Reference Manual for etaOpt

— Version 15.10.2001 - 22:59 —

Contents

This page has been automatically generated with DOC++
DOC++ is ©1995 by Roland Wunderling
Malte Zöckler

Liste aller Contexte:

- etaOpt‘
- Hilfsfunktionen‘

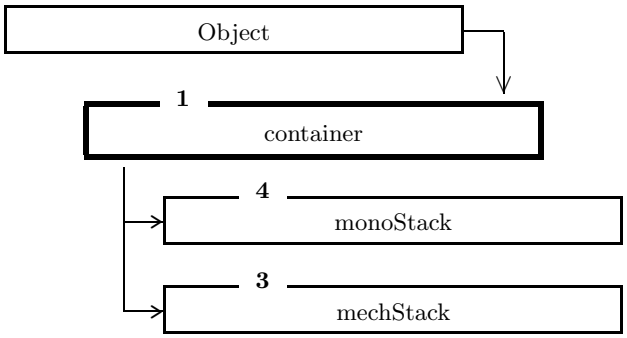
Enjoy!

1

class **container** : public Object

Class *(Notebook: etaOpt.m) (Context: etaOpt‘)*

Inheritance



Public Members

This page has been automatically generated with DOC++
DOC++ is ©1995 by Roland Wunderling
Malte Zöckler

Protected Members

new[] creates an Object container, which is used to navigate through a list of items. Following options are avalaible:
name->´´´´.... each container can have a name, which can be retrieved with giveName[].
item->{}..... a list of items, which can be anything, i.e. an object, a number, another container...

1.4

Public Method **appendTo** ()

appendTo[item] appends ´´item´´ to the itemlist.

1.5

Public Method **evaluateForEachItem** ()

evaluateForEachItem[method] this method makes only sense, if container only contains container and Objects. In this case evaluateForEachItem goes down in the hierachy of container, and if it finds an object which is not a container, evaluates the method, and returns the returnvalue if any. I.e. evaluateForEachItem[giveName[]] returns {{cell1,cell2},{cell3}}

1.6

Public Method **giveCurrent** ()

giveCurrent[] returns the current item in the list. If there is no list or no item to return, it returns Null.

1.7

Public Method **giveFirst** ()

giveFirst[] returns the first item in the list. If there is no list or no item to return, it returns Null.

1.8

Public Method **giveItem** ()

giveItem[position] returns the item at given position in the list. I.e. giveItem[3] returns the 3th item in the list. If there is no list or no item to return, it returns Null.

1.9

Public Method **giveLast** ()

giveLast[] returns the last item in the list. If there is no list or no item to return, it returns Null.

1.10

Public Method **giveList** ()

giveList[] returns the whole list of items.

1.11

Public Method **giveName** ()

giveName[] returns the name of the container as string.

1.12

Public Method **giveNext** ()

giveNext[] returns the next item in list. If there is no list or no item to return, it returns Null.

1.13

Public Method **giveNumberOf** ()

giveNumberOf[] returns the number of items in the list.

1.14

Public Method **givePosition** ()

givePosition[] returns the current position in the item list. I.e. giveItem[givePosition] is the same as giveCurrent[].

1.15

Public Method **givePrevious** ()

givePrevious[] returns the previous item in list. If there is no list or no item to return, it returns Null.

1.16

Public Method **isFirst** ()

isFirst[] returns the True if the current item is the first item. If there is no list or no item to return, it returns False.

1.17

Public Method **isLast** ()

isLast[] returns the True if the current item is the last item. If there is no list or no item to return, it returns False.

1.18

Public Method **moveTo** ()

moveToLast[position] moves to a given position in the list but returns nothing. I.e. moveTo[3];givePosition[] returns 3.

1.19

Public Method **moveToFirst** ()

moveToFirst[] moves to the first item of the list but returns nothing. I.e. moveToFirst[];isFirst[] returns True.

1.20

Public Method **moveToLast** ()

moveToLast[] moves to the last item of the list but returns nothing. I.e. moveToLast[];isLast[] returns True.

This page has been automatically generated with DOC++
DOC++ is ©1995 by Roland Wunderling
Malte Zöckler

1.21

Public Method **setList** ()

setList[list] sets the whole list. I.e. setList[{item1,item2}].

1.22

Public Method **setName** ()

setName[name] sets the name of the container. I.e. setName[´´my list´´].

1.1

Instancevariable **curPos**

1.2

Instancevariable **myList**

1.3

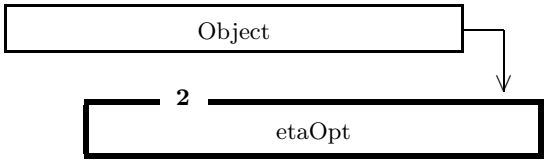
Instancevariable **myName**

2

class **etaOpt** : public Object

Class *(Notebook: etaOpt.m) (Context: etaOpt‘)*

Inheritance



Public Members

This page has been automatically generated with DOC++
DOC++ is ©1995 by Roland Wunderling
Malte Zöckler

Protected Members

new[] creates an object of type etaOpt. Following options are available (showing default values):
name->´´´´´..... sets the name of the simmulation. this is used by save an export messages for comments.
item->mechStack.new[].... tells etaOpt which mechStack to use for calculations.
spectrum->spectrum.new[]. tells etaOpt which spectrum to use.
lambdaBegin->0..... tells etaOpt where to begin the spectrum of the top cell
bandgapRange->{}..... tells etaOpt for which bandgaps the calculations should be performed. for each junction the list must contain a list with minimum value, maximum value, and step size in eV. I.e. To calculate for tandem the top cell 1.1 - 2 eV every 0.05 eV and bottom cell the range 0.5-1 eV every 0.1 eV following must be typed: {{1.1,2,0.05},{0.5,1,0.1}}

2.8

Public Method **calcAll** ()

calcAll[] starts the calculation through the whole range. For calculating the efficiency spectrum.giveTotalPower[] is used.
The result can be read with giveData[], graphically displayed with plot[], saved with saveAll[] or exported ap plain text for use in other programs like Origin with export[]

2.9

Public Method **calcEtaMax** ()

calcEtaMax[] tries to find the maximum efficiency for a specified range.

Options (given values are defaults):
bandgapRangeList->{}... This option must be specified, and gives the inital set of bandgaps.
finestStep->0.01..... if all bandgapsteps are less or equal to this the calculation stops.
stepsPerInterval->3... how many points should be calculated in each interval. Must be bigger than 3.
debug->False..... If set to true some debug information will be printed on the screen during calc.

I.e. for a tandem cell: calcEtaMax[bandgapRangeList->{{1,2},{0.4,1.4}},finestStep->0.01, stepsPerInterval->4].
In a first step eta is calculated for all bandgaps defined by step ({1,1.25,1.5,1.75,2} for the top cell {0.4,0.65,0.9,1.15,1.4} for the bottom cell. Than the maximum is calculated. The new range is set to maxvalue +/- interval size. So if 1.4 eV was found in a first step for the top cell the new interval becomes 1.4 +/- 0.25. This is devided in 4 steps.....This iteration continous until the stepsize is less or equal to finestStep.

2.10

Public Method **calcMpp** ()

calcMpp[bandgapList] starts the calculation of the power at maximum powerpoint for a given bandgapList. I.e. for a tandem calcMpp[{{2,1}}] calculates the power for top cell set to 2 eV bottom cell set to 1 eV.
For calculating the efficiency spectrum.giveTotalPower[] is used.
The results can be retrieved as a List {eta,...} with giveMpp[].

