MatPar

```
#- $Id: //tcad/support/main/examples/opto/solarcell/iiiv/sc-iiiv-dj-
epi/mpr_mpr.cmd#2 $
#setdep @previous@
#----------
# global user variables for e.g. material processing
set temp 300

#----------
# load the layer variables. The epi tool saves the layer data typically in
# ./n@node|epi@_epi.tcl layer data can be accessed through
source "@pwd@/@epitcl@"

#----------
# configure mpr
# specify Global parameter database default is $STREPODIR/pardb
# set mprGlobalParDb ~/db/myrepository

# configure local parameter directory
# set mprLocalParDir ./par

# Specify main parameter file name
# set mprMainParameterFile n@node@_mpr.par

# If set to 1, do not print any lines starting with # in the preprocessed files
# set mprNoComments 0

# Set directory where the preprocessed parameter files are stored
# set mprPpParDir ./npar

# choose the preprocessor used by mpr internal | spp
# set mprPreprocessor internal
```

SDE

```
;;------------------------------------------------------------------------
;;$Id: //tcad/support/main/examples/opto/solarcell/iiiv/sc-iiiv-dj-
epi/sde_dvs.cmd#4 $
;;--------------
;; Create the layer stack,
(sdeepi:create-layerstack "@epicsv@")
;;--------------
;; Create the tcl file:
(sdeepi:tcl "@epicsv@" "@epitcl@")
;;--------------
;; Create the scm file:
(sdeepi:scm "@epicsv@" "@episcm@")
;;--------------
(display "  Reading in epi layer system") (newline)
(load "@episcm@")
;;--------------
(display "  Geometric variables") (newline)
(define wfront @wfront@)
(define dmgf 0.107)
```

```
(define dtiox 0.065)
(define dtdmetal 0.001)
;;--------------
(display "  Refinements variables") (newline)
(define myRefOptScale 1.2)
(define myRefOptYmin 0.001)
(define myRefOptDepth 2)
(define myRefGlobalX 50)
(define myRefGlobalY 1.5)
(define myRefFrontContactOffset 2)
(define myRefFrontContactDepth 0.3)
(define myRefFrontContactX 5)
;;--------------
(display "  other layer variables") (newline)
;; extract tunneldiode doping level to create component cells
(define Ntdn (abs @Ntdn@))
(define Ntdp (abs @Ntdp@))

;;----------------------------------------------------------------------
(display "Modifying EPI") (newline)
;;--------------
(display "  etching cap") (newline)
(sdegeo:set-default-boolean "ABA")
;; In 1d remove cap completely
(define delme (sdegeo:create-rectangle
      (position wfront Y0_cap 0 )
      (position Xmax Y1_cap 0 ) "xx" "xx"))
(entity:delete delme)
;;--------------
(display "  deposit ARC") (newline)
(sdegeo:create-rectangle
      (position wfront Y0_topfsf   0 )
      (position Xmax (- Y0_topfsf dtiox)  0 )
      "TiOx" "arc1")
(sdegeo:create-rectangle
      (position wfront (- Y0_topfsf dtiox)  0 )
      (position Xmax (- Y0_topfsf dtiox dmgf)  0 )
      "MgF" "arc2")

;;----------------------------------------------------------------------
(display "Add Contacts") (newline)
;--------------
(display "  top contact") (newline)
(sdegeo:define-contact-set "cathode" 4  (color:rgb 1 0 0 ) "##" )
(sdegeo:set-current-contact-set "cathode")
;; In 1d put cathod on top of fsf, in 2d on top of cap
(if (> wfront 0)
  (begin
    (define x (/ wfront 2))
    (define y Y0_cap)
    ) ;else
  (begin
    (define x (/ Xmax 2))
    (define y Y0_topfsf)
    ));endif
(sdegeo:define-2d-contact (find-edge-id (position x y 0)) "cathode")
;;-------------
