```
Attribute VB_Name = "Maths"
Option Explicit
Rem
Rem Author R. J. Spriggs August 2005
Rem Programmers: The Visual Basic source code in this document may be copied
Rem and reused any for non commercial use.
Rem This module will also be needed when programme sources are used
Rem GCircle.Bas , GPS_OS_bas , GPS_UTM_bas
Rem

Rem Maths Functions and Subroutines.
Rem ================================
Rem
Rem         Functions and Subroutines located in this Module.
Rem
Rem     Type        Name                    Comment
Rem
Rem     Function    ACos                Inverse CoSine Routine  (Ready and Now Used)
Rem     Function    ASin                Inverse Sine Routine    (Ready and Now Used)
Rem     Function    ATan                Inverse Tangent Routine (Ready but Not Used)
Rem     Function    C2M                 Converts Compass angles to Maths Angles
Rem     Function    D2R                 Converts Degrees to Radians
Rem     Function    R2D                 Converts Radians to Degrees
Rem     Subroutine  D2DMS               Converts Degrees to Degrees,Minutes,Seconds
Rem     Subroutine  DMS2D               Converts Degrees,Minutes,Seconds to Degrees


' Calculation Constants
Public Const Pi = 3.14159265358979      'High Accuracy PI
'Public Const Pi = 3.141592654          'Low  Accuracy PI


Public Function ACos(Ang As Double)
    'Inverse CoSine Routine
    'This routine does NOT exist as an Intrinsic VB Function
    'x is the cosine of the angle you want and must be from -1 to 1.
    'This routine returns an angle in radians in the range 0 to Pi
    'Author R. J. Spriggs  Creation Date 04/12/2005  Last update 12/12/05
    'Mod RJS 10/12/2005 90 degree (x=1) check and correct
    'Mod RJS 12/04/2006 Algorithm error corrected
    'Mod RJS 08/08/2012 Stop Removed, Invalid value limited to Max or Min
    Dim x As Double

    x = Ang                                             'Hold Entry Value
    'If x > 1 Or x < -1 Then Stop                       'Value outside valid range
    If x > 1 Then x = 1                                 'Limit to Max +90 Degrees
    If x < -1 Then x = -1                               'Limit to Min -90 Degrees
    If x = 1 Or x = -1 Then                             'Check if +/-90 Degrees
        ACos = (1 - x) * Pi / 2
    Else
        'Ang = x / Sqr(-x * x + 1)
        'ACos = Atn(Ang) + Pi / 2
        'ACos = Atn(x / Sqr(-x * x + 1)) + Pi / 2   'removed 12/04/2006
        ACos = (Pi / 2) - Atn(x / Sqr(-x * x + 1))  'added 12/04/2006
    End If
End Function

Public Function ASin(x As Double)
    'Inverse Sine Routine
    'This routine does NOT exist as an Intrinsic Function
    'x is the sine of the angle you want and must be from -1 to 1.
    'This routine returns an angle in radians in the range -Pi/2 and Pi/2
    'Author R. J. Spriggs  Creation Date 04/12/2005  Last update 12/12/05
    'Mod RJS 10/12/2005 90 degree (x=1) check and correct
    'Mod RJS 30/04/2012 Invalid entry = Responce=0 to allow routine to continue (removed)

    If x > 1 Or x < -1 Then Stop                    'Value outside valid range
    'If X > 1 Or X < -1 Then                         'Value outside valid range
    '    ASin = 0: Exit Function                     'A wrong Answer
    'End If
```

```
    If x = 1 Or x = -1 Then
        ASin = (Pi / 2) * x
    Else
        ASin = Atn(x / Sqr(-x * x + 1))
    End If
End Function

'Public Function ATan(Angle As Double) As Double
'    'Inverse Tangent Routine
'    'This routine does exist as an Intrinsic Function
'    'Angle is the tangent of the angle you want and must be from -inf to inf.
