Cosworth Electronics Limited
Brookfield Technology Centre, Twentypence Road, Cottenham, Cambridge CB24 8PS, UK
Tel: +44 (0)1954 253600 Fax: +44 (0)1954 253601 www.cosworth.com
Registered in England No: 5276665. VAT Reg No: GB 927 1888 86.
Registered address: The Octagon, 44 St James Mill Road, Northampton, NN5 5RA

Advanced Maths User Guide

Author:	Neal Bateman
Email:	neal.bateman@cosworth.com
Date:	24/02/2017

This guide details how Advanced Maths works and the syntax involved. It also contains examples of the code that can be used.

Background

C#

C# is a Microsoft developed programming language which is used within Pi Toolbox to write Pi Advanced Math. C# is also the language used in the examples for basic functions.

Attributes

Functions and Parameter can have attributes associated with them. Advanced Math uses these attributes within Toolbox. Some of these attributes are compulsory others are optional.

See http://msdn2.microsoft.com/en-us/library/z0w1kczw.aspx for more information.

Basic Functions

Basic Functions are pre-compiled assembly files (libraries) which contain Function definitions.

Scripts

Scripts are C# files that are compiled when Toolbox is started or a build is started from the script Editor (F6).

Assemblies (Libraries)

A pre-compiled library can be added to Toolbox, this can contain either the basic functions or Channels/Events/Metric as created in the scripts.

Basic Editor

The editor in which a Channel/Event/Metric is created a function and it's parameters selected.

Script Editor

The Script Editor allows for the editing of script files and for the selection of the folders to look in for script files.

Basic Function

A basic function must contain an attribute that describes if it a Channel, Event or Metric Function, below is the simplest example of a channel function:

```
[ChannelFunction("A Simple Function", Quantity.UserType)]
```

```
public static Channel SimpleFunction ()
{
        Channel velocityChannel = Channel.GetChannel("A Channel",  Quantity.Velocity);

        return velocityChannel;
}
```

Class

All basic functions must be created within a class.

```
public class MyClass
{
        …
}
```

Parameters

The following parameters are permitted for functions, if any other parameter is present then the function will not be available.

bool
string
Scalar (requires a ScalarParameter attribute to be specified)
int
double
QuantifiedName (requires a ChannelParameter attribute to be specified)

Attributes

Attributes are used on methods and parameters, all methods must have an attribute.

Method Attributes
Channels

Definition
```
public ChannelFunctionAttribute(string name, Quantity quantity)
```

Useage
```
[ChannelFunction("A Channel Function", Quantity.UserType)]
```

Events

Definition
```
public EventSetFunctionAttribute(string name)
```

Useage
```
[EventSetFunction("An Event Function")]
```
Metrics

Two versions of the Metrics attribute exits
Definition
```
public MetricFunctionAttribute(string name, string description, Quantity quantity, Unit unit, int decimalPlaces)
```

Useage
```
[MetricFunction("A Metric Function", "Metric Description", Quantity.Acceleration, Unit.Mps_2, 2)]
```

Definition
public MetricFunctionAttribute(string name, string description, Quantity quantity, string userUnit, int decimalPlaces)

Useage
[MetricFunction("A Metric Function", "Metric Description", Quantity.Acceleration, ("My Units", 2)]

Parameter Attributes

Optional Parameters Attributes

These are attributes that are optional for all parameters, these attributes determine what is displayed to the user when they are entering values for the function parameters in the basic Editor.

1. Description

This is the description that appears in the Property grid when the parameter is selected, this is useful for providing information about the parameter. This attribute is defined in the System.ComponentModel namespace.

[Description("A Desription of the parameter that will appear in the Property grid.")]

1. Display Name

This is the name of the Parameter that is displayed in the property grid, if this attriubtes isn't defined then the actual name of the parameter will be used. This attribute is useful when you want to use spaces in the parameter name.

[DisplayName("Function Name")]

1. Category

All parameter will be displayed in the Property grid under a category, if no category is specified then "Misc" is used. This attribute is defined in the System.ComponentModel namespace.

[Category("My Category")]

1. Example

The following show how all of the optional pararmeters can be used with a string parameter.

```
[ChannelFunction("Optional Parameter Example", Quantity.UserType)]
public static Channel MyFunction(
        [Description("Description of the Parameter")]
        [ParameterDisplayName("Parameter Name")]
        [Category("Parameter Category")]
        string name)
{
        …
}
```

Additional Parameters Attributes

These are attributes that state what the parameter is, for example a ChannelParameterAttribute is used when QuantifiedName should be a list of known channels that are of a given quanitity.

Channel Parameter

The Channel parameter should be used when you want a list of all known channels that are of the defined Quantity, the input parameter be be a QuantifiedName.

Definition

```
public ChannelParameterAttribute(Quantity quantity)
```

Useage

```
[ChannelParameter(Quantity.UserType)]
```

Example

```
[ChannelFunction("A Channel Parameter Test", Quantity.UserType)]
public static Channel MyFunction (
        [ChannelParameter(Quantity.Velocity)]
        QuantifiedName name)
{
        …
}
```

Event Parameter

The Event parameter should be used when you want a list of all known Events. The input parameter must be a string.

Definition

```
public EventSetParameterAttribute()
```

Useage

```
[EventSetParameter]
```

Example

```
[ChannelFunction("An Event Parameter Test", Quantity.UserType)]
public static Channel MyFunction(
        [EventSetParameter] string eventToCopy)
{
        …
}
```

Scalar Parameter

The Scalar Parameter should be used when you want to select a Global or local Constant (a value can also be entered). The input parameter must be a Scalar.

