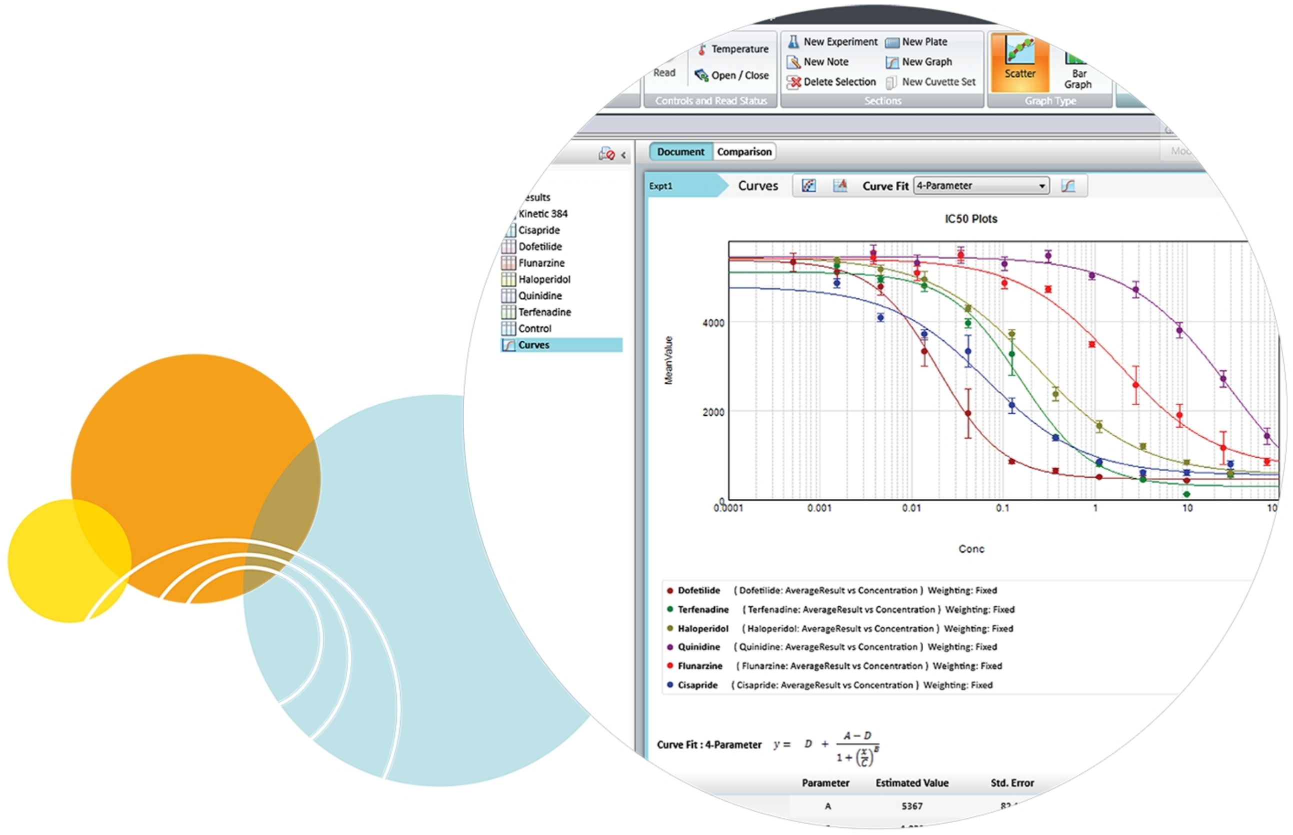


Temperature
Read
Open / Close
Controls and Read Status
New Experiment
New Plate
New Note
New Graph
Delete Selection
New Cuvette Set
Sections
Scatter
Bar Graph
Graph Type
Document
Comparison
Kinetic 384
Cisapride
Dofetilide
Flunarzine
Haloperidol
Quinidine
Terfenadine
Control
Curves
Expt1
Curves
Curve Fit
4-Parameter
IC50 Plots
MeanValue
Conc
Dofetilide (Dofetilide: AverageResult vs Concentration) Weighting: Fixed
Terfenadine (Terfenadine: AverageResult vs Concentration) Weighting: Fixed
Haloperidol (Haloperidol: AverageResult vs Concentration) Weighting: Fixed
Quinidine (Quinidine: AverageResult vs Concentration) Weighting: Fixed
Flunarzine (Flunarzine: AverageResult vs Concentration) Weighting: Fixed
Cisapride (Cisapride: AverageResult vs Concentration) Weighting: Fixed
Curve Fit : 4-Parameter
Parameter
Estimated Value
Std. Error
A
5367

SoftMax® Pro

Data Acquisition and Analysis Software

Version 7.1.1

Formula Reference Guide

5049093 D

December 2019

Contents

Chapter 1: Introduction

1

The *SoftMax® Pro Data Acquisition and Analysis Software Formula Reference Guide* identifies and enumerates the details about formulas, accessors, and functions that you can use to create powerful data analysis templates for Protocol files in order to completely automate the analysis and results reporting for your plate reads.

There are three sections in this guide that describe the parts of a formula:

- **Operators** - Describe the different types of operators. See Operators on page 13.
- **Functions** - Describe the different types of built-in functions you can use to assemble formulas. See Functions on page 27.
- **Accessors** -Describe the special functions that provide access to other specific information. See Accessors on page 85.

Before you use this *SoftMax Pro Data Acquisition and Analysis Software Formula Reference Guide*, make sure that you understand the basics of the SoftMax Pro Software. For more information about how to use the software, see the *SoftMax Pro Data Acquisition and Analysis Software User Guide* and the SoftMax Pro Software application help.

Formula Categories

There are three formula categories:

- **Reduction Formulas** - Apply a formula to entire Plate sections and Cuvette Set sections. See Reduction Formulas on page 6.
- **Column Formulas** - Apply a formula to a single column in Group sections. See Column Formulas on page 7.
- **Summary Formulas** - Reduce data to a single value, list, or array in Group sections and Note sections. See Summary Formulas on page 8.

In a Graph section, you can assign formulas to the X and Y data values.

Reduction Formulas

You can use Reduction formulas in Plate sections and Cuvette Set sections. Reduction formulas apply to the Plate section or Cuvette Set section that you select in the Navigation Tree and displays in the Workspace. These formulas determine the values that the software reports in the Values column of the Group sections. Unlike formulas in Group sections and Summary formulas, Reduction formulas must evaluate to a number or list of numbers, but 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 use.

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

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

Entering Custom Reduction Formulas

The software includes many reduction formulas. If default reductions do not meet your needs, you can create custom formulas.

1. In the Ribbon, on the Home tab, in a Plate section toolbar, or in the Workspace Reduction Settings area, click the **Reduction** Σ icon to display the Data Reduction dialog.
2. Select **Custom** from one of the drop-down lists. The Data Reduction dialog displays an additional field and an icon to enable you to define a custom formula. The icon depends on the read mode and instrument type.
3. Click =f(x) or to display the Calculation dialog or the Formula Editor dialog where you create or edit a formula.
4. Enter the formula in the field and click **OK** to return to the Data Reduction dialog.
5. Click **OK** to set the formula parameters.

For example, in a Plate section or Cuvette Set section, 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

You can use Column formulas in Group sections. Column formulas act on either raw or reduced numbers from Plate sections or Cuvette Set sections and evaluate to a list of numbers or text. Each column has a Header (name) and a formula. Other Column or Summary formulas can reference Column formulas by name.

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

The Reduction formula in the Plate section or Cuvette Set section dictates the numbers in the Values column of the Group section. The formula for the Values column is **!Wellvalues**.

Note: Values is the default name which you can change.

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

Column formulas evaluate an entire column on a row-by-row basis. When you write column formulas to access data from two or more Group sections, the Group sections must contain the same number of rows. 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 to sort 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 PlateX is !Wavelengthrun@plateX and !B1Lm1@PlateX.

Summary Formulas

You can use Summary formulas in Group sections and Note sections to evaluate to either a single number or a list of numbers. Summary formulas report information from Plate sections, Cuvette Set sections, Graph sections, Group sections, or to a combination of these sections.

The Summary formula consist of five parts:

- Summary Name (used to refer to the Summary in other formulas)
- Description (optional)
- Formula
- Data Display (where you specify the numeric notation)
- Comments (optional)

To edit a summary, double-click the summary to display the Formula Editor.

Formula Size

While there are no practical restrictions on formula length, you should 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 Column formulas and Summary formulas, each containing a portion of the calculation, rather than one Column formula 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 Summary formulas to contain intermediate steps. If a formula becomes lengthy, you can break it into several smaller formulas and shorten the object names.

