```
Attribute VB_Name = "GPS_UTM"
Option Explicit
Rem
Rem With thanks to
Rem Copyright 1997-1998 by Charles L. Taylor (Chuck Taylor) author of the javascript
Rem examples with reference to
Rem <P>Programmers: The JavaScript source code in this document may be copied
Rem and reused without restriction.</P>
Rem
Rem With thanks to
Rem Steven Dutch Natural and Applied Science (University of Wisconsin Green Bay)
Rem Converting UTM to Latitude and Longitude (Or Vice Versa).
Rem Constants Information located in document.

Rem
Rem Converted to VB6 by R. J. Spriggs June 2013
Rem Programmers: The Visual Basic source code in this document may be copied
Rem and reused any for non commercial use.
Rem

Rem
Rem         Functions and Subroutines located in this Module.
Rem
Rem     Type        Name                    Comment
Rem

Rem     Subroutine  CvtLL2UTMCode           Convert Latitude and Longitude (in degrees)
Rem                                         to UTM Grid Code and Central Meridian
Rem     Subroutine  CvtLL2UTM               Convert Latitude and Longitude (in degrees)
Rem                                         to UTM Numeric Grid
Rem     Subroutine  CvtLL2UTM4DP            Convert Latitude and Longitude (in degrees)
Rem                                         to UTM Numeric Grid to nearest metre
Rem     Function    ArcLengthOfMeridian     Computes the ellipsoidal distance from the
Rem                                         equator to a point at a given latitude.
Rem     Subroutine  MapLatLonToXY           Converts a latitude/longitude pair to x,y
Rem                                         coordinates in the Transverse Mercator
Rem                                         projection.
Rem                                         Note that Transverse Mercator is not
Rem                                         the same as UTM a scale factor is required
Rem                                         to convert between them.
Rem     Subroutine  CvtUTM2LL               Convert UTM Numeric Grid to
Rem                                         Latitude and Longitude (in degrees)
Rem     Function    CvtUTMFull2Short        Convert a Full UTM Zone code to a short
Rem                                         version that only contains North and
Rem                                         Southern Hemisphere codes.
Rem     Function    UTMCodeTidy             Will Tidy a UTM code.
Rem     Function    FootpointLatitude       Computes the footpoint latitude for use in
Rem                                         converting transverse Mercator coordinates
Rem                                         to ellipsoidal coordinates.
Rem     Subroutine  MapXYToLatLon           Converts x and y coordinates in the
Rem                                         Transverse Mercator projection to
Rem                                         a latitude/longitude pair.
Rem
Rem ============================    Warning ======================================
Rem                                         The following routine must be called before
Rem                                         any other routine in this set is used.
Rem     Subroutine  SetMajorMinorAxis       Initialise Major/Minor UTM Axis.
Rem ==============================================================================


Rem ==================================================================
Rem Universal Transverse Mercator UTM Conversion Section
Rem ==================================================================

Rem Definitions of Common UTM information stores
Global sm_a As Double                       'Ellipsoid model major axis.
Global sm_b As Double                       'Ellipsoid model minor axis.
'Global sm_EccSquared As Double              'Eccentricity squared (Never used)
Global UTMScaleFactor As Double             'Scale along central meridian
```

```
Rem Functions and Subroutines for UTM Conversions

Rem --------------------   Latitude/Longitude to UTM set   --------------------

Public Sub CvtLL2UTMCode(UTM As String, CMed As Double, Lat As Double, Lng As Double, 
Status As String)
Rem Convert Latitude and Longitude (in degrees) to UTM Grid Code and Central Meridian
Rem Initial Design 15/May/2013 Author R. J. Spriggs

Dim Zone As Integer
Dim Cnt As Integer
Dim Pnt As Integer
Dim Codes As String
Dim Bounds(24) As Integer


    Rem Calculate Code Number
    Codes = "ABCDEFGHJKLMNPQRSTUVWXYZ"
    Pnt = 1
    Bounds(Pnt) = -90: Pnt = Pnt + 1
    Bounds(Pnt) = -80: Pnt = Pnt + 1
    For Pnt = 3 To 21
        Bounds(Pnt) = Bounds(Pnt - 1) + 8
    Next Pnt
    Bounds(Pnt) = 84: Pnt = Pnt + 1
    Bounds(Pnt) = 87: Pnt = Pnt + 1
    Bounds(Pnt) = 90: Pnt = Pnt + 1

    Cnt = -1: For Pnt = 1 To 24
        If Lat < Bounds(Pnt) And Cnt < 0 Then
            Cnt = Pnt
        End If
    Next Pnt
    If Cnt > 0 Then UTM = " " + Mid$(Codes, Cnt, 1)

    Rem Calculate Zone Number
    Zone = Int((Lng + 180) / 6) + 1
    UTM = V2S$(Zone) + UTM
    CMed = Int(Lng / 6) * 6 + 3                 'Calculate Central Meridian of Zone
End Sub

Sub CvtLL2UTM(UTM As String, CMed As Double, Lat As Double, Lng As Double, Status As 
String)
'Rem Convert Latitude and Longitude (in degrees) to UTM Numeric Grid
Dim Phi As Double
Dim lambda As Double
Dim lambda0 As Double
Dim easting As Double
Dim northing As Double

    Phi = Lat * Pi / 180
    lambda = Lng * Pi / 180
    lambda0 = CMed * Pi / 180
    MapLatLonToXY Phi, lambda, lambda0, easting, northing

    Rem Adjust easting and northing for UTM system.
