Kripke - Scattering Kernel

```
for(int nm = 0; nm < num_moments; ++nm)
   for(int g = 0; g < num_groups; ++g)
      for(int gp = 0; gp < num_groups; ++gp)
         for(int zone = 0; zone < num_zones; ++zone)
            for(int mix = z_mixed[z]; mix < z_mixed[z]+num_mixed[z]; ++mix) {
               int material = mixed_material[mix];
               double fraction = mixed_fraction[mix];
               int n = moment_to_coeff[nm];

               #####
               # Address calculation to be in
               #####

               *phi_out += *sigs * *phi * fra
            }
```

```
datalayout=enum("DZG","DGZ","GDZ","GZD","ZDG","ZGD");
CodeReg Scattering {
  if (datalayout == "DGZ") {
     omploop="0.0.0.0";
  } elif (datalayout == "GDZ") {
     looporder=[1,2,0,3,4];
     omploop="0.0.0.0";
  } elif (datalayout == "GZD") {
     looporder=[1,2,3,4,0];
     omploop="0.0.0";
  } elif (datalayout == "ZGD") {
     looporder=[3,4,1,2,0];
     omploop="0";
  } elif (datalayout == "ZDG") {
     looporder=[3,4,0,1,2];
     omploop="0";
  } elif (datalayout == "DZG") {
     looporder=[0,3,4,1,2];
     omploop="0.0";
  }
  sourcepath="scatter_"+datalayout+".txt";
  BuiltIn.Altdesc(stmt="0.0.0.0.0.3", source=sourcepath);
  RoseLocus.Interchange(order=looporder);
  RoseLocus.LICM();
  RoseLocus.ScalarRepl();
  Pragma.OMPFor(loop=omploop);
}
```

Kripke - Scattering Kernel

```
for(int nm = 0; nm < num_moments; ++nm)
   for(int g = 0; g < num_groups; ++g)
      for(int gp = 0; gp < num_groups; ++gp)
         for(int zone = 0; zone < num_zones; ++zone)
            for(int mix = z_mixed[z]; mix < z_mixed[z]+num_mixed[z]; ++mix) {
               int material = mixed_material[mix];
               double fraction = mixed_fraction[mix];
               int n = moment_to_coeff[nm];

               #####
               # Address calculation to be in
               #####

               *phi_out += *sigs * *phi * fra
            }
```

```
datalayout=enum("DZG","DGZ","GDZ","GZD","ZDG","ZGD");
CodeReg Scattering {
  if (datalayout == "DGZ") {
     omploop="0.0.0.0";
  } elif (datalayout == "GDZ") {
     looporder=[1,2,0,3,4];
     omploop="0.0.0.0";
  } elif (datalayout == "GZD") {
     looporder=[1,2,3,4,0];
     omploop="0.0.0";
  } elif (datalayout == "ZGD") {
     looporder=[3,4,1,2,0];
     omploop="0";
  } elif (datalayout == "ZDG") {
     looporder=[3,4,0,1,2];
     omploop="0";
  } elif (datalayout == "DZG") {
     looporder=[0,3,4,1,2];
     omploop="0.0";
  }
  sourcepath="scatter_"+datalayout+".txt";
  BuiltIn.Altdesc(stmt="0.0.0.0.0.3", source=sourcepath);
  RoseLocus.Interchange(order=looporder);
  RoseLocus.LICM();
  RoseLocus.ScalarRepl();
  Pragma.OMPFor(loop=omploop);
}
```