesh output scale 1.0e-4

# Output the Exodus II mesh
export mesh "pn-diode.gen" dimension 2
```

Figure 4.3. Cubit journal file used to generate the mesh illustrated in Figure 4.2.

Once the mesh and the input-command files are in place the actual simulations can be carried out. For the purposes of this example the simulation name will be *pn-diode*. This means that the *NLP* input command file, given in Appendix A.1, should be named *pn-diode.nlp.inp*, the drift-diffusion case file, given in Appendix A.2, should be named *pn-diode.case* and the mesh file should be copied to two files named *pn-diode.nlp.gen* and *pn-diode.dd.gen*. This is summarized in Table 4.1. Assuming the run is to be performed

File Name	Purpose
<i>pn-diode.nlp.gen</i>	Mesh geometry input file for <i>NLP</i> simulation
<i>pn-diode.dd.gen</i>	Mesh geometry input file for drift-diffusion simulation. Note that this file is identical to <i>pn-diode.nlp.gen</i> .
<i>pn-diode.nlp.inp</i>	Input command file for the <i>NLP</i> simulation portion of the run.
<i>pn-diode.case</i>	Input command file for drift-diffusion portion of the run. This file contains template parameters that will be replaced by <i>iv-generate.pl</i> .

Table 4.1. Summary of the input files names for example *PN*-diode problem in Section 4.1.

on a Linux workstation the command to perform the actual run is shown in Figure 4.4. An explanation of the options to *iv-generate.pl* is given in Table 4.2.

```
Charon_distrib_1.0/src/charon/util-bin/iv-generate.pl --vstart=0.0 \
--nsteps=12 --vinc=0.05 --mpi \
--source-directory=~ /Charon_distrib_1.0/src/ \
--build-opts=2D+64BITgnu4+opt+tri8 \
pn-diode
```

Figure 4.4. Linux command for executing **Charon** run for example diode problem in Section 4.1. Note that this is wrapped for readability. Actual usage should be a single line. Also the name of the executable may vary depending on what build options were used during the build process.

Upon successful completion of the run there will be a number of output files. The most relevant files for post-run analysis are the Exodus II output files, with a *exo* extension, and the file that contains the scalar electric current at the contacts for each bias point named

Argument	Purpose
<code>-vstart</code>	The starting applied potential.
<code>-vinc</code>	The applied potential increment for each step.
<code>-nsteps</code>	The number of steps to take. Given the <code>-vstart</code> of 0.0 and <code>-vinc</code> value of 0.05 the end voltage can be calculated as $end\ voltage = vstart + nsteps * vinc$. For the example invocation shown in Figure 4.4 the ending voltage would be 0.6V.
<code>-mpi</code>	This means that the executable is an <i>MPI</i> compiled version. In general you always need to specify this option.
<code>-source-directory</code>	The source directory for Charon .
<code>-build-opts</code>	Build options for Charon .
<code>-help</code>	This will give you a summary of the options available to <i>iv-generate.pl</i> . (NOTE: Not used in example.)

Table 4.2. Options for *iv-generate.pl*.

currents.dat. In addition to the Exodus II file containing the results for the *NLP* simulation, *pn-diode.nlp.exo*, there are Exodus II results files for each bias point in the run with full drift-diffusion results. Typically these Exodus II results files contain data for the electric potential and carrier densities at each vertex, or node, in the mesh and can be visualized using post-processing tools such as *Enight*[4] or *Paraview* [5]. An example of the results of visualizing the base 10 logarithm of the electron density at bias point 0.3V is shown in Figure 4.5.

Often the only quantities of interest are the scalar electric currents, in which case the file *currents.dat* can be utilized. It contains, for each bias point, the scalar electric current at each of the defined contacts. A plot of the current at the contact for the problem in Section 4.1 is shown in Figure 4.1.

If any problems occur during a run the screen output of the individual simulations is output to a file with a *runout* extension and any error messages are captured in these files. Also, if any errors are encountered while parsing the input-command file the files with the *ech* extension will show where the parsing error occurred, otherwise these files are simply echoes of the input-command file as it's read by **Charon**.

Figure 4.5. Plot of the quantity $\log_{10}(\text{electron density})$ at bias $0.3V$ for the simple diode example problem in Section 4.1.

4.2 A two-dimensional Bipolar Junction Transistor

BJT under normal operating conditions

In this example problem a bipolar junction transistor, or *BJT*, with defects, will be simulated. The geometry of the problem is shown in Figure 4.7.

The first step in the process is to generate a mesh for the geometry shown in Figure 4.7. As was the case in the diode example problem in Section 4.1 this is accomplished here with the meshing program *Cubit*[3]. The *Cubit journal* file used to generate the mesh is shown in Figure 4.8.

To begin the actual simulation process the *BJT* will be simulated under normal operating conditions and an *IV* sweep will be performed. From the *IV* sweep a *Gummel* plot will be generated. This serves two purposes. First it is easier to verify the correctness of the input-command file, and the problem setup in general, without the added complication of defect physics. Second, it allows the user to find the steady-state operating point of the

Figure 4.6. Electrical current at the contacts of the example diode problem in Section 4.1.

Figure 4.7. The geometry of a two-dimensional bipolar junction transistor.

```
# The two-dimensional mesh will be defined on a surface of
# this brick
brick x 1.0 y 0.5 z 1

# This will make all coordinates positive. Not necessary
# but can be convenient
move vertex 4 location 0 0 0

# Cut the volume in the location of the emitter and base
# contacts
webcut volume 1 plane xplane offset 0.3
merge all
webcut volume 1 plane xplane offset 0.7
merge all

# Define the sidesets that will form the contacts
sideset 1 curve 26
sideset 2 curve 35
sideset 3 curve 28 41 36

# Set the density/size of the volumes/elements
surface 15 24 20 interval size 0.005

# Element blocks for a single material
block 1 surface 15 24 20
block 1 element type QUAD4

# Mesh the geometry
mesh surface 15 24 20

# Charon uses CGS units so transform output to
# be in centimeters
transform mesh output scale 0.0001

# Output the Exodus II mesh
export mesh "bjt.gen" dimension 2 overwrite
```

Figure 4.8. Cubit journal file used to generate the mesh for the small *BJT* example problem geometry of Figure 4.7.

device at which defects will be introduced and to use the files obtained during the *IV* sweep as the initial guess for the defect problem.

As was the case for the example diode problem in Section 4.1, to accomplish the steady-state analysis requires two input-command files, the nonlinear Poisson, or *NLP*, input-command file and the templated input-command file for the drift-diffusion *IV* sweep. The *NLP* input-command file is given in Appendix A.3 and the templated file is given in Appendix A.4. In the *bjt.case* file the actual templated keywords have been highlighted in red. Aside from the highlighted keywords, the *bjt.case* file is identical to an input-command file for a drift-diffusion run.

In addition to the two aforementioned input-command files, a third input file, *doping.inp*, is used. This file is included, literally, in the other two input-command files via the line `include "doping.inp"`. The file contains definitions of analytic functions that are often used to generate approximate doping profiles from data, such as that obtained via secondary ion mass spectroscopy (SIMS). The contents of the file are given in Appendix A.5. A plot of the doping profile is shown in Figure 4.9.

The last set of files required are the input mesh geometry, or **.gen*, Exodus II files. While two are required, one for the *NLP* portion of the simulation and one for the drift-diffusion portion, they contain identical data and so on Unix systems you can simply create a logical link for one of them to the other. A summary of the files required for the *BJT* simulation under normal operating conditions is given in Table 4.3.

File Name	Purpose
<i>bjt.nlp.gen</i>	Mesh geometry input file for <i>NLP</i> simulation
<i>bjt.dd.gen</i>	Mesh geometry input file for drift-diffusion simulation. Note that this file is identical to <i>bjt.nlp.gen</i> .
<i>bjt.nlp.inp</i>	Input command file for the <i>NLP</i> simulation portion of the run.
<i>bjt.case</i>	Input command file for drift-diffusion portion of the run. This file contains template parameters that will be replaced by <i>iv-generate.pl</i> .
<i>doping.inp</i>	Portion of the input-command file defining doping profiles. This file is included by the other two input-command files.

Table 4.3. Summary of the input files names for the *BJT* example problem in Section 4.2.

Running the *BJT* problem is similar to running the diode problem of Section 4.1. The script

Figure 4.9. Doping concentrations taken from the top of the device under the emitter contact. Note that the emitter doping drops off in both the x and y directions with a Gaussian variation. The contents of the file that generates this doping profile is given in Appendix A.5.

iv-generate.pl will be used to run an *IV* sweep over a portion of the operational range of the *BJT*. The difference is that the *BJT* has three contacts, as opposed to the two contacts on the diode. In this case we will simulate the *BJT* operating as a gain device. This means forward biasing the *EB*-junction and keeping *BC*-junction reverse biased. The easiest way to accomplish this with the *iv-generate.pl* script is to bias the base and collector contacts with an equal positive voltage while holding the emitter bias at zero volts. This means that V_{bc} , the voltage across the *BC* junction, will be maintained at zero volts while V_{be} , the voltage across the *BE* junction, will be increasingly forward biased, the value of which will be the voltage applied to the base contact. A circuit diagram illustrating the simulation parameters is shown in Figure 4.10.

Figure 4.10. Circuit diagram for *BJT* problem.

The necessary files to run this portion of the example problem are distributed in the directory

```
Charon_distrib_1.0/doc/examples/bjt/run-gummel.
```

For information on the **Charon** distribution see Section 2.1.

Assuming a Unix-like environment the command to execute the simulation is shown in Figure 4.11. Upon completion of the command given in Figure 4.11 there will be a number of results files. These consist of fourteen *Exodus II* files with the *.exo* extension, one for each

bias point in the drift-diffusion portion of the simulation and one for the *NLP* portion of the simulation. These files contain the original mesh and the results for the carrier concentrations and electric potential values. For each *Exodus II* file there is also a corresponding output file, with a *.runout* extension, that captures what would normally be output to the screen. The last result file is *currents.dat*. This file contains the scalar electric current at each of the contacts. By convention if electric current is exiting the device it's sign is positive and if electric current is entering the device the sign will be positive.