2.11

Public Method **export** ()

export[] saves a subset of values as plain text to a file for use in other programs. Options:
subset→{}.... a list specifying which subset of data should be retrieved. x and y are placeholders other bandgaps must be set to a fix value. if only x is specified than a line of data will be returned. see also giveSubset
value→´´eta´´.. a string specifying the value which should be retrieved. at the moment only ´´eta´´ is available. if this option is not used, value->´´eta´´ is used.
file→´´´´..... the filename where to save the file. If this Option is not given a filename is generated via generateFilename and saved in the current dirctory use Directory[] to get the current directory.
comment->{{}}.... additional comments for export2Origin can be applied
I.e. export[subset->{x,y,0.7}] saves

2.12

Public Method **generateFilename** ()

generateFilename[] returns a filename based on the most important simulation parameters like cellstructure, spectrum, temperature, junction model, currentmodelling as string. I.e. (EE)(E)_100xAM1_5d_300K_1D_CM.dat. Available options:
extension->´´dat´´.. appends the given extension instead of the default ´´dat ´´. i.e. extension->´´txt´´ add->´´´´.. appends the given string to the filename without extension. i.e. generateFilename[add->´´muell´´]->´´(E)_am15d_1000x1000W_300K_1D_CM_1muell.dat´´

2.13

Public Method **giveBandgapList** ()

giveBandgapList[] returns a table of all bandgap combination for which the calculations should be performed. the range is set by the bandgapRAnge option in new[] or the setBandgapList[] method. giveBandgapPatternString is used to determine the cell structure.
For a tandem cell the list looks like this:
{{1.8,0.8},{1.9,0.8},{2,0.8},{1.8,0.9}...}

2.14

Public Method **giveBandgapRangeList** ()

giveBandgapRangeList[] returns the bandgap range list for which the calculation should be done. I.e. a ´´RangeList´´ in the form of {{2,3,0.05}{1,2,0.1}} means that the bandgap of the first cell should be varried from 2 to 3 eV with stepheights of 0.05 eV the bandgap of the second cell from 1 - 2 eV with a step height of 0.1eV.

2.15

Public Method **giveCell** ()

giveCell[] returns the cell as object.

2.16

Public Method **giveData** ()

giveData[] returns the raw data in the form {{{E1,E2,},{eta}},}.

2.17

Public Method **giveEta** ()

giveEta[] returns the efficiency calculated by calcMpp in %.

2.18

Public Method **giveLambdaBegin** ()

giveLambdaBegin[] returns begining of the spectrum of the top cell in nm.

2.19

Public Method **giveMatrix** ()

giveMatrix[] gives the calculated value in form of a Matrix. I.e. value->´´eta´´,subset->{1,2,x,y} gives a matrix of efficiency, where cell1 = 1eV,cell2=2eV and cell3 and cell4 is varried. Options:
subset→{}.... a list specifying which subset of data should be retrieved. x and y are placeholders other bandgaps must be set to a fix value. if only x is specified than a line of data will be returned. see also giveSubset

value→´´eta´´.. a string specifying the value which should be retrieved. at the moment only ´´eta´´ is available. if this option is not used, value->´´eta´´ is used. I.e. giveMatrix[subset->{x,y,0.7}] returns a subset of data of a tripple cell where the third bandgap = 0.7 eV.

2.20

Public Method **giveMaxValue** ()

giveValueMax[value] returns the complete data set for which value becomes maximal. I.e. giveValueMax[´´eta´´] returns the maximum for a given set like this: {{1.6,1.},{46.81}}

2.21

Public Method **giveMpp** ()

giveMpp[] returns the results of the calculation initiated with calcMpp[].

2.22

Public Method **giveName** ()

giveName[] returns the name of the calculation as string.

2.23

Public Method **giveSpectrum** ()

giveSpectrum[] returns the selected spectrum as object.

2.24

Public Method **giveSubset** ()

giveSubset[] returns a subset of values. Options:
subset→{}.... a list specifying which subset of data should be retrieved. x and y are placeholders other bandgaps must be set to a fix value. if only x is specified than a line of data will be returned. see also giveSubset
value→´´eta´´.. a string specifying the value which should be retrieved. at the moment only ´´eta´´ is available. if this option is not used, value->´´eta´´ is used.
I.e. giveSubset[subset->{x,y,0.7}] returns a subset of data of a tripple cell where the third bandgap = 0.7 eV.

2.25

Public Method **giveSubset2Matrix** ()

giveSubset2Matrix[subset] converts a subset {{E11,E21,E31},,value1},}to a matrix{{value1,value2},{value10},}

2.26

Public Method **giveTitle** ()

giveTitle[] returns

2.27

Public Method **loadData** ()

loadData[] loads back a simulation previously saved with saveData; uses Get[]Options:
file->´´´´. must be specified. loads this file.

2.28

Public Method **plot** ()

plot[] a contour or a line plot of the calculated data Options:
subset→{}. see givesubset[]
value→´´eta´´.. see givesubset[]
shaded→True.. only valid for contour plots, if True the area between to lines is filled, else only lines with values area drawn.
color→True... only valid for contour plots, if True color is used else a greyscale will be used.

2.29

Public Method **saveData** ()

saveData[] saves the whole data with ´´Save[]´´, so that it can be loaded back with loadData[]. Options:
file->´´generateFilename[]´´.. saves to the file given by this option. If this is option is not specified, saveData uses generateFilename[] to automatically set a filename.

2.30

Public Method **setBandgapRangeList** ()

setBandgapRangeList[RangeList] sets the range, for which the calculation should be done. I.e. a ´´RangeList´´ in the form of {{2,3,0.05}{1,2,0.1}} means that the bandgap of the first cell should be varried from 2 to 3 eV with stepheights of 0.05 eV the bandgap of the second cell from 1 - 2 eV with a step height of 0.1eV.

2.31

Public Method **setCell** ()

setCell[mechStackCell] sets the complete stack of cell to the object of type mechStackCell.

2.32

Public Method **setLambdaBegin** ()

setLambdaBegin[] sets the begining of the spectrum of the top cell in nm.

2.1

Instancevariable **myCell**

2.2

Instancevariable **myData**

2.3

Instancevariable **myEta**

2.4

Instancevariable **myLambdaBegin**

2.5

Instancevariable **myName**

2.6

Instancevariable **myRangeList**

2.7

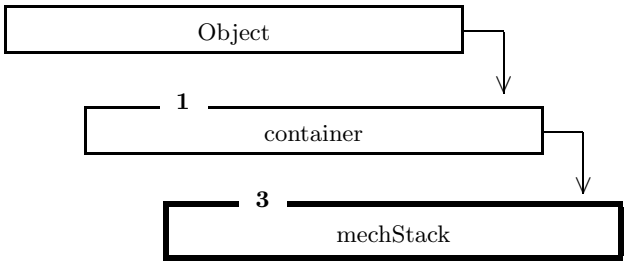
Instancevariable **mySpectrum**

3

class **mechStack** : public container

Class *(Notebook: etaOpt.m) (Context: etaOpt‘)*

Inheritance



Public Members

Protected Members

This page has been automatically generated with DOC++
DOC++ is ©1995 by Roland Wunderling
Malte Zöckler

new[] creates an object of type mechStack which contains monolithic stacks. As this class is derived from the class container every option of container can be used in addition.
Example: to create a mechanically stacked tandem of two onediode junctions:
mechStack.new[name->´´mechanically stacked tandem´´,item->{
monoStack.new[name->´´top monoStack´´,item->{ oneDiodeJunction[name->´´top cell´´,bandgap->2] }],
monoStack.new[name->´´bottom monoStack´´,item->{ oneDiodeJunction[name->´´bottom cell´´,bandgap->1] }]
}]

3.5

Public Method **calcMpp** ()

calcMpp[] calculates the maximum power point of all monoStacks at sets Pmpp which can be retrieved with givePmpp[].

