



# **SoftMax® Pro Software**

## **Formula Reference Guide**

5000941 B  
July 2010

---

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

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

Portions of this document may make reference to other manufacturers and/or their products, which may contain parts whose names are registered as trademarks and/or function as trademarks of their respective owners. Any such usage is intended only to designate those manufacturers' products as supplied by Molecular Devices for incorporation into its equipment and does not imply any right and/or license to use or permit others to use such manufacturers' and/or their product names as trademarks.

Molecular Devices makes no warranties or representations as to the fitness of this equipment for any particular purpose and assumes no responsibility or contingent liability, including indirect or consequential damages, for any use to which the purchaser may put the equipment described herein, or for any adverse circumstances arising therefrom.

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



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

Product manufactured by Molecular Devices, Inc.

1311 Orleans Drive, Sunnyvale, California, United States of America 94089.

Molecular Devices, Inc. is ISO 9001 registered.

© 2010 Molecular Devices, Inc.

All rights reserved.

Printed in the USA.

---

# Contents

---

<b>Chapter 1 Introduction</b>	<b>5</b>
Formula Categories	5
Reduction Formulas	5
Column Formulas	6
Summary Formulas	7
Formula Size	8
Formula Building Blocks	8
Lists of Numbers and Arrays of Numbers	9
Formula Naming Conventions in SoftMax® Pro Software	9
Referencing Objects in Different Sections	9
Ordering Functions in Complex Formulas	10
Rules to Follow when Writing Formulas	10
Case	10
Special Characters	11
Referencing Columns	11
Evaluating Column Formulas	11
Strings of Formulas	11
<b>Chapter 2 Operators</b>	<b>13</b>
Mathematical Operators	13
Using Mathematical Operators	14
Comparison (Logical) Operators	15
Conditional (Boolean) Operators	16
Using Comparison and Conditional Operators	17
Concatenation Operators	23
Concatenating Text Strings	23
Concatenating Numbers	23

<b>Chapter 3 Functions . . . . .</b>	<b>27</b>
Mathematical Functions . . . . .	27
Statistical Functions . . . . .	31
Interpolation Functions . . . . .	33
VMax® Reduction Functions . . . . .	40
Other Functions . . . . .	44
NaNs/MakeErrs . . . . .	49
Text Functions . . . . .	52
<b>Chapter 4 Accessors . . . . .</b>	<b>55</b>
Plate/CuvetteSet Setup Information Accessors . . . . .	56
Kinetic and Spectrum Data Accessors . . . . .	59
Plate Data Accessors . . . . .	64
PathCheck® Accessors . . . . .	68
Group and Well Information Accessors . . . . .	69
Blank Accessors . . . . .	70
<b>Index. . . . .</b>	<b>73</b>

# Introduction

---

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

Chapter 1 provides a general introduction to SoftMax Pro formulas, while Chapters 2 through 4 list the building blocks used in SoftMax Pro Software formulas and give brief examples of how to use them.

This Formula Reference Guide assumes that the user understands the basics of SoftMax Pro software. For more information on using the software, please review the SoftMax Pro Software User Guide.

## Formula Categories

Formulas are broken down into categories:

- Reduction formulas are used in Plate and CuvetteSet sections and apply a formula to an entire Plate section or CuvetteSet section.
- Column formulas are used in Group sections and apply a formula to a single column in the Group section.
- Summary formulas may be used in either Group or Notes sections and are generally used to reduce data to a single value or array.

### Reduction Formulas

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

**Wavelength Combination:** This formula modifies raw data. It is available for all read modes with all instruments and is the only reduction available for Endpoint or Dual Read (not available on all instruments; please see your specific instrument manual for more information). These formulas act on all data from each well or cuvette in the Plate/CuvetteSet sections.

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

## Entering a Custom Reduction Formula

If default reductions do not meet your needs, custom formulas can be created. To enter a custom formula:

1. Click the Reduction button or select **Plate > Reduction**.
2. Select Custom from one of the drop-down lists (the choices that are available in the Reduction dialog box will depend on the read mode and instrument type).
3. Click the  $=f[x]$  button to open a Calculation dialog box. This dialog box has only one field in which to create or edit a formula.
4. Enter the formula into the field and click OK.
5. Click OK in the Reduction dialog box to close that dialog box and set the parameters.