Formula Building Blocks

You can use some formula building blocks alone and some formula building blocks require other building blocks. A complete list of the building blocks, along with examples that show how to use them, is included in this document.

There are four types of formula building blocks:

- **Operators** - Enable you to combine other building blocks together. See Operators on page 13.
- **Functions** - Mathematical or text items that do operations within the formula. For example, standard deviation or slope. See Functions on page 27.
- **NANs** (Not A Number) - Text items that the software considers to be numbers. See NANs and MakeErrs on page 71.
- **Accessors** - Enable you to access data and other information from objects (plates, cuvette sets, and so on) and functions. For example, Vmax rate. Accessors are unique to the SoftMax Pro Software. See Accessors on page 85.

Lists of Numbers and Arrays of Numbers

The 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 × 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 that contains the optical densities at each time point in a Kinetic run: it is a Y × X array, where Y is the number of ODs in each row and X is the number of samples in the group assay, and is also the number of rows in the Group section.

Formula Name Conventions

SoftMax Pro Software uses a hierarchical naming structure, much like the directories and subdirectories that computer file systems use. All objects (for example, columns, Summaries, wells, graphs, plots) in the software have names and you use the software object name to refer to an object when you write formulas.

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 software uses the @ symbol to combine these references:

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

Columns in two different Group sections can have the same name, but the 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 software uses the first object with that name that it finds in the file.

For example, suppose Group sections 1 and 2 contain a column named Values, Group section 3 does not contain a column with that name, and you write 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
```

The software finds the first occurrence of an object named "Values" in the file and fills in:

```
ColX – Values@GroupSectionName1
```

The software subtracts the Values column from Group section 1 from ColX.

To write the formula with the proper result, you must specify the section that contains the correct Values column:

```
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 when they combine several functions or accessors together. These more complicated formulas are called nested formulas because you use parentheses to nest different functions within each other.

The order in which you nest the functions determines the order in which the software processes the formula. The 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.

Formula Rules

Case

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: # $ % / * ?, you must enclose the name in single quotation marks. For example, ‘OD Values’.

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

Reference Columns

When you write 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 multiplies values from each Group section row by row, the output is undefined for rows 11 and 12.

Evaluate Column Formulas

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

You can use NANs to put text into a column of numbers, because the software treats NANs as numbers. You can also report numbers as text strings.

Strings in Formulas

When a formula includes a text result that you want to display, you must enclose the text results in double quotation marks, for example: “Out of Range”.

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

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 level of technical service.

Our Support website, www.moleculardevices.com/service-support, has a link to the Knowledge Base, which contains technical notes, software upgrades, safety data sheets, and other resources. If you still need assistance after consulting the Knowledge Base, you can submit a request to Molecular Devices Technical Support.

You can contact your local representative or Molecular Devices Technical Support 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.

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

Chapter 2: Operators

2

Operators enable you to combine and compare numbers, functions, accessors, and text strings in a variety of ways within formulas.

The software uses the following types of operators:

- Mathematical Operators (see below)
- Comparison (Logical) Operators on page 16
- Conditional (Boolean) Operators on page 18
- Concatenation Operators on page 24

Mathematical Operators

The software uses 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.

The software supports the following mathematical operators.

* for Multiplication

Enter ***** 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

Enter **/** 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

Enter **+** 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.

```
!WellLm1 + !WellLm2
```

- for Subtraction

Enter **-** 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

Enter **^** for exponent.

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

```
!WellLm1 ^ 2
```

Mod for Remainder

Enter **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 mod 3
```

( ) for Parentheses

Enter **(** 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 to 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 values, for example, optical densities by a constant in a Group table.

Example: Multiplying Lists of Numbers

When you multiply lists of numbers, you can refer to them by name, or you can use functions (see Functions on page 27) or accessors (see Accessors on page 85) 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)". Suppose also you have a column named Dilution that contains the formula !Factor or !SampleDescriptor.

The formula for AdjResults can be written in any 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 concentration is:

```
!WellLm1*50
```

Comparison (Logical) Operators

Comparison operators enable you 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 18. When used by themselves, comparison operators return True if the argument is true, and False if the argument is false.

The software supports the following Comparison operators.

= for Equals

Enter **=** for equals.

Equals determines if two numbers, lists of numbers, arrays of numbers, or text strings are equal to each other. For text strings, the Operator is not case-sensitive.

!Lm1 = !Lm1 is True

!Lm1 = !A12Lm1 is False

> for Greater Than

Enter **>** 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

Enter **>=** 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

Enter **<** 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

Enter <= 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

Enter **<>** 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 to 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 enable you to compare results in Plate sections, Cuvette Set sections, Group sections, and Summary formulas. Use conditional operators to set criteria to evaluate numerical results. Group sections and Summary formulas can evaluate either to a number, a list of numbers, 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.

The software supports the following Conditional operators.

If

Enter **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@PlateX, “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 PlateX, return “Above threshold”, otherwise return “Below threshold”.

And

Enter **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@PlateX And !WellLm2 > !A12Lm2@PlateX

Or

Enter **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@PlateX Or !WellLm2 > !A12Lm2@PlateX

Not

Enter **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@PlateX And !WellLm2 > !A12Lm2@PlateX)

False Returns the logical value False.

True Returns the logical value True.

False

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

True

Enter **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 enables you to combine several simple mathematical operations into a single, more complex operation or function. The software also enables you to nest (embed) conditional statements within other conditional statements, to permit 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)

Simple 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 !WellLm2 > 0.4	”Pass”	”Fail”
If (MeanValue > Summary1@Control and MeanValue < Summary2@Control, “Acceptable”, “Out of Specification”)	MeanValue > Summary1@Control and MeanValue < Summary2@Control	“Acceptable”	“Out of Specification”

Nested Conditionals

You can write conditional statements 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 (Condition1,Outcome if Condition1 is True, (if(Condition2, Outcome if Condition2 is True, Outcome if Condition2 is False))

In this formula, if Condition1 is true, an outcome is returned, but if Condition1 is false, Condition2 is evaluated. If Condition2 is true, a second outcome is returned; if it is false, a third outcome is returned.

You can create 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(Condition1, outcome when Condition1 is True, (if(Condition2, outcome when Condition2 is True,(if(Condition3, outcome if Condition3 is True...(if ConditionX, outcome if ConditionX is True, outcome if ConditionX 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 shows how to 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 Options 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 27.

Formula

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

Breakdown

Condition1: !WellLm1<!A12Lm1@PlateX

Value-if-Condition1 True: MakeErr(118) (= “Low”)

Condition2: !WellLm1>!H1Lm1@PlateX

Value-if-Condition2 True: MakeErr(117) (= “High”)

Value-if-Condition2 False: MakeErr(123) (= “*****”)

Tips for Writing Conditional Formulas

- When you write 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 open and close parentheses.
- In general, one more outcome than the number of conditions is 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 NANs to include text in the results.

  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, you should write conditional formulas 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 is collected.

Concatenation Operators

Use operators to concatenate text strings, lists of numbers, and arrays of numbers. These operators enable you to use relatively simple formulas to do complex data analyses.

+ (Plus) to Concatenate Text Strings

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

& (Ampersand) to Concatenate Numbers to a List

Enter **&** (ampersand) to combines several numbers or lists of numbers into a single list of numbers.

X&Y&Z

~ (Tilde) to Concatenate Numbers to an Array

Enter **~** (tilde) to combines several numbers or lists of numbers into an array of numbers.

X~Y~Z

Concatenating Numbers

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

Original Lists Before Concatenation

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

Concatenation Using &

A&B
1
2
3
4
5
6
7
8

Concatenation Using ~

A~B	
1	5
2	6
3	7
4	8

Concatenation Operator Examples

The following examples show how to use Concatenation operators.

Example: Using & (Ampersand) in a Column Formula

Suppose you have plate section with kinetic multiwavelength data and an associated group table.

Suppose also that you want to find the maximum value of wavelengths 2 and 4 in each row of the group table.

To do so, create a column with the following formula:

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

This example can be found in the WP (Wavelength Precision) group table of SpectraTest ABS1 validation protocols.

Example: Using ~ (Tilde) in a Group

Suppose you 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. You can define a new column MaxOD to report the maximum OD from each row of the table using the ~ or the &:

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

Example: Using ~ (Tilde) 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@Plate1 ~ !B12@Plate1 ~ !C12@Plate1 ~ !D12@Plate1 ~ !E12@Plate1 ~
!F12@Plate1 ~ !G12@Plate1 ~ !H12@Plate1)
```

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 you bring Spectrum scans or Kinetic run information into a Group section and do complex analyses such as averaging wells on a point-by-point basis, graph the wells (or groups of wells) in the Graph section and apply different curve fits, or subtract two Spectrum scans on a point-by-point basis and then plot 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.

