```
{
      long i, j;
      long ix, jx, iDir, jDir;
      int Direct[2] = {1, -1};

      CellList.Add(Index(jStart,iStart));

static BOOL NewWay = TRUE;
if (NewWay) {
      for (iDir=0; iDir<2; iDir++) {
            for (ix=iDir; ix<=loop; ix++) {
                  i = iStart + Direct[iDir]*ix;
                  if (i<0 || i>=m_ClGrid.GetNRows())
                        break;

//                  if (Checked[Index(jStart, i)] && ix)      // if hit a checked box, finished moving this
direction
//                        continue; //break;

                  for (jDir=0; jDir<2; jDir++) {
                        for (jx=jDir; jx<=loop; jx++) {     // 2nd pass starts at jx=1 since checked 0
already
                              if (!ix && !jx)
                                    continue;

                              j = jStart + Direct[jDir]*jx;
                              if (j<0 || j>=m_ClGrid.GetNCols())
                                    break;

                              if (Checked[Index(j,i)])      // if hit a checked box, finished moving this
direction
                                    continue;   //break;

                              BOOL CurNoMerge = m_CurClust->IsNoMerge(m_ClGrid.GetMember(Index(j,i)),
m_NM_Count);
                              if ((m_ClGrid.GetMember(Index(jStart,iStart)) ==
m_ClGrid.GetMember(Index(j,i))) ||
                                                      (!NoMerge && !CurNoMerge)) {

                                    BOOL CloseEnough = TRUE;

                                    if (m_ClGrid.GetLines(Index(j,i)))
                                          CloseEnough = WithinMaxDist(CellList, Index(j,i), WithinClusts,
RejectClusts);
```

© PNR and Associates, All rights reserved

```
                                    if (CloseEnough) {
                                          Checked[Index(j,i)] = TRUE;

                                          if (m_ClGrid.GetLines(Index(j,i))) {
//                                              while (jx >= jDir) {
//                                                    j = jStart + Direct[jDir]*jx;
//                                                    Checked[Index(j,i)] = 2;
//                                                    jx--;
//                                              }

                                              CheckAdjacent(i, j, Checked, CellList, loop, Mergable,
NoMerge, WithinClusts, RejectClusts);
//                                              break;
                                          }
                                    }
                              }
                        }
                  }
            }
      }
/*
      for (ix=0; ix<loop; ix++) {
            for (jx=0; jx<loop; jx++) {
                  if (!ix && !jx)
                        continue;

                  i = iStart + ix;
                  j = jStart + jx;

                  if (Checked[Index(j,i)])
                        break;
            }

            if (!jx)
                  break;
      }
*/


} else {
      int BegLoop = 0-loop;
      int EndLoop = loop+1;
```

© PNR and Associates, All rights reserved

```
//    for (int n=0; n<loop; n++) {
            for (int ix=BegLoop; ix<EndLoop; ix++) {
                  for (int jx=BegLoop; jx<EndLoop; jx++) {
//                for (int jpos=0; j<2; j++) {
//                      int jx = (j) ? -loop : loop;

                        if (!ix && !jx)
                              continue;

                        i = iStart + ix;
                        j = jStart + jx;

                        if (i<0 || j<0 || i>=m_ClGrid.GetNRows() || j>=m_ClGrid.GetNCols())
                              continue;

                        if (!Checked[Index(j,i)])
                        {
//                            BOOL NewMergable = m_CurClust->IsMergable((int)m_ClGrid.GetMember(Index(j,i)),
m_LM_Count);
                              BOOL CurNoMerge = m_CurClust->IsNoMerge((int)m_ClGrid.GetMember(Index(j,i)),
m_NM_Count);

//                            if ((m_ClGrid.GetMember(Index(jStart,iStart)) ==
m_ClGrid.GetMember(Index(j,i))) ||
//                                      ((Mergable || (!Mergable && NewMergable)) &&
//                                      (!NoMerge || m_ClGrid.GetMember(Index(j,i))==0) && !CurNoMerge)) {

                              if ((m_ClGrid.GetMember(Index(jStart,iStart)) ==
m_ClGrid.GetMember(Index(j,i))) ||
                                                      (!NoMerge && !CurNoMerge)) {

                                    BOOL CloseEnough = TRUE;

                                    if (m_ClGrid.GetLines(Index(j,i)))
                                          CloseEnough = WithinMaxDist(CellList, Index(j,i), WithinClusts,
RejectClusts);

                                    if (CloseEnough) {
                                          Checked[Index(j,i)] = TRUE;

//                                        if (!NewMergable)
//                                              Mergable = NewMergable;
//static BOOL ClusterEmpties = FALSE;
                                          if (m_ClGrid.GetLines(Index(j,i)))
```

© PNR and Associates, All rights reserved

```
                                                    CheckAdjacent(i, j, Checked, CellList, loop, Mergable,
NoMerge, WithinClusts, RejectClusts);
//                                          else if (ClusterEmpties)
//                                                  CellList.Add(Index(j,i));
                                        }
                                    }
                                }
                            }
                        }
//      }
}

}

void CClustComp::ComputeClusters(SPECS* Specs, BOOL Outliers, COutput *OutputFile)
{
        // Assign member variables
        m_NM_Count = Specs->nMaxMerge;
        if (!Outliers)
        {
                m_MinLines = Specs->nMinN;
                m_LM_Count = Specs->nLimitedMerge;
        }
        else
        {
                m_MinLines = 0;
                m_LM_Count = Specs->nMinN;
        }


        // Declare and initialize
        long nRows = m_ClGrid.GetNRows();
        long nCols = m_ClGrid.GetNCols();
        long dFeet = m_ClGrid.GetDistance();

        if (!Outliers)
                delete m_CurClust;
        if (!m_CurClust)
                m_CurClust = new CClustSet(m_nPoints, dFeet);

        m_CurClust->SetOutlier(Outliers, m_LastCluster);
        CClustSet *NextClust = new CClustSet(m_CurClust);
        CClustSet *MoveClust = NextClust;
        NextClust = m_CurClust;
```