```

```
(display "  bottom contact") (newline)
(sdegeo:define-contact-set "anode" 4  (color:rgb 1 0 0 ) "##" )
(sdegeo:set-current-contact-set "anode")
(sdegeo:define-2d-contact (find-edge-id (position (/ Xmax 2) Ymax 0)) "anode")
(sde:refresh) ; to display contacts in gui mode

;;------------------------------------------------------------------------
(display "Creating Component") (newline)
(sdedr:define-constant-profile "ndop_component" "ArsenicActiveConcentration"
Ntdn)
(sdedr:define-constant-profile "pdop_component" "BoronActiveConcentration" Ntdp)
#if [string match "*topcell*" "@geo@"]
;-------------
(display "  topcell\n")
(sdedr:define-refinement-window "p_ref_component" "Rectangle"
  (position Xmin Y0_tdn 0)
  (position Xmax Ymax 0) )
(sdedr:define-constant-profile-placement "p_component" "pdop_component"
"p_ref_component" 0 "Replace")
#elif [string match "*bottomcell*" "@geo@"]
;-------------
(display "  bottomcell\n")
(sdedr:define-refinement-window "n_ref_component" "Rectangle"
  (position Xmin Ytop 0)
  (position Xmax Y1_tdp 0) )
(sdedr:define-constant-profile-placement "n_component" "ndop_component"
"n_ref_component" 0 "Replace")
#endif

;;------------------------------------------------------------------------
(display "Add Refinements") (newline)
;-------------
(if (> wfront 0)
      (begin
        (display "  global refinement") (newline)
        (sdedr:define-refinement-window "global" "Rectangle"
          (position  Xmin Ymin 0)
          (position  Xmax Ymax 0 ) )
        (sdedr:define-refinement-size "global"
          myRefGlobalX myRefGlobalY 0
          myRefGlobalX myRefGlobalY 0)
        (sdedr:define-refinement-placement "frontContact" "frontContact"
"frontContact" )
        ));endif
;;-------------
(display "  optical refinement") (newline)
(sdedr:define-refinement-window "optics" "Rectangle"
      (position wfront Y0_topfsf 0)
  (position Xmax myRefOptDepth 0) )
(sdedr:define-multibox-size "optics" Xmax 10 Xmax myRefOptYmin 1 myRefOptScale )
(sdedr:define-multibox-placement "optics" "optics" "optics" )
;;-------------
(if (> wfront 0)
      (begin
           (display "  front contact refinement") (newline)
           (sdedr:define-refinement-window "frontContact" "Rectangle"
             (position  Xmin Ymin 0)
             (position  (+ Xmin wfront myRefFrontContactOffset) (+ Y0_topfsf
myRefFrontContactDepth) 0) )
          (sdedr:define-refinement-size "frontContact"
               (/ wfront myRefFrontContactX) 10 0
               (/ wfront myRefFrontContactX) 10 0 )
          (sdedr:define-refinement-placement "frontContact" "frontContact"
"frontContact" )
          ));endif

;;------------------------------------------------------------------------
(display "saving & meshing") (newline)
(sde:build-mesh "snmesh" "-a -y 1e6 -r 1e6" "n@node@_msh")
(display "... done.") (newline)
```

SDEVICE_DES

```
#setdep @previous@

*--------------------------------------------------
* create bias spectrum.
* For IV cut of spectrum at wend.
* For EQE scale intensity for non-investigated cells
!(
set spectrum "@pwd@/par/spectra/@spectrum@"
set wcrossover 0.65 ;# bandedge wavelength of top cell in um
set wend 0.9 ;# end of spectrum in um
set sfactor 2.0 ;# scaling factor for the bias light of the non-investigated cell
spectrum range.