```
Charon_distrib_1.0/src/charon/util-bin/iv-generate.pl --vstart=0.0 \  
--nsteps=14 --vinc=0.05 --mpi \  
--source-directory=~ /Charon_distrib_1.0/src/ \  
--build-opts=2D+64BITgnu4+opt+tri8 \  
bjt
```

Figure 4.11. Unix command for executing **Charon** run for example *BJT* problem in Section 4.2. Also see Figure 4.4 and Table 4.2. Note that this is wrapped for readability. Actual usage should be a single line. Also the name of the executable may vary depending on what build options were used during the build process.

A plot of the collector and base currents, often referred to as a Gummel plot, generated from the data in the file *currents.dat*, is shown in Figure 4.12. A related plot, the device gain, or $\frac{I_c}{I_b}$, is shown in Figure 4.13.

BJT with transient defects

Now that a baseline of operation has been established for the *BJT* under normal, steady-state, operating conditions a set of transient defects will be added and the performance will be studied. **Charon** typically models damage due to “heavy” radiation such as neutron radiation which causes displacement of atoms in the semiconductor lattice. This is commonly referred to as a Frenkel defect. Within **Charon** the resulting defects are modeled as “species” which are able to react with other defects and with other atoms within the semiconductor. A summary of these reactions is given in Appendix B. To quantify the chemistry that is associated with the defect physics implemented in **Charon**, **Charon** relies on *Chemkin* [10, 11].

In addition to specifying each defect as an individual species the charge state of a particular defect is also treated as an individual defect. In other words, if you have a defect, V ,

Figure 4.12. Gummel plot for the *BJT* illustrated in Figure 4.7 under normal operating conditions.

Figure 4.13. Current gain for the *BJT* illustrated in Figure 4.7 under normal operating conditions.

which has three possible charge states, negative, neutral and positive then that defect has three species associated with it. In general the convention for this is to use the letter “*M*” to indicate the negative charge state a “*O*” (zero) to indicate a neutral charge, and the letter “*P*” to indicate a positive charge state. To make this concept more concrete consider a vacancy defect represented by the letter “*V*”. If the vacancy supports the three aforementioned charge states then the three species in **Charon** would be:

- V0 – Neutral vacancy,
- VP – Vacancy with a positive charge,
- VM – Vacancy with a negative charge.

With this concept in mind consider the *Chemkin* input file to be utilized for this example is given in Appendix A.6.

This page is left intentionally blank

Figure 1. The study area and the location of the study sites.

```

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ Charon-specific input section
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
CHARON

    $ Physics setup
    physics = 1, semiconductor, nlp

        contact 1
            sideset id = 1
        end

        contact 2
            sideset id = 2
        end

    output current components

end
```

```
$ generally semiconductor problems need to be scaled.
scaled variables = true
scaled mesh = max_x

$ NOX parameters
nox parameters

    use relative tolerance test = false
    use absolute tolerance test = true
    absolute tolerance = 1.0e-4
    wrms absolute tolerance = 1.0e-10
    wrms relative tolerance = 1.0e-6
    maximum nonlinear iterations = 30
    linear scaling = row sum

    sublist "Direction"
        "Method" = "Newton"
        sublist "Newton"
            "Rescue Bad Newton Solve" = "true"
            sublist "Linear Solver"
                "Aztec Solver" = "GMRES"
                "Max Iterations" = 500
                "Tolerance" = 1.0e-6
                "Output Frequency" = 20
                "Preconditioner" = "Aztec00"
                "Aztec Preconditioner" = "ilut"
                "Preconditioner Operator" = "Use Jacobian"
            END
        END
    END

end

$ Element block to material map
block 1
    physics 1
    material 1
end

jacobian = ad

initial guess = true

$ Nevada time parameters
time step scale = 1.0
```

```

constant time step = 1.0
gradual startup factor = 1.0

$ Defines a function implemented as a C++ method.
$ This test is for electron mobility
METHOD FUNCTION 1
    key = e_mobility
    model name = arora
    300.15 $ Temperature (Kelvin)
END
METHOD FUNCTION 2
    key = h_mobility
    model name = arora
    300.15 $ Temperature (Kelvin)
END
METHOD FUNCTION 3
    key = e_diff_coef
    300.15 $ Temperature (Kelvin)
END
METHOD FUNCTION 4
    key = h_diff_coef
    300.15 $ Temperature (Kelvin)
END

METHOD FUNCTION 10
    key = doping
    model name = step junction
    junction location = 0.5e-4
    Na = 1.0e16
    Nd = 1.0e16
    configuration = pn
END

END $CHARON

plot variable
    electric_potential
    electron_density
    hole_density
    doping
end

$$$$$$$$$$$$$$$$$$$$
$ Generic Nevada input
$$$$$$$$$$$$$$$$$$$$

```

```

double precision exodus
termination cycle = 1
emit plot, cycle 1

$$$$$$$$$$$$$$$$$$$$
$ Material specs
$$$$$$$$$$$$$$$$$$$$
Material 1 SI
  model 1
end

MODEL 1 CHARON SEMICONDUCTOR
  MATERIAL NAME = SI
  RELATIVE PERMITTIVITY = 11.8
  ELECTRON MOBILITY = method function 1
  HOLE MOBILITY = method function 2
  ELECTRON DIFFUSION COEFFICIENT = method function 3
  HOLE DIFFUSION COEFFICIENT = method function 4
  DOPING = method function 10
  ELECTRON LIFETIME = 1.0e-7
  HOLE LIFETIME = 1.0e-7
END

crt: off

```

A.2 Case file for IV-sweep simulation of PN-diode example problem

```

$-----
$ ID:      diode dd template for IV-curve generation
$ Title: CHARON: Test of semiconductor drift-diffusion model on
$          a simple diode
$-----

title
  CHARON: example of drift-diffusion in a simple diode

$$$$$$$$$$$$$$$$$$$$
$ Charon-specific input section
$$$$$$$$$$$$$$$$$$$$
CHARON

```

```
$ Physics setup
physics = 1, semiconductor, dd

species source = auger
  off
end

contact 1
  sideset id = 1
end

contact 2
  sideset id = 2
end

current linear solver tolerance = 1.0e-12

use current convergence in nox = true
current wrms relative tolerance = 1.0e-8
current wrms absolute tolerance = 1.0e-14

supg tau calculation = linear
supg length scale calculation = stream

end

stabilization options
  supg = true
end

$ generally semiconductor problems need to be scaled.
scaled variables = true
scaled mesh = max_x

$ NOX/NL parameters
nox parameters

  use relative tolerance test = true
  use absolute tolerance test = false
  absolute tolerance = 1.0e-12
  relative tolerance = 1.0e-4
  wrms absolute tolerance = 1.0e-16
  wrms relative tolerance = 1.0e-8
  wrms alpha = 0.0
  wrms beta = 0.5
```

```

maximum nonlinear iterations = 30
linear scaling = row sum

print solution statistics = true

sublist "Direction"
  "Method" = "Newton"
  sublist "Newton"
    "Rescue Bad Newton Solve" = "true"
    sublist "Linear Solver"
      "Aztec Solver" = "GMRES"
      "Max Iterations" = 800
      "Tolerance" = 1.0e-06
      "Output Frequency" = 20
      "Overlap" = 1
      "Preconditioner" = "New Ifpack"
      sublist "Ifpack"
        "schwarz: reordering type" = "rcm"
        "fact: level-of-fill" = 3
      end
    end
  end
END
END
END

end

$ Element block to material map
block 1
  physics 1
  material 1
end

jacobian = ad

initial guess = exodus
  file name = "TEMPLATE_PARAM_FN"
  time index = 1
end

$ Applied potential. For forward bias apply positive potential to
$ sideset 1
dirichlet bc, electric_potential, sideset 1 TEMPLATE_PARAM_1

$ Nevada time parameters
time step scale = 1.0

```

```
constant time step = 1.0
gradual startup factor = 1.0

METHOD FUNCTION 1
  key = e_mobility
  model name = arora
END
METHOD FUNCTION 2
  key = h_mobility
  model name = arora
END
METHOD FUNCTION 3
  key = e_diff_coef
END
METHOD FUNCTION 4
  key = h_diff_coef
END

METHOD FUNCTION 5
  key = h_lifetime
  model name = consrh
END

METHOD FUNCTION 6
  key = e_lifetime
  model name = consrh
END

METHOD FUNCTION 10
  key = doping
  model name = step junction
  junction location = 0.5e-4
  Na = 1.0e16
  Nd = 1.0e16
  configuration = pn
END

END $CHARON

plot variable
  electric_potential
  electron_density
  hole_density
  doping
end
```

```

$$$$$$$$$$$$$$$$$$$$
$ Generic Nevada input
$$$$$$$$$$$$$$$$$$$$
double precision exodus
termination cycle = 1
emit plot, cycle 1

$$$$$$$$$$$$$$$$$$$$
$ Material specs
$$$$$$$$$$$$$$$$$$$$
Material 1 SI
  model 1
end

MODEL 1 CHARON SEMICONDUCTOR
  MATERIAL NAME = SI
  RELATIVE PERMITTIVITY = 11.8

  ELECTRON MOBILITY = method function 1
  HOLE MOBILITY = method function 2

  ELECTRON DIFFUSION COEFFICIENT = method function 3
  HOLE DIFFUSION COEFFICIENT = method function 4

  HOLE LIFETIME = method function 5
  ELECTRON LIFETIME = method function 6

  DOPING = method function 10
END

crt: off

```

A.3 Input-command file for nonlinear Poisson simulation for the example BJT problem.

```

$-----
$ ID: BJT NLP
$ Title: CHARON: Nonlinear Poisson on a BJT
$-----

```



```

title
  CHARON: BJT NLP simulation

debug mode, parser    $ This will give better info if parsing fails

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ Charon-specific input section
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
CHARON

$ Physics setup
physics = 1, semiconductor, nlp

  contact 1
    sideset id = 1
  end

  contact 2
    sideset id = 2
  end

  contact 3
    sideset id = 3
  end

  output current components

end

$ generally semiconductor problems need to be scaled.
scaled variables = true
scaled mesh = max_x

$ NOX parameters
nox parameters

  maximum nonlinear iterations = 100
  use absolute tolerance test = false
  use relative tolerance test = true
  absolute tolerance = 1.0e-6
  relative tolerance = 1.0e-4
  wrms absolute tolerance = 1.0e-10
  wrms relative tolerance = 1.0e-6
  wrms tolerance = 1.0
  wrms alpha = 1.0

```

```

wrms beta = 0.5
linear scaling = row sum

sublist "Direction"
  "Method" = "Newton"
  sublist "Newton"
    "Rescue Bad Newton Solve" = "true"
    sublist "Linear Solver"
      "Aztec Solver" = "GMRES"
      "Max Iterations" = 200
      "Tolerance" = 1.0e-6
      "Output Frequency" = 20
      "Aztec Preconditioner" = "ilut"
      "Preconditioner" = "Aztec00"
      "Preconditioner Operator" = "Use Jacobian"
    END
  END
END

end

$ Element block to material map
block 1
  material 1
  physics 1
end

jacobian = ad

initial guess = true

$ Nevada time parameters
time step scale = 1.0
constant time step = 1.0
gradual startup factor = 1.0

$ Defines a function implemented as a C++ method.
$ This test is for electron mobility
METHOD FUNCTION 1
  key = e_mobility
  model name = arora
  300.15 $ Temperature (Kelvin)
END
METHOD FUNCTION 2
  key = h_mobility

```

```

    model name = arora
    300.15 $ Temperature (Kelvin)
END
METHOD FUNCTION 3
    key = e_diff_coef
    300.15 $ Temperature (Kelvin)
END
METHOD FUNCTION 4
    key = h_diff_coef
    300.15 $ Temperature (Kelvin)
END

```

```
$ Doping specification
include "doping.inp"
```

END \$CHARON

```
plot variable
  electric_potential
  electron_density
  hole_density
  doping
end
```

```

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ Generic Nevada input
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
double precision exodus
termination cycle = 1
emit plot, cycle 1

```

