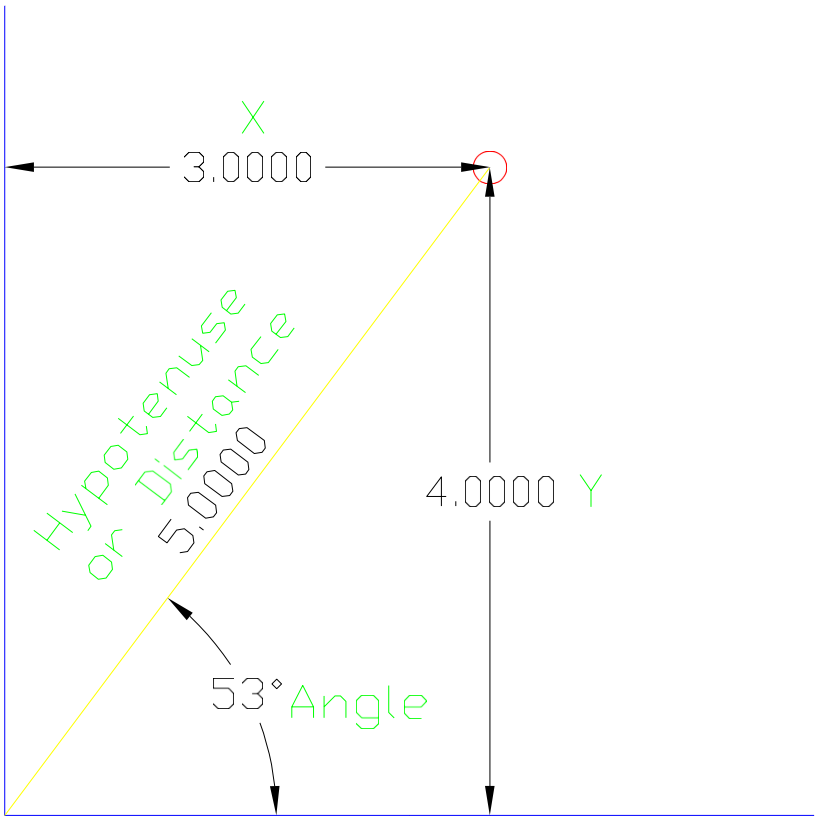
傻瓜式CORDIC

CORDIC是一种使用更简单的数学运算来计算一个数学函数的方法，在一个叫做二进制搜索的循环中。最常见的CORDIC是用来计算一个点的ATAN2（角度）和Hypotenuse（距离）。
CORDIC也可以用来计算其他数学函数，如SIN和COS。

假设你有一个点的坐标（X，Y），你想计算这个点的角度，你0,0.可以用ATAN2来做这个计算。

从坐标（X,Y）到（角度，距离）的转换也被称为矩形到极地的转换。

在下面的图片中，红色点的坐标是3,4。通过使用CORDIC算法，我们可以计算出斜边（5）和角度（53）。



在进入CORDIC之前，让我们看看二进制搜索是如何工作的。

猜谜游戏（高/低）。

我在想一个0到100之间的数字。你猜一猜，我会告诉你是太高还是太低。如果你想用最少的猜测次数找到这个数字，你的第一个猜测应该是什么？

你应该猜到50。为什么？因为这将告诉你这个数字是在0到50之间还是50到100之间。有效地将可能的数值分成了一半。

现在如果我告诉你，你的猜测太高了，那么你就会知道这个数字一定在0和49之间。那么你的下一个猜测是什么？

对
25.这将告诉你这个数字是在（0到25）还是（25到50）之间。你会在每次猜测中不断除以该值。这是寻找数值的最有效方法，被称为 "二进制搜索"。

这里有一个示范。未

知值是 80

50= 太低，所以+25
75=太低，所以+12
87=太高了，所以-
6=81太高了，所以
-
378=太低了，所以
+2=80你得到了。

猜谜游戏（加/减法）。

好的，现在是同样的游戏，只是这次我会想到一个从0到90的数字。你告诉我从我想到的数字中加减一个值，我会告诉你结果是正数还是负数。你是想让这个数字等于零。那么加减的总和就是你想猜的原值。

