# AN1061

# Efficient Fixed-Point Trigonometry Using CORDIC Functions For PIC16F

*Author: Jose Benavides*
*Microchip Technology Inc.*

## INTRODUCTION

This application note presents an implementation of the following fixed-point math routines for the PIC16F microcontroller families:

- SIN(X), COS(X)
- ATAN(X)

CORDIC is an acronym for COordinate Rotation DIgital Computer and was first developed by Jack Volder in 1959. The CORDIC transforms are a collection of iterative, shift-add algorithms used to compute a wide range of trigonometric and hyperbolic functions on a digital computer.

With proper modification, these routines can also be used to implement the $\sin^{-1}$, $\cos^{-1}$, polar/rectangular coordinate conversion, hyperbolic, and even multiply/divide functions. More detail on these modifications can be found in a paper titled, "*A Survey of CORDIC Algorithms for FPGA-Based Computers*" by Ray Andraka.

The structure of the CORDIC transform lends itself to hardware implementations. Typical applications of the CORDIC transform include FPGA-based applications. In fact, entire Arithmetic Logic Units have been implemented based on the CORDIC transform. However, the software-based CORDIC algorithm presented in this application note will provide a sufficient performance improvement for most applications.

These fixed-point CORDIC math routines are considerably faster than other more traditional methods based on the Taylor expansion. This makes these routines ideal for real-time applications requiring very fast calculations. The SINCOS function, which simultaneously calculates the sine and cosine values of a given angle using the CORDIC transform, will typically take 370 µs to compute on a PIC16F microcontroller running at 20 MHz. This is in contrast to 1.9 ms using a sin(x) function call in C using the standard math.h include file. Both the SINCOS and ATAN functions take up 190 bytes of program memory and 11 bytes of data memory. Table 1 shows the CORDIC algorithm having over four times the efficiency as that of a standard C math function.

**TABLE 1: SIN(x) FUNCTION CALL SPECIFICATIONS WITH PIC16F877A AT 20 MHZ**

| | CORDIC in asm | Math.h in C |
|---|---|---|
| Time | 370 µs | 1.9 ms |
| Flash | 190 words | 1,117 words |
| RAM | 11 bytes | 40 bytes |

## CORDIC THEORY

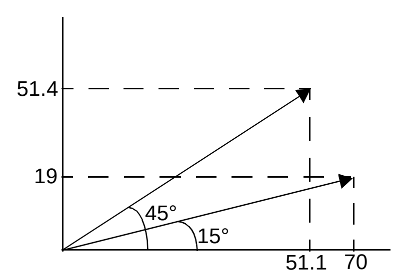
The CORDIC transform is based on the idea that all the trigonometric functions can be calculated using vector rotations. Equation 1 shows how to do vector rotations. Its derivation is presented in Appendix B.

**EQUATION 1: ROTATION OF VECTOR (X, Y) BY $\phi$**

$$\bar{x} = x\cos\phi - y\sin\phi$$
$$\bar{y} = y\cos\phi + x\sin\phi$$

Figure 1 shows an example of Equation 1 by rotating a vector (70, 19), by 30°.

**FIGURE 1: ROTATION OF VECTOR (70,19) BY 30°**




$\bar{x}$ = (70)cos(30°) - (19)sin(30°) = 51.12
$\bar{y}$ = (19)cos(30°) + (70)sin(30°) = 51.45

The CORDIC transform gives an iterative method for performing vectors rotations using only the shift and the add operations. The CORDIC transform is derived by starting with Equation 1 and re-writing it as Equation 2, remembering that tan($\phi$) = sin($\phi$)/cos($\phi$).

**EQUATION 2: ROTATION OF VECTOR (X, Y) BY $\phi$**

$$\bar{x} = \cos\phi\,[x - y\tan\phi]$$
$$\bar{y} = \cos\phi\,[y + x\tan\phi]$$

If the angle of rotation is restricted such that $\tan(\phi) = +/-2^{-i}$, then multiplication by $\tan(\phi)$ is equivalent to a shift operation. This is shown in Equation 3. The i represents the iteration number of the CORDIC transform.