```

$$$$$$$$$$$$$$$$$$$
$ Material specs
$$$$$$$$$$$$$$$$$$$
Material 1 SI
    model 1
end

```

MODEL 1 CHARON SEMICONDUCTOR
RELATIVE PERMITTIVITY = 11.8

```
ELECTRON MOBILITY = method function 1
HOLE MOBILITY = method function 2
```

ELECTRON DIFFUSION COEFFICIENT = method function 3

```

HOLE DIFFUSION COEFFICIENT = method function 4

DOPING = method function 5
MATERIAL NAME = SI
END

crt: off

```

A.4 Templated input-command file for the example BJT problem

```

$-----
$ ID: Small BJT
$ Title: CHARON: Full drift-diffusion SGFVM simulation for a BJT
$-----

title
    CHARON: drift-diffusion in a small BJT

debug mode, parser    $ This will give better info if parsing fails

$$$$$$$$$$$$$$$$$$$$
$ Charon-specific input section
$$$$$$$$$$$$$$$$$$$$
CHARON

    $ Physics setup
    physics = 1, semiconductor, sg

    contact 1
        sideset id = 1
    end

    contact 2
        sideset id = 2
    end

    contact 3
        sideset id = 3
    end

    user current convergence in nox = true

```

```

    current wrms relative tolerance = 1.0e-8
    current wrms absolute tolerance = 1.0e-14
    current calculation verbose = false

end

$ generally semiconductor problems need to be scaled.
scaled variables = true
scaled mesh = max_x

$ NOX parameters
nox parameters

    maximum nonlinear iterations = 100
    use absolute tolerance test = false
    use relative tolerance test = true
    absolute tolerance = 1.0e-6
    relative tolerance = 1.0e-4
    wrms absolute tolerance = 1.0e-10
    wrms relative tolerance = 1.0e-6
    wrms tolerance = 1.0
    wrms alpha = 1.0
    wrms beta = 0.5
    linear scaling = row sum

    sublist "Direction"
        "Method" = "Newton"
        sublist "Newton"
            "Rescue Bad Newton Solve" = "true"
            sublist "Linear Solver"
                "Aztec Solver" = "GMRES"
                "Max Iterations" = 200
                "Tolerance" = 1.0e-6
                "Output Frequency" = 20
                "Aztec Preconditioner" = "ilut"
                "Preconditioner" = "Aztec00"
                "Preconditioner Operator" = "Use Jacobian"
            END
        END
    END

end

$ Element block to material map
block 1

```

```
material 1
physics 1
end

jacobian = ad

initial guess = exodus
  file name = "TEMPLATE_PARAM_FN"
  time index = 2
end

dirichlet bc, electric_potential, sideset 2 TEMPLATE_PARAM_1
dirichlet bc, electric_potential, sideset 3 TEMPLATE_PARAM_1

$ Nevada time parameters
time step scale = 1.0
constant time step = 1.0
gradual startup factor = 1.0

$ Defines a function implemented as a C++ method.
$ This test is for electron mobility
METHOD FUNCTION 1
  key = e_mobility
  model name = arora
  300.15 $ Temperature (Kelvin)
END
METHOD FUNCTION 2
  key = h_mobility
  model name = arora
  300.15 $ Temperature (Kelvin)
END
METHOD FUNCTION 3
  key = e_diff_coef
  300.15 $ Temperature (Kelvin)
END
METHOD FUNCTION 4
  key = h_diff_coef
  300.15 $ Temperature (Kelvin)
END

$ Doping specification
include "doping.inp"

END $CHARON
```

```

plot variable
  electric_potential
  electron_density
  hole_density
  doping
end

$$$$$$$$$$$$$$$$$$$$
$ Generic Nevada input
$$$$$$$$$$$$$$$$$$$$
double precision exodus
termination cycle = 1
emit plot, cycle 1

$$$$$$$$$$$$$$$$$$$$
$ Material specs
$$$$$$$$$$$$$$$$$$$$
Material 1 SI
  model 1
end

MODEL 1 CHARON SEMICONDUCTOR
  RELATIVE PERMITTIVITY = 11.8

  ELECTRON MOBILITY = method function 1
  HOLE MOBILITY = method function 2

  ELECTRON DIFFUSION COEFFICIENT = method function 3
  HOLE DIFFUSION COEFFICIENT = method function 4

  DOPING = method function 5
  MATERIAL NAME = SI
END

crt: off

```

A.5 Doping input-command file for example BJT problem

```

METHOD FUNCTION 5
  key = doping
  model name = function

```

```

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ Base gaussian P-doping
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
function_1 = m_gauss
nmax_1 = 1.0e17
type_1 = p

ymin_1 = 0.38e-4
ymax_1 = 1.0
ychar_1 = 0.065e-4

$ No variation in x
xmin_1 = -1.0
xmax_1 = 1.0
xchar_1 = 1.0

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ Emitter gaussian N-doping
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
function_2 = m_gauss
nmax_2 = 1.0e19
type_2 = n

ymin_2 = 0.5e-4
ymax_2 = 0.5e-4
ychar_2 = 0.038e-4

xmin_2 = 0.0e-4
xmax_2 = 0.2e-4
xchar_2 = 0.1e-4

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ Subcollector N-doping
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
function_3 = lerfc
type_3 = n
sign_3 = 1
direction_3 = y

nmin_3 = 1.0e15
nmax_3 = 1.0e18
location_3 = 0.08e-4
width_3 = 0.02e-4

```


A.6 Example input file for defects induced by pulse radiation

169

```
contact 2
  sideset id = 2
  type = pnp oned
end

contact 3
  sideset id = 3
end

supg tau calculation = tanh

output current components

bandgap narrowing = true

use fast current calculation = true

$ the parameters are set to match those set within r15
  mparam, si, ni = 1.0733e10
  mparam, si, eff_dens_cond = 2.8557e19
  mparam, si, eff_dens_vale = 3.0948e19
  mparam, si, elec_thermal_vel = 2.0335e7
  mparam, si, hole_thermal_vel = 1.6805e7
  mparam, si, nv300 = 3.0948e19
  mparam, si, eg300 = 1.1242
  mparam, si, n0_bgn = 1.3e17
  mparam, si, v0_bgn = 6.92e-3
$   mparam, si, con_bgn = 1.3e17
  mparam, si, lat_temp = 293.9
end

$ generally semiconductor problems need to be scaled.
scaled variables = true
scaled mesh = max_x

sensitivities
  param = "CONTACT VOLTAGE 1"
end

objective function
  name = current
  contacts = 1
  targets = -4.59e-10
end
```

```
time variation = transient

$ Time integration setup
time integration
  algorithm = generalized alpha
  initial time step size = 1e-9
  time step control = variable
  constant time step count = 2
  parameter, rho_inf = 0.0
  max number of step failures = 10
  absolute error = 1e-6
  relative error = 1e-3
  end time = 1.0
  output frequency type = step count
  output step count = 5
  max number of time steps = 500
  break points = 1.0 end
end $$time integration

$ NOX parameters
nox parameters

  maximum nonlinear iterations = 8
  use absolute tolerance test = false
  use relative tolerance test = true
  absolute tolerance = 1.0e-6
  relative tolerance = 1.0e-3
  wrms absolute tolerance = 1.0e-8
  wrms relative tolerance = 1.0e-4
  wrms tolerance = 1.0
  wrms alpha = 1.0
  wrms beta = 0.5
  linear scaling = row sum

  print solution statistics = true

  enable objective function constraint = true

  sublist "Constraints"
    "Constraint Parameter 1" = "CONTACT VOLTAGE 1"
    "Bordered Solver Method" = "Householder"
  END
  sublist "Direction"
    "Method" = "Newton"
```

```

    sublist "Newton"
      "Rescue Bad Newton Solve" = "true"
      sublist "Linear Solver"
        "Aztec Solver" = "GMRES"
        "Max Iterations" = 200
        "Tolerance" = 1.0e-9
        "Output Frequency" = 20
        "Preconditioner" = "New Ifpack"
        "Preconditioner Operator" = "Use Jacobian"
        "Overlap" = 1
        sublist "Ifpack"
          "schwarz: reordering type" = "rcm"
          "fact: level-of-fill" = 2
        end
      end
    END
  END
END

end

stabilize, convection = true
# lump, source = true

$ Element block to material map
block 1
  physics 1
  material 1
end

# block 2
#   material 1
# end

jacobian = ad

initial guess = exodus
  file name = "2c2907.dd.const_current.exo"
  ignore missing variables = true
  time index = 20
end

# Static Ohmic Boundary Conditions
dirichlet bc, electric_potential, sideset 1 0.61
dirichlet bc, electric_potential, sideset 2 0.0
dirichlet bc, electric_potential, sideset 3 -10.07

```

```
dirichlet bc, VMM, sideset 1 0.0
dirichlet bc, VM, sideset 1 0.0
dirichlet bc, V0, sideset 1 0.0
dirichlet bc, VP, sideset 1 0.0
dirichlet bc, VPP, sideset 1 0.0
dirichlet bc, SII0, sideset 1 0.0
dirichlet bc, SIIMM, sideset 1 0.0
dirichlet bc, SIIM, sideset 1 0.0
dirichlet bc, SIIPP, sideset 1 0.0
dirichlet bc, SIIP, sideset 1 0.0
dirichlet bc, BI0, sideset 1 0.0
dirichlet bc, BIM, sideset 1 0.0
dirichlet bc, BIP, sideset 1 0.0

dirichlet bc, VMM, sideset 3 0.0
dirichlet bc, VM, sideset 3 0.0
dirichlet bc, V0, sideset 3 0.0
dirichlet bc, VP, sideset 3 0.0
dirichlet bc, VPP, sideset 3 0.0
dirichlet bc, SII0, sideset 3 0.0
dirichlet bc, SIIMM, sideset 3 0.0
dirichlet bc, SIIM, sideset 3 0.0
dirichlet bc, SIIPP, sideset 3 0.0
dirichlet bc, SIIP, sideset 3 0.0
dirichlet bc, BI0, sideset 3 0.0
dirichlet bc, BIM, sideset 3 0.0
dirichlet bc, BIP, sideset 3 0.0

$ Nevada time parameters
time step scale = 1.0
gradual startup factor = 1.0

$ Defines a function implemented as a C++ method.
$ This test is for electron mobility
METHOD FUNCTION 1
  key = e_mobility
  model name = philips
  300.15 $ Temperature (Kelvin)
END
METHOD FUNCTION 2
  key = h_mobility
  model name = philips
  300.15 $ Temperature (Kelvin)
END
```

```

METHOD FUNCTION 3
  key = e_diff_coef
  300.15 $ Temperature (Kelvin)
END
METHOD FUNCTION 4
  key = h_diff_coef
  300.15 $ Temperature (Kelvin)
END
METHOD FUNCTION 5
  key = e_lifetime
  model name = consrh
END
METHOD FUNCTION 6
  key = h_lifetime
  model name = consrh
END
METHOD FUNCTION 8
  key = intrinsic concentration
  model name = medici
END
METHOD FUNCTION 9
  key = bandgap
  model name = medici
END