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

$!Lm1 * 1.2$

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

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

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

## Column Formulas

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

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

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



---

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

---

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

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

There are occasions when one does not want the data in a column of a Group section sorted by template order. In the case of a Spectrum scan, one might want the wavelength values in one column and OD values for a single well in another. The corresponding formulas for well B1 in Plate#X would be !Wavelengthrun@plate#X and !B1Lm1@Plate#X.

## Summary Formulas

Summaries consist of four parts:

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

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

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

## Formula Size

Formula length in SoftMax Pro software is limited to 512 characters. This limitation can be an issue when creating complex formulas or if object names are long.

It is generally recommended to break up complex formulas into several simpler formulas, and to use short object names (Experiment name, Section names, Group names, etc.).

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

Alternatively, you can use Summaries to contain intermediate steps. If the 512-character limit becomes a problem and you cannot break the formula up into several smaller formulas, you can also shorten all of the object names.

## Formula Building Blocks

SoftMax Pro software uses four types of formula building blocks:

- Operators allow you to combine other building blocks together. Operators are discussed in [Chapter 2](#).
- Functions are mathematical or text items that perform operations within SoftMax Pro software (for example, standard deviation or slope). Functions are discussed in [Chapter 3](#).
- NANs (Not A Number) are text items that are considered to be numbers by SoftMax Pro software. NANs are discussed in [Chapter 3](#).
- Accessors allow you to access data and other information from objects (Plates, Cuvette-Sets, etc.) and functions (Vmax Rate, the wavelength at which a Spectrum scan was begun, etc.). Accessors are unique to SoftMax Pro software and are discussed in [Chapter 4](#).

Some formula building blocks can be used alone. Others must always be used in conjunction with other building blocks. A complete listing of these building blocks, along with examples showing how to use them, is provided in [Chapter 2](#) through [Chapter 4](#).



## Lists of Numbers and Arrays of Numbers

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

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

## Formula Naming Conventions in SoftMax® Pro Software

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

An object's full name 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. SoftMax Pro software uses the @ symbol to combine these references:

ColumnName@GroupSectionName@ExperimentName

PlotName@GraphName@ExperimentName

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

OD@GroupSectionName1@Experiment1

OD@GroupSectionName2@Experiment1

OD@GroupSectionName1@Experiment2

## Referencing Objects in Different Sections

If a formula refers to an object that is in the same section as the formula, the object's section and experiment names are not required.

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

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

If you write:

ColX - Values

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

ColX - Values@GroupSectionName1

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

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

ColX - Values@GroupSectionName2

## Ordering Functions in Complex Formulas

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

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

Sqrt(Average(Values))

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

## Rules to Follow when Writing Formulas

### Case

SoftMax Pro 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 any of the following special characters: # \$ % / \* ?, the name must be enclosed in single quotation marks. For example, 'OD Values'.