    easting = easting * UTMScaleFactor + 500000
    northing = northing * UTMScaleFactor
    If northing < 0 Then northing = northing + 10000000

    UTM = Str$(easting / 1000) + "kmE  " + Str$(northing / 1000) + "kmN"

End Sub


Sub CvtLL2UTM4DP(UTM As String, CMed As Double, Lat As Double, Lng As Double, Status As 
String)
Rem Convert Latitude and Longitude (in degrees) to UTM Numeric Grid (to 4 Decimal Places)
Dim Phi As Double
Dim lambda As Double
```

```
Dim lambda0 As Double
Dim easting As Double
Dim northing As Double

    Phi = Lat * Pi / 180
    lambda = Lng * Pi / 180
    lambda0 = CMed * Pi / 180
    MapLatLonToXY Phi, lambda, lambda0, easting, northing

    Rem Adjust easting and northing for UTM system.
    easting = easting * UTMScaleFactor + 500000
    northing = northing * UTMScaleFactor
    If northing < 0 Then northing = northing + 10000000

    UTM = Str$(Int(easting + 0.5) / 1000) + "kmE  " + Str$(Int(northing + 0.5) / 1000) +
"kmN"

End Sub


Function ArcLengthOfMeridian(Phi As Double)
Rem
Rem Computes the ellipsoidal distance from the equator to a point at a
Rem given latitude.
Rem
Rem Reference: Hoffmann-Wellenhof, B., Lichtenegger, H., and Collins, J.,
Rem GPS: Theory and Practice, 3rd ed.  New York: Springer-Verlag Wien, 1994.
Rem
Rem Inputs:
Rem     phi - Latitude of the point, in radians.
Rem
Rem Globals:
Rem     sm_a - Ellipsoid model major axis.
Rem     sm_b - Ellipsoid model minor axis.
Rem
Rem Returns:
Rem     The ellipsoidal distance of the point from the equator, in meters.

Dim n As Double                                 'Work Variable
Dim alpha As Double                             'Work Variable
Dim beta As Double                              'Work Variable
Dim gamma As Double                             'Work Variable
Dim delta As Double                             'Work Variable
Dim epsilon As Double                           'Work Variable
Dim Tmp1 As Double, Tmp2 As Double, Tmp3 As Double, Tmp4 As Double

    n = (sm_a - sm_b) / (sm_a + sm_b)           'Precalculate n
                                                'Precalculate alpha
    alpha = ((sm_a + sm_b) / 2) * (1 + ((n ^ 2) / 4) + ((n ^ 4) / 64))
                                                'Precalculate beta
    beta = (-3 * n / 2) + (9 * (n ^ 3) / 16) + (-3 * (n ^ 5) / 32)
                                                'Precalculate gamma
    gamma = (15 * (n ^ 2) / 16) + (-15 * (n ^ 4) / 32)
                                                'Precalculate delta
    delta = (-35 * (n ^ 3) / 48) + (105 * (n ^ 5) / 256)
                                                'Precalculate epsilon
    epsilon = (315 * (n ^ 4) / 512)

    Rem Now calculate the sum of the series and return
    Tmp1 = (beta * Sin(2 * Phi))                'Interim result
    Tmp2 = (gamma * Sin(4 * Phi))          'Interim result
    Tmp3 = (delta * Sin(6 * Phi))          'Interim result
    Tmp4 = (epsilon * Sin(8 * Phi))        'Interim result

    ArcLengthOfMeridian = alpha * (Phi + Tmp1 + Tmp2 + Tmp3 + Tmp4)

End Function

Sub MapLatLonToXY(Phi As Double, lambda As Double, lambda0 As Double, x As Double, y As
Double)
```

```
Rem
Rem Converts a latitude/longitude pair to x and y coordinates in the
Rem Transverse Mercator projection.  Note that Transverse Mercator is not
Rem the same as UTM; a scale factor is required to convert between them.