$ setting up the transient source function as a step function over
METHOD FUNCTION 25
  key = transient source
  model name = onedim profile
  fit method = cubic spline
  TIME_FILE_NAME_V0 = "iv_time_profile.txt"
  FILE_NAME_V0 = "iv_depth_profile.txt"
  NMAX_V0 = 1.0000000000e+00
  TIME_FILE_NAME_SII0 = "iv_time_profile.txt"
  FILE_NAME_SII0 = "iv_depth_profile.txt"
  NMAX_SII0 = 1.0000000000e+00
  TIME_FILE_NAME_E = "eh_time_profile.txt"
  FILE_NAME_E = "eh_depth_profile.txt"
  NMAX_E = -1.0
  TIME_FILE_NAME_H = "eh_time_profile.txt"
  FILE_NAME_H = "eh_depth_profile.txt"
  NMAX_H = -1.0
END

$*****

```

```

$ VM data output (must also add output
$ variable names to plot data)
$*****
VARIABLE MANAGER DATA OUTPUT
  "ELECTRON LIFETIME", q, elife, unscale="t0"
  "ELECTRON MOBILITY", q, emob, unscale="u0"
  "ELECTRON DIFFUSION COEFFICIENT", q, ediff, unscale="D0"
  "HOLE LIFETIME", q, hlife, unscale="t0"
  "HOLE MOBILITY", q, hmob, unscale="u0"
  "HOLE DIFFUSION COEFFICIENT", q, hdiff, unscale="D0"
END

$*****
$ Doping specification
$*****
include "doping_analytic_species.inp"

$*****
$ Defect specification
$*****
include "charon_prototype_defects.inp"

END $CHARON

plot variable
  electric_potential
  electron_density
  hole_density
  doping
  $ VM data
  elife
  emob
  ediff
  hlife
  hmob
  hdiff
  vv0
  vvp
  vvm
  vvmm
  v0
  vm
  vmm
  vp
  vpp

```

```

bm
bv0
bvp
b0
pp
pv0
pvm
p0
sii0
siip
siipp
siim
siimm
bi0
bim
bip
bib0
bibm
C
CIO
CIM
CIP
OIO
OIVM
OIVO
OIBIO
OIBIP
end

$$$$$$$$$$$$$$$$$$$$
$ Generic Nevada input
$$$$$$$$$$$$$$$$$$$$
double precision exodus
termination cycle = 1
termination time = 1.0
emit plot, cycle 1

$$$$$$$$$$$$$$$$$$$$
$ Material specs
$$$$$$$$$$$$$$$$$$$$

Material 1 SI
  model 1
end

```



```
MODEL 1 CHARON SEMICONDUCTOR
  MATERIAL NAME = SI
  RELATIVE PERMITTIVITY = 11.8

  ELECTRON MOBILITY = method function 1
  HOLE MOBILITY = method function 2

  ELECTRON DIFFUSION COEFFICIENT = method function 3
  HOLE DIFFUSION COEFFICIENT = method function 4

  ELECTRON LIFETIME = method function 5
  HOLE LIFETIME = method function 6

  DOPING = 0.0
  INITIAL SPECIES DENSITY = method function 15

  DEFECTS = method function 20
  TRANSIENT SOURCE = method function 25

  INTRINSIC CONCENTRATION = method function 8
  BANDGAP = method function 9
END

crt: off
```

A.7 Example include file for doping, doping_analytic_species.inp

```
METHOD FUNCTION 15
  key = initial species density
  model name = function

  $ Collector
  function_1 = uniform
  nmax_1 = 2.02700E+15
  type_1 = bm
  xmin_1 = 0.0E-4
  xmax_1 = 200.0E-4

  $ Subcollector
  function_2 = M_GAUSSIAN
```

```
type_2 = bm
nmax_2 = 1.69088E+18
xmin_2 = 13.25E-04
xmax_2 = 200.0E-4
xchar_2 = 1.8469E-04
ymin_2 = 0.0E-4
ymax_2 = 200.0E-4
ychar_2 = 1.0E-04

$ Emitter 1
function_3 = M_GAUSSIAN
type_3 = bm
nmax_3 = 2.76254E+19
xmax_3 = 1.1412949E-04
xmin_3 = 0.0E-04
xchar_3 = 0.3639141E-04
ymin_3 = 0.0E-4
ymax_3 = 200.0E-4
ychar_3 = 1.0E-04

$ Emitter 2
function_4 = M_GAUSSIAN
type_4 = bm
nmax_4 = 2.77666E+19
xmax_4 = 0.62E-04
xmin_4 = 0.42E-04
xchar_4 = 0.3887082E-04
ymin_4 = 0.0E-4
ymax_4 = 200.0E-4
ychar_4 = 1.0E-04

$ Base
function_5 = M_GAUSSIAN
type_5 = pp
nmax_5 = 0.271421E+19
xmin_5 = -1.4812309E-04
xmax_5 = -1.4812309E-04
xchar_5 = 1.9613315E-04
ymin_5 = 0.0E-4
ymax_5 = 200.0E-4
ychar_5 = 1.0E-04

$Oxygen
function_6 = uniform
type_6 = oi0
```

```
nmax_6 = 1.5e17
xmin_6 = 0.0e-4
xmax_6 = 11.0e-4

$$Carbon
function_7 = uniform
type_7 = c
nmax_7 = 0.7e16
xmin_7 = 0.0e-4
xmax_7 = 11.0e-4

END

exit
```

A.8 Example include file for defect properties, charon_prototype_defects.inp

```
METHOD FUNCTION 20
key = defects
model name = CHEMKIN REACTION
CHEMKIN_FILENAME = "chem_prototype.asc"
ENERGY_A_SIIMM = 0.0
DIFF_PRE_SIIMM = 1.0e-4
ENERGY_A_SIIM = 0.0
DIFF_PRE_SIIM = 1.0e-4
ENERGY_A_SII0 = 0.0
DIFF_PRE_SII0 = 1.0e-4
ENERGY_A_SIIP = 0.0
DIFF_PRE_SIIP = 1.0e-4
ENERGY_A_SIIPP = 0.0
DIFF_PRE_SIIPP = 1.0e-4
ENERGY_A_VMM = 0.18
DIFF_PRE_VMM = 0.00649
ENERGY_A_VM = 0.45
DIFF_PRE_VM = 1.3e-3
ENERGY_A_VO = 0.45
DIFF_PRE_VO = 0.000376
ENERGY_A_VP = 0.32
DIFF_PRE_VP = 9.6e-5
ENERGY_A_VPP = 0.32
```

```
DIFF_PRE_VPP = 9.6e-5
ENERGY_A_VVMM = 0.0
DIFF_PRE_VVMM = 0.0
ENERGY_A_VVM = 0.0
DIFF_PRE_VVM = 0.0
ENERGY_A_VVO = 0.0
DIFF_PRE_VVO = 0.0
ENERGY_A_VVP = 0.0
DIFF_PRE_VVP = 0.0
ENERGY_A_BM = 0.0
DIFF_PRE_BM = 0.0
ENERGY_A_BO = 0.0
DIFF_PRE_BO = 0.0
ENERGY_A_BIM = 0.53
DIFF_PRE_BIM = 2.3e-5
ENERGY_A_BIO = 0.53
DIFF_PRE_BIO = 2.3e-5
ENERGY_A_BIP = 0.53
DIFF_PRE_BIP = 2.3e-5
ENERGY_A_BVO = 0.0
DIFF_PRE_BVO = 0.0
ENERGY_A_BVP = 0.0
DIFF_PRE_BVP = 0.0
ENERGY_A_PO = 0.0
DIFF_PRE_PO = 0.0
ENERGY_A_PP = 0.0
DIFF_PRE_PP = 0.0
ENERGY_A_PVM = 0.0
DIFF_PRE_PVM = 0.0
ENERGY_A_PVO = 0.0
DIFF_PRE_PVO = 0.0
ENERGY_A_OIO = 0.0
DIFF_PRE_OIO = 0.0
ENERGY_A_OIVM = 0.0
DIFF_PRE_OIVM = 0.0
ENERGY_A_OIVO = 0.0
DIFF_PRE_OIVO = 0.0
ENERGY_A_BIBM = 0.0
DIFF_PRE_BIBM = 0.0
ENERGY_A_BIBO = 0.0
DIFF_PRE_BIBO = 0.0
ENERGY_A_OIBIO = 0.0
DIFF_PRE_OIBIO = 0.0
ENERGY_A_OIBIP = 0.0
DIFF_PRE_OIBIP = 0.0
```

```
ENERGY_A_C = 0.0
DIFF_PRE_C = 0.0
ENERGY_A_CIM = 0.0
DIFF_PRE_CIM = 0.0
ENERGY_A_CIO = 0.0
DIFF_PRE_CIO = 0.0
ENERGY_A_CIP = 0.0
DIFF_PRE_CIP = 0.0
END

exit
```

A.9 Example include file for transient source rate, iv_time_profile.txt

```
1.0000E-05 7.1732E+17
2.0000E-05 9.5496E+17
3.0000E-05 1.2850E+18
4.0000E-05 1.7119E+18
6.0000E-05 2.8077E+18
8.0000E-05 4.0135E+18
1.0000E-04 6.6187E+18
1.2000E-04 1.1165E+19
1.4000E-04 1.8694E+19
1.5000E-04 2.3733E+19
1.6000E-04 2.9445E+19
1.7000E-04 3.5479E+19
1.8000E-04 4.1147E+19
1.9000E-04 4.5429E+19
2.0000E-04 4.7269E+19
2.1000E-04 4.5799E+19
2.2000E-04 4.0901E+19
2.3000E-04 3.3406E+19
2.4000E-04 2.4869E+19
2.5000E-04 1.6850E+19
2.6000E-04 1.0447E+19
2.7000E-04 6.0246E+18
2.8000E-04 3.3006E+18
3.0000E-04 1.0034E+18
3.2000E-04 3.8140E+17
3.5000E-04 1.7603E+17
```

```
4.0000E-04 8.8015E+16
4.5000E-04 5.7210E+16
5.0000E-04 4.4008E+16
6.0000E-04 3.0805E+16
8.0000E-04 1.7603E+16
1.0000E-03 0.0000E+00
10.0 0.0
```

A.10 Example include file for source depth distribution, iv_depth_profile.txt

```
0.00E+00      0.00E+00
1.00E-04      0.00E+00
1.20E-04      0.00E+00
1.40E-04      0.00E+00
1.60E-04      0.00E+00
1.80E-04      0.00E+00
1.85E-04      1.00E+00
1.90E-04      1.00E+00
2.00E-04      1.00E+00
2.10E-04      1.00E+00
2.15E-04      1.00E+00
2.20E-04      0.00E+00
2.40E-04      0.00E+00
2.70E-04      0.00E+00
3.00E-04      0.00E+00
5.00E-04      0.00E+00
7.00E-04      0.00E+00
9.00E-04      0.00E+00
```

A.11 Example chemkin input file, chemkin_prototype.inp