© PNR and Associates, All rights reserved

```
      m_CurClust = MoveClust;
      NextClust->ResetClusts();

      char *Checked = new char[nRows * nCols];

      long n;
      long TempClust = 10000;

      // Cluster
      long iterations = Specs->nMaxClusFt / Specs->nGridSize;
      for (int l=0; l<iterations; l++) {
            for (n=0; n<(long)nRows*nCols; n++)
                  Checked[n] = FALSE;

            if (Outliers) {
                  for (n=0; n<(long)m_nPoints; n++) {
                        long row = m_ClGrid.GetPRow(n);
                        long col = m_ClGrid.GetPCol(n);
                        long Member = m_ClGrid.GetMember(Index(col,row));
//                      if (Member && Member < m_CurClust->GetOutStart())
                        if (Member && Member < m_CurClust->m_OutlMemStart)          // If not preset / non-
outlier
                              Checked[Index(col, row)] = TRUE;
                  }
            }

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//static BOOL TestOrder=TRUE;
//if (TestOrder) {
//
//    for (int i=0; i<m_CurClust->GetSize(); i++) {
//    }
//
//
//
//}
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

            for (long i=0; i<nRows; i++) {
                  for (long j=0; j<nCols; j++) {
```

© PNR and Associates, All rights reserved

```
                        ProcessCell(i,j, Checked, l+1, NextClust);

                    }
            }

            CClustSet *MoveClust = NextClust;
            NextClust = m_CurClust;
            m_CurClust = MoveClust;
            NextClust->ResetClusts();

#ifdef _DEBUG
{
      CString Output;
      char buff[360];

      sprintf(buff, "%d Clusters at iteration %d\n", m_CurClust->m_nClusters, l);
      TRACE(buff);
      m_fTrace.WriteString(buff);

      for (long i=0; i<m_CurClust->m_nClusters; i++) {
            CDWordArray *CurList = (CDWordArray *)m_CurClust->m_ACellList[i];
            sprintf(buff, "%04d)   %4d   %4ld   %f\n", i+1, m_CurClust->m_Cluster[i], CurList->GetSize(),
m_CurClust->m_NumLines[i]);
            TRACE(buff);
            m_fTrace.WriteString(buff);
      }

      TRACE("\n");
      m_fTrace.WriteString("\n");
}
#endif

      }

      for (n=0; n<(long)m_nPoints; n++) {
            long row = m_ClGrid.GetPRow(n);
            long col = m_ClGrid.GetPCol(n);
            m_CurClust->SetPointClust(n, m_ClGrid.GetMember(Index(col, row)));
      }

//=========================================
//CPointArray coord;
//ClusterCoord(0, coord);
```

© PNR and Associates, All rights reserved

```
//=========================================

      delete[] Checked;
NextClust->m_ACellList.RemoveAll();
      delete NextClust;
}

static BOOL TestEncompassed = FALSE;

void CClustComp::ProcessCell(int row, int col, char *Checked, int loop, CClustSet *NextClust)
{
      long n,     CurClust;
      CDWordArray *CellList = NULL;

      if (!Checked[Index(col,row)]) {
            Checked[Index(col,row)] = TRUE;

            if (m_ClGrid.GetLines(Index(col,row))) {
                  CWordArray WithinClusts;
                  CWordArray RejectClusts;

                  if (m_ClGrid.GetMember(Index(col,row)))
                        WithinClusts.Add(m_ClGrid.GetMember(Index(col,row)));

                  CellList = new CDWordArray;
//                BOOL Mergable = m_CurClust->IsMergable(m_ClGrid.GetMember(Index(col,row)), m_LM_Count);
      // $CHANGE$ remove Mergable
BOOL Mergable=TRUE;     // $CHANGE$ remove Mergable

                  BOOL NoMerge = m_CurClust->IsNoMerge(m_ClGrid.GetMember(Index(col,row)), m_NM_Count);
#ifdef _DEBUG
static long brow=180, bcol=96, bloop=1;
if(row==brow && col==bcol && loop == bloop)
      col = col;
#endif

                  CheckAdjacent(row, col, Checked, *CellList, loop, Mergable, NoMerge, WithinClusts,
RejectClusts);          // $CHANGE$ remove Mergable
#ifdef _DEBUG
static long bsize = 10;
if (CellList->GetSize() > bsize)
      col = col;
#endif
```

© PNR and Associates, All rights reserved

```
                CurClust = (WithinClusts.GetSize()==1) ? WithinClusts[0] : 0;

if (TestEncompassed)
                Encompassed(CellList, Checked);

                long CellCount = CellList->GetSize();
                float LineCount = 0;
                for (n=0; n<CellList->GetSize(); n++)
                      LineCount += m_ClGrid.GetLines(CellList->GetAt(n));

                if (LineCount < m_MinLines)
                      CurClust = 0;
                else if (!CurClust || CurClust == -1)
                      CurClust = ++m_LastCluster;

                for (n=0; n<CellList->GetSize(); n++)
                      m_ClGrid.SetMember(CellList->GetAt(n), CurClust);

                if (CurClust) {
                NextClust->Add(CurClust, CellCount, LineCount, CellList);
                } else {
                      delete CellList;
                }
            }
      }
}

void CClustComp::ClusterCoord(int clus, CPointArray& coord)
{
      // Initialize
      long i, j, sz;
      Point pt;
      CDWordArray p;
      CPointArray pXY;

      // Calculate perimiter points
      CPerim* pPerim = new CPerim;
      CDWordArray *ClustList = m_CurClust->GetClustList(m_CurClust->GetCluster(clus));
      for (i = 0; i < ClustList->GetSize(); i++)
      {
            pt.y = (long)floor(ClustList->GetAt(i) / m_ClGrid.GetNCols());
            pt.x = ClustList->GetAt(i) - pt.y * m_ClGrid.GetNCols();
            pXY.Add(pt);
      }
```

© PNR and Associates, All rights reserved

```
pPerim->Compute(pXY, p);
delete pPerim;

// Determine cell coordinates of perimiter line
CLine line;
p.Add(p[0]);
sz = (long)p.GetSize();
coord.Add(pXY.GetAt(p[0]));

for (i = 1; i < sz; i++)
{
      coord.Add(pXY.GetAt(p[i]));
      line.Set(pXY.GetAt(p[i - 1]), pXY.GetAt(p[i]));

      if (line.IsHorizontal())
      {
            if (line.GetDir() == "W")
                  line.UTurn();

            pt.y = line.GetPtA()->y;
            for (j = (long)line.GetPtA()->x + 1;  j < line.GetPtB()->x; j++) {
                  pt.x = j;
                  coord.Add(pt);
            }
      } else {
            if (line.GetDir().GetAt(0) == 'S')
                  line.UTurn();

            BOOL bVert = line.IsVertical();
            if (bVert)
                  pt.x = line.GetPtA()->x;

            for (j = (long)line.GetPtA()->y + 1; j < line.GetPtB()->y; j++) {
                  pt.y = j;
                  if (!bVert) pt.x = (int)((((line.GetPtB()->x - line.GetPtA()->x) / (line.GetPtB()->y
- line.GetPtA()->y)) * (j - line.GetPtA()->y) + line.GetPtA()->x) + 0.5);
                  coord.Add(pt);
            }
      }
}

// Add cell coordinates of inside the perimeter
coord.DeDup();
coord.SortByYX();
```

