

MATH



## [THE BLAZINGCORE SERIES]

WITH SUPERIOR NUMBER CRUNCHING ABILITIES AND PERIPHERAL HANDLING ON OUR CUSTOM EMBEDDED OS,  
RAPID PROTOTYPING IS NOW EASY... AND BLAZING FAST.

## MATH

BCore supports various mathematical operations beyond the arithmetic operations listed in the Language Reference section. These mathematical operations are categorized into Integer Math (IMATH) and Floating Point Math (FMATH). Each category best suited for the operations its namespace suggests.

## IMATH

Provides access to methods for efficient **integer math** operations

Members	Description
Sqr	Returns the square root of a specified value
Within	Check if specified value is within 2 numbers
Trig	Trigonometry Methods

## SQR

Returns the square root of a specified value

```
Variable = IMath.Sqr(ByVal Value As Integer) As Integer
```

## WITHIN

Check if specified value is within 2 numbers (A,B) (Inclusive)

```
Variable = IMath.Within(ByVal Value As Integer, ByVal A As Integer, ByVal B As Integer) As Boolean
```

**A ≤ Value ≤ B**

Returns;

- **True**, if value is within A & B
- **False**, If value is not within A & B

## TRIG

Provides access to Trigonometry Methods

Members	Description
ASin2	Inverse Sine; $\sin^{-1}$
ACos2	Inverse Cosine; $\cos^{-1}$
ATan2	Inverse Tangent; $\tan^{-1}$
Sin	Sine
Cos	Cosine
Tan	Tangent

## .ASIN2

Inverse Sine;  $\sin^{-1}$

*Variable* = **IMath.Trig.ASin2**(ByVal X As Integer, ByVal Y As Integer) As Integer

## .ACOS2

Inverse Cosine;  $\cos^{-1}$

*Variable* = **IMath.Trig.ACos2**(ByVal X As Integer, ByVal Y As Integer) As Integer

## .ATAN2

Inverse Tangent;  $\tan^{-1}$

*Variable* = **IMath.Trig.ATan2**(ByVal X As Integer, ByVal Y As Integer) As Integer

Eg. If Value = 0.5

Represented in Fraction =  $\frac{1}{2}$

Nominator = 1

Denominator = 2

### Parameter/s:

- X ; Nominator
- Y ; Denominator

### Returns;

- Degrees to nearest whole number

## .SIN

Sine

*Variable* = **IMath.Trig.Sin**(ByVal Degree As Integer) As Integer

## .COS

Cosine

*Variable* = **IMath.Trig.Cos**(ByVal Degree As Integer) As Integer

## .TAN

Tangent

*Variable* = **IMath.Trig.Tan**(ByVal Degree As Integer) As Integer

## FMATH

Provides access to methods for efficient **floating point math** operations

Members	Description
<code>Sqr</code>	Returns the square root of a specified value

## SQR

Returns the square root of a specified value

`Variable = FMATH.Sqr (ByVal Value As Integer) As Single`

## G2D

Provides access to methods for 2D Geometric operations

Members	Description
Point	Consists of geometric 2D operations for points
Shape	Consists of geometric 2D operations for shapes

## POINT

Consists of geometric 2D operations for points

Members	Description
Length	Returns the length between 2 points
Inside	Check if Point is inside a specified shape

## .LENGTH

Returns the length between 2 points

```
Variable = G2D.Point.Length(ByRef P1 As Point, ByRef P2 As Point) As Integer
```

## .INSIDE

Check if Point is inside a specified shape

Members	Description
Rectangle	Check if Point is inside a Rectangle
Circle	Check if Point is inside a Circle

## .RECTANGLE

Check if point is inside a Rectangle

```
Variable = G2D.Point.Inside.Rectangle(ByRef PT As Point, ByRef PTL As Point, ByRef PBR As Point) As Boolean
```

### Parameter/s:

- **PT** ; Point To Check
- **PTL** ; Top Left Point of Rectangle
- **PBR** ; Bottom Right Point of Rectangle

