Epi/MatPar User Guide

April 2008

SYNOPSYS®

Contents

About this manual

This user guide describes two new utilities, Epi and MatPar, and how they are integrated into the TCAD Sentaurus tool suite. Both Epi and MatPar, with other TCAD tools, are useful for simulating compound semiconductor devices based on epitaxial multilayer stacks. Such structures are typically grown using epitaxial growth techniques such as metal-organic chemical vapor deposition or molecular beam epitaxy. Both Epi and MatPar facilitate structure generation and material parameter file handling for epitaxial multilayer stacks.

A typical epitaxial structure consists of multiple layers. The approach to structure generation by Sentaurus Structure Editor is to use several Scheme commands to generate each layer, to dope each layer, and to perform mesh refinement repeatedly. Epi avoids this: From a description of the structure in a comma-separated value (CSV) file, Epi automatically generates the required Scheme commands.

The approach for handling material parameters of ternary and quaternary materials in Sentaurus Device involves using a piecewise cubic spline interpolation scheme. The material parameters for ternaries and quaternaries depend on the x– and y–mole fractions, and are expressed as analytic equations in the literature. To create parameter files using these equations, the equations must be converted to a piecewise cubic spline format. Better handling of these parameters is obtained by directly using the analytic equations to compute the material parameters. This is accomplished using MatPar.

The Material Parameter Framework is a tool command language (Tcl)–based framework that efficiently implements new models for all of the material properties. The new models are implemented in terms of the built-in default models of Sentaurus Device. Using these models, MatPar can easily create customized material parameter files.

There is no unique set of material parameters for any particular property of a particular compound semiconductor. Typically, you would perform an extensive literature survey to obtain information about the material parameters for compounds. The Material Parameter Database provides a collection of material parameter values for several materials. It has been implemented so that it can also be used to visualize the variation of material properties as a function of doping concentration, mole fraction, and temperature using the Material Parameter Plotter.

Features of Epi

With Sentaurus Structure Editor, Epi can create epitaxial multilayer stacks. Process simulation is not involved in structure creation. Epi has the following features:

- The properties of the multilayer stack are specified in a CSV file that can be easily edited, for example, in a spreadsheet application. Epi uses this information to automatically generate a Scheme script file that contains the necessary commands for generating the structure using Sentaurus Mesh. This saves effort in creating the Scheme script file for Sentaurus Structure Editor.
- Sentaurus Structure Editor creates the complete device structure by processing the multilayer stack created by Epi. Additional processing steps such as patterning, layer deposition, and contact definitions can be implemented.
- Both two-dimensional and three-dimensional multilayer stacks can be created.
- The epitaxial layers can have constant or graded doping and mole-fraction profiles.
- The source parameter files used to generate user-defined material parameter files for use in Sentaurus Device are specified in the CSV file. Both regionwise and materialwise parameter files can be generated.
- Mesh refinement strategy can be specified in the CSV file.
- The extension column in the CSV file can contain arbitrary text. The column content can be written automatically into the Sentaurus Device command file. This is useful for inserting the same regionwise `Physics` statements for several regions in a multiple quantum-well structure.

Features of MatPar

MatPar extends the capabilities of Sentaurus Device by providing a framework for implementing user-defined models and creating user-defined material parameter files using these models. MatPar has the following features:

- It consists of the Material Parameter Framework, Material Parameter Database, and Material Parameter Plotter.
- The Material Parameter Framework is a general Tcl-based framework for implementing user-defined models and creating user-defined material parameter files.
- Using this framework, material property dependencies on physical variables such as temperature, doping, mole fraction, and wavelength can be easily defined using Tcl expressions. Equation-based and table-based models can be implemented.
- The Material Parameter Database is a collection of material property models and model parameter values for several compound semiconductor materials. It consists of Sentaurus Device models with Sentaurus Device default parameter values as well as new models.

- For each of the implemented models, parameter sets from different references from the literature have been implemented. The Material Parameter Database provides good initial parameter values that may serve as a baseline for custom calibrations.
- The Material Parameter Database is open and extensible. The model files are Tcl script files. You can read the model files and modify the files. These can be used as template files for implementing new models or different model parameter sets for existing models.
- The Material Parameter Plotter, which is an Inspect-based utility, visualizes the variation of material properties as a function of doping, temperature, mole fraction, wavelength, and so on. It can also be used to compare a particular material property value for different models in the Material Parameter Database.

Typographic conventions

Convention	Explanation
Blue text	Identifies a cross-reference (only on the screen).
Bold text	Identifies a selectable icon, button, menu, or tab. It also indicates the name of a field, window, dialog box, or panel.
`Courier font`	Identifies text that is displayed on the screen or that the user must type. It identifies the names of files, directories, paths, parameters, keywords, and variables.
Italicized text	Used for emphasis, the titles of books and journals, and non-English words. It also identifies components of an equation or a formula, a placeholder, or an identifier.
Key+Key	Indicates keyboard actions, for example, Ctrl+I (press the I key while pressing the Control key).
Menu > **Command**	Indicates a menu command, for example, **File** > **New** (from the **File** menu, select **New**).
NOTE	Identifies important information.

Contacting your local TCAD Support Team directly

Send an e-mail message to:

- support-tcad-us@synopsys.com from within North America and South America.
- support-tcad-eu@synopsys.com from within Europe.
- support-tcad-ap@synopsys.com from within Asia Pacific (China, Taiwan, Singapore, Malaysia, India, Australia).
- support-tcad-kr@synopsys.com from Korea.
- support-tcad-jp@synopsys.com from Japan.

CHAPTER 1 Simulation tool flow

This chapter describes the simulation tool flow using Epi and MatPar.

Project directory structure

Epi and MatPar are distributed as part of a special Sentaurus Workbench project called the Epi/MatPar project. The Sentaurus Workbench directory name for this project is `epi`. The Epi/MatPar project consists of:

- Epi, creates the multilayer stack.
- Sentaurus Structure Editor, which performs further device processing on the multilayer stack, adds contacts, and creates the complete device for device simulation.
- MatPar, which creates the main parameter file and user-defined parameter files.

The project (`epi`) is contained in the parent directory `epi_template`, which is referred to as the Epi/MatPar template directory. As summarized in Table 1, `epi_template` also contains the subdirectories `lib`, `par`, and `pardb`. The `epi_template` directory serves as a template for creating customized applications that use Epi and MatPar (see Chapter 5 on page 85).

Table 1 Subdirectories in template directory, all paths are relative to epi_template

Subdirectory	**Description**
`epi`	Subdirectory name of the Epi/MatPar project, containing input and output files.
`epi/npar`	Subdirectory containing user-defined parameter files created by MatPar. **NOTE** The location for storing the user-defined parameter files cannot be changed.
`lib`	Subdirectory containing source files and library files for Epi and MatPar. The library files contain helper procedures. Some files are part of the Material Parameter Framework.
`par`	Subdirectory containing user-specified source parameter files, which are part of the Material Parameter Framework.
`pardb`	Directory name of Material Parameter Database, containing model files.
`pardb/mprPlotter`	Directory name of Material Parameter Plotter.

The Epi/MatPar template project is distributed as part of the application note *Template for Creating and Simulating Multilayered Heterostructure Devices with TCAD Sentaurus* [1].

Files in template directory

Table 2 summarizes all of the files specific to the `epi_template` directory. The files are divided into three categories:

- Tool-specific internal files: These contain tool-specific source code and utility functions. Internal files for both Epi and MatPar are located in the `lib` subdirectory. It is not recommended that users modify these files.
- Tool input files: These are user-supplied input command files for Epi and MatPar. Most are in the `epi` subdirectory. The source parameter files are located in the `par` subdirectory.
- Tool output files: These are output files created by the tools. Most are in the `epi` subdirectory. The user-defined material parameter files created by MatPar are in the `epi/npar` subdirectory.

In Table 2, if a file is listed as `epi/epi_epi.csv`, the location of this file is `epi_template/epi/epi_epi.csv`.

In addition, `n@node@` is replaced by `nX` in the file names. For example, the file `n@node@_epi.scm` is listed as `nX_epi.scm` for brevity.

Table 2 Files in subdirectories of template directory

File type	File	Description
	`epi/gtooldb.tcl`	Contains commands that add Epi and MatPar to a Sentaurus Workbench project.
Epi		
Internal	`lib/epi.gif`	Image file for Epi icon.
	`lib/epi.tcl`	Tcl source code for Epi.
	`lib/helper.tcl`	Contains helper procedures for Epi and MatPar.
	`lib/layers.tcl`	Contains additional procedures to create multilayer stacks.
Input	`epi/epi_epi.csv`	User-specified CSV file, containing all relevant data for each layer of the multilayer stack.
Output	`epi/nX_AxisAligned.txt`	Contains `AxisAligned` section [2] of Sentaurus Mesh command file. Created only if `(yref qw)` is used in the `Refinement` field of the Epi command file.
	`epi/nX_epi.scm`	Contains Scheme commands to create multilayer stack and to perform mesh refinement.
	`epi/nX_epi.tcl`	Contains information about each layer in the form of Tcl variables.
	`epi/nX_epi_gen.scm`	Uses `nX_epi.scm` to create the input files, `nX_epi_msh.bnd` and `nX_epi_msh.cmd`, for the mesher.

Table 2 Files in subdirectories of template directory

File type	File	Description
Output	`epi/nX_epi_msh.bnd`	Input boundary file for the mesher.
	`epi/nX_epi_msh.cmd`	Input command file for the mesher.
	`epi/nX_epi_msh.tdr`	Multilayer stack created by Epi. Used only to view and verify the multilayer stack.
Sentaurus Structure Editor		
Input	`epi/sde_dvs.cmd`	User-specified command file that is used to generate the complete device from the multilayer stack. Contains commands for performing additional device-processing steps on the multilayer stack and for adding contacts.
Output	`epi/nX_msh.bnd`	Input boundary file for Sentaurus Mesh.
	`epi/nX_msh.cmd`	Input command file for Sentaurus Mesh.
	`epi/nX_msh.tdr`	Complete device structure. Used for device simulation.
MatPar		
Internal	`lib/compound_lib.tcl`	Contains procedures for interpolating property values for compound semiconductors.
	`lib/helper.tcl`	Contains helper procedures for Epi and MatPar.
	`lib/MakeParDoc.tcl`	Tcl script file for creating online HTML documentation of Material Parameter Database.
	`lib/mpr.gif`	Image file for MatPar icon.
	`lib/mpr.tcl`	Tcl source file for MatPar.
	`lib/PhysicalConstants.tcl`	Contains definitions of fundamental physical constants and user-defined constants. Used by model files to calculate model parameters and material properties.
	`lib/printPar.tcl`	Contains procedures for printing the parameter sections in user-defined parameter files.
Input	`epi/MatPar_mpr.cmd`	User-specified command file for MatPar. Used to define some variables and invoke some procedures.
	`par/<material>.par`	User-specified source parameter file. There is one `<material>.par` file for each material or region. MatPar does not process this file, and it is included as is, in the user-defined parameter file. Use this file if Tcl procedures are not needed.
	`par/<material>.tcl`	User-specified Tcl source parameter file. There is one `<material>.tcl` file for each material or region. You specify the material property models in this file, and MatPar processes it to create user-defined parameter files.

Table 2 Files in subdirectories of template directory

File type	File	Description
Input	`par/material_template.tcl`	A template Tcl source parameter file. Useful for creating the Tcl source parameter file for any material.
	`pardb/../<model>.tcl`	Model files. Material parameter calculations are implemented in model files. All model files are in an appropriate directory under `pardb`.
Output	`epi/nX_mpr.par`	The main parameter file required by Sentaurus Device. The name of this file can be specified by the user in the MatPar command file.
	`epi/npar/<parFileMask>.par`	User-defined parameter files created by MatPar. The name of regionwise files depends on the value of `parFileMask` defined in `epi_epi.csv`. The name of materialwise files is `n(node)_(material).par`.

NOTE In the context of Table 2 on page 2 and Figure 1 on page 5, `nX` is part of the file name of the output files of both Epi and MatPar. In general, in this manual, `nX` is part of the file name of the output file of any TCAD Sentaurus tool.

Typical device simulation tool flow

Figure 1 shows a device simulation tool flow. It starts with Epi, which generates the multilayer stack. Sentaurus Structure Editor creates the complete device structure (the multilayer stack with contacts). MatPar creates the user-defined material parameter files, which are used by Sentaurus Device to simulate the electrical/optical characteristics of the device. Finally, Tecplot SV is used to visualize the output from the simulation in 2D and 3D, and Inspect plots the electrical characteristics (not shown in Figure 1).

Epi creates a multilayer stack with the help of Sentaurus Structure Editor and Sentaurus Mesh, which is not explicitly included in the Sentaurus Workbench tool flow. Instead, Sentaurus Mesh is called from Sentaurus Structure Editor internally. The input file for Epi, `epi_epi.csv`, contains information about each layer as well as the entire structure. It also contains information about the source parameter files (`<material>.tcl` or `<material>.par`) to be used by MatPar to generate user-defined material parameter files.

Epi generates Scheme script files, `n@node@_epi.scm` and `n@node@_epi_gen.scm`, and a Tcl script file, `n@node@_epi.tcl`. Epi internally invokes Sentaurus Structure Editor and the mesher to create the multilayer stack and to store the structure in `n@node@_epi_msh.tdr` as follows.

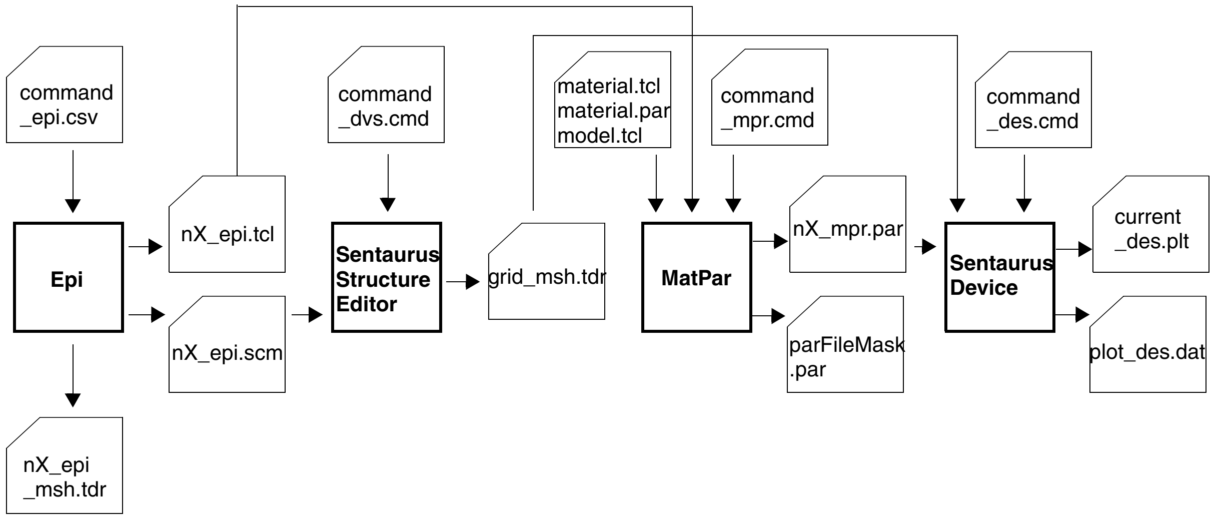


Figure 1 Device simulation tool flow in Sentaurus Workbench that includes Epi, Sentaurus Structure Editor, MatPar, and Sentaurus Device

The file `n@node@_epi.scm` contains commands for creating the multilayer stack and for mesh refinement in the y-direction, and it is sourced inside `n@node@_epi_gen.scm`, which is used by Sentaurus Structure Editor to generate the input files of the mesher, `n@node@_epi_msh.bnd` and `n@node@_epi_msh.cmd`. Finally, Sentaurus Mesh creates the multilayer stack in the file `n@node@_epi_msh.tdr`. The TDR file created at this stage can be used to verify the multilayer stack.

All Scheme variables from `n@node@_epi.scm` and complete information about all of the layers in the structure and the source material parameter files are stored in `n@node@_epi.tcl` as Tcl variables.

The Scheme script file `n@node@_epi.scm` is sourced in the command file `sde_dvs.cmd` for the subsequent Sentaurus Structure Editor instance to create the complete device structure. In this instance of Sentaurus Structure Editor, additional device-processing steps, such as etching of layers and deposition of additional layers, can be performed. This is followed by the addition of contacts and mesh refinement. Finally, the device structure is generated using Sentaurus Mesh and is stored in `n@node@_msh.tdr`.

Sentaurus Device needs a main parameter file that contains a list of references to the user-defined regionwise and materialwise parameter files. MatPar creates these parameter files. The layer information is loaded into MatPar by sourcing `n@node@_epi.tcl` in the MatPar command file `MatPar_mpr.cmd`. The models are specified for each material in the Tcl source parameter file (`<material>.tcl`) for each material. The models and the model parameters are implemented in the model files (`<model>.tcl`) in the Material Parameter Database.

MatPar uses all of these input files to create the main parameter file, `n@node@_mpr.par`, as well as the user-defined parameter files. The user-defined parameter files are stored in the `epi/npar` subdirectory.

File references: @file_type@

Table 3 lists some of the file references `@file_type@` defined in the file `gtooldb.tcl`.

Table 3 Description of file references defined in gtooldb.tcl

File reference	File name
`@episcm@`	`epi/nX_epi.scm`
`@epitcl@`	`epi/nX_epi.tcl`
`@mprpar@`	`epi/nX_mpr.par`
`@tdr@`	`epi/nX_msh.tdr` (electrical grid)
`@tdrop@`	`epi/nX_op_msh.tdr` (optical grid; useful for dual-grid simulation)

References

[1] *Template for Creating and Simulating Multilayered Heterostructure Devices with TCAD Sentaurus*, TCAD Sentaurus application note, available from SolvNet at <https:\\solvnet.synopsys.com>, April 2008.

[2] *Mesh Generation Tools User Guide*, Version A-2007.12, Mountain View, California: Synopsys, Inc., 2007.

CHAPTER 2

Using Epi

This chapter describes how to use Epi to create a multilayer stack.

Overview

Epi creates a multilayer stack from the user-specified description of the stack in the Epi command file `epi_epi.csv`.

Figure 2 shows an example of a multilayer stack created by Epi. The structure is described in terms of the:

- Geometric properties of the complete multilayer stack (see Global section on page 9).
- Properties of each layer (see Layers section on page 13).
- Mesh refinement strategy (see Layers section on page 13).
- Mesh generation and the options for invoking the meshing engine (see Mesh refinement strategy on page 18).
- Names of the Extension columns (see Extension columns on page 31).

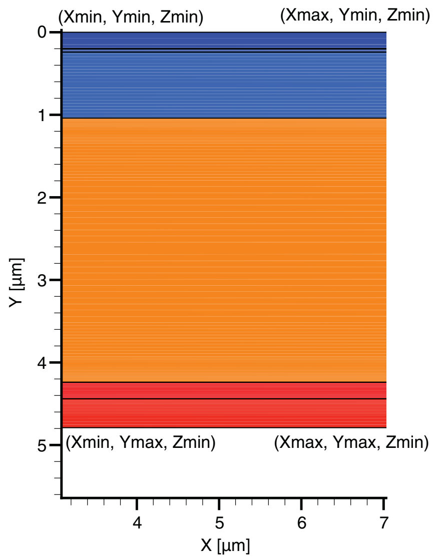


Figure 2 Epi-generated multilayer stack with the coordinate system; positive z-direction points into the plane of the paper

Epi uses the structure information to generate automatically a Scheme script file (`n@node@_epi.scm`). This file contains commands to generate the structure using Sentaurus Mesh (`n@node@_epi_msh.tdr`). This saves effort in creating the Scheme script file. Epi also generates a Tcl script file, `n@node@_epi.tcl`, which contains Tcl variables that store all of the information from the CSV file.

In addition to this information, the following MatPar-specific information is specified in the CSV file (see Layers section on page 13):

- Source material parameter files used by MatPar to create the user-defined material parameter files.
- Regionwise statements inserted by MatPar into the preprocessed Sentaurus Device command file.

Starting Epi

The Epi source file is the Tcl script file, `epi.tcl`, which is located in the `lib` subdirectory. The Tcl interpreter `gtclsh` executes Epi. `gtclsh` is a Tcl shell that has been extended with all of the internal commands of the Sentaurus Workbench framework [1].

