```
'========================================================================
' check length of connecting rods.
'========================================================================
' first; two dimensional vector, then arithmetic vector functions
'------------------------------------------------------------------------
Type v2d    ' Vector, 2 dimensional = point coordinate = complex number
      As Double x     ' real component
      As Double y     ' imaginary component
    Declare Operator Cast() As String
End Type

#define xy(p) (p.x,p.y)  ' convert v2d to a comma separated pair for FB_GFX

' here are the vector operators needed for solving linkages
Operator v2d.cast () As String
Return "(" + Str(x) + ", " + Str(y) + ")"
End Operator

Operator + ( Byval a As v2d, Byval b As v2d ) As v2d    ' addition
Return Type( a.x + b.x, a.y + b.y )
End Operator

Operator - ( Byval a As v2d, Byval b As v2d ) As v2d    ' subtraction
Return Type( a.x - b.x, a.y - b.y )
End Operator

Operator * ( Byval a As v2d, Byval b As v2d ) As v2d ' multiplication is
Return Type( a.x * b.x - a.y * b.y , a.x * b.y + a.y * b.x ) ' vector rotation
End Operator

Operator * ( Byval a As Double, Byval b As v2d ) As v2d ' scalar * vector
Return Type( a * b.x , a * b.y )
End Operator

Operator * ( Byval a As v2d, Byval b As Double ) As v2d ' vector * scalar
Return Type( a.x * b , a.y * b )
End Operator

Function Cinv( Byval a As v2d ) As v2d      ' complex reciprocal
    Dim As Double sos = a.x * a.x + a.y * a.y   ' sum of squares
    Return Type( a.x / sos, -a.y / sos )    ' uses scalar division
End Function

Operator / ( Byval a As v2d, Byval b As v2d ) As v2d    ' vector division
Return a * Cinv( b )
End Operator

Function Radius(Byval a As v2d) As Double   ' vector length, or radius
    Return Sqr( a.x * a.x + a.y * a.y )
End Function

Function Angle(Byval a As v2d) As Double    ' +X axis is zero. +Y is positive.
    Return Atan2( a.y, a.x )                ' -Pi < angle < +Pi, in radians
End Function

Function Conj( Byval a As v2d ) As v2d  ' conjugate = negate imaginary part
    Return Type( a.x , -a.y )
End Function    ' * Conj( ) causes time and vectors to rotate backwards

'------------------------------------------------------------------------
Function circles_intersections(_ ' Do NOT Edit this Function, you will break it
    _   ' input parameters
    Byref c1 As v2d,_       ' circle 1 centre
    Byref r1 As Double,_    '          radius
```

```
    Byref c2 As v2d,_       ' circle 2 centre
    Byref r2 As Double,_    '          radius
    _   ' output parameters, viewed from c1 to c2, the intersection
    Byref lhs As v2d,_      ' point on the left
    Byref rhs As v2d _      ' point on the right
    ) As Integer  ' returns the number of points of intersection { 0, 1 or 2 }
    '  W A R N I N G :  Here be dragons, hidden in the unusual exceptions
    Dim As v2d s = c2 - c1      ' centre difference vector from c1 to c2
    Dim As Double rs = radius( s )      ' distance between centres
    If rs >     ( r1 + r2 ) Then Return 0   ' circles are quite separate
    If rs <= Abs( r1 - r2 ) Then Return 0   ' one inside the other, or identical
    s = s * ( r1 / rs )         ' scale s to length of r1
    If rs < ( r1 + r2 ) Then    ' there are two points of intersection
        Dim As Double cosa = ( rs*rs + r1*r1 - r2*r2 ) / ( 2 * rs * r1 )
        Dim As v2d t = Type( cosa, Sqr( 1 - cosa*cosa ) )  ' cos(a), +sin(a)
        lhs = c1 + s * t    ' on the left when viewed from c1 towards c2
        rhs = c1 + s * Conj( t ) ' conjugate, gives the point on the right
        Return 2
    End If
    lhs = c1 + s    ' osculation, so both points of intersection
    rhs = lhs       '    are assigned identical values
    Return 1
End Function

' Circles_Intersections()   Notes;
'
' Viewing from the centre c1 toward the centre c2, the first intersection
'   point "lhs" is on the left, while the second, "rhs" is on the right.
' This function returns an integer indicating the number of points of
'   contact determined. That integer can be 0, 1 or 2.