```
ELEMENTS E V /1.00/ VF /4.00/ P B C O SI
END
SPECIES
E
H
```

```

SIIMM
SIIM
SIIO
SIIP
SIIPP
VMM
VM
VO
VP
VPP
VVM
VVO
VVP
BM
BO
BIM
BIO
BIP
BVO
BVP
PO
PP
PVM
PVO
PIO
PIVM
PIVO
BIBM
BIBO
OIBIO
OIBIP
C
CIM
CIO
CIP
SI
END
THERMO
VO          J 3/67V  -3VF  100 0000  G   300.000  5000.000          1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14 2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05 3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01          4
VF          J 3/67V   400 0000 0000 00G   300.000  5000.000          1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14 2

```

```

0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05 3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01 4
E      J 3/67E 100 0000 0000 00G 300.000 5000.000 1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14 2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05 3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01 4
H      J 3/67E -100 0000 0000 00G 300.000 5000.000 1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14 2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05 3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01 4
VVO    J 3/67V -6VF 2 0000 0000G 300.000 5000.000 1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14 2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05 3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01 4
VVP    J 3/67V -6VF 2E -100 00G 300.000 5000.000 1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14 2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05 3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01 4
VVM    J 3/67V -6VF 2E 100 00G 300.000 5000.000 1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14 2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05 3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01 4
VPPP   J 3/67V -6VF 2E -200 00G 300.000 5000.000 1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14 2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05 3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01 4
VMMM   J 3/67V -6VF 2E 200 00G 300.000 5000.000 1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14 2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05 3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01 4
VP      J 3/67V -3E -1VF 1 G 300.000 5000.000 1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14 2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05 3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01 4
VM      J 3/67V -3E 1VF 1 G 300.000 5000.000 1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14 2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05 3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01 4
VPP     J 3/67V -3E -2VF 1 00G 300.000 5000.000 1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14 2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05 3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01 4
VMM     J 3/67V -3E 2VF 1 00 G 300.000 5000.000 1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14 2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05 3

```



```

-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01      4
PP      J 3/67P 1E -100 0000 00G 300.000 5000.000                1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14      2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05      3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01      4
P0      J 3/67P 1 00 0000 00G 300.000 5000.000                1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14      2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05      3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01      4
PV0     J 3/67P 1V -3VF 1 G 300.000 5000.000                1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14      2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05      3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01      4
PVM     J 3/67P 1V -3VF 1E 1G 300.000 5000.000                1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14      2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05      3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01      4
BM      J 3/67B 1E 100 0000 00G 300.000 5000.000                1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14      2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05      3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01      4
B0      J 3/67B 1 00 0000 00G 300.000 5000.000                1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14      2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05      3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01      4
BV0     J 3/67B 1V -3VF 1 G 300.000 5000.000                1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14      2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05      3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01      4
BVP     J 3/67B 1V -3VF 1E -1G 300.000 5000.000                1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14      2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05      3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01      4
SII0    J 3/67SI 1VF -1V 300 0G 300.000 5000.000                1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14      2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05      3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01      4
SIIP    J 3/67SI 1VF -1V 3E -1G 300.000 5000.000                1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14      2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05      3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01      4
SIIPP   J 3/67SI 1VF -1V 3E -2G 300.000 5000.000                1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14      2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05      3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01      4

```

```

SIIM          J 3/67SI  1VF -1V  3E  1G  300.000  5000.000      1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14      2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05      3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01      4
SIIMM         J 3/67SI  1VF -1V  3E  2G  300.000  5000.000      1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14      2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05      3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01      4
BIO           J 3/67B  1VF -1V  3 00  G  300.000  5000.000      1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14      2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05      3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01      4
BIM           J 3/67B  1VF -1V  3E  1G  300.000  5000.000      1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14      2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05      3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01      4
BIP           J 3/67B  1VF -1V  3E  -1G 300.000  5000.000      1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14      2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05      3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01      4
BIBO          J 3/67B  2VF -1V  3 00  G  300.000  5000.000      1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14      2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05      3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01      4
BIBM          J 3/67B  2VF -1V  3E  1G  300.000  5000.000      1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14      2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05      3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01      4
CIO           J 3/67C  1VF -1V  300  0G  300.000  5000.000      1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14      2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05      3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01      4
CIP           J 3/67C  1VF -1V  3E  -1G 300.000  5000.000      1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14      2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05      3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01      4
CIM           J 3/67C  1VF -1V  3E  1G  300.000  5000.000      1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14      2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05      3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01      4
OIO           J 3/67D  1VF -1V  300  0G  300.000  5000.000      1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14      2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05      3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01      4
OIVO          J 3/67D  1          G  300.000  5000.000      1

```

```

0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14 2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05 3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01 4
OIVM J 3/670 1E 1 G 300.000 5000.000 1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14 2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05 3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01 4
OIBIO J 3/670 1B 1VF -2V 6G 300.000 5000.000 1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14 2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05 3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01 4
OIBIP J G 300.000 5000.000 1&
O 1B 1VF -2V 6E -1
0.26506014E 01-0.35763852E-03 0.29592293E-06-0.72804829E-10 0.57963329E-14 2
0.53437054E 05 0.52204057E 01 0.31793537E 01-0.27646992E-02 0.44784038E-05 3
-0.32833177E-08 0.91213631E-12 0.53339032E 05 0.27273204E 01 4
END

```

REACTIONS

```

E + SIIM => SIIMM 3e-16 0.0 0.0
SIIMM => E + SIIM 3e-16 0.0 0.109
E + SII0 => SIIM 3e-15 0.0 0.0
SIIM => E + SII0 3e-15 0.0 0.261
E + SIIP => SII0 3e-14 0.0 0.0
SII0 => E + SIIP 3e-14 0.0 0.622
E + SIIPP => SIIP 3e-14 0.0 0.0
SIIP => E + SIIPP 3e-14 0.0 0.466
E + VM => VMM 3e-16 0.0 0.0
VMM => E + VM 3e-16 0.0 0.09
E + V0 => VM 2.89e-14 0.0 0.0
VM => E + V0 2.89e-14 0.0 0.40
E + VP => V0 3e-14 0.0 0.0
V0 => E + VP 3e-14 0.0 1.07
E + VPP => VP 3e-14 0.0 0.0
VP => E + VPP 3e-14 0.0 0.99
E + VVM => VVMM 3e-16 0.0 0.0
VVMM => E + VVM 3e-16 0.0 0.23
E + VVO => VVM 3e-15 0.0 0.0
VVM => E + VVO 3e-15 0.0 0.41
E + VVP => VVO 3e-14 0.0 0.0
VVO => E + VVP 3e-14 0.0 0.91
E + BIO => BIM 3e-15 0.0 0.0
BIM => E + BIO 3e-15 0.0 0.37
E + BIP => BIO 3e-14 0.0 0.0
BIO => E + BIP 3e-14 0.0 0.13

```

E + BVP => BV0	3e-14	0.0	0.0
BV0 => E + BVP	3e-14	0.0	0.62
E + PP => P0	3e-16	0.0	0.0
P0 => E + PP	1.5e-13	0.0	0.045
E + PVO => PVM	1.72e-15	0.0	0.0
PVM => E + PVO	1.72e-15	0.0	0.44
E + OIVO => OIVM	3e-15	0.0	0.0
OIVM => E + OIVO	3e-15	0.0	0.17
E + BIBO => BIBM	9.9e-16	0.0	0.0
BIBM => E + BIBO	9.9e-16	0.0	0.82
E + OIBIP => OIBIO	9.9e-15	0.0	0.0
OIBIO => E + OIBIP	9.9e-15	0.0	0.26
E + CIO => CIM	3e-15	0.0	0.0
CIM => E + CIO	3e-15	0.0	0.10
E + CIP => CIO	3e-14	0.0	0.0
CIO => E + CIP	3e-14	0.0	0.84
H + SIIMM => SIIM	3e-14	0.0	0.0
SIIM => H + SIIMM	3e-14	0.0	1.011
H + SIIM => SII0	3e-14	0.0	0.0
SII0 => H + SIIM	3e-14	0.0	0.859
H + SII0 => SIIP	3e-15	0.0	0.0
SIIP => H + SII0	3e-15	0.0	0.498
H + SIIP => SIIPP	3e-16	0.0	0.0
SIIPP => H + SIIP	3e-16	0.0	0.654
H + VMM => VM	3e-14	0.0	0.0
VM => H + VMM	3e-14	0.0	1.03
H + VM => V0	3.37e-13	0.0	0.0
V0 => H + VM	3.37e-13	0.0	0.72
H + V0 => VP	3e-15	0.0	0.0
VP => H + V0	3e-15	0.0	0.05
H + VP => VPP	3e-16	0.0	0.0
VPP => H + VP	3e-16	0.0	0.13
H + VVMM => VVM	3e-14	0.0	0.0
VVM => H + VVMM	3e-14	0.0	0.89
H + VVM => VV0	3e-14	0.0	0.0
VV0 => H + VVM	3e-14	0.0	0.71
H + VV0 => VVP	3e-15	0.0	0.0
VVP => H + VV0	3e-15	0.0	0.21
H + BM => B0	3e-16	0.0	0.0
B0 => H + BM	7.5e-14	0.0	0.045
H + BIM => BIO	3e-14	0.0	0.0
BIO => H + BIM	3e-14	0.0	0.75
H + BIO => BIP	3e-15	0.0	0.0
BIP => H + BIO	3e-15	0.0	0.99
H + BV0 => BVP	3e-15	0.0	0.0

```

BVP => H + BV0      3e-15      0.0      0.50
H + PVM => PV0      4.9e-14     0.0      0.0
PV0 => H + PVM      4.9e-14     0.0      0.68
H + OIVM => OIVO     3e-14      0.0      0.0
OIVO => H + OIVM     3e-14      0.0      0.95
H + BIBM => BIBO     9.9e-15     0.0      0.0
BIBO => H + BIBM     9.9e-15     0.0      0.30
H + OIBIO => OIBIP    9.9e-16     0.0      0.0
OIBIP => H + OIBIO    9.9e-16     0.0      0.86
H + CIM => CIO       3e-14      0.0      0.0
CIO => H + CIM       3e-14      0.0      1.02
H + CIO => CIP       3e-15      0.0      0.0
CIP => H + CIO       3e-15      0.0      0.28
SIIMM + V0 => SI + 2E      4      0.0      0.0
SIIMM + VP => SI + E       1      0.0      0.0
SIIMM + VPP => SI        1      0.0      0.0
SIIMM + VVO => SI + VMM    1      0.0      0.0
SIIMM + VVP => SI + VM     1      0.0      0.0
SIIMM + B0 => SI + BIM + E   1      0.0      0.0
SIIMM + BV0 => SI + BM + E   1      0.0      0.0
SIIMM + BVP => SI + BM      1      0.0      0.0
SIIMM + PV0 => SI + P0 + 2E  1      0.0      0.0
SIIMM + OIVO => SI + OIO + 2E 1      0.0      0.0
SIIMM + C => SI + CIM + E    1      0.0      0.0
SIIM + V0 => SI + E        4      0.0      0.0
SIIM + VP => SI          1      0.0      0.0
SIIM + VPP => SI + H        1      0.0      0.0
SIIM + VVO => SI + VM       1      0.0      0.0
SIIM + VVP => SI + V0       1      0.0      0.0
SIIM + B0 => SI + BIM       1      0.0      0.0
SIIM + BV0 => SI + BM       1      0.0      0.0
SIIM + BVP => SI + B0       1      0.0      0.0
SIIM + PV0 => SI + P0 + E    1      0.0      0.0
SIIM + OIVO => SI + OIO + E  1      0.0      0.0
SIIM + C => SI + CIM        1      0.0      0.0
SII0 + VMM => SI + 2E       4      0.0      0.0
SII0 + VM => SI + E         4      0.0      0.0
SII0 + V0 => SI           4      0.0      0.0
SII0 + VP => SI + H         4      0.0      0.0
SII0 + VPP => SI + 2H       4      0.0      0.0
SII0 + VVMM => SI + VMM     1      0.0      0.0
SII0 + VVM => SI + VM       1      0.0      0.0
SII0 + VVO => SI + V0       1      0.0      0.0
SII0 + VVP => SI + VP       1      0.0      0.0
SII0 + BM => SI + BIM       1      0.0      0.0

```