You can start Epi from either the command line or Sentaurus Workbench.

Epi is driven by the command file `epi_epi.csv`. Epi can be invoked from the command line by executing the following command from inside the `epi` subdirectory:

```
gtclsh ../lib/epi.tcl <command_file>
```

The file `gtooldb.tcl` contains appropriate commands to launch Epi automatically through the Scheduler when working inside Sentaurus Workbench.

The Epi command file `epi_epi.csv` can be easily edited in either a text editor or a spreadsheet application. The CSV file can be directly accessed from Sentaurus Workbench by using one of the following methods:

- To open the file in SEdit, right-click the Epi icon in the tool row, and select **Edit Input** > **Commands**.
- To open the file in a spreadsheet application, for example, OpenOffice on Linux, right-click the Epi icon and select **Edit Input** > **Edit in Spreadsheet Application**.

NOTE If `epi_epi.csv` is opened in a spreadsheet application, the file must be saved in the CSV format.

Saving command file in CSV format in OpenOffice

To save the command file in OpenOffice, in the CSV format:

1. **File** > **Save As**.
2. In the **File Selection** dialog box:
   a) Select **Text CSV (.csv;.txt)** as the file name extension.
   b) Select **Edit filter settings**.
   c) Click **Save**.
3. In the **Overwrite file?** dialog box, select **Yes**.
4. In the **Export of text files** dialog box:
   a) Select “,” as the field delimiter.
   b) Leave the **Text delimiter** field empty.
5. Click **OK**.

Epi command file

The Epi command file `epi_epi.csv` contains all of the information needed to create the multilayer stack and consists of the:

- Global section, which contains variable definitions related to the complete multilayer stack.
- Layers section, which defines layer properties and source parameter files for user-defined regionwise parameter files.
- Material and Interface section, which defines source parameter files for user-defined materialwise parameter files and materialwise interface parameter files.

Global section

The Global section defines variables called Epi variables, which define the properties of the complete multilayer stack. The value of a particular Epi variable is the same for all layers in the stack. The following types of Epi variable can be defined:

- *Predefined Epi variables* are predefined by Epi and have default values. You can modify the value of these variables. Table 4 on page 11 describes these variables.
- *User-defined Epi variables* are defined by the user.

All Epi variables are available to Sentaurus Structure Editor in the subsequent node as Scheme variables (through `n@node@_epi.scm`), as well as to MatPar (through `n@node@_epi.tcl`) as Tcl variables.

All statements in the Global section begin with one of the following two characters:

- The hash (`#`) is used to insert comments into the CSV file or to add Sentaurus Workbench preprocessor #-commands [1].
- The dollar sign (`$`) precedes the `global` command and the extended commands `repeat`, `end`, and `i` (see Extended commands on page 31).

The `global` command is used either to modify the predefined Epi variables or to create user-defined Epi variables. The following syntax is used:

```
$global variable1=<value>, variable2=<value>, ...
```

An example of a typical Global section is:

```
# Simple solar cell structure
$global Ytop=0.0
$global Xmin=0.0, Xmax=@wtot@
$global wtot=@wtot@
$global dXmin=5, dXmax=5, dYmin=0.5, dYmax=0.5
$global generate=tdr
$global parFileMask=./npar/n(node)_(material)_(doping).par
$global append columnNames extension
```

where:

- `Ytop`, `Xmin`, and `dXmin` are predefined Epi variables.
- `@wtot@` is a Sentaurus Workbench variable.
- `wtot` is a user-defined Epi variable.

As this example shows, Sentaurus Workbench variables can also be accessed in the CSV file.

Table 4 Structure information in Global section of CSV file, listing predefined Epi variables

Structure information	Predefined Epi variables	Description	Default value	Unit
Geometric properties of complete multilayer stack in all three directions				
Structure thickness	`Ytop`	The y-coordinate of the top edge of the top layer.	`0`	µm
	`Ybot`	The y-coordinate of the bottom edge of the bottom layer. **NOTE** Calculated by Epi. Do not define this variable.	–	µm
		`Ybot` – `Ytop` = structure thickness. There is no default value for structure thickness since Epi calculates it.		
Structure width	`Xmin`	Leftmost extent of the multilayer stack.	`0`	µm
	`Xmax`	Rightmost extent of the multilayer stack.	`1`	µm
		`Xmax` – `Xmin` = structure width. The default value of structure width is $1\ \mu m$.		
Structure depth	`Zmin`	Frontmost extent of the multilayer stack.	`0`	µm
	`Zmax`	Rearmost extent of the multilayer stack.	`0`	µm
		`Zmax` – `Zmin` = structure depth. The default value of structure depth is $0\ \mu m$. **NOTE** If `Zmax` – `Zmin` = 0, a 2D structure is generated. If `Zmax` – `Zmin` ≠ 0, a 3D structure is generated.		
Global mesh refinement strategy for entire structure				
Minimum element size	`dXmin, dYmin, dZmin`	Minimum background element size in the x-, y-, and z-direction.	`9999.0`	µm
Maximum element size	`dXmax, dYmax, dZmax`	Maximum background element size in the x-, y-, and z-direction.	`9999.0`	µm
Electrode information				
Top contact	`topContact`	If defined, places a contact on the top of the multilayer stack.	–	–
Bottom contact	`bottomContact`	If defined, places a contact at the bottom of the multilayer stack.	–	–
Mesher information				
Tool name	`mesher`	Specifies the name of the meshing tool: `<snmesh|mesh>`.	`snmesh`	–
Tool options	`meshOptions`	Optional commands for the mesher.	–	–

Table 4 Structure information in Global section of CSV file, listing predefined Epi variables

Structure information	**Predefined Epi variables**	**Description**	**Default value**	**Unit**
Structure file	`generate`	Specifies the format of the structure file (`n@node@_epi_msh`) used to store the multilayer stack created by Epi: `<tdr|grd>`. If `generate` is not specified, the file is not created.	–	–
Information used by MatPar				
Parameter file name	`parFileMask`	Determines the file name of the user-defined parameter files. In general: `parFileMask=./npar/n(node)_(material)_(doping)_(xMole)_(yMole).par`	Regionwise files: `./npar/n(node)_(material)_(xMole).par` Materialwise files: `./npar/n(node)_(material).par`	–
Extension column names	–	A command is used to add extension columns: `$global append columnNames <column1> <column2> ...`	–	–

Extent of structure

The variables `Xmin`, `Xmax`, `Ytop`, `Ybot`, `Zmin`, and `Zmax` describe the extent of the multilayer stack in all three directions. Figure 2 on page 7 illustrates the coordinate system. The positive z-axis points into the plane of the paper. Epi creates the multilayer stack with increasing y-coordinate as the growth direction. The coordinates of the starting point are defined by `Xmin`, `Ytop`, and `Zmin`.

By default, `Xmin` = `Ytop` = `Zmin` = `0`. In addition, as shown in Figure 2, `Xmin` < `Xmax`, `Ytop` < `Ybot`, and `Zmin` < `Zmax`. All layers have the same width and depth, which can be defined using `Xmin`, `Xmax`, `Zmin`, and `Zmax` (see Table 4 on page 11).

`Ytop` can be redefined, whereas `Ybot` is calculated from the thickness of each layer in the structure. Epi decides the dimensionality of the structure using the values of `Zmin` and `Zmax` (see Table 4).

Global mesh refinement strategy

`dXmin` and `dXmax` are minimum and maximum element sizes, respectively, for global mesh refinement in the x-direction for the entire structure. `dYmin`, `dYmax`, `dZmin`, and `dZmax` have similar meaning.

NOTE Epi generates global refinement commands only if the value of at least one of these variables is modified by the user.

If you define `dXmin=0`, or `dYmin=0`, or `dZmin=0`, Epi resets this minimum element size to be equal to the corresponding maximum element size. The global refinement window is positioned to cover the complete multilayer stack.

NOTE Epi saves the structure in a TDR file (`n@node@_epi_msh`) only if you define the `generate` variable. In addition, this TDR file is used only for verifying the multilayer stack.

NOTE For details on `parFileMask`, see Material and Interface section on page 19.

Layers section

All layers of the multilayer stack are defined in the Layers section, which consists of one statement for each layer to be created, starting with the top layer of the structure. The layer description in each statement consists of several comma-separated fields:

```
<Region>, <Material>, <SourceParFile>, <Thickness>, <Doping>, <MoleFraction>,
<Refinement>, <Extension>
```

The fields contain information about each layer that Epi uses to create the multilayer stack. Table 5 describes the fields of the Layers section.

Table 5 Fields in Layers section of CSV file

Field	**Description**	**Unit**
`Region`	Name of the region to be created.	–
`Material`	Material names that Sentaurus Device recognizes.	–
`SourceParFile`	Name of the source parameter file used to create the user-defined material parameter files. If the file extension is `.tcl`, it is processed. If the file extension is `.par`, it is included as is.	–
`Thickness`	Thickness of each layer. If this field is empty, the region is not created.	μm
`Doping`	Doping of each layer. It is specified using commands in Table 7 on page 16. A negative sign indicates p-type doping; a positive sign indicates n-type doping. Certain doping profiles can also be specified.	cm^{-3}
`MoleFraction`	Mole fraction for ternary or quaternary materials. It is specified using commands in Table 8 on page 18. Linear mole-fraction profiles can also be specified.	[1]

Table 5 Fields in Layers section of CSV file

Field	Description	Unit
`Refinement`	Regionwise mesh refinement strategy for each layer. It can be omitted if a global refinement strategy is defined in the Global section. It is specified using commands in Table 9 on page 18.	µm
`Extension`	This field is for arbitrary content. The column contents can be inserted into the preprocessed Sentaurus Device command file. Multiple Extension columns are allowed.	–

Table 6 shows an example of the content of the Layers section and Material and Interface section in tabular format. The corresponding listing in the CSV file is:

```
# Layers section
# Region, Material, SourceParFile, Thickness, Doping, MoleFraction,
# Refinement, Extension
cap,GaAs,,0.5,-5.00e19,,(yref 0.1),
fsf,AlGaAs,AlGaAs.tcl,0.03,-4.00e18,0.8,(yref 0.01),
emitter,GaAs,GaAs.tcl,0.5,-5.00e17,,(mbox 0.05 1.1
both),Recombination(Radiative)
base,GaAs,,2.5,1.00e17,,(mbox 0.05 1.1 both),Recombination(Radiative)
substrate,Germanium,Ge.tcl,5.0,1.00e18,,(mbox 0.05 1.1 both),
# Material and Interface section
# Materialwise parameter file definition
,GaAs,GaAs.tcl,,1.00e18,,,
,MgF,MgF.par,,,,
# Materialwise interface parameter file definition
,GaAs/AlGaAs,GaAs_AlGaAs.par,,,
```

Table 6 Example of Layers and Material and Interface sections of epi_epi.csv

Region	Material	SourceParFile	Thickness	Doping	Mole Fraction	Refinement	Extension
# Layers section:							
`cap`	`GaAs`		`0.5`	`-5.00e19`		`(yref 0.1)`	
`fsf`	`AlGaAs`	`AlGaAs.tcl`	`0.03`	`-4.00e18`	`0.8`	`(yref 0.01)`	
`emitter`	`GaAs`	`GaAs.tcl`	`0.5`	`-5.00e17`		`(mbox 0.05 1.1 both)`	`Recombination (Radiative)`
`base`	`GaAs`		`2.5`	`1.00e17`		`(mbox 0.05 1.1 both)`	`Recombination (Radiative)`
`substrate`	`Germanium`	`Ge.tcl`	`5.0`	`1.00e18`		`(mbox 0.05 1.1 both)`	

Table 6 Example of Layers and Material and Interface sections of epi_epi.csv

Region	Material	SourceParFile	Thickness	Doping	Mole Fraction	Refinement	Extension
# Material and Interface section:							
# Materialwise parameter file definition							
	`GaAs`	`GaAs.tcl`		`1.00e18`			
	`MgF`	`MgF.par`					
# Materialwise interface parameter file definition							
	`GaAs/AlGaAs`	`GaAs_AlGaAs.par`					

NOTE If all fields following a particular field are empty in a layer definition, the number of commas after this field is irrelevant. For example, all fields after `Doping` are empty for the materialwise parameter file definition of GaAs. Therefore, this can be written in one of two ways:

```
,GaAs,GaAs.tcl,,1.00e18,,,
```

```
,GaAs,GaAs.tcl,,1.00e18,
```

Region and material names

Epi creates a region for each layer. The name of the region is specified in the `Region` field. Epi uses the region name to create regionwise doping profiles, mole-fraction profiles, and mesh refinements. For each layer, the names of materials that Sentaurus Device recognizes are specified in the `Material` field. For example, for a germanium layer, `Germanium` is specified in the `Material` field instead of Ge.

To generate a list of all materials that Sentaurus Device recognizes, use:

```
sdevice -L:materials
```

Thickness

The thickness of each layer in micrometers is specified in the `Thickness` field. If the `Thickness` field is empty, Epi does not create the layer and, therefore, does not create the region.

Doping profiles

The doping information for each layer is specified in the `Doping` field. A constant doping concentration or a graded doping profile can be specified using the commands in Table 7. These commands create region-based doping profiles for the layer in the y-direction. The following types of doping profile can be defined in the y-direction:

- Linear function
- Error function
- Gaussian function

Table 7 Commands for Doping field

Command	Description
`<doping>`	Constant doping concentration.
Linear function doping profile	
`(lin <Ntop> <Nbot>)`	Linear region-based doping profile using analytic profile.
`Ntop`	The value of doping concentration at the top edge of the layer [cm^{-3}].
`Nbot`	The value of doping concentration at the bottom edge of the layer [cm^{-3}].
`(lin <Ntop> <Nbot>)` `(default <doping>)`	Linear doping profile with a default reference doping concentration value of `<doping>` [cm^{-3}].
Error function or Gaussian function doping profile	
`(<erf | gauss> <Nmax> <ymax> <length> <ue | de>)` `(default <doping>)`	Error or Gaussian function doping profile with a default reference doping concentration value of `<doping>` [cm^{-3}].
`Nmax`	Maximum concentration/peak concentration [cm^{-3}].
`ymax`	Symmetry position/peak position [$\mu\mathrm{m}$].
`length`	Length [$\mu\mathrm{m}$].
`<ue | de>`	Controls the edge or face of the layer at which the baseline is placed: `ue` = Baseline is placed at the top edge or face. `de` = Baseline is placed at the bottom edge or face.

As discussed in Chapter 1 on page 1, Epi generates a command file for the meshing tool, `nX_epi_msh.cmd`. For each layer, the above analytic profiles are created with the following parameters [2] in the different sections of this command file:

- The `AnalyticalProfile` section of the `Definition` section:
  - `Species = ArsenicActiveConcentration | BoronActiveConcentration` (for n-type or p-type doping, respectively)
  - `Function = General | Error | Gauss` (for linear, error, or Gaussian function doping profile, respectively)
  - `LateralFunction = Erf(Factor=0)` (for error and Gaussian profiles)
- The `AnalyticalProfile` section of the `Placements` section:
  - The baseline for each layer is the bottom or top edge (2D) or face (3D), depending on the specification of `ue | de` in the `Doping` field (see Table 7 on page 16).
  - The variable `Direction` is not specified in the `ReferenceElement` section since the function values are computed on both sides of the baseline.
  - All analytic doping profiles are placed using the `EvaluateWindow` section. For each layer, the domain for the evaluation of the profile is the region name and `DecayLength=0`.
  - The variable `Replace` is not specified.

As discussed in Creating user-defined parameter files for regions with doping/mole-fraction profiles on page 29, a default value for the reference doping concentration must be defined for layers with graded doping profiles. The command `default` is used for this purpose.

Mole-fraction profiles

The mole-fraction information about layers consisting of ternary or quaternary compound semiconductor materials is described in the `MoleFraction` field, which is empty for elemental or binary semiconductors. A constant mole fraction or a linearly graded mole-fraction profile can be specified using the commands in Table 8 on page 18. These commands create region-based mole-fraction profiles for the layer in the y-direction.

A default reference x–mole fraction value must be defined for layers with graded x–mole fraction profiles (see Creating user-defined parameter files for regions with doping/mole-fraction profiles on page 29). The command `default` is used for this purpose.

NOTE Currently, this feature does not support layers with graded y–mole fraction profiles.

Table 8 Commands for MoleFraction field

Command	Description
Constant mole fraction	
`<xMole>`	Constant x–mole fraction for ternary compound semiconductors.
`(<xMole> <yMole>)`	Constant x–mole fraction and y–mole fraction for quaternary compound semiconductors.
Linear mole-fraction profile in y-direction	
`(lin <xMoleTop> <xMoleBot>)` `(default <xMole>)`	Linear region-based x–mole fraction profile with a default reference x–mole fraction value of `<xMole>`.
`(y lin <yMoleTop> <yMoleBot>)`	Linear region-based y–mole fraction profile [2].
`xMoleTop | yMoleTop`	The value of x–mole fraction or y–mole fraction at the top edge of the layer.
`xMoleBot | yMoleBot`	The value of x–mole fraction or y–mole fraction at the bottom edge of the layer.

Mesh refinement strategy

The global refinement for the complete multilayer stack is defined in the Global section (see Global section on page 9). As discussed in Chapter 3 on page 35, commands for refinements in the x-direction and z-direction are specified in the Sentaurus Structure Editor command file `sde_dvs.cmd`.

The `Refinement` field in the CSV file performs refinement in the y-direction. Mesh refinement in the y-direction for a particular layer is performed by the commands `yref` and `mbox` (see Table 9). The terminology related to mesh refinement is discussed in [2] and [3].

Table 9 Commands for Refinement field

Command	Description
Uniform regionwise refinement	
`(xref <elsize>)`	Uniform regionwise refinement in the x-direction.
`(yref <elsize>)`	Uniform regionwise refinement in the y-direction.
`elsize`	Specifies the element size (mesh spacing) in the appropriate direction. By default, the element size for the x-direction and z-direction is the width and depth of the structure, respectively.
`(yref qw)`	Creates a three-point quantum-well mesh in the y-direction [4].

Table 9 Commands for Refinement field

Command	Description
Multibox regionwise refinement	
`(mbox <minelsize> <ratio> <direction>)`	Defines a multibox regionwise meshing strategy for a layer in the y-direction by specifying: `minelsize`: Specifies the minimum element size in the y-direction. By default, the minimum element size in the x-direction and z-direction is the width and depth of the structure, respectively. The maximum element size in the x-, y-, and z-direction is the width, thickness, and depth of the structure, respectively. `ratio`: The ratio controls how fast the mesh spacing increases from the minimum to the maximum value. Unlike the convention that Sentaurus Structure Editor uses, `ratio` is a positive number between 1 and 2. `direction`: Controls the edge (2D) or the face (3D) of the multibox from which the meshing-line density grading starts. Options are: • `direction=up`: Grading starts from the top edge of the layer, that is, the mesh spacing increases from minimum to maximum in the positive y-direction. • `direction=down`: Grading starts from the bottom edge of the layer, that is, the mesh spacing increases from minimum to maximum in the negative y-direction. • `direction=both`: Grading starts from both edges. The mesh spacing is minimum at the top and bottom of the layer, and increases in both directions towards the center of the layer. This option is useful for creating a fine mesh at interfaces.

Material and Interface section

MatPar generates the main material parameter file required by Sentaurus Device as well as the user-defined material parameter files. User-defined parameter files are defined in Overview on page 37.

MatPar can generate following types of user-defined material parameter file:

- Materialwise (see Materialwise user-defined parameter file on page 20)
- Regionwise (see Regionwise user-defined parameter file on page 22)
- Materialwise interface (see Creating materialwise interface parameter files on page 30)

The main parameter file contains a list of references to the user-defined material parameter files by using the keyword `Insert` [4]. For example, by default, the main parameter file `n@node@_mpr.par` generated by MatPar for the structure listed in Table 6 on page 14 contains the following statements:

```
Material = "GaAs" { Insert = "./npar/n5_GaAs.par" }
Material = "MgF" { Insert = "./npar/n5_MgF.par" }
Region = "fsf" { Insert = "./npar/n5_AlGaAs_0.8.par" }
Region = "emitter" { Insert = "./npar/n5_GaAs.par" }
MaterialInterface = "GaAs/AlGaAs" { Insert = "./npar/n5_GaAs_AlGaAs.par" }
```

MatPar creates the user-defined parameter files by processing source parameter files, either `<material>.tcl` or `<material>.par` (see Source parameter files on page 52). In the above example:

- The first two statements correspond to materialwise parameter files.
- The next two statements correspond to regionwise parameter files.
- The last statement corresponds to a materialwise interface parameter file.