如果你想用最少的猜测次数使数字归零，你的第一个 "猜测 "应该是什么？你应该告诉我 "减去45"。

为什么？因为这将告诉你这个数字是在0到45之间还是45到90之间。如果减去后的数字是负数，那么45,它一定是0，如果45.它仍然是正数，那么它一定是45和90。

假设我说这个数字现在是负数（所以它一定是在和045之间）。你的第二个 "猜测 "应该是什么？你应该告诉我 "加22"。

现在我说这个数字是正数（所以它一定是在和2245之间）。你的下一个猜测应该是减去 11.

我们将继续下去，你的值是前一个值的一半。而操作（加或减）取决于现在的数字是负数还是正数。如果数字是负的，你会告诉我加这个值。如果数字是正的，你将告诉我减去这个值。这样做才有意义，因为我们毕竟是要让这个数字等于零。

如果你跟踪每一个操作，你可以很接近地知道原来的值是什么。下面是一个示范。
未知值是 43

减去	45（-2）负数
添加	22（+20）正数
减去	11（+9）正数
减去	5（+4）正数
减去	2（+2）正数
减去	1+1正
减去	10

和=-45+22-11-5-2-1-1=-43，使数字归零

猜谜游戏（用SIN和COS寻找ATAN）。

现在让我们用这个猜谜游戏的技巧来计算一个点的角度。假设你有一个点在x=5；y=10。从0,0到的角度是多少5,10？为了找到这个角度，你可以使用正切函数，如angle=arctangent(y/x)。

所以答案是63.435度。由于我们想找到度数，而不是简单地加减一个数值，我们需要加减度数。因此，实际上我们将顺时针(+)或逆时针(-)旋转该点。

下面是旋转一个点的经典公式。

逆时针。

```
Xnew = X * COS(angle) - Y * SIN(angle)
Ynew = Y * COS(angle) + X * SIN(angle)
Angle = Angle / 2
```

顺时针方向。

```
Xnew = X * COS(angle) + Y * SIN(angle)
Ynew = Y * COS(angle) - X * SIN(angle)
Angle = Angle / 2
```

用经典的方法来旋转一个点，你需要使用SIN和COS函数。现在你可能会问："我们到底要如何在微控制器上计算SIN和COS呢？我们不需要这样做。因为我们提前知道我们需要使用什么角度（45，等等）22,11,，我们可以简单地预先计算出这些角度的SIN和COS，并把它们放在一个表格里。

```
CosTable = Cos(45), Cos(22), Cos(11), Cos(5), Cos(2), Cos(1), Cos(1)
SinTable = Sin(45), Sin(22), Sin(11), Sin(5), Sin(2), Sin(1), Sin(1)
CurAngle = 45
和值角 = 0
循环数 = 0
```

逆时针。

```
Xnew = X * CosTable(LoopNum) - Y * SinTable(LoopNum)
Ynew = Y * CosTable(LoopNum) + X * SinTable(LoopNum)
SumAngle = SumAngle + CurAngle
CurAngle = CurAngle / 2
LoopNum = LoopNum + 1
```

顺时针方向。

```
Xnew = X * CosTable(LoopNum) + Y * SinTable(LoopNum)
Ynew = Y * CosTable(LoopNum) - X * SinTable(LoopNum)
SumAngle = SumAngle - CurAngle
CurAngle = CurAngle / 2
LoopNum = LoopNum + 1
```

正如你所看到的，即使我们使用SIN(angle)和COS(angle)的表格，仍然有相当多的乘法需要我们去做。但这

种方法对单片机来说是可行的。

当你把一个点旋转到零度时，所有旋转的总和等于原来的角度，结束的X坐标将等于原点的半径（斜面）。而Y坐标将为零。

猜谜游戏（用TAN找到ATAN（半径被缩放））。

为了减少乘法运算的数量，我们可以简化公式，使用TAN函数代替SIN和COS。唯一的副作用是斜边不能保持它的比例。但这种影响在每个循环中都是恒定的，所以在最后进行一次乘法运算就能使数值恢复到它应该有的样子。