At the first iteration, when i=0, the input vector is rotated by $\tan^{-1}(2^{-0}) = 45°$. The second iteration rotates by $\tan^{-1}(2^{-1}) = 26.56°$, then 14.03°, and so on.

**EQUATION 3: ROTATION OF VECTOR (X, Y) BY TAN$^{-1}$2$^{-i}$ DEGREES**

$$\bar{x} = \cos\left(\tan^{-1}\ 2^{-i}\right)[x - y\ 2^{-i}]$$
$$\bar{y} = \cos\left(\tan^{-1}\ 2^{-i}\right)[y + x\ 2^{-i}]$$

At each iteration i, a decision is made to rotate in the direction of the desired final angle. The angle of rotation becomes successively smaller at each iteration i until the vector converges to the desired angle. With enough iterations, one can rotate by any arbitrary angle. Equation 4 shows the iterative rotational transform.

**EQUATION 4: ITERATIVE ROTATIONS**

$$x_{i+1} = k_i\left[x_i - y_i \cdot d_i \cdot 2^{-i}\right]$$
$$y_{i+1} = k_i\left[y_i + x_i \cdot d_i \cdot 2^{-i}\right]$$
$$k_i = \cos\left(\tan^{-1}\ 2^{-i}\right) = \frac{1}{\sqrt{1 + 2^{-2i}}}$$
$$d_i = \pm 1$$

The d term is always +1 or -1 depending on the direction of rotation for that iteration. A d of +1 will rotate the vector counter clockwise, while a d of -1 will rotate the vector clockwise.

The $\cos[\tan^{-1}(2^{-i})]$ term from Equation 3 becomes a constant for each iteration which is now called $k_i$. Each iteration's $k_i$ term is independent of that iterations direction of rotation because cosine is an even function, $\cos(\phi) = \cos(-\phi)$. In fact, if the total number of iterations is fixed, the $k_i$ terms can be factored out entirely. The product of all the $k_i$ terms is represented as $K_n$, with n being the total number of iterations counting from 0. Equation 5 shows how to calculate $K_n$ by multiplying together all the $k_i$ terms from 0 to n. For a sufficiently large number of iterations n, (like 15) $K_n$ will converge to approximately 0.607253. The rotational algorithm calculated without the $k_i$ terms will have a total gain of $1/K_n$ or $A_n$.

**EQUATION 5: THE ROTATIONAL ALGORITHM GAIN**

$$K_n = \prod_{i=0}^{n} \frac{1}{\sqrt{1 + 2^{-2i}}} \qquad A_n = \frac{1}{K_n}$$

A third difference equation is added to track the composite angle of rotation. This is shown in Equation 6.

**EQUATION 6: TOTAL ANGLE OF ROTATION**

$$z_{i+1} = z_i - d_i\ \tan^{-1}\left(2^{-i}\right)$$

This iterative rotational transform is normally used in one of two modes, Rotational mode or Vectoring mode.

In Rotational mode, the input vector is rotated by a given input angle $z_0$. The sign of $z_i$ determines the direction of rotation at each iteration, such that its absolute value is diminished after each iteration. The full Rotational mode is presented in Equation 7. $x_n$ and $y_n$ represent the final values of x and y.