3.6

Public Method **giveBandgapPatternString** ()

giveBandgapPatternString[] returns a string containing the structure of the cell. I.e. a monolithical tandem with a single junction under it has a structure of {{E1,E2},{E3}}.

3.7

Public Method **givePmpp** ()

givePmpp[] returns the power at maximum power point previously calculated with calcMpp[] in mW/cmˆ2.

3.8

Public Method **setBandgap** ()

setBandgap[list] sets the bandgaps of all junctions to the values in list. I.e. setBandgap[{3,2,1}] sets the bandgap of the first cell to 3 eV the second 2 eV...

3.9

Public Method **setSpectrumBegin** ()

setSpectrumBegin[] sets the spectrum and the starting point for calculating Isc of the junctions. Following options can be used (showing default values):
spectrum->..... no default value, an object of type spectrum
lambdaBegin->.. starting point of the spectrum in nm begining at this wavelength the current integration should be done.

3.10

Public Method **setTemperature** ()

setTemperature[T] sets the Temperature of each junction in monoStack to T in K.

3.1

Instancevariable **curPos**

3.2

Instancevariable **myList**

3.3

Instancevariable **myName**

3.4

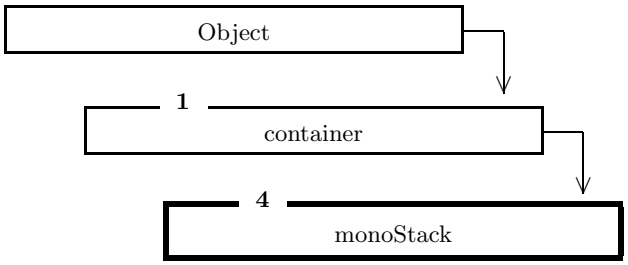
Instancevariable **myPmpp**

4

class **monoStack** : public container

Class *(Notebook: etaOpt.m) (Context: etaOpt‘)*

Inheritance



Public Members

This page has been automatically generated with DOC++
DOC++ is ©1995 by Roland Wunderling
Malte Zöckler

Protected Members

new[] creates an object of type monoStack. Following options are available (default values are shown):
currentMatching->True... if set to true an algorithm which implements semi-transparent top cells is used to achieve current matching
As this class is derived from the class container every option of container can be used in addition.
Example: to create a monolithic tandem of two onediode junctions:
monoStack.new[name->´´monolithic tandem´´,item->{
oneDiodeJunction[name->´´top cell´´,bandgap->2],
oneDiodeJunction[name->´´bottom cell´´,bandgap->1]
}]

4.9

Public Method **calcIsc** ()

calcIsc[] calculates the shortcircuit current of the stack. If currentMatching is True an algorithm which implements semi-transparent top cells is used to achieve current matching. Otherwise the shortcircuit current of all junction is set to their minimum.

4.10

Public Method **calcMpp** ()

calcMpp[] calculates the current at maximum power point. Before this method can be used Isc of the junctions must be calculated. To do this use calcIsc[].
I0 is calculated in this method calling calcI0 of the single junctions. Mpp is found by building a function P=I* giveV(I) where giveV is a method of the single junction. and then setting the first derivative=0
The values for Impp, Pmpp and Vmpp are set and can be retrieved by the apropriate give method - i.e. giveImpp[]...

4.11

Public Method **giveCurrentMatching** ()

giveCurrentMatching[] returns True if currentmatching is turned on for this stack otherwise False.

4.12

Public Method **giveImpp** ()

giveImpp[] returns the current at maximum power point in mA/cmˆ2 calculated with calcMpp[]. giveImpp[] returns 0 if calcMpp[] was not called in advance.

4.13

Public Method **giveIsc** ()

giveIsc[] returns the shortcircuit current in mA/cmˆ2 calculated with calcIsc[]. giveIsc[] returns 0 if calcIsc[] was not called in advance.

4.14

Public Method **givePmpp** ()

givePmpp[] returns the power at maximum power point in mW/cm^2 calculated with calcMpp[]. givePmpp[] returns 0 if calcMpp[] was not called in advance.

4.15

Public Method **giveVmpp** ()

giveVmpp[] returns the voltage at maximum power point in mV calculated with calcMpp[]. giveVmpp[] returns 0 if calcMpp[] was not called in advance.

4.16

Public Method **setBandgap** ()

setBandgap[bandgap] sets the bandgap of junction in eV. I.e. setBandgap[{3,2,1}] sets the bandgap of the first cell to 3 eV the second 2 eV...

4.17

Public Method **setCurrentMatching** ()

setCurrentMatching[True/False] if parameter is True (False) turns currentmatching on (off).

4.18

Public Method **setSpectrumBegin** ()

setSpectrumBegin[spectrum,lambdaBegin] Perform two things:
1. let all junctions know, which spectrum to use for calulating Isc.
2. let all junctions know, where to start integration of the spectrum. Thereby the start point of the topmost junction is set to lambdaBegin. The starting point of each following junction is set to the bandgap of his previous neighbour.

4.19

Public Method **setTemperature** ()

setTemperature[temp] sets the temperature of each junction in the stack to temp in K.

4.1

Instancevariable **curPos**

4.2

Instancevariable **currentMatching**

4.3

Instancevariable **myImpp**

4.4

Instancevariable **myIsc**

4.5

Instancevariable **myList**

4.6

Instancevariable **myName**

4.7

Instancevariable **myPmpp**

4.8

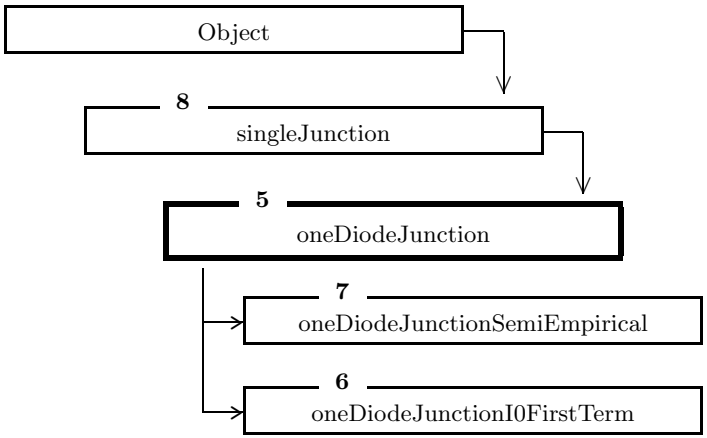
Instancevariable **myVmpp**

5

class **oneDiodeJunction** : public singleJunction

Class *(Notebook: etaOpt.m) (Context: etaOpt‘)*

Inheritance



Protected Members

This page has been automatically generated with DOC++
DOC++ is ©1995 by Roland Wunderling
Malte Zöckler

new[] creates an single Junction object based on the one diode model. The dark current Io is calculated with the following formula:
(2 q Pi)/(hˆ3cˆ2) [k T(q Eg)ˆ2 - 2(k T)ˆ2 (q Eg) + 2(k T)ˆ3] Exp[-(q Eg)/(k T)] 0.1
All options of the abstract parentclass ´´singleJunction´´ can be used for setup. only the methods calcI0, calcIsc, calcVoc and giveV are overwriten.