Rem
Rem Reference: Hoffmann-Wellenhof, B., Lichtenegger, H., and Collins, J.,
Rem GPS: Theory and Practice, 3rd ed.  New York: Springer-Verlag Wien, 1994.
Rem
Rem Inputs:
Rem    phi - Latitude of the point, in radians.
Rem    lambda - Longitude of the point, in radians.
Rem    lambda0 - Longitude of the central meridian to be used, in radians.
Rem
Rem Outputs:
Rem    x,y - Contains the x=Easting and y=Northing coordinates
Rem         of the computed point.
Rem
Dim n As Double, nu2 As Double, ep2 As Double, t As Double, t2 As Double, l As Double
Dim l3coef As Double, l4coef As Double, l5coef As Double, l6coef As Double
Dim l7coef As Double, l8coef As Double
Dim Tmp1 As Double, Tmp2 As Double, Tmp3 As Double, Tmp4 As Double

    'ep2 = (Math.pow (sm_a, 2.0) - Math.pow (sm_b, 2.0)) / Math.pow (sm_b, 2.0)
    ep2 = ((sm_a ^ 2) - (sm_b ^ 2)) / (sm_b ^ 2)    'Precalculate ep2

    'nu2 = ep2 * Math.pow (Math.cos (phi), 2.0);
    nu2 = ep2 * Cos(Phi) ^ 2                        'Precalculate nu2

    'N = Math.pow (sm_a, 2.0) / (sm_b * Math.sqrt (1 + nu2));
    n = (sm_a ^ 2) / (sm_b * Sqr(1 + nu2))          'Precalculate N

    't = Math.tan (phi);
    't2 = t * t;
    'tmp = (t2 * t2 * t2) - Math.pow (t, 6.0);
    t = Tan(Phi)                                    'Precalculate t
    t2 = t * t
    Rem Tmp = (t2 * t2 * t2) - (t ^ 6)              '(Never Used)

    l = lambda - lambda0                            'Precalculate l

Rem     Precalculate coefficients for l**n in the equations below
Rem     so a normal human being can read the expressions for easting
Rem     and northing
Rem     -- l**1 and l**2 have coefficients of 1.0

    l3coef = 1 - t2 + nu2
    l4coef = 5 - t2 + 9 * nu2 + 4 * (nu2 * nu2)
    l5coef = 5 - 18 * t2 + (t2 * t2) + 14 * nu2 - 58 * t2 * nu2
    l6coef = 61 - 58 * t2 + (t2 * t2) + 270 * nu2 - 330 * t2 * nu2
    l7coef = 61 - 479 * t2 + 179 * (t2 * t2) - (t2 * t2 * t2)
    l8coef = 1385 - 3111 * t2 + 543 * (t2 * t2) - (t2 * t2 * t2)

     Tmp1 = (n / 6 * Cos(Phi) ^ 3 * l3coef * (l ^ 3))           'Interim Result
     Tmp2 = (n / 120 * Cos(Phi) ^ 5 * l5coef * l ^ 5)           'Interim Result
     Tmp3 = (n / 5040 * Cos(Phi) ^ 7 * l7coef * l ^ 7)          'Interim Result
     x = n * Cos(Phi) * l + Tmp1 + Tmp2 + Tmp3                  'Calculate easting (x)

     Tmp1 = (t / 2 * n * Cos(Phi) ^ 2 * l ^ 2)                   'Interim Result
     Tmp2 = (t / 24 * n * Cos(Phi) ^ 4 * l4coef * l ^ 4)        'Interim Result
     Tmp3 = (t / 720 * n * Cos(Phi) ^ 6 * l6coef * l ^ 6)       'Interim Result
     Tmp4 = (t / 40320 * n * Cos(Phi) ^ 8 * l8coef * l ^ 8)     'Interim Result
     y = ArcLengthOfMeridian(Phi) + Tmp1 + Tmp2 + Tmp3 + Tmp4   'Calculate northing (y)
End Sub

Rem --------------------   UTM to Latitude/Longitude set   --------------------


Sub CvtUTM2LL(UTMCode As String, easting As Double, northing As Double, Lat As Double,
Lng As Double, Status As String)
Rem Convert UTM Numeric Grid to Latitude and Longitude (in degrees)
```

```
Rem UTMCode takes the form Xnn where nn=Zone Code(1-60) and X=(N=North,S=South)
Dim x As Double                                 'UTM x
Dim y As Double                                 'UTM y
Dim LatR As Double                              'Latitude in Radians
Dim LngR As Double                              'Longitude in Radians
Dim Wrk As String                               'Work String