Chapter 3: Functions

3

This section describes the different types of functions you can use to assemble formulas. In contrast to Operators, Functions have names and take 0 to 4 parameters.

The general format for the many of the software built-in functions is:

FunctionName(Parameter1, Parameter2, Parameter3)

You can input zero to four parameters into functions. Input parameters can include lists of numbers, arrays of numbers, and lists of text strings 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 you enter the parameters in the incorrect order, the function returns erroneous results.

Software functions are organized in the following categories:

- Mathematical Functions on page 28
- Statistical Functions on page 35
- Graph Functions on page 41
- Vmax Reduction Functions on page 53
- Peak Pro Analysis Functions on page 58
- Time Functions on page 62
- Interpolation Functions on page 63
- Array Functions on page 64
- NANs and MakeErrs on page 71
- Text Functions on page 76
- Curve Fit Functions on page 79
- Miscellaneous Functions on page 84

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 software are all common mathematical, algebraic, and trigonometric functions, and you can use them 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).

Use mathematical functions singly as shown in the examples, or 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.

You can use mathmatical functions 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 software is set to display results to a set number of decimal places, and if the number of decimal places you specify is larger than the number you specify for the individual function, the extra decimal places are filled with zeroes. For example, if the software is set to display three decimal places and you round a number to two decimal places, the number displays as X.YZ0. Similarly, integers display as X.000.

The 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)

Acosh

Acosh(Parameter)

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

Acosh(3) = 1.762747

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.

Antilog(1) = 2.718

Antilog(Summary1)

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.

Antilog10(1) = 10

Antilog(Results)

Asin

Asin(Parameter)

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

Asin(-0.25) = 0.25268

Asin(ColumnX)

Asinh

Asinh(Parameter)

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

Asinh(3) = 1.818446

Atan

Atan(Parameter)

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

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.

Atan2(1, 1) = 0.78540

Atan2(Values, Concentration)

Atanh

Atanh(Parameter)

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

Atanh(0.5) = 0.549306

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 31.

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(!WellLm1)

Erf

Erf(parameter)

Returns the error function erf(x)

Exp

Exp(Parameter)

Returns e raised to the power defined by the parameter of a number, list of numbers, or array of numbers.

Exp(3) = 20.08554

Exp(Summary3)

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(Col1)

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 30.

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)

Gamma

Gamma(Parameter)

Returns the gamma function for a number, list of numbers, or array of numbers.

Gamma(0.5) = 1.772454

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)

LambertW

LambertW(Parameter)

Returns the Lambert W function (principal branch) for a number, list of numbers, or array of numbers.

LambertW(-0.367879441) = -0.999969

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 below.

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 above.

Log(5) = 1.60944

Log(Values)

Log10

Log10(Parameter)

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

Log10(5) = 0.69897

Log10(!Lm1)

Pi

Pi

Returns the value of π. No parameters.

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.

RandNormWithSeed

RandNormWithSeed(Parameter)

Same as for operator RandNorm but specifying a number for the seed of the random number generator so that the result does not change when recalculated.

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.

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.

The software supports the following statistical functions.

ANOVAFStatAndProb

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

ANOVA F-Statistic and Probability

Performs 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:

ANOVAFStatAndProb((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(chiSqd, 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 (chiSqd).

For the parameters, **chiSqd** 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 (data, x spacing)

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.

ESDMarkOutliers

ESDMarkOutliers(data,maxOutliers,significance)

Returns the items in a list or array of numbers, with outliers, as identified by the Rosner extreme studentized deviate test, replaced by MakeErr(139) which is displayed as "Outlier". The maximum number of outliers and significance for the test must be specified.

ESDPercentagePoint

ESDPercentagePoint(*α*,*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:

- *α* = significance level (probability)
- *n* = number of data points
- *i* = number of outliers

ESDOutlierIndices

ESDOutlierIndices(data,max,Outliers,significance)

Returns the indices of any items in a list or array of numbers identified as outliers by the Rosner extreme studentized deviate test. The maximum number of outliers and significance for the test must be specified.

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. It is useful for locating peaks and valleys in a spectrum, where the derivative 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.

NormalCDF

NormalCDF(Parameter)

Returns the normal distribution cumulative distribution function. The Parameter is a number, list, or array of numbers.

NormalCDFInv

NormalCDFInv(Parameter)

Returns the inverse of the normal distribution cumulative distribution function.

ROUTMarkOutliers

ROUTMarkOutliers(data,functionSelector,Q)

Given an array list of data in the form x~y or x~y~w (x, y, w are equal-sized arrays of the x, y, and weight values respectively), returns an array of the y-values, with any values identified as outliers by the ROUT test (Motulsky and Brown BMC Bioinformatics 2006 7:123), replaced by MakeErr(139).

Outliers are identified based on a curve fit model specified by the function selector (see FunctionSelector on page 80).

Q is a number between 0 and 1 which tunes the selectivity of the algorithm, and which becomes more aggressive in identifying outliers as Q increases; the authors recommend the value 0.01.

If the calculation fails the result is an empty array.

See ROUTOutlierDataIndices and ROUTOutlierIndices.

ROUTOutlierDataIndices

ROUTOutlierDataIndices(PlotName,Q)

Returns a list of the indices of data points identified as outliers by the ROUT method with parameter Q.

See ROUTMarkOutliers and ROUTOutlierIndices.

ROUTOutlierIndices

ROUTOutlierIndices(data,functionSelector,Q)

Given data, function selector, and Q value, returns the indices of any items identified as outliers by the ROUT test. See ROUTMarkOutliers above.

See ROUTMarkOutliers and ROUTOutlierDataIndices.

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.

ShapiroWilk

ShapiroWilk(Parameter)

Returns the Shapiro Wilk statistic of a list or array of up to fifty numbers.

ShapiroWilkPercentagePoint

ShapiroWilkPercentagePoint(p,N)

Returns the Shapiro Wilk percentage point with percentage p for N samples. To be used for normality testing in conjunction with the ShapiroWilk function. Supported percentages are: 1,2,5,10,50,90,95,98,99.

ShapiroWilkRoystonProbability

ShapiroWilkRoystonProbability(Parameter)

Returns the p-value for the Shapiro Wilk normality test as extended by Royston. The parameter is a list or array of up to five thousand numbers.

Smooth

Smooth (Parameter)

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 to 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 Array Functions on page 64.

Graph Functions

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

The InterpX and InterpY functions are generally used in Group section column formulas.

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 Data Acquisition and Analysis Software User Guide* or the application help.

The software supports the following graph functions.

AICc

AICc(PlotName)

Returns the corrected Akaike criterion for the curve fit model. This can be used to compare models which need not be nested.

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. You can also use it 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.

    AreaUnderFit(Plotname@Graphname, 1, 1000)

AreaUnderPlot

AreaUnderPlot(PlotName,Xranges)

Similar to AreaUnderFit, but x-ranges are specified as an array of pairs of numbers, and a more accurate integration algorithm is used.

ChiProbability

ChiProbability(PlotName@GraphSection)

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

ChiProbabilityPLA

ChiProbabilityPLA (PlotName@GraphSection)

Chi-squared Probability for Parallel Line analysis

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, Plot1@Graph1, 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(Plotname@Graphsection)

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 is no longer supported. See DegreesOfFreedom above.

Note: If you have a protocol or data file that uses DF, versions of the software in which it was deprecated automatically update the formula to use DegreesOfFreedom.

FProbCompare

FProbCompare(PlotName1,PlotName2)

Returns the probability corresponding to FStatCompare.

FProbLackOfFit

FProbLackOfFit(PlotName)

Returns the probability corresponding to FStatLackOfFit.

FProbLackOfFitPLA

FProbLackOfFitPLA(PlotName)

Returns the probability corresponding to FStatLackOfFitPLA.

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.

FProbRegression

FProbRegression(PlotName)

Returns the probability corresponding to FStatRegression. It is an F-test where the null hypothesis is that slope of the data is zero.

FStatCompare

FStatCompare(PlotName1,PlotName2)

Returns the F-statistic for extra-sum-of squares ANOVA comparing two nested curve fit models.

PlotName1 correponds to the model with fewer variable parameters.

FStatLackOfFit

FStatLackOfFit(PlotName)

Returns the lack-of-fit F-statistic, which is applicable to data sets with replicate measurements.

FStatLackOfFitPLA

FStatLackOfFitPLA(PlotName)

Returns the lack-of-fit F-statistic for a PLA fit, which is applicable to data sets with replicate measurements.