```

SII0 + B0 => SI + BIO      1      0.0      0.0
SII0 + BV0 => SI + B0      1      0.0      0.0
SII0 + BVP => SI + B0 + H   1      0.0      0.0
SII0 + PVM => SI + P0 + E   1      0.0      0.0
SII0 + PV0 => SI + P0      1      0.0      0.0
SII0 + OIVM => SI + OIO + E 1      0.0      0.0
SII0 + OIVO => SI + OIO     1      0.0      0.0
SII0 + C => SI + CIO        1      0.0      0.0
SIIP + VMM => SI + E        1      0.0      0.0
SIIP + VM => SI             1      0.0      0.0
SIIP + V0 => SI + H         4      0.0      0.0
SIIP + VVMM => SI + VM      1      0.0      0.0
SIIP + VVM => SI + V0       1      0.0      0.0
SIIP + VVO => SI + VP       1      0.0      0.0
SIIP + BM => SI + BIO       1      0.0      0.0
SIIP + B0 => SI + BIP       1      0.0      0.0
SIIP + BV0 => SI + B0 + H   1      0.0      0.0
SIIP + PVM => SI + P0       1      0.0      0.0
SIIP + PV0 => SI + PP       1      0.0      0.0
SIIP + OIVM => SI + OIO     1      0.0      0.0
SIIP + OIVO => SI + OIO + H 1      0.0      0.0
SIIP + C => SI + CIP        1      0.0      0.0
SIIPP + VMM => SI          1      0.0      0.0
SIIPP + VM => SI + H        1      0.0      0.0
SIIPP + V0 => SI + 2H       4      0.0      0.0
SIIPP + VVMM => SI + V0     1      0.0      0.0
SIIPP + VVM => SI + VP      1      0.0      0.0
SIIPP + VVO => SI + VPP     1      0.0      0.0
SIIPP + BM => SI + BIP      1      0.0      0.0
SIIPP + B0 => SI + BIP + H   1      0.0      0.0
SIIPP + BV0 => SI + B0 + 2H 1      0.0      0.0
SIIPP + PVM => SI + PP      1      0.0      0.0
SIIPP + PV0 => SI + PP + H   1      0.0      0.0
SIIPP + OIVM => SI + OIO + H 1      0.0      0.0
SIIPP + OIVO => SI + OIO + 2H 1      0.0      0.0
SIIPP + C => SI + CIP + H   1      0.0      0.0
VMM + V0 => VVMM            1      0.0      0.0
VMM + VP => VVM             1      0.0      0.0
VMM + VPP => VVO            1      0.0      0.0
VMM + BIO => BM + E         4      0.0      0.0
VMM + BIP => BM             1      0.0      0.0
VMM + PP => PVM             1      0.0      0.0
VMM + OIO => OIVM + E       1      0.0      0.0
VMM + CIO => C + 2E         4      0.0      0.0
VMM + CIP => C + E          1      0.0      0.0

```

```

VM + VO => VVM      1      0.0      0.0
VM + VP => VVO      1      0.0      0.0
VM + VPP => VVP     1      0.0      0.0
VM + BIO => BM       4      0.0      0.0
VM + BIP => BO       1      0.0      0.0
VM + PO => PVM       1      0.0      0.0
VM + PP => PVO       1      0.0      0.0
VM + OIO => OIVM     1      0.0      0.0
VM + CIO => C + E    4      0.0      0.0
VM + CIP => C        1      0.0      0.0
VO + VO => VVO      1      0.0      0.0
VO + VP => VVP     1      0.0      0.0
VO + VPP => VVP + H 1      0.0      0.0
VO + BO => BVO      1      0.0      0.0
BVO => VO + BO      1      0.0      1.0
VO + BIM => BM       4      0.0      0.0
VO + BIO => BO       4      0.0      0.0
VO + BIP => BO + H   4      0.0      0.0
VO + PO => PVO      1      0.0      0.0
VO + OIO => OIVO     1      0.0      0.0
VO + CIM => C + E    4      0.0      0.0
VO + CIO => C        4      0.0      0.0
VO + CIP => C + H    4      0.0      0.0
VP + BM => BVO      1      0.0      0.0
BVO => VP + BM      1      0.0      1.0
VP + BO => BVP      1      0.0      0.0
BVP => VP + BO      1      0.0      1.0
VP + BIM => BO       1      0.0      0.0
VP + BIO => BO + H   4      0.0      0.0
VP + OIO => OIVO + H 1      0.0      0.0
VP + CIM => C        1      0.0      0.0
VP + CIO => C + H    4      0.0      0.0
VPP + BM => BVP      1      0.0      0.0
BVP => VPP + BM     1      0.0      1.0
VPP + BIM => BO + H   1      0.0      0.0
VPP + BIO => BO + 2H  4      0.0      0.0
VPP + OIO => OIVO + 2H 1      0.0      0.0
VPP + CIM => C + H    1      0.0      0.0
VPP + CIO => C + 2H   4      0.0      0.0
VVMM + BIO => BVO + 2E 1      0.0      0.0
VVMM + BIP => BVO + E 1      0.0      0.0
VVM + BIO => BVO + E 1      0.0      0.0
VVM + BIP => BVO     1      0.0      0.0
VVO + BIM => BVO + E 1      0.0      0.0
VVO + BIO => BVO     1      0.0      0.0

```

A.12 Input file for nonlinear energy simulation for PN-diode problem

192


```
        output current components
    end

    report fill status = false

    $ generally semiconductor problems need to be scaled.
    scaled variables = true
    scaled mesh = max_x

    $ NOX parameters
    nox parameters

        wrms absolute tolerance = 1.0e-12
        maximum nonlinear iterations = 100
        wrms relative tolerance = 1.0e-4
        linear scaling = row sum

        sublist "Direction"
            "Method" = "Newton"
            sublist "Newton"
                "Rescue Bad Newton Solve" = "true"
                sublist "Linear Solver"
                    "Aztec Solver" = "GMRES"
                "Max Iterations" = 500
                "Tolerance" = 1.0e-12
                "Output Frequency" = 20
                "Aztec Preconditioner" = "ilut"
                "Preconditioner" = "Aztec00"
                "Preconditioner Operator" = "Use Jacobian"
            END
        END
    END

end

$ A scale factor for the named Charon_FillOperators in the
$ specific physics module used
operator scaling, n_ss_stab_term=0.0
operator scaling, p_ss_stab_term=0.0

$ Element block to material map
block 1
    material 1
    physics 1
```

```
end

jacobian = ad

initial guess = true

$ Nevada time parameters
time step scale = 1.0
constant time step = 1.0
gradual startup factor = 1.0

$ Defines a function implemented as a C++ method.
$ This test is for electron mobility
METHOD FUNCTION 1
  key = e_mobility
  model name = tempmob
  300.15 $ Temperature (Kelvin)
END
METHOD FUNCTION 2
  key = h_mobility
  model name = tempmob
  300.15 $ Temperature (Kelvin)
END
METHOD FUNCTION 3
  key = e_diff_coef
  300.15 $ Temperature (Kelvin)
END
METHOD FUNCTION 4
  key = h_diff_coef
  300.15 $ Temperature (Kelvin)
END

$ Doping specification
METHOD FUNCTION 5
  key = doping
  model name = step junction
  junction location = 0.5e-4
  Na = 1.0e16
  Nd = 1.0e16
  configuration = pn
END

$ Electron lifetime specification
METHOD FUNCTION 6
  key = e_lifetime
```

```
    model name = consrh
END

$ Hole lifetime specification
METHOD FUNCTION 7
    key = h_lifetime
    model name = consrh
END

METHOD FUNCTION 8
    key = electron relaxation
    model name = carrtemp
END

METHOD FUNCTION 9
    key = hole relaxation
    model name = carrtemp
END

METHOD FUNCTION 10
    key = specific heat
END

METHOD FUNCTION 11
    key = thermal conductivity
END

END $CHARON

plot variable
    electric_potential
    electron_density
    hole_density
    electron_temperature
    hole_temperature
    lattice_temperature
    doping
end

$$$$$$$$$$$$$$$$$$$$
$ Generic Nevada input
$$$$$$$$$$$$$$$$$$$$
double precision exodus
termination cycle = 1
emit plot, cycle 1
```

```

$$$$$$$$$$$$$$$$
$ Material specs
$$$$$$$$$$$$$$$$
Material 1 SI
  model 1
end

MODEL 1 CHARON SEMICONDUCTOR
  MATERIAL NAME = SI
  RELATIVE PERMITTIVITY = 11.8
  ELECTRON MOBILITY = method function 1
  HOLE MOBILITY = method function 2
  ELECTRON DIFFUSION COEFFICIENT = method function 3
  HOLE DIFFUSION COEFFICIENT = method function 4
  DOPING = method function 5
  ELECTRON LIFETIME = method function 6
  HOLE LIFETIME = method function 7
  ELECTRON RELAXATION = method function 8
  HOLE RELAXATION = method function 9
  SPECIFIC HEAT = method function 10
  THERMAL CONDUCTIVITY = method function 11
END

crt: off

```

A.13 Input file for energy transport simulation for PN-diode problem

```

$-----
$ ID: diode et
$ Title: CHARON: Test of semiconductor energytransport model on
$               a simple diode
$-----

title
  CHARON: example of energytransport in a simple diode

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ Charon-specific input section
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

```

```
CHARON

    debug parameter, nodal doping = 1

$  debug level = 1

$ Physics setup
physics = 1, semiconductor, energytransport

    contact 1
        sideset id = 1
    end

    contact 2
        sideset id = 2
    end

$    enable mobility derivative terms = true
    current linear solver tolerance = 1.0e-10

    use current convergence in nox = true
    current wrms relative tolerance = 1.0e-6
    current wrms absolute tolerance = 1.0e-12
    current calculation verbose = false

supg tau calculation = tanh
end

lump, source = false
stabilization options
    supg = true
end

$ generally semiconductor problems need to be scaled.
scaled variables = true
scaled mesh = max_x

$ NOX parameters
nox parameters

    use relative tolerance test = true
    use absolute tolerance test = false
    absolute tolerance = 1.0e-6
```