5.1

Instancevariable **myBandgap**

5.2

Instancevariable **myEqeMax**

5.3

Instancevariable **myI0**

5.4

Instancevariable **myIsc**

5.5

Instancevariable **myLambdaBegin**

5.6

Instancevariable **myName**

5.7

Instancevariable **myno**

5.8

Instancevariable **mynu**

5.9

Instancevariable **mySpectrum**

5.10

Instancevariable **myTemperature**

This page has been automatically generated with DOC++
DOC++ is ©1995 by Roland Wunderling
Malte Zöckler

5.11

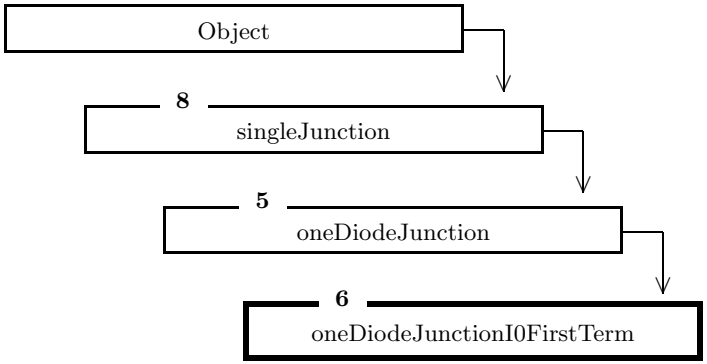
Instancevariable **myVoc**

6

class **oneDiodeJunctionI0FirstTerm** : public oneDiodeJunction

Class *(Notebook: etaOpt.m) (Context: etaOpt`)*

Inheritance



Protected Members

This page has been automatically generated with DOC++
DOC++ is ©1995 by Roland Wunderling
Malte Zöckler

new[] creates an single Junction object based on the one diode model. The dark current Io is calculated with the following formula:
(2 q Pi (myno^2+mynu^2))/(h^3 c^2)(k T(q Eg)^2)Exp[(-q Eg)/(k T)] 0.1;
All options of the abstract parentclass ´´oneDiodeJunction´´ can be used for setup. only the methods calcI0, calcIsc, calcVoc and giveV are overwriten.

6.1

Instancevariable **myBandgap**

6.2

Instancevariable **myEqeMax**

6.3

Instancevariable **myI0**

6.4

Instancevariable **myIsc**

6.5

Instancevariable **myLambdaBegin**

6.6

Instancevariable **myName**

6.7

Instancevariable **myno**

6.8

Instancevariable **mynu**

6.9

Instancevariable **mySpectrum**

6.10

Instancevariable **myTemperature**

This page has been automatically generated with DOC++
DOC++ is ©1995 by Roland Wunderling
Malte Zöckler

6.11

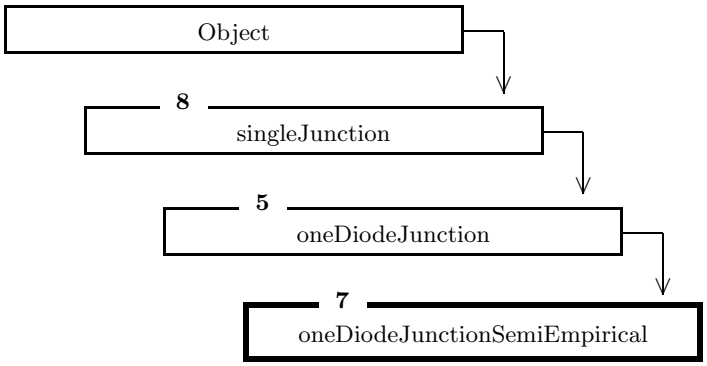
Instancevariable **myVoc**

7

class **oneDiodeJunctionSemiEmpirical** : public oneDiodeJunction

Class *(Notebook: etaOpt.m) (Context: etaOpt‘)*

Inheritance



Public Members

Protected Members

This page has been automatically generated with DOC++
DOC++ is ©1995 by Roland Wunderling
Malte Zöckler

new[] creates an single Junction object based on the one diode model. The dark current Io is calculated with an empirical formula which can be set with setI0Func.

7.13

Public Method **giveI0Func** ()

giveI0Func[] returns the pure function to calculated I0 with. The pure function is called with 3 slots. 1. slot: bandgap [eV] 2. slot: temperature [K] 3. slot: concentration [suns]. The function returns the current in mA/cm^2. E.g.: func=giveI0Func[];func[1.3,300,100]=>...

7.14

Public Method **setI0Func** ()

setI0Func[purefunction] sets the function to calculated I0 with. to purefunction. pureFunction must be a pure function with up to 3 slots. 1. slot: bandgap [eV] 2. slot: temperature [K] 3. slot: concentration [suns]. the function must return the current in mA/cm^2 E.g.: setI0Func[(#3 Exp[-#1/(k #3)])&]

7.1

Instancevariable **myBandgap**

7.2

Instancevariable **myEqeMax**

7.3

Instancevariable **myI0**

7.4

Instancevariable **myI0Func**

7.5

Instancevariable **myIsc**

7.6

Instancevariable **myLambdaBegin**

7.7

Instancevariable **myName**

7.8

Instancevariable **myno**

7.9

Instancevariable **mynu**

7.10

Instancevariable **mySpectrum**

7.11

Instancevariable **myTemperature**

7.12

Instancevariable **myVoc**

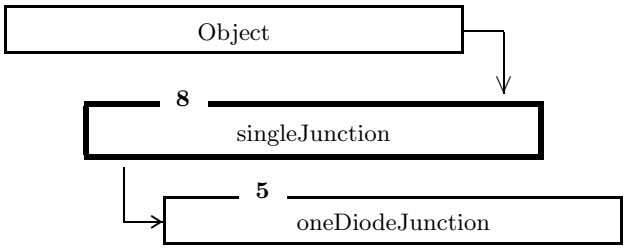
This page has been automatically generated with DOC++
DOC++ is ©1995 by Roland Wunderling
Malte Zöckler

8

class **singleJunction** : public Object

Class *(Notebook: etaOpt.m) (Context: etaOpt`)*

Inheritance



Public Members

This page has been automatically generated with DOC++
DOC++ is ©1995 by Roland Wunderling
Malte Zöckler

Protected Members

This page has been automatically generated with DOC++
DOC++ is ©1995 by Roland Wunderling
Malte Zöckler

new[] Creates an abstract class of a singleJunction. Following options can be used:
bandgap->0..... sets the bandgap of the junction in eV
name->´´´´..... sets the name of the junction.
temperature->300. sets the temperature of the junction in K
eqeMax->1..... sets the external quantum efficiency of a junction is a squarefunction with EQE = EqeMax for E>=Eg and EQE = 0 for E<Eg.
no->1..... sets the value of refractive index of the material over the junction. For more information see setno[].
nu->0..... sets the value of refractive index of the material under the junction. For more information see setnu[].