The main parameter file is created in the `epi` subdirectory, and all the user-defined parameter files are created in the `epi/npar` subdirectory.

NOTE You cannot change these locations.

Materialwise user-defined parameter file

If many regions share the same material, a single user-defined parameter file can be created for the material. Such a file is a *materialwise parameter file*.

To create a materialwise parameter file:

1. Leave the `SourceParFile` field empty for regions in the Layers section:

   ```
   # Layers section
   <Region>, <Material>,, <Thickness>, <Doping>, <MoleFraction>, <Refinement>,
   <Extension>
   ```

2. Append a statement for each such material in the Material and Interface section. If the source parameter file is a `<material>.par` file, the `Region`, `Thickness`, `Doping`, `MoleFraction`, `Refinement`, and `Extension` fields are empty in each statement:

   ```
   # Material and Interface section
   , <Material>, <material.par>,,,,,
   ```

3. If a Tcl source parameter file is used to create the materialwise parameter file:
   - For binaries, a reference doping-concentration value is required by the model files and is specified as follows (see Creating user-defined parameter files for regions with doping/mole-fraction profiles on page 29):

     ```
     # Material and Interface section
     , <Material>, <material.tcl>,, <Doping>,,,
     ```

   - For ternaries and quaternaries, a reference doping-concentration and mole-fraction value is required (see Creating user-defined parameter files for regions with doping/mole-fraction profiles on page 29):

     ```
     # Material and Interface section
     # Ternary
     , <Material>, <material.tcl>,, <Doping>, <xMole>,,
     # Quaternary
     , <Material>, <material.tcl>,, <Doping>, (<xMole yMole>),,
     ```

Another method for specifying the reference doping-concentration value is to:

1. Add the following in the CSV file:

   ```
   # Material and Interface section
   , <Material>, <material.tcl>,,,,,
   ```

2. Specify the following in the `<material>.tcl` file:

   ```
   if {[info exists doping] && ([string trim $doping] =="")} {
      set doping <user_defined_value>
   }
   ```

This method can also be used to specify `xMole` and `yMole` in the `<material>.tcl` files for ternaries and quaternaries.

In general, one materialwise user-defined parameter file can be created for each material.

For the structure in Table 6 on page 14, both cap and base regions are GaAs layers. A single materialwise parameter file, `n5_GaAs.par`, for these regions can be created by specifying the following in the CSV file:

```
# Global section
parFileMask=./npar/n(node)_(material)_(xMole).par
# Layers section
cap,GaAs,,0.5,-5.00e19,,(yref 0.1),
base,GaAs,,2.5,1.00e17,,(mbox 0.05 1.1 both),Recombination(Radiative)
# Material and Interface section
,GaAs,GaAs.tcl,,1.00e18,,,
```

Here, a reference doping-concentration value of `1e18` is used for both GaAs regions, the cap and base.

The name of the regionwise parameter files is controlled by the variable `parFileMask` (see Naming parameter files). In the above example, since the doping variable is not included in `parFileName` (defined in Naming parameter files) and there is no x–mole fraction value for GaAs, the materialwise parameter file name for GaAs is `n5_GaAs.par`.

Regionwise user-defined parameter file

In general, each layer in the Layers section of the CSV file is defined as a region. If a regionwise parameter file is required, the source parameter file is specified in the `SourceParFile` field of the corresponding layer definition in the Layers section. In general, one regionwise user-defined parameter file can be created for each layer in the multilayer stack.

For example, for the structure in Table 6 on page 14, `AlGaAs.tcl` and `GaAs.tcl` are specified for the fsf and emitter regions, respectively. Therefore, the regionwise parameter files `n5_AlGaAs_0.8.par` and `n5_GaAs.par` are created for the fsf and emitter regions, respectively:

```
# Global section
parFileMask=./npar/n(node)_(material)_(xMole).par
# Layers section
fsf,AlGaAs,AlGaAs.tcl,0.03,-4.00e18,0.8,(yref 0.01),
emitter,GaAs,GaAs.tcl,0.5,-5.00e17,,(mbox 0.05 1.1 both),
Recombination(Radiative)
```

The name of the regionwise parameter files is controlled by the variable `parFileMask` (see Naming parameter files). In the above example, since `xMole` is specified in `parFileName` and there is no x–mole fraction value for GaAs, the regionwise parameter file names for the fsf and emitter regions are `n5_AlGaAs_0.8.par` and `n5_GaAs.par`, respectively.

Naming parameter files

The name of the regionwise and materialwise user-defined parameter files and the directory for these parameter files are defined by the variable `parFileMask` in the Global section of the CSV file using the syntax:

```
parFileMask=./npar/<parFileName>.par
```

All paths in the command files of Epi and MatPar are specified relative to the `epi` subdirectory, where `./npar` specifies that the user-defined parameter files are stored in the `epi/npar` subdirectory.

NOTE The location for storing the user-defined parameter files cannot be changed.

The variable `parFileName` specifies the name of the user-defined parameter file. This name consists of a string formed by combining the values of the physical variables, corresponding to some fields in the CSV file, separated by underscores. These variables are described in Table 10. The complete string for `parFileName` is:

```
parFileName=n(node)_(material)_(doping)_(xMole)_(yMole)
```

Using the variables `doping`, `xMole`, and `yMole` is optional.

NOTE For binaries, `xMole=" "`. For binaries and ternaries, `yMole=" "`. Empty strings are omitted from file names. The underscore preceding the empty string is also omitted.

Table 10 Variables used in parFileName

Variable	**Description**
`parFileDir`	Directory for saving user-defined parameter files, which is `./npar`.
`n(node)`	Node number of MatPar in the Sentaurus Workbench project.
`material`	Layer material.
`doping`	Doping concentration of each layer: positive for n-type and negative for p-type semiconductor. This variable is optional.
`xMole`	The x–mole fraction of the layer. For binaries, `xMole=" "`. This variable is optional.
`yMole`	The y–mole fraction of the layer. For binaries and ternaries, `yMole=" "`. This variable is optional.

Naming materialwise user-defined parameter files

The default value of `parFileName` for materialwise user-defined parameter files is:

```
parFileName=n(node)_(material)
```

If you do not specify `parFileMask`, the file name of the materialwise user-defined parameter file has the format `n(node)_(material).par`.

If doping is included in `parFileName` and a doping concentration value is specified in the Material and Interface section, the materialwise parameter file name has the format `n(node)_(material)_(doping).par`. If the doping concentration value is not specified in the Material and Interface section, the materialwise parameter file name has the format `n(node)_(material).par`.

If `xMole` and `yMole` are included in `parFileName`, and the mole-fraction value is also specified in the Material and Interface section, the materialwise parameter file name has the format:

- `n(node)_(material)_(doping)_(xMole).par` for ternaries.
- `n(node)_(material)_(doping)_(xMole)_(yMole).par` for quaternaries.

If the mole-fraction values are not specified in the Material and Interface section, the materialwise parameter file name has the format `n(node)_(material).par`.

Naming regionwise user-defined parameter files

The default value of `parFileName` for regionwise user-defined parameter files is:

```
parFileName=n(node)_(material)_(xMole)
```

If you do not specify `parFileMask`, the file name of the regionwise user-defined parameter file has the format:

- `n(node)_(material)_(xMole).par` for ternaries.
- `n(node)_(material).par` for binaries.

If regions have constant doping and mole fractions, the corresponding layer values are used for `doping`, `xMole`, and `yMole` in `parFileName`. If regions have doping and x–mole fraction profiles, the default value is used (see Creating user-defined parameter files for regions with doping/mole-fraction profiles on page 29).

NOTE The case of y–mole fraction profiles is not yet supported.

Order in which MatPar processes source parameter files

MatPar processes the source parameter files to create user-defined parameter files in the following order:

1. If the variable `substrate` is defined in `MatPar_mpr.cmd`, the Tcl source parameter file corresponding to the substrate material is processed first.
2. The source parameter files specified in the materialwise parameter file definition in the Material and Interface section of the CSV file are processed sequentially in the order in which they appear in this section.
3. The source parameter files specified in the Layers section of the CSV file are processed sequentially in the order in which they appear in this section.
4. The source parameter files specified in the materialwise interface parameter file definition in the Material and Interface section of the CSV file are processed sequentially in the order in which they appear in this section.

Example 1

For the structure in Table 6 on page 14, the complete string for `parFileName` is used; therefore, `parFileMask` is:

```
parFileMask=./npar/n(node)_(material)_(doping)_(xMole)_(yMole).par
```

In this case, the main parameter file generated by MatPar for the structure listed in Table 6 contains the following statements:

```
Material = "GaAs" { Insert = "./npar/n5_GaAs.par" }
Material = "MgF" { Insert = "./npar/n5_MgF.par" }
Region = "fsf" { Insert = "./npar/n5_AlGaAs_-4.00e18_0.8.par" }
Region = "emitter" { Insert = "./npar/n5_GaAs_-5.00e17.par" }
MaterialInterface = "GaAs/AlGaAs" { Insert = "./npar/n5_GaAs_AlGaAs.par" }
```

MatPar processes the source parameter files in the following order:

1. `GaAs.tcl` to create the materialwise parameter file `n5_GaAs.par`, which Sentaurus Device uses for all GaAs regions that do not use regionwise parameter files. Here, `n5_GaAs.par` is used for the cap and base regions.
2. `MgF.par` to create the materialwise parameter file `n5_MgF.par`, which is used for all MgF_2 regions.
3. `AlGaAs.tcl` to create the regionwise parameter file `n5_AlGaAs_-4.00e18_0.8.par`, which is used for the fsf region.
4. `GaAs.tcl` to create the regionwise parameter file `n5_GaAs_-5.00e17.par`, which is used for the emitter region.
5. `Ge.tcl` to create the regionwise parameter file `n5_Ge_1.00e18.par`, which is used for the substrate region.
6. `GaAs_AlGaAs.par` to create the materialwise interface parameter file `n5_GaAs_AlGaAs.par`, which is used for all GaAs/AlGaAs interfaces.

In this example, since empty strings are omitted from file names, the regionwise parameter file for fsf is `n5_AlGaAs_-4.00e18_0.8.par`.

NOTE You must take into account the order in which MatPar processes the source parameter files when specifying the source parameter file names and `parFileMask`.

If the default value of `parFileMask` is used, the name of a materialwise parameter file is the same as the name of a regionwise parameter file for elemental and compound semiconductors. In this case, a parameter file created from a source parameter file specified in the Material and Interface section may be overwritten by a parameter file created from a source parameter file specified in the Layers section. As a result, instead of using a materialwise parameter file for a particular material, you may use unintentionally a regionwise parameter file. This is illustrated in Example 2.

Example 2

For the structure in Table 6 on page 14, if a regionwise parameter file is required for the emitter region and a materialwise parameter file is required for all other GaAs regions, the command file is:

```
# Layers section
cap,GaAs,,0.5,-5.00e19,,(yref 0.1),
fsf,AlGaAs,AlGaAs.tcl,0.03,-4.00e18,0.8,(yref 0.01),
emitter,GaAs,GaAs.tcl,0.5,-5.00e17,,(mbox 0.05 1.1
both),Recombination(Radiative)
base,GaAs,,2.5,1.00e17,,(mbox 0.05 1.1 both),Recombination(Radiative)
# Material and Interface section
,GaAs,GaAs.par,,1.00e18,,,
,MgF,MgF.par,,,,,
```

Here, `GaAs.tcl` has been replaced with `GaAs.par` in the materialwise parameter file definition for GaAs. In addition, the substrate region and materialwise interface parameter file definition have been removed.

Since `parFileMask` is not specified, MatPar uses the default value of `parFileMask`. As a result, MatPar processes the source parameter files in the following order:

1. Source parameter files in the Material and Interface section:
   a) `GaAs.par` is processed to create the materialwise parameter file `n5_GaAs.par`.
   b) `MgF.par` is processed to create `n5_MgF.par`.
2. Source parameter files in the Layers section:
   a) `AlGaAs.tcl` is processed to create `n5_AlGaAs_-4.00e18_0.8.par`.
   b) `GaAs.tcl` is processed to create the regionwise parameter file `n5_GaAs.par`, which is used for the emitter region.

The materialwise parameter file `n5_GaAs.par` created in Step 1a is overwritten by the regionwise parameter file created in Step 2b. The regionwise parameter file `n5_GaAs.par` is used for all GaAs regions instead of a separate regionwise parameter file for the emitter region and a materialwise parameter file for all other GaAs regions.

In such cases, you must ensure that the source parameter file for a particular material is the same in the Material and Interface section and the Layers section. If different source parameter files are specified for the same material, this problem can also be solved by using:

```
parFileMask=./npar/n(node)_(material)_(doping).par
```

With the proper choice of `parFileName`, MatPar can create different regionwise parameter files for two different regions with the same material but different doping concentrations. This is illustrated in Example 3.

Example 3

For the structure in Table 6 on page 14, MatPar can create different regionwise parameter files for the cap and emitter regions as follows:

- Specify `parFileMask=./npar/n(node)_(material)_(doping).par`.
- Specify `GaAs.tcl` in the cap layer definition.

As a result, MatPar creates the main parameter file containing the statements:

```
Material = "GaAs" { Insert = "./npar/n5_GaAs.par" }
Material = "MgF" { Insert = "./npar/n5_MgF.par" }
Region = "cap" { Insert = "./npar/n5_GaAs_-5.00e19.par" }
Region = "fsf" { Insert = "./npar/n5_AlGaAs_-4.00e18.par" }
Region = "emitter" { Insert = "./npar/n5_GaAs_-5.00e17.par" }
MaterialInterface = "GaAs/AlGaAs" { Insert = "./npar/n5_GaAs_AlGaAs.par" }
```

Here, both the cap and emitter regions are made of GaAs, but they have different doping concentrations. Since the variable `doping` is included in `parFileName`, MatPar creates different parameter files for these regions. Instead of using the materialwise parameter file `n5_GaAs.par`, a regionwise parameter file, `n5_GaAs_-5.00e19.par`, is used for the cap region. The base region continues to use `n5_GaAs.par` since `GaAs.tcl` was not specified in the base layer definition in the CSV file.

Creating user-defined parameter files for regions/materials not in Epi structure

As discussed in Chapter 3 on page 35, Sentaurus Structure Editor further processes the multilayer stack created by Epi. These processing steps may result in the:

- Formation of new regions (regions not included in the CSV file).
- Addition of regions with new materials (materials not included in the CSV file).
- Removal of a region that was defined in the CSV file.

You can create user-defined parameter files for the first two cases by specifying appropriate source parameter files in the CSV file.

To create user-defined parameter files for materials not included in the CSV file, specify the new material name and source parameter file name in the Material and Interface section of the CSV file (see Materialwise user-defined parameter file on page 20):

```
# Material and Interface section
, <New_Material>, <SourceParFile>,,,,,
```

For example, for the multilayer stack defined in Table 6 on page 14, an MgF_2 layer is deposited using Sentaurus Structure Editor. Therefore, the following is included in the CSV file:

```
# Material and Interface section
, MgF, MgF.par,,,,,
```

To create regionwise or materialwise parameter files for new regions, follow the procedure in Materialwise user-defined parameter file on page 20 with the modification of leaving the `Thickness` and `Refinement` fields empty in the Layers section:

```
# Layers section
<New_Region>, <Material>, <SourceParFile>,, <Doping>, <MoleFraction>,,
<Extension>
```

Epi does not create the `<New_Region>` layer since the `Thickness` field is empty. Epi uses the `Doping` and `MoleFraction` fields to define the name of the regionwise parameter file for `<New_Region>`, and MatPar uses these fields to create the regionwise parameter file.

If a region included in the CSV file is removed by Sentaurus Structure Editor, leave the `SourceParFile`, `Refinement`, and `Extension` fields empty for that region. MatPar does not create a parameter file for that region:

```
# Layers section
<Region>, <Material>,, <Thickness>, <Doping>, <MoleFraction>,,
```

This information is applicable when Sentaurus Structure Editor creates lumped regions. If Sentaurus Structure Editor separates a region called `emitter` into two disconnected lumps and if the new region names are `emitter_lump_1` and `emitter_lump_2`, the following listing in the CSV file creates parameter files for the lumped regions, and a parameter file is not created for the emitter region:

```
# Layers section
emitter, GaAs,, 2, -5.00e17,,,
emitter_lump_1, GaAs, GaAs.tcl,, -5.00e17,,,
emitter_lump_1, GaAs, GaAs.tcl,, -5.00e17,,,
```

Creating user-defined parameter files for regions with doping/mole-fraction profiles

For each layer in the multilayer stack, the physical variables `doping`, `xMole`, and `yMole` are used to:

- Define the name of the regionwise user-defined parameter file (see Materialwise user-defined parameter file on page 20).
- Compute material properties by some of the model files (see Model files on page 58). In the case of the TableODB parameter section, for some models in the Material Parameter Database[1], these variables are also used to select the appropriate data file required for printing this section.

For each layer, these variables should have a unique value. By default, these variables have a unique value for regions with uniform doping concentration or mole fraction. A unique value for `doping` and `xMole` can be assigned to regions with doping and x–mole fraction profiles by using the command `default` in the `Doping` and `MoleFraction` fields in the Layers section of the CSV file as follows:

```
(<profile_definition>)(default <value>)
```

The syntax for specifying `<profile_definition>` is discussed in Table 7 on page 16 and Table 8 on page 18.

For example:

```
# Global section
parFileMask=./npar/n(node)_(doping)_(xMole).par
# Layers section
region1, GaAs, GaAs.tcl, 5,, (lin 1e15 1e18)(default 1e17),, (yref 1)
region2, AlGaAs, AlGaAs.tcl, 2, 1e18, (lin 0 1)(default 0.5),, (yref 1)
```

NOTE The use of this syntax for regions with y–mole fraction profiles is not yet supported. In addition, the use of the `default` command is not required if the source parameter file is a `<material>.par` file.

For binaries, if the structure consists of several regions with identical materials and doping profiles, a unique value for `doping` can be specified in the Material and Interface section without using the `default` command as follows:

1. Omit the source parameter file name and the `default` command from the `Doping` field of the regions.
2. Include the corresponding default value in a Material and Interface section.

1. For example, the model file `pardb/GaAs/Optics/TableODB/Levinshtein.tcl` implements the doping-dependent complex refractive index model from Levinshtein [5].

For example:

```
# Global section
$global parFileMask=/npar/n(node)_(material)_(doping).par
# Layers section
region1, GaAs,, 5,, (lin 1e15 1e18),, (yref 1)
region2, GaAs,, 2,, (lin 1e15 1e18),, (yref 1)
region3, GaAs, GaAs.tcl, 0.3,, (lin 5e18 6e18)(default 1e15),, (yref 0.1)
# Material and Interface section
, GaAs, GaAs.tcl,,, 1e17,,
```

Here, a materialwise parameter file `nX_GaAs_1e17.par` is created for `region1` and `region2`, and a regionwise parameter file `nX_GaAs_6e18.par` is created for `region3`.

Creating materialwise interface parameter files

A user-defined materialwise interface parameter file can be created by using the following syntax in the Material and Interface section:

```
,<material1/material2>,,<SourceParFile>,,,
```

MatPar uses the source parameter file `SourceParFile` to create a materialwise interface parameter file, `n(node)_<material1>_<material2>.par`, for the interface between `material1` and `material2`.

NOTE Here, the source parameter file should have the extension `.par`. A Tcl source parameter file cannot be specified.

In general, one materialwise interface parameter file can be created for each pair of material interfaces.

Example 4

For the structure in Table 6 on page 14, MatPar can create a materialwise interface parameter file for GaAs/AlGaAs interfaces using the commands:

```
# Material and Interface section
# Materialwise interface parameter file definition
,GaAs/AlGaAs,GaAs_AlGaAs.par,,,,
```

In this case, MatPar uses the parameter file `GaAs_AlGaAs.par` to create the user-defined materialwise interface parameter file `nX_GaAs_AlGaAs.par`. As a result, the main parameter file contains the line:

```
MaterialInterface = "GaAs/AlGaAs" { Insert = "./npar/n5_GaAs_AlGaAs.par" }
```

and Sentaurus Device uses this parameter file for all GaAs/AlGaAs interfaces in the structure.