逆时针。

```
Xnew = X - Y * TAN(angle)
Ynew = Y + X * TAN(angle) Angle = Angle / 2
```

顺时针

```
Xnew = X + Y * TAN(angle) Ynew = Y - X * TAN(angle) Angle = Angle / 2
```

同样，在微控制器中，TAN（角度）值将被存储在一个表中。

```
TanTable = Tan(45), Tan(22), Tan(11), Tan(5), Tan(2), Tan(1), Tan(1)
CurAngle = 45
和值角 = 0
循环数 = 0
```

逆时针。

```
Xnew = X - Y * TanTable(LoopNum)
Ynew = Y + X * TanTable(LoopNum)
SumAngle = SumAngle + CurAngle
CurAngle = CurAngle / 2
LoopNum = LoopNum + 1
```

顺时针

```
Xnew = X + Y * TanTable(LoopNum)
Ynew = Y - X * TanTable(LoopNum)
SumAngle = SumAngle - CurAngle
CurAngle = CurAngle / 2
LoopNum = LoopNum + 1
```

TanTable中的第一个值是Tan(45)。而恰好Tan(45)= 所以1.这很容易。

我们的下一个角度是22，所以Tan(22)=0.40402
嗯，太糟糕了，它不是0.5，我们可以用简单的移位操作来做，而不是完全的乘法。(提示...提示)。

下一个角度将是11，所以Tan(11)=0.19438
嗯，太糟糕了，这不是0.25，我们可以用简单的移位操作来实现（提示...提示）。

猜谜游戏（用TAN(角度)=1/(2^N)找到ATAN)

正如我们在上一个例子中看到的，我们仍然需要做一些乘法。但如果我们不使用45、22、11等最佳角度值呢？如果我们使用的角度值能给我们带来乘法，我们可以用移位来执行（比如00.5,.25，0.125，等等）。那么第一个角度45真的很容易，因为它是1.0。

那么，什么角度会给我们带来Tan(Angle)= 0.500.原来是
现在26.56505118.这并不完全是一半，但45,这并不重要。只要它小于45，大于或等于22.5，就可以了。

继续下去，我们将得到的角度0.25,是，14.03624347这个值在和之间，22.511.25.如果我们继续这样下去，找到0.125、0.0625等的角度。我们现在可以大大简化操作，只需要加、减和移位。但由于角度是特殊的值，我们需要保留一个表格，记录我们在乘以0.5、0.25、0.125等时旋转的角度。通过加减表中的角度值，我们将知道原来的角度是什么。而这正是我们要找的东西。

AngTable = 45,26.565,14.036,7.125,3.576,1.790,0.895,0.448
和值角 = 0
循环数 = 0

如果Y是正数。

```
Xnew = X + (Y >> LoopNum)
Ynew = Y - (X >> LoopNum)
SumAngle = SumAngle + AngTable[LoopNum]
LoopNum = LoopNum + 1
```

如果Y是负数。

```
Xnew = X - (Y >> LoopNum)
Ynew = Y + (X >> LoopNum)
SumAngle = SumAngle - AngTable[LoopNum]
LoopNum = LoopNum + 1
```

以上是CORDIC方法。

让我们试试一个例子。

X = 200
Y = 100

' Y是正数
Xnew = + 200100>>　　0Xnew= 300
Ynew = -100200>　　0Ynew= -100
SumAngle = SumAngle +　　AngTable[0]SumAngle= 45

' Y是负数
Xnew = - 300　　(-100) >> 1Xnew= +300 = 50350
Ynew = -100 + 300>>　　1Ynew= -100 + = 15050
SumAngle = SumAngle -　　AngTable[1]SumAngle= 45 - = 26.56518.483