© PNR and Associates, All rights reserved

```
	sz = coord.GetSize() - 1;	for (i = 0; i < sz; i++) {
		if (coord[i].y == coord[i + 1].y) {
			pt.y = coord[i].y;
			pt.x = coord[i].x + 1;
			while (pt.y == coord[i + 1].y && pt.x < coord[i + 1].x) {
				coord.InsertAt(++i, pt);
				pt.x++;
				sz++;
			}

			i++;
		}
	}

	return;
}


void CClustComp::Encompassed(CDWordArray *ClustList, char *Checked)
{
	// Initialize
	long i, j, sz;
	Point pt;
	CDWordArray p;
	CPointArray pXY;
	CPointArray coord;

	// Calculate perimiter points
	CPerim pPerim;
	for (i = 0; i < ClustList->GetSize(); i++) {
		pt.y = (long)floor(ClustList->GetAt(i) / m_ClGrid.GetNCols());
		pt.x = ClustList->GetAt(i) - pt.y * m_ClGrid.GetNCols();
		pXY.Add(pt);
	}
	pPerim.Compute(pXY, p);

	// Determine cell coordinates of perimiter line
	CLine line;
	p.Add(p[0]);
	sz = (long)p.GetSize();
	coord.Add(pXY.GetAt(p[0]));

	for (i = 1; i < sz; i++) {
		coord.Add(pXY.GetAt(p[i]));
```

© PNR and Associates, All rights reserved

```
            line.Set(pXY.GetAt(p[i - 1]), pXY.GetAt(p[i]));

            if (line.IsHorizontal()) {
                  if (line.GetDir() == "W")
                        line.UTurn();

                  pt.y = line.GetPtA()->y;
                  for (j = (long)line.GetPtA()->x + 1;  j < line.GetPtB()->x; j++) {
                        pt.x = j;
                        coord.Add(pt);
                  }
            } else {
                  if (line.GetDir().GetAt(0) == 'S')
                        line.UTurn();

                  BOOL bVert = line.IsVertical();
                  if (bVert)
                        pt.x = line.GetPtA()->x;

                  for (j = (long)line.GetPtA()->y + 1; j < line.GetPtB()->y; j++) {
                        pt.y = j;
                        if (!bVert) pt.x = (int)((((line.GetPtB()->x - line.GetPtA()->x) / (line.GetPtB()->y
- line.GetPtA()->y)) * (j - line.GetPtA()->y) + line.GetPtA()->x) + 0.5);
                        coord.Add(pt);
                  }
            }
      }

      // Add cell coordinates of inside the perimeter
      coord.DeDup();
      coord.SortByYX();
      sz = coord.GetSize() - 1;
      long Added=0;
      for (i = 0; i < sz; i++) {
            if (coord[i].y == coord[i + 1].y) {
                  pt.y = coord[i].y;
                  pt.x = coord[i].x + 1;
                  while (pt.y == coord[i + 1].y && pt.x < coord[i + 1].x) {
//                      coord.InsertAt(++i, pt);
//                      sz++;
                        long pos = Index((long)pt.x,(long)pt.y);
                        if (!Checked[pos]) {
                              Checked[pos] = TRUE;
                              ClustList->Add(pos);
```

© PNR and Associates, All rights reserved

```
                                Added++;
                            }
                            pt.x++;
                        }
                        i++;
                    }
            }

            return;
}


// CPath.h : header file

#ifndef PATH__H
#define PATH__H

#include "Status.h"
#include "Distance.h"

#include <stdio.h>
#include <math.h>

/////////////////////////////////////////////////////////////////////////////
// CPath

class CPath : public CObject
{
// Construction
public:
        CPath();    // standard constructor
        virtual ~CPath();

// Attributes
protected:
        long m_nCases;
        double m_Dist;
        double m_DistFlag;
        double m_PointMax;
        CStatus* m_Status;

// Functions
public:
```

© PNR and Associates, All rights reserved

```
	void Run(CDistance&, long, CDWordArray&, CStringArray&, CDWordArray* Flag = NULL, CStatus* Stat =
NULL);
	void Run(CDistance&, CDWordArray&, CDWordArray&, CStringArray&, CDWordArray* Flag = NULL, CStatus*
Stat = NULL);

	long GetNCases();
	double GetDist();
	double GetDistFlag();
	double GetPointMax();

protected:
	void NextLabel(CString&, CDWordArray&, CStringArray&);
	void Init(CDWordArray&, CDWordArray&, CDWordArray&, CDWordArray&, CStringArray&);
	void Calculate(CDWordArray&, CDWordArray&, CStringArray&, CDistance&, CDWordArray* Flag = NULL);
};

#endif

// Path.cpp : implementation file

#include "stdafx.h"
#include "Path.h"

CPath::CPath()
{
	m_nCases = 0;
	m_Dist = 0.0;
	m_DistFlag = 0.0;
	m_PointMax = 0.0;
	m_Status = NULL;
}

CPath::~CPath()
{
}

void CPath::Run(CDistance& DistMat, long nCases, CDWordArray& Path, CStringArray& Label, CDWordArray* Flag
/*NULL*/, CStatus* Stat /*NULL*/)
{
	// Initialize computed variables
	m_Dist = 0.0;
	m_DistFlag = 0.0;
	m_PointMax = 0.0;
```