**EQUATION 7: ROTATIONAL MODE**

$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i}$$
$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$
$$z_{i+1} = z_i - d_i \cdot \tan^{-1} 2^{-i}$$
$$d_i = -1 \text{ if } z_i < 0, +1 \text{ otherwise}$$
$$A_n = \prod_{i=0}^{n} \sqrt{1 + 2^{-2i}}$$
$$x_n = A_n[x_0 \cos(z_0) - y_0 \sin(z_0)]$$
$$y_n = A_n[x_0 \cos(z_0) + y_0 \sin(z_0)]$$

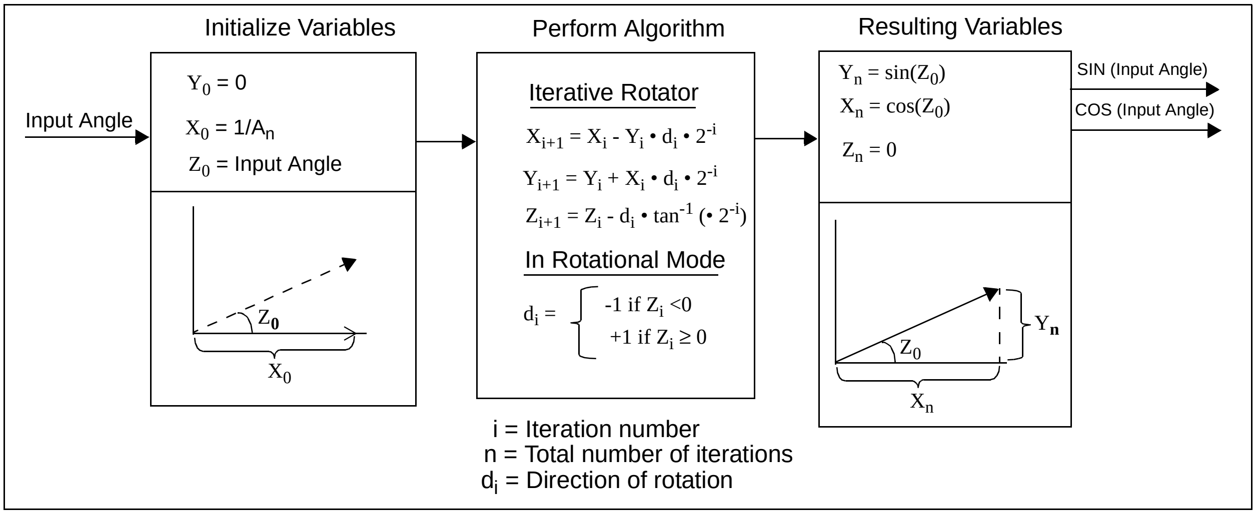
An example of its use is shown in Table 2 with the vector (70,19) being rotated by 30° is initialized to the desired angle. At each iteration, a rotational direction (d) is chosen that will minimize the angle accumulator z. This means d is equal to the sign of the previous z value. In other words, if the angle accumulator is positive, then the vector is rotated counter clockwise. If z is negative, the vector is rotated clockwise. Notice that the final values of x and y must be de-scaled by $A_n$ because the $K_i$ terms were left out of the algorithm.

**TABLE 2: FIRST ROTATIONAL MODE EXAMPLE**

| i | X | Y | Z | d | $\tan^{-1}(2^{-i})$ | $\tan^{-1}(y/x)$ |
|---|---|---|---|---|---|---|
| 0 | 70.00000 | 19.00000 | 30.00000 | 1 | 45.00000 | 15.19 |
| 1 | 51.00000 | 89.00000 | -15.00000 | -1 | 26.56505 | 60.19 |
| 2 | 95.50000 | 63.50000 | 11.56505 | 1 | 14.03624 | 33.62 |
| 3 | 79.62500 | 87.37500 | -2.47119 | -1 | 7.12502 | 47.66 |
| 4 | 90.54688 | 77.42188 | 4.65382 | 1 | 3.57633 | 40.53 |
| 5 | 85.70801 | 83.08105 | 1.07749 | 1 | 1.78991 | 44.11 |
| 6 | 83.11172 | 85.75943 | -0.71242 | -1 | 0.89517 | 45.90 |
| 7 | 84.45172 | 84.46081 | 0.18275 | 1 | 0.44761 | 45.00 |
| 8 | 83.79187 | 85.12059 | -0.26486 | -1 | 0.22381 | 45.45 |
| 9 | 84.12437 | 84.79328 | -0.04105 | -1 | 0.11191 | 45.23 |
| 10 | 84.28998 | 84.62897 | 0.07085 | 1 | 0.05595 | 45.11 |
| 11 | 84.20733 | 84.71129 | 0.01490 | 1 | 0.02798 | 45.17 |
| 12 | 84.16597 | 84.75240 | -0.01307 | -1 | 0.01399 | 45.20 |
| 13 | 84.18666 | 84.73185 | 0.00091 | 1 | 0.00699 | 45.18 |
| 14 | 84.17632 | 84.74213 | -0.00608 | -1 | 0.00350 | 45.19 |
| 15 | 84.18149 | 84.73699 | -0.00258 | -1 | 0.00175 | 45.19 |
| Descaled: | 84.1/An = 51.11 | 84.7/An = 51.45 | | | | |