Extension columns

The `Extension` field can have arbitrary content and can be used by MatPar to insert this content in the preprocessed Sentaurus Device command file. There can be multiple columns in the `Extension` field.

NOTE By default, there is no `Extension` field. In addition, the `Extension` field is not applicable to the Material and Interface section.

Creating multiple columns

To create multiple columns in the `Extension` field, use the command:

```
$global append columnNames <extensionColumnName1> <extensionColumnName2> ...
```

where `extensionColumnName1` and `extensionColumnName2` are the column names of the `Extension` field. These columns are saved in a Tcl array in the file `n@node@_epi.tcl`. The content of these columns can be accessed from the MatPar command file (see Inserting Extension columns in Sentaurus Device command file on page 49).

Extended commands

Epi uses the extended commands `repeat`, `end`, and `i` to generate a substack of layers repeatedly. This reduces the manual entry of the required number of layers for highly repetitive structures such as distributed Bragg reflectors (DBRs) in vertical-cavity surface-emitting lasers.

Table 11 shows part of a multiple–quantum well structure, in which the quantum well (`qwell$i`) and the barrier (`qwbarrierSCH$i`) are repeated.

Table 11 Epi command file consisting of part of the multiple–quantum well structure

Region	**Material**	**SourcePar File**	**Thickness**	**Doping**	**Mole Fraction**	**Refinement**	**Extension**
`pbarrierSCH`	`GaN`	`GaN.tcl`	`0.007`	`-3.00e15`		`(mbox 0.001 1.2 both)`	
`$repeat 2`						`(yref qw)`	
`qwell$i`	`InGaN`	`InGaN.tcl`	`0.001`	`-3.00e15`	`0.08`	`(yref qw)`	`Physics(region= qwell$i) {Recombination (-Radiative) Active}`

Table 11 Epi command file consisting of part of the multiple–quantum well structure

Region	Material	SourcePar File	Thickness	Doping	Mole Fraction	Refinement	Extension
qwbarrierSCH$i	GaN	GaN.tcl	0.007	-3.00e15		(mbox 0.001 1.2 both)	
$end							
nbarrier	GaN	GaN.tcl	0.007	-3.00e15		(mbox 0.001 1.2 both)	

The syntax for the `repeat` command is:

```
$repeat <N>
layer1
layer2
$end
```

All lines between `$repeat` and `$end` are repeated `<N>` times. In each line, the string `$i` is replaced by the line number starting from 1 to `<N>`.

For example, after processing the CSV file consisting of the structure in Table 11 on page 31, Epi generates the structure in Table 12.

NOTE The region names in the region columns as well as the extension columns are numbered.

Table 12 Structure generated from Epi command file containing structure in Table 11

Region	Material	SourcePar File	Thickness	Doping	Mole Fraction	Refinement	Extension
pbarrierSCH	GaN	GaN.tcl	0.007	-3.00e15		(mbox 0.001 1.2 both)	
qwell1	InGaN	InGaN.tcl	0.001	-3.00e15	0.08	(yref qw)	Physics(region= qwell1) {Recombination (-Radiative) Active}
qwbarrierSCH1	GaN	GaN.tcl	0.007	-3.00e15		(mbox 0.001 1.2 both)	
qwell2	InGaN	InGaN.tcl	0.001	-3.00e15	0.08	(yref qw)	Physics(region= qwell2) {Recombination (-Radiative) Active}

Table 12 Structure generated from Epi command file containing structure in Table 11

Region	Material	SourcePar File	Thickness	Doping	Mole Fraction	Refinement	Extension
`qwbarrierSCH2`	`GaN`	`GaN.tcl`	`0.007`	`-3.00e15`		`(mbox 0.001 1.2 both)`	
`nbarrier`	`GaN`	`GaN.tcl`	`0.007`	`-3.00e15`		`(mbox 0.001 1.2 both)`	

References

[1] *Sentaurus Workbench User Guide*, Version A-2007.12, Mountain View, California: Synopsys, Inc., 2007.

[2] *Mesh Generation Tools User Guide*, Version A-2007.12, Mountain View, California: Synopsys, Inc., 2007.

[3] *Sentaurus Structure Editor User Guide*, Version A-2007.12, Mountain View, California: Synopsys, Inc., 2007.

[4] *Sentaurus Device User Guide*, Version A-2007.12, Mountain View, California: Synopsys, Inc., 2007.

[5] M. Levinshtein, S. Rumyantsev, and M. Shur, *Handbook Series on Semiconductor Parameters, Si, Ge, C (Diamond), GaAs, GaP, GaSb, InAs, InP, InSb*, vol. 1, Singapore: World Scientific, 1996.

CHAPTER 3 Using Sentaurus Structure Editor

This chapter discusses how Epi works with Sentaurus Structure Editor.

Overview

Epi creates the multilayer stack or wafer, whereas Sentaurus Structure Editor is used to modify the structure and to create the complete device that can be used for performing device simulation.

Figure 3 shows a multilayered GaAs solar cell structure, illustrating the differences in the structure created by Epi and Sentaurus Structure Editor.

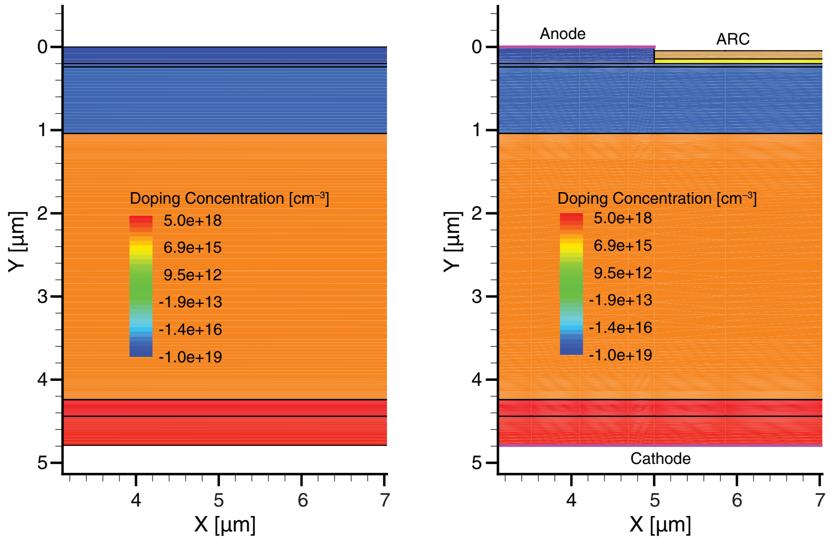


Figure 3 (*Left*) Multilayer stack from Epi and (*right*) device structure from Sentaurus Structure Editor

Scheme script files

In general, Sentaurus Structure Editor performs the following steps:

- The Scheme script file `n@node@_epi.scm`, created by Epi, contains commands for:
  - Creating the multilayer stack.
  - Creating each layer as rectangular regions.
  - Adding doping and mole-fraction profiles to the structure.
  - Performing regionwise mesh refinement in the y-direction for each layer.

  The file is sourced in the Sentaurus Structure Editor command file `sde_dvs.cmd` using the Scheme command:

  ```
  (load "n@node|epi@_epi.scm")
  ```

  After this command, all Epi variables are available to the Sentaurus Structure Editor command file as Scheme variables.
- Various device-processing steps such as etching and additional layer deposition can be performed using the Scheme variables `Y0_<region>` and `Y1_<region>`.

  `Y0_<region>` is the y-coordinate of the top edge of the region, and `Y1_<region>` is the y-coordinate of the bottom edge of the region.

  The file `n@node@_epi.scm` defines these variables for each region. For the solar cell structure in Figure 3 on page 35, the top GaAs layer is etched partially, and a double-layer antireflective coating is deposited.
- Electrical contacts are added to the multilayer stack.
- Mesh refinement commands for refinements in the x-direction and z-direction are added. For example, for the solar cell structure in Figure 3, a fine mesh is added under the contacts in the x-direction. In addition, a finer mesh is added in the y-direction in the top layer to resolve the optical generation profile.
- The complete device structure is generated using Sentaurus Mesh and is stored in the file `n@node@_msh.tdr`.

Epi creates the predefined Epi variables `Ytop` and `Ybot` (see Table 4 on page 11) and defines the corresponding Scheme variables `Ytop` and `Ybot` in `n@node@_epi.scm`. These Scheme variables can be used to define the top and bottom electrodes in a device. For example, a contact at the top and bottom of a device can be added with the following Scheme commands:

```
(sdegeo:define-2d-contact (find-edge-id (position (/ wtop 2) Ytop 0)) "anode")
(sdegeo:define-2d-contact (find-edge-id (position (/ wbottom 2) Ybot 0))
  "cathode")
```

where `wtop` is the width of the top layer, and `wbottom` is the width of the bottom layer.

CHAPTER 4 Using MatPar

This chapter describes how to use MatPar with Sentaurus Device.

Overview

Sentaurus Device requires a main parameter file that contains a list of references to other material parameter files.

A material parameter file is divided into several parameter sections. Each parameter section corresponds to a single material property or a group of related material properties. The dependency of a material property on physical variables such as temperature, doping concentration, mole fraction, and the wavelength of incident light is described by a physical model.

These models are represented usually by equations. The coefficients in these equations are called *model parameters*. The collection of model parameters for a particular model is called a *model parameter set*. A parameter section in the parameter file defines the values of model parameters for a specific model.

The model parameter set consisting of a specific value for each model parameter in the set is called a *model parameter set value*. For example, the `Bandgap` section in the parameter file corresponds to the band gap and electron affinity of a material. It contains model parameter set values for the Varshni bandgap model [1], which is the model implemented in Sentaurus Device. The Varshni model is described by:

$$E_g(T) = E_g(0) + \frac{\alpha T_{par}^2}{T_{par} + \beta} - \frac{\alpha T^2}{T + \beta} \qquad (1)$$

The Varshni bandgap model describes the variation of band gap with temperature. Here, band gap is a material property, and T is a physical variable. α, β, $E_g(0)$, and T_{par} are the model parameters of the Varshni model and are represented by `alpha`, `beta`, `Eg0`, and `Tpar` in the `Bandgap` section, respectively. A typical `Bandgap` section in the parameter file is:

```
Bandgap
{ * Eg = Eg0 + alpha Tpar^2 / (beta + Tpar) - alpha T^2 / (beta + T)
  * Parameter 'Tpar' specifies the value of lattice
  * temperature, at which parameters below are defined
  * Chi0 is electron affinity.
   Chi0 = 4.07        # [eV]
   Bgn2Chi = 0.5      # [1]
```

```
    Eg0 = 1.519         # [eV]
    alpha = 5.4050e-04  # [eV K^-1]
    beta = 2.0400e+02   # [K]
    Tpar = 0.0000e+00   # [K]
}
```

The set of model parameters {`Chi0, Bgn2Chi, Eg0, alpha, beta, Tpar`} is the complete model parameter set of the Varshni model. The set of values of these model parameters {`4.07, 0.5, 1.519, 5.4050e-04, 2.0400e+02, 0.0000e+00`} is the model parameter set value.

In this manual, the models implemented in Sentaurus Device are called *default models*, and the models not implemented in Sentaurus Device are called *new models*.

The equations for the default models are implemented in Sentaurus Device [2]. Sentaurus Device provides built-in default parameter values in the default material parameter files for many materials.

Sentaurus Device uses the default model parameter set to calculate the value of material properties using the default models for each parameter section. For example, the above `Bandgap` section defines the default parameter values for the default GaAs parameter file. To use a different set of model parameter values for device simulation, a new material parameter file can be created by redefining the default model parameter values. For example, a user-defined parameter file for GaAs can be created with a different set of model parameter values for the `Bandgap` section:

```
Bandgap
{ * Eg = Eg0 + alpha Tpar^2 / (beta + Tpar) - alpha T^2 / (beta + T)
  * Parameter 'Tpar' specifies the value of lattice
  * temperature, at which parameters below are defined
  * Chi0 is electron affinity.
   Chi0 = 4.07         # [eV]
   Bgn2Chi = 0.5       # [1]
   Eg0 = 1.517         # [eV]
   alpha = 5.5000e-04  # [eV K^-1]
   beta = 2.2500e+02   # [K]
   Tpar = 0.0000e+00   # [K]
}
```

The new material parameter file is called the user-defined material parameter file. This approach of creating user-defined material parameter files is referred to as the *Sentaurus Device approach*.

MatPar provides an additional approach – the MatPar approach – for creating user-defined parameter files. MatPar is used to implement the default models with either the default model parameter set value or model parameter set value obtained from the literature.

User-defined models can also be added to the Material Parameter Database. MatPar can create user-defined parameter files consisting of model parameter set values computed from new models or default models for all parameter sections.

Starting MatPar

The source file for MatPar is a Tcl script file, `mpr.tcl`, which is located in the `lib` subdirectory. The Tcl interpreter `gtclsh` is used to execute MatPar.

MatPar can be started either from the command line or Sentaurus Workbench. MatPar is driven by a command file with the default name `MatPar_mpr.cmd`.

To start MatPar from the command line, execute the following command from inside the `epi` subdirectory:

```
gtclsh ../lib/mpr.tcl <command_file>
```

The file `gtooldb.tcl` contains appropriate commands to launch MatPar automatically through the Scheduler when working inside Sentaurus Workbench.

The MatPar command file `MatPar_mpr.cmd` can be accessed directly from Sentaurus Workbench.

To open `MatPar_mpr.cmd` in SEdit:

- Right-click the MatPar icon in the tool row, and select **Edit Input** > **Commands**.

MatPar Components

MatPar consists of:

- Material Parameter Framework (see Material Parameter Framework on page 40).
- Material Parameter Database (see Material Parameter Database on page 41).
- Material Parameter Plotter (see Material Parameter Plotter on page 74).

The main steps to create user-defined parameter files using the MatPar approach are:

1. Model specification in Tcl source parameter files (see Source parameter files on page 52).
2. Model parameter set computation in model files (see Model files on page 58).
3. Parameter section printing in user-defined parameter files (see Parameter Initialization section and Parameter Printing section on page 56).

MatPar creates each user-defined regionwise or materialwise parameter file by processing a source parameter file (either `<material>.tcl` or `<material>.par`).

The `<material>.tcl` files are called Tcl source parameter files. There is one Tcl source parameter file for each material. These files contain user-specified model names for each parameter section to be printed in the user-defined parameter file. The model parameter set for a particular model is computed in model files (`<model>.tcl`) in the Material Parameter Database.

MatPar processes each Tcl source parameter file, computes the model parameter set from model files, and prints the parameter sections in the user-defined parameter files by invoking print procedures. If the source parameter file is a `<material>.par` file, the user-defined parameter file is a copy of the source parameter file. Detailed steps are discussed in How MatPar creates user-defined parameter files on page 73.

For a particular material and material property, different models as well as different model parameter set values for the same model can be implemented in the Material Parameter Database. The default models are also implemented.

In the MatPar approach for creating user-defined parameter files, the user-defined parameter files are created by MatPar from the user-specified model names in the Tcl source parameter file. You can switch easily between different models for a particular parameter section by changing the model names in the Tcl source parameter file.

Material Parameter Framework

The Material Parameter Framework provides a Tcl-based framework for:

- Creating user-defined material parameter files.
- Implementing material property models.
- Choosing models for a particular material property.
- Computing the model parameter values.
- Printing the user-defined parameter file.

These tasks are achieved by several procedures in the framework. It also includes a collection of Tcl source parameter files in which the model names used for creating the user-defined parameter files are specified. These files are described in Source parameter files on page 52.

All framework files are in the `lib` and `par` subdirectories (see Table 2 on page 2).

The file `lib/helper.tcl` contains helper procedures for the Material Parameter Framework and Epi. It contains several procedures that are useful for computing the material parameters.

Some of these procedures are discussed in Helper procedures for calculating model parameters/material properties on page 69.

The file `lib/helper.tcl` also includes `initProperty`, the procedure used to initiate the computation of material parameters. The procedures for printing the parameter sections in the user-defined parameter files are defined in `lib/printPar.tcl`. Both these procedures are discussed in Parameter Initialization section and Parameter Printing section on page 56.

The file `PhysicalConstants.tcl` contains definitions of fundamental physical constants, which are useful for computing material properties in some model files (see Input Validation and Header Printing section on page 54).

The file `compound_lib.tcl` contains procedures for interpolating material parameter/property values for compound semiconductors. These procedures are discussed in Interpolation procedures for ternary and quaternary compound semiconductors on page 72.

Material Parameter Database

The Material Parameter Database is a collection of model parameters from different references for the material properties of several materials. The model parameters are computed in model files. The directory of the Material Parameter Database is `pardb`.

Directory structure

The model files are located in a three-level directory structure inside `pardb`:

```
<material>/<propertyGroup>/<parameterSection>/<model>.tcl
```

This directory structure is shown in Figure 4 on page 43 and consists of:

- The `<material>` directory, which is named after the material.
- The `<propertyGroup>` and `<parameterSection>` subdirectories: Related parameter sections in the parameter file are grouped into several property groups. Table 13 on page 42 lists the names of the parameter sections for each property group. There are several `<parameterSection>` subdirectories under `<propertyGroup>` subdirectories.

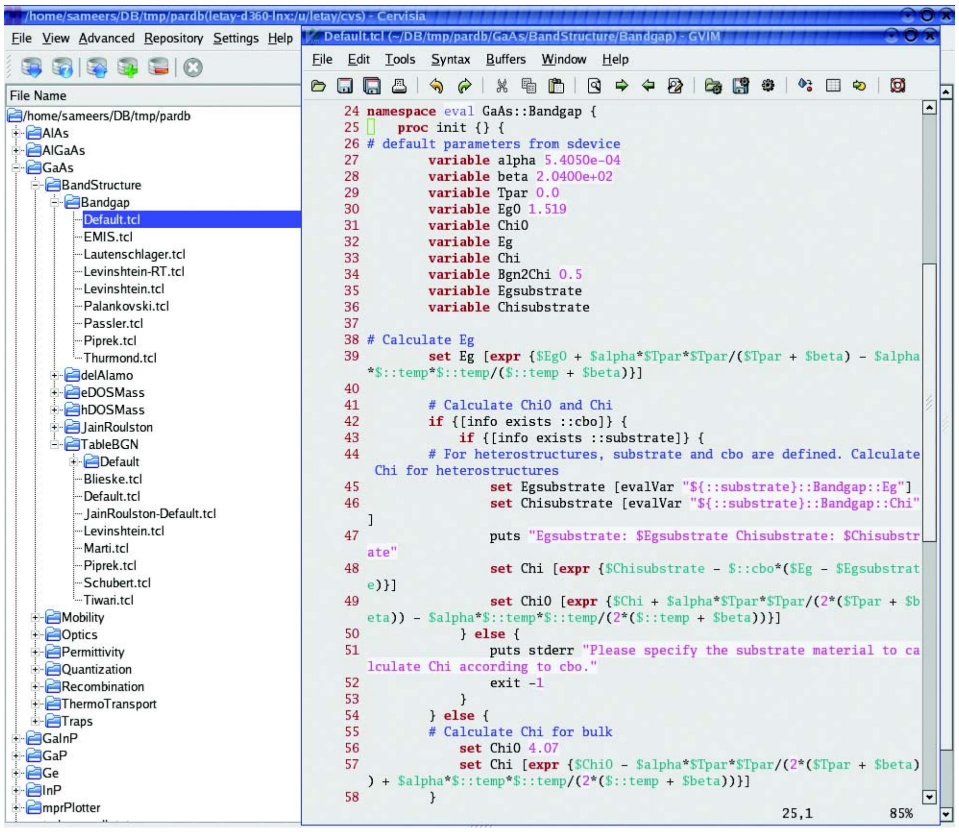
  The `parameterSection` subdirectories are named after the corresponding parameter section in the parameter file. For example, the parameter sections `Bandgap`, `TableBGN`, `eDOSMass`, and `hDOSMass` are grouped under a `propertyGroup` called `BandStructure`. Under every `<material>` directory, there is a directory called `BandStructure`, which contains the subdirectories `Bandgap`, `TableBGN`, `eDOSMass`, and `hDOSMass`.

NOTE Currently, only the parameter sections listed in Table 13 can be printed in the user-defined parameter file.

- The model files: For a particular material and property group, the `<parameterSection>` subdirectory can contain several model files that correspond to different models or different values of the parameter set for the same model. The names of these model files have the format `<model>.tcl`, where `model` is the model name (see Model files on page 58).