8.12

Public Method **calcAll** ()

calcAll[] calculates Isc, I0, Voc of the cell in this order.

8.13

Public Method **calcI0** ()

calcIsc[] calculates I0. This method must be defined in child classes.

8.14

Public Method **calcIsc** ()

calcIsc[] calculates Isc for a given spectrum. This method must be defined in child classes.

8.15

Public Method **calcVoc** ()

calcVoc[] calculates Voc. This method must be defined in child classes.

8.16

Public Method **giveBandgap** ()

giveBandgap[] returns the Bandgap of the Junction in eV (same as giveEg[]).

8.17

Public Method **giveEg** ()

giveEg[] returns the Bandgap of the Junction in eV (same as giveBandgap[]).

8.18

Public Method **giveEqeMax** ()

giveEqeMax[] returns the maximum of the eqe. The external quantum efficiency of a junction is a squarefunction with EQE = EqeMax for E>=Eg and EQE = 0 for E<Eg.

This page has been automatically generated with DOC++
DOC++ is ©1995 by Roland Wunderling
Malte Zöckler

8.19

Public Method **giveI0** ()

giveI0[] returns I0 in mA/cm^2. Must be calculated with calcI0[]

8.20

Public Method **giveIsc** ()

giveIsc[] returns Isc in mA/cm^2. Must be calculated with calcIsc[] or set with setIsc[].

8.21

Public Method **giveLambdaBegin** ()

giveLambdaBegin[] returns the beginning of the spectra in nm. LambdaBegin is used to calculate Isc

8.22

Public Method **giveLambdaG** ()

giveLambdaG[] returns the Bandgap of the junction in nm (see giveBandgap[]).

8.23

Public Method **giveModelAcronym** ()

giveModelAcronym[] returns a acronym for the JunctionModel eg. SJ for SIngleJunction, 1D for oneDiodeModel 1D1T for oneDiodeJunctionI0FirstTerm.

8.24

Public Method **giveName** ()

giveName[] returns the name of the junction as String.

8.25

Public Method **giveno** ()

giveno[] returns the refractive index of the material situate over the junction. This factor is used for calculating I0 see setno[].

8.26

Public Method **givenu** ()

givenu[] returns the refractive index of the material situate under the junction. This factor is used for calculating I0 see setnu[].

8.27

Public Method **giveSpectrum** ()

giveSpectrum[] returns the spectrum from which Isc can be calculated as object of type spectrum.

8.28

Public Method **giveTemperature** ()

giveTemperature[] returns the temperature of the junction in K.

8.29

Public Method **giveV** ()

giveV[I] returns the voltage mV for a given current in mA/cm^2.

8.30

Public Method **setBandgap** ()

setBandgap[bandgap] sets the bandgap of junction in eV.

8.31

Public Method **setEqeMax** ()

setEqeMax[EqeMax] assumes the EQE of a junction to be a squarefunction with EQE = EqeMax for E>=Eg and EQE = 0 for E<Eg.

8.32

Public Method **setI0** ()

setI0[I0] sets I0 of the junction in mA/cmˆ2.

8.33

Public Method **setIsc** ()

setIsc[Isc] sets Isc of the junction in mA/cmˆ2. This is needed to achieve currentmatching for semi-transparent cells

8.34

Public Method **setLambdaG** ()

setLambdaG[bandgap] sets the bandgap of junction in nm.

8.35

Public Method **setName** ()

setName[name] sets the name of junction. Name is a string.

8.36

Public Method **setno** ()

setno[value] sets the refractive index of the material situate over the junction
A value of 0 means that no radiation take place on this side
A value of 1 means that all photons with in a cone of sin(theta)<1/n will emerge of the cell - where theta is the angle between the ray and the surface normal
A value of n (where n is the refractive index of the junction itself and typically around 36) means that all photons will emerge of the cell
Default is 1
In the calculation of I0 no and nu are used as a factor (noˆ2+nuˆ2).

8.37

Public Method **setnu** ()

setnu[value] sets the refractive index of the material situate under the junction
A value of 0 means that no radiation take place on this side
A value of 1 means that all photons with in a cone of sin(theta)<1/n will emerge of the cell - where theta is the angle between the ray and the surface normal
A value of n (where n is the refractive index of the junction itself and typically around 36) means that all photons will emerge of the cell
Default is 0
In the calculation of I0 no and nu are used as a factor (noˆ2+nuˆ2)

8.38

Public Method **setSpectrumBegin** ()

setSpectrumBegin[spectrum,lambdaBegin] sets the spectrum, and the begining of the spectrum. This must be done before calculating Isc. lambdaBegin is needed because not the whole spectrum is transfered to Isc. spectrum is an object of type spectrum. lambdaBegin is in nm.

8.39

Public Method **setTemperature** ()

setTemperature[temp] sets the temperature of junction in K.

8.1

Instancevariable **myBandgap**

8.2

Instancevariable **myEqeMax**

8.3

Instancevariable **myI0**

8.4

Instancevariable **myIsc**

8.5

Instancevariable **myLambdaBegin**

8.6

Instancevariable **myName**

8.7

Instancevariable **myno**

8.8

Instancevariable **mynu**

8.9

Instancevariable **mySpectrum**

This page has been automatically generated with DOC++
DOC++ is ©1995 by Roland Wunderling
Malte Zöckler

8.10

Instancevariable **myTemperature**

8.11

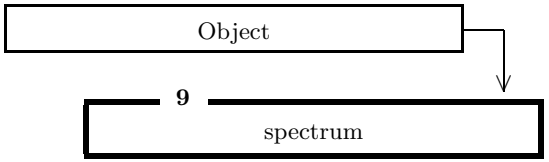
Instancevariable **myVoc**

9

class **spectrum** : public Object

Class *(Notebook: etaOpt.m) (Context: etaOpt`)*

Inheritance



Public Members

Protected Members

This page has been automatically generated with DOC++
DOC++ is ©1995 by Roland Wunderling
Malte Zöckler

sp = spectrumnew[] generates an new spectrum Object. Special care has been taken for scaling the spectrum. Following parameter are essential:
P_0,org = total power [0, ∞] of the origin spectrum as loaded from the file or generated for blackbodies (given)
P_ur,org= integrated spectrum [λ_b,λ_e] of the origin spectrum as loaded from the file or generated for blackbodies (given)
C = concentration of the spectrum in suns (to be set)
P_1= total power [0,∞] fo the spectrum for one-sun concentration (C=1) (to be set)
P_0 = total power [0,∞] of the scaled spectrum
P_ur = integrated power [λ_b,λ_e] of the scaled spectrum
Calculation is performed as followed:
P_0= C P_1, P_ur = P_0/P_0,org P_ur,org

All properties of the spectrum can be modified during creation by Options. In the following options with there default values are given:
spectrumPath->whichFile[´´pvsim´´,´´Math/QuellDaten/Spectra´´]. The default path, from where the spectra files should be loaded
concentration->1..... the concentration factor.
genFuncType->1..... sets the type for calculating the number of photons see setGenFuncType for more information.
powerOfOneSun->1000.....power of One Sun in W/mˆ2
name->´´AM1_5d/am15d.dat´´....the name for the spectral file including subdirs - this option can only be used if no blackbody spectrum with ´´blackbody->True´´ is generated. If you want to use your own spectra use ´´spectrumPath´´ to switch to your directory and look at ˜pvsim/Math/QuellDaten/Spectra/AM1_5g/iec1000.dat for an example of data file.
blackBody->False.....if set to True a blackbody spectrum is generated and the options for ´´generateBlackBody[]´´ are available in addition. (see generateBlackBody[])

A spectrum.new[] with no options generates a blackbody with T=6000 K.