set signalIntensity 0.01 ;# intensity of the measurement light in W/cm^2
# read in spectrum

set fid [open $spectrum r]
set wilist {}
set headerlist {}
while { [gets $fid LINE] >= 0 } {
     set LINE [string trim $LINE]
     if {[string range $LINE 0 0] == "#" || [string length $LINE] == 0}
{continue}
     if {![regexp {^[0-9\.eE\-\+]+} [lindex $LINE 0]]} {
       lappend headerlist $LINE
       continue
     }
     set w [lindex $LINE 0]
     set i [lindex $LINE 1]
     if {$w <= $wend} {lappend wilist [list $w $i]}
}
close $fid
set wilist [lsort -real -index 0 $wilist]
set wstart [lindex $wilist 0 0]
set wend [lindex $wilist end 0]
set wsteps [llength $wilist]
# write out bias spectrum with reduced values for investigated cell
# transform wavelength list to time list [0..1]
set fid  [open "pp@node@_spec.txt" w]
foreach h $headerlist { puts $fid $h }
```

```
set timelist {}
foreach wi $wilist {
  set w [lindex $wi 0]
  set i [lindex $wi 1]
  # apply sfactor to not investigate cell's spectrum
  if {[string match "*eqetop*" "@mode@"] && $w >= $wcrossover || \
      [string match "*eqebot*" "@mode@"] && $w <= $wcrossover} {
    set i [expr $i*$sfactor]
  }
  puts $fid "$w $i"
     lappend timelist [expr 1.*($w-$wstart)/($wend-$wstart)]
}
close $fid
# put some info on the spectrum in the preprocessed command file
puts "* wavelength range: \[$wstart, $wend\] entries: $wsteps"
puts "* wavelength crossover: $wcrossover sfactor: $sfactor"
)!

*---------------------------------------------------
***** Sourcing the @epitcl@ file for geometric parameters used for nonlocal mesh
#if ![file exists "@epitcl@"]
* Warning: File @epitcl@ does not exists yet. Run epi node first.
#exit
#endif
!(source @epitcl@)!

*---------------------------------------------------
File {
  *-Input
  Grid=      "@tdr@"
  Parameter= "@mprpar@"
  #if "@spectrum@" != "_"
  * use spectrum from tcl pp above
  IlluminationSpectrum= "pp@node@_spec.txt"
  #endif
  *-Output
  Output  = "@log@"
  Current=   "@plot@"
  Plot=      "@tdrdat@"
}

*---------------------------------------------------
Electrode {
  { Name= "anode"  Voltage= 0.0 }
  { Name= "cathode"  Voltage= 0.0}
}

*---------------------------------------------------
Plot {
  *- Doping Profiles
  DopingConcentration DonorConcentration AcceptorConcentration
  *- Band structure
  BandGap BandGapNarrowing ElectronAffinity
  ConductionBandEnergy ValenceBandEnergy
  eQuasiFermiEnergy hQuasiFermiEnergy
  *- Carrier Densities:
  eDensity hDensity
  EffectiveIntrinsicDensity IntrinsicDensity
  *- Fields, Potentials and Charge distributions
  ElectricField/Vector
  Potential
  SpaceCharge
  *- Currents
  Current/Vector eCurrent/Vector  hCurrent/Vector
  CurrentPotential  * for visualizing current flow lines
  eMobility hMobility
  *- Generation/Recombination
  SRHRecombination AugerRecombination TotalRecombination SurfaceRecombination
  RadiativeRecombination eLifeTime hLifeTime
  #if "@spectrum@" != "_"
  *- Optical Generation
  ComplexRefractiveIndex QuantumYield
  OpticalIntensity AbsorbedPhotonDensity OpticalGeneration
  #endif
  #if [string match "*nlm*" "@model@"]
  eBarrierTunneling hBarrierTunneling NonLocal
  #endif
}

*---------------------------------------------------
CurrentPlot {
  #if [string match "*eqe*" "@mode@"]
  ModelParameter="Wavelength"
  #endif
  #if "@spectrum@" != "_"
  AbsorbedPhotonDensity(Integrate(Semiconductor))
  OpticalGeneration(Integrate(Semiconductor))
  OpticalGeneration(Integrate(Material="GaInP"))
  OpticalGeneration(Integrate(Material="GaAs"))
  #endif
  SRH(Integrate(Semiconductor))
  Auger(Integrate(Semiconductor))
  Radiative(Integrate(Semiconductor))
}

*---------------------------------------------------
* Global physics
Physics {
  AreaFactor= @<1e11/wtot>@ * to get current in mA/cm^2
  Fermi
  HeteroInterface
  ThermionicEmission
  #if [string match "*nlm*" "@model@"]
     eBarrierTunneling "NLM"(
          Band2Band
          TwoBand
     )
     hBarrierTunneling "NLM"(
          Band2Band
          TwoBand
     )
     #endif
  #if "@spectrum@" != "_"
  Optics (
    ComplexRefractiveIndex(WavelengthDep(Real Imag))
    OpticalGeneration(
      QuantumYield(StepFunction(EffectiveBandgap)) * generated carriers/photon,
default: 1
      ComputeFromSpectrum(
        * for the floating tunneljunction we need to find the initial solution in
transient
        #if [string match "*transient*" "@model@"]
        TimeDependence(
          WaveTime = (1, 1000)
          WaveTSlope =1
        )
        scaling = 1
        #else
        scaling = @suns@
        #endif
      )
      * for eqe we need an additional monochromatic measurement source
      #if [string match "*eqe*" "@mode@"]
      ComputeFromMonochromaticSource()
      #endif
    )
    Excitation (
      Theta= 0         * Normal incidence
      Polarization= 0.5    * Unpolarized light
      #if [string match "*eqe*" "@mode@"]
      Wavelength = !(puts -nonewline $wstart)!