Definition

```
public ScalarParameterAttribute(Quantity qty)
```

Useage

```
[ScalarParameter(Quantity.Time)]
```

Example

```
[ChannelFunction("A Scalar Parameter Test", Quantity.UserType)]
public static Channel MyFunction(
        [ScalarParameter(Quantity.Velocity)] Scalar myScalar)
{
        …
}
```

Examples

Channels

Some examples of Channel Functions.

A Channel function that takes no parameter inputs.

```
[ChannelFunction("A Simlple Function", Quantity.UserType)]
public static Channel SimpleFunction()
{
        Channel velocityChannel = Channel.GetChannel("A Channel",  Quantity.Velocity);

        return velocityChannel;
}
```

A Channel Function that takes a channel as a parameter

```
[ChannelFunction("A Velocity Function", Quantity.UserType)]
public static Channel VelocityFunction([ChannelParameter(Quantity.Velocity)]
        [Description("Will copy the velocity channel")]
        [ParameterDisplayName("Velocity Channel")]
        [Category("channels")]
        QuantifiedName name)
{
        Channel velocityChannel = Channel.GetChannel(name);

        return velocityChannel;
}
```

A Channel Function that takes two channels as parameters

```
[ChannelFunction("AveFrontSusTravel", Quantity.Length)]
public static Channel AveFrontSusTravel(
//This creates the UI in the function editor for an input channel called FLDamper
        [ChannelParameter(Quantity.Length)]
        [Description("Name of your Front Left Damper Channel")] //This name will appear in the description field
        at the bottom of the UI
        [ParameterDisplayName ("Front Left Damper")] //This will be the alias for the channel and will appear with
        a dropdown list filtered for legal channels
```

```
        [Category("Channels")] //This is the group that the dropdown will appear.
        QuantifiedName FLDamper, //This essentially creates an internal name for the channel

//This creates the UI in the function editor for an input channel called FRDamper
        [ChannelParameter(Quantity.Length)]
        [Description("Name of your Front Right Damper Channel")
        [ParameterDisplayName ("Front Right Damper")]
        [Category("Channels")]
        QuantifiedName FRDamper)
        {

//This section of code reads in the channel values in SI units and then calculates the output and returns the result
     // obtain the FL damper length
     Channel Chan_FLDamper = Channel.GetChannel(FLDamper);
     // obtain the FR damper length
     Channel Chan_FRDamper = Channel.GetChannel(FRDamper);
     // calculate the average front suspension travel
     Channel Chan_Susp_Travel_Front = (Chan_FLDamper - Chan_FRDamper) / 2;
     // return the average front suspension travel channel
     return Chan_Susp_Travel_Front;
      }
```

A Channel Function that takes a channel and event as a parameter

```
[ChannelFunction("An Angle Function with Event", Quantity.UserType)]
public static Channel AngleFunctionWithEvent(
        [ChannelParameter(Quantity.Angle)]
        [Description("Will copy the angle channel.")]
        [ParameterDisplayName("Angle Channel")]
        [Category("channels")]
        QuantifiedName name,
        [EventSetParameter]
        [Description("An event that does nothing")]
        [ParameterDisplayName("Event")]
        [Category("events")]
        string eventName)
{
        Channel AngleChannel = Channel.GetChannel(name);

        return AngleChannel;
}
```

A Channel Function that takes a channel and a constant as a parameter

```
[ChannelFunction("A Constant Test", Quantity.UserType)]
public static Channel TestOfConstants(
        [ChannelParameter(Quantity.Velocity)]
        [Description("Will copy the velocity channel")]
        [ParameterDisplayName ("Velocity Channel")]
        [Category("channels")]
        QuantifiedName name,
        [ScalarParameter(Quantity.Velocity)]
        [Description("Will create a Scaler from a constant")]
        [ParameterDisplayName ("Test Constant")]
        [Category("scalar")]
        Scalar myScaler)
{
        Channel AirDensity = Channel.GetChannel(name);
```

```
    Scalar RefAirDensity = myScaler * 5

    return    AirDensity / RefAirDensity;
}
```

A Channel Function that takes a string as a parameter.

```
[ChannelFunction("A Velocity Function 2", Quantity.UserType)]
public static Channel ManualVelocityFunction(
    [Description("Will copy the velocity channel")]
    [ParameterDisplayName ("Velocity Channel")]
    [Category("channels")]
    string name)
{
    Channel velocityChannel = Channel.GetChannel(name);

    return velocityChannel;
}
```

Events

Events take the same inputs as Channels and Metrics, the previous samples show how to use multiple parameters.

An Event Function that takes an Event as a parameter.

```
[EventSetFunction("An Event copy")]
public static EventSet MyEvent2(
    [EventSetParameter]
    [Description("Will copy the event.")]
    [ParameterDisplayName ("Event To Copy")]
    [Category("events")]
    string eventToCopy)
{
    EventSet gearShift = EventSet.GetEvents(eventToCopy);
    return gearShift;
}
```

Metric

Metrics take the same inputs as Channels and Events, the previous samples show how to use multiple parameters.

A Metric Function that takes a Channel as a parameter

```
[MetricFunction("Max Accel", "Metric to show maximum Aceeleration", Quantity.Acceleration, Unit.Mps_2, 2)]
public static Scalar maxaccel(
    [ChannelParameter(Quantity.Velocity)]
    [Description("Velocity channel to calculate the metric from")]
    [ParameterDisplayName("Velocity Channel")]
    [Category("channels")]
    QuantifiedName name)
{
```

```
        Channel speed = Channel.GetChannel(name, Unit.Mps);    Channel accel = PiMath.Derivative(speed,
        DerivativeMode.AllowPartial, true);

        return (Scalar)Statistics.Max(accel);
}
```