Table 13 Grouping of parameter sections in property group

Property group	Parameter section	Print procedure in printPar.tcl
`BandStructure`	`Bandgap`	`BandgapSection`
	`delAlamo`	`delAlamoSection`
	`eDOSMass`	`eDOSMassSection`
	`hDOSMass`	`hDOSMassSection`
	`JainRoulston`	`JainRoulstonSection`
	`TableBGN`	`TableBGNSection`
`Mobility`	`ConstantMobility`	`ConstantMobilitySection`
	`DopingDependence`	`DopingDependenceSection`
	`HighFieldDependence`	`HighFieldDependenceTEM2Section`
`Optics`	`FreeCarrierAbsorption`	`FreeCarrierAbsorptionSection`
	`RefractiveIndex`	`RefractiveIndexSection`
	`TableODB`	`TableODBSection`
`Permittivity`	`Epsilon`	`EpsilonSection`
`Quantization`	`SchroedingerParameters`	`SchroedingerParametersSection`
`Recombination`	`Auger`	`AugerSection`
	`BarrierTunneling`	`BarrierTunnelingSection`
	`RadiativeRecombination`	`RadiativeRecombinationSection`
	`Scharfetter`	`ScharfetterSection`
`ThermoTransport`	`Kappa`	`KappaSection`
	`LatticeHeatCapacity`	`LatticeHeatCapacitySection`
`Traps`	`PooleFrenkel`	`PooleFrenkelSection`


/home/sameers/DB/tmp/pardb(letay-d360-lnx:/u/letay/cvs) - Cervisia
File View Advanced Repository Settings Help
File Name
Default.tcl (~/DB/tmp/pardb/GaAs/BandStructure/Bandgap) - GVIM
File Edit Tools Syntax Buffers Window Help
25,1 85%

```
/home/sameers/DB/tmp/pardb
+ AlAs
+ AlGaAs
- GaAs
  - BandStructure
    - Bandgap
        Default.tcl
        EMIS.tcl
        Lautenschlager.tcl
        Levinshtein-RT.tcl
        Levinshtein.tcl
        Palankovski.tcl
        Passler.tcl
        Piprek.tcl
        Thurmond.tcl
    + delAlamo
    + eDOSMass
    + hDOSMass
    + JainRoulston
    - TableBGN
      + Default
        Blieske.tcl
        Default.tcl
        JainRoulston-Default.tcl
        Levinshtein.tcl
        Marti.tcl
        Piprek.tcl
        Schubert.tcl
        Tiwari.tcl
  + Mobility
  + Optics
  + Permittivity
  + Quantization
  + Recombination
  + ThermoTransport
  + Traps
+ GaInP
+ GaP
+ Ge
+ InP
+ mprPlotter
```

```
namespace eval GaAs::Bandgap {
    proc init {} {
# default parameters from sdevice
        variable alpha 5.4050e-04
        variable beta 2.0400e+02
        variable Tpar 0.0
        variable Eg0 1.519
        variable Chi0
        variable Eg
        variable Chi
        variable Bgn2Chi 0.5
        variable Egsubstrate
        variable Chisubstrate

# Calculate Eg
        set Eg [expr {$Eg0 + $alpha*$Tpar*$Tpar/($Tpar + $beta) - $alpha*$::temp*$::temp/($::temp + $beta)}]

        # Calculate Chi0 and Chi
        if {[info exists ::cbo]} {
            if {[info exists ::substrate]} {
        # For heterostructures, substrate and cbo are defined. Calculate Chi for heterostructures
                set Egsubstrate [evalVar "${::substrate}::Bandgap::Eg"]
                set Chisubstrate [evalVar "${::substrate}::Bandgap::Chi"]
                puts "Egsubstrate: $Egsubstrate Chisubstrate: $Chisubstrate"
                set Chi [expr {$Chisubstrate - $::cbo*($Eg - $Egsubstrate)}]
                set Chi0 [expr {$Chi + $alpha*$Tpar*$Tpar/(2*($Tpar + $beta)) - $alpha*$::temp*$::temp/(2*($::temp + $beta))}]
            } else {
                puts stderr "Please specify the substrate material to calculate Chi according to cbo."
                exit -1
            }
        } else {
        # Calculate Chi for bulk
            set Chi0 4.07
            set Chi [expr {$Chi0 - $alpha*$Tpar*$Tpar/(2*($Tpar + $beta)) + $alpha*$::temp*$::temp/(2*($::temp + $beta))}]
        }
```

Figure 4 Directory structure in pardb

Adding new models and new materials

New materials and models can be added in the Material Parameter Database for the parameter sections listed in Table 13 on page 42.

To add new models to existing materials in the Material Parameter Database:

1. Implement the model in a model file.
2. Add the model file to an appropriate `<parameterSection>` subdirectory under the corresponding material directory.

To add new models for new materials to the Material Parameter Database:

1. Create the corresponding Tcl source parameter file.
2. Add the `<material>` subdirectory under `pardb`.

   If Sentaurus Device does not recognize the material, follow the procedure for adding user-defined materials described in [2].
3. Add the `<propertyGroup>` and `<parameterSection>` subdirectories.

   At least, add the `<parameterSection>` subdirectories corresponding to the critical models [2].
4. Add the model files to the `<parameterSection>` subdirectories.

To create a model file, see Sections of model file on page 60.

Model documentation

The Material Parameter Database is documented in HTML format. The documentation summarizes the current lists of materials, property groups, parameter sections, and models in the database as well as describes the models. The header section in the model files provides the model description (see Figure 5 on page 45).

Adding new documentation

If new materials or models are added to the Material Parameter Database, the corresponding documentation can be added to the existing documentation:

- Right-click the MatPar icon in the tool row, and select **Edit** > **Refresh Documentation**.

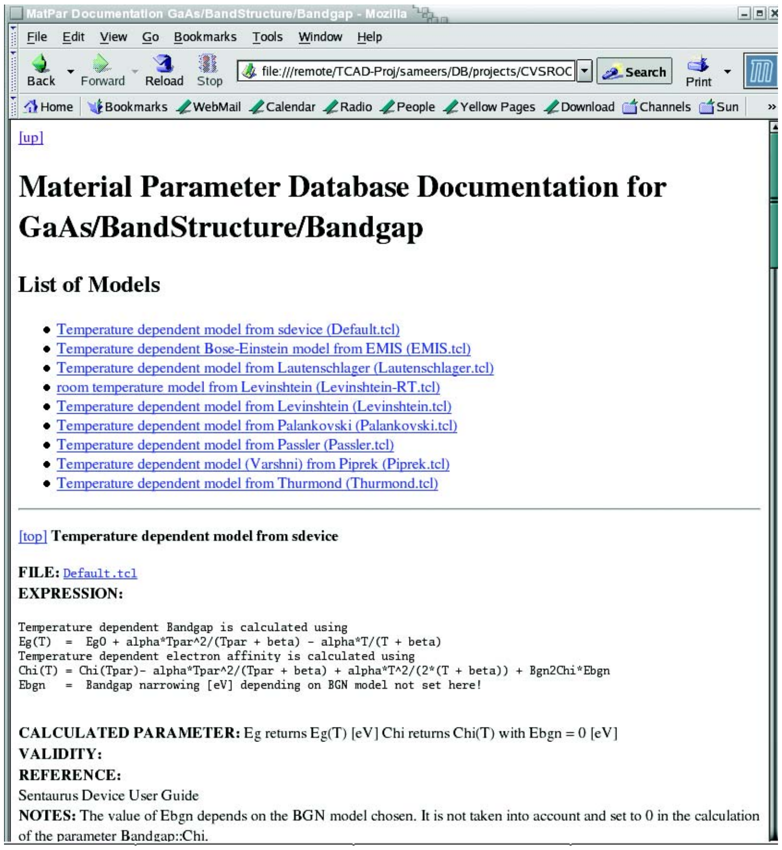
Viewing documentation

To view the documentation on UNIX, set the browser by adding the following line to the `$STDB/gpref_<user>.$STRELEASE` file:

```
genesis*WB_binaries.editor,html_browser:<browser>
```

To access the documentation from the MatPar node on UNIX:

- Right-click the MatPar icon in the tool row, and select **Edit** > **View Parameter Library (HTML)**.


MatPar Documentation GaAs/BandStructure/Bandgap - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
file:///remote/TCAD-Proj/sameers/DB/projects/CVSROC
Search
Print
Home Bookmarks WebMail Calendar Radio People Yellow Pages Download Channels Sun

[up]

Material Parameter Database Documentation for GaAs/BandStructure/Bandgap

List of Models

- Temperature dependent model from sdevice (Default.tcl)
- Temperature dependent Bose-Einstein model from EMIS (EMIS.tcl)
- Temperature dependent model from Lautenschlager (Lautenschlager.tcl)
- room temperature model from Levinshtein (Levinshtein-RT.tcl)
- Temperature dependent model from Levinshtein (Levinshtein.tcl)
- Temperature dependent model from Palankovski (Palankovski.tcl)
- Temperature dependent model from Passler (Passler.tcl)
- Temperature dependent model (Varshni) from Piprek (Piprek.tcl)
- Temperature dependent model from Thurmond (Thurmond.tcl)

[top] **Temperature dependent model from sdevice**

FILE: `Default.tcl`
EXPRESSION:

```
Temperature dependent Bandgap is calculated using
Eg(T)  =  Eg0 + alpha*Tpar^2/(Tpar + beta) - alpha*T/(T + beta)
Temperature dependent electron affinity is calculated using
Chi(T) = Chi(Tpar)- alpha*Tpar^2/(Tpar + beta) + alpha*T^2/(2*(T + beta)) + Bgn2Chi*Ebgn
Ebgn   =  Bandgap narrowing [eV] depending on BGN model not set here!
```

CALCULATED PARAMETER: Eg returns Eg(T) [eV] Chi returns Chi(T) with Ebgn = 0 [eV]
VALIDITY:
REFERENCE:
Sentaurus Device User Guide
NOTES: The value of Ebgn depends on the BGN model chosen. It is not taken into account and set to 0 in the calculation of the parameter Bandgap::Chi.

Figure 5 Example of documentation for the GaAs bandgap models

MatPar command file

The command file for MatPar, `MatPar_mpr.cmd`, is a Tcl script file. It consists of some mandatory and some optional commands. These commands define a few variables and invoke some procedures as described here.

MatPar ignores character strings starting with #. Therefore, the # can be used to insert comments in the command file.

Defining variables in command file

Table 14 summarizes the variables that can be defined in the MatPar command file. MatPar does not define any default values for these variables. All variables are available to the Material Parameter Framework and can be accessed from both Tcl source parameter files and model files.

Table 14 Variables defined in MatPar command file

Variable	Description	Unit	Example
`lambda`	Wavelength of the incident light on the device.	µm	`0.5`
`parameterOutputFile`	The main parameter file required by Sentaurus Device. Must be defined.	–	`n@node@_mpr.par`
`pardb`	Location of the Material Parameter Database.	–	`"../pardb"`
`substrate`	Reference material for calculating electron affinity values for heterostructures.	–	`GaAs`
`temp`	Operating temperature of the device. Must be defined.	K	`300`

Variables such as the device-operating temperature, wavelength of incident light on the device, and reference material for electron affinity can be defined in the MatPar command file using the Tcl command `set`:

```
set temp <value>
set lambda <value>
set substrate <material_name>
```

NOTE Since many models are temperature dependent, it is mandatory to define `temp`. The Sentaurus Workbench project aborts if `temp` is not defined.

Sentaurus Workbench variables can also be accessed in `MatPar_mpr.cmd`. This feature can be used to pass the Sentaurus Workbench variables to the Tcl source parameter files and model files.

For heterostructure simulations, the reference material for the electron affinity and conduction-band offset (cbo) value of each material with respect to the reference material may be specified (see Simulating band structure of heterostructures on page 63).

In MatPar, this reference material is specified using the Tcl variable `substrate` in the MatPar command file, and the cbo value for each material is specified in the corresponding Tcl source parameter file using the Tcl variable `cbo`.

The syntax for specifying the reference material is:

```
set substrate <material>
```

Usually, the substrate material is chosen as the reference material. For the structure defined in Table 6 on page 14, Ge can be defined as the reference material:

```
set substrate Ge
```

In this case, the electron affinity of AlGaAs and GaAs are calculated using the `cbo` values of AlGaAs and GaAs with respect to Ge, respectively. The `cbo` values of the materials AlGaAs and GaAs are specified in the respective Tcl source parameter files `AlGaAs.tcl` and `GaAs.tcl`.

If `substrate` is defined, the corresponding `<material>.tcl` file is always processed first, so that the electron affinity and band gap of the substrate material are computed by MatPar and are available for the calculation of the electron affinity value of each material in the stack.

In the above example, `Ge.tcl` is processed before processing the other source parameter files. For the definition of cbo and electron affinity calculation, see Simulating band structure of heterostructures on page 63.

`parameterOutputFile` defines the name of the main parameter file that contains all of the regionwise or materialwise material parameter files. It is mandatory to define this variable since it is required by Sentaurus Device, and the syntax is:

```
set parameterOutputFile <Main_parameter_file>
```

The location of the Material Parameter Database is defined using the variable `pardb`, which is specified using the Tcl command `set`:

```
set ::pardb <location_of_pardb_directory>
```

Since `pardb` must be defined in the global namespace[1], it is necessary to provide the '::' prefix. The location of the `pardb` subdirectory is defined with respect to the `epi` subdirectory. Usually, `pardb` and `epi` are at the same level, therefore, `pardb` is defined by:

```
set ::pardb "../pardb"
```

In addition to defining the above variables, `n@node@_epi.tcl`, which contains all of the information about the layered structure, is sourced in the MatPar command file:

```
source "./n@node|epi@_epi.tcl"
```

1. Namespaces provide scope for procedures and global variables. A namespace is a collection of procedures and variables. Namespaces ensure that the procedures and variables of a particular namespace do not interfere with the procedures and variables of other namespaces. By default, all Tcl variables are defined in the global namespace of Tcl. The global namespace also contains all global variables and commands. For a detailed discussion, refer to the literature [3].

With this file, MatPar has information as to which source parameter file to process and the name of the corresponding user-defined parameter files. In addition, all of the variables defined in Epi and MatPar are available to the Material Parameter Framework as Tcl variables and can be accessed from both the Tcl source parameter files and model files.

Referencing Sentaurus Workbench variables

The Sentaurus Workbench variables can be referenced in the MatPar command file, Tcl source parameter files, and model files using two methods:

- @-substitution method
- gparam method

The @-substitution method

The usual Sentaurus Workbench syntax can be used in `MatPar_mpr.cmd` as follows:

```
set <Tcl_variable> <@SWB_variable@>
```

This converts the Sentaurus Workbench variable `SWB_variable` into a Tcl variable `Tcl_variable`, which can be referenced in all Tcl source parameter files and model files. For example, the Sentaurus Workbench variable `temperature` can define the Tcl variable `temp` in `MatPar_mpr.cmd` as follows:

```
set temp @temperature@
```

When `temp` is defined in the MatPar command file, it is accessible to all Tcl source parameter files and all model files.

The gparam method

The procedure `exportParams` is defined in the Material Parameter Framework. If the `exportParams` procedure is invoked in `MatPar_mpr.cmd`, Sentaurus Workbench variables can be referenced in Tcl source parameter files and model files using the format:

```
$gparam(<SWB_variable>)
```

`exportParams` is invoked from the MatPar command file using the statement:

```
exportParams @node@
```

where `@node@`, as usual, indicates the node number of MatPar in Sentaurus Workbench.

For example, if a parameter called `model` is defined in Sentaurus Workbench and its value equals `Levinshtein`, it can be referenced in any Tcl source parameter file as follows:

```
initProperty $material Permittivity Epsilon $gparam(model)
```

The syntax of the `initProperty` procedure is discussed in Source parameter files on page 52.

In the Material Parameter Framework, the @-substitution method also works directly in `<material>.tcl` files. Instead of using the gparam method in the above example, the following command can be used:

```
initProperty $material Permittivity Epsilon @model@
```

Inserting Extension columns in Sentaurus Device command file

When `n@node@_epi.tcl` is sourced in `MatPar_mpr.cmd`, the content of all columns in the `Extension` field can be accessed from `MatPar_mpr.cmd` using the following Tcl commands in `MatPar_mpr.cmd`:

```
foreach key [getLayerKeyList "region,*,<extensionColumnName1>
   <extensionColumnName2> ..."] {
      loadLayerVariables $key
      <user-defined commands: $<extensionColumnName1>
      <user-defined commands: $<extensionColumnName2>
      ...
}
close $FID
```

The `foreach` Tcl command loops through all region layers for which the extension columns, `<extensionColumnName1>`, `<extensionColumnName2>`, and so on are defined. The command:

```
loadLayerVariables $key
```

creates a variable with the same name as the name of the extension column, so that the content of the extension columns can be accessed with the Tcl syntax `$<extensionColumnName>` and can be manipulated with user-defined commands:

```
<user-defined commands>
```

For example, to write the content of a single extension column with the name extension in the file `n@node@_extension.cmd`, use:

```
set FID [open "n@node@_extension.cmd" w]
foreach key [getLayerKeyList "region,*,extension"] {
   loadLayerVariables $key
   puts $FID "$extension"
}
close $FID
```

The content of the `n@node@_extension.cmd` file can be inserted in the preprocessed Sentaurus Device command file by using the `Insert` command in the Sentaurus Device command file `sdevice_des.cmd`:

```
Insert= "n@node|Matpar@_extension.cmd"
```

The complete procedure for inserting arbitrary content into the Sentaurus Device command file is illustrated with the following examples.

Example 1: Heterostructure

In Table 6 on page 14, the emitter and base regions contain the statement `Recombination(Radiative)` in the Extension column. Regionwise `Physics` statements for emitter and base regions can be inserted in the Sentaurus Device command file by using the following procedure:

1. Add Sentaurus Device after the MatPar node in the tool row in the `epi` project.
2. Add this command to the Global section of the Epi command file `epi_epi.csv`:

   ```
   $global append columnNames extension
   ```

3. Add this code to the MatPar command file `MatPar_mpr.cmd` after sourcing `n@node@_epi.tcl`:

   ```
   set FID [open "n@node@_extension.cmd" w]
   foreach key [getLayerKeyList "region,*,extension"] {
      loadLayerVariables $key
      puts $FID "Physics(Region=\"$region\") \{$extension\}"
   }
   close $FID
   ```

   After execution of the MatPar node, if the MatPar node number is `9`, MatPar creates the file `n9_extension.cmd`.

4. Add this line to the Sentaurus Device command file `sdevice_des.cmd`:

   ```
   Insert= "n@node|MatPar@_extension.cmd"
   ```

5. Preprocess the Sentaurus Workbench project.

   The following statement appears in the preprocessed command file:

   ```
   Insert= "n9_extension.cmd"
   ```

6. Run the Sentaurus Workbench project.

During run-time, Sentaurus Device uses the following regionwise `Physics` statements:

```
Physics(Region="emitter") {Recombination(Radiative)}
Physics(Region="base") {Recombination(Radiative)}
```

Example 2: Multiple–quantum well structure

To insert regionwise `Physics` statements for a multiple–quantum well structure shown in Table 11 on page 31:

1. Add Sentaurus Device after the MatPar node in the tool row in the `epi` project.
2. Add this command to the Global section of the Epi command file `epi_epi.csv`:

   ```
   $global append columnNames extension
   ```

3. Add this code to the MatPar command file `MatPar_mpr.cmd` after sourcing `n@node@_epi.tcl`:

   ```
   set FID [open "n@node@_extension.cmd" w]
   foreach key [getLayerKeyList "region,*,extension"] {
      loadLayerVariables $key
      puts $FID "$extension"
   }
   close $FID
   ```

   After execution of the MatPar node, if the MatPar node number is `9`, MatPar creates the file `n9_extension.cmd`.

4. Add this line to the Sentaurus Device command file `sdevice_des.cmd`:

   ```
   Insert= "n@node|MatPar@_extension.cmd"
   ```

5. Preprocess the Sentaurus Workbench project.

   The following statement appears in the preprocessed command file:

   ```
   Insert= "n9_extension.cmd"
   ```

6. Run the Sentaurus Workbench project.

During run-time, Sentaurus Device uses the following regionwise `Physics` statements:

```
Physics(region=qwell1){Recombination(-Radiative) Active}
Physics(region=qwell2){Recombination(-Radiative) Active}
```

Source parameter files

MatPar uses the source parameter files to create the user-defined material parameter files. The names of these files are specified in the Epi command file (see Material and Interface section on page 19).

