

# SoftMax® Pro

Data Acquisition and Analysis Software

Version 7.0

## Formula Reference Guide

This document is provided to customers who have purchased Molecular Devices equipment, software, reagents, and consumables to use in the operation of such Molecular Devices equipment, software, reagents, and consumables. This document is copyright protected and any reproduction of this document, in whole or any part, is strictly prohibited, except as Molecular Devices may authorize in writing.

Software that may be described in this document is furnished under a non-transferrable license. It is against the law to copy, modify, or distribute the software on any medium, except as specifically allowed in the license agreement. Furthermore, the license agreement may prohibit the software from being disassembled, reverse engineered, or decompiled for any purpose.

Portions of this document may make reference to other manufacturers and/or their products, which may contain parts whose names are registered as trademarks and/or function as trademarks of their respective owners. Any such usage is intended only to designate those manufacturers' products as supplied by Molecular Devices for incorporation into its equipment and does not imply any right and/or license to use or permit others to use such manufacturers' and/or their product names as trademarks. Each product is shipped with documentation stating specifications and other technical information. Molecular Devices products are warranted to meet the stated specifications. Molecular Devices makes no other warranties or representations express or implied, including but not limited to, the fitness of this product for any particular purpose and assumes no responsibility or contingent liability, including indirect or consequential damages, for any use to which the purchaser may put the equipment described herein, or for any adverse circumstances arising therefrom. The sole obligation of Molecular Devices and the customer's sole remedy are limited to repair or replacement of the product in the event that the product fails to do as warranted.



**For research use only. Not for use in diagnostic procedures.**

The trademarks mentioned herein are the property of Molecular Devices, LLC or their respective owners. These trademarks may not be used in any type of promotion or advertising without the prior written permission of Molecular Devices, LLC.

Patents: <http://www.moleculardevices.com/productpatents>

Product manufactured by Molecular Devices, LLC.  
1311 Orleans Drive, Sunnyvale, California, United States of America 94089.  
Molecular Devices, LLC is ISO 9001 registered.  
©2016 Molecular Devices, LLC.  
All rights reserved.

# Contents

|  |           |
|--|-----------|
| <b>Chapter 1: Introduction</b> .....                     | <b>5</b>  |
| Formula Categories .....                                 | 5         |
| Formula Size .....                                       | 8         |
| Formula Building Blocks .....                            | 9         |
| Lists of Numbers and Arrays of Numbers .....             | 9         |
| Formula Naming Conventions in SoftMax Pro Software ..... | 10        |
| Ordering Functions in Complex Formulas .....             | 11        |
| Rules to Follow When Writing Formulas .....              | 12        |
| Obtaining Support .....                                  | 13        |
| <b>Chapter 2: Operators</b> .....                        | <b>15</b> |
| Mathematical Operators .....                             | 15        |
| Mathematical Operator Examples .....                     | 17        |
| Comparison (Logical) Operators .....                     | 18        |
| Comparison Operator Example .....                        | 19        |
| Conditional (Boolean) Operators .....                    | 19        |
| Using Comparison and Conditional Operators .....         | 21        |
| Simple Conditional Formulas .....                        | 21        |
| Nested Conditionals .....                                | 22        |
| Conditional Operator Examples .....                      | 23        |
| Tips for Writing Conditional Formulas .....              | 25        |
| Concatenation Operators .....                            | 26        |
| Concatenating Numbers .....                              | 26        |
| Concatenation Operator Examples .....                    | 27        |
| <b>Chapter 3: Functions</b> .....                        | <b>29</b> |
| Mathematical Functions .....                             | 30        |
| Statistical Functions .....                              | 36        |
| Statistical Functions Example .....                      | 40        |
| Graph Functions .....                                    | 41        |
| Graph Functions Examples .....                           | 51        |
| Vmax Reduction Functions .....                           | 52        |
| Vmax Reduction Functions Examples .....                  | 55        |

|   |            |
|---|------------|
| Peak Pro Analysis Functions .....                               | 57         |
| Peak Pro Analysis Functions Examples .....                      | 60         |
| Time Functions .....  | 61         |
| Interpolation Functions .....                                   | 62         |
| Other Functions .....   | 63         |
| Other Functions Examples .....                                  | 67         |
| NANs and MakeErrs .....   | 69         |
| NANs Manipulation Functions .....                               | 71         |
| NANs and MakeErrs Examples .....                                | 71         |
| Text Functions .....  | 72         |
| Text Functions Example .....                                    | 75         |
| <b>Chapter 4: Accessors .....</b>                               | <b>77</b>  |
| Using !Well as an Accessor Prefix .....                         | 77         |
| Plate and Cuvette Set Setup Information Accessors .....         | 78         |
| Plate and Cuvette Set Setup Information Accessors Example ..... | 83         |
| Kinetic and Spectrum Data Accessors .....                       | 84         |
| Kinetic and Spectrum Data Accessors Examples .....              | 89         |
| Plate Data Accessors .....                                      | 91         |
| Plate Data Accessors Examples .....                             | 96         |
| Injector Data Accessors .....                                   | 98         |
| Injector Data Accessors Example .....                           | 100        |
| PathCheck Technology Accessors .....                            | 101        |
| PathCheck Technology Accessors Examples .....                   | 102        |
| Group and Well Information Accessors .....                      | 103        |
| Blank Accessors .....   | 105        |
| Blank Accessors Examples .....                                  | 106        |
| Imaging Data Accessors .....                                    | 107        |
| <b>Index .....</b>  | <b>111</b> |

## Chapter 1: Introduction

The *SoftMax® Pro Software Formula Reference Guide* identifies and enumerates all the details about formulas, accessors, and functions that you can use to create powerful data analysis templates, that when saved as Protocol files, can completely automate the analysis and results reporting for all your microplate reads.

This document is broken into four main sections:

- This Introduction provides a general introduction to formulas.
- [Operators on page 15](#) describes the different types of operators.
- [Functions on page 29](#) describes the different types of built-in functions that can be used to assemble formulas.
- [Accessors on page 77](#) describes the special functions that provide access to other specific information.

Before using this *Formula Reference Guide*, make sure that you understand the basics of SoftMax Pro Software. For more information on using the software, see the SoftMax Pro Software application help or user guide.

### Formula Categories

Formulas are defined in categories:

- Reduction formulas are used in **Plate** sections and **Cuvette Set** sections and apply a formula to the entire section. See [Reduction Formulas on page 6](#).
- Column formulas are used in **Group** sections and apply a formula to a single column in the section. See [Column Formulas on page 7](#).
- Summary formulas can be used in either **Group** sections or **Note** sections and are generally used to reduce data to a single value or array. See [Summary Formulas on page 8](#).

In a **Graph** section, formulas can be assigned to the X and Y axes.

## Reduction Formulas

Reduction formulas can be used in Plate sections and Cuvette Set sections. Reduction formulas apply only to the specific Plate section or Cuvette Set section for which they are written, and the formulas determine the values that are reported in the Values column of the Group sections. Unlike formulas in Group sections and Summaries, Reduction formulas must evaluate to a number, not text. Up to two types of formulas, that work in conjunction with each other, can be used to reduce plate or cuvette data, depending on the instrument and read mode you are using.

**Wavelength Combination:** This formula modifies raw data. It is available for all read modes with all instruments and is the only reduction available for Endpoint or Dual Read. These formulas act on all data from each well or cuvette in the Plate section or Cuvette Set section. Wavelength Combination is not available for all instruments. See the user guide for your specific instrument.

**Kinetic, Spectrum, or Well Scan Reduction:** These formulas use the data values after the Wavelength Combination and produce a single, reduced value for each well or cuvette.

### Entering a Custom Reduction Formula

If default reductions do not meet your needs, custom formulas can be created.

To enter a custom formula:

1. Click **Reduction** on the **Home** tab in the ribbon or in the toolbar at the top of the section to open the **Data Reduction** dialog.
2. Select **Custom** from one of the lists. The choices that are available in the **Data Reduction** dialog depend on the read mode and instrument type.
3. Click the **=f[x]** button to open the **Calculation** dialog. This dialog has only one field in which to create or edit a formula.
4. Type the formula in the field and click **OK**.
5. Click **OK** in the **Data Reduction** dialog to close that dialog and set the parameters.

For example, you can multiply the raw data by a constant such as:

$!Lm1*1.2$

With two-wavelength reads, a ratio analysis might be desired:

$(!Lm1 - !Lm2)/!Lm2$

where Lm1 and Lm2 represent raw data (OD, RFU, RLU) for the two wavelengths respectively.

## Column Formulas

Column formulas act on either raw or reduced numbers from Plate section or Cuvette Set section and evaluate to a list of numbers or text. Column formulas are used in Group sections and contain a name that is used as the header of the column (such as, its name in the Group section) and a formula. Column formulas can be referenced by name in other formulas.

Group sections sort information by the sample names assigned in the Template Editor. Data comes into the Group section in “template order” (the order assigned in the template) rather than in “well order” (the order in which it was acquired from the instrument). For example, if wells A1, A2, B1, B2, C1, and C2 are assigned to a single group named Samples, data from those wells is displayed in a Group section named Samples and is sorted by the sample name.

The numbers in the Values column of the Group section are dictated by the Reduction formula in the Plate section. The formula for the Values column is **!Wellvalues**.



**Note:** **Values** is the default name, but it can easily be changed.

---

When writing custom formulas to access raw data in the Plate section, it is necessary to include **Well** so that the values are sorted by template order. Examples include **!WellLm1**, **!WellLm2**, and **!WellPathlength**.

Column formulas evaluate an entire column on a row-by-row basis. When writing column formulas to access data from two or more Group sections, the Group sections must have the same number of rows in them. If Group section A has 3 rows and Group section B has 4 rows, a formula in Group section B that refers to a column in Group section A is evaluated only for the first 3 rows, because only 3 row-by-row comparisons can be made.

In some cases, you might not want the data in a column of a Group section sorted by template order. In the case of a Spectrum scan, you might want the wavelength values in one column and OD values for a single well in a different column. The corresponding formulas for well B1 in Plate#X is **!Wavelengthrun@plate#X** and **!B1Lm1@Plate#X**.

## Summary Formulas

Summaries consist of four parts:

- A Summary name (which is used to refer to the Summary in other formulas).
- A description (optional).
- The formula itself.
- A specification for the number of decimal places to which the formula results should be displayed.

Summaries can be used in Group sections and Note sections, and evaluate to either a single number or a list of numbers. Summaries report information from Plate sections, Cuvette Set sections, Graph sections, Group sections, or to a combination of these.

Summaries can be edited by double-clicking them or by highlighting them and then clicking the **=f(x)** button in the tool bar to open the **Calculation** dialog.

## Formula Size

While there are no practical restrictions on formula length, Molecular Devices recommends that you break up complex formulas into several simpler formulas, and use short object names such as Experiment name, Section names, and Group names for easier handling and error checking.

To do this, you can create two or three columns, or a combination of columns and summaries, each containing a portion of the calculation, rather than one column with an extremely complex formula. To simplify the view of the Group section, you can hide the columns that contain intermediate steps.

As an alternative, you can use Summaries to contain intermediate steps. If a formula becomes lengthy, you can break it into several smaller formulas, and you can shorten all the object names.



## Formula Building Blocks

SoftMax Pro Software uses four types of formula building blocks:

- Operators let you combine other building blocks together. See [Operators on page 15](#).
- Functions are mathematical or text items that do operations within (for example, standard deviation or slope). See [Functions on page 29](#).
- NANs (Not A Number) are text items that are considered to be numbers by SoftMax Pro Software. See [Functions on page 29](#).
- Accessors let you access data and other information from objects (Plates, Cuvette-Sets, and so on) and functions (Vmax Rate, the wavelength at which a Spectrum scan was begun). Accessors are unique to SoftMax Pro Software. See [Accessors on page 77](#).

Some formula building blocks can be used alone. Others must always be used in conjunction with other building blocks. A complete listing of these building blocks, along with examples showing how to use them, can be found in the sections referenced above.

## Lists of Numbers and Arrays of Numbers

SoftMax Pro Software can do calculations on lists of numbers and arrays (lists of lists) of numbers. An example of a list of numbers is a single column in a Group section or a single set of Endpoint values. The column is also a  $1 \times N$  array, where N is the number of rows in the column or values in the set of endpoints.

An example of an array of numbers is a column containing the optical densities at each time point in a Kinetic run: it is a  $Y \times X$  array, where Y is the number of ODs in each row and X is the number of samples in the group that was assayed, and is also the number of rows in the Group section.

## Formula Naming Conventions in SoftMax Pro Software

SoftMax Pro Software uses a hierarchical naming structure, much like the directories and subdirectories used in computer file systems. All objects (for example, columns, Summaries, wells, graphs, plots) in the SoftMax Pro Software have names and, when writing formulas, the names are used to refer to the objects.

The full name of an object consists of the name of the object plus the name of the section the object is in, plus the name of the experiment in which the section is located. This path structure is known as the "scope" of the formula. The SoftMax Pro Software uses the @ symbol to combine these references:

- ColumnName@GroupSectionName@ExperimentName
- PlotName@GraphName@ExperimentName

So, columns in two different Group sections can have the same name, but the SoftMax Pro Software differentiates between them because their full names, including the scope, are different:

- OD@GroupSectionName1@Experiment1
- OD@GroupSectionName2@Experiment1
- OD@GroupSectionName1@Experiment2

When a formula refers to the data in the same section as the formula, the software removes extraneous scope information from the formula to create a relative path to the data.

For example, if the formula **!Lm1@Plate1** is in the section named **Plate1**, then the scope is removed from the formula and simplified to **!Lm1** so that it will always refer to itself.

However, if the formula **!Lm1@Plate2** is in the section named **Plate1**, then the scope is retained as **!Lm1@Plate2** so that it will always refer to the section named **Plate2**.

## Referencing Objects in Different Sections

If the object is not in the same section as the formula and you do not reference the object's section or experiment name, the SoftMax Pro Software uses the first object of that name that it finds in the file.

For example, suppose Group sections 1 and 2 contain a column called Values, Group section 3 does not contain a column of that name, and you are writing a formula in Group section 3 that subtracts Group section 2's Values column from a column named "ColX" in Group section 3.

If you write:

```
ColX – Values
```

SoftMax Pro Software finds the first occurrence of an object named "Values" in the file and fills in:

```
ColX – Values@GroupSectionName1
```

That is, it subtracts the Values column from Group section 1 from ColX.

To write the formula to give the desired result, you must specify the section in which the desired Values column is found:

```
ColX – Values@GroupSectionName2
```

## Ordering Functions in Complex Formulas

Formulas can be very simple. For example, Average(Values) takes the average of a Group section column named Values and returns a single value. Formulas can also be quite complex, combining several functions or accessors together. These more complicated formulas are called nested formulas because parentheses are used to nest different functions within each other.

The order in which the functions are nested determines the order in which they will be processed. The SoftMax Pro Software processes the information contained within parentheses before the information outside the parentheses. If multiple sets of parentheses are present, information is processed from inside to outside, starting with the innermost set of parentheses. An example of a simple nested formula is:

```
Sqrt(Average(Values))
```

This formula first calculates the average of the column named "Values" and then takes the square root of that average.

## Rules to Follow When Writing Formulas

### Case

SoftMax Pro Software formulas are not case sensitive: “Average”, “average”, and “AVERAGE” are treated the same and return the same values.

### Special Characters

If an object’s name contains a space, starts with a number, or includes one or more of the following special characters: # \$ % / \* ? , the name must be enclosed in single quotation marks. For example, ‘OD Values’.

Do not use ^ @ ~ or & in object names because these characters have special meanings in SoftMax Pro Software. Using these characters in formulas can cause errors when the objects are referred to in other formulas.

### Referencing Columns

When writing a formula that references columns in more than one Group section, make sure the Group sections all have the same number of rows (samples).

For example, if one Group section has 10 rows and a different Group section has 12 rows, and a formula is multiplying values from each Group section row by row, the output is undefined for rows 11 and 12.

### Evaluating Column Formulas

Column formulas must evaluate either to text, to numbers, or to a Boolean (for example, to True or False) because the SoftMax Pro Software does not let a column contain text (“Pass”) and also numeric (“1.002”) results. The resulting type mismatch renders an error of **The operand is of the wrong type.**

However, you can use NANs to put text into a column of numbers, because the SoftMax Pro Software treats NANs as numbers. You can also report numbers as text strings.

### Strings in Formulas

When a formula includes a result that you want displayed, these text results must be enclosed in double quotation marks, for example: “Out of Range”.

Wavelength reduction formulas in Plate or Cuvette Set sections must evaluate to numbers. However, NANs are considered numbers and can be used to bypass this restriction.

## Operators, Functions, and Accessors

When you name a column that you plan to reference in a formula, make sure that you do not use reserved characters or names of operators, functions, or accessors in the name of the column.

These names are referenced in the following sections:

- [Operators on page 15](#)
- [Functions on page 29](#)
- [Accessors on page 77](#)

## Obtaining Support

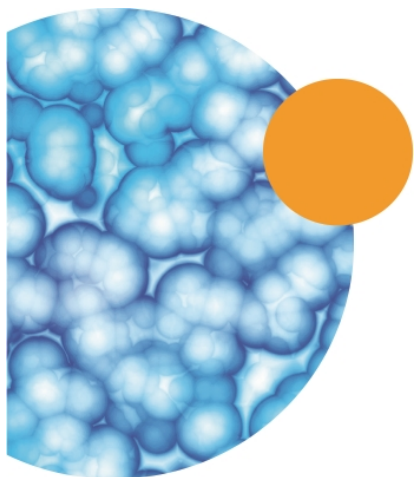
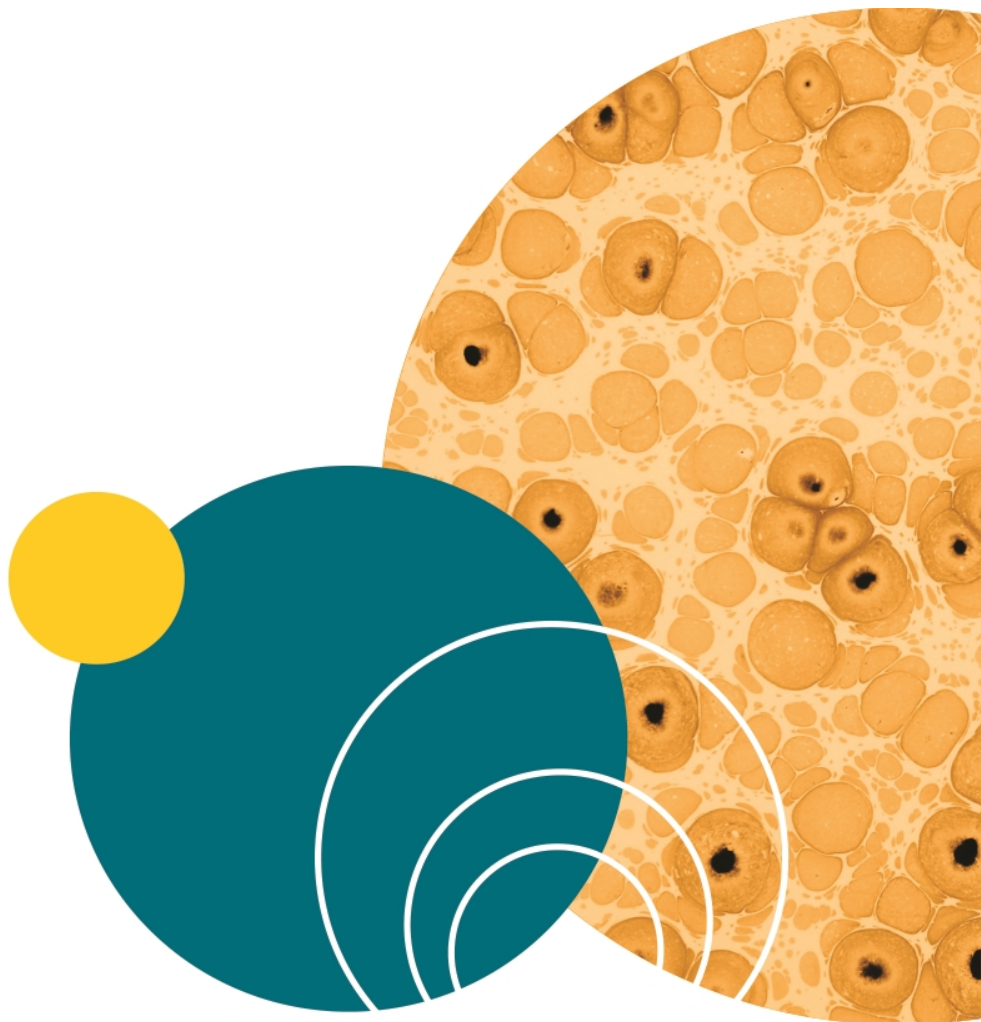
Molecular Devices is a leading worldwide manufacturer and distributor of analytical instrumentation, software, and reagents. We are committed to the quality of our products and to fully supporting our customers with the highest possible level of technical service.