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.

FStatRegression

FStatRegression(PlotName)

Returns the F-statistic comparing the model to a linear fit with slope zero.

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, Plot1@Graph1, or Std@Standardcurve.

    InterpX(Std@StandardCurve, YValues)

InterpX is used to calculate concentrations of unknowns from a standard curve.

InterpXInRange

InterpXInRange(PlotName,Yvalues,x1,x2)

Similar to InterpX but an allowable range for x is specified by parameters x1, x2.

InterpY

InterpY (PlotName@GraphName, Xvalues)

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

```
InterpY(Plot6@Graph3, XValues)
```

InterpYCILower

InterpYCILower(PlotName,Xvalues)

Returns the confidence interval lower limit of the function value (confidence band) at the specified X values.

InterpYCILowerInterpX

InterpYCILowerInterpX(PlotName,Yvalues)

Returns X values of lower confidence band at the specified Y values.

InterpYCILowerInterpXInRange

InterpYCILowerInterpXInRange(PlotName,Yvalues,x1,x2)

Returns X values, between x1 and x2, of lower confidence band at the specified Y values.

InterpYCIUpper

InterpYCIUpper(PoltName,Xvalues)

Returns the confidence interval upper limit of the function value (confidence band) at the specified X values.

InterpYCIUpperInterpX

InterpYCIUpperInterpX(PlotName,Yvalues)

Returns X values of upper confidence band at the specified Y values.

InterpYCIUpperInterpXInRange

InterpYCIUpperInterpXInRange(PlotName,Yvalues,x1,x2)

Returns X values, between x1 and x2, of upper confidence band at the specified Y values.

InterpYPILower

InterpYPILower(PlotName,Xvalues)

Returns the prediction interval lower limit of the function value (prediction band) at the specified X values.

InterpYPILowerInterpX

InterpYPILowerInterpX(PlotName,Yvalues)

Returns X values of lower prediction band at the specified Y values.

InterpYPILowerInterpXInRange

InterpYPILowerInterpXInRange(PlotName,Yvalues,x1,x2)

Returns X values, between x1 and x2, of lower prediction band at the specified Y values.

InterpYPIUpper

InterpYPIUpper(PlotName,Xvalues)

Returns the prediction interval upper limit of the function value (prediction band) at the specified X values.

InterpYPIUpperInterpX

InterpYPIUpperInterpX(PlotName,Yvalues)

Returns X values of upper prediction band at the specified Y values.

InterpYPIUpperInterpXInRange

InterpYPIUpperInterpXInRange(PlotName,Yvalues,x1,x2)

Returns X values, between x1 and x2, of upper prediction band at the specified Y values.

InterpYVariance

InterpYVariance(PlotName,Xvalues)

Returns the variance of the function value at the specified X values.

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.

Parm_

Note: Replace the _ with A, B, C, D, G, H, I, J for all functions that return a parameter property.

Parm_(PlotName@GraphName)

Returns the A, B, C, D, or 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.

Parm functions serve to extract curve fit Parameter values and their property values.

Parm_CILower

Replace _ with either A, B, C, D, or G.

ParmACILower (PlotName@GraphName)

Returns the lower limit of the confidence interval for the 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.

Parm_CIUpper

Replace _ with either A, B, C, D, or G.

ParmACIUpper (PlotName@GraphName)

Returns the upper limit of the confidence interval for the 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.

Parm_Covar

Replace _ with either A, B, C, D, G, H, I, or J.

Parm_Covar(PlotName)

Returns, as a list, the covariance of the parameter with respect to all other parameters used in the curve fit.

Parm_RatioFiellerCI

Replace _ with either A, B, C, D, G, H, I, or J.

Parm_RatioFiellerCI(PlotName1,PlotName2,alpha)

Returns the confidence interval of the ratio of the parameter for the specified plots (PlotName1 corresponds to the numerator in the ratio). The significance for the confidence interval is specified by alpha.

Parm_StdError

Replace _ with either A, B, C, D, or G.

ParmAStdError(PlotName@GraphName)

Returns the standard error of the curve fit for the parameter for the specified plot.

Parm_Variance

Replace _ with either A, B, C, D, or G.

ParmAVariance(PlotName@GraphName)

Returns the variance of the curve fit parameter 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 value for the StandardPlotName curve to the C value for the TestPlotName curve.

With a 5-parameter curve fit, the C values are no longer the mid-point between the max and min.

The formula for EC50/IC50 is:

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

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.

Residuals

Residuals(PlotName)

Returns a list of residuals of the plot. Useful in assessing the suitability of a curve fit model, either visually by plotting or with a normality test.

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(Parameter)

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

WeightingFormula

WeightingFormula(PlotName)

Returns (as text) the curve fit weighting formula.

Graph Functions Examples

The following examples show how to use graph functions.

Example: Using InterpX and InterpY in Group Sections

Suppose you have a Graph section named "Graph1", in which you have a plot named "Plot1", 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(Plot1@Graph1, MeanValue).
- To create a column of interpolated Y values, use the formula InterpY(Plot1@Graph1, 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:

```
Slope((!TimeRun)^2, !Lm1)
```

The reduced plot display for Kinetic plots in Plate sections or Cuvette Set sections always displays the results of Wavelength Options 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

To determine Vmax or the Time to Vmax, use the functions in the Data Reduction dialog. As an alternative, 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.

Time to Vmax is an alternative method to analyze non-linear Kinetic reactions. It 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 use for the calculation (for example, !CombinedPlot, !WellLm1, !Lm1).

Second Parameter

The number of Vmax Points to use in the calculation (a constant set by the user in the Data Reduction dialog). 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 are 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 85.

Third Parameter

The read interval to use in the calculation. A constant that represents 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 you specify in the Settings dialog is used. See Accessors on page 85.

The 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. Time to Vmax is the time to the midpoint of the line defined by Vmax Points and used to calculate Vmax or units/second.

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 or units/second) 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 (units/second) 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 units/second) 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 **!*WellID*XVals@*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 as a custom reduction 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 as a custom reduction 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.

Note: The user specifies the number of Vmax points in the Plate section Data Reduction dialog.

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 to 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 use to calculate the VmaxRate. For a complete discussion of the !WellLx accessors, see Plate Data Accessors on page 99. The !ReadInterval accessor specifies the read interval you define in the Settings dialog.

Using fewer Vmax Points than the total number of points lets the 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 to use Vmax Points, and how Vmax is calculated, see the *SoftMax Pro Data Acquisition and Analysis Software User Guide* or the application help.

Example: Viewing the Vmax Correlation Coefficient as a Plate Reduction

To display the Vmax Correlation Coefficient as a plate reduction:

1. In the Ribbon on the Home tab or in the Plate section toolbar, click the **Display Options** icon to display the Display Options dialog.
2. Choose to plot raw data with reduced number and click **OK**.
3. In the Ribbon on the Home tab, in the Plate section toolbar, or in the Workspace Reduction Settings area, click the **Data Reduction** Σ icon to display the Reduction dialog.
4. Select **Custom** for the kinetic reduction.
5. Enter 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

When a Plate section collects kinetic data for which the Data Reduction dialog sets the reduction to Vmax, use the accessor !WellValues to put the reduced number from the Plate section into the Group section.

To view the Vmax Correlation Coefficient, use 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 Accessors on page 85.

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

When you use a number to specify how many Vmax Points should be used to calculate the Vmax, the software uses the number of Vmax Points 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 number of Vmax Points you specify, the software uses only the number of points that are available. For a complete discussion of how Vmax is calculated, see the *SoftMax Pro Data Acquisition and Analysis Software User Guide* or the application help.

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

    VmaxPtsUsed(!WellLm1, !VmaxPoints, !ReadInterval)

Peak Pro Analysis Functions

Peak Pro™ Analysis functions provide advanced peak detection and characterization.

The software supports the following Peak Pro Analysis functions.