Dim Zone As Integer                             'UTM Zone Number
Dim CMed As Double                              'Central merdian of UTM Zode
    Wrk = UTMCodeTidy(UTMCode)                  'Force format of UTM code

    Rem Calculate Zone Number
    Zone = Val(Mid$(Wrk, 2))                    'Extract and calculate Zone Number
    If Zone <= 0 Or Zone > 60 Or Mid$(Wrk, 1, 1) = "?" Then
        Status = "Invalid Zone " + Wrk
        Exit Sub                                'Quit as Job Done
    End If
    'CMed = (Zone * 6 - 180)-3                   'Calculate Central Meridian of Zone
    CMed = Zone * 6 - 183                       'Calculate Central Meridian of Zone
    CMed = CMed / 180 * Pi                      'Convert to Radians

    x = (easting * 1000) - 500000               'Hold centralised easting in metres
    x = x / UTMScaleFactor                      'Correct for UTM
    Rem If in southern hemisphere, adjust y accordingly.
    y = northing * 1000                         'Hold North Value in metres
    If Mid$(Wrk, 1, 1) = "S" Then y = y - 10000000
    y = y / UTMScaleFactor                      'Correct for UTM
    MapXYToLatLon x, y, CMed, LatR, LngR        'Convert to Lat/Long (in Radians)

    Lat = LatR / Pi * 180                       'Convert Latitude  to Degrees
    Lng = LngR / Pi * 180                       'Convert Longitude to Degrees

End Sub

Function CvtUTMFull2Short(Raw As String)
Rem This routine will convert a Full UTM Zone code to a short version that
Rem only contains North and Southern Hemisphere codes.
Rem Output takes the form Xnn where X=(N=North,S=South) and nn=Zone Code(01-60)
Rem Initial Design 01/June/2013 Author R. J. Spriggs
Dim Zone As Integer                                 'Zone Number
Dim Hem As String                                   'North or South Code
Dim CPnt As Integer                                 'General Character Pointer
Dim Pnt As Integer                                  'General Pointer
Dim Char As String                                  'Character store
    Hem = "N"                                       'Assume Northern Hemisphere
    Zone = 0                                        'Assume an Invalid Zone

    For CPnt = 1 To Len(Raw)                        'Parse all characters in Raw String
        Char = Mid$(Raw, CPnt, 1)                   'Extract a character
        Pnt = InStr("0123456789", Char)             'Check if valid Digit
        If Pnt <> 0 Then                            'When Valid, update Zone value
            Zone = Zone * 10 + Pnt - 1              'Calculate new Zone value
        Else                                        'Try to Locate Hemisphere code
            If InStr("NPQRSTUVWXYZ", Char) <> 0 Then Hem = "N"
            If InStr("ABCDEFGHJKLM", Char) <> 0 Then Hem = "S"
        End If
    Next CPnt
    If Zone < 10 Then Hem = Hem + "0"               'Zone always 2 digits
    CvtUTMFull2Short = Hem + Mid$(Str$(Zone), 2)    'Produce form Xnn

End Function

Function UTMCodeTidy(Raw As String)
Rem This routine will Tidy a UTM code (redundant characters removed)
Rem UTMCode takes the form Xnn where nn=Zone Code(1-60) and X=an alpha A->H,J->N,P->Z
Rem Initial Design 02/June/2013 Author R. J. Spriggs
Dim Zone As Integer                                 'Zone Number
Dim Cde As String                                   'Zone Code
Dim Ref As String                                   'Valid Zone Codes