© PNR and Associates, All rights reserved

```
	// Initialize with parameter variables
	m_nCases = nCases;
	m_Status = Stat;

	// Initialize case array for ALL cases
	CDWordArray Cases;
	Cases.SetSize(nCases);
	for (long i = 0; i < nCases; i++)
		Cases[i] = i;

	// Calculate path and total distance
	Calculate(Cases, Path, Label, DistMat, Flag);

	return;
}

void CPath::Run(CDistance& DistMat, CDWordArray& SpecificCases, CDWordArray& Path, CStringArray& Label,
CDWordArray* Flag /*NULL*/, CStatus* Stat /*NULL*/)
{
	// Initialize computed variables
	m_Dist = 0.0;
	m_DistFlag = 0.0;
	m_PointMax = 0.0;

	// Initialize with parameter variables
	m_nCases = SpecificCases.GetSize();;
	m_Status = Stat;

	// Calculate path and total distance
	Calculate(SpecificCases, Path, Label, DistMat, Flag);

	return;
}

/////////////////////////////////////////////////////////////////////////////////
//
//  CPath:  Path functions
//
/////////////////////////////////////////////////////////////////////////////////

void CPath::Init(CDWordArray& Cases, CDWordArray& Assigned, CDWordArray& Unassigned, CDWordArray& Path,
CStringArray& Label)
{
	// Initialize path and cases
```

© PNR and Associates, All rights reserved

```
	Unassigned.SetSize(m_nCases);
	for (long i = 0; i < m_nCases; i++) {
		// Initialize all case paths
		Path[Cases[i]] = 0;

		// Add cases to unassigned list
		Unassigned[i] = Cases[i];
	}

	// Set the first assignment
	Path[Cases[0]] = 1;
	Assigned.Add(Cases[0]);
	Unassigned.RemoveAt(0);

	return;
}

void CPath::Calculate(CDWordArray& Cases, CDWordArray& Path, CStringArray& Label, CDistance& DistMat,
CDWordArray* Flag /*NULL*/)
{
	// Declare variables
	double closest, dist;
	unsigned long pt1, pt2, i, j, iLoop, jLoop;
	CDWordArray Unassigned, Assigned;

	// Initialize
	Init(Cases, Assigned, Unassigned, Path, Label);

	// Determine path
	CString origMsg;
	if (m_Status) origMsg = m_Status->GetText();
	while (Unassigned.GetSize())
	{
		pt1 = 0;
		pt2 = 0;
		closest = DistMat.Get(Assigned[0], Unassigned[0]);

		iLoop = (unsigned)Assigned.GetSize();
		jLoop = (unsigned)Unassigned.GetSize();

		if (m_Status)
			m_Status->SetText(origMsg, iLoop);

		// Assign household to path
```

© PNR and Associates, All rights reserved

```
            for (i = 0; i < iLoop; i++)
                  for (j = 0; j < jLoop; j++)
                  {
                        dist = DistMat.Get(Assigned[i], Unassigned[j]);

                        if (dist > m_PointMax)
                              m_PointMax = dist;

                        if (dist < closest)
                        {
                              pt1 = i;
                              pt2 = j;
                              closest = dist;
                        }
                  }

            // Add to path distance
            m_Dist += closest;
            if (Flag && (Flag->GetAt(Assigned[pt1]) == 1 || Flag->GetAt(Unassigned[pt2]) == 1))
                  m_DistFlag += closest;

            // Assign and remove
            Path[Unassigned[pt2]] = Path[Assigned[pt1]] + 1;
            Label[Unassigned[pt2]] = Label[Assigned[pt1]];
            NextLabel(Label[Unassigned[pt2]], Assigned, Label);

            Assigned.Add(Unassigned[pt2]);
            Unassigned.RemoveAt(pt2);
            Unassigned.FreeExtra();
      }

      return;
}

void CPath::NextLabel(CString& AddTo, CDWordArray& Assigned, CStringArray& Label)
{
      long add, sz, length, i;
      add = 1;
      sz = Assigned.GetSize();
      length = AddTo.GetLength();

      for (i = 0; i < sz; i++)
            if (Label[Assigned[i]].Left(length) == AddTo)
                  if (Label[Assigned[i]].GetLength() == length + 3)
```

© PNR and Associates, All rights reserved

```
                        add++;

      char buff[5];
      if (add < 10)
            AddTo += "00";
      else if (add < 100)
            AddTo += "0";
      AddTo += itoa(add, buff, 10);
}

/////////////////////////////////////////////////////////////////////////////////
//
//  CPath:  Attribute functions
//
/////////////////////////////////////////////////////////////////////////////////

long CPath::GetNCases()
{
      return m_nCases;
}

double CPath::GetDist()
{
      return m_Dist;
}

double CPath::GetDistFlag()
{
      return m_DistFlag;
}

double CPath::GetPointMax()
{
      return m_PointMax;
}

// Perim.h : definition of the CArea class

#ifndef PERIM__H
#define PERIM__H

#include "Global.h"

class CClustGrid;
```

© PNR and Associates, All rights reserved

```
class CPerim : public CObject
{
// Construction
public:
      CPerim();

public:
      void Compute(CPointArray &pXY, CDWordArray& nPerimPts);

protected:
      void ConvexHull(CPointArray& pXY, CDWordArray& nPerim);
      void BendPerimiter(CPointArray& pXY, CDWordArray& nPerimXY);

      double Dot(Point& p1, Point& p2);
      double PointToLine(Point p, Line line, Point& pline, double& pctline);
      double Distance(Point& pt1, Point& pt2);
};

#endif


#include "stdafx.h"
#include "Perimeter.h"

#include "2dch.h"
#include "GrdClust.h"

#include <math.h>

/////////////////////////////////////////////////
CPerim::CPerim()
{
}

void CPerim::Compute(CPointArray &pXY, CDWordArray& nPerimPts)
{
      nPerimPts.RemoveAll();

      if (pXY.GetSize() <= 1) {
            nPerimPts.Add(0);
            return;
      }
```

© PNR and Associates, All rights reserved

```
	// Determine convex hull
	ConvexHull(pXY, nPerimPts);

	// Bend perimiter lines
	//BendPerimiter(pXY, nPerimXY);

	return;
}

void CPerim::ConvexHull(CPointArray& pXY, CDWordArray& nPerim)
{
	// Calculate convex hull (perimiter)
	C2dch* pCH = new C2dch;
	pCH->Compute(&pXY, nPerim);
	delete pCH;
	return;
}

void CPerim::BendPerimiter(CPointArray& pXY, CDWordArray& nPerimXY)
{
	unsigned long nPoints = (unsigned long)pXY.GetSize();
	unsigned long nPerimPts = (unsigned long)nPerimXY.GetSize();
	unsigned long i, j;
	double pctline, PerimLineLen, PtLineLen;

	BOOL bNewPt = FALSE;
	Line line;
	Point ptonline;
	CDWordArray nNotPerimXY;

	// Determine point consideration set to be added to perimeter
	nNotPerimXY.SetSize(nPoints);
	for (i = 0; i < nPoints; i++)
		nNotPerimXY[i] = i;
	for (i = 0; i < nPerimPts; i++)
		nNotPerimXY.RemoveAt(nPerimXY[i]);
	nNotPerimXY.FreeExtra();

	// Bend perimiter lines
	for (i = 0; i < (unsigned long)nPerimXY.GetSize() - 1; i++)
	{
		line.pt1 = pXY[nPerimXY[i]];
		line.pt2 = pXY[nPerimXY[i + 1]];
		PerimLineLen = Distance(line.pt1, line.pt2);
```