There are two types of source parameter file:

- The `<material>.tcl` files are Tcl source parameter files and contain Tcl commands enclosed in a single Tcl command block[1]. MatPar processes these files to create the user-defined material parameter files.
- The `<material>.par` files have the same format as Sentaurus Device parameter files. A `<material>.par` file can be either a Sentaurus Device default parameter file or a user-defined parameter file created using the Sentaurus Device approach. MatPar includes these files without processing in the user-defined material parameter files.

NOTE There can be only one `<material>.tcl` and one `<material>.par` file for each material or region.

All source parameter files reside in the `par` subdirectory. The material name in the `<material>.tcl` and `<material>.par` files follows the standard convention for naming materials. For example, the Tcl source parameter files corresponding to germanium, gallium arsenide, and aluminum gallium arsenide are `Ge.tcl`, `GaAs.tcl`, and `AlGaAs.tcl`.

Tcl source parameter files

All Tcl source parameter files in the Sentaurus Workbench project can be directly accessed from Sentaurus Workbench.

To open all `material.tcl` files simultaneously in a text editor:

- Right-click the MatPar icon in the tool row, and select **Edit Input** > **Edit All material.tcl Files**.

All `material.tcl` files are loaded in the text editor, but only one `material.tcl` file opens in the text editor. The other files are listed in the **File** menu of the editor and are opened by selecting **File** > **<path/material.tcl>**.

1. A Tcl command block consists of an arbitrary set of Tcl command lines that are delimited by "!(" and ")!" [4].

The Tcl source parameter files have four sections:

- Variable Definition section: Defines the variables `material` and `cbo`.
- Input Validation and Header Printing section: Validates the value of variables, and prints the header section in user-defined parameter files.
- Parameter Initialization section: Calculates the model parameter set value for each parameter section.
- Parameter Printing section: Prints the parameter section in user-defined parameter files.

Variable Definition section

The Variable Definition section defines the two variables `material` and `cbo` (see Table 15) using the `set` Tcl command.

Table 15 Variables and procedures used in Variable Definition section

Variable/Procedure	Description
`cbo`	Conduction-band offset value with respect to reference material for electron affinity.
`material`	Name of material corresponding to the `material.tcl` file. Mandatory variable.
`initProperty <material> <propertyGroup> <parameterSection>`	This procedure calculates the model parameters for the specified material, parameter section, and model.
`${material}::<parameterSection>::print`	This procedure prints the specified parameter section in the user-defined material parameter file.

The variable `cbo` is defined if the multilayer stack consists of a heterostructure and if the `<material>.tcl` file does not correspond to the reference material for electron affinity. For example, for a GaAs/Ge heterostructure, if Ge is the reference material for electron affinity and the cbo of GaAs with respect to Ge is 0.35, the Variable Definition section of `GaAs.tcl` is:

```
set material GaAs
set cbo 0.35
```

and the Variable Definition section of `Ge.tcl` is:

```
set material Ge
```

The reference material and cbo are defined in Simulating band structure of heterostructures on page 63.

Input Validation and Header Printing section

This section comprises two mandatory commands:

```
source ../lib/printPar.tcl
printPar::header
```

Paths to all directories and files in the Tcl source parameter file are specified relative to the `epi` subdirectory. The `printPar.tcl` file resides in the `lib` subdirectory. The file `PhysicalConstants.tcl` is sourced in `printPar.tcl` and contains the definitions of several fundamental physical constants such as Planck's constant, the speed of light, the Boltzmann constant, and electron charge. It can also be used to define miscellaneous constants such as π. These constants can be referenced from all Tcl source parameter files and model files. These constants are used by some model files to compute material properties.

The file `printPar.tcl` also validates the value of the physical variables `temp`, `cbo`, `xMole`, and `yMole`:

- If the device temperature is not defined in `MatPar_mpr.cmd` or if it is defined to be negative, the project aborts.
- If `cbo` is defined in the Tcl source parameter file and is less than 0 or greater than 1, the project aborts.
- For ternaries, `xMole` is defined in the CSV file of Epi. If `xMole` is less than 0 or greater than 1, the project aborts.
- For quaternaries, both `xMole` and `yMole` are defined in the CSV file. If `xMole` and `yMole` are each less than 0 or greater than 1, or if $\texttt{xMole} + \texttt{yMole} \neq 1$, the project aborts.

The user-defined parameter files consist of a header section followed by several parameter sections. The header section contains values of the variables:

- `material`
- `temp`
- `substrate`
- `cbo`
- `doping`
- `xMole`
- `yMole`

Of these variables, `material`, `cbo`, `doping`, `xMole`, and `yMole` are defined for each layer in the CSV file of Epi.

The variables `temp` and `substrate` are defined for the complete device in `MatPar_mpr.cmd`. If any of these variables is not defined, the string `"--"` is printed as the value of the variable in the header section.

The following is an example of the header section for a user-defined GaAs parameter file corresponding to a GaAs layer with doping concentration of 1×10^{18} cm^{-3}:

```
***************************************************
* Material parameter file for GaAs
* Temperature                : 300 [K]
* Substrate                  : --
* conduction band offset ratio: -- [1]
* Doping concentration       : 1e18 [cm-3]
* Mole fraction x            : -- [1]
* Mole fraction y            : -- [1]
**************************************************
```

For the doping and x–mole fraction profiles, the default value is printed. All lines in the header section are commented out in the user-defined parameter file.

Print procedures

The file `printPar.tcl` contains procedures for printing the header section and various parameter sections in the user-defined parameter files. All procedures are defined in the namespace `printPar` in the file `printPar.tcl`.

To print the header section, use the procedure:

```
printPar::header
```

To print the parameter section, use the procedure:

```
printPar::<ParameterSection>Section
```

For example, to print the Epsilon section in the user-defined parameter file, use:

```
printPar::EpsilonSection
```

Table 13 on page 42 lists the names of all parameter sections for which a print procedure is defined in `printPar.tcl` and the name of the print procedure for each parameter section.

Some models offer a choice between different representations or formulas for the same (or equivalent) physical parameters, for example, effective mass or density-of-states. A particular representation is selected by using the parameter `Formula` [2]. In such cases, the model parameter set of a particular parameter section and, therefore, the name of the print procedure is different for different formulas.

For example, the procedure `printPar::HighFieldDependenceTEM2Section` prints the `HighFieldDependence` parameter section with model parameters for the transferred electron model and the velocity saturation model corresponding to Model 2 [2].

Parameter Initialization section and Parameter Printing section

The Parameter Initialization section consists of a group of `initProperty` statements, followed by the Parameter Printing section, which contains a group of print procedures. There is exactly one `initProperty` statement and one corresponding print procedure for each parameter section to be printed in the parameter file:

```
initProperty <material> <propertyGroup> <parameterSection> <model>
${material}::<parameterSection>::print
```

Printing a particular parameter section in a user-defined parameter file requires two main steps:

1. Computing the model parameter set value.
2. Printing the parameter section along with the computed value of the model parameters in the user-defined parameter file.

The execution of the `initProperty` statement in a Tcl source parameter file initiates Step 1; whereas, the execution of the `print` statement in the Tcl source parameter file initiates Step 2 as discussed below.

The `initProperty` procedure sources the file `pardb/<material>/<propertyGroup>/<parameterSection>/<model>.tcl` and:

- The model parameter set for the corresponding parameter section is computed in the namespace `${material}::<parameterSection>`.
- The procedure for printing the parameter section, `print`, which is defined in the namespace `${material}::<parameterSection>` in the `<model>.tcl` file, is initialized.

The procedure `${material}::<parameterSection>::print` prints the parameter section along with the computed value of the model parameters as follows:

- Executing `${material}::<parameterSection>::print` invokes the print procedure defined in the namespace `${material}::<parameterSection>` in the model file. Usually, this print procedure is called `printPar::<ParameterSection>Section`.
- Executing `printPar::<ParameterSection>Section` from the `<model>.tcl` file invokes the `<ParameterSection>Section` procedure in the `printPar` namespace defined in the `printPar.tcl` file, which finally prints the parameter section along with the computed model parameter set in the user-defined parameter file.

For example, the statements:

```
set material GaAs
initProperty GaAs BandStructure Bandgap Levinshtein
${material}::Bandgap::print
```

calculate the Levinshtein parameter set for the GaAs `Bandgap` section and print the `Bandgap` section in the user-defined GaAs parameter file using the procedure `GaAs::Bandgap::print`. The procedure `GaAs::Bandgap::print` invokes the procedure `printPar::BandgapSection`, defined in `printPar.tcl`.

NOTE In the case of `<material>.par` source parameter files, the `<material>.par` file contains the value of model parameters for all parameter sections. The source parameter file is included as is in the user-defined materialwise parameter file.

Using parameter sections in Tcl source parameter files

Instead of using a pair of `initProperty` and `print` statements to print parameter sections in a user-defined parameter file, you can explicitly add the parameter section outside the Tcl command block in a Tcl source parameter file. The parameter section is printed as is in the user-defined parameter file. For example, the following listing in the `GaAs.tcl` file prints both the `Bandgap` and `Epsilon` sections:

```
!(
set material GaAs
initProperty GaAs BandStructure Bandgap Levinshtein
${material}::Bandgap::print
)!

Epsilon
{ * Ratio of the permittivities of material and vacuum

  * epsilon() = epsilon
   epsilon = 13.18   # [1]
}
```

As explained in Run-time and preprocessed models on page 67, for regions with doping or mole-fraction profiles, the parameter section for the default models must be included explicitly in the Tcl source parameter file. This ensures that Sentaurus Device computes the correct material property values during run-time.

NOTE Doping, mole-fraction, and temperature profiles are not supported for new models.

Template Tcl source parameter file

The `epi/par` subdirectory includes a template file `material_template.tcl`, which includes all of the abovementioned sections of the Tcl source parameter file. It contains `initProperty` statements and calls to print procedures for all parameter sections for which a print procedure is defined in `printPar.tcl` (see Table 13 on page 42).

You can use the `material_template.tcl` file as a template to create a customized Tcl source parameter file, `<material>.tcl`, for any material by replacing all instances of `<...>` in `material_template.tcl` with the appropriate numbers or character strings.

The following list shows part of the `material_template.tcl` file:

```
set material <material>
set cbo <cbo>
source ../lib/printPar.tcl
printPar::header
initProperty $material Permittivity Epsilon <model>
${material}::Epsilon::print
```

Model files

A particular model file in the Material Parameter Database corresponds to a particular material property model with a model parameter set value. All model files are Tcl script files.

A model file:

- Implements a default model or a new model.
- Computes the model parameter set corresponding to the default model for a particular parameter section and material.
- Defines the print procedure for the corresponding parameter section.

Some model files also compute the relevant material property in order to print it in the user-defined parameter file and to visualize its variation as a function of a physical variable in the Material Parameter Plotter.

Model types and naming model files

The name of model files in the Material Parameter Database has the general format `<model>.tcl`, where `model` is the name of the model file. The guidelines used to name the model files in the Material Parameter Database are presented here. As discussed in Material

Parameter Database on page 41, model files are located in the Material Parameter Database in the directory `<parameterSection>`:

```
pardb/<material>/<propertyGroup>/<parameterSection>/<model>.tcl
```

The following model types are implemented in the model files:

- *Equation-based* models where the dependency of a material property on physical variables is expressed in terms of equations.
- *Table-based* models where the dependency of a material property on a particular physical variable is expressed in terms of tabular data.

For example, the Jain–Roulston bandgap narrowing model is an equation-based model, and the TableBGN bandgap narrowing model is a table-based model.

In the literature of compound semiconductors, for a particular material property of a material:

- Various authors have published different values for the parameter set of a particular model.
- Different models exist.

Both of these possibilities can be implemented in the model files. The model parameter set for a particular parameter section of the material parameter file corresponds to the default model for the corresponding material property and is called the *default parameter set*. All model files compute a value for this default parameter set. A particular model file implements one of the following model types:

- A *default model* for a particular material property is a model that has been implemented in Sentaurus Device. If a model file corresponding to the default model computes the default model parameter set using values from the default material parameter file, the model file is called `Default.tcl`.

  If the default model parameter set is computed using the value of the default model parameter set from another source, the model file is named after the first author of the reference: `<FirstAuthor>.tcl`. For example, the model file `Default.tcl` in the `Bandgap` folder for GaAs computes the default model parameter set using the parameter `set` from the `Bandgap` section of the default parameter file. The model file `Lautenschlager.tcl` in the same folder computes the default model parameter set from the model parameter set from Lautenschlager [5]. For details, see Examples of model implementation on page 65.

- A *new model* for a particular material property is a model that has not been implemented in Sentaurus Device. In this case, the new model parameter set is not the same as the default model parameter set. A model file for a new model implements the new model and computes the default model parameter set in terms of the model parameters of the new model. For example, the model file `Passler.tcl` in the `Bandgap` folder for GaAs computes the default model parameter set using model parameter set from Pässler [6]. For details, see Examples of model implementation on page 65.

If a particular reference discusses different models for a particular material property and material, the model file names can be `<FirstAuthor>-<modelname>.tcl`. If a reference discusses only one model, the corresponding model file can be `<FirstAuthor>.tcl`.

For a few properties, the parameter section can contain different parameter sets depending on the choice of submodels. In such a case, the model file can be named with an appropriate combination of the model names. For example, the `HighFieldDependence` section corresponds to the high-field mobility models and contains different parameter sets depending on the choice of various submodels. The name of the model file that implements the default transferred electron model with the velocity saturation model called Model 2 is `Default-TEM2.tcl`.

In addition, you can implement some equation-based default models in terms of table-based models. For example, the Jain–Roulston model for GaAs has been implemented in the form of a table-based model, and the model file is:

```
pardb/GaAs/Bandgap/TableBGN/Default-JainRoulston.tcl
```

For some material properties, multiple models are implemented in Sentaurus Device, that is, there are multiple default models. In such cases, there can be more than one way of implementing a new model using the default models. In this case, the model file name is `<FirstAuthor>-<DefaultModel>.tcl`.

For example, the Arora and Masetti models are examples of doping-dependent mobility models. They are also default models since they are implemented in Sentaurus Device. The mobility model discussed in a paper by Sotoodeh has been implemented for GaAs in the Material Parameter Database in terms of both the Arora model and the Masetti model, and the model files are called `Sotoodeh-Arora.tcl` and `Sotoodeh-Masetti.tcl`, respectively.

Sections of model file

The computation of the default model parameter set as well as the printing of the parameter sections are through Tcl commands.

A model file contains:

- A header section, which is used to document the model.
- A model implementation and printing section, which is used to compute the default model parameter set and to define or invoke the print procedure for the parameter section in the user-defined parameter file.

Header section

The header section contains several fields for documentation purposes (see Table 16). Most fields can be accessed in the documentation of the Material Parameter Database. Each line in the header section begins with `#!`. A field and its content can be written in either of the following ways:

- A line contains the field name, followed by the field content. For example:

```
#! <Field>: <Content of the field>
```

- A line contains the field name, followed by the field content on subsequent lines. For example:

```
#! <Field>:
#! <Content of the field>
#! <Content of the field>
```

Table 16 Fields in header section of model file

Field	Description
`CALCULATED PARAMETER`	Calculated parameters or material property or both.
`EXPRESSION`	Equation used to calculate the model parameters or material property or both.
`MATERIAL`	Name of the material.
`MODEL`	Brief description of the model.
`NOTES`	Recommendations on using the model or further explanation of the model.
`PROPERTY`	Name of the material property corresponding to the parameter section.
`REFERENCE`	Source of the model.
`SDEVICE`	Syntax for activating the model in Sentaurus Device. This field is empty if the model does not require explicit activation.
`VALIDITY`	Conditions under which the model is valid. If the reference does not specify the validity, this field is empty.

By default, there is no empty line in the HTML documentation of the content of the field. If there are several lines of content, the documentation can be made more readable by adding `<pre>` on a separate line between two lines of content.

As a result, an empty line is inserted between the two subsequent lines. For example, as a result of the following header section, an empty line is inserted in the documentation between the lines starting with `Temperature` and `Eg(T)`:

```
#! EXPRESSION
#! Temperature-dependent band gap is calculated using
#! <pre>
#! Eg(T) = Eg0 + alpha*Tpar^2/(Tpar + beta) - alpha*T/(T + beta)
```

Some models in Sentaurus Device are activated by a specific command in the Sentaurus Device command file. The `SDEVICE` field specifies the syntax of this command. If the model does not need to be activated, this field is empty.

Model implementation and printing section

The model implementation and printing section is used to calculate the default model parameter set and material property, and to define the print procedure to be used for the parameter section in the user-defined parameter file. It contains the namespace `<material>::<parameterSection>` in which two procedures `init` and `print` are defined. For example, all model files that compute the model parameter set for the `Bandgap` section define `init` and `print` procedures in the namespace `GaAs::Bandgap`.

The calculation of the model parameter set is implemented in the `init` procedure, which returns the default model parameter set and, for some parameter sections, the value of the relevant material property.

Sentaurus Device uses the default model and the default model parameter set values to calculate the value of the relevant material property. In addition to this, the calculation of material property value is implemented in some model files for the purpose of printing it in a commented line in the parameter section.

For such parameter sections, the value of the material property can be obtained without performing the device simulation. In addition, these material properties can be plotted using the Material Parameter Plotter. All model parameters and property values to be printed in the user-defined parameter file must be declared using the Tcl command `variable`:

```
variable <modelParameter>
variable <materialProperty>
```

For example, in the `GaAs::Bandgap` namespace, the default model parameter set and the material property are declared using the `variable` command as follows:

```
# Default model parameter set
variable alpha
variable beta
variable Tpar
variable Eg0
variable Bgn2Chi
variable Chi0

# Material properties
variable Chi
variable Eg
```

The value of these parameters is computed in the rest of the code in the corresponding model file.

The `print` procedure contains a single statement of the form:

```
printPar::<ParameterSection>Section
```

This invokes the appropriate procedure for printing the parameter sections, which are defined in `printPar.tcl`.

The last line of the model file executes the `init` procedure in the namespace `<material>::<parameterSection>`:

```
<material>::<parameterSection>::init
```

When a model file is sourced through a call to the `initProperty` procedure, the execution of the last line in the model file results in the computation of the default model parameter set. When the `print` procedure is executed in the Tcl source parameter file, it invokes the print procedure in the namespace `<material>::<parameterSection>`, which then prints the parameter section.

Simulating band structure of heterostructures

Sentaurus Device uses the electron affinity model [7] to calculate the band diagram of a heterostructure. According to this model, the conduction-band discontinuity (CBO) between the two semiconductors in a heterostructure is given by:

$$\Delta E_C = -\Delta\chi \tag{2}$$

and the valence-band discontinuity (VBO) is given by:

$$\Delta E_V = \Delta E_g - \Delta E_C \tag{3}$$

where:

- χ is electron affinity of a material.
- E_g is the band gap of a material.
- E_C is the conduction-band edge, E_V is the valence-band edge, and ΔE denotes the difference in these energies.

Sentaurus Device uses the electron affinity (`Chi0`) and bandgap (`Eg0`) values of the materials from the `Bandgap` section in the corresponding material parameter files of the materials in the heterostructure.

Experimental measurements of the band discontinuities in a heterostructure may differ from the values obtained from the electron affinity model. In this case, to simulate the correct band diagram of the heterostructure, one of the materials is chosen as a reference material, and the electron affinity value of the other material is modified using:

$$\chi = \chi_{\mathrm{ref}} - \Delta E_{\mathrm{C}}(\mathrm{measured}) \tag{4}$$

where χ_{ref} is the electron affinity of the reference material, and χ is the electron affinity of the other material. To simulate the correct band diagram, the value of `Chi0` in the reference material parameter file is not changed; whereas, the value of `Chi0` in the material parameter file of the other material is modified to the value obtained from Eq. 4.

For many heterostructures, experimental measurements of the conduction-band offset (cbo) or valence-band offset (vbo) are available. The cbo is defined as:

$$\mathrm{cbo} = \frac{\Delta E_{\mathrm{C}}}{\Delta E_{\mathrm{g}}} \tag{5}$$

and the vbo is defined as:

$$\mathrm{vbo} = \frac{\Delta E_{\mathrm{V}}}{\Delta E_{\mathrm{g}}} \tag{6}$$

If the cbo is known, the electron affinity of one of the materials in the heterostructure is modified to:

$$\chi = \chi_{\mathrm{ref}} - \mathrm{cbo} \bullet (E_{\mathrm{g}} - E_{\mathrm{g,ref}}) \tag{7}$$

where E_{g} and $E_{\mathrm{g,ref}}$ are the band gaps of the material and reference material, respectively. Again, it is assumed that the electron affinity model is valid.