Dim CPnt As Integer                                 'General Character Pointer
Dim Pnt As Integer                                  'General Pointer
```

```
Dim Char As String                                  'Character store
    Cde = "?"                                       'Assume unknown Region
    Zone = 0                                        'Assume an Invalid Zone
    Ref = "ABCDEFGHJKLMNPQRSTUVWXYZ"

    For CPnt = 1 To Len(Raw)                        'Parse all characters in Raw String
        Char = UCase$(Mid$(Raw, CPnt, 1))           'Extract a character in Uppercase
        Pnt = InStr("0123456789", Char)             'Check if valid Digit
        If Pnt <> 0 Then                            'When Valid, update Zone value
            Zone = Zone * 10 + Pnt - 1              'Calculate new Zone value
        Else                                        'Try to Locate Valid Hemisphere code
            If InStr(Ref, Char) <> 0 Then Cde = Char
        End If
    Next CPnt
    If Zone > 60 Or Zone < 1 Then Zone = 0          'Validate Zone Range
    If Zone < 10 Then Cde = Cde + "0"               'Zone always 2 digits
    UTMCodeTidy = Cde + Mid$(Str$(Zone), 2)         'Produce form Xnn
End Function

Function FootpointLatitude(y As Double)
Rem Computes the footpoint latitude for use in converting transverse
Rem Mercator coordinates to ellipsoidal coordinates.
Rem
Rem Reference: Hoffmann-Wellenhof, B., Lichtenegger, H., and Collins, J.,
Rem GPS: Theory and Practice, 3rd ed.  New York: Springer-Verlag Wien, 1994.
Rem
Rem Inputs:
Rem     y - The UTM northing coordinate, in meters.
Rem
Rem Returns:
Rem     The footpoint latitude, in radians.
Rem
Dim y_ As Double, alpha_ As Double, beta_ As Double, gamma_ As Double
Dim delta_  As Double, epsilon_ As Double, n As Double
Dim Tmp1 As Double, Tmp2 As Double, Tmp3 As Double

    n = (sm_a - sm_b) / (sm_a + sm_b)       'Precalculate n         (Eq. 10.18)
                                            'Precalculate alpha_    (Eq. 10.22)
                                            '(Same as alpha in       Eq. 10.17)
    alpha_ = ((sm_a + sm_b) / 2) * (1 + (n ^ 2 / 4) + (n ^ 4 / 64))
    y_ = y / alpha_                         'Precalculate y_        (Eq. 10.23)
                                            'Precalculate beta_     (Eq. 10.22)
    beta_ = (3 * n / 2) + (-27 * n ^ 3 / 32) + (269 * n ^ 5 / 512)
                                            'Precalculate gamma_    (Eq. 10.22)
    gamma_ = (21 * n ^ 2 / 16) + (-55 * n ^ 4 / 32)
                                            'Precalculate delta_    (Eq. 10.22)
    delta_ = (151 * n ^ 3 / 96) + (-417 * n ^ 5 / 128)
    epsilon_ = (1097 * n ^ 4 / 512)         'Precalculate epsilon_  (Eq. 10.22)

    Rem Now calculate the sum of the series                         (Eq. 10.21)
    Tmp1 = (gamma_ * Sin(4 * y_))           'Interim Result
    Tmp2 = (delta_ * Math.Sin(6 * y_))      'Interim Result
    Tmp3 = (epsilon_ * Math.Sin(8 * y_))    'Interim Result

    FootpointLatitude = y_ + (beta_ * Sin(2 * y_)) + Tmp1 + Tmp2 + Tmp3
End Function

Sub MapXYToLatLon(x As Double, y As Double, lambda0 As Double, Lat As Double, Lng As
Double)

Rem Converts x and y coordinates in the Transverse Mercator projection to
Rem a latitude/longitude pair.  Note that Transverse Mercator is not
Rem the same as UTM; a scale factor is required to convert between them.