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

## Referencing Columns

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

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

## Evaluating Column Formulas

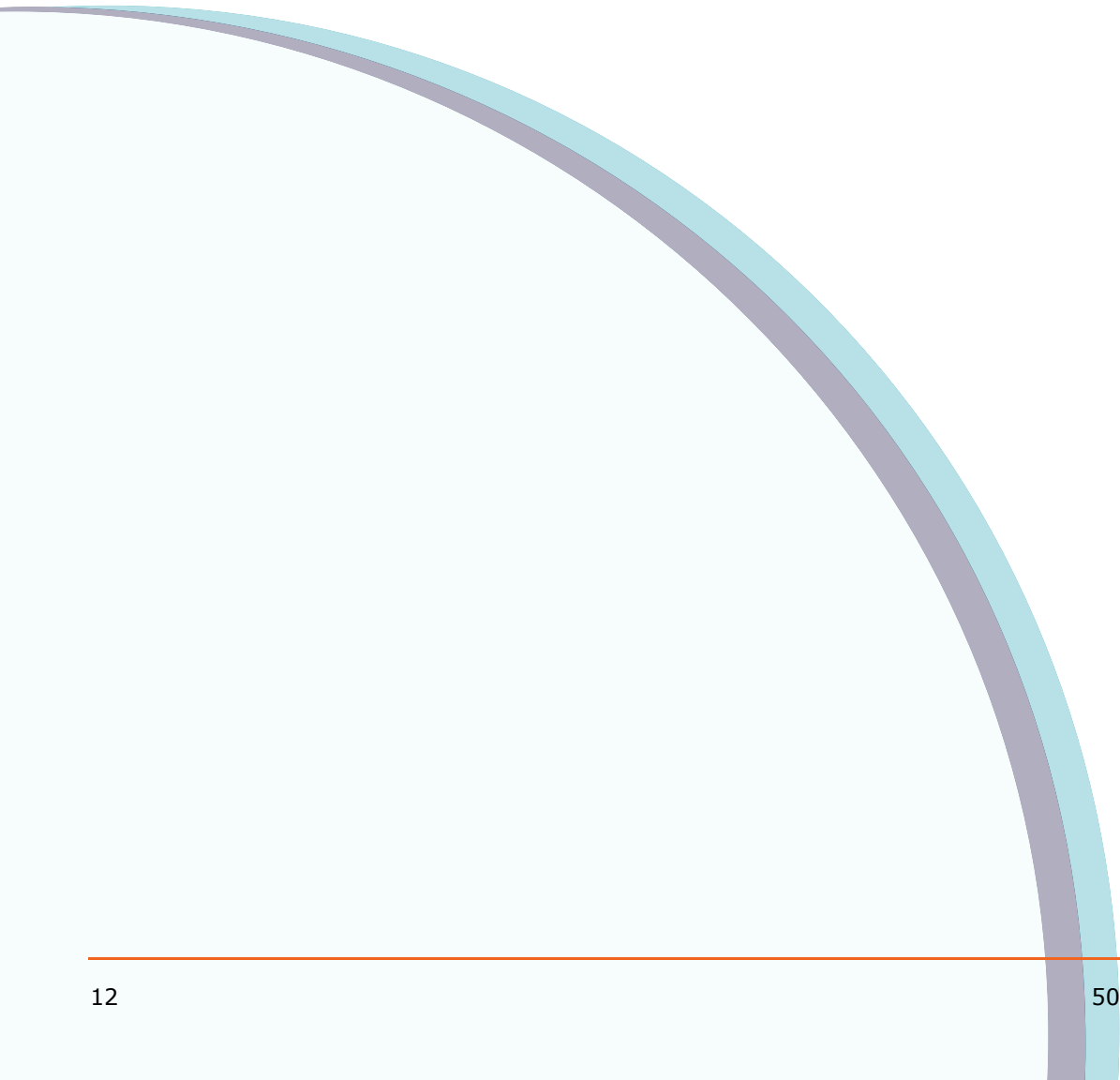
Column formulas must evaluate either to text, to numbers, or to a Boolean (i.e., to True or False) because SoftMax Pro software does not allow a column to contain both text ("Pass") and numeric ("1.002") results.

However, you can use NANs to put text into a column of numbers (because SoftMax Pro software treats NANs as numbers) or you can report numbers as text strings.

## Strings of Formulas

When a formula includes a result that the user wants displayed, these text results must be enclosed in double quotation marks (""), for example: "Out of Range".

Wavelength reduction formulas in Plate or CuvetteSet sections must evaluate to numbers (however, NANs are considered numbers and can be used to bypass this restriction).



# Operators

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

Recall that formulas beginning with !Lm1, !Lm2, etc., are suitable for Reduction formulas in Plate/CuvetteSet sections. In Group sections, the analogous formulas must begin with !WellLm1, !WellLm2, etc., to ensure sorting according to the Template. To access data from single wells, the accessors are !a1Lm1, !a2Lm1, etc., (in Reduction formulas) and !a1Lm1@Plate#X, !a2Lm1@Plate#X, etc., (in Column or Summary formulas).

## Mathematical Operators

The list below reports the mathematical operators that can be used in SoftMax Pro formulas and describes their function, with which types of data they can be used, and gives examples of how to use them.

SoftMax Pro software uses the standard mathematical order for operators: multiplication and division are performed before addition and subtraction (regardless of their order in the equation) unless you use parentheses to order the operations.

**Table 2-1** Mathematical Operators

Mathematical Operator	Description
*	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
/	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
+	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

**Table 2-1** Mathematical Operators

Mathematical Operator	Description
-	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
^	Exponent: Raises a number to another power. You can take the exponent of individual numbers, lists of numbers, or arrays of numbers. !WellLm1 ^ 2
Mod	Remainder: Returns the remainder from dividing 2 numbers. You may take the remainder for individual numbers, a list of numbers, or an array of numbers. 11 mod 3
( )	Parentheses: Used to order multiple mathematical operations; operations in inner parentheses are performed first. (!Lm1 + !A12Lm1) * !H1Lm1

**Using Mathematical Operators**

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

**Example: Multiplying Lists of Numbers**

When multiplying lists of numbers in SoftMax Pro software you can refer to them by name, or use functions (see [Chapter 3](#)) or accessors (see [Chapter 4](#)) 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 another column that is itself generated by the formula "Average(Results)".