```

relative tolerance = 1.0e-5
wrms absolute tolerance = 1.0e-6
maximum nonlinear iterations = 70
wrms relative tolerance = 1.0e-2
wrms alpha = 0.0
wrms beta = 1.0
linear scaling = row sum

print solution statistics = true

include "line-search.inp"

sublist "Direction"
  "Method" = "Newton"
  sublist "Newton"
    "Rescue Bad Newton Solve" = "true"
    sublist "Linear Solver"
      "Aztec Solver" = "GMRES"
    "Max Iterations" = 800
    "Tolerance" = 1.0e-06
    "Output Frequency" = 20
    "Preconditioner" = "New Ifpack"
    sublist "Ifpack"
      "schwarz: reordering type" = "rcm"
      "fact: level-of-fill" = 3
    end
  end
END
END
END

end
$ A scale factor for the named Charon_FillOperators in the
$ specific physics module used
$ operator scaling, Tn_ss_stab_term=0.0
$ operator scaling, Tp_ss_stab_term=0.0
$ operator scaling, Tn_ss_stab_conv_term = 0.0
$ operator scaling, Tp_ss_stab_conv_term = 0.0

$ Element block to material map
block 1
  material 1
  physics 1
end

```

```
jacobian = ad

initial guess = exodus
  file name = "diode.nle.exo"
  time index = 2
END

restart variable list
  electric_potential
  electron_density
  hole_density
  electron_temperature
  hole_temperature
  lattice_temperature
END

$ Applied potential. For reverse bias apply positive potential to
$ sideset 2
$ dirichlet bc, electron_temperature, sideset 2 300.15
$ dirichlet bc, hole_temperature, sideset 2 300.15
$ dirichlet bc, lattice_temperature, sideset 2 300.15

$ Applied potential. For forward bias apply positive potential to
$ sideset 1
$ dirichlet bc, electron_temperature, sideset 1 300.15
$ dirichlet bc, hole_temperature, sideset 1 300.15
$ dirichlet bc, lattice_temperature, sideset 1 300.15

$ Nevada time parameters
time step scale = 1.0
constant time step = 1.0
gradual startup factor = 1.0

$ Defines a function implemented as a C++ method.
$ This test is for electron mobility
METHOD FUNCTION 1
  key = e_mobility
  model name = tempmob
  300.15 $ Temperature (Kelvin)
END
METHOD FUNCTION 2
  key = h_mobility
  model name = tempmob
  300.15 $ Temperature (Kelvin)
END
```

```
METHOD FUNCTION 3
  key = e_diff_coef
  300.15 $ Temperature (Kelvin)
END
METHOD FUNCTION 4
  key = h_diff_coef
  300.15 $ Temperature (Kelvin)
END

$ Doping specification
METHOD FUNCTION 5
  key = doping
  model name = step junction
  junction location = 0.5e-4
  Na = 1.0e16
  Nd = 1.0e16
  configuration = pn
END

$ Electron lifetime specification
METHOD FUNCTION 6
  key = e_lifetime
  model name = consrh
END

$ Hole lifetime specification
METHOD FUNCTION 7
  key = h_lifetime
  model name = consrh
END

METHOD FUNCTION 8
  key = electron relaxation
  model name = carrtemp
END

METHOD FUNCTION 9
  key = hole relaxation
  model name = carrtemp
END

METHOD FUNCTION 10
  key = specific heat
END
```



```
METHOD FUNCTION 11
    key = thermal conductivity
END

END $CHARON

plot variable
    electric_potential
    electron_density
    hole_density
    electron_temperature
    hole_temperature
    lattice_temperature
    doping
end

$$$$$$$$$$$$$$$$$$$$
$ Generic Nevada input
$$$$$$$$$$$$$$$$$$$$
double precision exodus
termination cycle = 1
emit plot, cycle 1

$$$$$$$$$$$$$$$$$$$$
$ Material specs
$$$$$$$$$$$$$$$$$$$$
Material 1 SI
    model 1
end