Our support web site, [www.moleculardevices.com/support](http://www.moleculardevices.com/support), has a link to the Knowledge Base with technical notes, software upgrades, safety data sheets, and other resources. If you do not find the answers you are seeking, follow the links to the Technical Support Service Request Form to send an email message to a pool of technical support representatives.

You can contact your local representative or contact Molecular Devices Technical Support by telephone at 800-635-5577 (North America only) or +1 408-747-1700. In Europe call +44 (0) 118 944 8000.

To find regional support contact information, visit [www.moleculardevices.com/contact](http://www.moleculardevices.com/contact).

Please have your instrument serial number or Work Order number, and your software version number available when you call.



## Chapter 2: Operators

This section describes the different types of operators that can be used to assemble formulas in SoftMax® Pro Software. Operators let you combine and compare numbers, functions, accessors, and text strings in a variety of ways within formulas.

Formulas that start with !Lm1, !Lm2, and so on, are suitable for Reduction formulas in Plate sections or Cuvette Set sections. In Group sections, the analogous formulas must start with !WellLm1, !WellLm2, and so on, to ensure sorting based on the Template. To access data from single wells, the accessors are !a1Lm1, !a2Lm1, and so on in Reduction formulas, and !a1Lm1@Plate#X, !a2Lm1@Plate#X, and so on in Column or Summary formulas.

SoftMax Pro Software uses the following types of operators:

- [Mathematical Operators on page 15](#)
- [Comparison \(Logical\) Operators on page 18](#)
- [Conditional \(Boolean\) Operators on page 19](#)
- [Concatenation Operators on page 26](#)

### Mathematical Operators

SoftMax Pro Software uses the standard mathematical order for operators. Multiplication and division are done before addition and subtraction, regardless of their order in the equation, unless you use parentheses to order the operations.

SoftMax Pro Software supports the following mathematical operators.

#### \* for Multiplication

Type \* for multiplication.

Multiplication returns the product of 2 numbers, 2 lists of numbers, or 2 arrays of numbers. You can also multiply a list or array of numbers by a constant.

!Lm1 \* 10

#### / for Division

Type / for division.

Division returns the quotient of 2 numbers, 2 lists of numbers, or 2 arrays of numbers. You can also divide a list or array of numbers by a constant.

!WellLm1 / !WellLm2

### + for Addition

Type + for addition.

Addition returns the sum of 2 numbers, 2 lists of numbers, or 2 arrays of numbers. You can also add a constant to a list or array of numbers.

$$!Wellm1 + !Wellm2$$

### - for Subtraction

Type - for subtraction.

Subtraction returns the difference between 2 numbers, 2 lists of numbers, or 2 arrays of numbers. You can also subtract a constant from a list or array of numbers.

$$!Lm1 - !A12Lm1$$

### ^ for Exponent

Type ^ for exponent.

Exponent raises a number to a power. You can take the exponent of individual numbers, lists of numbers, or arrays of numbers.

$$!Wellm1 ^ 2$$

### Mod for Remainder

Type **Mod** for remainder (Modulo).

Mod returns the remainder from dividing two numbers. You can take the remainder for individual numbers, a list of numbers, or an array of numbers.

$$11 \text{ mod } 3$$

### ( ) for Parentheses

Type ( and ) to place parentheses around an operation.

Parentheses are used to order multiple mathematical operations. Operations in inner parentheses are done first.

$$(!Lm1 + !A12Lm1) * !H1Lm1$$



## Mathematical Operator Examples

The following examples show different ways in which you can use mathematical operators. The first example shows how to multiply two lists of numbers in a Group section. The second example shows how to multiply optical densities in a Plate section by a constant.

### Example: Multiplying Lists of Numbers

When multiplying lists of numbers in you can refer to them by name, or you can use functions (see [Functions on page 29](#)) or accessors (see [Accessors on page 77](#)) to refer to them. For example, suppose you have a column named “AdjResults” that contains the average of the result for each sample multiplied by the dilution factor, generated by the formula “MeanResult \* Factor”. MeanResult is a different column that is itself generated by the formula “Average(Results)”.

The formula for AdjResults can be written in one of the following ways:

```
MeanResult*Factor
MeanResult*Dilution
MeanResult*!SampleDescriptor
Average(Results)*!SampleDescriptor
Average(Results)*Dilution
```

### Example: Multiplying Optical Densities by a Constant

DNA samples were read in a SpectraMax® Microplate Reader at 260 nm with the PathCheck® Pathlength Measurement Technology option selected. The optical density values were multiplied in the Group section by a constant to report the concentration of DNA in each well.

The formula for the reduced number is:

```
!WellLm1*50
```

## Comparison (Logical) Operators

Comparison operators are used to compare numbers, lists of numbers, arrays of numbers, or text strings. They are used most frequently in conditional statements. See [Conditional \(Boolean\) Operators on page 19](#). When used by themselves, comparison operators return “True” if the argument is true, and “False” if the argument is false.

SoftMax Pro Software supports the following Comparison operators.

### = for Equals

Type = for equals.

Equals determines if two numbers, lists of numbers, or arrays of numbers are equal to each other.

```
!Lm1 = !Lm1 is True
!Lm1 = !A12Lm1 is False
```

### > for Greater Than

Type > for Greater Than.

Greater Than determines if a number, list of numbers, or array of numbers is greater than a different number, list of numbers, or array of numbers.

```
(2*!Lm1) > !Lm1 is True
!Lm1 > !Lm1 is False
```

### >= for Greater Than or Equals

Type >= for Greater Than or Equals.

Greater Than or Equals determines if a number, list of numbers, or array of numbers is greater than or equal to a different number, list of numbers or array of numbers.

```
Lm1 >= !Lm1 is True
(2*!Lm1) >= !Lm1 is True
!Lm1 >= (2*!Lm1) is False
```

### < for Less Than

Type < for Less Than.

Less Than determines if a number, list of numbers or array of numbers is less than a different number, list of numbers or array of numbers.

```
Lm1 < (2*!Lm1) is True
(2*!Lm1) < !Lm1 is False
```

## <= for Less Than or Equals

Type <= for Less Than or Equals.

Less Than or Equals determines if a number, list of numbers, or array of numbers is less than or equal to a different number, list of numbers, or array of numbers.

`!Lm1 <= !Lm1` is True

`(2*!Lm1) <= !Lm1` is False

`!Lm1 <= (2*!Lm1)` is True

## <> for Does Not Equal

Type <> for Does Not Equal.

Does Not Equal determines if two numbers, lists of numbers, or arrays of numbers are not equal to each other.

`!Lm1 <> !Lm1` is False

`(2*!Lm1) <> !Lm1` is True

## Comparison Operator Example

The following example shows how you can use a Comparison operator to report the status of data.

### Example: Reporting Whether Data Reaches a Threshold Value

Create a column with the formula:

`MeanValue > 0.2`

This comparison formula evaluates to True if an optical density (for example, the MeanValue) is greater than 0.2 and False if it is not.

## Conditional (Boolean) Operators

Conditional operators let you compare results in Plate sections, Cuvette Set sections, Group sections, and Summaries. They can be used to set criteria for evaluating numerical results. Group sections and Summaries can evaluate either to a number or to text, while Plate sections and Cuvette Set sections can evaluate only to a number.

For example, an ELISA kit might want you to determine whether a sample is positive, negative, or requires re-testing based on the values of the assay controls.

SoftMax Pro Software supports the following Conditional operators.

## If

Type **If** to compare numbers, lists of numbers, or arrays of numbers to each other.

If (Condition, Value-If-True, Value-If-False)

If (!WellLm1 > !A12Lm1@Plate#X, "Above threshold", "Below threshold")

For this example: If the value at wavelength 1 in a well is greater than the value in well A12 of Plate#X, return "Above threshold", otherwise return "Below threshold".

## And

Type **And** to compare numbers, lists of numbers, or arrays of numbers.

Returns True if all of the arguments are true.

Returns False if one or more of the arguments are not true.

WellLm1 > !A12Lm1@Plate#X And

!WellLm2 > !A12Lm2@Plate#X

## Or

Type **Or** to compare numbers, lists of numbers, or arrays of numbers.

Returns True if one or more of the arguments are true.

Returns False only if all of the arguments are false.

WellLm1 > !A12Lm1@Plate#X Or

!WellLm2 > !A12Lm2@Plate#X

## Not

Type **Not** to reverse the logic of an argument.

If something is not true, it is false; if it is not false, it is true.

Not(!WellLm1 > !A12Lm1@Plate#X And !WellLm2 > !A12Lm2@Plate#X)

False Returns the logical value False.

True Returns the logical value True.

## False

Type **False** to return the logical value False.

## True

Type **True** to return the logical value True.

## Using Comparison and Conditional Operators

Conditional formulas use Boolean logic, which treats various classes of data as algebraic quantities. The use of Boolean (conditional) and logical (comparison) operators in formulas lets you combine several simple mathematical operations into a single, more complex operation or function. The SoftMax Pro Software also lets you nest (embed) conditional statements within other conditional statements, permitting multiple conditions and outcomes in a single formula.

### Simple Conditional Formulas

Conditional formulas have the general form:

If (Condition, Value-If-True, Value-If-False)

The following table shows several examples of relatively simple conditional formulas.

**Table 2-1: Conditional Formulas**

| Conditional Formula   | Condition   | Value-If-True | Value-If-False            |
|---|---|---------------|---------------------------|
| If (!WellLm1 > 0.5, "Pass", "Fail")   | !WellLm1 > 0.5  | "Pass"        | "Fail"                    |
| If (!WellLm1 > 0.5 And !WellLm2 > 0.4, "Pass", "Fail")  | !WellLm1 > 0.5 And<br>!WellLm2 > 0.4  | "Pass"        | "Fail"                    |
| If (MeanValue > Summary#1@Control<br>and<br>MeanValue < Summary#2@Control,<br>"Acceptable", "Out of Specification") | MeanValue ><br>Summary#1@Control<br>and<br>MeanValue <<br>Summary#2@Control | "Acceptable"  | "Out of<br>Specification" |

## Nested Conditionals

Conditional statements can also be written to have multiple conditions and multiple outcomes. A conditional statement always has one outcome more than the number of conditions.

For example, a conditional statement with two conditions has three outcomes:

If (Condition#1, Outcome if Condition#1 is True, (if(Condition#2, Outcome if Condition#2 is True, Outcome if Condition#2 is False))

In this formula, if Condition#1 is true, an outcome is returned, but if Condition#1 is false, Condition#2 is evaluated. If Condition#2 is true, a second outcome is returned; if it is false, a third outcome is returned.

You can create very complicated formulas by nesting multiple conditional statements that must specify an outcome if true and an outcome if false for each condition.

The formula must state a condition and the outcome if true, then state a new nested condition if the previous condition was false and the outcome if the second condition is true, and so on until the last condition, with final outcomes for if the last condition is true or false.

The general format is:

If(Condition#1, outcome when Condition#1 is True, (if(Condition#2, outcome when Condition#2 is True, (if(Condition#3, outcome if Condition#3 is True... (if Condition#X, outcome if Condition#X is True, outcome if Condition#X is False))))...).

An outcome in this type of conditional statement can be conditional, with its own True/False outcome embedded in the Value-If-True outcome of the first condition.

## Conditional Operator Examples

The following examples show how you can use Conditional operators. The first example shows a simple conditional statement, and the second example shows a nested conditional statement.

### Example: A Conditional Formula with One Condition and Two Outcomes

Below is an example of a simple conditional statement, written as a Summary formula for a Group section. This conditional statement has one condition with two outcomes:

If the minimum number in a column named Values is greater than 2, the formula reports “Acceptable”.

If the minimum number is equal to or less than 2, the formula reports “Out of Specification”.

#### Formula

```
If(Min(Values)>2,“Acceptable”,“Out Of Specification”)
```

#### Breakdown

Condition: Min(Values)>2

Value-If-True: “Acceptable”

Value-If-False: “Out of Specification”

If the Group section does not contain data, the formula evaluates to the default (False condition) and reports “Out of Specification”.

## Example: A Conditional Statement with Two Conditions and Three Possible Outcomes

Below is an example of a wavelength combination formula that has 2 conditions and 3 outcomes:

The first condition stipulates that if the values for Lm1 in all wells are less than the value of Lm1 in well A12, then report the NAN MakeErr(118) (which displays the word “Low”). If they are equal to or greater than the value in well A12, then apply the second condition.

The second condition stipulates that if the values for Lm1 in all wells are greater than the value of Lm1 in well H1, then report NAN MakeErr(117) (which displays the word “High”). Otherwise, if this condition is not true, then report NAN MakeErr(123) (which displays “\*\*\*\*\*”).

NANs, including MakeErr functions, are discussed in detail in [Functions on page 29](#).

### Formula

```
If(!WellLm1<!A12Lm1@Plate#X, MakeErr(118),
  (If(!WellLm1>!H1Lm1@Plate#X, MakeErr(117), MakeErr(123))))
```

### Breakdown

Condition#1: !WellLm1<!A12Lm1@Plate#X

Value-if-Condition#1 True: MakeErr(118) (= “Low”)

Condition#2: !WellLm1>!H1Lm1@Plate#X

Value-if-Condition#2 True: MakeErr(117) (= “High”)

Value-if-Condition#2 False: MakeErr(123) (= “\*\*\*\*\*”)



## Tips for Writing Conditional Formulas

- When writing a complex or nested conditional statement, it is best to break it into several simple or smaller conditional statements and then combine them.
- Nesting conditional statements requires multiple pairs of parentheses. Make sure that you have the same number of opening and closing parentheses.
- In general, one more outcome than the number of conditions is always present. For example, 1 condition with 2 outcomes, or 2 conditions with 3 outcomes, and so on.
- Outcomes must evaluate to all numbers or all text. For example, if the True outcome of a formula evaluates to a number, the False outcome of the same formula must evaluate to a number and cannot evaluate to text. However, if a conditional statement evaluates to numbers, you can use the predefined NANs to include text in the results.

Likewise, if a conditional statement evaluates to text, you can return numbers as a result as long as the numbers are treated as text.

- When no data is present in a file (for example, data has not yet been collected), conditional formulas default to return the False outcome. Therefore, conditional formulas should be written such that the last outcome is the outcome you want to see as the default.

For example, if a conditional formula reports “Pass” or “Fail”, you should write the formula so that it defaults to reporting “Fail” so it does not return a positive result when the file does not contain data.

For example, the two sample formulas below report “Pass” if the Values are less than or equal to 0.2 and report “Fail” if they are greater than 0.2.

- `If(Values > 0.2, "Fail", "Pass")`
- `If(Values <= 0.2, "Pass", "Fail")`

Because the first formula is written such that “Pass” is the False outcome, the default result is “Pass” when no data is present. The second formula is a better way to write this conditional because “Fail” is the False outcome and is reported as the default result when no data has been collected.

## Concatenation Operators

In SoftMax Pro Software, operators can be used to concatenate text strings, lists of numbers, and arrays of numbers. These operators let you do complex data analyses using relatively simple formulas.

### + to Concatenate Text Strings

Type + to concatenate strings of text. For example, "Soft"+"Max" returns "SoftMax" as the result.

### & to Concatenate Numbers to a List

Type & (ampersand) to combines several lists of numbers into a single list of numbers.

X&Y&Z

### ~ to Concatenate Numbers to an Array

Type ~ (tilde) to combines several lists of numbers into an array of numbers.

X~Y~Z

## Concatenating Numbers

The following tables illustrate the difference between ~ and & in brief.

**Table 2-2: Original Lists Before Concatenation**

| List A | List B |
|--------|--------|
| 1      | 5      |
| 2      | 6      |
| 3      | 7      |
| 4      | 8      |

**Table 2-3: Concatenation Using &**

| A&B |
|-----|
| 1   |
| 2   |
| 3   |
| 4   |
| 5   |
| 6   |
| 7   |
| 8   |

**Table 2-4: Concatenation Using ~**

| A~B |   |
|-----|---|
| 1   | 5 |
| 2   | 6 |
| 3   | 7 |
| 4   | 8 |

## Concatenation Operator Examples

The following examples show how you can use Concatenation operators.

### Example: Using & in a Column Formula

Suppose you have a Plate section that collected Kinetic data at four wavelengths and Lm2 and Lm4 are identical. !WellLm2 and !WellLm4 are the optical densities collected at each time point in each well at the second and fourth wavelengths, reported into the Group section in template (or group) order.

The & operator can be used to combine !WellLm2 and !WellLm4 into a single list of numbers, which can then be used to determine the maximum value of the list:

```
Max(!WellLm2 & !WellLm4)
```

This formula returns the maximum value of the item within the parentheses.

### Example: Using ~ in a Group

Suppose we have four columns AvgOD490, AvgOD590, AvgOD620, and AvgOD670 that report the average optical density value of several wells at the specified wavelengths from a Spectrum scan. We can define a new column MaxOD to report the maximum OD from each of the columns using the ~ function:

```
Max(AvgOD490 ~ AvgOD590 ~ AvgOD620 ~ AvgOD670)
```

### Example: Using ~ in a Group Directly from the Wells

You can use the tilde operator to put information into a Group section directly from the wells of the plate, bypassing the group structure setup in the template (for example, the data are entered in well order, not group order). Consider the following formula:

```
Average(!A12@Plate#1 ~ !B12@Plate#1 ~ !C12@Plate#1 ~ !D12@Plate#1 ~ !E12@Plate#1  
~ !F12@Plate#1 ~ !G12@Plate#1 ~ !H12@Plate#1)
```

The information from the Spectrum scans in these wells is averaged together on a wavelength-by-wavelength basis and is reported in a column. For example, the data obtained from wells A12, B12, C12, D12, E12, F12, G12, and H12 at 190 nm are averaged, the data obtained from wells A12, B12, C12, D12, E12, F12, G12, and H12 at 191 nm are averaged, and so on.

Formulas such as this one are very powerful because they let to bring Spectrum scan or Kinetic run information into a Group section and do complex analyses such as averaging wells on a point-by-point basis, graphing the wells (or groups of wells) in the Graph section and applying different curve fits, or subtracting two Spectrum scans on a point-by-point basis and then plotting the difference.

Formulas that put information into the Group section directly from the Plate section, bypassing the template, should be used with caution because the information in the Group section does not correspond to the well assignments in the template.

This section describes the different types of built-in functions that can be used to assemble formulas in SoftMax® Pro Software.

The general format for the many SoftMax Pro Software built-in functions is:

FunctionName(Parameter1, Parameter2, Parameter3)

Functions can have zero to three parameters input to them. Lists of numbers, arrays of numbers, and lists of text strings can be used as input parameters, and the function can return either lists or arrays of numbers or text as applicable. Functions return numbers, text, or booleans.

If a function has more than one parameter, the order in which the parameters are listed is extremely important. If the parameters are entered in the incorrect order, the function will return erroneous results.