      Intensity = 0
      #endif
      Window (
        Line (
          * for 1D we have no front metalization x=[0..1]
          #if @wfront@ == 0
          X1=0
          #else
          X1= @<wfront+1.e-6>@
          #endif
          X2= @wtot@
        ) *end Line
      ) * end window
    ) * end Excitation
    OpticalSolver (
      TMM (
        IntensityPattern= StandingWave
        LayerStackExtraction (
          * assume infinite thick substrate at the rear
          Medium (
            Location= bottom
            Material= "GaAs"
          )
        )*end LayerStackExtraction
      ) *end TMM
    ) * end OpticalSolver
  ) * end optics
  #endif
}
```

```
*--------------------------------------------------
* Regionwise specific physics
Physics (material="AlInP") {
  EffectiveIntrinsicDensity(NoBandgapNarrowing)
  Mobility()
  Recombination( Radiative )
}

Physics (material="GaInP") {
  EffectiveIntrinsicDensity(NoBandgapNarrowing)
  Mobility()
  Recombination( Radiative )
}

Physics (Region="topbase") {
  EffectiveIntrinsicDensity(NoBandgapNarrowing)
  Mobility()
  Recombination( Radiative )
  #if "@spectrum@" != "_"
  * for thick regions avoid interpolation errors due to non resolved standing
wave patterns
  Optics(OpticalSolver(TMM(IntensityPattern = Envelope)))
  #endif
}

Physics (material="GaAs") {
  Mobility(DopingDependence)
  Recombination(
    SRH Auger Radiative
  )
  EffectiveIntrinsicDensity(
    BandgapNarrowing( TableBGN )
  )
}

Physics (Region="botbase") {
  Mobility(DopingDependence)
  Recombination( SRH Auger Radiative )
  EffectiveIntrinsicDensity(
    BandgapNarrowing( TableBGN )
  )
  #if "@spectrum@" != "_"
  * for thick regions avoid interpolation errors due to non resolved standing
wave patterns
  Optics(OpticalSolver(TMM(IntensityPattern = Envelope)))
  #endif
}

*--------------------------------------------------
Math{
  Iterations= 16
  Extrapolate
  RhsMin=1e-15
  RHsMax=1e15
  RHSFactor=1e20
  RelErrControl
  ErReff(electron)= 100
  ErReff(hole)= 100
  #if [string match "*extended*" "@model@"]
  * for floating regions use extended precision for convergence
  ExtendedPrecision # (128) in case of convergence issues
  Digits= 15
  #else
  Digits= 6
  #endif
  DirectCurrentComputation
  ExitOnFailure
  -ExitOnUnknownParameterRegion
  #if "@spectrum@" != "_"
  * for IV stop voltage ramp after Voc
  BreakCriteria {
    Current (Contact= "cathode" minval= -1e-3)
  }
  #endif
  #if [string match "*nlm*" "@model@"]
  Nonlocal "NLM" (
    RegionInterface="tdn/tdp"
    * get length for non local mesh from epi layer information
    Length = !(puts -nonewline [expr ($epi(region,tdn,thickness)-0.5e-3)*1e-4])!
# [cm] distance to anchor point
    Permeation = !( puts -nonewline [expr ($epi(region,tdp,thickness)-0.5e-3)*1e-
4])!
    * non local mesh created in both directions, one is enough.