'    'This routine returns an angle in radians in the range -Pi/2 and Pi/2
'    'Author R. J. Spriggs  Creation Date 13/02/2012  Last update 13/02/12
'    'Problems NOT very accurate Drifts as we approach +45 or -45 degs
'    'Mod RJS 29/03/2012 Recipical Concept implemented (Using code by Judson D McClendon)
'    ' This corrected problems >45 to 90 or  <-45 to -90 degs
'    'Mod RJS 29/03/2012 Using rewritten QB45 code by Judson D McClendon Accuracy
improved
'
'Dim Ang As Double
'Dim Cnt As Double
'Dim Val As Double
'Dim Recip As Boolean
'
'Dim ItCnt As Integer
'Dim ItVal As Double
'
'Dim C1 As Double                        'Variables used by J D McClendon
'Dim CX As Double
'Dim Dif As Double
'Dim LastSum As Double
'Dim N As Double
'Dim NCX As Double
'Dim NSX As Double
'Dim S1 As Double
'Dim SX As Double
'Dim Sign As Boolean
'Dim Sum As Double
'Dim Term As Double
'Dim X As Double
'Dim XSQ As Double
'Dim Y As Double
'
'    Recip = False                   'Assume Reciprocal not used
'    Sign = False                    'Assume No Sign CHange
'    Ang = Angle                     'Hold Entry Angle
'
'    If Ang < 0 Then
'        Ang = -Ang                  'Angle Now Positive
'        Sign = True                 'Indicate sign changed
'    End If
'
'    If Ang > 1 Then                 'When Over 45 degs
'        Ang = 1 / Ang               'Hold reciprocal Angle
'        Recip = True                'Indicate Reciprocal angle used
'    End If
'
'
''    'ATan = X - X ^ 3 / 3 + X ^ 5 / 5 - X ^ 7 / 7 + X ^ 9 / 9 - X ^ 11 / 11 + X ^ 13 /
13 - X ^ 15 /

15
''    'ATan = X - X ^ 3 / 3 + X ^ 5 / 5 - X ^ 7 / 7 + X ^ 9 / 9 - X ^ 11 / 11 + X ^ 13 /
13
''    Val = 0
''    For Cnt = 1 To 19 Step 4
''        Val = Val + (Ang ^ Cnt) / Cnt
''        Val = Val - (Ang ^ (Cnt + 2)) / (Cnt + 2)
''    Next Cnt
'
```

```
'
'    Rem This is the start of Judson D McClendon main conversion code Modified by RSP
'
'    CX = 1 / Sqr(1 + Ang * Ang)
'    SX = CX * Ang
'
'    Dif = 0
'    ItVal = 1                              'Start Iteration Value
'    For ItCnt = 1 To 3
'        ItVal = ItVal / 10                 'Calculate New step 0.1 , 0.01 , 0.001
'        C1 = Cos(ItVal)
'        S1 = Sin(ItVal)
'        While SX > S1
'            Dif = Dif + ItVal
'            NCX = CX * C1 + SX * S1        'Calculate Next Iteration value
'            NSX = SX * C1 - CX * S1
'            CX = NCX                       'Hold Iteration value
'            SX = NSX
'        Wend
'    Next ItCnt                             'All 3 Iterations processed
'
'    X = SX / CX
'    XSQ = X * X
'    Y = XSQ / (1 + XSQ)
'    Term = 1
'    Sum = 1
'    N = 0
'
'    Do
'        N = N + 2
'        Term = (Term * N * Y) / (N + 1)
'        LastSum = Sum
'        Sum = Sum + Term
'    Loop While (Sum <> LastSum)
'
'    Val = Sum * Y / X + Dif
'
'    Rem This is the end of Judson D McClendon main conversion code Modified by RSP
'
'    If Recip = True Then            'When Reciprocal used
'        Val = (Pi / 2) - Val        'Count Angle back from 90 degs
'    End If
'    If Sign = True Then             'When Sign Changed
'        Val = -Val                  'Negate Angle
'    End If
'    ATan = Val                      'Reply with Converted Angle
'
'End Function

Public Function C2M(Angle As Double) As Double
Rem This routine will convert Compass angles to Maths Angles
Rem Initial Design 5th May 2012 R. J. Spriggs
Dim Ang