The SINCOS function utilizes the Rotational mode to calculate the sine and cosine functions directly by initializing $y_0$ to zero and $x_0$ to the inverse of $A_n$. Figure 2 shows this in block form. By initializing $x_0$ to the inverse of $A_n$, the final values are correct without need for de-scaling.

**FIGURE 2: SINCOS FUNCTION**




Many applications for the sine and cosine functions use them to modulate a magnitude value. By initializing $x_0$ to that magnitude value divided by $A_n$, this routine will do that modulation without ever having to use a separate multiplier. Table 3 presents an example of modulating or scaling the value 3 by the sine of 30° and the cosine of 30°. $x_0$ is initialized to $3/A_n$= 1.8219. $x_n$ equals 3*cos(30°) ≈ 2.59821 and $y_n$ equals 3*sin(30°) ≈ 1.50023.

The example in Table 3 shows one way to use the SINCOS function. This is in contrast to the example in Table 2 that simply rotated the (70, 19) vector by 30°.

**TABLE 3: SECOND ROTATIONAL MODE EXAMPLE**

| i | X | Y | Z | d | atan(2^-i) | tan-1(y/x) |
|---|---|---|---|---|---|---|
| 0 | 1.82190 | 0.00000 | 30.00000 | 1 | 45.00000 | 0.00 |
| 1 | 1.82190 | 1.82190 | -15.00000 | -1 | 26.56505 | 45.00 |
| 2 | 2.73285 | 0.91095 | 11.56505 | 1 | 14.03624 | 18.43 |
| 3 | 2.50511 | 1.59416 | -2.47119 | -1 | 7.12502 | 32.47 |
| 4 | 2.70438 | 1.28102 | 4.65382 | 1 | 3.57633 | 25.35 |
| 5 | 2.62432 | 1.45005 | 1.07749 | 1 | 1.78991 | 28.92 |
| 6 | 2.57900 | 1.53206 | -0.71242 | -1 | 0.89517 | 30.71 |
| 7 | 2.60294 | 1.49176 | 0.18275 | 1 | 0.44761 | 29.82 |
| 8 | 2.59129 | 1.51210 | -0.26486 | -1 | 0.22381 | 30.26 |
| 9 | 2.59720 | 1.50197 | -0.04105 | -1 | 0.11191 | 30.04 |
| 10 | 2.60013 | 1.49690 | 0.07085 | 1 | 0.05595 | 29.93 |
| 11 | 2.59867 | 1.49944 | 0.01490 | 1 | 0.02798 | 29.99 |
| 12 | 2.59794 | 1.50071 | -0.01307 | -1 | 0.01399 | 30.01 |
| 13 | 2.59830 | 1.50007 | 0.00091 | 1 | 0.00699 | 30.00 |
| 14 | 2.59812 | 1.50039 | -0.00608 | -1 | 0.00350 | 30.01 |
| 15 | 2.59821 | 1.50023 | -0.00258 | -1 | 0.00175 | 30.00 |

In Vectoring mode, the input angle will be rotated by any angle necessary to align the vector to the x-axis such that the y component is zero. The direction of rotation for each iteration is determined by the sign of y such that its absolute value is diminished after each iteration until it's nearly zero. The resulting traversed angle is stored in z. The following difference equations describe Vectoring Mode.