© PNR and Associates, All rights reserved

```
            for (j = 0; i < (unsigned long)nNotPerimXY.GetSize(); i++)
            {
                  PtLineLen = PointToLine(pXY[nNotPerimXY[j]], line, ptonline, pctline);
                  if (pctline >= 0.0 && pctline <= 100.0)
                  {
                        if (PtLineLen >= PerimLineLen * 0.25 && PtLineLen <= PerimLineLen * 0.75)
                        {
                              nPerimXY.InsertAt(i + 1, nNotPerimXY[j]);
                              nNotPerimXY.RemoveAt(j);
                              i--;
                              bNewPt = TRUE;
                        }
                  }

                  if (bNewPt)
                        break;
            }
      }

      return;
}

double CPerim::PointToLine(Point p, Line line, Point& pline, double& pctline)
{
      //-----------------------------------------------------------------------
      // Returns the distance of a point to a line.
      //   Calculates the closest x,y point on the line.
      //   Calculates the percentage up the line the closest x,y point occupies.
      //-----------------------------------------------------------------------

      double dist = -1.0;
      Point diff = { p.x - line.pt1.x, p.y - line.pt1.y};

      pctline = Dot(line.pt2, diff) / Dot(line.pt2, line.pt2);
      if (pctline < 0.0 || pctline > 1.0)
            return dist;

      diff.x -= pctline * line.pt2.x;
      diff.y -= pctline * line.pt2.y;

      pline.x = line.pt1.x + pctline * line.pt2.x;
      pline.y = line.pt1.y + pctline * line.pt2.y;
      dist = sqrt(Dot(diff,diff));
```

© PNR and Associates, All rights reserved

```
      return dist;
}

double CPerim::Dot( Point& p1,  Point& p2)
{
      return p1.x * p2.x + p1.y * p2.y;
}

double CPerim::Distance(Point& pt1, Point& pt2)
{
      double x1 = pt1.x;
      double y1 = pt1.y;
      double x2 = pt2.x;
      double y2 = pt2.y;
    double Radius = 3956.0;                          // approx earth radius in miles
    double Radians = acos(-1.0) / 180.0;  // pi / 180
      double Distance = 0.0;

      if (x1 != x2 || y1 != y2)
      {
            x1 *= Radians;
            x2 *= Radians;
            y1 *= Radians;
            y2 *= Radians;

            Distance = pow(sin((y2-y1) / 2.0), 2.0) + cos(y1) * cos(y2) * pow(sin((x2-x1) / 2.0), 2.0);
            Distance = (2 * asin(min(1, pow(Distance, .5)))) * Radius;
      }

      return Distance;
}
```

