```
Attribute VB_Name = "GCircle"
Option Explicit
Rem
Rem Author R. J. Spriggs August 2012
Rem Programmers: The Visual Basic source code in this document may be copied
Rem and reused any for non commercial use.
Rem

Rem Utility Functions and Subroutines.
Rem ================================
Rem
Rem         Functions and Subroutines located in this Module.
Rem
Rem     Type        Name                    Comment
Rem
Rem     Function    DMS2V                   Converts Degrees/Mins/Secs to Single Value
Rem     Function    DMSformat               Will Text Format Degrees,Minutes,Seconds
Rem     Function    GCDist                  Calculate Greater Circle Distance
Rem     Function    GCDistLat               Calculate Greater Circle Latitude
Rem     Function    GCDistLong              Calculate Greater Circle Longitude
Rem     Subroutine  V2DMS                   Converts Single Value to Degrees/Mins/Secs


Rem ================================================================
Rem Greater Circle Conversion Section
Rem ================================================================

Rem Definitions of Common Greater Circle information stores

'Public Const GCMeanDist = 6371.01           'Mean Distance/Rad of Earth approx 0.55%
accuracy
'Public Const GCMeanDist = 6378.137          'Mean Distance/Rad of Earth Default (Google
Maps)
Public Const GCMeanDist = 6377.563396       'Airy 1830 major & minor semi-axes
'Public Const GCKmDeg = 111.1                'Mean km per degree Distance of Earth
Public Const GCKmDeg = GCMeanDist / 180 * Pi 'Mean km per degree Distance of Earth

Rem Functions and Subroutines for Greater Circle Calculations

Public Function DMS2V(D As Integer, M As Integer, S As Integer) As Double
Rem This routine will convert Degrees,Minutes,Seconds to Single Value
Rem Initial Design 07/Aug/2012 Author R. J. Spriggs
Rem Mod 08/08/12 RJS use integer parameters to avoid decimal fraction units
Rem Mod 08/08/12 RJS Correct for Negative Angles
Dim IntVal As Double                            'Intermediate Value Store
Dim SumVal As Double                            'Summation Value Store
Dim NegPos As Boolean                           'Negative Angle Given

    NegPos = False                              'Assume Positive Angle
    If D < 0 Then NegPos = True                 'Angle is actually Negative
    SumVal = Abs(D)                             'Hold Degrees (Sign removed)
    IntVal = Abs(M)                             'No Negative Minutes
    SumVal = SumVal + IntVal / 60               'Add in Minutes
    IntVal = Abs(S)                             'No Negative Seconds
    SumVal = SumVal + IntVal / 3600             'Add in Seconds
    'DMS2V = D + M / 60 + S / 3600               'Convert Value
    If NegPos = True Then SumVal = -SumVal      'Correct Angle sign
    DMS2V = SumVal                              'Hold Converted Value
End Function

Public Function DMSformat(D As Integer, M As Integer, S As Integer) As String
Rem This routine will Text Format Degrees,Minutes,Seconds
Rem Initial Design 10/Aug/2012 Author R. J. Spriggs
Dim A$, B$                                      'General String Stores

    A$ = Str$(Abs(D))                           'Convert Degrees to String
    If Len(A$) < 3 Then A$ = " 0" + Mid$(A$, 2) 'Pad to 3 Chars +
    B$ = A$                                     'Build Reply
    A$ = Str$(M)                                'Convert Minutes to String
    If Len(A$) < 3 Then A$ = " 0" + Mid$(A$, 2) 'Pad to 3 Chars +
```

```
    B$ = B$ + A$                                'Build Reply
    A$ = Str$(S)                                'Convert Seconds to String
    If Len(A$) < 3 Then A$ = " 0" + Mid$(A$, 2) 'Pad to 3 Chars +
    DMSformat = B$ + A$                         'Build Reply
End Function

Public Function GCDist(LaR As Double, LoR As Double, LaD As Double, LoD As Double)
Rem This routine will calculate the Greater Circle Distance between two sets
Rem of Latitude/Longitude points.
Rem Latitude  N/S in degrees (Positive=North Negative = South)
Rem Longitude E/W in degrees (Positive=West  Negative = East)
Rem Constant Used Mean Radius per Radian = 6371.01 km or
Rem                           per degree = 111.1949266 km circa 0.5% accuracy
Dim AbsVal As Double                            'Abs Calculation store
Dim CosVal As Double                            'Cos Calculation store
Dim SinVal As Double                            'Sin Calculation store

    AbsVal = Abs(LoR - LoD)                     'Hold Diference between Longitudes
    CosVal = (Cos(D2R(LaR)) * Cos(D2R(LaD)))    'Hold part Cosine Component
    CosVal = CosVal * Cos(D2R(AbsVal))          'Hold Cosine Component
    SinVal = (Sin(D2R(LaR)) * Sin(D2R(LaD)))    'Hold Sine Component
    GCDist = ACos(SinVal + CosVal) * GCMeanDist 'Calculate distance
End Function

Public Function GCDistLong(LaR As Double, LoR As Double, LaD As Double, Dist As Double)
Rem This routine will calculate the Longitude of a point to left or right
Rem a specified Longitude in degrees given a distance in kms
Dim CosVal As Double                            'Cos Calculation store
Dim IntVal As Double                            'Intermediate Calculation store
Dim SinVal As Double                            'Sin Calculation store
    CosVal = (Cos(D2R(LaR)) * Cos(D2R(LaD)))    'Hold part Cosine Component
    SinVal = (Sin(D2R(LaR)) * Sin(D2R(LaD)))    'Hold Sine Component
    SinVal = Cos(Dist / GCKmDeg) - SinVal       'Hold Cos-Sine Component
    IntVal = ACos(SinVal / CosVal)              'Calc angle diference
    If Dist < 0 Then
        IntVal = LoR - IntVal                   'Calc New Longitude
    Else
        IntVal = LoR + IntVal                   'Calc New Longitude
    End If
    GCDistLong = IntVal                         'Reply with converted value
End Function

Public Function GCDistLat(LaR As Double, Dist As Double)
Rem This routine will calculate the Latitude of a point above or below Equator
Rem a specified Latitude in degrees given a distance in kms
Dim Ang As Double                               'Angle Store
    Ang = (Dist / GCMeanDist) * (180 / Pi)      'Calculate Angular offset
    Ang = Ang + LaR                             'Calculate Latitude
    If Ang > 180 Then Ang = Ang - 360           'Correct excessive Latitude
    If Ang < -180 Then Ang = Ang + 360          'Correct excessive Latitude
    GCDistLat = Ang                             'Give Conversion
End Function

Public Sub V2DMS(V As Double, D As Integer, M As Integer, S As Integer)
Rem This routine will convert Single Value to Degrees,Minutes,Seconds
Rem Initial Design 08/Aug/2012 Author R. J. Spriggs
Rem Mod 08/08/12 RJS Correct for Negative Angles
Dim IntVal As Double                            'Intermediate Value Store
Dim NegPos As Boolean                           'Negative Angle Given

    IntVal = Abs(V)                             'Hold Positive Form of Angle
    D = Int(IntVal)                             'Hold Degrees
    If V < 0 Then D = -D                        'Angle is actually Negative
    IntVal = (Abs(V) - Abs(D)) * 60             'No Negative Minutes or Seconds
    M = Int(IntVal)                             'Hold Minutes
    IntVal = (IntVal - M) * 60                  'Keep remainder
    S = Int(IntVal)                             'Hold Seconds
End Sub
```