' Y是正数
Xnew = + 35050 >>　　2Xnew= +250 = 12362
Ynew = -50350>　　2Ynew=50 - 87= -37
SumAngle = SumAngle +　　AngTable[2]SumAngle= +18.483 = 14.03632.519

' Y是负数
Xnew = - 362(-37) >>　　3Xnew= +362 = 4366
Ynew = -37 + 362>>　　3Ynew= -37 + = 458
SumAngle = SumAngle -　　AngTable[3]SumAngle= 32.519-7.125 = 25.394

' Y是正数
Xnew = + 3668 >>　　4Xnew= +366 = 0366
Ynew = -8366>　　4Ynew= - 8= 22-14
SumAngle = SumAngle +　　AngTable[4]SumAngle= 25.394 + = 3.57628.97

'Y是负数
Xnew =366 - (-14) >>　　5Xnew= +366 = 0366
Ynew = -14 + 366>>　　5Ynew= -14 + = -311
SumAngle = SumAngle +　　AngTable[5]SumAngle= -28.97 = 1.79027.18

'Y是负数
Xnew = - 366(-3) >>　　6Xnew= +366 0 = 366
Ynew = -3 + 366>>　　6Ynew= -3 + 5 = 2
SumAngle = SumAngle -　　AngTable[6]SumAngle= 27.18-0.895 = 26.285

' Y是正数
Xnew = + 3662 >>　　7Xnew= +366 = 0366
Ynew =2 - 366 >>　　7Ynew=2 - 2 = 0
SumAngle = SumAngle +　　AngTable[7]SumAngle= 26.285 + = 0.44826.733

'说完了
' SumAngle = (26.73326.565是实际角度)
' X = 366，只要使用8个循环，任何一点都可以按0.60726的比例计算 = 222.25（223.6为实际斜边）

' Y = 0

使用CORDIC寻找Hypotenuse

为了将数学理论降到最低，我有点掩盖了我们如何从使用SIN/COS到使用TAN的过程。所以这里有一个更详细的解释。

由于TAN(角度)=SIN(角度)/COS(角度)，我们可以重写这些公式。

Xnew = X * COS(角度) - Y * SIN(角度)
Xnew = X * COS(angle) - Y * SIN(angle) * COS(angle) / COS**'**(angle) / **COS(angle) = 1**
Xnew = COS(angle) * [X - Y * SIN(angle) / COS(angle)] **因素'out * COS(angle)**
Xnew = COS(angle) * [X - Y * TAN(angle)] **用TAN(angle)代替'SIN(angle) / COS(angle)。**

在CORDIC中，我们只使用X - Y * TAN(angle)，而忽略了 "COS(angle)
*"部分。因此，这些值被缩放为1/COS(角度)。这样做的好处是，COS（角度）=COS（-
角度）。这意味着，不管我们是顺时针还是逆时针旋转，比例都是一样的。如果我们将CORDIC中要使用的每个角度的COS(角度)相乘，我们就可以用最终的X值乘以该值来获得正确的比例。

COS(45)*COS(26.565)*COS(14.036)*COS(7.125)*COS(3.576)*COS(1.790)*COS(0.895)*COS(0.448) = 0.60726

这个值被称为CORDIC "增益"。并取决于你使用多少个角度。对于角度来说8，它是 0.60726.

使用CORDIC来寻找SIN和COS

在前几页中，我们用cordic将一个点旋转一个未知的角度，使其达到零度。通过记录我们如何移动它，我们知道它开始时一定是什么样子。例如，如果我们必须旋转总共有
-如果要让点变成零度(Y=0)，那么这个点一开始就必须是在+53度。

在寻找SIN和COS时，我们有相反的情况。我们在零度（Y=0）处取一个点，然后把它旋转多少度。当我们这样做时，点的最终X和Y位置将等于该角度的SIN和COS。我们只能通过CORDIC表中的角度进行旋转（45，26.565，14.036，等等）。每个角度都要加或减以达到指定的角度。比方说，你想找到角度的SIN和COS 30.