**EQUATION 8: VECTORING MODE**

$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i}$$

$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$

$$z_{i+1} = z_i - d_i \cdot \tan^{-1} 2^{-i}$$

$$d_i = +1 \text{ if } y_i < 0, -1 \text{ otherwise}$$

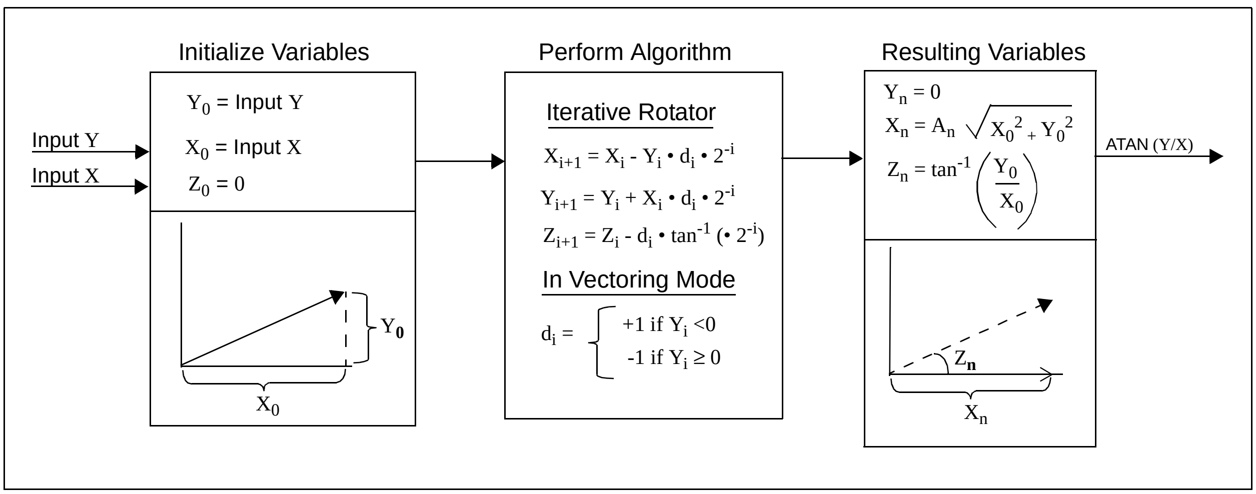
$$A_n = \prod_{i=0}^{n} \sqrt{1 + 2^{-2i}}$$

$$x_n = A_n \sqrt{x_0^2 + y_0^2}$$

$$y_n = 0 \quad z_n = z_0 + \tan^{-1}\left(\frac{y_0}{x_0}\right)$$

The arctangent function is directly computed using the Vectoring mode. The initial angle $z_0$ is set to zero. The arctangent function operates on the ratio of $y\phi/x\phi$. Figure 3 shows the ATAN function block diagram.

**FIGURE -3: ATAN FUNCTION**




It is important to note that the CORDIC algorithm as presented in this application note will only work for angles between +90° and -90°.

## EXCEL

The Excel workbook has four worksheets. The first worksheet lists the relevant difference equations for reference purposes. The second worksheet, COR_SIM, is the CORDIC transform simulated in both Rotational and Vectoring mode. It shows two examples. The first example is a sin/cos computation and the second is an $\tan^{-1}$ computation. The third worksheet, Cor_bitSIM_SINCOS, simulates the bit-for-bit CORDIC transform as it would be computed on a PIC® microcontroller. The bit manipulation functions were implemented in Visual Basic using Excel's Integrated Visual Basic Editor. This "bit-accurate" simulation allows for very detailed testing verification of the algorithm once it's on the PIC microcontroller. Also in the third worksheet is a plot of many possible inputs and outputs to show graphically its operation. The fourth worksheet, COR_bitSIM_ATAN, contains a similar layout for the Vectoring mode of the transform.