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 PathCheck® Pathlength measurement technology selected. The optical density values were multiplied in the Group section by a constant to report the concentration of DNA in each well.

The formula for the reduced number is:

$!WellLm1 * 50$

## Comparison (Logical) Operators

Comparison operators are used to compare numbers, lists of numbers, arrays of numbers, or text strings. They are used most frequently in conditional statements (see [Conditional \(Boolean\) Operators on page 16](#)). When used by themselves, comparison operators return "True" if the argument is true, and "False" if the argument is false.

**Table 2-2** Comparison Operators

Comparison Operator	Description
=	Equals: Determines if two numbers, lists of numbers, or arrays of numbers are equal to one another. $!Lm1 = !Lm1$ is True $!Lm1 = !A12Lm1$ is False
>	Greater Than: Determines if a number, list of numbers, or array of numbers is greater than another number, list of numbers, or array of numbers. $(2*!Lm1) > !Lm1$ is True $!Lm1 > !Lm1$ is False
>=	Greater Than or Equals: Determines if a number, list of numbers, or array of numbers is greater than or equal to another number, list of numbers or array of numbers. $Lm1 >= !Lm1$ is True $(2*!Lm1) >= !Lm1$ is True $!Lm1 >= (2*!Lm1)$ is False

**Table 2-2** Comparison Operators

Comparison Operator	Description
<	Less Than: Determines if a number, list of numbers or array of numbers is less than another number, list of numbers or array of numbers. Lm1 < (2*!Lm1) is True (2*!Lm1) < !Lm1 is False
<=	Less Than or Equals: Determines if a number, list of numbers, or array of numbers is less than or equal to another number, list of numbers, or array of numbers. !Lm1 <= !Lm1 is True (2*!Lm1) <= !Lm1 is False !Lm1 <= (2*!Lm1) is True
<>	Does Not Equal: Determines if two numbers, lists of numbers, or arrays of numbers are not equal to one another. !Lm1 <> !Lm1 is False (2*!Lm1) <> !Lm1 is True

**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 (i.e., the MeanValue) is greater than 0.2 and False if it is not.

**Conditional (Boolean) Operators**

Conditional operators allow you to compare results in Plate/CuvetteSet sections, Group sections, and Summaries. They can be used to set criteria for evaluating numerical results. Group sections and Summaries can evaluate either to a number or to text, whereas Plate/ CuvetteSet sections can evaluate only to a number.

For example, an ELISA kit may ask you to determine whether a sample is positive, negative, or requires retesting based on the values of the assay controls.



**Table 2-3** Conditional Operators

Conditional Operator	Description
If	Compares numbers, lists of numbers, or arrays of numbers to each other. If (Condition, Value-If-True, Value-If-False) If (!WellLm1 > !A12Lm1@Plate#X, "Above threshold", "Below threshold") For this example: If the value at wavelength 1 in a well is greater than the value in well A12 of Plate#X, return "Above threshold", otherwise return "Below threshold".
And	Compares numbers, lists of numbers, or arrays of numbers. Returns True if all of the arguments are true. Returns False if any of the arguments are not true. WellLm1 > !A12Lm1@Plate#X And !WellLm2 > !A12Lm2@Plate#X
Or	Compares numbers, lists of numbers, or arrays of numbers. Returns True if any of the arguments are true. Returns False only if all of the arguments are false. WellLm1 > !A12Lm1@Plate#X Or !WellLm2 > !A12Lm2@Plate#X
Not	Reverses the logic of an argument. If something is not true, it is false; if it is not false, it is true. Not(!WellLm1 > !A12Lm1@Plate#X And !WellLm2 > !A12Lm2@Plate#X)
False	Returns the logical value False.
True	Returns the logical value True.

## Using Comparison and Conditional Operators

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

**Simple Conditional Formulas**

Conditional formulas have the general form:

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

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

**Table 2-4**

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 > Summary#1@Control and MeanValue < Summary#2@Control, "Acceptable", "Out of Specification")	MeanValue > Summary#1@Control and MeanValue < Summary#2@Control "Acceptable" "Out of Specification"

**Nested Conditionals**

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

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

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

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

You can create very complicated formulas by nesting multiple conditional statements, with the following constraints:

- The entire formula must not exceed 512 characters.
- The formula must specify an outcome if true and an outcome if false for each condition.
- The formula must state a condition, the outcome if true; then a new nested condition if the previous condition was false and the outcome if the second condition is true; and so on until the last condition, with final outcomes for if the last condition is true or false.