In the MatPar approach, the reference material is defined in the MatPar command file using the Tcl variable `substrate`, and the cbo value of a material is defined using the Tcl variable `cbo` in the Tcl source parameter file of the corresponding material.

In the Material Parameter Database, all of the model files in the `Bandgap` folder calculate the value of the band gap and the electron affinity for a material. This value of the electron affinity can be called the *unmodified electron affinity* of the material.

If a reference material is defined and the cbo value for a particular material is defined, the model file (which is used to calculate the model parameters of the `Bandgap` section) for the material calculates the modified value of `Chi` using Eq. 7. MatPar prints this modified value of the electron affinity in the corresponding user-defined parameter file.

For simulations involving heterostructures with more than two types of material, the electron affinity (χ) of each material in the multilayer stack is calculated with respect to the electron affinity of the reference material using Eq. 7. Similar to the case of a multilayer stack consisting of only two materials, in the MatPar approach, you must specify the reference material in the MatPar command file and the cbo of each material with respect to the reference material in the corresponding Tcl source parameter files.

When using MatPar, if you do not define `substrate` and `cbo`, Sentaurus Device uses the bandgap values and unmodified electron affinity values to simulate the heterostructure band diagram.

If the experimental values of vbo, CBO, or VBO are available, instead of specifying `cbo` in the Tcl source parameter file, `vbo`, `CBO`, or `VBO` can be specified; and the corresponding model file of the material can be modified accordingly to calculate the modified value of `Chi` using equations similar to Eq. 4 or Eq. 7.

Examples of model implementation

These examples illustrate model implementation for both default models and new models.

Implementing default bandgap model

The Varshni bandgap model (Eq. 1, p. 37) is the default bandgap model. Therefore, the Varshni bandgap model parameter set is the default model parameter set. The default model parameter set {`Eg(0), Chi(0), alpha, beta, Tpar, Bgn2Chi`} is defined in the GaAs bandgap default model file (`pardb/GaAs/BandStructure/Bandgap/Default.tcl`) using the Tcl variables {`Eg0, Chi0, alpha, beta, Tpar, Bgn2Chi`}:

```
variable alpha 5.4050e-04
variable beta 2.0400e+02
variable Tpar 0.0
variable Eg0 1.519
variable Bgn2Chi 0.5
```

The value defined above for each parameter is from the `Bandgap` section of the default material parameter file for GaAs. If the Tcl variables `substrate` and `cbo` are defined, the value of `Chi` for GaAs is calculated using Eq. 7, followed by the calculation of the value of `Chi0`. Otherwise, the default value of `Chi0` is used, followed by the calculation of `Chi`:

```
variable Chi0 4.07
variable Chi [expr {$Chi0 - $alpha*$Tpar*$Tpar/(2*($Tpar + $beta)) +
$alpha*$::temp*$::temp/(2*($::temp + $beta))}]
```

In addition to the default model parameter set, this model file calculates the band gap `Eg` (using Eq. 1, p. 37) and the electron affinity `Chi` at temperature `temp`:

```
variable Eg [expr {$Eg0 + $alpha*$Tpar*$Tpar/($Tpar + $beta) -
$alpha*$::temp*$::temp/($::temp + $beta)}]
```

Even though both `Eg` and `Chi` are not required for the `Bandgap` section, the default model file calculates these material properties for:

- Printing these values in commented lines in the `Bandgap` section.
- Plotting using the Material Parameter Plotter.

Implementing Lautenschlager bandgap model

The Lautenschlager bandgap model [8] is the same as the Sentaurus Device default model. The model file `pardb/GaAs/BandStructure/Bandgap/Lautenschlager.tcl` implements the calculation of the default model parameter set, using the Varshni model parameter set from Lautenschlager.

Since Lautenschlager does not mention any parameter values for `Chi0` and `Bgn2Chi`, the default model parameter values are used for these parameters:

```
# Parameters from Lautenschlager
variable alpha 5.5000e-04
variable beta 2.2500e+02
variable Tpar 0.0
# Parameters from default model parameter set
variable Eg0 1.517
variable Bgn2Chi 0.5
```

The values of `Chi` and `Chi0` are calculated as in Implementing default bandgap model on page 65.

Implementing Pässler bandgap model

The Pässler model [6] expresses the temperature dependency of band gap using:

$$E_g(T) = E_g(0) - \frac{\alpha\Theta}{2}\left[\sqrt[p]{1 + \left\langle\frac{2T}{\Theta}\right\rangle^p} - 1\right] \qquad (8)$$

where $E_g(0)$, α, p, and Θ are parameters. The Pässler model parameter set {$E_g(0)$, α, p, Θ} is represented by the Tcl variables {`Eg00, alpha, p, theta`} in the Pässler model file.

The Pässler model is implemented by calculating the default model parameter set in terms of the Pässler model parameter set as follows:

```
# Parameters from Passler
set p 2.44
set a 0.472e-3
set theta 230.00
set Eg00 1.519

# Calculate default model parameter set in terms of Passler model parameters
variable alpha 0.0
variable beta 1e-5
variable Tpar 0.0
variable Bgn2Chi 0.5
variable Eg0 [expr {$Eg00 - ($a*$theta/2)*(pow(1.00+pow((2*$::temp/$theta),
$p), 1.00/$p)-1)}]
```

where `Eg0` is the value of the band gap calculated from the Pässler model for a particular temperature `temp`. By defining `alpha=0` and `beta=0`, the temperature dependency of `Eg` is suppressed in the Varshni model for Sentaurus Device calculation, that is, `Eg0=Eg`. The temperature dependency is taken into account by calculating the band gap using the Pässler model and setting `Eg0` to this value.

Since the Pässler model does not define any value for `Chi0`, a value from another reference can be used. In `Passler.tcl`, the electron affinity value at room temperature, `Chi300`, from Piprek [9] is used to calculate `Chi0` and `Chi`:

```
# Parameters from Piprek
variable Chi300 4.07
variable Chi0 [expr {$Chi300 - $alpha*300.0*300.0/(2*(300.0 + $beta))}]
variable Chi [expr {$Chi0 - $alpha*$Tpar*$Tpar/(2*($Tpar + $beta)) +
$alpha*$::temp*$::temp/(2*($::temp + $beta))}]
```

Run-time and preprocessed models

Several Sentaurus Device default models and the corresponding model parameters depend on the local values of structure parameters (such as local values of the doping concentration or mole-fraction profiles) or solution variables (such as the local values of the lattice or carrier temperature). For such models, the actual value of the model parameters can only be evaluated at run-time[1].

1. As mentioned in Model implementation and printing section on page 62, the material property is also calculated in many of the model files corresponding to the default models for the purpose of printing in the user-defined parameter file or plotting with the Material Parameter Plotter.

NOTE It is recommended to rely on the built-in default models if the dependency of the model parameters on local or solution fields or both is important. Since the default model parameters are used to compute the material properties during run-time, the default models are also called run-time models.

Often, however, the model parameters do not need to depend on local or solution fields or both. Examples are layers with constant doping and mole fractions as well as isothermal simulations. In such cases, you can compute all model parameters before starting the actual simulation (that is, at a preprocessing stage). This additional flexibility and the MatPar approach are based on preprocessing model parameters, based on regionwise predefined doping and mole-fraction values (defined in Epi) and global parameters such as the temperature.

The new models are implemented in terms of the default models. For example, as discussed in Implementing Pässler bandgap model on page 66, the Pässler bandgap model is implemented in terms of the Varshni bandgap model, which is the default bandgap model. In such cases, the material parameter values of the default model parameter set are computed so that:

- The material property as a function of physical variables is calculated in the model file and is set equal to one of the default material parameters of the corresponding default model.
- A few parameters in the default parameter set are set to zero in the model file, so that the dependency of the material property on the physical variables is suppressed during run-time and, as far as Sentaurus Device is concerned, these material properties are independent of the physical variables.

The new models are also called preprocessed models since the values of material properties computed by the corresponding model files are used by Sentaurus Device during run-time instead of the values computed during run-time. As a result, even though the doping, mole fraction, or temperature may vary as a function of position in a device or may depend on the applied bias, the material property is constant. Therefore, the simulation of devices containing profiles of the physical variables does not give correct results if the new models are used.

For example, the Varshni model is the default bandgap model and {`Chi0, Bgn2Chi, Eg0, alpha, beta, Tpar`} is the corresponding default model parameter set. The Pässler bandgap model, which is a new model, has been implemented for some materials (for example, GaAs) in the Material Parameter Database.

In the model file `Passler.tcl`, the band gap and electron affinity are calculated as a function of temperature and are set equal to `Eg0` and `Chi0`, respectively. `alpha` and `beta` are set equal to zero. When Sentaurus Device calculates the band gap as a function of temperature during run-time, the band gap is constant throughout the device, even if the temperature varies at different positions inside the device.

For devices containing regions with doping and x–mole fraction profiles, a default reference value for the corresponding variables is specified (see Creating user-defined parameter files for regions with doping/mole-fraction profiles on page 29). In addition, as discussed in Using parameter sections in Tcl source parameter files on page 57, parameter sections can be explicitly inserted into the Tcl source parameter files, especially for regions with mole-fraction profiles.

Helper procedures for calculating model parameters/ material properties

Table 17 describes some helper procedures in the `lib/helper.tcl` file. These procedures are helpful in computing model parameters or material properties in the model files.

Table 17 Procedures in helper.tcl

Procedure	**Description**
`readTable <file> <tableType>`	Reads tabular data from data file `<file>`, and returns the column values in individual lists.
`<tableType>=<TableODB | TableBGN>`	Controls the variable lists returned by `readTable`. If `tableType=TableODB`, `readTable` returns three lists: wavelength $[\mu m]$, refractive index [1], and extinction coefficient [1]. The data files contain three columns, one for each of these variables. The columns are separated by blank spaces. If `tableType=TableBGN`, `readTable` returns two lists: doping concentration $[cm^{-3}]$ and bandgap reduction [eV]. The data files contain two comma-separated columns for these variables.
`linspace <min> <max> <N>`	Creates a list of N linearly spaced numbers between `min` and `max` (including `min` and `max`). `linspace` is usually used for temperature and mole fraction. N, `min`, and `max` have these constraints: • N should be greater than 1. • `min != max`. • `min` < `max`.
`logspace <min> <max> <N>`	Creates a list of N logarithmically spaced numbers between `min` and `max` (including `min` and `max`). `logspace` is used for doping concentration. N, `min`, and `max` have these constraints: • N should be greater than 1. • `min != max`. • `min` < `max`. • `min` > 0 and `max` > 0.

Table 17 Procedures in helper.tcl

Procedure	Description
`decadelinspace <min> <max> <N>`	Creates a list of linearly spaced numbers between each decade interval specified using `min` and `max`. There are `N` numbers in each decade interval. The generated list contains 2`N` – 1 numbers. `decadelinspace` is used for doping concentration. `N`, `min`, and `max` have these constraints: • `N` should be greater than 1. • `min != max`. • `min` < `max`. • `min` > 0 and `max` > 0.
`maxima <list_of_numbers>`	Finds the maxima in a list of numbers.
`maxIndex <list_of_numbers>`	Finds the maxima in a list of numbers and the index corresponding to the maxima.
`minima <list_of_numbers>`	Finds the minima in a list of numbers.
`minIndex <list_of_numbers>`	Finds the minima in a list of numbers and the index corresponding to the minima.

The model files for the parameter sections `TableODB` and `TableBGN` implement table-based models. These model files read the columnar data in the data files stored in a subdirectory and store the data as separate lists. Generally, one list is created for each data column. This is accomplished by using the `readTable` procedure.

For example, the Levinshtein model for `TableODB` (`pardb/GaAs/Optics/TableODB/Levinshtein.tcl`) and the default model for `TableBGN` (`pardb/GaAs/BandStructure/TableBGN/Default.tcl`) use the `readTable` procedure.

The procedures `linspace`, `logspace`, and `decadelinspace` are useful for generating lists of numbers. They generate lists of physical variables such as doping concentration, temperature, and mole fraction. These procedures are used in model files as well as the command file of the Material Parameter Plotter. For example, the command:

```
linspace 200 300 11
```

generates a list of 11 linearly spaced temperature values between 200 and 300:

```
200, 210, 220, ..., 290, 300
```

The command:

```
logspace 1e15 1e16 10
```

generates a list of ten logarithmically spaced doping concentration values between 1e15 and 1e16:

```
1e15, 1.2915e+15, 1.6681e+15, 2.1544e+15, 2.7826e+15, 3.5938e+15, 4.6416e+15,
5.9948e+15, 7.7426e+15, 1e16
```

The procedure `decadelinspace` generates a linearly spaced list of numbers in a decade interval. For example, the commands:

```
decadelinspace 1e15 1e16 10
decadelinspace 1e16 1e17 10
```

generate ten linearly spaced doping concentration values between 1e15 and 1e16, and another ten values between 1e16 and 1e17:

```
1e15, 2e15, 3e15, ..., 9e15, 1e16
1e16, 2e16, 3e16, ..., 9e16, 1e17
```

The command:

```
decadelinspace 1e15 1e17 10
```

generates a list of 19 numbers:

```
1e15, 2e15, 3e15, ..., 9e15, 1e16, 2e16, 3e16, ..., 9e16, 1e17
```

The file `pardb/GaAs/BandStructure/TableBGN/Blieske.tcl` uses the procedure `decadelinspace` to generate a list of doping concentration values and to calculate bandgap narrowing.

The procedure `minima` returns the minima of a list of numbers, and `minIndex` returns the minima as well as the index of the minima in the list. For example:

```
minIndex {4 8 7 5 9 1 8}
```

results in:

```
1 5
```

The model file `pardb/AlGaAs/Bandgap/Levinshtein.tcl` uses `minIndex` to calculate the GaAs band gap as the minima of band gaps of the Γ-, L-, and X-bands.

Interpolation procedures for ternary and quaternary compound semiconductors

Material parameters of ternary alloys are calculated by interpolating the material parameters of binary compounds. Similarly, for quaternary alloys, material parameters of binary or ternary semiconductors are used.

The interpolation equations discussed in [2][9][10] are implemented in several procedures in the compound library file `lib/compound_lib.tcl`. Table 18 lists these procedures. Here, the material parameter can be a material property (for example, a dielectric constant) or a model parameter (for example, `Eg0` and `Chi0`).

Table 18 Interpolation procedures in compound_lib.tcl

Procedure	**Description**
`compute_ternary x AB BC K`	Returns the ternary alloy `A(x)B(1-x)C` material parameter by an interpolation between the binary semiconductor parameters AB and BC. K is the bowing parameter. `P(A(x)B(1-x)C) = AB*x+BC*(1-x)-K*x*(1-x)` [9]. By default, `K=0`.
`compute_2_2_quaternary x y ABC ABD BCD ACD`	Returns the quaternary alloy `A(1-x)B(x)C(y)D(1-y)` material parameter from an interpolation between the ternary alloy parameters: ABC, ABD, BCD, and ACD. `V(A(1-x)B(x)C(y)D(1-y)) = (x*(1-x)*(y*ABC+(1-y)*ABD)+ y*(1-y)*(x*BCD+(1-x)*ACD))/(x*(1-x)+y*(1-y))` [2]
`comp_3_1_QB x y AD BD CD k_ABD k_ACD k_BCD`	Returns the quaternary alloy `A(x)B(y)C(1-x-y)D` material parameter from an interpolation between the binary alloy parameters: AD, BD, and CD, with bowing parameters for the ternaries `k_ABD`, `k_ACD`, and `k_BCD`. By default, `k=0`.
`comp_quaternary x y B1 B2 B3 B4 C12 C14 C23 C34 D`	Returns the quaternary alloy material parameter from a second-order interpolation between the binary alloys `B1`, `B2`, and `B3`, and `B4`, with bowing parameters of the ternary alloys `C12`, `C14`, `C23`, and `C34`, and the quaternary bowing parameter D. The interpolation formula can be expressed as an inner product with a 3 x 3 matrix for the alloy parameters [10].

How MatPar creates user-defined parameter files

MatPar utilizes the user-specified inputs (MatPar command file, source parameter files, and model files) to create the user-defined parameter files for a multilayer stack as follows:

- The MatPar node is executed, and the commands in the MatPar command file (`MatPar_mpr.cmd`) are executed.
- Information about the layered structure is loaded into MatPar by sourcing `n@node@_epi.tcl` through the MatPar command file. This includes the following information relevant for the creation of a user-defined parameter file:
  - Doping and mole fraction of each layer.
  - Source parameter file for each layer for creating user-defined regionwise parameter files.
  - `parFileMask` value for determining the names of user-defined regionwise parameter files.
  - Source parameter files for creating materialwise parameter files.
- The following variables are defined using appropriate commands in the MatPar command file:
  - Variables relevant for the complete device structure, for example, `temperature` and `wavelength`.
  - Location of the parameter database directory `pardb`.
- For each layer, MatPar determines the names of the user-defined regionwise and materialwise parameter files to be created. It also determines the names of the source parameter files used for creating these parameter files.
- MatPar processes each source parameter file sequentially to create the user-defined parameter files (see Order in which MatPar processes source parameter files on page 24).

  If the source parameter file is a `material.par` file, the user-defined parameter file is a copy of the source parameter file. If it is a Tcl source parameter file, MatPar processes it to compute the default model parameter set and to print the parameter sections in a user-defined parameter file.

Each Tcl source parameter file is processed as described here:

- Variables that represent layer properties such as `material` and `cbo` are defined.
- The model parameter set is calculated by executing `initProperty` procedures and sourcing model files: The parameter section to be printed and the model to be used for printing the parameter section are specified in an `initProperty` statement. The format of the `initProperty` statement is:

```
initProperty <material> <propertyGroup> <parameterSection> <model>
```

The arguments to the `initProperty` procedure are also used to build the path of the model file. The path of the model file is:

```
pardb/<material>/<propertyGroup>/<parameterSection>/<model>.tcl
```

The default model parameter set is calculated by sourcing the model file. The parameters are calculated in the namespace `material::parameterSection`, which is defined in the model file. For example, to print the GaAs bandgap parameter section using model parameters from the Levinshtein model, the following `initProperty` statement must be specified:

```
initProperty GaAs BandStructure Bandgap Levinshtein
```

The model parameters `Chi0`, `Eg0`, `alpha`, `beta`, and `Tpar` are calculated by sourcing the Levinshtein model file `pardb/GaAs/BandStructure/Bandgap/Levinshtein.tcl`. These model parameters are calculated in the namespace `GaAs::Bandgap`.

- Parameter sections are printed by executing print procedures: In a Tcl source parameter file, for every `initProperty` statement, there is a `print` statement that prints the parameter section along with the computed model parameter set in the user-defined parameter file. The print procedure is defined in the `lib/printPar.tcl` file, and it has the name `printPar::<parameterSection>`. This procedure is called from the namespace `<material>::print` (defined in the model file), which is called by the statement `<material>::<parameterSection>::print` from the Tcl source parameter file `<material>.tcl`.

  For example, to print the GaAs `Bandgap` section using the model parameters from Levinshtein, the procedure `GaAs::Bandgap::print` from `GaAs.tcl` calls the procedure `printPar::BandgapSection` through the file `pardb/GaAs/BandStructure/Bandgap/Levinshtein.tcl`.

Material Parameter Plotter

The Material Parameter Plotter is an Inspect-based utility for plotting the variation of material parameters (material property as well as model parameters) as a function of the physical variables such as doping concentration, temperature, mole fraction, and wavelength. It can visualize:

- The variation of a material property as a function of physical variables for multiple models. For example, the temperature variation of the GaAs band gap for multiple models such as Levinshtein, default, and Pässler can be plotted.
- The variation of auxiliary material properties as a function of physical variables. An auxiliary material property is a property that is calculated from the material properties computed from model files in the Material Parameter Database. The auxiliary material properties are not defined in Sentaurus Device. For example, lifetime and doping-dependent mobility can be computed from the model files and can be used to calculate the

diffusion length. The diffusion length can be plotted as a function of doping concentration for multiple models.

- Equation-based models and table-based models. For example, the refractive index and extinction coefficient can be plotted as a function of wavelength from models for the `TableODB` section of the parameter file.