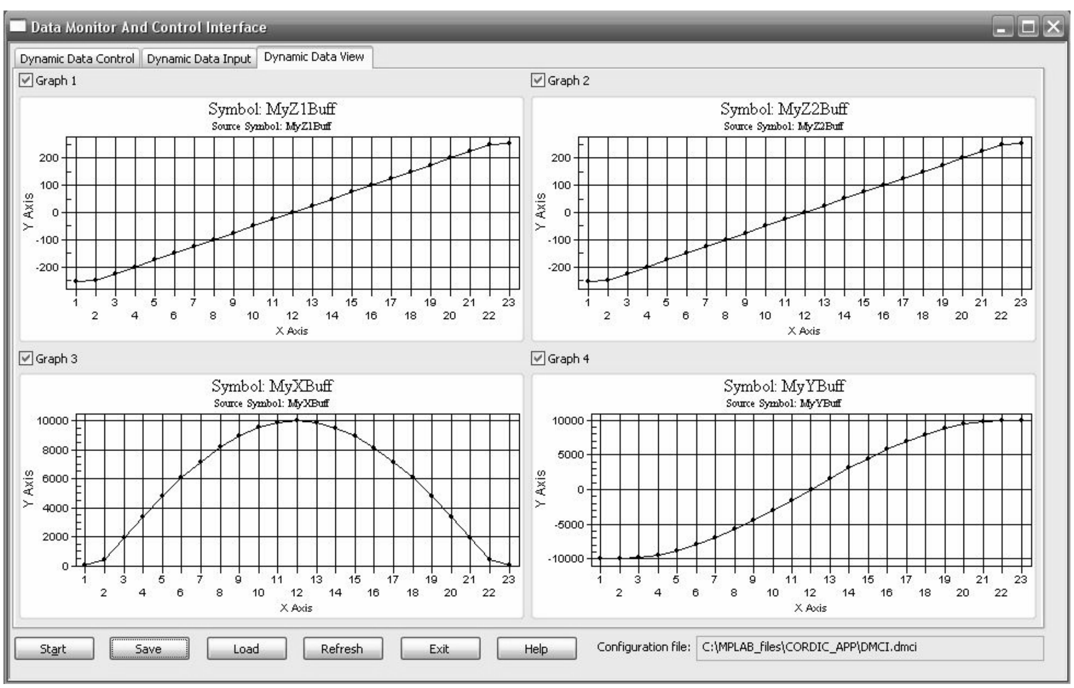
This workbook uses circular references with multiple iterations to implement the CORDIC algorithms. The operation of the algorithms can be better analyzed when the iterations are advanced one at a time manually. This is done by selecting “Options” under the tools menu and setting the max iterations under the “Calculation” tab to 1. Pressing the F9 key advances each iteration of the circular references.

## PIC16F IMPLEMENTATION

The included demo code was written for the PIC16F877A on the PICDEM™ 2 Plus Development Board. The potentiometer is used for angle input and the results are displayed on both the two line LCD and through the serial connector at 9600 baud. These results can be viewed on a PC using the hyper terminal. The main program will take the input angle Z1 and use the SINCOS function to calculate the SIN and COS values. It will then take those values and use the ATAN function to determine the output angle Z2. The display will show Z1 (the input angle in degrees), the calculated SIN and COS values (Y and X, respectively), and Z2 (the ATAN output angle in degrees).