9.18

Public Method **generateBlackBody** ()

generateBlackBody[] generates a blackbody spectrum with a certain emisivity in a rang from lambdaBegin to lambdaEnd with numberOfLambda values in the list. The total spectral power (P0org) is calculated via the T^4 Stefan Boltzmann law. The OneSunPower is set to this value automatically. The parameters are set with options. In the following options with there default values are given:
temperature->2000..... the temperature of the black body in K
epsilon->1..... the emissivity of the body (1 for blackbody is default)
lambdaBegin->1..... begining of the spectral range in nm. 0 < lambdaBegin < lambdaEnd!
lambdaEnd->10^9..... end of the spectral range in nm
numberOfLambdas->2.... number of datapoints for the list. All calculations are performed with the analytic expression. Thus this option does not influence any integrated results like power generatedPhotons....
solidAngle->4 Pi..... sets the solidangle in which the spectrum is emitted. This scales the intensity of the spectrum, and has only effect on the PowerOfOriginSpectrum as the spectrum is once more scaled to fit the OneSunPower
dLambda->..... optional to ´´numberOfLambdas´´ dLambda can be used, which give the distance between two spectral datapoints in nm. All calculations are performed with the analytic expression. Thus this option does not influence any integrated results like power generatedPhotons....

9.19

Public Method **giveConcentration** ()

giveConcentration[] returns the concentration factor.

9.20

Public Method **giveFileName** ()

giveFileName[] returns the name of the loaded file. if a blackbody was generated ´´´´ is returned

9.21

Public Method **giveGenFunc** ()

giveGenFunc[] returns the function used in giveNumberOfPhotons to calculated the number of photons Integrate[GenFunc]. To use the function do something like f= sp.giveGenFunc[]; f[400]

9.22

Public Method **giveGenFuncType** ()

giveGenFuncType[type]returns the integration method in giveNumberOfPhotons to calculate the number of photons. type can be one of the following:
1: Integrate[λ Interpolation[{λ,E},Order->0]]=$\sum$E_m λ_m, E_m and λ_m are meanvalues of the interval
recommended by ASTM
2: Integrate[λ Interpolation[{λ,E},Order->1]]
possible method
3: Integrate[Interpolation[{λ,E λ},Order->1]]
quick and dirty but false

9.23

Public Method **giveList** ()

giveList[] returns a list containing wavelength in nm and the spectraldensity in {nm,W/(mˆ2 μm)}, eg. {{400,1.2223},{450,1.432},...}

9.24

Public Method **giveName** ()

giveName[] returns the name of the selected spectrum with information on concentration and P0 as string, eg. ´´am15d_100x1000W´´

9.25

Public Method **giveNameList** ()

giveNameList[] gives a list of names of all available spectra, eg. {´´AM1_5g/standard.dat´´,´´AM0/standard.dat´´}

9.26

Public Method **giveNumberOfPhotons** ()

giveNumberOfPhotons[] gives the number of photons of the spectrum. If no options are used, the whole spectrum is used. With the following options one can limit the range for which the number of photons should be calculated:
lambdaBegin->´´Begin of spectra´´.. gives the wavelength in nm where to start calculations - default is spectrum begin.
lambdaEnd->´´End of spectra´´..... gives the wavelength in nm where to end calculations - default is spectrum end.

9.27

Public Method **givePower** ()

givePower[] returns the integrated power of the spectrum for the used range determined by lambdaBegin and lambdaEnd in W/mˆ2 taking into account the

PowerOfOneSun and the concentration factor. Options:
lambdaBegin->´´Begin of spectra´´.. gives the wavelength in nm where to start calculations - default is spectrum begin.
lambdaEnd->´´End of spectra´´..... gives the wavelength in nm where to end calculations - default is spectrum end.

9.28

Public Method **givePowerOfOneSun** ()

givePowerOfOneSun[] returns the total power [0,∞] for 1 sun concentration of the spectrum in W/mˆ2

9.29

Public Method **givePowerOfOrigin** ()

givePowerOfOrigin[] returns the integrated power for spectrum as loaded from the file or generated for blackbody W/mˆ2.

9.30

Public Method **giveSpectrumPath** ()

giveSpectrumPath[] give the absolute path to the spectra data files, which can be loaded via setSpectrum. See also setSpectrumPath, giveNameList, giveName.

9.31

Public Method **giveTemperature** ()

giveTemperature[] returns the temperature of the blackbody spectrumin K.

9.32

Public Method **giveTotalPower** ()

giveTotalPower[] returns the total power [0,∞] of the spectrum in W/mˆ2 taking into account the PowerOfOneSun and the concentration factor.

9.33

Public Method **giveTotalPowerOfOrigin** ()

giveTotalPowerOfOrigin[] returns the total power [0,∞] of the origin spectrum as loaded from the file or generated for blackbody in W/mˆ2.

9.34

Public Method **giveValue** ()

giveValue[λ] returns for a given λ [in nm] the spectraldensity in W/(mˆ2 μm).

9.35

Public Method **giveValueList** ()

giveValueList[] returns a list containing the spectraldensity in W/(mˆ2 μm), eg. {1.2223,1.432,...}

9.36

Public Method **giveValueMax** ()

giveValueMax[] returns the maximum of the spectraldensity in W/(mˆ2 μm), eg. 1.432. For a loaded spectrum the list is used. for blackbody WiensKonstant is used I(λ_max), λ_max=2.897756 10ˆ6/T [nm]

9.37

Public Method **giveValueMin** ()

giveValueMin[] returns the minmum of the spectraldensity in W/(mˆ2 μm), eg. 1.2223. For a loaded spectrum the list is used for blackbody returns 0.

9.38

Public Method **giveWavelengthList** ()

giveWavelengthList[] returns a list containing the wavelengths in nm, eg. {450,470...}

9.39

Public Method **giveWavelengthMax** ()

giveWavelengthMax[] returns the maxmum of the wavelength range in nm, eg. 1200

9.40

Public Method **giveWavelengthMin** ()

giveWavelengthMin[] returns the minmum of the wavelength range in nm, eg. 450

9.41

Public Method **setConcentration** ()

setConcentration[Concentration] Scales the power density of the spectrum according to the OneSunPower. Default = 1.

9.42

Public Method **setGenFunc** ()

setGenFunc[pureFunction] sets the function used in giveNumberOfPhotons to calculated the number of photons Integrate[GenFunc]. purFunction must be a pure function with one slot. E.g. setGenFunc[(100 + #)&]

This page has been automatically generated with DOC++
DOC++ is ©1995 by Roland Wunderling
Malte Zöckler

9.43

Public Method **setGenFuncType** ()

setGenFuncType[type] sets the Integration Method in giveNumberOfPhotons to calculate the number of photons. type must be one of the following: 1: Integrate[λ Interpolation[{λ,E }],Order->0]]=∑E_mλ_m, E_m and λ_m are meanvalues of the interval recommended by ASTM 2: Integrate[λ Interpolation[{λ,E},Order->1]] possible method 3: Integrate[Interpolation[{λ,E λ},Order->1]] quick and dirty but false
m

9.44

Public Method **setPowerOfOneSun** ()

setPowerOfOneSun[PowerInWatt] scales the spectrum, so that the integrated powerdensity = PowerInWatt. Default is 1000 W/mˆ2.