The Material Parameter Plotter uses Inspect, a user-specified Inspect command file, several helper procedures defined in `lib/helper.tcl` and `pardb/mprPlotter/computematpar.tcl`, and model files in the Material Parameter Database.

The Material Parameter Plotter is located in the `pardb` directory as a Sentaurus Workbench project called `mprPlotter`. The `mprPlotter` project comprises several instances of Inspect. Each Inspect instance has a command file that contains commands for plotting the material properties using Inspect. The collection of all Inspect instances in `mprPlotter` forms the example library of the Material Parameter Plotter (see Example library on page 80).

Each instance of Inspect demonstrates the use of a combination of Tcl, Material Parameter Plotter, and Inspect commands to plot material properties as a function of physical variables and is a use example of the Material Parameter Plotter. For example, the Inspect instance identified by the tool name `EgVsT` demonstrates the use of the Material Parameter Plotter to plot band gap as a function of temperature.

Command file

The default command file name for Inspect is `inspect_ins.cmd`. The command file uses standard Tcl commands, Material Parameter Plotter commands, and Inspect commands to compute and plot the curves in Inspect.

Properties from both equation-based and table-based models can be plotted as a function of the physical variables such as doping concentration, temperature, x–mole fraction, y–mole fraction, and wavelength of incident light.

The Material Parameter Plotter commands consist of the following Tcl procedures:

- `getDataFromEqn`
- `getXdataFromTable`
- `getYdataFromTable`

These procedures are defined in `pardb/mprPlotter/computematpar.tcl` (see Table 19).

Table 19 Procedures defined in computematpar.tcl

Procedure	Description
`getDataFromEqn <Xdata>`	Returns a list of computed `<Ydata>` from a list of `<Xdata>`. Calculates a list of physical property values (`<Ydata>`) from equation-based models. `<Xdata>` is the list of physical variable values used to calculate `<Ydata>`.
`getXdataFromTable`	Returns a list of `<Xdata>`. Used for table-based models.
`getYdataFromTable`	Returns a list of `<Ydata>`. Used for table-based models.

The Material Parameter Plotter uses model information and physical variable information to compute the physical property. The Material Parameter Plotter commands are used to compute the physical property and to store the values of the physical variable and physical property in two separate lists. The Material Parameter Plotter uses the Inspect commands `cv_createFromScript` and `cv_display` to create a curve and to display the curve in Inspect [11]. In general, all Tcl variables are defined in the global namespace in the command file of the Material Parameter Plotter.

Plotting and displaying a single curve in Inspect

To create and display a single Inspect curve, follow the sections here.

Specifying path of pardb directory and source library files

The Material Parameter Plotter uses helper procedures (`lib/helper.tcl`) and model files from the `pardb` directory, as well as procedures defined in `pardb/mprPlotter/computematpar.tcl` (see Table 19).

To specify the path of the `pardb` directory, use the Tcl command:

```
set ::pardb <path_to_epi_directory>
```

The paths of all directories and files in the Material Parameter Plotter command file are specified relative to the `mprPlotter` directory. In addition, the Tcl variable `pardb` is defined in the global namespace of the Tcl interpreter. The library files are sourced using:

```
source ../../lib/helper.tcl
source computematpar.tcl
```

Defining model and material property

The model is defined by specifying the Tcl variables `material`, `propertyGroup`, `parSection`, and `model` in the global namespace:

```
set ::material <material_Name>
set ::propertyGroup <propertyGroup_Name>
set ::parSection <parameterSection_Name>
```

The names of the property group and parameter section are described in Table 13 on page 42. The Material Parameter Plotter uses the above Tcl variables to locate the model file, specified by:

```
set ::model <model_name>
```

The material property or model parameter of interest is specified by:

```
set ::propertyVariable <propertyVariable_Name>
```

where:

```
<propertyVariable_Name>=<materialProperty_Name | modelParameter_Name>
```

The model file returns the value of this variable to the Material Parameter Plotter. It is usually plotted on the y-axis in Inspect. The `propertyVariable_Name` can be found in the `CALCULATED PARAMETER` field of the documentation on model files (see Table 16 on page 61). As discussed in Model implementation and printing section on page 62, these are specified in the model files using the syntax:

```
variable <propertyVariable_Name>
```

For example, the default GaAs bandgap model file `pardb/GaAs/BandStructure/Bandgap/Default.tcl` calculates the model parameters represented by the Tcl variables `Eg0`, `Chi0`, `alpha`, `beta`, and `Tpar` and the material property bandgap represented by the Tcl variable `Eg`.

To plot the GaAs band gap as a function of temperature, specify the following code in the Inspect command file:

```
set ::material GaAs
set ::propertyGroup BandStructure
set ::parSection Bandgap
set ::propertyVariable Eg
```

Specifying physical variable name and parameter value

Many equation-based model files calculate the material property value as a function of the physical variables doping concentration, temperature, x–mole fraction, y–mole fraction, and wavelength of incident light.

The Material Parameter Plotter usually plots the material property on the y-axis and one of the physical variables on the x-axis. For equation-based models, the name of this physical variable is assigned to a Tcl variable `xvar` as follows:

```
set ::xvar <physicalVariable_Name>
```

where `physicalVariable_Name` can be `doping`, `temp`, `xMole`, `yMole`, or `lambda`. If a model file calculates the physical property value as a function of more than one physical variable, the name of one of the physical variables is used to define `xvar`; whereas, the other physical variables are called parameters in this context. The value of the parameter is assigned to a Tcl variable named after the parameter as follows:

```
set ::<parameter_Name> <parameter_Value>
```

where `parameter_Name` can be any one of `physicalVariable_Name`, which is not assigned to `xvar`.

For example, for binaries, mobility depends on both doping concentration and temperature. In a command file that plots mobility as a function of doping concentration at a temperature of 300 K, the following commands are used:

```
set ::xvar doping
set ::temp 300
```

To plot mobility as a function of temperature for the doping concentration of $1 \times 10^{18}\ \text{cm}^{-3}$, the following commands are used:

```
set ::xvar temp
set ::doping 1e18
```

For table-based models, the name of the physical variable is assigned to the Tcl variable `XdataVar`:

```
set ::XdataVar <physicalVariable_Name>
```

These names are available in the corresponding model files. For example, the default GaAs TableBGN model calculates `deltaEg_n` and `deltaEg_p` as a function of `Nd` and `Na`.

To plot the donor doping concentration on the x-axis, use:

```
set ::XdataVar Nd
```

Creating list of the physical variable values

A list of values of the physical variable is created and stored in the Tcl variable `<Xdata>`. For equation-based models, this list can be generated using the procedures `linspace`, `logspace`, or `decadelinspace` (see Table 17 on page 69). The following command is used:

```
set <Xdata> [<function> <min> <max> <N>]
```

where `function=<linspace | logspace | decadelinspace>`.

The table-based models read a list of the physical variables and physical properties from data files and store these in separate lists. The procedure `getXdataFromTable` is used to read the list of physical variables and to store the values in `<Xdata>`:

```
set <Xdata> [getXdataFromTable]
```

Calculating list of material property values from model file

The list of values of the material property is computed from model files and is stored in a Tcl variable `<Ydata>` using the following command for equation-based models:

```
set <Ydata> [getDataFromEqn <Xdata>]
```

and using the following command for table-based models:

```
set <Ydata> [getYdataFromTable]
```

`getDataFromEqn` sources the model file, computes the list of the material property from the list of the physical variable, and returns the list of the computed material property. The path to the model file is built using the Tcl variables defined in Defining model and material property on page 77.

`getYdataFromTable` reads the material property values stored in the corresponding list in the model file and returns this list.

Creating and displaying curve in Inspect

The Inspect curve is created and displayed using the commands:

```
set curveName <curveName>
cv_createFromScript $curveName <Xdata> <Ydata>
cv_display $curveName
```

where `<curveName>` is the user-specified name of the curve.

Various properties of the plot such as curve attributes and axis attributes can be specified using appropriate Inspect commands [11].

Plotting and displaying multiple curves in Inspect

To plot and display multiple curves in a single instance of Inspect:

1. Plot a single curve as in Plotting and displaying a single curve in Inspect on page 76.
2. Modify the definition of Tcl variables defined in:
   a) Defining model and material property on page 77
   b) Specifying physical variable name and parameter value on page 78
   c) Creating list of the physical variable values on page 79
3. Compute the list of the material property variable `<Ydata>`.
4. Create and display the curve in Inspect (see Creating and displaying curve in Inspect on page 79).

The above steps are illustrated by some examples from the Material Parameter Plotter example library. Following these steps, multiple curves created from:

- Equation-based models can be plotted together (see Plotting GaAs band gap (EgVsT)).
- Table-based models can be plotted together.
- Table-based and equation-based models can be plotted together (see Plotting n-GaAs bandgap narrowing (EbgnVsN) on page 82).

Example library

Table 20 lists the examples in the library of the Material Parameter Plotter. Each example is identified by the Inspect tool name, which is visible in each instance of the Inspect node.

Table 20 Examples in Material Parameter Plotter library

Inspect tool name	Description
`EbgnVsN`	Plots bandgap narrowing of n-GaAs as a function of doping concentration for table-based and equation-based models.
`EgVsT`	Plots band gap of GaAs as a function of temperature for multiple models.
`EgVsxMole`	Plots room temperature band gaps of AlGaAs as a function of `xMole`.

Table 20 Examples in Material Parameter Plotter library

Inspect tool name	Description
`muVsNtemp`	For a single model, plots electron mobility of GaAs as a function of doping concentration at different temperatures.
`nkVslambda`	For a single model, plots n and k of n-GaAs as a function of wavelength at different doping concentrations.

Plotting GaAs band gap (EgVsT)

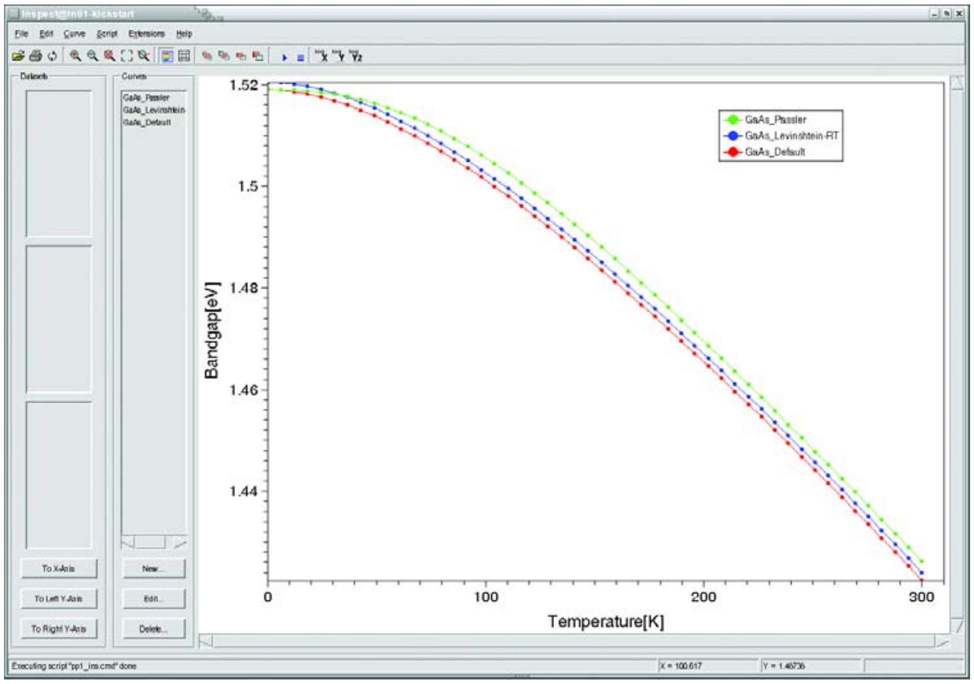
The following listing plots the GaAs band gap as a function of temperature using the default model:

```
set ::material GaAs
set ::propertyGroup BandStructure
set ::parSection Bandgap
set ::model Default
set ::propertyVariable Eg
set ::xvar temp
set Xdata [linspace 0 300 50]
set Ydata [getDataFromEqn $Xdata]
set curveName $::model
cv_createFromScript $curveName $Xdata1 $Ydata1
cv_display $curveName
```

To plot an additional curve for the temperature variation of GaAs band gap using the Levinshtein-RT model, only the model name is redefined, followed by commands for computing the list of the material property variable, creating the curve, and plotting the curve:

```
set ::model Levinshtein-RT
set Ydata1 [getDataFromEqn $Xdata]
set curveName $::model
cv_createFromScript $curveName $Xdata $Ydata1
cv_display $curveName
```

Figure 6 shows a plot of the temperature variation of the GaAs band gap for different models.



Executing script "pp1_ins.cmd" done
X = 100.617
Y = 1.48736

Figure 6 Plot of GaAs band gap as a function of temperature for multiple models

Plotting n-GaAs bandgap narrowing (EbgnVsN)

The following listing plots n-GaAs bandgap narrowing as a function of donor concentration using the table-based Jain–Roulston default model:

```
set ::material GaAs
set ::propertyGroup BandStructure
set ::parSection TableBGN
set ::model JainRoulston-Default
set ::propertyVariable deltaEg_n
set ::XdataVar Nd
set Xdata [getXdataFromTable]
set Ydata [getYdataFromTable]
set curveName $::model
cv_createFromScript $curveName $Xdata $Ydata
cv_display $curveName
```

The following listing plots an additional curve for the table-based Schubert model:

```
set ::model Schubert
set Xdata [getXdataFromTable]
set Ydata [getYdataFromTable]
set curveName $::model
```

```
cv_createFromScript $curveName $Xdata $Ydata
cv_display $curveName
```

The following listing plots an additional curve for the equation-based Blieske model:

```
set ::model Blieske
set ::propertyVariable deltaEg
set ::xvar doping
set Xdata [decadelinspace 1e15 1e20 10]
set Ydata [getDataFromEqn $Xdata]
set curveName $::model
cv_createFromScript $curveName $Xdata $Ydata
cv_display $curveName
```

References

[1] Y. P. Varshni, “Temperature Dependence of the Energy Gap in Semiconductors,” *Physica*, vol. 34, no. 1, pp. 149–154, 1967.

[2] *Sentaurus Device User Guide*, Version A-2007.12, Mountain View, California: Synopsys, Inc., 2007.

[3] B. B. Welch, K. Jones, and J. Hobbs, *Practical Programming in Tcl and Tk*, Upper Saddle River, New Jersey: Prentice Hall PTR, 4th ed., 2003.

[4] *Sentaurus Workbench User Guide*, Version A-2007.12, Mountain View, California: Synopsys, Inc., 2007.

[5] P. Lautenschlager *et al.*, “Interband critical points of GaAs and their temperature dependence,” *Physical Review B*, vol. 35, no. 17, pp. 9174–9189, 1987.

[6] R. Pässler, “Parameter Sets Due to Fittings of the Temperature Dependencies of Fundamental Bandgaps in Semiconductors,” *Physica Status Solidi (b)*, vol. 216, no. 2, pp. 975–1007, 1999.

[7] V. V. Mitin, V. A. Kochelap, and M. A. Stroscio, *Quantum Heterostructures: Microelectronics and Optoelectronics*, Cambridge: Cambridge University Press, 1999.

[8] M. Levinshtein, S. Rumyantsev, and M. Shur, *Handbook Series on Semiconductor Parameters, Si, Ge, C (Diamond), GaAs, GaP, GaSb, InAs, InP, InSb*, vol. 1, Singapore: World Scientific, 1996.

[9] J. Piprek, *Semiconductor Optoelectronic Devices: Introduction to Physics and Simulation*, Amsterdam: Academic Press, 2003.

[10] G. P. Donati, R. Kaspi, and K. J. Malloy, “Interpolating semiconductor alloy parameters: Application to quaternary III–V band gaps,” *Journal of Applied Physics*, vol. 94, pp. 5814–5819, November 2003.

[11] *Inspect User Guide*, Version A-2007.12, Mountain View, California: Synopsys, Inc., 2007.

CHAPTER 5 Creating applications using template directory

This chapter describes how to create applications with the template directory.

Overview

In general, the applications that use Epi and MatPar reside in the Meta project (`<MetaProject>`) directory, which contains the `epi` Sentaurus Workbench project and the subdirectories `lib`, `par`, and `pardb`.

It also contains multiple Sentaurus Workbench custom projects (`<CustomProject>`). The custom projects are created by copying and renaming the Sentaurus Workbench project.

There can be any number of custom projects in a `<MetaProject>` directory. Usually, a `<MetaProject>` corresponds to a particular device, and the custom projects correspond to various Sentaurus Workbench projects for the device. In addition, there can be multiple `<MetaProject>` directories in `$STDB`. For a given `<MetaProject>`, there is a common `lib`, `par`, and `pardb` subdirectory.

Creating a Meta project

To use the Epi/MatPar template directory to create a Meta project and custom projects:

1. Download and unpack the Epi/MatPar template directory `epi_template.gzp` using the download instructions in the application note *Template for Creating and Simulating Multilayered Heterostructure Devices with TCAD Sentaurus* [1].
2. Uncompress and untar `epi_template.tar.gz` in any directory under `$STDB`.

   The resulting directory structure is shown in Table 1 on page 1.
3. Create a `<MetaProject>` folder:
   a) Copy and paste the `epi_template` folder into any directory under `$STDB`.

   b) Rename the folder.

4. To create a single Sentaurus Workbench custom project:
   a) Copy and paste the `epi` Sentaurus Workbench project into the `<MetaProject>` directory.
   b) Rename the project.
   c) Similarly, you can create multiple Sentaurus Workbench custom projects under a single `<MetaProject>` directory:
5. Select model names for each property and material by consulting the documentation of the Material Parameter Database (see Material Parameter Database on page 41).

   To create new model files, see Adding new models and new materials on page 43 and Model files on page 58.
6. Prepare the source parameter files to be used for creating user-defined regionwise or materialwise parameter files. There can be one `<material>.par` and one `<material>.tcl` file for each material or region. These source parameter files are used for all the custom projects:
   a) The template Tcl source parameter file can be used to create the `<material>.tcl` file for each material: Specify the material name, the cbo value in the case of heterostructures, the commands in the Input Validation and Header Printing section, and one `initProperty` and `print` statement for each section to be printed. Specify the model names in the `initProperty` statements. See Source parameter files on page 52 and Template Tcl source parameter file on page 58.
   b) The `<material>.par` files can be created by either using the default parameter files or changing the model parameters in the default parameter files (see Source parameter files on page 52).

For a particular custom project:

1. Specify the following information in the Epi command file `epi_epi.csv` (see Chapter 2 on page 7):
   a) Description of the multilayer stack.
   b) Description of each layer.
   c) Global mesh refinement strategy.
   d) Regionwise mesh refinement strategy in the y-direction.
   e) Source parameter files.
   f) Extension column content to be inserted in preprocessed Sentaurus Device command file.
2. Verify the multilayer stack created by Epi using Tecplot SV. The structure is stored in `n@node@_epi_msh.tdr`.

3. Add the following Scheme commands in the command file (`sde_dvs.cmd`) of the subsequent Sentaurus Structure Editor node (see Chapter 3 on page 35):
   a) Load the Scheme script file `n@node@_epi.scm` generated by Epi.
   b) Perform further device processing.
   c) Add contacts.
   d) Define the mesh refinement strategy in the y-direction and z-direction (only for 3D).
   e) Invoke the mesher.
4. Edit the MatPar command file `MatPar_mpr.cmd`, specifying (see MatPar command file on page 45):
   a) Device temperature.
   b) Wavelength of incident light if required.
   c) Reference material for heterostructures.
   d) Name of the main parameter file.
   e) Location of the Material Parameter Database.
   f) Commands for inserting extension columns in preprocessed Sentaurus Device command file.
5. Add the required tools after the MatPar node in the tool row, and edit the command files of these tools.

The simulation project for the application note *Simulation of 2D Single-junction AlGaAs/GaAs Solar Cells with Transfer Matrix Method* [2] has been created using the above procedure.

References

[1] *Template for Creating and Simulating Multilayered Heterostructure Devices with TCAD Sentaurus*, TCAD Sentaurus application note, available from SolvNet at <https://solvnet.synopsys.com>, April 2008.

[2] *Simulation of 2D Single-junction AlGaAs/GaAs Solar Cells with Transfer Matrix Method*, TCAD Sentaurus application note, available from SolvNet at <https://solvnet.synopsys.com>, April 2008.