Program operation can also be verified using the MPLAB® simulator and Data Monitor and Control Interface (DMCI) tool, which permits one to monitor arrays and buffers. The simulator stimulus files are included that will simulate a sinusoidal input. The demo code continuously records its output into RAM. The DMCI tool, when loaded with the included DMCI file, graphs the values recorded in RAM. A screen capture is shown below with Figure 4. When using the simulator, one must uncomment the “simulating” define statement and re-compile. This is so that the LCD routines don’t cause problems for the simulation.

**FIGURE 4: DMCI**



## References


- Andraka, Ray. A survey of CORDIC Algorithms for FPGA-based computers. Andraka Consulting Group, Inc. Copyright 1998.
- Crenshaw, Jack. Real Time Tool Kit for Embedded Systems. CMP Books. Copyright 2000. ISBN: 1-929629-09-5.
- Turkowski, Ken. Fixed-Point Trigonometry with CORDIC Iterations. Apple Computer. January 17, 1990.
- Testa, Frank J. AN575, AN617, and AN660.

## APPENDIX A: THE COS($TAN^{-1}$X) TRIG IDENTITY

**EQUATION A-1:**

$$\mathrm{COS}\left[\tan^{-1}(X)\right] = \frac{1}{\sqrt{X^2+1}}$$

$$\mathrm{COS}^2\left[\tan^{-1}(X)\right] = \frac{1}{X^2+1}$$

$$\frac{1}{2} + \frac{\mathrm{COS}\left[2 \bullet \tan^{-1}(X)\right]}{2} = \frac{1}{X^2+1}$$

$$\cos(2 \bullet \mu) = \frac{2}{X^2+1} - 1 \qquad (\text{set } \mu = \tan^{-1}(X))$$

$$\frac{1-\tan^2(\mu)}{1+\tan^2(\mu)} = \frac{2}{X^2+1} - 1$$

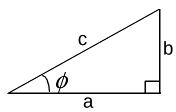
$$\frac{1-X^2}{1+X^2} = \frac{2}{X^2+1} - 1$$

$$1 - X^2 = 2 - (1 + X^2) = 1 - X^2$$

## APPENDIX B: ROTATION BY ANGLE $\phi$

In an effort to show where the rotational transform comes from, a derivation for Equation 1 is presented below. It is important to remember the following identities of a right triangle.

**FIGURE B-1: RIGHT ANGLE IDENTITIES**

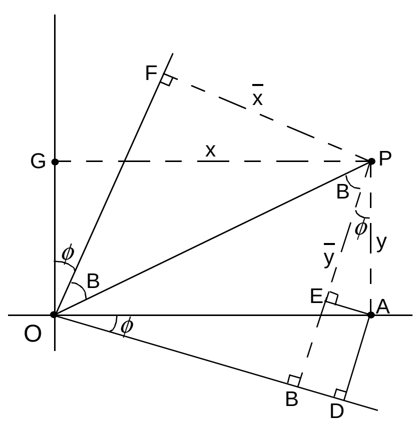



$$\cos(\phi) = \frac{a}{c} \Rightarrow C \cdot \cos(\phi) = a$$

$$\sin(\phi) = \frac{b}{c} \Rightarrow C \cdot \sin(\phi) = b$$

In a Cartesian plane, the vector P has coordinates (X, Y), and is shown in Figure B-2. The figure shows coordinate X is equal to line segment OA and coordinate Y is equal to line segment AP. Rotating the vector P counter clockwise by the angle $\phi$ is equivalent to rotating it's frame of reference by the same angle $\phi$ in the clockwise direction. This is shown in Figure B-2. The new coordinates of P under the new frame of reference are $\overline{X}$=0B and $\overline{Y}$=BP.

**FIGURE B-2: ROTATION OF VECTOR P BY ANGLE $\phi$**




Equation B-1 relates the old coordinates X and Y to the new coordinates $\overline{X}$ and $\overline{Y}$, effectively rotating the vector P by angle $\phi$. Equation B-1 can also be shown in matrix form as shown in Equation B-1.