© PNR and Associates, All rights reserved

HATFIELD 5.0: DEFAULT vs. FCC UNIFIED INPUTS

STATE	COMPANY	DEFAULT NID Per Line	Distribution Per Line	Concentrator Per Line	Feeder Per Line	Total Loop Per Line	FCC Inputs NID Per Line	Distribution Per Line
Alaska	Anchorage Tel Util	$20.07	$169.82	$92.86	$125.97	$408.71	$20.52	$338.27
Alabama	Contel Of The South Dba Gte South	$25.07	$949.08	$256.52	$557.53	$1,788.21	$25.84	$1,275.72
Alabama	Gte And Contel Of Alabama	$24.16	$712.57	$201.66	$362.27	$1,300.66	$24.86	$930.27
Alabama	South Central Bell-Al	$22.74	$436.24	$141.95	$173.59	$774.53	$23.30	$596.91
Arkansas	Southwestern Bell-Arkansas	$22.55	$430.90	$137.22	$187.01	$777.69	$23.13	$596.48
Arizona	Mountain Bell-Arizona	$22.15	$249.80	$126.93	$95.44	$494.32	$22.67	$408.66
California	Contel Of California - California	$22.74	$559.00	$197.59	$294.69	$1,074.01	$23.37	$792.79
California	Gte Of California	$21.35	$199.30	$90.19	$86.80	$397.64	$21.89	$369.60
California	Pacific Bell	$19.58	$177.37	$94.40	$85.02	$376.38	$20.02	$348.49
California	Roseville Telephone Company	$22.61	$189.65	$108.25	$73.86	$394.36	$23.25	$371.49
Colorado	Mountain Bell-Colorado	$20.96	$264.50	$115.71	$114.42	$515.59	$21.46	$442.36
Connecticut	Southern New England Tel	$22.59	$295.24	$113.67	$79.88	$511.38	$23.19	$446.16
District of Columbia	C And P Telephone Company Of Wa Dc	$12.93	$54.36	$40.21	$65.31	$172.80	$13.12	$130.14
Delaware	Diamond State Tel Co	$21.56	$287.45	$121.61	$97.13	$527.75	$22.11	$438.01
Florida	Central Tel Co Of Florida	$21.32	$362.20	$124.64	$159.16	$667.32	$21.82	$517.49
Florida	Gte Floridainc	$22.95	$250.44	$117.25	$75.80	$466.44	$23.57	$388.95
Florida	Southern Bell-Fl	$22.59	$236.14	$120.99	$85.99	$465.70	$23.19	$386.43
Florida	United Tel Co Of Florida	$23.62	$360.55	$139.33	$122.04	$645.55	$24.29	$503.23
Georgia	Alltel Georgia Communications Corp	$23.19	$663.15	$188.69	$296.10	$1,171.14	$23.83	$867.19
Georgia	Southern Bell-Ga	$21.41	$327.63	$136.45	$109.24	$594.73	$21.94	$464.09
Hawaii	Gte Hawaiian Telephone Co Inc	$21.64	$245.12	$75.87	$116.23	$458.85	$22.20	$411.60
Iowa	Gte And Contel Of Iowa	$24.09	$878.58	$257.68	$607.65	$1,768.00	$24.81	$1,194.20
Iowa	Northwestern Bell-Ia	$21.56	$338.44	$109.70	$140.56	$610.25	$22.09	$495.59
Idaho	Gte Northwest Inc - Idaho	$23.63	$622.54	$198.95	$406.49	$1,251.60	$24.29	$871.66
Idaho	Mountain Bell-Idaho	$22.57	$405.24	$159.63	$182.39	$769.83	$23.15	$603.07
Illinois	Central Tel Co Of Il	$19.75	$222.57	$107.89	$98.13	$448.34	$20.15	$396.34
Illinois	Contel Of Illinois Inc Dba Gte - Illinois	$23.82	$789.60	$219.51	$436.11	$1,469.04	$24.52	$1,061.34
Illinois	Gte Of Illinois	$23.41	$597.10	$164.68	$284.24	$1,069.43	$24.07	$819.43
Illinois	Illinois Bell Tel Co	$19.92	$186.54	$93.17	$79.35	$378.98	$20.38	$346.25
Indiana	Contel Of Indiana Inc Dba Gte - Indiana	$23.38	$738.14	$186.96	$339.71	$1,288.19	$24.05	$983.58
Indiana	Gte Of Indiana	$22.71	$433.46	$129.99	$139.00	$725.16	$23.33	$598.75
Indiana	Indiana Bell Tel Co	$20.79	$281.59	$119.85	$89.52	$511.76	$21.28	$425.33
Indiana	United Tel Co Of Indiana Inc	$23.39	$681.03	$171.37	$260.67	$1,136.47	$24.06	$902.84
Kansas	Southwestern Bell-Kansas	$21.69	$311.69	$129.30	$159.08	$621.76	$22.22	$458.19
Kentucky	Cincinnati Bell-Ky	$23.97	$390.28	$142.16	$118.15	$674.57	$24.65	$551.81
Kentucky	Gte South Inc - Kentucky	$22.93	$485.86	$146.66	$210.00	$865.44	$23.54	$676.77
Kentucky	South Central Bell-Ky	$23.82	$482.26	$148.51	$200.81	$855.39	$24.49	$674.73
Louisiana	South Central Bell-La	$22.83	$376.90	$133.09	$159.56	$692.38	$23.46	$546.29
Massachusetts	New England Tel-Ma	$21.48	$249.52	$93.34	$81.66	$446.00	$22.03	$421.49
Maryland	C And P Tel Co Of Md	$21.48	$263.46	$104.77	$89.91	$479.62	$22.03	$422.80
Maine	New England Tel-Maine	$23.32	$526.11	$141.59	$206.04	$897.06	$23.96	$726.92
Michigan	Gte North Inc-Mi	$24.31	$685.36	$176.32	$245.54	$1,131.53	$25.02	$898.11

HATFIELD 5.0: DEFAULT vs. FCC UNIFIED INPUTS

STATE	COMPANY	DEFAULT NID Per Line	Distribution Per Line	Concentrator Per Line	Feeder Per Line	Total Loop Per Line	FCC Inputs NID Per Line	Distribution Per Line
Michigan	Michigan Bell Tel Co	$21.71	$282.25	$120.91	$91.96	$516.83	$22.24	$437.27
Minnesota	Contel Of Minnesota Inc Dba Gte Minnesota	$24.78	$1,026.00	$318.95	$735.98	$2,105.71	$25.53	$1,388.81
Minnesota	Frontier Comm Of Minnesota Inc	$24.46	$572.77	$188.20	$311.05	$1,096.47	$25.18	$785.33
Minnesota	Northwestern Bell-Minnesota	$20.60	$272.70	$108.56	$125.66	$527.53	$21.08	$427.95
Minnesota	United Telephone Co Of Minn	$24.14	$598.55	$173.88	$283.78	$1,080.34	$24.84	$805.01
Missouri	Contel Of Missouri	$25.09	$866.08	$243.22	$564.55	$1,698.93	$25.82	$1,174.18
Missouri	Gte North Inc - Missouri	$24.48	$605.39	$169.56	$309.99	$1,109.42	$25.17	$819.57
Missouri	Southwestern Bell-Missouri	$22.24	$290.32	$114.24	$120.38	$547.18	$22.79	$456.53
Missouri	United Telephone Co Of Missouri	$23.36	$599.82	$170.96	$308.02	$1,102.16	$23.97	$847.37
Mississippi	South Central Bell-Mississippi	$23.86	$592.28	$174.01	$330.86	$1,121.01	$24.52	$812.39
Montana	Mountain Bell-Montana	$22.70	$468.27	$176.95	$288.97	$956.89	$23.29	$710.56
North Carolina	Alltel Carolina Inc-North	$24.26	$590.42	$152.62	$160.61	$927.91	$24.96	$756.68
North Carolina	Carolina Tel And Tel Co	$24.32	$583.03	$161.81	$235.03	$1,004.19	$25.02	$771.93
North Carolina	Central Tel Co-Nc	$24.10	$599.20	$162.24	$180.85	$966.39	$24.80	$768.03
North Carolina	Contel Of North Carolina Dba Gte No Carolina	$23.63	$760.26	$185.90	$316.06	$1,285.85	$24.29	$977.75
North Carolina	Gte South Inc - North Carolina	$20.88	$310.96	$113.64	$88.51	$533.99	$21.35	$435.35
North Carolina	North State Tel Co-Nc	$23.36	$383.84	$127.49	$82.22	$616.91	$23.99	$504.22
North Carolina	Southern Bell-Nc	$21.68	$351.86	$130.81	$94.62	$598.97	$22.22	$484.60
North Dakota	Northwestern Bell-North Dakota	$22.37	$344.73	$141.96	$290.69	$799.75	$22.94	$545.57
Nebraska	Lincoln Tel And Tele Co	$23.23	$469.57	$159.24	$312.98	$965.03	$23.86	$695.50
Nebraska	Northwestern Bell-Nebraska	$19.89	$268.82	$110.49	$155.95	$555.16	$20.30	$453.96
New Hampshire	New England Tel-Nh	$22.94	$436.61	$132.35	$146.15	$738.06	$23.54	$614.55
New Jersey	New Jersey Bell	$20.65	$198.41	$92.92	$80.63	$392.61	$21.15	$372.23
New Jersey	United Tel Co Of Nj Inc	$21.63	$493.28	$145.48	$99.75	$760.14	$22.20	$652.80
New Mexico	Mountain Bell-New Mexico	$22.24	$351.64	$139.05	$170.63	$683.56	$22.83	$536.06
Nevada	Central Telephone Company - Nevada	$21.89	$179.65	$111.47	$85.05	$398.06	$22.45	$340.95
Nevada	Nevada Bell	$21.49	$337.12	$143.98	$217.72	$720.31	$22.00	$590.22
New York	Citizens Telecom-Ny	$24.52	$744.44	$192.54	$331.23	$1,292.74	$25.26	$999.55
New York	New York Tel	$18.86	$179.96	$73.26	$92.02	$364.10	$19.32	$332.75
New York	Rochester Telephone Corp	$22.59	$297.87	$102.71	$101.02	$524.19	$23.18	$463.82
Ohio	Alltel Ohio Inc	$23.40	$433.49	$114.06	$131.80	$702.75	$24.03	$608.12
Ohio	Cincinnati Bell-Ohio	$22.66	$253.32	$102.35	$75.28	$453.62	$23.27	$400.94
Ohio	Gte North Inc-Oh	$24.21	$623.11	$156.76	$215.79	$1,019.87	$24.91	$825.94
Ohio	Ohio Bell Tel Co	$21.26	$245.21	$99.45	$77.62	$443.54	$21.77	$397.37
Ohio	United Tel Co Of Ohio	$23.12	$554.75	$146.66	$173.43	$897.96	$23.74	$735.74
Ohio	Western Reserve Tel Co	$22.86	$489.78	$147.12	$123.40	$783.16	$23.43	$639.38
Oklahoma	Gte Southwest Inc - Oklahoma	$24.22	$560.60	$166.91	$275.41	$1,027.14	$24.92	$771.33
Oklahoma	Southwestern Bell-Oklahoma	$22.75	$386.49	$132.26	$168.98	$710.49	$23.34	$561.29
Oregon	Gte Of The Northwest	$22.58	$359.80	$124.85	$143.61	$650.84	$23.16	$523.02
Oregon	Pacific Northwest Bell-Oregon	$22.14	$296.88	$119.51	$119.88	$558.40	$22.71	$464.76
Pennsylvania	Alltel Pennsylvaniainc	$23.49	$635.53	$166.59	$247.10	$1,072.72	$24.14	$880.46
Pennsylvania	Bell Of Pennsylvania	$22.50	$257.07	$95.86	$93.16	$468.59	$22.92	$435.86