'
' This function is engineered to use floating point numbers applied to
'   the physical world. The use of floating point numbers with granularity
'   generates noise on the centre position and on the radius.
'
' When circle circumferences cross, this function identifies the two points
'   of intersection and returns 2. External osculation gives both those
'   points identical values, and returns 1. Internal osculation and
'   non-contact both return 0, but do not change previous values.
'
' As two identical circles pass coincidence, the velocity of the points
'   of intersection approach infinity. This is not practical. It can be
'   seen as the equivalent of gimbal lock in a mechanical system. Adding
'   granular noise to the radius resolves the coincidence and avoids any need
'   to return an infinite number of contact points. I therefore make the
'   assumption that any two circles with coincident centres do not contact.
'
' Internal osculation in the presence of noise can also approach this
'   case, so it is best treated as a non-contact condition.

'-----------------------------------------------------------------------
Const As Double Pi = 4 * Atn( 1 )
Const As Double TwoPi = 2 * Pi

'=======================================================================
' from here specify the linkage dimensions
'=======================================================================
' Link spacing on short (motor) arm: 4.25in. (11cm)
' Link spacing on large disk: 7.25in. (18.5cm)
' Link length: 7.5in (19cm)
'-----------------------------------------------------------------------
' Link lengths

' The short motor bar pins, A above and B below, centre is O at origin
```

```
' The pins on the bar lie on a radius of; 110 / 2 = 55.0 mm
Dim As Double bar = 55      ' distance of bar pins from origin

' The disk pins, C above and D below, centred at E = ( axial, 0 )
' The pins on the disk lie on a radius of; 185 / 2 = 92.5 mm
Dim As Double disk = 92.5   ' distance of disk pins from E

' The link length is; 190 mm
Dim As Double rod = 190     ' Sqr( ( disk - bar )^2 + axial^2 ) ' length of rods

' find distance between bar and disk centres along the x-axis
Dim As Double axial = Sqr( rod^2 - (disk - bar)^2 )  ' 186.2625834675338
'
'  bar           disk                bar            disk
'                 C                                  C
'   A                                  A
'   O    axial    E    distorts to      O     axial     E
'   B                                    B
'                 D                                     D
'
' Capitalised variables are 2D vectors, point coordinates or complex numbers
Dim As v2d O, A, B, C, D, E

' there are two fixed points
O = Type(     0, 0 )    ' origin, centre of bar
E = Type( axial, 0 )    ' centre of disk on x-axis

'-------------------------------------------------------------------------
' Strategy
'   Assume the length of one connecting rod remains as 190 mm.
'   Compute the length of the other rod as the short bar rotates through 90°

'=========================================================================
Print using       " Fixed rod = ###.# mm"; rod
Print
Print             " Bar      Variable "
Print             " deg     rod length "
Dim As String f = "####     ####.###  "

' now solve linkage for bar rotation angle theta
Dim As Integer count    ' number of circle intersection points found, {0,1,2}
Dim As v2d trash        ' a place for the unwanted intersection
For deg As Integer = 0 To 90 Step 5
    Dim As Double theta = deg * Pi / 180    ' rotation of bar in radians
    A = Type( bar * Sin(theta), bar * Cos(theta) )    ' point A position
    B = Type( -A.x, -A.y )      ' B is across from the origin
    ' C is the LHS intersection point of two circular arcs when
    '      viewed from c1 to c2  ( c1, r1, c2,  r2, lhs,   rhs )
    count = circles_intersections(  A, rod, E, disk,  C, trash )
    If count < 1 Then Print " Failed to find C." : Sleep : Stop
    ' now we know C, we can solve for D
    D = Type( 2 * axial - C.x , -C.y )
    Print Using f; deg; radius( B - D )
Next deg

'-------------------------------------------------------------------------
' Fixed rod = 190.0 mm
'
' Bar      Variable
' deg     rod length
'   0      190.000
'   5      190.034
'  10      190.138
'  15      190.318
```

```
'  20      190.581
'  25      190.938
'  30      191.403
'  35      191.996
'  40      192.739
'  45      193.663
'  50      194.807
'  55      196.217
'  60      197.948
'  65      200.065
'  70      202.638
'  75      205.737
'  80      209.421
'  85      213.726
'  90      218.648
'
'=============================================================================
Sleep   ' end of file
'=============================================================================
```