SoftMax Pro Software functions are divided into the following categories:

- [Mathematical Functions on page 30](#)
- [Statistical Functions on page 36](#)
- [Graph Functions on page 41](#)
- [Vmax Reduction Functions on page 52](#)
- [Peak Pro Analysis Functions on page 57](#)
- [Time Functions on page 61](#)
- [Interpolation Functions on page 62](#)
- [Other Functions on page 63](#)
- [NaNs and MakeErrs on page 69](#)
- [Text Functions on page 72](#)

## Mathematical Functions

Mathematical functions take numbers, lists of numbers, or arrays of numbers as parameters and return corresponding numbers, lists of numbers, or arrays of numbers. For example, if a list of numbers is given as the parameter, a list of numbers is returned as the result. The mathematical functions in the SoftMax Pro Software are all common mathematical, algebraic, and trigonometric functions, and can be used alone or in combination with other functions, accessors, or operators to create more complex formulas.

Trigonometric functions are calculated in radians (180 degrees = pi radians).

The mathematical functions can be used singly as shown in the examples, or they can be used in combination with each other. For example:

`Ceil(Abs(Values))`

This formula takes the items in a column named Values, takes the absolute values, and rounds them up to the nearest integer.

These functions can also be used in combination with other mathematical functions, accessors, and operators to create more complex formulas.



**Note:** If you use Round, Int, Ceil, and Floor when the SoftMax Pro Software is set to display results to a set number of decimal places, and if the number of decimal places specified for is larger than the number specified for the individual function, the extra decimal places are filled with zeroes. For example, if the SoftMax Pro Software is set to display three decimal places and you are rounding a number to two decimal places, the number is displayed as X.YZ0. Similarly, integers are displayed as X.000.

---

SoftMax Pro Software supports the following mathematical functions.

### Abs

`Abs(Parameter)`

Returns the absolute value of a number, list of numbers, or array of numbers.

`Abs(-10) = 10`

### Acos

`Acos(Parameter)`

Returns the arccosine of a number, list of numbers, or array of numbers.

`Acos(-0.25) = 1.82348`

`Acos(!CombinedPlot)`

## AntiLog

AntiLog(Parameter)

Returns the antilog (base e) of a number, list of numbers, or array of numbers. It is equivalent to  $e^X$  where X is a number, list of numbers, or array of numbers.

$$\text{Antilog}(1) = 2.718$$

Antilog(Summary#1)

## AntiLog10

AntiLog10 (Parameter)

Returns the antilog (base 10) of a number, list of numbers, or array of numbers. It is equivalent to  $10^X$  where X is a number, list of numbers, or array of numbers.

$$\text{Antilog10}(1) = 10$$

Antilog(Results)

## Asin

Asin(Parameter)

Returns the arcsine of a number, list of numbers, or array of numbers.

$$\text{Asin}(-0.25) = 0.25268$$

Asin(ColumnX)

## Atan

Atan(Parameter)

Returns the arctangent of a number, list of numbers, or array of numbers.

$$\text{Atan}(0.25) = 0.24498$$

Atan(!Lm1)

## Atan2

Atan2(Parameter,Parameter)

Returns the arctangent from X-coordinates and Y-coordinates. The two parameters can be numbers, lists of numbers or arrays of numbers. Both parameters should be of the same type, and have the same number of items in them. The function returns a single number, list of numbers, or array of numbers from the two parameters entered.

$$\text{Atan2}(1, 1) = 0.78540$$

Atan2(Values, Concentration)

## Ceil

Ceil(Parameter)

Returns a number, list of numbers, or array of numbers rounded up to the nearest integer. This function is the opposite of Floor. See [Floor on page 33](#).

Ceil(1.9) = 2

Ceil(-1.9) = -1

Ceil(Results)

## Cos

Cos(Parameter)

Returns the cosine of a number, list of numbers, or array of numbers.

Cos(0.25) = 0.96891

Cos(Values)

## Cosh

Cosh(Parameter)

Returns the hyperbolic cosine of a number, list of numbers, or array of numbers.

Cosh(4) = 27.30823

Cosh(!WellM1)

## Exp

Exp(Parameter)

Returns the exponent (the initial value raised to the power defined by the parameter) of a number, list of numbers, or array of numbers.

Exp(3) = 20.08554

Exp(Summary#3)

## Fact

Fact(Parameter)

Returns the factorial of a number, list of numbers, or array of numbers. A factorial is obtained by multiplying a series of consecutive integers, with the start at 1 and the end at the specified number. For example, the factorial of 4 is 1 x 2 x 3 x 4.

Fact(4) = 24

Fact(Col#1)



## Floor

Floor(Parameter)

Returns a number, list of numbers, or array of numbers rounded down to the nearest integer. This function is the opposite of Ceil. See [Ceil on page 32](#).

Floor(1.9) = 1

Floor(-1.9) = -2

Floor(Results)

## Fract

Fract(Parameter)

Returns the fractional part of a number, list of numbers, or array of numbers.

Fract(34.567) = 0.567

Fract(Values)

## Int

Int(Parameter)

Returns the integer part of a number, list of numbers, or array of numbers.

Int(6.9) = 6

Int(-6.9) = -6

Int(Results)

## Ln

Ln(Parameter)

Returns the natural logarithm of a number, list of numbers, or array of numbers. This function is the same as Log. See [Log on page 34](#).

Ln(5) = 1.60944

Ln(Values)

## Log

Log(Parameter)

Returns the natural logarithm of a number, list of numbers, or array of numbers. This function is the same as Ln. See [Ln on page 33](#).

$$\text{Log}(5) = 1.60944$$

Log(Values)

## Log10

Log10(Parameter)

Returns the logarithm base 10 of a number, list of numbers, or array of numbers.

$$\text{Log10}(5) = 0.69897$$

Log10(!Lm1)

## Pi

Pi

Returns the value of  $\pi$ . No parameters.

$$\text{Summary\#1} * \text{Pi}$$

## Rand

Rand

Returns a random number between 0 and 1. Takes no parameters.

Rand

## RandNorm

RandNorm

Returns a random number, normally distributed around a mean of approximately 0, and a standard deviation of approximately 1.

## Round

Round(Parameter,# of digits)

Rounds a number, list of numbers, or array of numbers to the specified number of digits. The first parameter is a number, list of numbers, or array of numbers. The second parameter is the number of digits to the right of the decimal point the result will be rounded to.

$$\text{Round}(3.456,2) = 3.46$$

Round(Results,2) Rounds the values in a column named Results to the hundredths.

## Sign

Sign(Parameter)

Returns the sign of a number, list of numbers, or array of numbers.

If a number is positive, it returns 1. For a negative numbers, it returns -1. For zero, it returns 0.

Sign(25) = 1

Sign(-25) = -1

Sign (0) = 0

## Sin

Sin(Parameter)

Returns the sine of a number, list of numbers, or array of numbers.

Sin(2) = 0.90930

Sin(Column1)

## Sinh

Sinh(Parameter)

Returns the hyperbolic sine of a number, list of numbers, or array of numbers.

Sinh(1) = 1.17520

Sinh(Column1)

## Sqrt

Sqrt(Parameter)

Returns the positive square root of a number, list of numbers, or array of numbers.

Sqrt(64) = 8

Sqrt(Column1)

## Tan

Tan(Parameter)

Returns the tangent of a number, list of numbers, or array of numbers.

Tan(0.25) = 0.25534

Tan(ParmB)

## Tanh

Tanh(Parameter)

Returns the hyperbolic tangent of a number, list of numbers, or array of numbers.

Tanh(-2) = -0.96403

Tanh(ParmB)

## Statistical Functions

Statistical functions behave differently depending on whether they are in a Summary formula or a Column formula.

Summary formulas take a list of numbers as a parameter and return a single number.

Column formulas take a 1-dimensional list of numbers (row or column) as a parameter and return a single number. If the column is a 2-dimensional array, a list of numbers is returned, 1 value per row.

SoftMax Pro Software supports the following statistical functions.

### ANOVAStatAndProb

ANOVAStatAndProb((x1~x2~...~xN)&(y1~y2~...~yN))

Does standard ANOVA analysis on data specified as an array list of numbers. The format for the data is group1&group2&...&groupK, where each group is a list of numbers x1~x2~...~xN.

The returned value is an array of two numbers: the F-value and its associated probability.

Under the null hypothesis that all groups have the same mean, the F-value is an F-statistic with numerator degrees of freedom = (number of groups - 1) and denominator degrees of freedom = (number of data - number of groups).

For example:

ANOVAStatAndProb((1~2~3)&(4~5~6))

### Average

Average(Parameter)

Returns the average of a list or array of numbers.

### AvgDev

AvgDev(Parameter)

Returns the average of the absolute deviation from the mean of a list or array of numbers.

## ChiProbEx

ChiProbEx(chiSq, df)

Returns the probability that a variable that is chi-squared distributed with the specified degrees of freedom (df) exceeds the specified chi-squared value (chiSq).

For the parameters, **chiSq** must be non-negative and **df** must be a positive integer.

## Cv

Cv(Parameter)

Returns the coefficient of variation of a list or array of numbers.

## Cvp

Cvp(Parameter)

Returns the coefficient of variation based on the entire population given as a list or array of numbers.

## Derivative

Derivative

Given an array of y-values and an x spacing, returns an array of numbers estimating the derivative at each point using a Savitzky-Golay filter.

## ESDPercentagePoint

ESDPercentagePoint( $\alpha, n, i$ )

Returns the percentage point for the Rosner many-outlier procedure. See Rosner, B. Percentage Points for a Generalized ESD Many-Outlier Procedure. *Technometrics* **Vol. 25**, Pages 165–172 (1983).

Parameters:

- $\alpha$  = significance level (probability)
- $n$  = number of data points
- $i$  = number of outliers

## FDist

FDist(x, df1, df2)

Returns the probability for a given F statistic, x, with df1 numerator degrees of freedom, and df2 denominator degrees of freedom. It is equivalent to the Excel formula with identical syntax, FDIST(x, df1, df2).

## **FInv**

FInv(p,df1,df2)

Returns the inverse of the FDist formula; the numerator degrees of freedom df1, and denominator degrees of freedom df2, for which the probability is p.

p must satisfy  $0 < p < 1$ .

## **FirstZero**

FirstZero(numberarray,numberarray)

Given an array of y-values (first parameter) and an array of corresponding x-values with the same number of elements, returns the first x-value at which the interpolated y-value is zero.

## **Max**

Max(Parameter)

Returns the maximum value in a list or array of numbers.

## **MaxDev**

MaxDev(Parameter)

Returns the absolute value of the maximum absolute deviation from the mean of a list or array of numbers.

## **Median**

Median(Parameter)

Returns the median of the given list or array of numbers.

## **Min**

Min(Parameter)

Returns the minimum value in a list or array of numbers.

## **SecondDerivative**

SecondDerivative

Given an array of y-values and an x spacing, returns an array of numbers estimating the second derivative at each point using a Savitzky-Golay filter.

## **Smooth**

Smooth

Smooths an array of numbers using a Savitzky-Golay filter. The array to be smoothed is the parameter, and the returned array contains the smoothed values.

## StdErr

StdErr(Parameter)

Returns the standard error of a list or array of numbers, which is the standard deviation divided by the square root of the number of values in the list or array.

## StDev

StDev(Parameter)

Returns the standard deviation of the given list or array of numbers based on a sample.

## StDevp

StDevp(Parameter)

Returns the standard deviation of a list or array of numbers based on the entire population.

## Sum

Sum(Parameter)

Returns the sum of a list or array of numbers.

## TDist

TDist(x,df)

Returns the two-tailed probability for a given T-statistic, x, with df degrees of freedom. This T-distribution function is equivalent to the Excel formula TDist(x, df, 2).

The value for df must be a positive integer.

## TInv

TInv(p,df)

Returns the inverse of the TDist formula, with df degrees of freedom, for which the probability is p. This T-test function is equivalent to the Excel formula with identical syntax, TInv(p,df). Commonly available tabulations of T are often expressed in terms of the percent confidence level, which is  $1-p$ , or  $1-p/2$  if the table is one-tailed.

The value for p must satisfy  $0 < p < 1$ .

## Statistical Functions Example

The following example shows how you can use statistical functions.

### Example: Reducing an Array of Values to a Single Number per Well

Statistical functions are useful to reduce a list of values (for example, Spectrum scan optical densities) to a single number per well.

For example, if you want the maximum values in a Spectrum scan, you can use one of the following formulas:

Max(!Lm1) in the Plate section reduction

Max(!WellLm1) in a Column formula

Similarly, you can use Min(!Lm1) and Min(!WellLm1) to report the minimum optical density. You can also combine these formulas with the NullBetween and NullOutside functions to find an optical density at a wavelength other than max lambda. Also, the NthItem function can be used to report the OD at a selected wavelength in the scan. For more information on NullBetween, NullOutside, and NthItem, see [Other Functions on page 63](#).



## Graph Functions

Graph functions are used to interpolate data from curve fits in Graph sections, and to access curve fit parameters.

The InterpX, InterpY, and Slope functions are generally used in Group section column formulas. Although they are used in specialized Summaries, and the slope function is also used in Plate section reductions.

The parameter functions, Intercept, ParmA, ParmB, ParmC, ParmD, ParmG, and Rsquared, are generally used in Group section or Notes section Summary formulas.

For detailed information about the curve-fit parameters, see the SoftMax Pro Software application help or user guide.

SoftMax Pro Software supports the following graph functions.

### AreaUnder

AreaUnder(Xvalues, Yvalues)

Returns the total area under a curve. This function is included as one of the default Kinetic and Spectrum reductions in the Reduction dialog for Plate sections and Cuvette Set sections. It can also be used in a Summary formula in a Group section.

For example, if “Xvalues” is the name of the list of numbers to be used for the X values and “Yvalues” is the name of the list of numbers to be used for the Y values, the formula would be:

AreaUnder(Xvalues, Yvalues)

### AreaUnderFit

AreaUnderFit(PlotName, Xstart, Xstop)

Returns the area under the portion of the fit curve defined by the start and stop parameters. The curve is divided into 100 equal ‘slices’ between the start and stop parameters, and then the areas of the trapezoids defined by the ‘slices’ are summed to determine the area under the curve.

- PlotName is the name of the plot for which you want to determine the area under the curve.
- Xstart specifies the point on the X-axis at which to start the area calculation.
- Xstop specifies the point on the X-axis at which to stop calculating the area under the curve.

The area under the curve of a plot in a Graph section can be determined using the AreaUnderFit function. The function generates a series of trapezoids between the X-axis and the curve, with the start and end at points on the X-Axis specified by the second and third parameters of the function. The area within each trapezoid is calculated and then all of the trapezoidal areas are summed.

AreaUnderFit(Plotname@Graphname, 1, 1000)

## ChiProbability

ChiProbability(PlotName@GraphSection)

Returns the chi-squared probability associated with the specified plot.

## ChiProbabilityPLA

ChiProbabilityPLA (PlotName@GraphSection)

Returns the chi-squared probability associated with the difference in sum-of-squared errors of the parallel and independent models.

This can be used only when Parallel Line Analysis is enabled in the Parameter Settings.

The designation of a specific plot is somewhat arbitrary since the chi-squared probability is calculated from all plots in the designated graph.

## ChiSquared

ChiSquared(PlotName@ Graph Name)

Returns the value of the chi-squared statistic from the curve fit.

The chi-squared statistic is also know as sum-of-squared errors (SSE).

PlotName@GraphName is the full name of the plot, including the name of the graph. For example, Plot#1@Graph#1, or Std@Standardcurve.

For a PLA fit, ChiSquared returns the value of the chi-squared statistic for a reduced curve fit for the PLA (global) model, without respect to the plot parameter. This is the same as the value returned by ChiSquaredPLA.

## ChiSquaredPLA

ChiSquaredPLA (PlotName@ GraphSection)

Returns the value of the chi-squared statistic for a reduced curve fit. This can be used only when Parallel Line Analysis is enabled in the Parameter Settings.

The designation of a specific plot is somewhat arbitrary since the chi-squared value is calculated from all plots in the designated graph.

ChiSquaredPLA(StandardCurve@PLAGraph)

## ConfidenceLevel

ConfidenceLevel

Returns the confidence level (as a percentage) used by the specified plot for confidence interval calculations.

ConfidenceLevel(StandardCurve@PLAGraph)

## DegreesOfFreedom

DegreesOfFreedom(PlotName@GraphSection)

Returns the number of degrees of freedom of the curve fit for the specified plot.

DegreesOfFreedom was previously known as DF. DF has been deprecated.

## DF

DF

DF has been deprecated. See [DegreesOfFreedom](#) on page 43.

DF is still supported in version 7.0, but is not guaranteed to work in future versions of the software.



**Note:** If you have a protocol or data file that uses DF, the software automatically updates the formula to use DegreesOfFreedom.

---

## FProbPLA

FProbPLA(PlotName@GraphSection)

Returns the value of the F-test probability for a reduced curve fit.

This can be used only when Parallel Line Analysis is enabled in the Parameter Settings.

The designation of a specific plot is somewhat arbitrary since the F-test probability is calculated from all plots in the designated graph.

## FStatPLA

FStatPLA(PlotName@ GraphSection)

Returns the value of the F-test statistic for a reduced curve fit.

This can be used only when Parallel Line Analysis is enabled in the Parameter Settings.

The designation of a specific plot is somewhat arbitrary since the F-test statistic is calculated from all plots in the designated graph.

## Intercept

Intercept(Xvalues, Yvalues)

Returns the intercept of a linear regression line that was fit ( $Y=mX+b$ ) to the X and Y values as identified in the function.

- Xvalues is the list of numbers to be used for the X values.
- Yvalues is the list of numbers to be used for the Y values.

Intercept(Concentration, MeanValue)

## InterpX

InterpX (PlotName@GraphName, Yvalues)

Returns the X values of the specified plot interpolated using the Y values and curve fit.

PlotName@GraphName is the full name of the plot, including the name of the graph. For example, Plot#1@Graph#1, or Std@Standardcurve.

InterpX(Std@StandardCurve, YValues)

## InterpY

InterpY (PlotName@GraphName, Xvalues)

Returns the Y values of the specified plot interpolated using the X values and curve fit.

InterpY(Plot#6@Graph#3, XValues)

## NormalOrderStatisticMedians

NormalOrderStatisticMedians(*n*)

Returns the list of normal-ordered statistic medians for the specified number of data points (*n*).

A normal probability plot can be constructed by plotting the data against this list. For normally distributed data, the points should fall approximately on a straight line, so that normality can be assessed visually or by linear regression.

The parameter value must match the number of data points in the column or group of interest.

For example, if the data you are analyzing has 48 data points, the formula would be:

NormalOrderStatisticMedians(48)

## NumData

NumData(PlotName@GraphName)

Returns the number of data points in the specified plot.

## NumVarParams

NumVarParams(PlotName@GraphName)

Returns the number of variable parameters of the curve fit for the specified plot.

## ParmA

ParmA(PlotName@GraphName)

Returns the A parameter value of the curve fit assigned to the specified plot.

The ParmA, ParmB, ParmC, ParmD, and ParmG functions provide a way to export curve fit parameters with the analyzed data, and also to display the curve-fit parameters to a greater number of decimal points than are displayed by default in Graph sections.

## ParmB

ParmB(PlotName@ GraphName)

Returns the B parameter value of the curve fit assigned to the specified plot.

The ParmA, ParmB, ParmC, ParmD, and ParmG functions provide a way to export curve fit parameters with the analyzed data, and also to display the curve-fit parameters to a greater number of decimal points than are displayed by default in Graph sections.

## ParmC

ParmC(PlotName@GraphName)

Returns the C parameter value of the curve fit assigned to the specified plot.

The ParmA, ParmB, ParmC, ParmD, and ParmG functions provide a way to export curve fit parameters with the analyzed data, and also to display the curve-fit parameters to a greater number of decimal points than are displayed by default in Graph sections.

The ParmC function is particularly interesting as it is the IC50 of a 4-parameter logistic curve fit with well-defined upper and lower asymptotes.

## ParmD

ParmD(PlotName@GraphName)

Returns the D parameter value of the curve fit assigned to the specified plot.

The ParmA, ParmB, ParmC, ParmD, and ParmG functions provide a way to export curve fit parameters with the analyzed data, and also to display the curve-fit parameters to a greater number of decimal points than are displayed by default in Graph sections.

## ParmG

ParmG(PlotName@GraphName)

Returns the G parameter value of the curve fit assigned to the specified plot.