我们从角度零开始，Y=0，X=起始值。如果当前的角度小于期望的角度，那么我们加上 CORDIC
角度。如果期望的角度小于当前的角度，那么我们就减去CORDIC的角度。

0+ 45 - 26.565 + -14.036 + 7.1253.576 + 1.790 - 0.895 + = 0.44830.265

只要我们选择正确的起始X值，最终的X和Y值将是该角度的SIN和COS。由于SIN和COS的返回值为1.0或更小，我们需要将这些值放大一些系数（否则我们只能得到1或0作为答案。基本上我们需要指定一个值，该值将被乘以SIN和COS。假设你想得到256*SIN（角度）和256*COS（角度）。你可以从Y=0和X=256 * "cordic gain "开始。我们在上一节中看到，对于8个角度，线型增益是0.60726。因此，我们将从Y=0和X=0开始进行SIN、COS编码过程。 155.

在单片机上使用CORDIC

大多数微控制器都没有浮点运算功能。而那些有的微控制器通常可以更快地进行整数数学运算。为了实现CORDIC的整数数学，我们通常会采用 "定点
"算术的概念。固定点有点像改变数值所代表的单位。例如，对于货币，我们可以说值 "1
"是一美元。但我们也可以说，代表便士（或一美元的1/100）的值 "100 "也是美元。1

我们对定点也是这样做的。通常情况下，标度将是2的幂，因为它们与微控制器配合得很好。对于像螺旋桨这样的32位核心，我们可以使用1/256度的单位。因此，45度将是45*256的数值，或者用11,520.这种方法，TAN角的数值是。

45.000=	11520
26.565=	6801
14.036=	3593
7.125=	1824
3.576=	916
1.790=	458
0.895=	229
0.448=	115
0.224=	57
0.112=	28
0.056=	14
0.028=	7
0.014=	4
0.007=	2
0.003=	1

你可以从表中看到，我们的答案将有大约0.003度的分辨率，因为那是我们可以旋转的最小角度。

CORDIC只对零到90的角度工作。如果角度不在这个范围内，则需要对该值进行一些预处理（可能还有后处理）。

下面几页是两个显示如何使用CORDIC方法的程序。它们是为Parallax Propeller处理器用spin编写的。

首先是一个计算X、Y坐标的ATAN2和Hypotenuse的程序。第二个是计算一个

角度的SIN和COS的程序。

计算ATAN2和HYP的螺旋桨CORDIC程序

```
'CORDIC演示程序，计算X,Y坐标的ATAN2和Hypotenuse。 '
'请注意，X、Y的起始值和角度、hy值都是1/256单位。
'角度的单位是1/256度。如果想要弧度或布拉德，只需将ATAN_Table的值进行缩放即可
'锥
心
之
痛
  _clkmode = xtal1 + pll16x
  _xinfreq = 5_000_000

OBJ
  Debug : "FullDuplexSerial"

VAR
  LONG angle,
  hypong LONG X,
  Y
  LONG Xnew, Ynew
  LONG i

PUB启动
  X := -76800 * ('300256值 * 256)
  Y := -102400 * ('400256Value * 256)

  Debug.Start(31, 30,0,115200)
  WaitCnt(clkfreq * +4 cnt) 等待'四秒，让用户开始PST。

  Debug.Str(string(16, "CORDIC Test",
  13))Debug.Str(string(13, "X = "))
  12月256(X)
  Debug.Str(string(13, "Y =
  "))Dec256(Y)
  Debug.Tx(13)

  'CORDIC程序的开始 角度 := 0
  如果X < 0
    角度 := * 180X256
    := -X
    y :=-y
  elseif y < 0
    角度 := * 360256
  REPEAT i FROM TO
    0IF14 Y < 0
      '逆时针旋转 Xnew := X - (Y
      ~> i)
      Ynew := Y + (X ~> i)
      angle := angle - ATAN_Table[i]
    ELSE
      ' 顺时针旋转 Xnew :=
      X + (Y ~> i) Ynew :=
      Y - (X ~> i)
      angle := angle + ATAN_Table[i]
    X := Xnew
    Y := Ynew
```

```
hyp := (X ~> 1) + (X ** 1B74_EDA9) hyp' := x * 0.607252935
'CORDIC程序结束

Debug.Str(string("Angle =
"))Dec256(angle)
Debug.Str(string(13, "Hyp = ")
)Dec256(Hyp)
Debug.Str(string(13, "Finished...", 13))
```

```
PUB Dec256(给定) | temp
  if given < 0
    given := -given
    Debug.Tx("-")
  temp := given ~> 8
  Debug.Dec(temp)
  temp := given &
  temp255 := temp *
  1000temp := temp /
  256 Debug.Tx(".)
  如果temp < 100
    Debug.Tx("0")
    如果temp < 10
      Debug.Tx("0")
  Debug.Dec(temp)
  RETURN


DAT
  ' ATAN_Table是ATAN(1/(2^i))的值。*256的度数(不是RADIANS) ATAN_Table LONG
  11520,6801,3593,1824,916,458,229,115,57,28,14,7,4,2,1
```

