```
###*****###
# Poission Equation Procedure #

#This first procedure can be used to set Poisson's Equation for a single material using the variable DevPsi

#Solutions Used
#DevPsi - the device potential and is the controlling variable for this pde
#Elec - the electron concentration
#Hole - the hole concentration
#Donor - the ionized donors - can be complicated with partial ionization
#Acceptor - the ionized acceptors - can be complicated with partial ionization

#Solutions Created
#Econd - the conduction band edge
#Eval - the valence band edge

#Parameters Set
#DampValue is set to a thermal voltage at toom temperature
#AbsError is set to 1mV
#RelError is set to 0.01

#Parameters Used
#RelEps is the relative permittivity, and should be in the PDB as a double "$Mat DevPsi RelEps"
#Affinity is the electron affinity, and should be in the PDB as a double "$Mat Affinity"
#Eg is the band gap, and should be in the PDB as a double "$Mat Eg"

#the electronic charge and permittivity of free space
set q 1.60218e-19
set eps0 8.854e-14

#set Poisson's Equation using the above for the material called
proc Poisson {Mat} {
  global q eps0

  pdbSetDouble $Mat DevPsi DampValue 0.025
  pdbSetDouble $Mat DevPsi Abs.Error 0.001
  pdbSetDouble $Mat DevPsi Rel.Error 0.01

  solution add name=Econd solve $Mat const val = "((-[pdbGetDouble $Mat Affinity])+(DevPsi))"
  solution add name=Eval solve $Mat const val = "((-[pdbGetDouble $Mat Affinity])-([pdbGetDouble
$Mat Eg])+(DevPsi))"

  set eqn "- ($eps0 * [pdbDelayDouble $Mat DevPsi RelEps] * grad(DevPsi) / $q) + Donor - Elec + Hole -
Acceptor + Doping"
  pdbSetString $Mat DevPsi Equation $eqn
}

proc Poisson_Ins {Mat} {
  global q eps0

  pdbSetDouble $Mat DevPsi DampValue 0.025
```

```
   pdbSetDouble $Mat DevPsi Abs.Error 0.001
   pdbSetDouble $Mat DevPsi Rel.Error 0.01

   set eqn "- ($eps0 * [pdbDelayDouble $Mat DevPsi RelEps] * grad(DevPsi) / $q) "
   pdbSetString $Mat DevPsi Equation $eqn
}

proc Poisson_metal {Mat} {
   global q eps0

   pdbSetDouble $Mat DevPsi DampValue 0.025
   pdbSetDouble $Mat DevPsi Abs.Error 0.001
   pdbSetDouble $Mat DevPsi Rel.Error 0.01

   set eqn "- ($eps0 * [pdbDelayDouble $Mat DevPsi RelEps] * grad(DevPsi) / $q) "
   pdbSetString $Mat DevPsi Equation $eqn

   set eqn "- ($eps0 * [pdbDelayDouble $Mat DevPsi RelEps] * grad(Qfn) / $q) "
   pdbSetString $Mat Qfn Equation $eqn

   set eqn "- ($eps0 * [pdbDelayDouble $Mat DevPsi RelEps] * grad(Qfp) / $q) "
   pdbSetString $Mat Qfp Equation $eqn
}

###*****###
# Donor Trap Procedure #

#This procedure creats a partially ionized donor state and adds it to the existing donors
#this procedure creates a deep level donor trap in material Mat, with total concentration Ntrap at
#energy of Etrap.  The traps are distributed via a Gaussian in energy space around Etrap with a full width
half max
#of the Gaussian given by Efwhm

#Solutions used
#Econd - the position of the conduction band
#Donor - the current donor number, which is modified by this routine!