The ParmA, ParmB, ParmC, ParmD, and ParmG functions provide a way to export curve fit parameters with the analyzed data, and also to display the curve-fit parameters to a greater number of decimal points than are displayed by default in Graph sections.

ParmG refers to the fifth parameter of a 5-parameter logistical curve fit.

## ParmACILower

ParmACILower (PlotName@GraphName)

Returns the lower limit of the confidence interval for the A parameter of a curve fit assigned to the specific plot.



**Note:** Calculations of confidence intervals including specification of the confidence level as a percentage must be set in the Curve Fit Settings for the Graph section.

---

## ParmBCILower

ParmBCILower (PlotName@GraphName)

Returns the lower limit of the confidence interval for the B parameter of a curve fit assigned to the specific plot.



**Note:** Calculations of confidence intervals including specification of the confidence level as a percentage must be set in the Curve Fit Settings for the Graph section.

---

## ParmCCILower

ParmCCILower (PlotName@GraphName)

Returns the lower limit of the confidence interval for the C parameter of a curve fit assigned to the specific plot.



**Note:** Calculations of confidence intervals including specification of the confidence level as a percentage must be set in the Curve Fit Settings for the Graph section.

---

## ParmDCILower

ParmDCILower (PlotName@GraphName)

Returns the lower limit of the confidence interval for the D parameter of a curve fit assigned to the specific plot.



**Note:** Calculations of confidence intervals including specification of the confidence level as a percentage must be set in the Curve Fit Settings for the Graph section.

---

## ParmGCILower

ParmGCILower (PlotName@GraphName)

Returns the lower limit of the confidence interval for the G parameter of a curve fit assigned to the specific plot.



**Note:** Calculations of confidence intervals including specification of the confidence level as a percentage must be set in the Curve Fit Settings for the Graph section.

---

## ParmACIUpper

ParmACIUpper (PlotName@GraphName)

Returns the upper limit of the confidence interval for the A parameter of a curve fit assigned to the specific plot.



**Note:** Calculations of confidence intervals including specification of the confidence level as a percentage must be set in the Curve Fit Settings for the Graph section.

---

## ParmBCIUpper

ParmBCIUpper (PlotName@GraphName)

Returns the upper limit of the confidence interval for the B parameter of a curve fit assigned to the specific plot.



**Note:** Calculations of confidence intervals including specification of the confidence level as a percentage must be set in the Curve Fit Settings for the Graph section.

---

## ParmCCIUpper

ParmCCIUpper (PlotName@GraphName)

Returns the upper limit of the confidence interval for the C parameter of a curve fit assigned to the specific plot.



**Note:** Calculations of confidence intervals including specification of the confidence level as a percentage must be set in the Curve Fit Settings for the Graph section.

---

## ParmDCIUpper

ParmDCIUpper (PlotName@GraphName)

Returns the upper limit of the confidence interval for the D parameter of a curve fit assigned to the specific plot.



**Note:** Calculations of confidence intervals including specification of the confidence level as a percentage must be set in the Curve Fit Settings for the Graph section.

---

## ParmGCIUpper

ParmGCIUpper (PlotName@GraphName)

Returns the upper limit of the confidence interval for the G parameter of a curve fit assigned to the specific plot.



**Note:** Calculations of confidence intervals including specification of the confidence level as a percentage must be set in the Curve Fit Settings for the Graph section.

---

## ParmAVariance

ParmAVariance(PlotName@GraphName)

Returns the variance of the curve fit parameter A for the specified plot.

## ParmBVariance

ParmBVariance(PlotName@GraphName)

Returns the variance of the curve fit parameter B for the specified plot.

## ParmCVariance

ParmCVariance(PlotName@GraphName)

Returns the variance of the curve fit parameter C for the specified plot.

## ParmDVariance

ParmDVariance(PlotName@GraphName)

Returns the variance of the curve fit parameter D for the specified plot.

## ParmGVariance

ParmGVariance(PlotName@GraphName)

Returns the variance of the curve fit parameter G for the specified plot.



## RelativePotency

RelativePotency (Standard PlotName, TestPlotName)

Returns the Relative Potency of a Standard compared to a Test sample.

When using a 4-parameter curve fit, the Relative Potency is calculated as the ratio of the C values for the StandardPlotName curve to the TestPlotName curve.

With a 5-parameter curve fit, the C values are no longer the mid-point between the max and min. compensates by calculating the IC50 for each plot as:

$$C \times [2^{(1/G)} - 1]^{(1/B)}$$

before taking the ratio.

## RelPotCILowerPLA

RelPotCILowerPLA (PlotName@GraphName)

Returns the lower limit of the confidence interval for the relative potency as determined by the Parallel Line Analysis feature.



**Note:** Calculations of confidence intervals including specification of the confidence level as a percentage must be set in the Curve Fit Settings for the Graph section.

---

## RelPotCIUpperPLA

RelPotCIUpperPLA (PlotName@GraphName)

Returns the upper limit of the confidence interval for the relative potency as determined by the Parallel Line Analysis feature.



**Note:** Calculations of confidence intervals including specification of the confidence level as a percentage must be set in the Curve Fit Settings for the Graph section.

---

## RelPotPLA

RelPotPLA(PlotName@GraphName)

Returns the relative potency of the parallel line analysis curve fit for the specified plot.

## RelPotVariancePLA

RelPotVariancePLA(PlotName@GraphName)

Returns the variance of the relative potency of the parallel line analysis curve fit for the specified plot.

## RSquared

RSquared(PlotName@GraphName)

Returns the square of the correlation coefficient of the curve fit assigned to the specified plot.

## RSquaredPLA

RSquaredPLA(PlotName@GraphName)

Returns the coefficient of determination for the parallel line analysis curve fit for the specified plot.

## Slope

Slope(Xvalues, Yvalues)

Returns the slope of a line that was fit using linear regression ( $Y=mX + b$ ) to the X and Y values specified in the function.

The Slope function is included in the Kinetic Reduction list in the Reduction dialog for Kinetic runs.

- Xvalues is list of numbers to be used for the X values.
- Yvalues is the list of numbers to be used for the Y values.

## Sort

Sort(number | list)

Takes a list of numbers as its parameter, and returns a list of those numbers sorted in ascending order.

## Graph Functions Examples

The following examples show how you can use graph functions.

### Example: Using InterpX and InterpY in Group Sections

Suppose you have a Graph section named “Graph#1”, in which you have a plot named “Plot#1”, generated from the X values list Concentration and the Y values list MeanValue:

- To create a column of interpolated X values, use the formula `InterpX(Plot#1@Graph#1, MeanValue)`.
- To create a column of interpolated Y values, use the formula `InterpY(Plot#1@Graph#1, Concentration)`.

### Example: Determining the Slope of the Optical Density versus Time Squared for a Kinetic Plot

Slope is one of the predefined choices in the list of Kinetic reductions in the Reduction dialog for Plate sections and Cuvette Set sections. However, you can also use this function in custom calculations.

For example, you can plot Optical Density versus the square of the Time parameter, and the slope of the line can be calculated using the custom Kinetic reduction:

$$\text{Slope}(!\text{TimeRun})^2, !\text{Lm1}$$

The reduced plot display for Kinetic plots in Plate sections or Cuvette Set sections always displays the results of wavelength combination reductions, but not the results of custom Kinetic reductions. However, the reduced number for custom Kinetic reductions is calculated using the specified reduction.

## Vmax Reduction Functions

You can determine Vmax or the Time to Vmax by using the functions in the **Data Reduction** dialog. As an alternative, you can use Vmax reduction functions to customize Vmax reductions.

### Vmax

Vmax is the maximum slope of the Kinetic display of mOD/min or RFU/RLU per second. Vmax is calculated by measuring the slopes of a number of straight lines, where Vmax Points determines the number of contiguous points over which each straight line is defined.

This is an alternative method for analyzing non-linear Kinetic reactions that reports the elapsed time until the maximum reaction rate is reached, rather than reporting the maximum rate itself. Used in conjunction with Vmax Points, Time to Vmax is the time to the midpoint of the line defined by Vmax Points and used to calculate Vmax.

### Vmax Reduction

Vmax reduction functions require three parameters.

#### First Parameter

The list or array of numbers to be used for the calculation (for example, !CombinedPlot, !WellLm1, !Lm1).

#### Second Parameter

The number of Vmax Points to be used in the calculation. A constant (for example, 45) can be entered for the value of this parameter. If the constant specifies more Vmax Points than are contained in the list or array of numbers specified by the first parameter, then all points in the list/array will be used. As an alternative, the !VmaxPoints accessor can be used for this parameter. In this case, the number of Vmax Points specified in the Plate section or Cuvette Set section Reduction dialog is used. See [Accessors on page 77](#).

#### Third Parameter

The read interval to be used in the calculation. A constant representing the number of seconds (for example, 120) can be used for this parameter. As an alternative, the !ReadInterval accessor can be used for this parameter, in which case the read interval specified in the Settings dialog is used. See [Accessors on page 77](#).

SoftMax Pro Software supports the following Vmax reduction functions.

## TimeToVmax

TimeToVmax(Parameter1, Parameter2, Parameter3)

Returns the Time to Vmax for the list or array specified by the first parameter, based on the number of Vmax Points and the read interval specified in the second and third parameters, respectively.

## TimeToVmaxEx

TimeToVmaxEx(Parameter1, Parameter2, Parameter3)

Use TimeToVmaxEx only for data acquired from a FlexStation Instrument.

Returns the Time to Vmax for the list specified by **Parameter1**, based on the number of Vmax points specified by **Parameter2** and the list of time values specified by **Parameter3**.

## Vmax

Vmax(Parameter1, Parameter2, Parameter3)

Returns the Vmax (in milli-units/min) for the list or array specified in **Parameter1**, based on the number of Vmax Points specified in **Parameter2** and the read interval specified in **Parameter3**.

If you use all the points in a Kinetic run, Vmax application is the same as the slope.

## VmaxEx

VmaxEx(Parameter1, Parameter2, Parameter3)

Use VmaxEx only for data acquired from a FlexStation Instrument.

Returns the Vmax (in milli-units/min) for the list specified in **Parameter1**, based on the number of Vmax points specified in **Parameter2** and the time values specified in **Parameter3**.

**Parameter3** is the list of X values, or time values, for a particular well, since data from a FlexStation instrument are individually time-tagged.

For example, the syntax of **Parameter3** can be **!WellIDXVals@Platename**, where *WellID* is the ID for the well (such as, A1), and *Platename* is the name of the **Plate** section.

## VmaxCorr

VmaxCorr(Parameter1, Parameter2, Parameter3)

Returns the correlation coefficient for the Vmax Rate of the list or array specified in Parameter1, based on the number of Vmax Points and read interval specified by the second and third parameters, respectively. It can be used to display this value as the reduced number in a Plate section or as a column in a Group section.

## VmaxCorrEx

VmaxCorrEx(Parameter1, Parameter2, Parameter3)

Use VmaxCorrEx only for data acquired from a FlexStation Instrument.

Returns the correlation coefficient for the Vmax Rate of the list specified in **Parameter1**, based on the number of Vmax points specified by **Parameter2** and the list of time values specified by **Parameter3**. It can be used to display this value as the reduced number in a **Plate** section or as a column in a **Group** section.

## VmaxPerSec

VmaxPerSec(Parameter1, Parameter2, Parameter3)

Returns the Vmax (in units/sec) for the list or array specified by the first parameter based on the number of Vmax Points and the read interval specified in the second and third parameters, respectively.

If you use all the points in a Kinetic run, VmaxPerSec is the same as the slope.

## VmaxPerSecEx

VmaxPerSecEx(Parameter1, Parameter2, Parameter3)

Use VmaxPerSecEx only for data acquired from a FlexStation Instrument.

Returns the Vmax (in units/sec) for the list specified by **Parameter1**, based on the number of Vmax points specified by **Parameter2** and the list of time values specified by **Parameter3**.

If you use all the points in a Kinetic run, VmaxPerSecEx is the same as the slope.

## VmaxPtsUsed

VmaxPtsUsed(Parameter1, Parameter2, Parameter3)

Returns the number of points that will be used to calculate the Vmax for the list or array specified in the first parameter, based on the number of Vmax Points and the read interval specified in the second and third parameters, respectively.

## VmaxPtsUsedEx

VmaxPtsUsedEx(Parameter1, Parameter2, Parameter3)

Use VmaxPtsUsedEx only for data acquired from a FlexStation Instrument.

Returns the number of points that will be used to calculate the Vmax for the list specified in **Parameter1**, based on the number of Vmax points specified by **Parameter2** and the list of time values specified by **Parameter3**.

## Vmax Reduction Functions Examples

The following examples show how you can use Vmax reduction functions.

### Example: Calculating Vmax Using Different Numbers of Vmax Points in Columns of a Group Section

You can calculate Vmax using different numbers of Vmax points. For example, you can create columns named Vmax45, Vmax25, Vmax10, and Vmax5 using variations on the custom formula:



```
Vmax(!WellCombinedPlot, 45, !ReadInterval)
```

with the second parameter set to 45, 25, 10, or 5, respectively. !WellCombinedPlot specifies the list of numbers to be used to calculate the VmaxRate. For a complete discussion of the !WellLx accessors, see [Plate Data Accessors on page 91](#). The !ReadInterval accessor specifies that the read interval defined in the Settings dialog.

Using fewer Vmax Points lets the SoftMax Pro Software calculate the linear regression of subsets of the data in each well using creeping iteration, and then report the fastest rate (that is, the steepest slope) in each well. For a complete description of how Vmax Points are used, and how Vmax is calculated, please see the SoftMax Pro Software application help or user guide.

### Example: Viewing the Vmax Correlation Coefficient as a Plate Reduction

To display the Vmax Correlation Coefficient as a plate reduction:

1. Click **Display Options**  in **Plate Tools** on the **Home Tab** in the ribbon or in the toolbar at the top of the Plate section.
2. In the **Display Options** dialog, choose to plot raw data with reduced number and click **OK**.
3. Click **Data Reduction**  in **Plate Tools** on the **Home Tab** in the ribbon or in the toolbar at the top of the Plate section.
4. In the **Reduction dialog**, select **Custom** for the kinetic reduction.
5. Type the following formula:

```
VmaxCorr(!CombinedPlot, !VmaxPoints, !ReadInterval)
```

The reduced number is the correlation coefficient of the Vmax Rate (or slope) calculation.

## Example: Viewing the Vmax Correlation Coefficient as a Column Reduction with the Vmax Rate

If Kinetic data is collected in a Plate section for which the Data Reduction dialog was used to set the reduction to Vmax, using the accessor !WellValues can put the reduced number from the Plate section into the Group section.

The Vmax Correlation Coefficient has the custom formula:

```
VmaxCorr(!WellCombinedPlot, !VmaxPoints, !ReadInterval)
```

The formula's first parameter, !WellCombinedPlot, identifies the list that should be calculated for the Vmax correlation coefficient, !VmaxPoints specifies the use of the Vmax Points as defined in the Reduction dialog, and !ReadInterval directs the formula to use the Read Interval specified in the Settings dialog.

The !WellLm1, !VmaxPoints, and !ReadInterval accessors are discussed in detail in [Accessors on page 77](#).

## Example: Viewing the Number of Vmax Points Used to Calculate Vmax in a Group Section Column

If a number is used to specify how many Vmax Points should be used when calculating the Vmax, that number of Vmax Points is used if that many data points are available in the well or cuvette plot in the Plate section or Cuvette Set section.

However, if the number of available data points is less than the specified number of Vmax Points, only the number of points that are available are used. For a complete discussion of how Vmax is calculated, see the SoftMax Pro Software application help or user guide.

This example demonstrates how to employ the VmaxPtsUsed function to determine the actual number of Vmax Points that are used in determining the Vmax Rate:

```
VmaxPtsUsed(!WellLm1, !VmaxPoints, !ReadInterval)
```



## Peak Pro Analysis Functions

Peak Pro™ Analysis functions provide advanced peak detection and characterization. SoftMax Pro Software supports the following Peak Pro Analysis functions.

### PeakProAnalysis

PeakProAnalysis(Parameter1,Parameter2,Parameter3)

Does Peak Pro Analysis on specified data.

- Parameter1 is the array of x values.
- Parameter2 the array of y values.
- Parameter3 is the array of five numerical configuration parameters for the algorithm:
  - Fit Width should be an estimate of the number of points expected to be spanned by a peak. The minimum value is 3.
  - Smooth Width is the number of points to be used in a moving window for smoothing.
 

As part of peak detection the data are smoothed. For perfect data this could be set to 1, but for real data the count should be high enough to smooth out noise sufficiently for the peak detection to function as expected. Too small a smooth width might result in identification of noise spikes as peaks.
  - Slope Threshold is a number between 0 and 1 that is used to determine whether a change in sign of the smoothed data derivative implies the existence of a peak. Its dimensions are that of the inverse of the x-data, and generally its value is 0.001 or less.
  - Amplitude Threshold is a number, with dimensions that of the y-data, used as a threshold for y-values during peak detection.
  - Dynamic or Fixed Amplitude threshold, if set to zero, the actual amplitude threshold is proportional to that specified, but varies dynamically depending on the local data. If set to 1, the threshold is fixed as specified in the fourth parameter.

The PeakProAnalysis function returns the output as an array of 16 peak attributes, but not all 16 attributes are necessarily populated with values. The analysis is configured by the set of five numerical configuration parameters described in Parameter3 above.

The 16 peak attributes in the array occur in the following order:

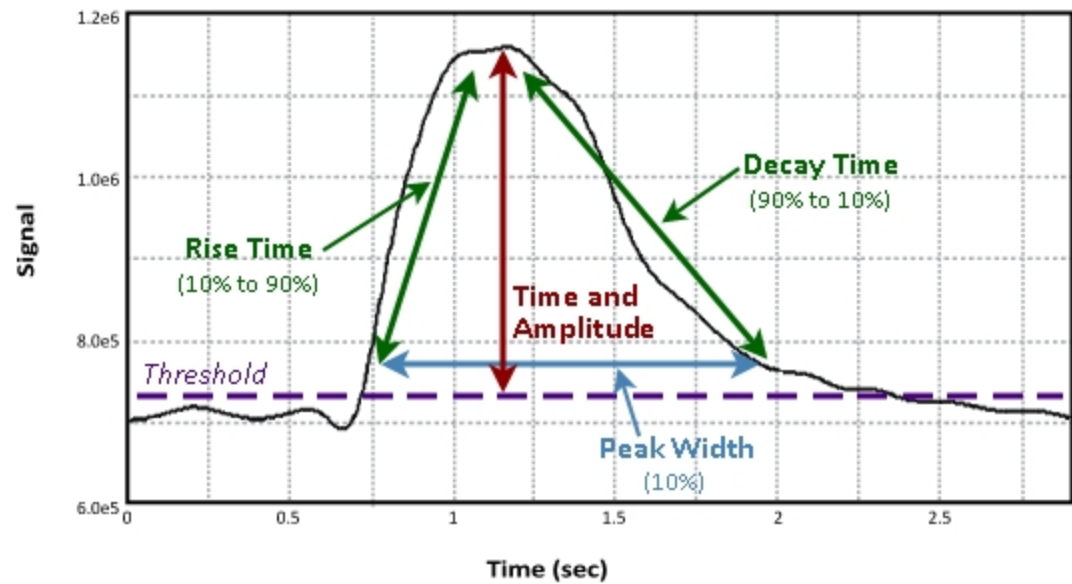
- Peak Count
- Peak Frequency (per minute)
- Peak Amplitude average
- Peak Amplitude standard deviation
- Peak Baseline average
- Peak Width average
- Peak Width standard deviation
- Peak Spacing average
- Peak Spacing standard deviation
- Peak Rise Time average
- Peak Rise Time standard deviation
- Peak Decay Time average
- Peak Decay Time standard deviation
- Peak Bottom Width average
- Peak Bottom Width standard deviation
- Peak Spacing Regularity

The index of each attribute in the array can be determined through the Peak Pro Index functions using the NthItem operator.

The following list shows the possible values for Peak Spacing Regularity:

- -1 = N/A
- 0 = Normal
- 1 = Extra Peaks
- 2 = Missing Peaks
- 3 = Irregular Peaks

The following figure illustrates some basic peak attributes.