计算SIN和COS的螺旋桨CORDIC程序

```
'CORDIC 计算角度的SIN和COS的演示程序 '
'注意指定的角度以及SIN和COS值都是以1/256为单位。
'角度的单位是1/256度。如果想要弧度或布拉德，只需将ATAN_Table的值进行缩放即可
'锥
心
之
痛
  _clkmode = xtal1 + pll16x
  _xinfreq = 5_000_000

OBJ
  Debug : "FullDuplexSerial"

VAR
  LONG angle, desiredAngle
  LONG X, Y
  LONG Xnew, Ynew
  LONG i
  LONG cos, sin

PUB启动
  Debug.Start(31, 30,0,115200)
  WaitCnt(clkfreq * +4 cnt) 等待'两秒钟让用户开始PST Debug.Str(string("CORDIC Test", 13))


  desiredAngle := 30*256 角度' *. 256

  Debug.Str(string(13, "Angle = "))Dec256(期望的角度)
  Debug.Tx(13)

  'CORDIC程序的开始 角度 := 0
  Y := 0
  X := *155'256CORDIC增益

  IF desiredAngle > 90*256
    angle := 180*256
  如果期望的角度 > 270*256 角
    度 := 360*256

  重复i FROM TO 014
    IF desiredAngle > angle
      '逆时针旋转 Xnew := X - (Y
      ~> i)
      Ynew := Y + (X ~> i)
      angle := angle + ATAN_Table[i]
    ELSE
      ' 顺时针旋转 Xnew :=
      X + (Y ~> i) Ynew :=
      Y - (X ~> i)
      angle := angle - ATAN_Table[i]
    X := Xnew
    Y := Ynew
```

```
如果(desiredAngle > 90*256) AND (desiredAngle < 270*256)
  X := -X
  Y := -Y

cos := X
sin := Y
'CORDIC程序结束
```

```
  Debug.Str(string("Sin =
  "))Dec256(sin)
  Debug.Str(string(13, "Cos =
  "))Dec256(cos)
  Debug.Str(string(13, "Finished...", 13))

PUB Dec256(给定) | temp
  if given < 0
    given := -given
    Debug.Tx("-")
  temp := given ~> 8
  Debug.Dec(temp)
  temp := given &
  temp255 := temp *
  1000temp := temp /
  256 Debug.Tx(".)
  如果temp < 100
    Debug.Tx("0")
    如果temp < 10
      Debug.Tx("0")
  Debug.Dec(temp)
  RETURN


DAT
  ' ATAN_Table是ATAN(1/(2^i))的值。*256的度数(不是RADIANS) ATAN_Table LONG
  11520,6801,3593,1824,916,458,229,115,57,28,14,7,4,2,1
```