HATFIELD 5.0: DEFAULT vs. FCC UNIFIED INPUTS

STATE	COMPANY	DEFAULT NID Per Line	Distribution Per Line	Concentrator Per Line	Feeder Per Line	Total Loop Per Line	FCC Inputs NID Per Line	Distribution Per Line
Pennsylvania	C-Tec Corp	$24.56	$700.49	$172.25	$237.46	$1,134.76	$25.28	$933.46
Pennsylvania	Gte North Inc-Pa And Contel	$23.96	$461.14	$129.44	$143.59	$758.13	$24.64	$668.59
Pennsylvania	United Tel Co Of Pa	$24.68	$616.20	$155.20	$202.51	$998.58	$25.40	$847.74
Puerto Rico	P R T C - Central	$24.79	$353.69	$117.57	$99.84	$595.89	$25.51	$726.96
Puerto Rico	Puerto Rico Tel Co	$24.57	$314.92	$128.07	$112.21	$579.78	$25.28	$746.62
Rhode Island	New England Tel-Ri	$22.86	$261.77	$107.96	$77.48	$470.07	$23.45	$431.44
South Carolina	Gte South Inc - South Carolina	$23.47	$480.06	$131.17	$186.96	$821.66	$24.11	$642.99
South Carolina	Southern Bell-Sc	$22.99	$421.89	$140.72	$130.66	$716.25	$23.61	$573.76
South Dakota	Northwestern Bell-South Dakota	$21.38	$394.57	$149.99	$276.34	$842.28	$21.89	$649.53
Tennessee	South Central Bell-Tn	$22.88	$400.25	$137.92	$145.40	$706.45	$23.49	$565.83
Tennessee	United Inter-Mountain Tel Co-Tn	$23.47	$503.54	$144.54	$131.68	$803.23	$24.12	$657.32
Texas	Central Telephone Company Of Texas	$21.92	$386.74	$127.30	$184.42	$720.38	$22.44	$548.11
Texas	Contel Of Texas Inc Dba Gte Texas	$23.22	$947.25	$287.90	$661.17	$1,919.54	$23.88	$1,282.47
Texas	Gte Southwest Inc - Texas	$22.09	$399.10	$141.93	$189.17	$752.28	$22.66	$594.15
Texas	Southwestern Bell-Texas	$21.03	$260.15	$126.12	$107.75	$515.04	$21.53	$414.96
Texas	United Telephone Co Of Texas Inc	$23.69	$780.83	$212.10	$480.91	$1,497.52	$24.38	$1,048.41
Utah	Mountain Bell-Utah	$21.66	$263.06	$111.97	$90.75	$487.44	$22.18	$418.34
Virginia	C And P Tel Co Of Va	$21.47	$273.92	$118.77	$107.23	$521.39	$21.99	$423.59
Virginia	Central Tel Co Of Va	$23.77	$672.17	$165.73	$308.88	$1,170.56	$24.45	$907.05
Virginia	Contel Of Virginia Inc Dba Gte Virginia	$23.52	$527.63	$179.88	$212.71	$943.73	$24.17	$720.68
Virginia	United Inter-Mountain Tel Co-Va	$25.58	$577.91	$148.71	$279.05	$1,031.25	$26.08	$777.88
Vermont	New England Tel-Vt	$22.74	$547.40	$145.06	$237.74	$952.94	$23.34	$786.46
Washington	Gte Northwest Inc - Washington	$23.04	$343.71	$125.46	$129.70	$621.91	$23.67	$507.00
Washington	Pacific Northwest Bell-Washington	$22.33	$265.44	$109.78	$98.64	$496.19	$22.92	$424.68
Washington	Telephone Utilities Of Wa Inc	$24.47	$801.42	$270.67	$533.26	$1,629.82	$25.20	$1,072.29
Wisconsin	Gte North Inc-Wi	$24.76	$764.71	$199.53	$368.00	$1,357.00	$25.49	$999.28
Wisconsin	Wisconsin Bell	$21.43	$254.52	$104.50	$87.14	$467.59	$21.95	$402.88
West Virginia	C And P Tel Co Of W Va	$24.00	$552.46	$147.88	$245.72	$970.06	$24.67	$775.56
Wyoming	Mountain Bell-Wyoming	$22.64	$529.14	$218.56	$396.83	$1,167.17	$23.22	$806.43