### PeakProCountIndex

PeakProCountIndex

Returns the index for the peak count in the Peak Pro Analysis results array.

### PeakProFrequencyIndex

PeakProFrequencyIndex

Returns the index for the peak frequency in the Peak Pro Analysis results array.

### PeakProAmplitudeIndex

PeakProAmplitudeIndex

Returns the index for the average peak amplitude in the Peak Pro Analysis results array.

### PeakProWidthIndex

PeakProWidthIndex

Returns the index for the average peak width in the Peak Pro Analysis results array.

### PeakProWidthStdDevnIndex

PeakProWidthStdDevnIndex

Returns the index for the peak width standard deviation in the Peak Pro Analysis results array.

### **PeakProRiseTimeIndex**

PeakProRiseTimeIndex

Returns the index for the average peak rise time in the Peak Pro Analysis results array.

### **PeakProRiseTimeStdDevnIndex**

PeakProRiseTimeStdDevnIndex

Returns the index for the peak rise time standard deviation in the Peak Pro Analysis results array.

### **PeakProDecayTimeIndex**

PeakProDecayTimeIndex

Returns the index for the average peak decay time in the Peak Pro Analysis results array.

### **PeakProDecayTimeStdDevnIndex**

PeakProDecayTimeStdDevnIndex

Returns the index for the peak decay time standard deviation in the Peak Pro Analysis results array.

## **Peak Pro Analysis Functions Examples**

The following example shows how you can use Peak Pro Analysis functions.

### **Example: Using Peak Pro Analysis on Kinetic Data**

This formula returns the average peak width in kinetic data from well B6:

```
NthItem (PeakProAnalysis(!B6XVals,!B6Lm1,10&10&0.001&0&0),PeakProWidthIndex)
```

The configuration parameters will in general need to be adjusted based on the type of data. If multiple peak attributes are required, it is recommended to invoke PeakProAnalysis once in a separate summary formula and then access the corresponding items, rather than invoking PeakProAnalysis for each attribute.

## Time Functions

Time functions return the time in numeric or text format. Each of these functions takes one non-negative numeric parameter: the number of seconds since the beginning of time. See [!TimeOrigin on page 87](#).

### LocalTimeNumeric

LocalTimeNumeric(Parameter)

Returns an array of numbers encoding the local time and date:

- Seconds
- Minutes
- Hours
- Day of the month (1 to 31)
- Month (1 to 12)
- Year
- Day of the week (1 to 7)
- Day of the year (1 to 366)
- Daylight saving time in use (0 or 1)

Example: LocalTimeNumeric(!A1Lm1TimeOrigin@Plate1)

See [!TimeOrigin on page 87](#).

### UTCTimeNumeric

UTCTimeNumeric(Parameter)

Returns an array of numbers encoding the universal coordinated time (UTC) and date:

- Seconds
- Minutes
- Hours
- Day of the month (1 to 31)
- Month (1 to 12)
- Year
- Day of the week (1 to 7)
- Day of the year (1 to 366)
- Daylight saving time in use (0 or 1)

### LocalTimeText

LocalTimeText(Parameter)

Returns a text string of the local time and date.

## Interpolation Functions

The interpolation functions can be used, in with the **!TimeOrigin** accessor, to align multiple kinetic data sets. See [!TimeOrigin on page 87](#).

### InterpolatedXData

InterpolatedXData(Parameter)

Takes one parameter, a list of number arrays, and returns an array of interpolated numbers. Each interpolated number is the average of the corresponding input values. The result is equivalent to using the **Average** function. See [Average on page 36](#).

Example: InterpolatedXData(!A1Lm1XVals~!A1Lm2XVals)

### InterpolatedYData

InterpolatedYData(ParameterX,ParameterY)

Takes two parameters, two lists of number arrays of the same size (X and Y), and returns a list of number arrays, with each array containing interpolated Y values.

Example: InterpolatedYData(!A1Lm1XVals~!A1Lm2XVals,!A1Lm1~!A1Lm2)

Returns the interpolated values for both wavelengths Lm1 and Lm2.

### InterpolatedYDataAtXPoints

InterpolatedYDataAtXPoints(ParameterX,ParameterY,ParameterZ)

Takes three parameters: an array of numbers X, array of numbers Y, and an array of x-interpolation points. Returns an array containing interpolated Y-values at the specified interpolation points.

Example: InterpolatedYDataAtXPoints(!A1Lm1XVals,!A1Lm1,InterpolatedXData(!A1Lm1XVals~!A1Lm2XVals))

Returns the interpolated values for wavelength Lm1.

## Other Functions

Other functions let you access individual items within lists of numbers, to eliminate portions of lists from the calculation, and to take the difference (or delta) between time points of a Kinetic run. These functions are generally used in conjunction with other functions and accessors.

### Count

Count(Parameter)

Returns the count of items in a list of numbers.

```
Count(list) = 5
```

This function does not count empty or error values in the list.

### Delta

Delta(Parameter)

Returns a list of the deltas (differences) between adjacent points in a list.

For a list named 'Values', the formula is:

```
Delta(Values)
```

### FirstArray

FirstArray(Parameter)

Reports the first array in a list of arrays. The returned values can be numbers, text, or booleans.

```
FirstArray(!WellM1)
```

### FirstItem

FirstItem(Parameter)

Reports the first item in a list or array of numbers

For example, the first OD in a Kinetic run, the first OD in a Spectrum scan, the first value in a column of numbers.

```
FirstItem(!CombinedPlot)
```

### Index

Index

No parameters.

Returns an index for the samples in a group, going from 1 to n, for n samples.

## IndexListFor

IndexListFor(Parameter)

Results in an index for a list or array of numbers. For example, for the list 123, 125, 130, 127... the index is 1=123, 2=125, 3=130, 4=127...)

IndexListFor(Values)

The IndexListFor function should be applied to arrays only. To generate a group column of sample indices, use the Index function. See [Index on page 63](#).

## IndexofMax

IndexofMax(Parameter)

Reports the index of the maximum of a list or array of numbers.

IndexofMax(Results)

## IndexofMin

IndexofMin(Parameter)

Reports the index of the minimum of a list or array of numbers.

IndexofMin(Results)

## IndexofNearest

IndexofNearest(Parameter1, Parameter2)

Parameter1 is a list or array of numbers. Parameter2 is a numerical value.

The function returns the index of the number in Parameter 1 that is closest to the value of Parameter2.

IndexofNearest(!CombinedPlot, 5)

## Item

Item(Parameter1, Parameter2)

Parameter1 is a list of numbers. Parameter2 is a numerical index value.

The function returns the value of the specified item in the list.

Item(Values,2)=25



## ItemCount

ItemCount(Parameter)

Returns the count of items (or number of items) in a list or array of numbers, text items, or booleans.

ItemCount(Values)

This function does not count empty or error values in the list.

## ItemIndex

ItemIndex(Parameter)

This function has been deprecated to `IndexListFor(Parameter)`. See [IndexListFor](#) on page 64.

## NearestTo

NearestTo(Parameter1, Parameter2)

Parameter1 is a list or array of numbers. Parameter2 is a numerical value.

The function returns the value of the number in Parameter 1 that is closest to the value of Parameter2.

NearestTo(Values, .5)

## NthArray

NthArray(Parameter1, N)

Parameter1 is a list of arrays. N is a number.

The function returns the 'Nth' array in Parameter1. The array can contain numbers, text, or booleans.

NthArray(!WellLm1, 8)

## NthItem

NthItem(Parameter1, N)

Parameter1 is a list or array of numbers. N is a number.

The function returns the value of the 'Nth' item in Parameter1.

NthItem(!Lm1, 50)

## NullBetween

NullBetween(Parameter1, Parameter2, Parameter3)

Omits selected items in a list or array of numbers and sets their value to the “empty” NAN. Parameter1 is the list or array of numbers to be acted on, Parameter2 is a number that indicates at which item in the list or array to start to omit, and Parameter3 is a number that indicates at which item in the list or array to stop to omit.

Items between the points indicated by parameters 2 and 3 will be omitted.

NullBetween(!WellM1, 50, 100)

## NullOutside

NullOutside(Parameter1, Parameter2, Parameter3)

Omits selected items in a list or array of numbers by setting their value to the “empty” NAN. Parameter1 is the list or array of numbers to be acted on, Parameter2 is a number indicating at which item in the list or array to stop omitting, and Parameter3 is a number indicating at which item in the list or array to resume omitting.

Items not between the points indicated by parameters 2 and 3 will be omitted.

NullOutside(!WellM1, 50, 100)

## Sideways

Sideways(Parameter)

In a Group section, takes lists or arrays of numbers that would normally be reported in a vertical column and reports them in a horizontal row.

Sideways(!Timerun)

## Other Functions Examples

The following examples show how you can use other functions.

### Example: Using NullBetween and NullOutside to Show Selected Portions of Kinetic Data

You can use the NullBetween and NullOutside functions to create new columns containing selected data, and then plot the new columns in a Graph section and determine the slope of the data there.

For example, you can retrieve selected data from wells A1, A2, and A3 as follows:

WellA1: NullOutside(!A1Lm1@Plate#1, 10, 20)

WellA2: NullOutside(!A2Lm1@Plate#1, 15, 20)

WellA3: NullBetween(!A3Lm1@Plate#1, 10, 34)

The NullOutside function excludes items in between the second and third parameters, but does not include the wells defined by the parameters. Therefore, Column WellA1 includes points 11 through 19 only, while WellA2 includes points 16 through 19 only.

The NullBetween function excludes all items in the list between and including the second and the third parameters. Therefore, Column WellA3 excludes points 10 through 34.

In this example, the columns are reporting data directly from the wells in the Plate section rather than using the “template order” normally used in Group sections. Use formulas of this type with caution.

## Example: Using IndexOfMax with NullOutside to Determine Multiple Peaks in a Spectrum Scan

To look for three peaks in a Spectrum scan, you can create three column formulas named Peak1, Peak2, and Peak3. The formulas for the columns named Peak1, Peak2, and Peak3 are the same except for the second and third parameters of the NullOutside function. The formula for the column named Peak1 is:

$$(\text{IndexOfMax}(\text{NullOutside}(!\text{WellLm1}, 20, 40)) - 1) * !\text{StepSweep}@!\text{Plate}\#1 + !\text{StartSweep}@!\text{Plate}\#1$$

The formula reports the index of the maximum item in the list of numbers named !WellLm1 that contains an array of all Spectrum scan optical densities from each well, reported in template order).

For each list included in the array, the formula ignores everything below item 20 (that is, the OD at the 20th wavelength in the scan) and above item 40 (that is, the OD at the 40th wavelength in the scan). After IndexOfMax is calculated, the formula subtracts 1 and then multiplies the result by the sweep increment (!StepSweep, specified in the Settings dialog) and adds the wavelength at which the Spectrum scan started (!StartSweep, also specified in the Settings dialog).

The !StepSweep and !StartSweep accessors are discussed in detail in [Kinetic and Spectrum Data Accessors on page 84](#).

## Example: Calculating the First Derivative of a Kinetic Plot

Generally, when calculating the reaction rate of a Kinetic run (Vmax), you are determining the slope of the line that results from plotting change in OD versus Time. However, in some situations you might want to calculate the slope of the line that results when you plot the delta (that is, the rate of change) in the ODs versus Time. The slope of this line is the first derivative of the Kinetic plot, and can be calculated as follows:

$$\text{Vmax}(\text{Delta}(!\text{KinPlot}), !\text{VmaxPoints}, !\text{ReadInterval})$$

## NANs and MakeErrs

SoftMax Pro Software contains a special class of numbers called NANs (Not A Number) that are used to report errors and special values. NANs are special because, although they look like text, they are actually numbers. Because NANs are numbers, numerical calculations can be done on lists or arrays of numbers that contain NANs.

NANs propagate through most operations and functions in SoftMax Pro Software.

SoftMax Pro Software uses some of the NANs to report errors and other messages in Plate sections, Cuvette Set sections, Group sections, and Graph sections.

Other NANs are used only to create user-defined custom formulas.

SoftMax Pro Software supports the following NANs as MakeErr codes.

**Table 3-1: NAN Class MakeErr Codes**

| NAN          | Returns  |
|--------------|--|
| MakeErr(101) | ""<br>An empty number.   |
| MakeErr(102) | "Name?"  |
| MakeErr(103) | "Masked"<br>Usually seen in a Group section column when wells in a Plate section or Cuvette Set section have been masked.  |
| MakeErr(104) | "NoFit"<br>Usually seen in Graph section when no fit has been chosen from the list of curve fits.  |
| MakeErr(105) | "FitError"<br>Usually seen in Graph section when is unable to apply the chosen curve fit to the data set.  |
| MakeErr(106) | "Range?"   |
| MakeErr(107) | "?????"<br>For use in custom formulas.   |
| MakeErr(108) | "Error"  |
| MakeErr(109) | "Domain"<br>Internal program use.  |
| MakeErr(110) | "PrLoss"<br>Internal program use.  |
| MakeErr(111) | "Path?"<br>Seen in Plate sections when the PathCheck® Technology is selected and the pre-read has been done, but the normal read has not yet been done. Also seen if was unable to do the PathCheck calculation after the normal read. |
| MakeErr(112) | "Open?"<br>Cuvette door was open.  |

**Table 3-1: NAN Class MakeErr Codes (continued)**

| NAN          | Returns                                    |
|--------------|--|
| MakeErr(113) | “Limits-“<br>Out of reduction limits.      |
| MakeErr(114) | “Limits+“<br>Out of reduction limits.      |
| MakeErr(115) | “Pass“<br>For use in custom formulas.      |
| MakeErr(116) | “Fail“<br>For use in custom formulas.      |
| MakeErr(117) | “High“<br>For use in custom formulas.      |
| MakeErr(118) | “Low“<br>For use in custom formulas.       |
| MakeErr(119) | “>>>>“<br>For use in custom formulas.      |
| MakeErr(120) | “<<<<<“<br>For use in custom formulas.     |
| MakeErr(121) | “+++++“<br>For use in custom formulas.     |
| MakeErr(122) | “- - - - -“<br>For use in custom formulas. |
| MakeErr(123) | “*****“<br>For use in custom formulas.     |
| MakeErr(124) | “&&&&“<br>For use in custom formulas.      |
| MakeErr(125) | “#Low“<br>For use in custom formulas.      |
| MakeErr(126) | “#Sat“<br>For use in custom formulas.      |
| MakeErr(128) | “Negative“<br>For use in custom formulas.  |
| MakeErr(129) | “Positive“<br>For use in custom formulas.  |

## NANs Manipulation Functions

The following functions can be used to manipulate NANs.

### IsEmpty

IsEmpty(Parameter)

Returns true if a number, list of numbers, or array of numbers is empty. Otherwise returns false.

### IsErr

IsErr(Parameter)

Returns true if a number, list of numbers, or array of numbers is a NAN. Otherwise returns false.

### NoNum

NoNum

Takes no parameters.

Returns an “empty” number. It is the same as MakeErr(101).

### WhatErr

WhatErr(Parameter)

Returns which error the NAN represents or 0 if the number, list of numbers, or array of numbers is not a NAN.

## NANs and MakeErrs Examples

The following examples show how you can use other functions.

### Example: Using NANs in a Plate Section

You can use NANs in custom reduction formulas in the Reduction dialog for a Plate section. For example, you can use the following conditional statement that returns the NAN MakeErr(118) (“Low”) if the optical density in a well is below the optical density in well A12, to return the NAN MakeErr(117) (“High”) if the optical density is higher than the OD in well H1, and to report the optical density if it is between the ODs in wells A1 and H1:

```
If(!Lm1 < !A12, MakeErr(118), (If (!Lm1>!H1, MakeErr(117), !Lm1)))
```

## Example: Using NANs to Create a Pass/Fail Column

The following examples use NANs in formulas. NANs are rarely, if ever, used by themselves. However, frequently they are used as part of conditional statements.

Using NANs lets you combine text with numbers in a column. This example creates a column that reports either the optical density or “Fail”, depending on the value of the optical density. Since treats NANs as if they are numbers, further calculations can be done on this column, even though it looks like it contains strings. The formula is a simple conditional statement:

```
If (Values >= MeanValue-(0.2*MeanValue) and Values <= MeanValue+ (0.2*MeanValue),
    Values, MakeErr(116))
```

## Text Functions

The following functions let you manipulate text strings in SoftMax Pro Software.

### ASCII

```
ASCII("Text")
```

Returns the ASCII value for the first character in the text string.

```
ASCII("Average") = 65
```

### Concat

```
Concat("Text1", "Text2", ... , "TextN")
```

Concatenates a series of text strings together.

```
Concat("SoftMax", " Pro ", " Software") = SoftMax Pro Software
```

This can also be done using the + operator. For more about the + operator, see [+ for Addition on page 16](#).

### Exact

```
Exact("Text1", "Text2")
```

Returns true if the two text strings are identical and false if they are not.

```
Exact("Softmax Pro", "SoftMax Pro") = False,
```

```
Exact("SoftMax Pro", "SoftMax Pro") = True.
```

This function is case sensitive. To do a comparison that is not case sensitive, use the “=” operator. See [= for Equals on page 18](#).



## Left

Left("Text",N)

Returns the first N characters from the left end of a text string.

Left("SoftMax Pro", 4) = "Soft"

## Len

Len("Text")

Returns the length of (or number of characters in) the text string.

Len("Molecular Devices") =17

## Lower

Lower("Text")

Returns the text in all lower-case characters.

Lower ("Molecular Devices") = "molecular devices"

## Mid

Mid("Text", N1, N2)

Returns the specified number of characters (N2) including the specified start character (N1).

Mid("SoftMax Pro", 5, 3) = "Max"

## Now

Now

Takes no parameters. Returns the current time of day.

Every time you open or recalculate the file, Now is updated to the current time.

Now = 9:05:30 AM

## Right

Right("Text",N)

Returns the first N characters from the right end of a text string.

Right("SoftMax Pro", 3) = "Pro"

## Text

Text(NumericalParameter)

Converts numbers to text. Returns a number, list of numbers, or array of numbers as text strings.

Text(26.00) = "26"

## Today

Today

Returns the current day and date.

Every time you open or recalculate the file, Today is updated to the current day and date.

Today = Friday, February 14, 2014

## Upper

Upper("Text")

Returns the text in all uppercase characters.

Upper ("SoftMax Pro") = "SOFTMAX PRO"

## Val

Val("Text")

Returns a number that corresponds to the numerical value of the start of the text string. If the start of the string is not a number, it returns 0.

Val("24") = 24.00

Val("24.4, 4asfg") = 24.4

Val("abcd24.4") = 0.0

## Text Functions Example

The following examples show how you can use other functions.

### Example: Combining Text with Numbers to Report a Results Column

In some situations, you might want to flag certain results in a column and need to have the flag and the numerical results in the same column. One way to do this is to convert the numbers to text.

Keep in mind that after the SoftMax Pro Software converts numbers to text, no further calculations can be done on the numbers.