Rem
Rem Reference: Hoffmann-Wellenhof, B., Lichtenegger, H., and Collins, J.,
Rem    GPS: Theory and Practice, 3rd ed.  New York: Springer-Verlag Wien, 1994.
Rem
Rem  Inputs:
Rem   x - The easting of the point, in meters.
Rem   y - The northing of the point, in meters.
```

```
Rem   lambda0 - Longitude of the central meridian to be used, in radians.
Rem
Rem  Outputs:
Rem   philambda - A 2-element containing the latitude and longitude
Rem               in radians. VB Mod Replaced with Lat and Lng
Rem
Rem  Remarks:
Rem   The local variables Nf, nuf2, tf, and tf2 serve the same purpose as
Rem   N, nu2, t, and t2 in MapLatLonToXY, but they are computed with respect
Rem   to the footpoint latitude phif.
Rem
Rem   x1frac, x2frac, x2poly, x3poly, etc. are to enhance readability and
Rem   to optimize computations.
Rem
Dim phif As Double, Nf As Double, Nfpow As Double, nuf2 As Double
Dim ep2 As Double, tf As Double, tf2 As Double, tf4 As Double, cf As Double
Dim x1frac As Double, x2frac As Double, x3frac As Double, x4frac As Double
Dim x5frac As Double, x6frac As Double, x7frac As Double, x8frac As Double
Dim x2poly As Double, x3poly As Double, x4poly As Double
Dim x5poly As Double, x6poly As Double, x7poly As Double, x8poly As Double
Dim Tmp1 As Double, Tmp2 As Double, Tmp3 As Double

                                                 'Get the value of phif,
    phif = FootpointLatitude(y)                  'the footpoint latitude.
    ep2 = (sm_a ^ 2 - sm_b ^ 2) / sm_b ^ 2       'Precalculate ep2
    cf = Cos(phif)                               'Precalculate cos (phif)
    nuf2 = ep2 * cf ^ 2                          'Precalculate nuf2
    Nf = sm_a ^ 2 / (sm_b * Sqr(1 + nuf2))       'Precalculate Nf
    Nfpow = Nf                                   'initialize Nfpow
    tf = Tan(phif)                               'Precalculate tf
    tf2 = tf * tf: tf4 = tf2 * tf2               'Tan(phif)**2 , Tan(phif)**4

    Rem  Precalculate fractional coefficients for x**n in the equations
    Rem  below to simplify the expressions for latitude and longitude.

    x1frac = 1 / (Nfpow * cf)

    Nfpow = Nfpow * Nf                           'now equals Nf**2
    x2frac = tf / (2 * Nfpow)

    Nfpow = Nfpow * Nf                           'now equals Nf**3
    x3frac = 1 / (6 * Nfpow * cf)

    Nfpow = Nfpow * Nf                           'now equals Nf**4
    x4frac = tf / (24 * Nfpow)

    Nfpow = Nfpow * Nf                           'now equals Nf**5
    x5frac = 1 / (120 * Nfpow * cf)

    Nfpow = Nfpow * Nf                           'now equals Nf**6
    x6frac = tf / (720 * Nfpow)

    Nfpow = Nfpow * Nf                           'now equals Nf**7
    x7frac = 1 / (5040 * Nfpow * cf)

    Nfpow = Nfpow * Nf                           'now equals Nf**8
    x8frac = tf / (40320 * Nfpow)

    Rem Precalculate polynomial coefficients for x**n.
    Rem -- x**1 does not have a polynomial coefficient.

    x2poly = -1 - nuf2
    x3poly = -1 - 2 * tf2 - nuf2
    x4poly = 5 + 3 * tf2 + 6 * nuf2 - 6 * tf2 * nuf2 - 3 * (nuf2 * nuf2) - 9 * tf2 *
(nuf2 * nuf2)
    x5poly = 5 + 28 * tf2 + 24 * tf4 + 6 * nuf2 + 8 * tf2 * nuf2
    x6poly = -61 - 90 * tf2 - 45 * tf4 - 107 * nuf2 + 162 * tf2 * nuf2
    x7poly = -61 - 662 * tf2 - 1320 * tf4 - 720 * (tf4 * tf2)
    x8poly = 1385 + 3633 * tf2 + 4095 * tf4 + 1575 * (tf4 * tf2)
```

```
    Rem Calculate latitude (was philambda [0])
    Tmp1 = x4frac * x4poly * x ^ 4              'Interim Result
    Tmp2 = x6frac * x6poly * x ^ 6              'Interim Result
    Tmp3 = x8frac * x8poly * x ^ 8              'Interim Result
    Lat = phif + x2frac * x2poly * (x * x) + Tmp1 + Tmp2 + Tmp3

    Rem Calculate longitude (was philambda [1])
    Tmp1 = x3frac * x3poly * x ^ 3
    Tmp2 = x5frac * x5poly * x ^ 5
    Tmp3 = x7frac * x7poly * x ^ 7
    Lng = lambda0 + x1frac * x + Tmp1 + Tmp2 + Tmp2

End Sub




Rem ======================================================================
Rem The following routine must be called before any UTM conversion
Rem are processed as it initialises then UTM Constants.