HATFIELD 5.0: DEFAULT vs. FCC UNIFIED INPUTS

STATE	COMPANY	Concentrator Per Line	Feeder Per Line	Total Loop Per Line	Change NID Per Line	Distribution Per Line	Concentrator Per Line	Feeder Per Line
Alaska	Anchorage Tel Util	$95.57	$192.50	$646.86	$0.45	$168.45	$2.70	$66.54
Alabama	Contel Of The South Dba Gte South	$258.49	$652.96	$2,213.01	$0.76	$326.64	$1.97	$95.42
Alabama	Gte And Contel Of Alabama	$203.83	$422.58	$1,581.55	$0.70	$217.70	$2.17	$60.31
Alabama	South Central Bell-Al	$144.49	$214.06	$978.75	$0.56	$160.66	$2.53	$40.46
Arkansas	Southwestern Bell-Arkansas	$140.20	$215.51	$975.32	$0.57	$165.58	$2.97	$28.50
Arizona	Mountain Bell-Arizona	$128.87	$126.46	$686.66	$0.51	$158.86	$1.94	$31.01
California	Contel Of California - California	$201.50	$351.11	$1,368.76	$0.63	$233.80	$3.91	$56.42
California	Gte Of California	$93.12	$105.18	$589.79	$0.54	$170.30	$2.93	$18.38
California	Pacific Bell	$97.16	$114.09	$579.76	$0.43	$171.12	$2.75	$29.07
California	Roseville Telephone Company	$111.25	$105.26	$611.25	$0.64	$181.84	$3.00	$31.40
Colorado	Mountain Bell-Colorado	$118.70	$146.49	$729.01	$0.50	$177.86	$2.99	$32.07
Connecticut	Southern New England Tel	$116.88	$96.95	$683.18	$0.60	$150.92	$3.20	$17.07
District of Columbia	C And P Telephone Company Of Wa Dc	$37.15	$84.76	$265.17	$0.19	$75.79	-$3.06	$19.45
Delaware	Diamond State Tel Co	$124.98	$124.68	$709.78	$0.55	$150.56	$3.37	$27.55
Florida	Central Tel Co Of Florida	$127.84	$185.72	$852.87	$0.50	$155.29	$3.20	$26.56
Florida	Gte Floridainc	$120.28	$94.99	$627.79	$0.62	$138.51	$3.03	$19.19
Florida	Southern Bell-Fl	$124.05	$112.82	$646.49	$0.61	$150.30	$3.06	$26.83
Florida	United Tel Co Of Florida	$142.89	$145.76	$816.17	$0.66	$142.67	$3.56	$23.73
Georgia	Alltel Georgia Communications Corp	$191.47	$345.67	$1,428.15	$0.63	$204.04	$2.78	$49.57
Georgia	Southern Bell-Ga	$139.59	$133.43	$759.04	$0.53	$136.47	$3.13	$24.18
Hawaii	Gte Hawaiian Telephone Co Inc	$79.09	$135.85	$648.73	$0.56	$166.49	$3.21	$19.62
Iowa	Gte And Contel Of Iowa	$254.23	$689.27	$2,162.51	$0.72	$315.63	-$3.45	$81.62
Iowa	Northwestern Bell-Ia	$112.36	$159.02	$789.05	$0.54	$157.15	$2.66	$18.46
Idaho	Gte Northwest Inc - Idaho	$201.45	$464.71	$1,562.11	$0.65	$249.12	$2.51	$58.22
Idaho	Mountain Bell-Idaho	$163.47	$219.81	$1,009.50	$0.58	$197.83	$3.84	$37.42
Illinois	Central Tel Co Of Il	$110.17	$135.67	$662.33	$0.39	$173.78	$2.28	$37.54
Illinois	Contel Of Illinois Inc Dba Gte - Illinois	$221.97	$492.02	$1,799.85	$0.69	$271.74	$2.46	$55.91
Illinois	Gte Of Illinois	$167.35	$320.81	$1,331.66	$0.66	$222.33	$2.68	$36.57
Illinois	Illinois Bell Tel Co	$95.85	$105.74	$568.22	$0.46	$159.71	$2.68	$26.39
Indiana	Contel Of Indiana Inc Dba Gte - Indiana	$192.21	$393.65	$1,593.48	$0.67	$245.44	$5.24	$53.93
Indiana	Gte Of Indiana	$134.34	$155.78	$912.21	$0.62	$165.29	$4.35	$16.79
Indiana	Indiana Bell Tel Co	$122.98	$108.02	$677.61	$0.49	$143.74	$3.12	$18.49
Indiana	United Tel Co Of Indiana Inc	$175.85	$295.03	$1,397.78	$0.67	$221.81	$4.48	$34.35
Kansas	Southwestern Bell-Kansas	$131.86	$194.55	$806.81	$0.52	$146.50	$2.55	$35.47
Kentucky	Cincinnati Bell-Ky	$146.67	$146.62	$869.76	$0.68	$161.53	$4.51	$28.48
Kentucky	Gte South Inc - Kentucky	$150.56	$243.18	$1,094.06	$0.62	$190.92	$3.91	$33.18
Kentucky	South Central Bell-Ky	$152.10	$234.38	$1,085.69	$0.67	$192.47	$3.59	$33.57
Louisiana	South Central Bell-La	$136.41	$189.98	$896.13	$0.63	$169.39	$3.31	$30.42
Massachusetts	New England Tel-Ma	$96.45	$99.42	$639.39	$0.55	$171.98	$3.11	$17.76
Maryland	C And P Tel Co Of Md	$107.94	$109.90	$662.67	$0.54	$159.35	$3.17	$19.99
Maine	New England Tel-Maine	$146.07	$242.83	$1,139.78	$0.64	$200.81	$4.48	$36.79
Michigan	Gte North Inc-Mi	$182.89	$278.55	$1,384.58	$0.71	$212.75	$6.57	$33.01