You can construct a new column from the column "Adj.Result" as follows:

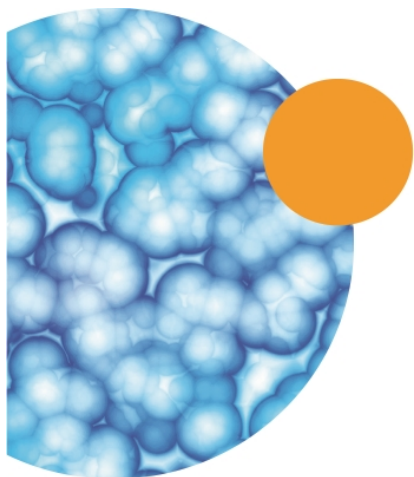
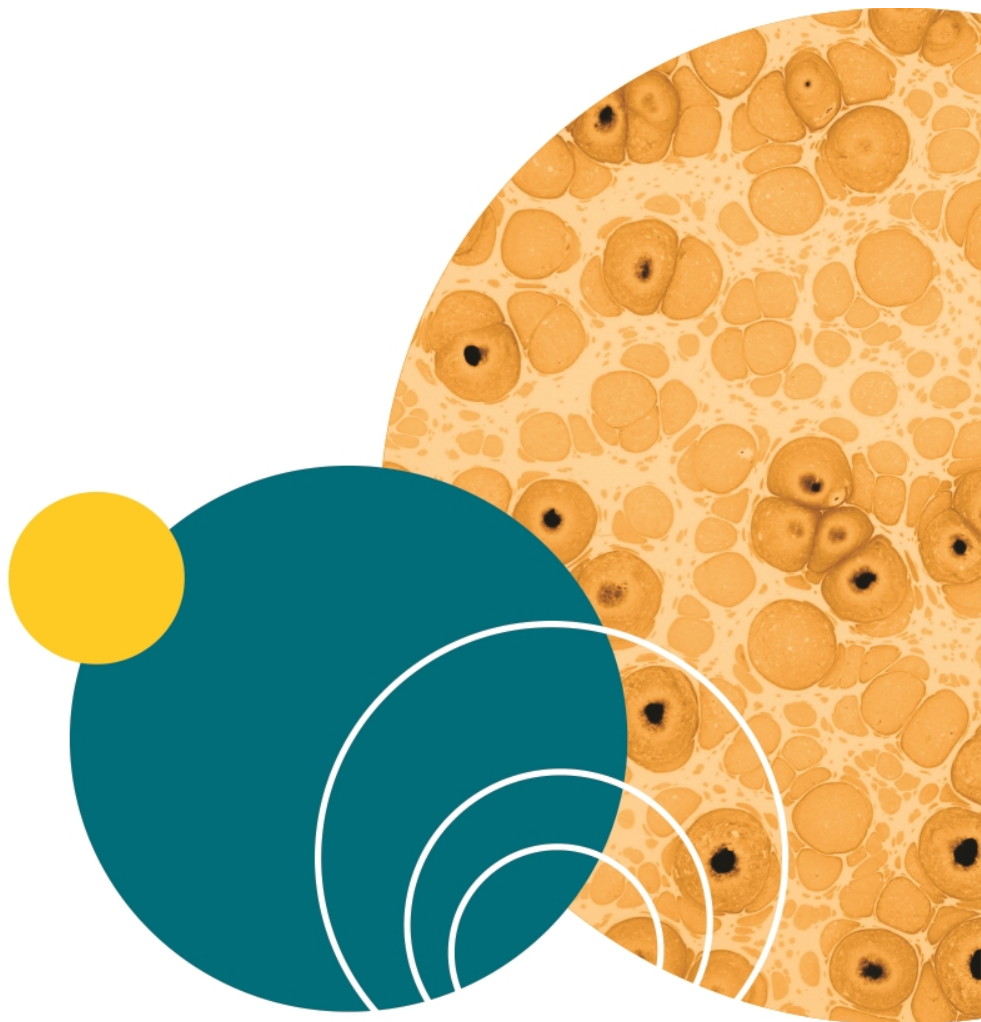
```
If (Adj.Result<0, "****" + Left((Text(Adj.Result)), 6) + "****", Left((Text(Adj.Result)), 6))
```

The conditional statement specifies:

If the adjusted result is less than zero, report **\*\*\*\*text string\*\*\*\***.

If not, report text string.

In the two cases, the text string is defined as `Left((Text(Adj.Result)), 6)`, because when a column of numbers is turned into text, the numbers are reported to a large number of decimal places. Therefore, the formula uses the `Left` function to limit the display of the text string to the six left-most characters.



## Chapter 4: Accessors

Accessors are special functions that provide access to other specific information in the program. For example, the number of Vmax Points used to calculate the Vmax Rate.

Unlike functions, accessors do not have parameters related to them. Accessors are always preceded by an exclamation point (!, also known as a “bang”).

Accessors are frequently used by themselves as column formulas. They are also used as parameters within several functions, most notably kinetic and spectrum reduction functions. Accessors can be combined with functions and mathematical operators to create complex formulas.

Accessors can be divided into the following categories:

- [Plate and Cuvette Set Setup Information Accessors on page 78](#)
- [Kinetic and Spectrum Data Accessors on page 84](#)
- [Plate Data Accessors on page 91](#)
- [Injector Data Accessors on page 98](#)
- [PathCheck Technology Accessors on page 101](#)
- [Group and Well Information Accessors on page 103](#)
- [Blank Accessors on page 105](#)
- [Imaging Data Accessors on page 107](#)

### Using !Well as an Accessor Prefix

Accessors can refer to the information in **Plate** sections and **Cuvette Set** sections in two ways: in read order (well A1, A2, A3, A4, ...) or in template or group order (wells A1, B1, A2, C2 on Plate#1 and wells G1, H1, G2, H2 on Plate #2 if they are part of the same group).

To report the data in read order use a well specification prefix, such as !A1:

`!A1Accessor`

To report the data in group order, use !Well as the prefix:

`!WellAccessor`

The !Well prefix is used only in **Group** sections. Using an accessor without the !Well prefix is almost never used in **Group** section tables.



**Note:** The !Well prefix is not valid with group and well information accessors. See [Group and Well Information Accessors on page 103](#).

---

## Plate and Cuvette Set Setup Information Accessors

**Plate** section and **Cuvette Set** section information accessors provide information about **Plate** section or **Cuvette Set** section settings.

SoftMax Pro Software supports the following Plate and Cuvette Set setup information accessors.

### **!AutoCutoffOn**

!AutoCutoffOn

!WellAutoCutoffOn

For Fluorescence mode only.

Returns True if the emission cutoff filter is set to Automatic in the Settings dialog and returns False if the emission cutoff filter is set to manual.

### **!AvgReadTemperature**

!AvgReadTemperature

!WellAvgReadTemperature

Returns, as a number, the average temperature measured while the plate or cuvette was read.

### **!EmCutoffs**

!EmCutoffs

!WellEmCutoffs

For Fluorescence mode only.

Reports the cutoff filter wavelengths set in the Settings dialog for Endpoint, Kinetic, and Spectrum reads.

### **!EmFixedWavelength**

!EmFixedWavelength

!WellEmFixedWavelength

For Fluorescence mode only.

Reports the wavelength at which emission is fixed when performing an excitation scan. If the emission wavelength is not fixed, no number is reported.

## **!EmWavelengths**

!EmWavelengths

!WellEmWavelengths

For Fluorescence mode only.

Reports the emission wavelengths chosen in the Settings dialog for Endpoint, Kinetic, and Spectrum reads.

All (for example, white light) is the default emission wavelength for luminescence assays and is returned unless a specific emission wavelength is chosen.

## **!ExFixedOn**

!ExFixedOn

!WellExFixedOn

For Fluorescence mode only.

Reports True if the instrument setting is set to do a Spectrum scan with fixed excitation and variable emission wavelengths.

## **!ExFixedWavelength**

!ExFixedWavelength

!WellExFixedWavelength

For Fluorescence mode only.

Reports the wavelength at which excitation is fixed when performing an emission scan. If the excitation wavelength is not fixed, no number is reported.

## **!ExWavelengths**

!ExWavelengths

!WellExWavelengths

For Fluorescence mode only.

Reports the excitation wavelengths specified in the Settings dialog for Endpoint, Kinetic, and Spectrum reads.

## **!FirstColumn**

!FirstColumn

!WellFirstColumn

Returns, as a number, the first column read in a specified Plate section.

This accessor is useful when reading only selected columns on a plate.

## **!Gfactor**

!Gfactor

Returns the current Gfactor value set in the Data Reduction dialog.

## **!IntegrationEnd**

!IntegrationEnd

!WellIntegrationEnd

For Time Resolved Fluorescence mode only.

Returns, as a number, the integration end time.

## **!IntegrationStart**

!IntegrationStart

!WellIntegrationStart

For Time Resolved Fluorescence mode only.

Returns, as a number, the integration start time.

## **!LastColumn**

!LastColumn

!WellLastColumn

Returns, as a number, the last column read in a specified Plate section.

This accessor is useful when reading only selected columns on a plate.

## **!NumWells**

!NumWells

!WellNumWells

For Fluorescence mode only.

Returns the number of wells in the plate type specified in the Settings dialog.



## !PlateName

!PlateName

!WellPlateName

Returns the name of the Plate section or Cuvette Set section as a text string.

This accessor is very useful when Group sections contain samples from multiple plates in which the samples are in the same wells, since it reports which plate the sample was in.

## !PMTSetting

!PMTSetting

!WellPMTSetting

For the SpectraMax M2, M2e, M3, M4, M5, and M5e Instruments in Fluorescence mode only.

Returns the text string “Automatic”, “High”, “Medium”, or “Low” depending on the PMT Setting chosen in the Settings dialog.

## !PreMixDuration

!PreMixDuration

!WellPreMixDuration

Returns the number of seconds Automix has been set to mix before reading in the Settings dialog.

For Kinetic runs, it reports the Automix duration before the first read. It does not report the Automix duration in between reads. See [Kinetic and Spectrum Data Accessors on page 84](#).

## !PreMixOn

!PreMixOn

!WellPreMixOn

Returns True if Automix has been turned on in the Settings dialog and False if Automix has not been turned on.

For Kinetic runs, this applies to Automix before the first read. For Automix between reads, see [Kinetic and Spectrum Data Accessors on page 84](#).

## **!ReadType**

!ReadType

!WellReadType

Not available on all instruments.

Returns the text string “Absorbance”, “Fluorescence”, “Luminescence”, or “Time Resolved”, depending on the instrument setup.

## **!SectionName**

!SectionName

Returns the name of the section containing the formula.

## **!TemperatureSetPoint**

!TemperatureSetPoint

!WellTemperatureSetPoint

Returns, as a number, the incubator set point (from the Settings dialog) at the time the plate or cuvette was read.

## **!TimeOfRead**

!TimeOfRead

!WellTimeOfRead

Returns the time and date the plate or cuvette was read.

This information can be found in the time and date stamp displayed in Plate section or Cuvette Set section after a reading.

## **!TimeResolvedOn**

!TimeResolvedOn

!WellTimeResolvedOn

For Time Resolved Fluorescence mode only.

Returns True if the Read Mode in the Settings dialog is set to Time Resolved. Returns False if the Read Mode is not set to Time Resolved.

## **!TransferRate**

!TransferRate1, !TransferRate2, !TransferRate3

!WellTransferRate1, !WellTransferRate2, !WellTransferRate3

For the FlexStation 3 reader only.

Returns the transfer rate in microliters per second for the corresponding compound transfer used with a Plate section.

## !TransferTime

!TransferTime1, !TransferTime2, !TransferTime3

!WellTransferTime1, !WellTransferTime2, !WellTransferTime3

For the FlexStation 3 reader only.

Returns the time points for the corresponding compound transfer used with a Plate section.

## !TransferVolume

!TransferVolume1, !TransferVolume2, !TransferVolume3

!WellTransferVolume1, !WellTransferVolume2, !WellTransferVolume3

For the FlexStation 3 reader only.

Returns the volume in microliters for the corresponding compound transfer used with a Plate section.

## !Wavelengths

!Wavelengths

!WellWavelengths

Returns as a text string (for example, "405 650") the wavelengths at which a Plate section or Cuvette Set section was read.

## Plate and Cuvette Set Setup Information Accessors Example

The following example shows how you can use plate setup information accessors.

### Example: Using Plate Information Accessors in Summary Lines

If a data file contains data from more than one plate, you can report the start and end time of the experiment using !TimeOfRead. If the first plate read was PlateX and the last plate was PlateY, the following two formulas return the start and end time, respectively:

```
!TimeOfRead@PlateX
```

```
!TimeOfRead@PlateY
```

The information is reported directly from the Plate sections, so "Well" is not used in the accessors.

## Kinetic and Spectrum Data Accessors

The following accessors are specific to Kinetic and Spectrum scan data. They let you:

- Access information about the run and customize analysis of Kinetic or Spectrum data.
- Manipulate the Y-axis information (OD) and the X-axis information (time for Kinetic runs, wavelength for Spectrum scans).

SoftMax Pro Software supports the following kinetic and spectrum data accessors.

### **!AutomixDuration**

!AutomixDuration

!WellAutomixDuration

Returns the number of seconds set for AutoMix between Kinetic Reads as specified in the Settings dialog.

### **!AutomixOn**

!AutomixOn

!WellAutomixOn

Returns a text string of True if Automix between reads is checked for Kinetic plates in the Settings dialog and False if it is not checked.

### **!EndLimit**

!EndLimit

!WellEndLimit

Returns the end time set for Kinetic run data analysis and the end wavelength set for Spectrum scan data analysis. They are set in the Reduction dialog.

Data collected after the end time or above the end wavelength are reported with the NAN MakeErr(114) "Limits+" and are not used in reductions.

### **!StartLimit**

!StartLimit

!WellStartLimit

Returns the lag time in seconds for Kinetic run data analysis and the start wavelength for Spectrum scans. They are set in the Reduction dialog.

Data collected before the lag time or below the starting wavelength are reported with the NAN MakeErr(113) "Limits-" and are not used in reductions.

## **!EndSweep**

!EndSweep

!WellEndSweep

Returns, as a number, the wavelength at which to end a Spectrum scan as specified in the Settings dialog.

## **!StartSweep**

!StartSweep

!WellStartSweep

Returns, as a number, the wavelength at which to start a Spectrum scan as specified in the Settings dialog.

## **!StepSweep**

!StepSweep

!WellStepSweep

Returns, in nm, the step size of a Spectrum scan as specified in the Settings dialog.

## **!MaxLimit**

!MaxLimit

!WellMaxLimit

Returns, as a number, the Max OD set in the Reduction dialog for a Spectrum scan or Kinetic run.

Data collected above the Max OD are reported with the NAN MakeErr(114) "Limits+" and are not used in reductions.

## **!MinLimit**

!MinLimit

!WellMinLimit

Returns, as a number, the Min OD set in the Reduction dialog for a Spectrum scan or Kinetic run.

Data collected below the Min OD are reported with the NAN MakeErr(113) "Limits-" and are not used in reductions.

## **!NumPoints**

!NumPoints

!WellNumPoints

Returns the number of time points to be collected in a Kinetic run or the number of wavelengths to be read in a Spectrum scan as specified in the Settings dialog.

## **!NumPointsRead**

!NumPointsRead

!WellNumPointsRead

For the SpectraMax Plus, SpectraMax 190, SpectraMax 340 PC, VersaMax, and SpectraMax Gemini readers only.

Returns the actual number of time points during which data was collected.

This accessor is very useful if you have stopped a Kinetic run sooner than was specified in the Settings dialog.

This accessor returns numbers and NANs. It might not detect missing data points in a Kinetic run or Spectrum scan due to low light. If you want to detect missing data points, use the [ItemCount](#) function. See [ItemCount on page 65](#).

## **!NumWavelengthsRead**

!NumWavelengthsRead

!WellNumWavelengthsRead

Returns the actual number of wavelengths that are read in an Endpoint run or Spectrum scan.

This accessor is very useful when you have stopped a Spectrum scan before the end wavelength is read, as selected in the Settings dialog.

The information reported by this accessor is invalid for Endpoint readings if PathCheck Technology is selected in the Settings dialog.

## **!OnsetValue**

!OnsetValue

!WellOnsetValue

Previously called !OnsetOD.

Returns, as a number, the onset value (OD, RFU, RLU) specified in the Reduction dialog to be used in calculating Onset Time if the reduction is set to Onset Time.

## !ReadInterval

!ReadInterval

!WellReadInterval

Returns the number of seconds specified in the Settings dialog for a Kinetic run read interval.

## !RunTime

!RunTime

!WellRunTime

Returns the time, in seconds, that it will take to read a Kinetic plate, as calculated from the information in the Settings dialog.

If the run is terminated before it can be completed, this accessor still reports the calculated time to finish the run, not the actual run time. However, !NumPoints can be used to calculate the actual number of data points collected. See [!NumPoints on page 86](#).

## !TemperatureRun

!TemperatureRun

!WellTemperatureRun

Returns a list of numbers that contains temperature data collected during the read. One temperature is reported for all wells at each time point.

This accessor can be used for Endpoint reads as well as for Kinetic runs and Spectrum scans. It is very useful for plotting the effects of temperature on your data.

## !TimeOrigin

!LmXTimeOrigin

!A1LmXTimeOrigin

!WellTimeOrigin

For a kinetic data set for the specified wavelength, this accessor returns the number of seconds since the “beginning of time” when the first measurement was recorded. The “beginning of time” is not specified. Generally, this accessor is used to determine the difference in time between two measurements, allowing different data sets to reference a common time origin.

## !AllTimeOrigins

!LmXAllTimeOrigins

For a kinetic data set for the specified wavelength, this accessor returns as an array the time origins for all wells in a plate.

## **!TimeRun**

!TimeRun

!WellTimeRun

!TimeRun returns a list of read times in a vertical column.

Although the individual wells of a 96-well plate are read fractions of a second apart during each time point in a Kinetic read, the same time point is reported for all wells. For example, if a series of wells is read at 9, 9.1, and 9.2 seconds, and then read again at 12, 12.1, and 12.2 seconds, the !TimeRun accessor returns 9 and then 12 for the read times.

All of the cuvettes in a Cuvette Set section also report the same time points because, although the Kinetic read on each cuvette is started at a different time, the accessor reports the elapsed time during the read.

## **!VmaxPoints**

!VmaxPoints

!WellVmaxPoints

Returns the number of points that would be used to calculate VMax if the reduction were set to VMax.

The accessor reports the number of VMax Points set in the Reduction dialog. The default is total number of time points collected during a Kinetic run.

To determine the actual number of VMax Points used in each well, use the VMaxPtsUsed reduction function. See [VmaxPtsUsed on page 54](#).

## **!WavelengthRun**

!WavelengthRun

!WellWavelengthRun

Returns, as a list of numbers, the wavelengths at which each of the Spectrum scan data points are collected.

This accessor is valid for Spectrum scans only.

## **!XVals**

!XVals

Returns the values related to the X-axis of a Kinetic or Spectrum read.

This is the same as !TimeRun for Kinetic data or !WavelengthRun for Spectrum data. See [!TimeRun on page 88](#) or [!WavelengthRun on page 88](#).



## Kinetic and Spectrum Data Accessors Examples

The following example shows how you can use plate setup information accessors.

### Example: Accessing Kinetic Run Settings Information

The !NumPoints and !NumPointsRead accessors can be used to determine the number of points specified for a Kinetic run and the number of points actually collected if the run is stopped early. The same information can be determined for a Spectrum scan by using the !NumPoints and !NumWavelengthsRead accessors.

You can create a column to calculate the actual number of Vmax Points used in each well to calculate the rate of the reaction with the formula:

```
VmaxPtsUsed(!WellLm1, !VmaxPoints@Plate#1, !ReadInterval@Plate#1)
```

You can also create Summaries to report the original settings and the actual number of readings.

Number of time points configured in Instrument Settings:

```
!NumPoints
```

Number of time points collected during Kinetic run:

```
!NumPointsRead
```

Run time configured in Instrument Settings:

```
!RunTime
```

Actual run time during Kinetic run:

```
NthItem(!TimeRun, !NumPointsRead)
```

This last formula returns the time at which the last data point was collected (that is, the run time).

### **Example: Summarizing Spectrum Scan Parameters Using Summaries**

Starting wavelength of the Spectrum scan:

!StartSweep

Ending wavelength of the Spectrum scan:

!EndSweep

Step increment in nm of scan:

!StepSweep

Starting wavelength of data analysis:

!StartLimit

End wavelength of data analysis:

!EndLimit

If more than one Plate section or Cuvette Set section is present in the Experiment section you need to identify the Plate section or Cuvette Set section in the formulas; for example, !StartSweep@Plate#1 or !EndSweep@CuvetteSet#1.

### **Example: Temperature, Time, and Wavelength Information from Kinetic Plots/Spectrum Scans**

If the Well prefix is used with the !TemperatureRun, !TimeRun, and !WavelengthRun accessors in Group sections, the lists of numbers are reported in the Group section as a horizontal array. When information is displayed in this fashion, you can do further mathematical manipulations on the array of numbers but you cannot plot it in a Graph section.

You can also access temperature, time, and wavelength information in a Group section independently of the template. If you do so, the information will be placed in a column, and you can graph optical density versus temperature, time, or wavelength; time versus temperature; or wavelength versus temperature.