PeakProAnalysis

PeakProAnalysis(Parameter1,Parameter2,Parameter3)

- 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 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:

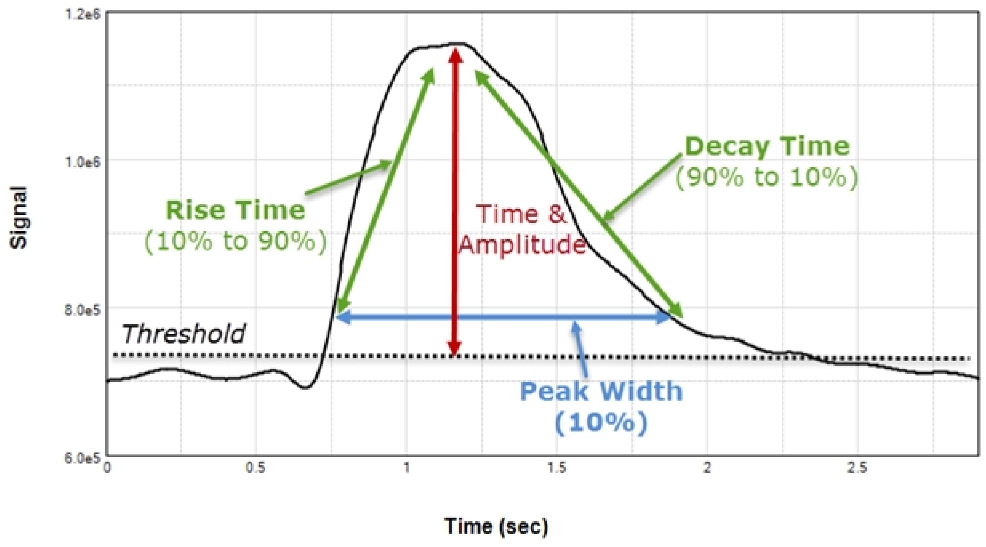
- Peak Count
- Peak Frequency (per minute)
- Peak Amplitude average
- Peak Amplitude standard deviation
- Peak Baseline average
- Peak Width average
- Peak Width standard deviation
- Peak Spacing average
- Peak Spacing standard deviation
- Peak Rise Time average
- Peak Rise Time standard deviation
- 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 to 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)
```

You need to adjust the configuration parameters based on the type of data. If multiple peak attributes are required, you should 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 start of the current epoch. See !TimeOrigin on page 95.

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 95.

LocalTimeText

LocalTimeText(Parameter)

Returns a text string of the local time and date.

UTCTimeNumeric

UTCTimeNumeric(Parameter)

Returns an array of numbers to encode 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)

Interpolation Functions

Use the interpolation functions with the !TimeOrigin accessor to align multiple kinetic data sets. See !TimeOrigin on page 95.

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 35.

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.

Array Functions

Array functions enable you to 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 used in conjunction with other functions and accessors.

The software supports the following array functions.

ArithmeticSeries

ArithmeticSeries(N,initial,increment)

Returns an array of numbers comprising an arithmetic series with N items, specifying the initial value and increment.

ArraySection

ArraySection(data,fromIndex,toIndex)

Returns an array of items from array data, specifying the range of indices.

BinnedItems

BinnedItems(data,bins)

Given a sorted (ascending order) array of numbers and an array list of bins as index pairs, returns an array list containing the binned items.

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(!WellLm1)

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)

GeometricSeries

GeometricSeries(N,initial,factor)

Returns an array of numbers comprising a geometric series with N items, specifying the initial value and multiplicative factor.

Index

Index

No parameters.

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

IndexedItems

IndexedItems(data,indices)

Returns an array containing items from the array data at the specified indices.

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 above.

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 65.

MakeBins

MakeBins(data,N)

Given a sorted (ascending order) array of numbers and number of bins N, returns N bins of uniform size as an array list of index pairs.

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(!WellLm1, 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(!WellLm1, 50, 100)

RandNormArray

RandNormArray(N)

Returns an array of N normally distributed random numbers with mean 0 and variance 1.

RandNormArrayWithSeed

RandNormArrayWithSeed(N)

As for RandNormArray but specifying the seed for the random number generator so that the result is reproducible.

RemoveIndexedItems

RemoveIndexedItems(data,indices)

Returns an array containing items from the array data with the specified items removed.

Reverse

Reverse(Parameter)

Returns the list or array parameter with items reversed.

SelectedIndices

SelectedIndices(Parameter)

Returns the indices of the items in the specified boolean array that are true.

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)

Transpose

Transpose(Parameter)

Returns the transpose of a list, array, or array list.

Array Functions Examples

The following examples show how to use array functions.

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

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@Plate1, 10, 20)

WellA2: NullOutside(!A2Lm1@Plate1, 15, 20)

WellA3: NullBetween(!A3Lm1@Plate1, 10, 34)

The NullOutside function excludes items in between the second and third 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 report 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

Used in conjunction with IndexofMax, the NullOutside function is useful for locating multiple peaks in a spectral scan.

For example, in the HolmiumOxide Group of SpectraTestABS1 validation protocols, there are columns named Peak1, Peak2 and Peak3.

The formula for the column named Peak1 is:

```
(IndexOfMax(NullOutside(!WellLm1, 20, 40))-1) * !StepSweep@Plate1 +
!StartSweep@Plate1
```

The formula reports the wavelength where the maximum OD is located within the 20th and 40th step increment.

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 92.

Example: Calculating the First Derivative of a Kinetic Plot

When you calculate the reaction rate of a Kinetic run (Vmax), you determine 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:

```
Vmax(Delta(!KinPlot), !VmaxPoints, !ReadInterval)
```

NANs and MakeErrs

The software contains a special class of numbers called NANs (Not A Number) that you use to report errors and special values. NANs are special because, although they look like text, they are actually numbers. Because NANs are numbers, the software can do numerical calculations on lists or arrays of numbers that contain NANs.

NANs propagate through most operations and functions in the software.

The software uses some NANs to report errors and other messages in Plate sections, Cuvette Set sections, Group sections, and Graph sections.

You can use other NANs to create user-defined custom formulas.

NANs as MakeErr Codes

NAN	Returns
MakeErr(101)	“” An empty number.
MakeErr(102)	“Name?”
MakeErr(103)	“Masked” Usually seen in a Group section column when wells in a Plate section or Cuvette Set section have been masked.
MakeErr(104)	“NoFit” Usually seen in Graph section when no fit has been chosen from the list of curve fits.
MakeErr(105)	“FitError” Usually seen in Graph section when is unable to apply the chosen curve fit to the data set.
MakeErr(106)	“Range?”
MakeErr(107)	“?????” For use in custom formulas.
MakeErr(108)	“Error”
MakeErr(109)	“Domain” Internal program use.
MakeErr(110)	“PrLoss” Internal program use.
MakeErr(111)	“Path?” 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?” Cuvette door was open.

NANs as MakeErr Codes (continued)

NAN	Returns
MakeErr(113)	“Limits-“ Out of reduction limits.
MakeErr(114)	“Limits+” Out of reduction limits.
MakeErr(115)	“Pass” For use in custom formulas.
MakeErr(116)	“Fail” For use in custom formulas.
MakeErr(117)	“High” For use in custom formulas.
MakeErr(118)	“Low” For use in custom formulas.
MakeErr(119)	“>>>>>” For use in custom formulas.
MakeErr(120)	“<<<<<” For use in custom formulas.
MakeErr(121)	“+++++” For use in custom formulas.
MakeErr(122)	“- - - - -“ For use in custom formulas.
MakeErr(123)	“*****” For use in custom formulas.
MakeErr(124)	“&&&&” For use in custom formulas.
MakeErr(125)	“#Low” For use in custom formulas.
MakeErr(126)	“#Sat” For use in custom formulas.
MakeErr(128)	“Negative” For use in custom formulas.
MakeErr(129)	“Positive” For use in custom formulas.
//Peak Pro	
MakeErr(130)	L "N/A"
MakeErr(131)	L"Normal"
MakeErr(132)	L"Extra Peaks"

NANs as MakeErr Codes (continued)

NAN	Returns
MakeErr(133)	L"Missing Peaks"
MakeErr(134)	L"Irregular Peaks"
//User-defined functions	
MakeErr(136)	L"Invalid Function"
MakeErr(137)	L"Invalid Parameters"
MakeErr(138)	L"Invalid Data"
MakeErr(139)	L"Outlier"

NANs Manipulation Functions

The following functions enable you 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

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 frequently used as part of conditional statements and are rarely used by themselves.

Use NANs to 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 enable you to manipulate text strings.

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.

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 16.

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

NowWithFormat

NowWithFormat(Parameter)

Returns the current date and time, with format specified by the text parameter. This is a generalization of the Now operator, for which the format is not configurable. The format string is as specified for the standard C library function *wcsftime*.

NowWithFormat("%A, %B %d %Y, %H:%M:%S") = Wednesday, January 10 2018, 10:59:09

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”

TextToArray

TextToArray("Text1","Text2")

Returns an array of text strings in Text1, with the separator character specified by Text2. An array of separators may also be specified.

TextToArray("500 600"," ") = 500
600

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

Example: Combining Text with Numbers to Report a Results Column

When you want to flag certain results in a column and you need to have the flag and the numerical results in the same column, you can convert the numbers to text.

Keep in mind that after the 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 software reports the numbers 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.

Curve Fit Functions

The functions in this section make it possible to perform least-squares regression without reference to a graph section. Data are specified as an array list x ~ y or x ~ y ~ w, where x, y, and w are lists of numbers for the independent variable, dependent variable and weighting respectively. Note that x ~ y1 ~ w1 ~ y2 ~ w2 ~..~ yN ~ wN may be used to specify N-replicates for each x. All lists must have the same number of items.

There are four ways to specify curve fit functions.

- Explicit Curve Fit Functions (see below)
- Implicit Curve Fit Functions on page 82
- 3D Curve Fit Functions on page 82
- Differential Curve Fit Functions on page 83

Explicit Curve Fit Functions

An explicit curve fit function is specified either by a function selector (as obtained from the FunctionSelector function) or as a formula f(x). The formula may be a text string, or optionally an array of strings with the parameter derivatives following the formula. The syntax is as in summary formulas, except that document references such as accessors are not allowed. The independent variable is denoted x.

The following functions accept an explicit curve fit function.

CurveFitParameters

CurveFitParameters(data,functionSelector)

Given data and function selector, returns a list of the parameter estimates.

CurveFitSumOfSquaredErrorsProfile

CurveFitSumOfSquaredErrorsProfile
(data,functionSelector,parameterIndex,parameterValues)

Given data, a function selector, a parameter index and list of parameter values, returns a list with the sum-of-squared errors for each parameter value.

FunctionDerivative

FunctionDerivative(functionSelector, parameters, x)

Given a function, a list of the parameter values, and a number or list of numbers x, returns the function derivative value(s) at x.

FunctionIntegral

FunctionIntegral(functionSelector, parameters, xRange)

Given a function, parameter values, and an array list of x ranges in the form x1 ~ x2, returns an array contining the definite integrals of the function over those ranges.

FunctionInverseValue

FunctionInverseValue(functionSelector, parameters, yWithRange)

Given a function, parameter values, and y with a range x1, x2 for its inverse in the form of an array list y ~ x1 ~ x2, returns x such that y = f(x) and x1 <= x <= x2.

FunctionParameterDerivatives

FunctionParameterDerivatives(functionSelector, parameters, x)

Given a function, a list of the parameter values, and a number x, returns the derivatives of the function at x with respect to the parameters.

FunctionSelector

FunctionSelector(key)

Returns a whole number which may be used as a selector for a standard curve fit function identified by a text key from the following list: LINEAR, QUADRATIC, CUBIC, QUARTIC, SEMILOG, LOGLOG, 4P, 5P, EXP, RH, 2PEXP, BIEXP, BIRH, 2SITECOMP, GAUSSIAN, BC, 5PALT, RHL.The key is not case sensitive.

FunctionValue

FunctionValue(functionSelector, parameters, x)

Given a function, a list of the parameter values, and a number or list of numbers x, returns the function value(s) at x.

GlobalRegressionMetrics

GlobalRegressionMetrics(data, functionSelector, initialParameters, indexMap)

As for GlobalRegressionParameters, but returns associated regression metrics as a list of numbers: the sum-of-squared errors, degrees of freedom, and coefficient of determination (R^2).

GlobalRegressionParameterCovarianceMatrix

GlobalRegressionParameterCovarianceMatrix(data, functionSelector, initialParameters, indexMap)

As for GlobalRegressionParameters, but returns the associated parameter covariance matrix.

GlobalRegressionParameters

GlobalRegressionParameters(data, functionSelector, initialParameters, indexMap)

Given multiple data sets in the form of an array list x1 ~y 1 ~ w1 ~ x2 ~ y2 ~ w2, ...xN ~y N ~ wN, a function, initial parameter values, and an index map specifying which parameters are used for which sets in the form of an array of lists of parameter indices, returns a list of the parameter estimates.

ParameterCovarianceMatrix

ParameterCovarianceMatrix(data,functionSelector,parameters)

Given data, a function, and a list of the parameter values, returns the parameter covariance matrix as an array list of numbers.

ParameterIndependenceValues

ParameterIndependenceValues(data, functionSelector, parameters)

Given data, a function, and a list of the parameter values, returns the independence values for all parameters.

ParameterMarginalIntervals

ParameterMarginalIntervals(data, functionSelector, parameters, SSE)

Given data, a function, a list of the parameter values, and a sum-of-squared errors, returns the marginal confidence intervals for all parameters.

RegressionMetrics

RegressionMetrics(data,functionSelector,Parameters)

Given data, a function, and parameter values, returns a list of numbers: the sum-of-squared errors, degrees of freedom, and coefficient of determination (R^2).

RegressionParameters

RegressionParameters(data,functionSelector,initialParameters)

Given data, a function, and a list of initial parameter values, returns a list of the parameter estimates.

RegressionParametersFixable

RegressionParametersFixable(data,functionSelector,initialParameters,parametersVariable)

As for RegressionParameters, but allowing some parameters to be fixed, as specified by the array of booleans parametersVariable, with a variable parameter designated as true.

Implicit Curve Fit Functions

The following functions accept a formula f(x, y) defining an implicit curve fit function y(x) such that f(x, y) = 0. The formula is specified by a text string referencing variables x and y.

The software supports the following implicit curve fit functions.

ImplicitFunctionInverseValue

ImplicitFunctionInverseValue(functionFormula, parameters, yWithRange)

Given an implicit function, parameter values, and y with a range x1, x2 for its inverse in the form of an array list y ~ x1 ~ x2, returns x such that f(x, y) = 0 and x1 <= x <= x2.

ImplicitFunctionValue

ImplicitFunctionValue(functionFormula, parameters, x)

Given an implicit function, a list of the parameter values, and a number or list of numbers x, returns the function value(s) at x.

ParameterCovarianceMatrixImplicit

ParameterCovarianceMatrixImplicit(data, functionFormula, parameters)

Given data, an implicit function, and a list of parameter values, returns the parameter covariance matrix as an array list of numbers.

RegressionParametersImplicit

RegressionParametersImplicit(data,functionFormula,initialParameters)

Given data, an implicit function, and initial parameter values, returns a list of the parameter estimates.

3D Curve Fit Functions

The following functions accept a formula defining a curve fit function of two independent variables f(x, y). The formula is specified by a text string referencing variables x and y. 3D data must be an array list with format x ~ y ~ z or x ~ y ~ z ~ w, where x, y, z, are lists of the data and w a list of the associated weights. All lists must have the same number of items.

The software supports the following 3D curve fit functions.

FunctionValue3D

FunctionValue3D(functionFormula, parameters, xyPoints)

Given a function, a list of the parameter values, and an array of lists of x and y values, returns the function value(s) at the specified x and y points.

MinimizeFunctionSumOverData

MinimizeFunctionSumOverData(data, functionFormula,initialParameters)

Given data, a function, and a list of initial parameter values, returns a list of the parameters that minimize the function.

RegressionParameters3D

RegressionParameters3D(data, functionFormula,initialParameters)

Given data, a function, and a list of initial parameter values, returns a list of the parameter estimates.

RegressionParameters3DFixable

RegressionParameters3DFixable(data, functionFormula, initialParameters, parametersVariable)

As for RegressionParameters3D, but allowing some parameters to be fixed, as specified by the array of booleans parametersVariable, with a variable parameter designated as true.

Differential Curve Fit Functions

The following functions accept a formula to define a function that is a solution of a system of first-order differential equations. A differential function is specified by an array list of text strings with one row for each component f_i with the format format f_i(initialX) ~ df_i /dx or f_i (initialX) ~ df_i /dx ~ J_i ~ dJ_i dx, where J denotes the Jacobian matrix. In the formula, components are identified by the symbols f1, f2, ... ,f8. The value of the function is that of the first component (f1).

The software supports the following differential curve fit functions.

RegressionParametersDifferential

RegressionParametersDifferential(data, initialX, functionFormula, initialParameters)

Given data, the initial x value, a function, and initial parameter values, returns a list of the parameter estimates.

DifferentialFunctionValue

DifferentialFunctionValue(initialX, functionFormula, Parameters,x)

Given data, a function, parameter values, and a number or list of numbers x, returns the function value(s) at x.

Miscellaneous Functions

The software supports the following miscellaneous functions.

KMeansClusters

KMeansClusters(data,N,i)

Given a multi-dimensional data set, specified by an array list of numbers, data, with a column for each dimension, the number of clusters, N, and the number of iterations, i, returns the clusters. The output is an array list with a row of data item indices for each cluster (can be used in KASP assay (genotyping) analysis).

LinearDiscriminantVector

LinearDiscriminantVector(Parameter1,Parameter2)

Returns the linear discriminant vector for two multi-dimensional data sets, parameter1 and parameter2. Each data set is specified by an array list of numbers, with a column for each dimension.

LinearSolution

LinearSolution(Parameter1,Parameter2)

Returns the solution of a system of linear equations Ax = b; parameter1 is an array list of numbers specifying the coefficients, A, and parameter2 is the right hand-side b. The returned array is the solution x (can be used for multi-component analysis).

PrincipalComponents

PrincipalComponents(Parameter)

Returns the principal component vectors of a multi-dimensional data set specified as an array list of numbers, with a column for each dimension.

PrincipalComponentsEigenvalues

PrincipalComponentsEigenvalues(Parameter)

Returns the principal component eigenvalues of a multi-dimensional data set specified as an array list of numbers, with a column for each dimension.

RowIndices

RowIndices(Parameter)

Given an array list of numbers, returns an array containing the index of the row in which each number occurs. Use in conjunction with KMeansClusters to determine cluster assignments.

Chapter 4: Accessors

4

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

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

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@PlateX, !a2Lm1@PlateX, and so on in Column or Summary formulas.

You can use Accessors by themselves as Column formulas and you can use Accessors as parameters within several functions, such as kinetic and spectrum Reduction functions. You can also combine Accessors with functions and mathematical operators to create complex formulas.

Accessors are grouped in the following categories:

- Plate and Cuvette Set Setup Accessors on page 86
- Kinetic and Spectrum Data Accessors on page 92
- Plate Data Accessors on page 99
- Injector Data Accessors on page 106
- PathCheck Technology Accessors on page 111
- Group and Well Information Accessors on page 113
- Blank Accessors on page 116
- Imaging Data Accessors on page 118

Using !Well as an Accessor Prefix

Accessors 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 Plate1 and wells G1, H1, G2, H2 on Plate2 if they are part of the same group).

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

!A1*Accessor*

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

!Well*Accessor*

The !Well prefix is used only in Group sections. Use of 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 113.

Plate and Cuvette Set Setup Accessors

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

The software supports the following Plate and Cuvette Set setup information accessors.

!AutoCutoffOn

!AutoCutoffOn

!WellAutoCutoffOn

For Fluorescence mode only.

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

For Fluorescence Polarization mode only.

!InstrumentSerial

!InstrumentSerial

Returns the serial number for SpectraMax L, i3, i3x, iD3, iD5, FilterMax, and Paradigm instruments.

!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

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 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”, “Low”, or "Manual" 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 92.

!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 92.

!ReadType

!ReadType

!WellReadType

Not available on all instruments.

Returns the text string “Absorbance”, “Fluorescence”, “Luminescence”, “Time Resolved”, or "Fluorescence Polarization" 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 Accessors Example

The following example shows how to use plate setup information accessors.

Example: Using Plate Information Accessors in Summary Lines

If a data file contains data from more than one plate, use !TimeOfRead to report the start and end time of the experiment. 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 enable you to:

- 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).

The 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 you select to shake between reads for Kinetic plates in the Settings dialog and False if the check box is not selected.

!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/RFU/RLU set in the Reduction dialog for a Spectrum scan or Kinetic run.

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

!MinLimit

!MinLimit

!WellMinLimit

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

Data collected below the Min value 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 66.

!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 94.

!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 start of the current epoch when the first measurement was recorded. 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 55.

!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 or !WavelengthRun above.

Kinetic and Spectrum Data Accessors Examples

The following examples shows how to use plate setup information accessors.

Example: Accessing Kinetic Run Settings Information

Use the !NumPoints and !NumPointsRead accessors to determine the number of points specified for a Kinetic run and the number of points actually collected if the run stops 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@Plate1, !ReadInterval@Plate1)
```