#ETemp - the local Electron temperature, equal to lattice temperature if no energy balance

#boltzmann's constant and the thermal voltage
set k 1.38066e-23

proc DonorTrap {Mat Ntrap Etrap Efwhm} {
   global k q

   set Donor [solution name=Donor $Mat print]
   set Vt ($k*ETemp/$q)

   #do a switch so we can figure out testing...
   if {$Efwhm == 0.0} {
         set e1 0.0
         set e2 "(($Ntrap) * (1 - (1 / (1 + 0.5 * exp( (Econd - $Etrap - Qfn) / ($Vt) )))))"
```

```
        set e3 0.0
    } else {
        #set the evaluation point for the Gaussian-Hermite Quadrature
        set Off [expr sqrt(12.0)*$Efwhm/2.0]
        set e1 "(($Ntrap/6.0) * (1 - (1 / (1 + 0.5 * exp( (Econd - $Etrap - $Off - Qfn) / ($Vt) )))))"
        set e2 "((2.0*$Ntrap/3.0) * (1 - (1 / (1 + 0.5 * exp( (Econd - $Etrap - Qfn) / ($Vt) )))))"
        set e3 "(($Ntrap/6.0) * (1 - (1 / (1 + 0.5 * exp( (Econd - $Etrap + $Off - Qfn) / ($Vt) )))))"
    }
    set Donor "$Donor + $e1 + $e2 + $e3"
    solution name=Donor $Mat solve const val = "$Donor"
}

###*****###
# Donor Trap Procedure, other formulation#

#This procedure creats a partially ionized donor state and adds it to the existing donors
#this procedure creates a deep level donor trap in material Mat, with total concentration Ntrap at
#energy of Etrap.

#Solutions used
#Elec - the electron concentration
#Donor - the current donor number, which is modified by this routine!

proc DonorTrap_trans {Mat Ntrap Etrap} {
    global k q

set Donor [solution name=Donor $Mat print]
    set Vt ($k*ETemp/$q)

    set eqn "($Ntrap *(1 / (1 + (2 * (Elec / [pdbDelayDouble $Mat Elec Nc]) * (exp( $Etrap / ($Vt)))))))"
    set Donor "$Donor + $eqn"
    solution name=Donor $Mat solve const val = "$Donor"
 }

###*****###
# Acceptor Trap Procedure #

#This procedure creats a partially ionized donor state and adds it to the existing donors
#this procedure creates a deep level donor trap in material Mat, with total concentration Ntrap at
#energy of Etrap.  The traps are distributed via a Gaussian in energy space around Etrap with a full width
half max
#of the Gaussian given by Efwhm

#Solutions used
#Eval - the position of the conduction band
#Acceptor - the current donor number, which is modified by this routine!
#HTemp - the local Electron temperature, equal to lattice temperature if no energy balance

#boltzmann's constant and the thermal voltage
set k 1.38066e-23
```

```
proc AcceptorTrap {Mat Ntrap Etrap Efwhm} {
   global k q

   set Acceptor [solution name=Acceptor $Mat print]
   set Vt ($k*HTemp/$q)

   #do a switch so we can figure out testing...
   if {$Efwhm == 0.0} {
         set e1 0.0
         set e2 "(($Ntrap) / (1 + 4.0 * exp( (Eval + $Etrap - Qfp) / ($Vt) )))"
         set e3 0.0
   } else {
         #set the evaluation point for the Gaussian-Hermite Quadrature
         set Off [expr sqrt(12.0)*$Efwhm/2.0]
         set e1 "(($Ntrap/6.0) * ((1 / (1 + 4.0 * exp( (Eval + $Etrap + $Off - Qfp) / ($Vt) )))))"
         set e2 "((2.0*$Ntrap/3.0) * ((1 / (1 + 4.0 * exp( (Eval + $Etrap - Qfp) / ($Vt) )))))"
         set e3 "(($Ntrap/6.0) * ((1 / (1 + 4.0 * exp( (Eval + $Etrap + $Off - Qfp) / ($Vt) )))))"
   }
   set Acceptor "$Acceptor + $e1 + $e2 + $e3"
   solution name=Acceptor $Mat solve const val = "$Acceptor"
}
```