MODEL 1 CHARON SEMICONDUCTOR
MATERIAL NAME = SI
RELATIVE PERMITTIVITY = 11.8 $$ Not really used by charon
ELECTRON MOBILITY = method function 1
HOLE MOBILITY = method function 2
ELECTRON DIFFUSION COEFFICIENT = method function 3
HOLE DIFFUSION COEFFICIENT = method function 4
DOPING = method function 5
ELECTRON LIFETIME = method function 6
HOLE LIFETIME = method function 7
ELECTRON RELAXATION = 2.0e-13
HOLE RELAXATION = 2.0e-13
SPECIFIC HEAT = 0.7
THERMAL CONDUCTIVITY = 1.5
```

END

crt: off

DRAFT

B. Defect Reactions Implemented in Charon

This appendix covers the reaction set implemented in **Charon**.

B.1 Electron Capture

Also see Section 1.3.

Symbolic reaction:

Generic source term:

$$R(X_j^{m+1}, e^-) = \sigma(X_j^{m+1}, e^-) \nu_n n X_j^{m+1} \quad (\text{B.2})$$

Source terms implemented in **Charon**:

$$R(X_j^{m+1}) = -R(X_j^{m+1}, e^-) \quad (\text{B.3})$$

$$R(e^-) = -R(X_j^{m+1}, e^-) \quad (\text{B.4})$$

$$R(X_i^m) = R(X_j^{m+1}, e^-) \quad (\text{B.5})$$

where

- $\sigma(X_j^{m+1}, e^-)$ – electron capture cross-section for species X_j^{m+1}
- ν_n – electron thermal velocity within material (for Si = 2.3×10^7 cm/s @ 300K)

B.2 Electron Emission

Also see Section 1.3.

Symbolic reaction:

Generic source term:

$$R(X_j^{m+1} \mapsto e^-) = \sigma(X_j^{m+1}, e^-) \nu_n N_c X_i^m \frac{\gamma(X_j^{m+1})}{\gamma(X_i^m)} \exp\left(\frac{E_{X_i^m} - E_c}{kT}\right) \quad (\text{B.7})$$

Source terms implemented in **Charon**:

$$R(X_j^{m+1}) = -R(X_j^{m+1} \mapsto e^-) \quad (\text{B.8})$$

$$R(e^-) = R(X_j^{m+1} \mapsto e^-) \quad (\text{B.9})$$

$$R(X_i^m) = R(X_j^{m+1} \mapsto e^-) \quad (\text{B.10})$$

where

- N_c – effective density of states in conduction band (for Si = $2.86 \times 10^{19} \text{ cm}^{-3}$ @ 300K)
- $\gamma(X_j^{m+1}), \gamma(X_i^m)$ – state degeneracies for X_j^{m+1} and X_i^m respectively (these values are typically 1.0 for the majority of defect species)
- $E_{X_i^m}$ – energy of the electron state associated with X_i^m
- E_c – conduction band minimum energy
- $E_{X_i^m} - E_c \equiv \Delta E_e^{X_i^m}$ – activation energy for electron emission of species X_i^m

B.3 Hole Capture

Symbolic reaction:

Generic source term:

$$R(X_i^m, h^+) = \sigma(X_i^m, h^+) \nu_p X_i^m \quad (\text{B.12})$$

Source terms implemented in **Charon**:

$$R(X_i^m) = -R(X_i^m, h^+) \quad (\text{B.13})$$

$$R(h^+) = -R(X_i^m, h^+) \quad (\text{B.14})$$

$$R(X_j^{m+1}) = R(X_i^m, h^+) \quad (\text{B.15})$$

where

- $\sigma(X_i^m, h^+)$ – hole capture cross-section for species X_i^m ,
- ν_p – electron thermal velocity within material (for Si = 1.9×10^7 cm/s @ 300K)

B.4 Hole Emission

Symbolic reaction:

Generic source term:

$$R(X_j^{m+1} \mapsto h^+) = \sigma(X_i^m, h^+) \nu_p N_v X_j^{m+1} \frac{\gamma(X_i^m)}{\gamma(X_j^{m+1})} \exp\left(\frac{E_v - E_{X_j^{m+1}}}{kT}\right) \quad (\text{B.17})$$

Source terms implemented in **Charon**:

$$R(X_j^{m+1}) = -R(X_j^{m+1} \mapsto h^+) \quad (\text{B.18})$$

$$R(h^+) = R(X_j^{m+1} \mapsto h^+) \quad (\text{B.19})$$

$$R(X_i^m) = R(X_j^{m+1} \mapsto h^+) \quad (\text{B.20})$$

where

- N_v – effective density of states in valence band (for Si = 2.66×10^{19} cm⁻³ @ 300K),
- $\sigma(X_j^{m+1})$, $\gamma(X_i^m)$ – state degeneracies for X_j^{m+1} and X_i^m , respectively (these values are typically 1.0 for the majority of defect species),
- $E_{X_j^{m+1}}$ – energy of the hole state associated with X_j^{m+1} ,
- E_v – valence band maximum energy,
- $E_v - E_{X_j^{m+1}} \equiv \Delta E_h^{X_j^{m+1}}$ – activation energy for hole emission of species X_j^{m+1} .

B.5 Complex Recombination Charge-Charge

Symbolic reaction:

Generic source term:

$$R(X_i^m, X_j^n) = 4\pi r_c [D(X_i^m) + D(X_j^n)] X_i^m X_j^n \quad (\text{B.22})$$

Source terms implemented in **Charon**:

$$R(X_i^m) = -R(X_i^m, X_j^n) \quad (\text{B.23})$$

$$R(X_j^n) = -R(X_i^m, X_j^n) \quad (\text{B.24})$$

$$R(X_i X_j^{m+n}) = R(X_i^m, X_j^n) \quad (\text{B.25})$$

where

- $D(X_i^m)$ – diffusion coefficient of species X_i^m ,
- $D(X_j^n)$ – diffusion coefficient of species X_j^n ,
- r_c – the reaction distance based on when the Coulumbic attraction is equivalent to kT and can be expressed as

$$r_c = \frac{|m \cdot n| (1.40 \times 10^{-4} \text{ cm} \cdot \text{K})}{T}$$

Note that this only includes reactions where $m \cdot n < 0$, i.e., species which are oppositely charged.

B.6 Complex Dissociation Charge-Charge

Symbolic reaction:

Generic source term:

$$R(X_i X_j^{m+n} \mapsto) = 4\pi r_c [D(X_i^m) + D(X_j^n)] \langle \text{Si} \rangle X_i X_j^{m+n} \frac{\gamma(X_i^m) \gamma(X_j^n)}{\gamma(X_i X_j^{m+n})} \exp\left(\frac{-\Delta E(X_i X_j^{m+n})}{kT}\right) \quad (\text{B.27})$$

Source terms implemented in **Charon**:

$$R(X_i^m) = R(X_i X_j^{m+n} \mapsto) \quad (\text{B.28})$$

$$R(X_j^n) = R(X_i X_j^{m+n} \mapsto) \quad (\text{B.29})$$

$$R(X_i X_j^{m+n}) = -R(X_i X_j^{m+n} \mapsto) \quad (\text{B.30})$$

where

- $\langle \text{Si} \rangle$ – the number of Si atoms per unit volume,
- $\gamma(X_i^m)$, $\gamma(X_j^n)$, $\gamma(X_i X_j^{m+n})$ – state degeneracies for X_i^m , X_j^n and $X_i X_j^{m+n}$ respectively (these values are typically 1.0 for the majority of defect species)
- $\Delta E(X_i X_j^{m+n})$ – binding energy of species $X_i X_j^{m+n}$

B.7 Complex Recombination Charge-Neutral

Symbolic reaction:

Generic source term:

$$R(X_i^0, X_j^n) = 4\pi r_l [D(X_i^0) + D(X_j^n)] X_i^0 X_j^n \quad (\text{B.32})$$

Source terms implemented in **Charon**:

$$R(X_i^0) = -R(X_i^0, X_j^n) \quad (\text{B.33})$$

$$R(X_j^n) = -R(X_i^0, X_j^n) \quad (\text{B.34})$$

$$R(X_i X_j^n) = R(X_i^0, X_j^n) \quad (\text{B.35})$$

where

- r_l – approximately the order of the lattice parameter (Si: 5×10^{-8} cm)
- X_i^0 – any neutral defect species

B.8 Complex Dissociation Charge-Neutral

Symbolic reaction:

Generic source term:

$$R(X_i X_j^n \mapsto) = 4\pi r_l [D(X_i^0) + D(X_j^n)] \langle \text{Si} \rangle X_i X_j^n \frac{\gamma(X_i^0) \gamma(X_j^n)}{\gamma(X_i X_j^n)} \exp\left(\frac{-\Delta E(X_i X_j^n)}{kT}\right) \quad (\text{B.37})$$

Source terms implemented in **Charon**:

$$R(X_i^0) = R(X_i X_j^n \mapsto) \quad (\text{B.38})$$

$$R(X_j^n) = R(X_i X_j^n \mapsto) \quad (\text{B.39})$$

$$R(X_i X_j^n) = -R(X_i X_j^n \mapsto) \quad (\text{B.40})$$

B.9 Complex Recombination Neutral-Neutral

This is currently implemented as a charge-neutral reaction described in Section B.7.

B.10 Complex Dissociation Neutral-Neutral

This is currently implemented as a charge-neutral reaction described in Section B.7.

B.11 Annihilation Charge-Charge

Symbolic reaction:

Generic source term:

$$R(X_i^m, V^n) = 4\pi r_c [D(X_i^m) + D(V^n)] X_i^m V^n \quad (\text{B.42})$$

Source terms implemented in **Charon**:

$$R(X_i^m) = -R(X_i^m, V^n) \quad (\text{B.43})$$

$$R(V^n) = -R(X_i^m, V^n) \quad (\text{B.44})$$

where

- V^n – charged vacancy
- $\langle X \rangle$ – lattice material, Si for example

B.12 Annihilation Neutral-Neutral

Symbolic reaction:

$$X_i^0 + V^0 \longrightarrow \langle X \rangle \quad (\text{B.45})$$

Generic source term:

$$R(X_i^0, V^0) = 4\pi r_l [D(X_i^0) + D(V^0)] X_i^0 V^0 \quad (\text{B.46})$$

Source terms implemented in **Charon**:

$$R(X_i^0) = -R(X_i^0, V^0) \quad (\text{B.47})$$

$$R(V^0) = -R(X_i^0, V^0) \quad (\text{B.48})$$

where

- V^0 – neutral vacancy

B.13 Annihilation Charge-Charge Create Electron

Symbolic reaction:

$$X_i^m + V^n \longrightarrow \langle X \rangle + e^- \quad \text{where } m + n = -1 \text{ and } m \cdot n < 0 \quad (\text{B.49})$$

Generic source term:

$$R(X_i^m, V^n) = 4\pi r_c [D(X_i^m) + D(V^n)] X_i^m V^n \quad (\text{B.50})$$

Source terms implemented in **Charon**:

$$R(X_i^m) = -R(X_i^m, V^n) \quad (\text{B.51})$$

$$R(V^n) = -R(X_i^m, V^n) \quad (\text{B.52})$$

$$R(e^-) = R(X_i^m, V^n) \quad (\text{B.53})$$

B.14 Annihilation Charge-Neutral Create Electron

Symbolic reaction:

Generic source term:

$$R(X_i^m, V^n) = 4\pi r_l [D(X_i^m) + D(V^n)] X_i^m V^n \quad (\text{B.55})$$

Source terms implemented in **Charon**:

$$R(X_i^m) = -R(X_i^m, V^n) \quad (\text{B.56})$$

$$R(V^n) = -R(X_i^m, V^n) \quad (\text{B.57})$$

$$R(e^-) = R(X_i^m, V^n) \quad (\text{B.58})$$

B.15 Annihilation Charge-Charge Create Hole

Symbolic reaction:

Generic source term:

$$R(X_i^m, V^n) = 4\pi r_c [D(X_i^m) + D(V^n)] X_i^m V^n \quad (\text{B.60})$$

Source terms implemented in **Charon**:

$$R(X_i^m) = -R(X_i^m, V^n) \quad (\text{B.61})$$

$$R(V^n) = -R(X_i^m, V^n) \quad (\text{B.62})$$

$$R(h^+) = R(X_i^m, V^n) \quad (\text{B.63})$$

B.16 Annihilation Charge-Neutral Create Hole

Symbolic reaction:

$$X_i^m + V^n \longrightarrow \langle X \rangle + h^+ \quad \text{where } m + n = 1 \text{ and } m \cdot n = 0 \quad (\text{B.64})$$

Generic source term:

$$R(X_i^m, V^n) = 4\pi r_l [D(X_i^m) + D(V^n)] X_i^m V^n \quad (\text{B.65})$$

Source terms implemented in **Charon**:

$$R(X_i^m) = -R(X_i^m, V^n) \quad (\text{B.66})$$

$$R(V^n) = -R(X_i^m, V^n) \quad (\text{B.67})$$

$$R(h^+) = R(X_i^m, V^n) \quad (\text{B.68})$$

This page is left intentionally blank

Bibliography

- [1] Kevin M. Kramer and W. Nicholas G. Hitchon. *Semiconductor Devices A Simulation Approach*. Prentice Hall PTR, Upper Saddle River, NJ, 1997.
- [2] The Exodus II file format. Internet URL http://analyst.ran.sandia.gov/index.php?option=com_content&task=view&id=122&Itemid=72.
- [3] The Cubit tool suite. Internet URL <http://cubit.sandia.gov>.
- [4] SNL ensight website. Internet URL <http://nestor.ran.sandia.gov/ensight>.
- [5] SNL paraview website. Internet URL <http://www.ran.sandia.gov/paraview>.
- [6] Synopsys. *Medici Two-Dimensional Device Simulation Program User Manual*, 2003. Version 2002.4.
- [7] J. W. Slotboom. The *pn*-product in silicon. *Solid-State Electronics*, 20:279–283, 1977.
- [8] The Trilinos Project website. Internet URL <http://trilinos.sandia.gov/>.
- [9] M. Heroux, R. Bartlett, V. Howle, R. Hoekstra, J. Hu, T. Kolda, R. Lehoucq, K. Long, R. Pawlowski, E. Phipps, A. Salinger, H. Thornquist, R. Tuminaro, J. Willenbring, and A. Williams. An Overview of Trilinos. Technical Report SAND2003-2927, Sandia National Laboratories, 2003.
- [10] SNL chemkin website. Internet URL <http://www.ca.sandia.gov/chemkin>.
- [11] *Reaction Design* chemkin website. Internet URL <http://www.reactiondesign.com/products/open/chemkin.html>.

This page is left intentionally blank

Index

B	
box method	see discretizations, box method
C	
Cubit	137
journal file	138, 139, 142, 144
D	
defect properties	
diffusion	17
mobility	17
defect reactions	
annihilation	
charge-charge	208
charge-charge create electron	209
charge-charge create hole	210
charge-neutral create electron	210
charge-neutral create hole	211
neutral-neutral	209
complex dissociation	
charge-charge	206
charge-neutral	208
neutral-neutral	208
complex recombination	
charge-charge	205
charge-neutral	207
neutral-neutral	208
electron capture	18, 203
electron emission	18, 203
hole capture	204
hole emission	205
discretizations	
box method	19
finite element	19
doping	
acceptor	16
donor	16
net	16
E	
Ensignt	141
equation(s)	
drift-diffusion	15
drift-diffusion with defects	16
nonlinear Poisson	16
Poisson with defects	18
Exodus II	see file formats, Exodus II
F	
file formats	
Exodus II	29, 138
I	
input file parameters	
Nevada Input Description	109
ANALYTIC SOLUTION	124
BANDGAP NARROWING	58
BOUNDARY CONDITIONS Description	114
CALCULATE VELOCITIES	59
CODE TIMING	114
Constant current boundary condition	106
Constant electric field BC	100
CONTACT SERIES RESISTANCE	107
CURRENT CONVERGENCE	56
DEBUG Description	110
Default MPARAM Values	125
DEFECTS	68
DIFFUSION	64
DOPING	69
erfc doping	75
erfc initial species density	87
FEM Oxide Electric Field BC	102
gaussian doping	76
gaussian initial species density	88
INCOMPLETE IONIZATION	109
Initial Species Density	84

- Intrinsic concentration 105
 JACOBIAN and FILL STATUS..... 113
 linearly-graded doping 82
 MATERIAL..... 37
 Medici-style gaussian doping 78
 Medici-style gaussian initial species density 90
 MESH REFINEMENT 54
 METAL 39
 MOBILITY 62
 MODEL..... 38
 MOLE FRACTION 94
 MPARAM 40
 Analytic mobility model..... 44
 Arora mobility model 43
 Carrier-carrier scattering mobility model 45
 III–V Analytic mobility model 44
 Impact ionization model 51
 Inversion and accumulation layer mobility model 46
 Lombardi surface mobility model .. 47
 Lucent mobility model 50
 Philips mobility model 49
 OPERATOR SCALING 62
 PHYSICS Description..... 51
 polygon doping 80
 polygon initial species density 91
 REGION 53
 SCALING PARAMETERS 53
 SGFVM Oxide Electric Field BC ... 103
 Silicon/Oxide Interface BC 101
 SOLVER..... 122
 SPECIES SOURCE..... 52
 SRH surface BC 104
 STABILIZATION..... 56
 STABILIZATION, DISCONTINUITY and LUMPING 111
 SURFACE CHARGE..... 59
 TIME INTEGRATION 119
 TRANSIENT SOURCE..... 95
 TRAPS 65
 uniform doping 74
 uniform initial species density 86
 VARIABLE MANAGER DATA OUTPUT 117
 yjunction doping 81
 yjunction initial species density 93
 iv-generate.pl *see* scripts

M
 Message Passing Interface 23, 141
 MPI *see* Message Passing Interface

N
 Nevada toolkit 21, 35

O
 OpenMPI 21

P
 Paraview 141

R
 runCharon *see* scripts

S
 Sandia National Laboratories 21, 138
 scripts
 iv-generate.pl 138, 147
 options 141
 runCharon 29, 30
 options 34
 SNL *see* Sandia National Laboratories

T
 third-party libraries
 building 24
 toolset 23
 tpl *see* third-party libraries