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 you need to identify the Plate section or Cuvette Set section in the formulas; for example, !StartSweep@Plate1 or !EndSweep@CuvetteSet1.

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

If you use the Well prefix 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 displays 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 is 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 enable you to 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.

The 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

Returns the result of the Wavelength Options reduction formula for each well or cuvette.

!LmX or !WellLmX return information before the Wavelength Options reduction is applied. See !LmX below.

!LmX

!LmX

!A1LmX

!WellLmX

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

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

!LmX accesses the raw values (not in template order) 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 returns the raw values in template order in a Group section. The values 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 parallel RFU values for wavelength X.

!WellLmXPRaw reports the raw parallel RFU values for wavelength X in template order in a Group section.

!A1LmXPRaw

!A1LmXPRaw

For Fluorescence Polarization readings only.

Reports the raw parallel RFU values for wavelength X for the designated well.

!Lm1DualReadM

!Lm1DualReadM

Returns the raw RLU value from each well of a Plate section for the Lm1 wavelength reading taken after an M-injection.

Note: This accessor applies to the SpectraMax L Microplate Reader.

!Lm1DualReadP

!Lm1DualReadP

Returns the raw RLU value from each well of a Plate section for the Lm1 wavelength reading taken after a P-injection.

Note: This accessor applies to the SpectraMax L Microplate Reader.

!Lm2DualReadM

!Lm2DualReadM

Returns the raw RLU value from each well of a Plate section for the Lm2 wavelength reading taken after an M-injection.

Note: This accessor applies to the SpectraMax L Microplate Reader.

!Lm2DualReadP

!Lm2DualReadP

Returns the raw RLU value from each well of a Plate section for the Lm2 wavelength reading taken after a P-injection.

Note: This accessor applies to the SpectraMax L Microplate Reader.

!LmXSRaw

!LmXSRaw

!WellLmXSRaw

For Fluorescence Polarization readings only.

Reports the raw perpendicular RFU values for wavelength X.

!WellLmXSRaw reports the raw perpendicular RFU values for wavelength X in template order in a Group section.

!A1LmXSRaw

!A1LmXSRaw

For Fluorescence Polarization readings only.

Reports the raw perpendicular RFU values for wavelength X for the designated well.

!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

Returns the raw RLU value from each well of a Plate section for the Lm1 wavelength for reads following the M-injection.

Note: This accessor applies to the SpectraMax L Microplate Reader.

!PlateBlankLm1DualReadM

!PlateBlankLm1DualReadM

Returns the plate blank M read for wavelength Lm1.

Note: This accessor applies to the SpectraMax L Microplate Reader.

!PlateBlankLm2DualReadM

!PlateBlankLm2DualReadM

Returns the plate blank M read for wavelength Lm2.

Note: This accessor applies to the SpectraMax L Microplate Reader.

!PlateBlankLm1DualReadP

!PlateBlankLm1DualReadP

Returns the plate blank P read for wavelength Lm1.

Note: This accessor applies to the SpectraMax L Microplate Reader.

!PlateBlankLm2DualReadP

!PlateBlankLm1DualReadP

Returns the plate blank P read for wavelength Lm2.

Note: This accessor applies to the SpectraMax L Microplate Reader.

!PlateBlankLmX

!PlateBlankLmX

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

!PlateBlankP

!PlateBlankP

For Fluorescence Polarization readings only.

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

    !PlateBlankLm1P

!PlateBlankLmXP

!PlateBlankLmXP

For Fluorescence Polarization readings only.

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

    !PlateBlankLm1P

!PlateBlankS

!PlateBlankS

For Fluorescence Polarization readings only.

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

    !PlateBlankLm1S

!PlateBlankLmXS

!PlateBlankLmXS

For Fluorescence Polarization readings only.

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

    !PlateBlankLm1S

!PlateKind

!PlateKind

Returns the type of plate selected in the Settings dialog.

!PValue

!PValue

Returns the raw RLU value from each well of a Plate section for the Lm1 wavelength for reads following the P-injection.

Note: This accessor applies to the SpectraMax L Microplate Reader.

!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

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

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 plate 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.

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

Use the following formula to calculate a column to show the reference value used for each cuvette:

!WellPreReadLm1

Creating a column of this kind is quite useful if you use 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 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

Injector Data Accessors

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

The 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 returns as 0.
- If injector 1 is not used, then the value returns as empty.

Use this accessor to retrieve the specified delay time for report 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 returns as 0.
- If injector 2 is not used, then the value returns as empty.

Use this accessor to retrieve the specified delay time for report purposes.

!Injector1Volume

!Injector1Volume

!WellInjector1Volume

For the reads with injection only.

Returns the volume of the injection in µL for injector 1. The expected values are 1 to the maximum volume of the well.

Use this accessor to retrieve the injected volume for activity calculation in a well.

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

!Injector2Volume

!Injector2Volume

!WellInjector2Volume

For the reads with injection only.

Returns the volume of the injection in µL for injector 2. The expected values are 1 to the maximum volume of the well.

Use this accessor to retrieve the injected volume for activity calculation in a well.