## Plate Data Accessors

Plate data accessors let you access optical density (raw data), reduced numbers, pre-read data, and pathlength correction information from Plate sections, Cuvette Set sections, and Group sections. These accessors can be used in Plate sections, Cuvette Set sections, and Group sections.

SoftMax Pro Software supports the following plate data accessors.

### **!AllValues**

!AllValues

Returns a list of the reduced numbers from all wells in a Plate section or all cuvettes in a Cuvette Set section.

### **!CombinedPlot**

!CombinedPlot

!WellCombinedPlot

This accessor can be used in Plate sections or Cuvette Set sections, for Kinetic runs and Spectrum scans.

Returns the result of the wavelength combination reduction formula for each well or cuvette.

!LmX or !WellLmX return information before the wavelength combination reduction is applied. See [!LmX on page 91](#).

### **!LmX**

!LmX

!A1LmX

!WellLmX

Reports as a number, in Endpoint readings, or list of numbers, in Kinetic/Spectrum scans, the raw OD values, before the wavelength combination reduction is applied, from wells or cuvettes.

LmX specifies the wavelength. For example, Lm1, Lm2, and so on.

!LmX accesses the raw values from all wells or cuvettes in the section at the given wavelength.

!A1LmX accesses the raw value from the specified well or cuvette at the specified wavelength. For example, !H2Lm1 or !B12Lm3.

!WellLmX accesses the raw optical densities reported in a Group section in template order. Optical densities at each Kinetic run time point or each Spectrum scan wavelength are reported in a horizontal array.

## **!LmXPRaw**

!LmXPRaw

!WellLmXPRaw

For Fluorescence Polarization readings only.

Reports the raw values, before reductions are applied, from wells or cuvettes for parallel polarized data of a specific wavelength (for example, Lm2 or 2nd wavelength set) for a given template group in a Group section.

!WellLmXPRaw accesses the raw values reported in a Group section in template order.

## **!A1LmXPRaw**

!A1LmXPRaw

For Fluorescence Polarization readings only.

Returns parallel, or an array of parallel, data for the given well (for example, A2) and specified wavelength (for example, Lm1 or 1st wavelength set).

## **!LmXSRaw**

!LmXSRaw

!WellLmXSRaw

For Fluorescence Polarization readings only.

Reports the raw values, before reductions are applied, from wells or cuvettes for perpendicular polarized data of a specific wavelength (for example, Lm1 or 1st wavelength set) for a given template group in a Group section.

!WellLmXSRaw accesses the raw values reported in a Group section in template order.

## **!A1LmXSRaw**

!A1LmXSRaw

For Fluorescence Polarization readings only.

Returns perpendicular, or an array of perpendicular, data for the given well (for example, A2) and specified wavelength (for example, Lm1 or 1st wavelength set).

## !LmXXVals

!LmXXVals

!A1LmXXVals

!WellLmXXVals

Reports as a list of numbers, in Fast Kinetic scans, the actual read time (time-tagged) values from wells.

LmX specifies the wavelength. For example, Lm1, Lm2, and so on.

!LmXXVals returns the read time (time-tagged) for the specified well at the indicated wavelength.

!WellLmXXVals accesses the actual read time reported in a Group section in template order. The read time (time-tagged) data is reported in a horizontal array.

## !MValue

!MValue

Not available on all instruments.

Returns the reduced number from each well of a Plate section or cuvette in a Cuvette Set section.

!MValue is returned as the default for readings taken after an injection.

!PValue is returned as the default for readings taken after a P-injection. See [!PValue on page 94](#).

## !PlateBlankLmX

!PlateBlankLmX

Returns the raw value for the given wavelength LmX in a Plate section.

## !PlateBlankP

!PlateBlankP

Returns parallel (or P) plate blank raw values for the given wavelength LmX in a Plate section.

!PlateBlankLm1P

### **!PlateBlankLmXP**

!PlateBlankLmXP

Returns parallel (or P) plate blank raw values for the given wavelength LmX in a Plate section.

!PlateBlankLm1P

### **!PlateBlanks**

!PlateBlanks

Returns perpendicular (or S) plate blank raw values for the given wavelength LmX in a Plate section.

!PlateBlankLm1S

### **!PlateBlankLmXS**

!PlateBlankLmXS

Returns perpendicular (or S) plate blank raw values for the given wavelength LmX in a Plate section.

!PlateBlankLm1S

### **!PlateKind**

!PlateKind

Returns the type of plate selected in the Settings dialog.

### **!PValue**

!PValue

Not available on all instruments.

Returns the reduced number from each well of a Plate section or cuvette in a Cuvette Set section.

!PValue is returned as the default for readings taken after an P-injection.

!MValue is returned as the default for readings taken after an injection. See [!MValue on page 93](#).

## **!PValueLm1**

!PValueLm1

Returns the reduced number from each well of a Plate section or cuvette in a Cuvette Set section for the Lm1 wavelength.

!PValueLm1 is returned as the default for readings taken after a P-injection.

## **!PValueLm2**

!PValueLm2

Returns the reduced number from each well of a Plate section or cuvette in a Cuvette Set section for the Lm2 wavelength.

!PValueLm2 is returned as the default for readings taken after a P-injection.

## **!A1Reduced**

!A1Reduced

Reports the reduced value for the designated well.

## **!StdDevPlateBlank**

!StdDevPlateBlank

Returns the standard deviation of the plate blank group.

## **!StdDevPlateBlankLmX**

!StdDevPlateBlankLmX

Returns the standard deviation of the plate blank group at the specified wavelength.

!StdDeviationPlateBlankLm2 returns the standard deviation of wavelength 2 of the plate blank group.

## Plate Data Accessors Examples

The following example shows how you can use plate setup information accessors.

### Example: Dividing All Optical Densities in a Plate Section by the Optical Density in a Specified Well

You can use a custom formula in the Reduction dialog to divide all optical densities in a Plate by the optical density in one well:

```
!Lm1/!A1Lm1
```

The formula divides the optical density in all wells at wavelength Lm1 (!Lm1) by the optical density in well A1 (!A1Lm1).

### Example: Viewing Optical Densities from Endpoint Reads at Multiple Wavelengths in a Single Group Section

Several DNA samples were read in a microplate at 260, 280, and 320 nm. PathCheck Pathlength Measurement Technology was applied to normalize the optical densities to a 1 cm pathlength, and a custom wavelength reduction formula (!Lm1\*50) was entered in the Reduction dialog of the Plate section to report the quantitation of the DNA samples.

You can report optical densities at each of the wavelengths in group order with the following formulas:

```
!WellLm1
```

```
!WellLm2
```

```
!WellLm3
```

If these columns are named "OD260", "OD280", and "OD320" respectively, you can report a Ratio column to calculate the protein contamination in the DNA samples:

```
(OD260-OD320)/(OD280-OD320)
```

Lastly, you can calculate the average pathlength, in centimeters, for each sample:

```
Average(!WellPathlength)
```



### Example: Viewing the REF Values for a Cuvette Set

You can calculate a column to show the reference value used for each cuvette using the formula:

```
!WellPreReadLm1
```

Creating a column of this kind is quite useful if you have used different reference values for cuvettes in the same Cuvette Set because it lets you see the individual reference values applied to each cuvette and note the differences between them.

If you have read a Cuvette Set at more than one wavelength, you can see the reference applied at each wavelength by creating more columns using modifications of the formula above:

```
!WellPreReadLm2
```

```
!WellPreReadLm3
```

### Example: Viewing the Optical Densities From a Spectrum Scan or Kinetic Run in a Group Section

When you use !WellLmX to report the optical densities from a Kinetic run or Spectrum scan in a Group section, the information comes into the table in a horizontal array. The !TemperatureRun, !WavelengthRun, and !TimeRun accessors also place information in horizontal arrays. The SoftMax Pro Software lets you do further mathematical manipulations on the horizontal array of numbers, but you cannot plot them in a Graph section.

Whether the **Absolute Values** check box in the **Reduction** dialog is selected effects the display of optical densities from a Kinetic run in the **Group** section.

- When the check box is cleared (by default), optical densities are displayed normalized to 0 to define the first value for each well as 0.000.
- When the check box is selected, absolute optical densities are displayed and the first OD for each well is no longer zero.

If you want to graph the optical densities from a Kinetic run or Spectrum scan, you need to put them into the Group section in a column format instead of a horizontal array.

## Injector Data Accessors

Injector data accessors let you access delay timing, injector volume, integration time, and baseline read data.

SoftMax Pro Software supports the following kinetic and spectrum data accessors.

### **!DelayTime1**

!DelayTime1

!WellDelayTime1

For the reads with injection only.

Returns the delay time in seconds between the injection and the read for injector 1.

The expected values are 0.001 to 100 seconds.

- If no delay step is used, then the value is returned as 0.
- If injector 1 is not used, then the value is returned as empty.

This accessor is useful for retrieving the specified delay time for reporting purposes.

### **!DelayTime2**

!DelayTime2

!WellDelayTime2

For the reads with injection only.

Returns the delay time in seconds between the injection and the read for injector 2.

The expected values are 0.001 to 100 seconds.

- If no delay step is used, then the value is returned as 0.
- If injector 2 is not used, then the value is returned as empty.

This accessor is useful for retrieving the specified delay time for reporting purposes.

### **!Injector1Volume**

!Injector1Volume

!WellInjector1Volume

For the reads with injection only.

Returns the volume of the injection in  $\mu\text{L}$  for injector 1. The expected values are 1 to the maximum volume of the well.

This accessor is useful for retrieving the injected volume for activity calculation in a well.

If injector 1 is not used, then the value is returned as empty.

## **!Injector2Volume**

!Injector2Volume

!WellInjector2Volume

For the reads with injection only.

Returns the volume of the injection in  $\mu\text{L}$  for injector 2. The expected values are 1 to the maximum volume of the well.

This accessor is useful for retrieving the injected volume for activity calculation in a well.

If injector 2 is not used, then the value is returned as empty.

## **!IntTime1Sequence**

!IntTime1Sequence

!WellIntTime1Sequence

For the reads with injection only.

Returns the integration time in milliseconds for luminescence reads with injection, or the number of flashes for fluorescence intensity reads with injection, for injector 1. The expected values are 1 to the maximum allowed integration time or number of flashes.

This accessor is useful for retrieving the specified integration time or number of flashes for reporting purposes.

If injector 1 is not used, then the value is returned as empty.

## **!IntTime2Sequence**

!IntTime2Sequence

!WellIntTime2Sequence

For the reads with injection only.

Returns the integration time in milliseconds for luminescence reads with injection, or the number of flashes for fluorescence intensity reads with injection, for injector 2. The expected values are 1 to the maximum allowed integration time or number of flashes.

This accessor is useful for retrieving the specified integration time or number of flashes for reporting purposes.

If injector 2 is not used, then the value is returned as empty.

## !NumBaselineReads1

!NumBaselineReads1

!WellNumBaselineReads1

For the kinetic reads with injection only.

Returns the number of baseline reads before the injection for injector 1.

The expected values are 1 to 999.

- If no baseline step is used, then the value is returned as 0.
- If injector 1 is not used, then the value is returned as empty.

This accessor is useful for calculating the average of the baseline read data, and then subtracting the value from the post-injection data.

## !NumBaselineReads2

!NumBaselineReads2

!WellNumBaselineReads2

For the kinetic reads with injection only.

Returns the number of baseline reads before the injection for injector 2.

The expected values are 1 to 999.

- If no baseline step is used, then the value is returned as 0.
- If injector 2 is not used, then the value is returned as empty.

This accessor is useful for calculating the average of the baseline read data, and then subtracting the value from the post-injection data.

## Injector Data Accessors Example

The following example shows how you can use injector data accessors.

### Example: Subtracting Baseline Data from Post-Injection Data

You can use a custom formula in the **Wavelength Options** step of the **Reduction** dialog to subtract the data for the baseline reads from the data for the post-injection reads:

```
!Lm1-Average(NullBetween(!Lm1,!NumBaselineReads1+1,ItemCount(!Lm1)))
```

The formula takes the average of the baseline reads and subtracts this value from the post-injection reads in all wells at wavelength Lm1 (!Lm1).

## PathCheck Technology Accessors

PathCheck® Pathlength Measurement Technology accessors return information about the values used in the PathCheck calculation as well as information about pathlength. These accessors apply to information in the Plate section and can be used in Plate sections or in column formulas in Group sections or in Summary formulas. These accessors are not used in Cuvette Set sections.

SoftMax Pro Software supports the following PathCheck Technology accessors.

### **!PathBackground**

!PathBackground

!PathBackgroundLm1, ... ,!PathBackgroundLm6

Returns the plate background constant that has been set for the specified wavelengths.

### **!PathCheckApplied**

!PathCheckApplied

Returns True if the PathCheck Reduction setting has been applied to a Plate section.

### **!PathCheckLm900**

!PathCheckLm900

Returns a number representing either the cuvette reference value read at 900 nm or the value for the Water Constant at 900 nm, depending on whether the PathCheck instrument settings are set to use a cuvette reference or the Water Constant.

### **!PathCheckLm1000**

!PathCheckLm1000

Returns a number representing either the cuvette reference value read at 1000 nm or the value for the Water Constant at 1000 nm, depending on whether the PathCheck instrument settings are set to use a cuvette reference or the Water Constant.

### **!PathCheckOn**

!PathCheckOn

Returns True if PathCheck has been selected in the Settings dialog, and returns False if it has not.

### **!PathCorrectionOn**

!PathCorrectionOn

Returns True if the Plate section has been set to use pathlength correction, and False if it has not.

## **!Pathlength**

!Pathlength

!A1Pathlength

!WellPathlength

Returns the pathlength determined by PathCheck Technology accessors. These accessors can be applied only to Plate section data.

!Pathlength returns the pathlength for all wells in a plate.

!A1Pathlength returns the pathlength for an individual well.

!WellPathlength returns the pathlengths for samples into a Group section, reporting the information in template order.

## **PathCheck Technology Accessors Examples**

The following examples show how you can use PathCheck Technology accessors.

### **Example: Accessing PathCheck Values in a Group Section**

The formulas !WellLm900 and !WellLm1000 report the optical density readings from each well at 900 nm and 1000 nm. These readings are used in the PathCheck calculation that determines the pathlength in each well.

You can use the formulas !PathCheckLm900 and !PathCheckLm1000 to return the optical density readings for the cuvette reference at 900 nm and 1000 nm.

If the Water Constant had been used instead of a cuvette reference, these would be the Water Constant values.

### **Example: Viewing Pathlength together with Pathlength-Corrected Optical Density in Group Sections**

If Pathlength correction is enabled, the !WellValues accessor puts the pathlength-corrected optical density from the Plate section into the Group section.

The formula !WellPathlength reports the pathlength for each well in the group.

## Group and Well Information Accessors

Group information accessors return information assigned to samples within groups in the Template Editor. Several of these accessors are the default formulas for the columns that are created when you create new groups in the Template Editor.

SoftMax Pro Software supports the following group and well information accessors.

### **!ColumnFormulas**

**!ColumnFormulas**

Returns a list of the column formulas in the specified Group section.

The formulas are returned in lexicographical order according to the column name. All column formulas are returned, even if the column is hidden in the user interface. The list updates when columns are added or removed, or when referenced formulas or sections are renamed.

### **!ColumnNames**

**!ColumnNames**

Returns a list of the column names in the specified Group section.

The names are returned in lexicographical order. All column names are returned, even if the column is hidden in the user interface. The list updates when columns are added, removed, or renamed.

### **!Concentration**

**!Concentration**

Returns the numerical value assigned by the user in the Sample Descriptor field in the Template Editor, usually concentration for standards or a dilution factor for unknowns.

The sample descriptor value can be used with the numerical value you want to associate with a sample. For example, if it is one of a series of samples drawn over time, time could be used as a sample descriptor.

### **!Factor**

**!Factor**

Returns the numerical value assigned by the user in the Sample Descriptor field in the Template Editor, usually concentration for standards or a dilution factor for unknowns.

The sample descriptor value can be used with the numerical value you want to associate with a sample. For example, if it is one of a series of samples drawn over time, time could be used as a sample descriptor.

## !SampleDescriptor

!SampleDescriptor

Returns the numerical value assigned by the user in the Sample Descriptor field in the Template Editor, usually concentration for standards or a dilution factor for unknowns.

The sample descriptor value can be used with the numerical value you want to associate with a sample. For example, if it is one of a series of samples drawn over time, time could be used as a sample descriptor.

## !SampleNames

!SampleNames

Returns, as a text string, the sample name assigned in the Template Editor.

## !SampleNumbers

!SampleNumbers

Returns the number of the sample.

## !SummaryFormulas

!SummaryFormulas

Returns a list of the summary formulas in the specified section.

The formulas are returned in order that they were created. All summary formulas are returned.



**Note:** The value of a formula using this accessor might not update automatically when changes are made to the summaries or referenced formulas. To make sure that the formula remains correct, go to the **Operations** tab in the ribbon and click **Recalculate Now**.

---

## !SummaryNames

!SummaryNames

Returns a list of the summary names in the specified section.

The names are returned in order that they were created. All summary names are returned.



**Note:** The value of a formula using this accessor might not update automatically when changes are made to the summaries or referenced formulas. To make sure that the formula remains correct, go to the **Operations** tab in the ribbon and click **Recalculate Now**.

---



## !Units

!Units

Returns, as a text string, the units assigned to the sample descriptor in the New Group or Edit Group dialog in the Template Editor.

## !WellIDs

!WellIDs

Returns the well identifier (A1 to H12) for each sample replicate.

## !WellNumbers

!WellNumbers

Returns the well number of each sample replicate in a group to the Group section as a list of numbers. For example, in a 96-well microplate, A1 = 1 and H12 = 96.

## Blank Accessors

Blank accessors let you access and manipulate plate blank and group blank information. The average of the plate blank is displayed in the Plate section for Endpoint reads. The average of the group blank is displayed below the Group section table.

There is no accessor to access the individual well values of the plate blank as there is for the group blank. To access these values and get the standard deviation, minimum, maximum, and so on, use the specific well accessor (for example, !A1LmX) in conjunction with a concatenation operator (& or ~) and the applicable statistical function (Stdev, Min, Max, and so on).

SoftMax Pro Software supports the following blank accessors. For more blank accessor descriptions, see [Plate Data Accessors on page 91](#).

### !GroupBlank

!GroupBlank@GroupName

Returns, as a number, the average of all wells that have been designated as group blanks for the specified group.

### !GroupBlankValues

!GroupBlankValues@GroupName

Returns, as a list of numbers, the values of the individual wells that have been designated as group blanks for the specified group.

## **!PlateBlank**

**!PlateBlank**

Returns the average plate blank value as a number.

### **Blank Accessors Examples**

The following examples show how you can use blank accessors.

#### **Example: Accessing Group Blank Information**

The following formulas can be used in Summaries to report blank group information.

Average group blank:

`!GroupBlank`

Standard deviation of the group blank:

`Stdev(!GroupBlankValues)`

Maximum of group blank values:

`Max(!GroupBlankValues)`

Minimum of group blank values:

`Min(!GroupBlankValues)`

#### **Example: Subtracting a Single Plate Blank Value from Multiple Plate Sections**

If you want to subtract the same plate blank from multiple groups on different plates, you can do the subtraction in the Plate sections.

In this example, the template from first Plate section includes the plate blank. In the second Plate section that contains samples that continue the group from the first Plate section, the reduction formula is set to be the optical density minus the plate blank from Plate #1. The custom formula for this reduction is:

`!Lm1-!PlateBlank@Plate#1`

### Example: Subtracting a Plate Blank in a Group Section

To subtract a plate blank from a single group that has samples on multiple plates, you must do the subtraction in a Group section:

1. Set up the first Plate section and define a plate blank on it.
2. In the Reduction dialog, clear the **Use Plate Blank** check box so that the plate blank is not subtracted automatically from the group.

It is very important to do this or the blank will be subtracted twice from the samples on this plate, once in the Plate section and again in the Group section.