    Ang = 180 - (Angle + 90)                    'Converted Angle
    If Ang < 0 Then Ang = Ang + 360             'Make angle +ve
    C2M = Ang                                   'Reply with Converted Angle
End Function

Public Function D2R(Angle As Double) As Double
Rem This routine will convert Degrees to Radians
'Mod RJS 16/02/2012 Algorithm error corrected (MOD operator Integer Rounds)
'Mod RJS 26/04/2012 Algorithm Modification    Convert -ve Angles to +ve Angles 0-360
degrees
Dim A As Double
    A = Angle: While A < 0: A = A + 360: Wend   'Keep Angle Positive
    'A = Angle Mod 360                          'This operator Integer rounds the angle
    While A > 360: A = A - 360: Wend            'Iterative MOD instruction

    'D2R = (Angle Mod 360) * Pi / 180           'Convert Value
    D2R = A * Pi / 180                          'Convert Value
```

```
End Function

Public Function R2D(Angle As Double) As Double
Rem This routine will convert Radians to Degrees (2*Pi Rads = 360 Degs)
Rem Initial Design 07/Aug/2012 Author R. J. Spriggs
'Mod RJS ../../2012 ?
'Dim A As Double
'    A = Angle                                  'Hold angle

    R2D = Angle * 180 / Pi                     'Convert Value
End Function

Public Sub D2DMS(D As String, Degs As String, Mins As String, Secs As String)
Rem This routine will convert Degrees to Degrees Minutes and Seconds
Rem Initial Design 13/Nov/2012 Author R. J. Spriggs
Dim Tmp As Double                              'General Work Value
Dim Wrk As Integer                             'General Work Value

    Tmp = 0                                    'Assume default conversion
    If IsNumeric(D) = True Then Tmp = Val(D)   'Convert I/P value
    If Tmp < 0 Then Tmp = -Tmp                 'Value always positive
    Wrk = Int(Tmp)                             'Hold whole Degrees Value
    Degs = Mid$(Str$(Wrk), 2)                  'Hold Extracted Degrees Value
    Tmp = (Tmp - Wrk) * 60                     'Shift Mins to Integer Value
    Wrk = Int(Tmp)                             'Hold whole Miniutes Value
    Mins = Mid$(Str$(Wrk), 2)                  'Hold Extracted Minutes Value
    Tmp = (Tmp - Wrk) * 60                     'Shift Secs to Integer Value
    Wrk = Int(Tmp)                             'Hold whole Seconds Value
    Secs = Mid$(Str$(Wrk), 2)                  'Hold Extracted Seconds Value
End Sub

Public Sub DMS2D(Degs As String, Mins As String, Secs As String, D As String)
Rem This routine will convert Degrees Minutes and Seconds to Degrees
Rem Initial Design 13/Nov/2012 Author R. J. Spriggs
Dim Tmp As Double                                  'General Work Value
Dim WD As Integer                                  'General Work Value
Dim WM As Integer                                  'General Work Value
Dim WS As Integer                                  'General Work Value

    WD = 0                                         'Assume default conversion
    If IsNumeric(Degs) = True Then WD = Val(Degs)  'Convert I/P value
    WM = 0                                         'Assume default conversion
    If IsNumeric(Mins) = True Then WM = Val(Mins)  'Convert I/P value
    WS = 0                                         'Assume default conversion
    If IsNumeric(Degs) = True Then WS = Val(Secs)  'Convert I/P value
    Tmp = WD + (WM / 60) + (WS / 3600)             'Calculate Composite Value
    D = Mid$(Str$(Tmp), 2)                         'Hold Composite Value
End Sub
```