**EQUATION B-1: DERIVATION OF THE ROTATIONAL TRANSFORM**

$$\begin{aligned}\overline{X} &= 0B = 0D - BD = 0D - EA \\ &= 0A \cdot \cos(\phi) - AP \cdot \sin(\phi) \\ &= X \cdot \cos(\phi) - Y \cdot \sin(\phi)\end{aligned}$$

$$\begin{aligned}\overline{Y} &= BP = BE + EP = DA + EP \\ &= 0A \cdot \sin(\phi) + AP \cdot \cos(\phi) \\ &= X \cdot \sin(\phi) + Y \cdot \cos(\phi)\end{aligned}$$

**EQUATION B-2: THE ROTATIONAL TRANSFORM**

$$\begin{bmatrix}\overline{X} \\ \overline{Y}\end{bmatrix} = \begin{bmatrix}\cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi)\end{bmatrix}\begin{bmatrix}X \\ Y\end{bmatrix}$$

**NOTES:**

**Note the following details of the code protection feature on Microchip devices:**

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.



**Trademarks**



**QUALITY MANAGEMENT SYSTEM**
**CERTIFIED BY DNV**
**ISO/TS 16949:2002**

*Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona, Gresham, Oregon and Mountain View, California. The Company's quality system processes and procedures are for its PIC® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.*

# Worldwide Sales and Service

## AMERICAS

**Corporate Office**
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://support.microchip.com
Web Address:
www.microchip.com

**Atlanta**
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

**Boston**
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

**Chicago**
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

**Dallas**
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

**Detroit**
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

**Kokomo**
Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

**Los Angeles**
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

**Santa Clara**
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

**Toronto**
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

## ASIA/PACIFIC

**Asia Pacific Office**
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Habour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

**Australia - Sydney**
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

**China - Beijing**
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

**China - Chengdu**
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

**China - Fuzhou**
Tel: 86-591-8750-3506
Fax: 86-591-8750-3521

**China - Hong Kong SAR**
Tel: 852-2401-1200
Fax: 852-2401-3431

**China - Qingdao**
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

**China - Shanghai**
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

**China - Shenyang**
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

**China - Shenzhen**
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

**China - Shunde**
Tel: 86-757-2839-5507
Fax: 86-757-2839-5571

**China - Wuhan**
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

**China - Xian**
Tel: 86-29-8833-7250
Fax: 86-29-8833-7256

## ASIA/PACIFIC

**India - Bangalore**
Tel: 91-80-4182-8400
Fax: 91-80-4182-8422

**India - New Delhi**
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

**India - Pune**
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

**Japan - Yokohama**
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

**Korea - Gumi**
Tel: 82-54-473-4301
Fax: 82-54-473-4302

**Korea - Seoul**
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

**Malaysia - Penang**
Tel: 60-4-646-8870
Fax: 60-4-646-5086

**Philippines - Manila**
Tel: 63-2-634-9065
Fax: 63-2-634-9069

**Singapore**
Tel: 65-6334-8870
Fax: 65-6334-8850

**Taiwan - Hsin Chu**
Tel: 886-3-572-9526
Fax: 886-3-572-6459

**Taiwan - Kaohsiung**
Tel: 886-7-536-4818
Fax: 886-7-536-4803

**Taiwan - Taipei**
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

**Thailand - Bangkok**
Tel: 66-2-694-1351
Fax: 66-2-694-1350

## EUROPE

**Austria - Wels**
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

**Denmark - Copenhagen**
Tel: 45-4450-2828
Fax: 45-4485-2829

**France - Paris**
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

**Germany - Munich**
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

**Italy - Milan**
Tel: 39-0331-742611
Fax: 39-0331-466781

**Netherlands - Drunen**
Tel: 31-416-690399
Fax: 31-416-690340

**Spain - Madrid**
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

**UK - Wokingham**
Tel: 44-118-921-5869
Fax: 44-118-921-5820

12/08/06