3. Define the samples belonging to the group in the subsequent Plate sections.
4. Create two custom columns called "Values" and "BlankSbtVal" using the following formulas respectively:
  - !Values
  - Values-!BlankSbtVal@Plate#1

The column named Values reports the reduced numbers from the Plate sections, while the column named BlankSbtVal subtracts the average OD of the plate blank from each of the optical densities in the Values column.

### Imaging Data Accessors

These accessors are used for Imaging read mode only.

Imaging data accessors let you access area, count, and intensity data from acquired images and data sets. The available accessors are dynamically added to the accessor list based on the acquisition and analysis settings in the **Settings** dialog.

The accessor names for imaging data are enclosed in square brackets to prevent conflicts with other formula elements. Within the brackets, most of these accessors can have a prefix that includes the related wavelength, classification, or both. The prefixes "TypeA" and "TypeB" are the default names for classifications in the software and are used in these examples. Custom classification prefixes are included in the list of accessors in the software based on custom classification names defined in the **Settings** dialog.

The names of the imaging accessors match the names of the selected measurements with the spaces removed. For example, if you define acquisition and analysis settings to count objects using transmitted light and define **Type A** and **Type B** classifications, the following accessors are available:

```
![ObjectCount]
![TypeAObjectCount]
![TypeBObjectCount]
```

If you define acquisition and analysis settings to count objects using a fluorescence emission of 541 nm and define **Type A** and **Type B** classifications, the following accessors are available:

```
![541ObjectCount]
![541TypeAObjectCount]
![541TypeBObjectCount]
```

When using a !Well prefix or well specification prefix, such as !A1, with imaging data accessors, make sure that this prefix comes before the square brackets. For example:

```
!Well[ObjectCount]
!A1[CoveredArea[%]]
```

See [Using !Well as an Accessor Prefix on page 77](#).

If you open a file that contains imaging data from a version of the SoftMax Pro Software before version 6.4, the legacy accessors are not enclosed in square brackets. For example, the following imaging accessors might be available in the list:

```
!AverageArea
!AverageIntegratedIntensity
!AverageIntensity
!CellCount
!CoveredArea
!ExpressionInImage
```

Legacy accessors can continue to be used for legacy data.



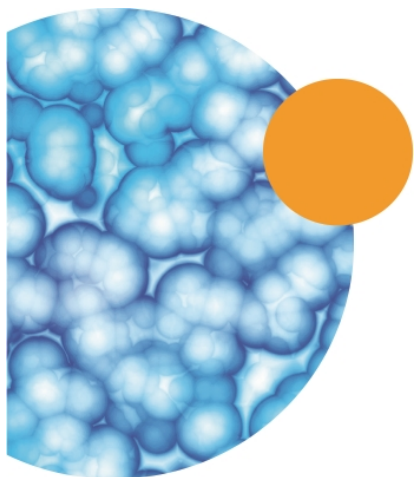
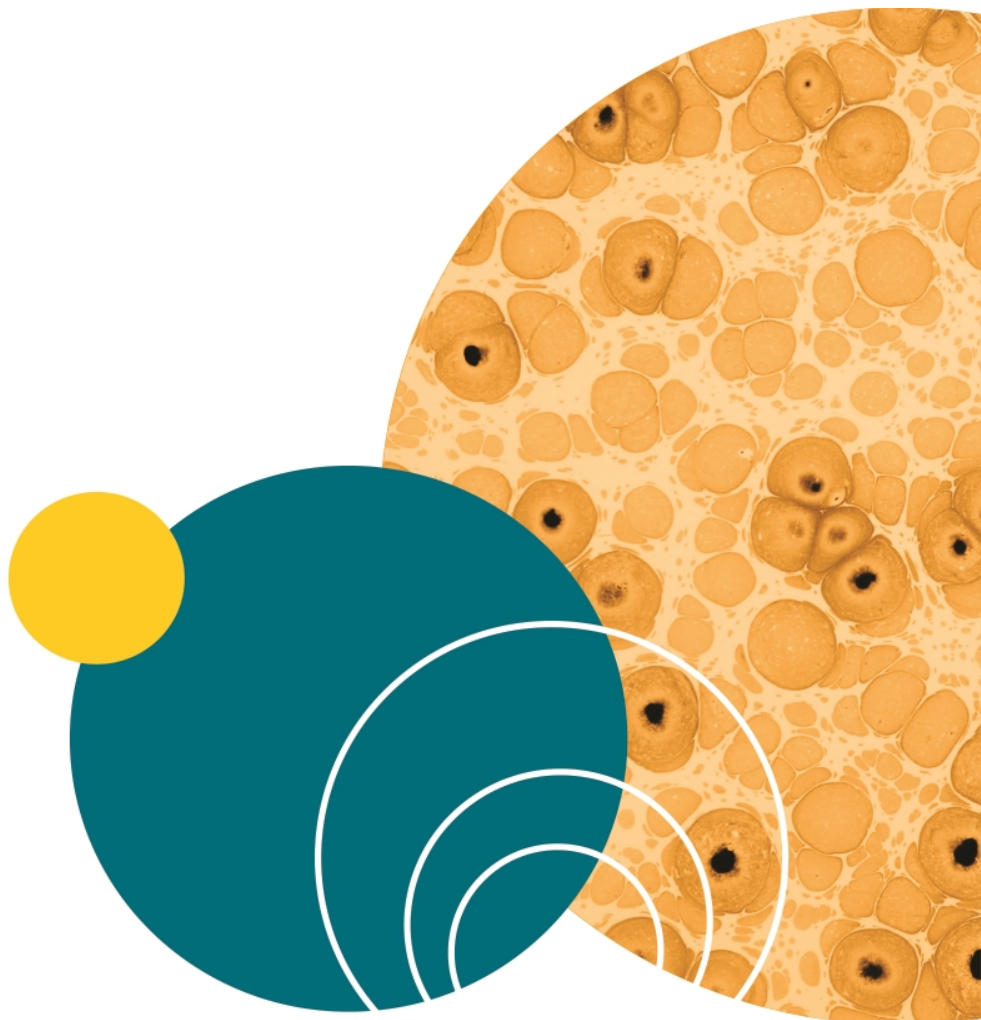
**Note:** The list of available imaging accessors grows as different measurement settings are included in **Plate** sections. These accessors remain in the list and are available for use with any SoftMax Pro Software file. Accessors for measurements that are not included in a **Plate** section are not relevant to the data in that section. Make sure that you select accessors that are relevant to your data.

---

## Measurements for Imaging Data

The measurements that follow are available in the SoftMax Pro Software **Settings** dialog. Separate measurements are available for each measured wavelength and classification, if applicable.

- **Object Count** gives the total number of objects detected in the image. This measurement is not used for field analysis of confluent areas.
- **Field Count** gives the total number of confluent areas detected in the image. This measurement is not used for discrete object analysis.
- **Object Percentage** gives the percentage the objects detected in the image by classification. This measurement is not used for field analysis of confluent areas.
- **Covered Area** gives the combined area of all the objects or confluent areas detected in the image as a percentage of the entire image area.
- **Object Area** gives the average area of the objects detected in the image expressed in  $\mu\text{m}^2$ . This measurement is not used for field analysis of confluent areas.
- **Field Area** gives the average area of the confluent areas detected in the image expressed in  $\mu\text{m}^2$ . This measurement is not used for discrete object analysis.
- **Object Roundness** gives the average roundness of each object detected in the image. A shape factor of 1.00 is perfectly round, while a shape factor of 0.00 is not round at all. This measurement is not used for field analysis of confluent areas.
- **Field Roundness** gives the average roundness of each confluent area detected in the image. A shape factor of 1.00 is perfectly round, while a shape factor of 0.00 is not round at all. This measurement is not used for discrete object analysis.
- **Object Average Intensity** gives the average fluorescent signal intensity of the objects detected in the image. This measurement is not used for field analysis of confluent areas. This measurement is not used for transmitted light (TL).
- **Field Average Intensity** gives the average fluorescent signal intensity of the confluent areas detected in the image. This measurement is not used for discrete object analysis. This measurement is not used for transmitted light (TL).
- **Object Intensity** gives the average total fluorescent signal intensity of the objects detected in the image. This measurement is not used for field analysis of confluent areas. This measurement is not used for transmitted light (TL).
- **Field Intensity** gives the average total fluorescent signal intensity of the confluent areas detected in the image. This measurement is not used for discrete object analysis. This measurement is not used for transmitted light (TL).
- **Total Intensity** gives the combined total fluorescent signal intensity of the objects or confluent areas detected in the image expressed in million intensity counts. This measurement is not used for transmitted light (TL).



# Index

## A

- abs 30
- absolute value 30
- accessor
  - !a1reduced 95
  - !alltimerorigins 87
  - !allvalues 91
  - !autocutoffon 78
  - !automixduration 84
  - !automixon 84
  - !avgreadtemperature 78
  - !columnformulas 103
  - !columnnames 103
  - !combinedplot 91
  - !concentration 103
  - !delaytime1 98
  - !delaytime2 98
  - !emcutoffs 78
  - !emfixedon 79
  - !emfixedwavelength 78
  - !emwavelengths 79
  - !endlimit 84
  - !endsweep 85
  - !exfixedwavelength 79
  - !exwavelengths 79
  - !factor 103
  - !firstcolumn 80
  - !gfactor 80
  - !groupblank 105
  - !groupblankvalues 105
  - !injector1volume 98
  - !injector2volume 99
  - !integrationend 80
  - !inttime1sequence 99
  - !inttime2sequence 99
  - !lastcolumn 80
  - !lmx 91
  - !lmxpraw 92
  - !lmxsraw 92
  - !lmxxvals 93
  - !maxlimit 85
  - !minlimit 85
  - !mvalue 93
  - !numbaselinereads1 100
  - !numbaselinereads2 100
  - !numpoints 86
  - !numpointsread 86
  - !numwavelengthsread 86
  - !numwells 80
  - !onsetvalue 86
  - !pathbackground 101
  - !pathcheckapplied 101
  - !pathchecklm1000 101
  - !pathchecklm900 101
  - !pathcheckon 101
  - !pathcorrectionon 101
  - !pathlength 102
  - !plateblank 106
  - !plateblanklmx 93
  - !plateblanklmp 94
  - !plateblanklms 94
  - !plateblankp 93
  - !plateblanks 94
  - !platekind 94

!platename 81  
 !pmtsetting 81  
 !premixduration 81  
 !premixon 81  
 !pvalue 94  
 !pvaluelm1 95  
 !pvaluelm2 95  
 !readinterval 87  
 !readtype 82  
 !runtime 87  
 !sampledescriptor 104  
 !samplenames 104  
 !samplenumbers 104  
 !sectionname 82  
 !startlimit 84  
 !startswEEP 85  
 !stddevplateblank 95  
 !stddevplateblanklmx 95  
 !stepsweep 85  
 !summaryformulas 104  
 !summarynames 104  
 !temperaturerun 87  
 !temperaturesetpoint 82  
 !timeofread 82  
 !timeresolvedon 82  
 !timerorigin 87  
 !timerun 88  
 !transferrate 82  
 !transfertime 83  
 !transfervolume 83  
 !units 105  
 !vmaxpoints 88  
 !wavelengthrun 88  
 !wavelengths 83  
 !wellidlmxpraw 92  
 !wellidlmxsraw 92  
 !wellids 105  
 !wellnumbers 105  
 !xvals 88  
 acos 30  
 addition 16  
 and 20  
 anovafstatandprob 36  
 antilog 31  
 antilog10 31  
 arccosine 30  
 arcsine 31  
 arctangent 31  
 areaunder 41  
 areaunderfit 41  
 ascii 72  
 asin 31  
 at (@) symbol 10  
 atan 31  
 atan2 31  
 average 36  
 avgdev 36  
  
**B**  
 baseline subtraction 100  
  
**C**  
 ceil 32  
 chiprobability 42  
 chiprobabilitypla 42  
 chiprobex 37  
 chisquared 42



- chisquaredpla 42
- concat 72
- concatenate
  - numbers in a list 26
  - numbers in an array 26
  - text strings 26, 72
- confidencelevel 42
- cos 32
- cosh 32
- cosine 32
- count 63
- customer support 13
- cv 37
- cvp 37
- D**
- degreesoffreedom 43
- delta 63
- derivative 37
- df 43
- division 15
- does not equal 19
- E**
- equals 18
- esdpercentagepoint 37
- exact 72
- exponent 16, 32
- F**
- fact 32
- factorial 32
- false 20
- fdist 37
- finv 38
- firstarray 63
- firstitem 63
- firstzero 38
- floor 33
- fprobpla 43
- fract 33
- fractional part 33
- fstatpla 43
- function
  - abs 30
  - acos 30
  - antilog 31
  - antilog10 31
  - areaunder 41
  - areaunderfit 41
  - ascii 72
  - asin 31
  - atan 31
  - atan2 31
  - average 36
  - avgdev 36
  - ceil 32
  - chiprobability 42
  - chiprobabilitypla 42
  - chiprobex 37
  - chisquared 42
  - chisquaredpla 42
  - concat 72
  - confidencelevel 42
  - cos 32
  - cosh 32

count 63  
 cv 37  
 cvp 37  
 degreesoffreedom 43  
 delta 63  
 derivative 37  
 df 43  
 esdpercentagepoint 37  
 exact 72  
 exponent 32  
 fact 32  
 fdist 36-37  
 finv 38  
 firstarray 63  
 firstitem 63  
 firstzero 38  
 floor 33  
 fprobpla 43  
 fract 33  
 fstatpla 43  
 index 63  
 indexlistfor 64  
 indexofmax 64  
 indexofmin 64  
 indexofnearest 64  
 int 33  
 intercept 43  
 interpolatedxdata 62  
 interpolatedydata 62  
 interpolatedydataatxpoints 62  
 interpx 44  
 interpy 44  
 isempty 71  
 iserr 71  
 item 64  
 itemcount 65  
 itemindex 65  
 left 73  
 len 73  
 ln 33  
 localtimenumeric 61  
 localtimetext 61  
 log 34  
 log10 34  
 lower 73  
 makeerr 69  
 max 38  
 maxdev 38  
 median 38  
 mid 73  
 min 38  
 nearestto 65  
 nonum 71  
 normalorderstatisticmedians 44  
 now 73  
 ntharray 65  
 nthitem 65  
 nullbetween 66  
 nulloutside 66  
 numdata 44  
 numvarparams 44  
 parma 45  
 parmacilower 46  
 parmaciupper 47  
 parmacivariance 48  
 parmb 45  
 parmbcilower 46  
 parmbciupper 47  
 parmbcivariance 48  
 parmc 45

parmccilower 46  
 parmcciuupper 47  
 parmccvariance 48  
 parmdcilower 46  
 parmdciupper 48  
 parmdcvariance 48  
 parmng 45  
 parmngcilower 47  
 parmngciupper 48  
 parmngcvariance 48  
 PeakProAmplitudeIndex 59  
 PeakProAnalysis 57  
 PeakProCountIndex 59  
 PeakProDecayTimeIndex 60  
 PeakProDecayTimeStdDevnIndex 60  
 PeakProFrequencyIndex 59  
 PeakProRiseTimeStdDevnIndex 60  
 PeakProWidthIndex 59  
 PeakProWidthStdDevnIndex 59-60  
 pi 34  
 rand 34  
 randnorm 34  
 relativepotency 49  
 relpotcilowerpla 49  
 relpotciupperpla 49  
 relpotpla 49  
 relpotvariancepla 49  
 right 73  
 round 34  
 rsquared 50  
 rsquaredpla 50  
 secondderivative 38  
 sideways 66  
 sign 35  
 sin 35  
 sinh 35  
 slope 50  
 smooth 38  
 sort 50  
 sqrt 35  
 stderr 39  
 stdev 39  
 stdevp 39  
 sum 39  
 tan 35  
 tanh 36  
 tdist 39  
 text 74  
 timetovmax 53  
 timetovmaxex 53  
 tinv 39  
 today 74  
 upper 74  
 utctimenumeric 61  
 val 74  
 vmax 53  
 vmaxcorr 53  
 vmaxcorrex 54  
 vmaxex 53  
 vmaxpersec 54  
 vmaxpersecex 54  
 vmaxptsused 54  
 vmaxptsusedex 54  
 whaterr 71

**G**

greater than 18  
 greater than or equals 18

## H

hyperbolic cosine 32  
hyperbolic sine 35  
hyperbolic tangent 36

## I

if 20  
imaging accessors 107  
index 63  
indexlistfor 64  
indexofmax 64  
indexofmin 64  
indexofnearest 64  
int 33  
integer part 33  
intercept 43  
interpolatedxdata 62  
interpolatedydata 62  
interpolatedydataatxpoints 62  
interpx 44  
interpy 44  
isempty 71  
iserr 71  
item 64  
itemcount 65  
itemindex 65

## L

left 73  
len 73  
less than 18

less than or equals 19

ln 33

localtimenumeric 61

localtimetext 61

log 34

log10 34

logarithm 33-34

lower 73

## M

makeerr 69

max 38

maxdev 38

median 38

mid 73

min 38

mod 16

multiplication 15

## N

NANs 69

nearestto 65

nonum 71

normalorderstatisticmedians 44

not 20

now 73

ntharray 65

nthitem 65

nullbetween 66

nulloutside 66

numdata 44

numvarparams 44

## O

## operator

- 16
- & 26
- () 16
- / 15
- ^ 16
- ~ 26
- + 16, 26
- < 18
- <= 19
- <> 19
- = 18
- > 18
- >= 18
- addition 16
- and 20
- concatenate numbers in a list 26
- concatenate numbers in an array 26
- concatenate text strings 26
- division 15
- does not equal 19
- equals 18
- exponent 16
- false 20
- greater than 18
- greater than or equals 18
- if 20
- less than 18
- less than or equals 19
- mod 16
- multiplication 15
- not 20

- or 20
- parentheses 16
- remainder 16
- subtraction 16
- true 20

or 20

## P

- parentheses 16
- parma 45
- parmacilower 46
- parmaciupper 47
- parmacivariance 48
- parmb 45
- parmbcilower 46
- parmbciupper 47
- parmbcivariance 48
- parmc 45
- parmccilower 46
- parmcciupper 47
- parmccivariance 48
- parmd 45
- parmdcilower 46
- parmdciupper 48
- parmdcivariance 48
- parmg 45
- parmgcilower 47
- parmgciupper 48
- parmgcivariance 48
- path (@) 10
- PeakProAmplitudeIndex 59
- PeakProAnalysis 57
- PeakProCountIndex 59

PeakProDecayTimeIndex 60  
PeakProDecayTimeStdDevnIndex 60  
PeakProFrequencyIndex 59  
PeakProRiseTimeIndex 60  
PeakProRiseTimeStdDevnIndex 60  
PeakProWidthIndex 59  
PeakProWidthStdDevnIndex 59  
pi 34

## R

rand 34  
randnorm 34  
random number 34  
relativepotency 49  
relpotcilowerpla 49  
relpotciupperpla 49  
relpotpla 49  
relpotvariancepla 49  
remainder 16  
right 73  
round 34  
round down 33  
round up 32  
rsquared 50  
rsquaredpla 50

## S

scope (@) 10  
secondderivative 38  
sideways 66  
sign 35  
sin 35

sine 35  
sinh 35  
slope 50  
smooth 38  
sort 50  
sqrt 35  
square root 35  
stderr 39  
stdev 39  
stdevp 39  
subtraction 16  
sum 39

## T

t-distribution 39  
t-test 39  
tan 35  
tangent 35  
tanh 36  
tdist 39  
technical support 13  
text 74  
timetovmax 53  
timetovmaxex 53  
tinv 39  
today 74  
true 20

## U

upper 74  
utctimenumeric 61

**V**

val 74

vmax 53

vmaxcorr 53

vmaxcorrex 54

vmaxex 53

vmaxpersec 54

vmaxpersecex 54

vmaxptsused 54

vmaxptsusedex 54

**W**

whaterr 71

## Contact Us

Phone: [+1-800-635-5577](tel:+1-800-635-5577)  
Web: [moleculardevices.com](http://moleculardevices.com)  
Email: [info@moldev.com](mailto:info@moldev.com)

Visit our website for a current listing of worldwide distributors.