9.45

Public Method **setSpectrum** ()

setSpectrum[spectrumName] sets the spectrum. Default is ´´AM1_5g/iec1000.dat´´. Use ´´giveNameList´´ to see a list of all available spectra, or use ´´setSpectrumPath[]´´ to change to the appropriate directory of spectra.

9.46

Public Method **setSpectrumPath** ()

setSpectrum[absDir] sets the directory to which the spectrumName will be appended. absDir must be absolute and can be in DOS or Unix convention but must be avalaible for the MathKernel.

9.1

Instancevariable **myC**

9.2

Instancevariable **myFileName**

9.3

Instancevariable **myFkt**

9.4

Instancevariable **myGenFunc**

9.5

Instancevariable **myGenFuncType**

This page has been automatically generated with DOC++
DOC++ is ©1995 by Roland Wunderling
Malte Zöckler

9.6

Instancevariable **myLambdaBegin**

9.7

Instancevariable **myLambdaEnd**

9.8

Instancevariable **myList**

9.9

Instancevariable **myName**

9.10

Instancevariable **myP0**

9.11

Instancevariable **myP0Org**

This page has been automatically generated with DOC++
DOC++ is ©1995 by Roland Wunderling
Malte Zöckler

9.12

Instancevariable **myP1**

9.13

Instancevariable **myPur**

9.14

Instancevariable **myPurOrg**

9.15

Instancevariable **mySolidAngle**

9.16

Instancevariable **mySpectrumPath**

9.17

Instancevariable **myT**

This page has been automatically generated with DOC++
DOC++ is ©1995 by Roland Wunderling
Malte Zöckler

10

Public Functions

Names

10.1

connectDirs()

(Notebook: Hilfsfunktionen.m)

Package: ../../Hilfsfunktionen.m ‖ Context: Hilfsfunktionen‘

connectDirs[dir1, dir2] verbindet dir1 mit dir2 und baut den Pfad unter berücksichtigung des Betriebssystems des Kernels zusammen. So kann man Kernel unter NT und Unix starten.

10.2

dateAndTime()

(Notebook: Hilfsfunktionen.m)

Package: ../../Hilfsfunktionen.m ‖ Context: Hilfsfunktionen‘

dateAndTime gives a String containing the current date and time including the name of the day and seconds.

10.3

dateTime()

(Notebook: Hilfsfunktionen.m)

Package: ../../Hilfsfunktionen.m ‖ Context: Hilfsfunktionen‘

dateTime gives a String containing the current date and time.

10.4 dbgPrint()

(Notebook: Hilfsfunktionen.m)

Package: ../../Hilfsfunktionen.m ‖ Context: Hilfsfunktionen‘

dbgPrint[id,msg] Prints the message msg if id is in Global‘debugList.
id should be a string, which can be remembered easily i.e. the class name.
if id is not given the string will be ´´misc´´
debugList is a list of strings. You can use the wildchar * to search for patterns.
i.e. debugList={´´*´´} prints all dbgPrint messages.
if debugList is not defined dbgPrint only takes 0.1 ms to evaluate.

10.5 export2Origin()

(Notebook: Hilfsfunktionen.m)

Package: ../../Hilfsfunktionen.m ‖ Context: Hilfsfunktionen‘

export2Origin[opts] writes out a file in a origin compatible format which can be read into origin using the package ´´ImportSpecial´´. This means that the file begins with a comment section in the form of ´´# variable = value´´ then the data follows.
Options must be specified in the form export2Origin[{´´option1´´->value1,´´option2´´->value2,...}]:

´´filename´´-> the filename where to wrote the parfile. This option must be specified.
´´data´´-> contains the data in the form {{x1,y1,z1,...},{x2,y2,z2,...},...}
´´dataType´´-> specifies how the data given by the option ´´data´´ should be wrote.
The value will be added as comment # dataType = ...
dataType can be one of the following:

- ´´string´´ the data will be written without any modification as given by data
- any other type the 2d-list of data will be writen in matrix form
{{v1,v2},{v3,v4}}->v1 v2\nv3 v4
´´dataExpLength´´-> 2 if dataType is not ´´string´´ this specifies the length of
the exponent see outStr
´´dataMantLength´´-> 8 if dataType is not ´´string´´ this specifies the length of
the manthis see outStr
´´wrapColumn´´->Infinity if this option is specified the comment is wrapped at the specified column. ´´comment´´-> if the value is a string, this string will be writen
without any
modifications if the value is a list, then it must be in the form
{{´´var1´´,val1},{´´var2´´,val2},...}. Each entry will be formated
in the following way: # var1 = val1. line wrapping is controlled by
the option ´´wrapColumn´´
Following options can be set in origin:
Worksheet options
wksname = name of worksheet
wkslabel = label of worksheet
datatype = type of the daten _list_ / matrix
col1name = name of worksheet column 1
col1label = label of worksheet column 1

Plot options:
plotaxistype = type of the plot axis (xy) _linlin_ loglin linlog loglog
plotlinetype = type of plotline _linesymb_ scatter line
plotname = name of the plotwindow
plotlabel = label of the plotwindow
plotlegend = shall I generate a legend? _yes_ no
title = title inserted as label right over the plot bold and big size
xtitle = title of the x-axis
xmin = Min. value of the x-axis
xmax = Max. value of the x-axis
ytitle = title of the y-axis
ymin = Min. value of the y-axis
ymax = Max. value of the y-axis

For contour-plots in addition to plot options:
ztitle = title of the z-axis
zmin = Min. value of the z-axis
zmax = Max. value of the z-axis

zcount = number of colours to be used

10.6 getFilename()

(Notebook: Hilfsfunktionen.m)

Package: ../../Hilfsfunktionen.m ‖ Context: Hilfsfunktionen‘

getFilename[str] returns the filename of string

10.7 getFilenameWithoutExtension()

(Notebook: Hilfsfunktionen.m)

Package: ../../Hilfsfunktionen.m ‖ Context: Hilfsfunktionen‘

getFilename[str] returns the filename of string without extension

10.8

getPath()

(Notebook: Hilfsfunktionen.m)

Package: ../../Hilfsfunktionen.m ‖ Context: Hilfsfunktionen‘

getPath[str] returns the path of string with slash (or backslash) at the end

10.9

interpolatingFunctionQ()

(Notebook: Hilfsfunktionen.m)

Package: ../../Hilfsfunktionen.m ‖ Context: Hilfsfunktionen‘

interpolatingFunctionQ[interpolationsFkt]
gibt True zurück wenn die Übergebene Funktion eine Interpolationsfunktion ist sonst False
z.B.: interpolatingFunctionQ[f]

10.10

makeUsage()

(Notebook: Hilfsfunktionen.m)

Package: ../../Hilfsfunktionen.m ‖ Context: Hilfsfunktionen‘

makeUsage[functionName_String, contextPath_String, usageText_String]; Defines the usage for the given function Name in two contexts:
a) In the current context, so that the function is public.
b) In the Context <Usages‘contextPath‘functionName>, so that the different usages can be defined for the same methodName. The contextPath usually has the form <className‘>.