    -Transparent(region="tdp")
  )
  #endif
}

*---------------------------------------------------
Solve {
  *------------------------------------------------
  #if "@mode@" == "Eg"
  * for band diagram plots
  Poisson
  Plot (FilePrefix= "n@node@_Poisson")
  *------------------------------------------------
  #else
  * find initial solution
  NewCurrentPrefix= "tmp_"
  Poisson
  #if [string match "*transient*" "@model@"]
  * transient to switch on light and for have floating region converge ...
  Transient(
    initialtime=0 finaltime= 1 initialstep=1e-2 minstep=1e-9 maxstep= 1
  ) {
    Coupled {Poisson Electron Hole }
  }
  * further ramp light to high concentration
  Quasistationary (
       DoZero
       InitialStep = @<10./suns>@ MaxStep = 1 Minstep = 1e-12 Increment = 5
Decrement = 5
       Goal { ModelParameter="Scaling" value=@suns@} * ramp up light
     ) {
```

```
     Coupled { Poisson electron Hole }
}
#endif
NewCurrentPrefix= ""

*-----------------------------------------------
#if "@mode@" == "jsc"
* short circuit current only
Coupled (Iterations= 100){ Poisson Electron Hole }
Plot (FilePrefix= "n@node@_jsc")
*-----------------------------------------------
#elif "@mode@" == "iv"
* light IV curve
Coupled (Iterations= 100){ Poisson Electron Hole }
Plot (FilePrefix= "n@node@_jsc")
* define parameters for voltage ramps depending on cell type
#if [string match "*topcell*" "@geo@"]
#define _V1  1.0
#define _di1 0.5
#define _dm1 1.0
#define _V2 1.5
#define _di2 0.05
#define _dm2 0.05
#elif [string match "*bottomcell*" "@geo@"]
#define _V1  0.6
#define _di1 0.5
#define _dm1 1.0
#define _V2 1.2
#define _di2 0.05
#define _dm2 0.05
#elif [string match "*tandemcell*" "@geo@"]
#define _V1  1.9
#define _di1 0.5
#define _dm1 1.0
#define _V2 2.7
#define _di2 0.05
#define _dm2 0.05
#endif
* quick coarse ramp right before mpp
Quasistationary (
  InitialStep= _di1 MinStep= 5e-4  MaxStep= _dm1 Increment= 2 Decrement= 2
  #Plot {Range= (0 1) Intervals= 3}
  Goal { Name="anode" Voltage= _V1 }
){ Coupled { Poisson Hole Electron }
  #Plot(FilePrefix= "n@node@_mpp" When(Contact = "anode" voltage = 0.9 ))
}
Plot(FilePrefix= "n@node@_mpp" )
* finer ramp until Voc
Quasistationary (
  InitialStep= _di2 MinStep= 0.001 MaxStep= _dm2 Increment= 2 Decrement= 2
  #Plot {Range= (0 1) Intervals= 3}
  Goal { Name="anode" Voltage= _V2 }
){
  Coupled { Poisson Hole Electron }
  #Plot(FilePrefix= "n@node@_0.8V" When(Contact = "anode" voltage = 0.8 ))
}
```

```
  *-----------------------------------------------
  #elif [string match "*eqe*" "@mode@"]
  * get bias current without monochromatic light, monochormatic intensity=0 here
  NewCurrentPrefix = "bias_"
  Coupled {Poisson Electron Hole}
  * switch on monochromatic light
  NewCurrentPrefix = ""
  Quasistationary (
    InitialStep = 1  MaxStep = 1  Minstep = 1e-5
    Goal { modelParameter="Intensity" value=!(puts -nonewline $signalIntensity)!
}
  ) {  Coupled (Iterations=30) {Poisson Electron Hole} }
  # Plot (FilePrefix="n@node@_bias")
  * ramp through wavelength list of spectrum
  Quasistationary (
    InitialStep = 1  MaxStep = 1  Minstep = 0.0001
    Goal { modelParameter="Wavelength" value=!(puts -nonewline $wend)! }
  ) {
     Coupled(Iterations=100) {Poisson Electron Hole}
     # Plot(FilePrefix=n@node@ NoOverwrite)
     CurrentPlot(Time=(!(puts -nonewline "[join [lrange $timelist 1 end]
"\;"]")!))
  }
  #endif
  System("rm -f tmp*") *remove the plots we dont need anymore
  #endif
}
```