Rem ======================================================================

Public Sub SetMajorMinorAxis(AreaSelect As Integer)
Rem Initialise Major/Minor UTM Axis
Rem Constants Information located in document
Rem Steven Dutch Natural and Applied Science (University of Wisconsin Green Bay)
Rem Converting UTM to Latitude and Longitude (Or Vice Versa)
Rem Initial Design 24/Jun/2013 Author R. J. Spriggs
Dim ecc As Double                           'Eccentricity
Dim ep2 As Double                           'Eccentricity Prime Squared

Rem sm_a is Equatorial Radius in metres
Rem sm_b is Polar      Radius in metres

    Select Case AreaSelect
        Case Is = 0                         'WSG84 NAD83 Global     (Rounded to 4dp)
            sm_a = 6378137                  'Ellipsoid model major axis.
            sm_b = 6356752.3142             'Ellipsoid model minor axis.

        Case Is = 1                         'GRS80 US               (Rounded to 4dp)
            sm_a = 6378137                  'Ellipsoid model major axis.
            sm_b = 6356752.3141             'Ellipsoid model minor axis.

        Case Is = 2                         'WSG72 NASA DOD
            sm_a = 6378135                  'Ellipsoid model major axis.
            sm_b = 6356750.5                'Ellipsoid model minor axis.

        Case Is = 3                         'Australia 1965
            sm_a = 6378160                  'Ellipsoid model major axis.
            sm_b = 6356774.7                'Ellipsoid model minor axis.

        Case Is = 4                         'Krasovsky 1940 Soviet Union
            sm_a = 6378245                  'Ellipsoid model major axis.
            sm_b = 6356863                  'Ellipsoid model minor axis.

        Case Is = 5                         'International (1924)-Hayford (1909) (global)
            sm_a = 6378388                  'Ellipsoid model major axis.
            sm_b = 6356911.9                'Ellipsoid model minor axis.

        Case Is = 6                         'Clark 1880 France Africa
            sm_a = 6378249.1                'Ellipsoid model major axis.
            sm_b = 6356514.9                'Ellipsoid model minor axis.

        Case Is = 7                         'Clark 1866 North America
            sm_a = 6378206.4                'Ellipsoid model major axis.
            sm_b = 6356583.8                'Ellipsoid model minor axis.

        Case Is = 8                         'Airy 1830 Great Britian
            sm_a = 6377563.4                'Ellipsoid model major axis.
```

```
            sm_b = 6356256.9                'Ellipsoid model minor axis.

        Case Is = 9                         'Bessel 1841 Central Europe Chilie Indonesia
            sm_a = 6377397.2                'Ellipsoid model major axis.
            sm_b = 6356079                  'Ellipsoid model minor axis.

        Case Is = 10                        'Everest 1830 South Asia
            sm_a = 6377276.3                'Ellipsoid model major axis.
            sm_b = 6356075.4                'Ellipsoid model minor axis.

        Case Else                           'Default WSG84  (Rounded to 3dp)
            sm_a = 6378137                  'Ellipsoid model major axis.
            sm_b = 6356752.314              'Ellipsoid model minor axis.
    End Select

    'ecc = Sqr(1 - (sm_b ^ 2 / sm_a ^ 2))    'Calculate eccentricity
    'ep2 = ecc ^ 2 / (1 - ecc ^ 2)           'e prime squared

    Rem sm_EccSquared = 0.00669437999013    'Never used
    UTMScaleFactor = 0.9996                 'A UTM default Constant
End Sub
```