10.11

num()

(Notebook: Hilfsfunktionen.m)

Package: ../../Hilfsfunktionen.m ‖ Context: Hilfsfunktionen‘

Gibt den Zahlenwert einer physikalischen Konstante aus dem Package Miscellaneous‘PhysicalConstants‘. Beispiel: num[ElectronCharge].

10.12

outFkt()

(Notebook: Hilfsfunktionen.m)

Package: ../../Hilfsfunktionen.m ‖ Context: Hilfsfunktionen‘

outFkt[value, mantissalength, exponentlength] formatierte Ausgabe mit FixedNumberForm, defaults: mantissalength=8, exponentlength=2

This page has been automatically generated with DOC++
DOC++ is ©1995 by Roland Wunderling
Malte Zöckler

10.13

outStr()

(Notebook: Hilfsfunktionen.m)

Package: ../../Hilfsfunktionen.m ‖ Context: Hilfsfunktionen‘

outStr[value, mantissalength, exponentlength] formatierte Ausgabe mit FixedNumberForm, Rueckgabe als String, defaults: mantissalength=8, exponentlength=2
Wird als exponentLength 0 angegeben so wird die Angabe des Exponenten weggelassen z.b. outStr[11.23654, 4, 0]->+11.24
mantissalength steht dann für die Gesamtlänge der Ziffern.

10.14

readListWithComment()

(Notebook: Hilfsfunktionen.m)

Package: ../../Hilfsfunktionen.m ‖ Context: Hilfsfunktionen‘

readListWithComment[file_String, opts] reads in a file similar to ReadList but ignoring any line beginning with #.
Options:
giveComment->False... if set to True only the Comment is returned. Default is False. Kleiner Tip: mit //TableForm sieht man den Kommentar schön formatiert.

10.15

showStatus()

(Notebook: Hilfsfunktionen.m)

Package: ../../Hilfsfunktionen.m ‖ Context: Hilfsfunktionen‘

showStatus[string] writes String to the StatusArea of a Notebook.

11 Functions sorted by Notebooks

Names

11.1 Package ../../Hilfsfunktionen.m

Names

This page has been automatically generated with DOC++
DOC++ is ©1995 by Roland Wunderling
Malte Zöckler

11.1.1

\hfill{}\Ref{connectDirs()}

@Man connectDirs()

11.1.2

\hfill{}\Ref{dateAndTime()}

@Man dateAndTime()

11.1.3

\hfill{}\Ref{dateTime()}

@Man dateTime()

11.1.4

\hfill{}\Ref{dbgPrint()}

@Man dbgPrint()

11.1.5

\hfill{}\Ref{export2Origin()}

@Man export2Origin()

11.1.6

\hfill{}\Ref{getFilename()}

@Man getFilename()

11.1.7

\hfill{}\Ref{getFilenameWithoutExtension()}

@Man getFilenameWithoutExtension()

11.1.8

\hfill{}\Ref{getPath()}

@Man getPath()

This page has been automatically generated with DOC++
DOC++ is ©1995 by Roland Wunderling
Malte Zöckler

11.1.9

\hfill{}\Ref{interpolatingFunctionQ()}

@Man interpolatingFunctionQ()

11.1.10

\hfill{}\Ref{makeUsage()}

@Man makeUsage()

11.1.11

\hfill{}\Ref{num()}

@Man num()

11.1.12

\hfill{}\Ref{outFkt()}

@Man outFkt()

11.1.13

\hfill{}\Ref{outStr()}

@Man outStr()

11.1.14

\hfill{}\Ref{readListWithComment()}

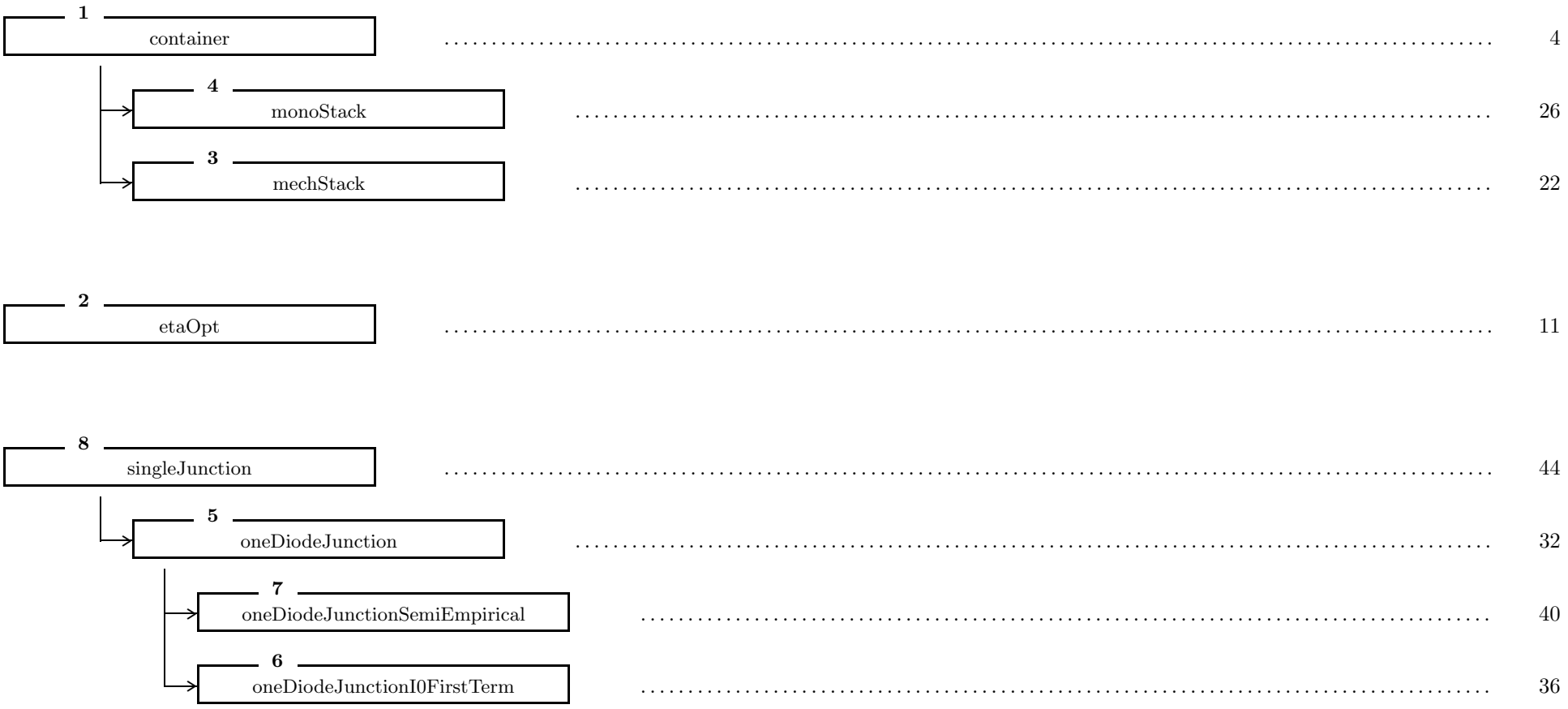
@Man readListWithComment()

11.1.15

\hfill{}\Ref{showStatus()}

@Man showStatus()

Class Graph



This page has been automatically generated with DOC++
DOC++ is ©1995 by Roland Wunderling
Malte Zöckler