### Returns;

- **True**, if Point is inside the Rectangle
- **False**, is Point is outside the Rectangle

## .CIRCLE

Check if point is inside a Circle

```
Variable = G2D.Point.Inside.Circle(ByRef PT As Point, ByRef PCC As Point, ByVal R As Integer) As Boolean
```

### Parameter/s:

- **PT** ; Point To Check
- **PCC** ; Centre Point of Circle
- **R** ; Radius of Circle

### Returns;

- **True**, if Point is inside the Circle
- **False**, is Point is outside the Circle

## SHAPE

Consists of geometric 2D operations for shapes

Members	Description
Overlap	Consists of geometric 2D operations for points

## .OVERLAP

Check if shapes are overlapping each other

Members	Description
Rectangles	Check if Rectangles are overlapping each other
Circles	Check if Circles are overlapping each other

## .RECTANGLES

Check if Rectangles are overlapping each other

`Variable = G2D.Shape.Overlap.Rectangles (ByRef PTL1 As Point, ByRef PBR1 As Point, ByRef PTL2 As Point, ByRef PBR As Point) As Boolean`

### Parameter/s:

- **PTL1** ; Top Left Point of Rectangle 1
- **PBR1** ; Bottom Right Point of Rectangle 1
- **PTL2** ; Top Left Point of Rectangle 2
- **PBR2** ; Bottom Right Point of Rectangle 2

### Returns;

- **True**, if the Rectangles are overlapping each other
- **False**, if the Rectangles are not overlapping each other

## .CIRCLES

Check if Circles are overlapping each other

`Variable = G2D.Shape.Overlap.Circles (ByRef PCC1 As Point, ByVal R1 As Integer, ByRef PCC2 As Point, ByVal R2 As Integer) As Boolean`

### Parameter/s:

- **PCC1** ; Centre Point of Circle 1
- **R1** ; Radius of Circle 1
- **PCC2** ; Centre Point of Circle 2
- **R 2**; Radius of Circle 2

### Returns;

- **True**, if the Circles are overlapping each other
- **False**, if the Circles are not overlapping each other

## LATEST DOCUMENTATION

All of our documentations are constantly updated to provide accurate and/or new information that we feel would help you with developing with our products.

The latest documentation may be obtained from our website: <http://www.aiscube.com/main/downloads.html>

## HOW YOU CAN HELP

You can help us to improve our documentations by emailing to us or posting a thread in our forum, reporting any mistakes/typos or errata that you might spot while reading our documentation.

Email: [TechSupport@aiscube.com](mailto:TechSupport@aiscube.com)

Forum: <http://forum.aiscube.com/index.php>

## DISCLAIMER

All information in this documentation is provided 'as-is' without any warranty of any kind.

The products produced by AIS Cube are meant for rapid prototyping and experimental usage; they are not intended nor designed for implementation in environments that constitute high risk activities.

AIS Cube shall assume no responsibility or liability for any indirect, specific, incidental or consequential damages arising out of the use of this documentation or product.

COPYRIGHT© 2009 - 2011 AIS CUBE. ALL RIGHTS RESERVED.

ALL PRODUCT AND CORPORATE NAMES APPEARING IN THIS DOCUMENTATION MAY OR MAY NOT BE REGISTERED TRADEMARKS OR COPYRIGHTS OF THEIR RESPECTIVE COMPANIES. AND ARE ONLY USED FOR IDENTIFICATION OR EXPLANATION FOR THE OWNER'S BENEFIT. WITH NO INTENT TO INFRINGE.

SONATA IDE AND BLAZINGCORE(BCORE) ARE TRADEMARKS OF AIS CUBE IN SINGAPORE AND/OR OTHER COUNTRIES. ALL IMAGES DEPICTING THE BLAZINGCORE OR ANY PART OF IT IS COPYRIGHTED.

ALL OTHER TRADEMARKS OR REGISTERED TRADEMARKS ARE THE PROPERTY OF THEIR RESPECTIVE OWNERS.