If injector 2 is not used, then the value returns 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.

Use this accessor to retrieve the specified integration time or number of flashes for report purposes.

If injector 1 is not used, then the value returns 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.

Use this accessor to retrieve the specified integration time or number of flashes for report purposes.

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

!MInjectionDelay

!MInjectionDelay

For the reads with injection only.

Returns the delay time in seconds between the M-injection and the read for the M-injection.

The expected values are 1 to 3600 seconds.

- If no delay step is used, then the value returns as 0.
- If the M-injection is not used, then the value returns as empty.

Use this accessor to retrieve the specified delay time for report purposes.

Note: This accessor applies to the SpectraMax L Microplate Reader.

!MInjectionShakeTime

!MInjectionShakeTime

For the reads with injection only.

Use this accessor to return the shake duration time in seconds of the M-injection.

Note: This accessor applies to the SpectraMax L Microplate Reader.

!MInjectionSpeed

!MInjectionSpeed

For the reads with injection only.

Use this accessor to return the speed in seconds of the M-injection and the read for the M-injection.

Note: This accessor applies to the SpectraMax L Microplate Reader.

!MInjectionVolume

!MInjectionVolume

For the reads with injection only.

Returns the volume of the M-injection in µL. The expected values are 10 to the maximum volume of the well.

Use this accessor to retrieve the injected volume for activity calculation in a well.

If M-injection is not used, then the value returns as empty.

Note: This accessor applies to the SpectraMax L Microplate Reader.

!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 returns as 0.
- If injector 1 is not used, then the value returns as empty.

Use this accessor to calculate the average of the baseline read data, and then subtract 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 returns as 0.
- If injector 2 is not used, then the value returns as empty.

Use this accessor to calculate the average of the baseline read data, and then subtract the value from the post-injection data.

!PInjectionDelay

!PInjectionDelay

For the reads with injection only.

Returns the delay time in seconds between the P-injection and the read for the P-injection.

The expected values are 1 to 3600 seconds.

- If no delay step is used, then the value returns as 0.
- If the P-injection is not used, then the value returns as empty.

Use this accessor to retrieve the specified delay time for report purposes.

Note: This accessor applies to the SpectraMax L Microplate Reader.

!PInjectionShakeTime

!PInjectionShakeTime

For the reads with injection only.

Returns the shake duration time in seconds of the P-injection.

Note: This accessor applies to the SpectraMax L Microplate Reader.

!PInjectionSpeed

!PInjectionSpeed

For the reads with injection only.

Returns the speed in seconds of the P-injection and the read for the P-injection.

Note: This accessor applies to the SpectraMax L Microplate Reader.

!PInjectionVolume

!PInjectionVolume

For the reads with injection only.

Returns the volume of the P-injection in µL. The expected values are 10 to the maximum volume of the well.

Use this accessor to retrieve the injected volume for activity calculation in a well.

If the P-injection is not used, then the value returns as empty.

Note: This accessor applies to the SpectraMax L Microplate Reader.

Injector Data Accessors Example

The following example shows how to use injector data accessors.

Example: Subtracting Baseline Data from Post Injection Data

Use a custom formula in the Wavelength Options step in 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 Column formulas in Group sections or in Summary formulas. These accessors are not used in Cuvette Set sections.

The software supports the following PathCheck Technology accessors.

!PathBackground

!PathBackground

!PathBackgroundLm1, ... ,!PathBackgroundLm6

Returns the plate background constant that is set for the specified wavelengths.

!PathCheckApplied

!PathCheckApplied

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

!PathCheckLm900

!PathCheckLm900

Returns a number that represents 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 the cuvette reference or the Water Constant.

!PathCheckLm1000

!PathCheckLm1000

Returns a number that represents 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 setting is set to use the cuvette reference or the Water Constant.

!PathCheckOn

!PathCheckOn

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

!PathCorrectionOn

!PathCorrectionOn

Returns True if the Plate section is 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 to use PathCheck Technology accessors.

Example: Accessing PathCheck Values in a Group Section

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

Use the formulas !PathCheckLm900 and !PathCheckLm1000 to return the optical density reads for the cuvette reference at 900 nm and 1000 nm.

If the Water Constant is used instead of a cuvette reference, these are the Water Constant values.

Example: Viewing Pathlength Together With Pathlength Corrected Optical Density in Group Sections

If you enable Pathlength correction, 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 dialog. Several of these accessors are the default formulas for the columns that are created when you create new groups in the Template Editor dialog.

The 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 return in lexicographical order according to the column name. All column formulas return, 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 return in lexicographical order. All column names return, 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 you assign in the Sample Descriptor field in the Template Editor, usually concentration for standards or a dilution factor for unknowns.

Use the sample descriptor value 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, use time as a sample descriptor.

!Factor

!Factor

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

Use the sample descriptor value 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, use time as a sample descriptor.

!SampleDescriptor

!SampleDescriptor

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

Use the sample descriptor value 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, use time as a sample descriptor.

!SampleNames

!SampleNames

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

!SampleNumbers

!SampleNumbers

Returns the number of the sample.

!SummaryFormulas

!SummaryFormulas

Returns a list of the Summary formulas in the section you specify.

The formulas return in the order that they are created.

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

!SummaryNames

!SummaryNames

Returns a list of the summary names in the section you specify.

The names return in order that they are created.

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

!Units

!Units

Returns, as a text string, the units you assign 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 plate, A1 = 1 and H12 = 96.

Blank Accessors

Blank accessors enable you to access and manipulate plate blank and group blank information. The average of the plate blank displays in the Plate section for Endpoint reads. The average of the group blank displays 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).

The software supports the following blank accessors. For more blank accessor descriptions, see Plate Data Accessors on page 99.

!GroupBlank

!GroupBlank@GroupName

Returns, as a number, the average of all wells that you designate as group blanks for the specified group.

!GroupBlankValues

!GroupBlankValues@GroupName

Returns, as a list of numbers, the values of the individual wells that you designate as group blanks for the specified group.

!PlateBlank

!PlateBlank

Returns the average plate blank value as a number.

Blank Accessors Examples

Example: Accessing Group Blank Information

Use the following formulas 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)

Ex: Subtracting Single Plate Blank Value from Multiple Plate Sections

To subtract the same plate blank from multiple groups on different plates, 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 Plate1. The custom formula for this reduction is:

    !Lm1-!PlateBlank@Plate1

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 is subtracted twice from the samples on this plate, once in the Plate section and again in the Group section.
3. Define the samples that belong 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@Plate1

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 the Imaging read mode only.

Imaging data accessors enable you to 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 you define 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 you use 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 85.

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 Settings dialog. Separate measurements are available for each measured wavelength and classification, if applicable.

- **Object Count**: The total number of objects detected in the image. Not used for field analysis of confluent areas.
- **Field Count**: The total number of confluent areas detected in the image. Not used for discrete object analysis.
- **Object Percentage**: The percentage the objects detected in the image by classification. Not used for field analysis of confluent areas.
- **Covered Area**: The combined area of all the objects or confluent areas detected in the image as a percentage of the entire image area.
- **Object Area**: The average area of the objects detected in the image expressed in µm^2. Not used for field analysis of confluent areas.
- **Field Area**: The average area of the confluent areas detected in the image expressed in µm^2. Not used for discrete object analysis.
- **Object Roundness**: 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. Not used for field analysis of confluent areas.
- **Field Roundness**: 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. Not used for discrete object analysis.
- **Object Average Intensity**: The average fluorescent signal intensity of the objects detected in the image. This measurement is not used for field analysis of confluent areas. Not used for transmitted light.
- **Field Average Intensity**: The average fluorescent signal intensity of the confluent areas detected in the image. This measurement is not used for discrete object analysis and not used for transmitted light.
- **Object Intensity**: The average total fluorescent signal intensity of the objects detected in the image. Not used for field analysis of confluent areas and not used for transmitted light.
- **Field Intensity**: The average total fluorescent signal intensity of the confluent areas detected in the image. Not used for discrete object analysis and not used for transmitted light.

- **Total Intensity**: The combined total fluorescent signal intensity of the objects or confluent areas detected in the image expressed in million intensity counts. Not used for transmitted light.

SoftMax Pro Software - GxP Edition Accessors

For users of the SoftMax Pro Software - GxP edition, the following accessors are available.

!ReadByUser

!ReadByUser

For the SoftMax Pro Software - GxP edition, returns the name of the current user at the point a plate or cuvette is read. For the SoftMax Pro Software - standard edition it returns empty text.

Index

Contact Us

Phone:	+1-800-635-5577
Web:	moleculardevices.com
Email:	info@moldev.com

Visit our website for a current listing of worldwide distributors.