The general format is:

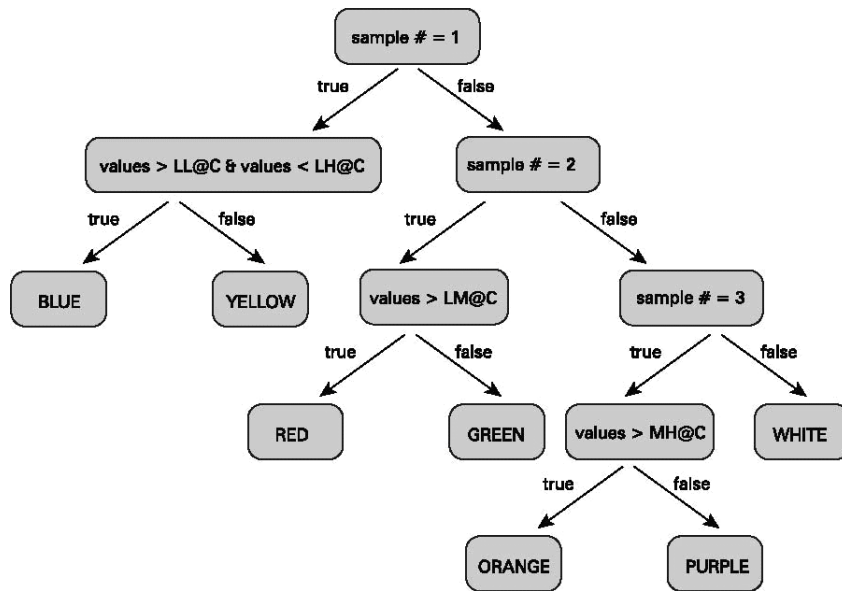
```
If(Condition#1, outcome when Condition#1 is True,  
(if(Condition#2, outcome when Condition#2 is  
True,(if(Condition#3, outcome if Condition#3 is True...(if  
Condition#X, outcome if Condition#X is True, outcome if  
Condition#X is False))))...).
```

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

An example of this type of complex, nested conditional statement is given below. The formula has been written to include several samples with 5 different controls. The information for the different controls is provided by Summary formulas named LL, LH, LM, MH and HH, respectively, and the Summary formulas are located in a Notes section named C.

### **Formula**

```
If( Sample#=1,  
If( Values>LL@C and Values<LH@C,"Blue", "Yellow"),  
If( Sample#=2,  
If( Values>LM@C, "Red", "Green"),  
If( Sample#=3,  
If( Values>MH@C and Values<HH@C,"Orange", "Purple" ),  
"White" )))
```



**Figure 2-1** Diagram

### 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 have any data, the formula evaluates to the default (False condition) and reports "Out of Specification".

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

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

- The first condition stipulates if the values for Lm1 in all wells are less than the value of Lm1 in well A12, report the NAN MakeErr(118) (which displays the word "Low"); if they are equal to or greater than the value in well A12, apply the second condition.
- The second condition stipulates if the values for Lm1 in all wells are greater than the value of Lm1 in well H1, report NAN MakeErr(117) (which displays the word "High"); otherwise, if this condition is not true, report NAN MakeErr(123) (which displays "\*\*\*\*\*").

NANs, including MakeErr functions, are discussed in detail in Chapter 3, "Functions".

#### **Formula**

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

#### **Breakdown**

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

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

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

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

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

## Tips for Writing Conditional Formulas

1. When writing a complex or nested conditional statement, it is best to break it into several simple or smaller conditional statements and then combine them.
2. Nesting conditional statements requires multiple pairs of parentheses. Make sure that you have the same number of opening and closing parentheses.
3. In general, one more outcome than the number of conditions is always present (i.e., 1 condition with 2 outcomes, or 2 conditions with 3 outcomes, etc.).
4. 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 SoftMax Pro software's predefined NANs to include text in the results (see [NANs/MakeErrs on page 49](#)).

Likewise, if a conditional statement evaluates to text, you can return numbers as a result as long as the numbers are treated as text (see [Text Functions on page 52](#)).

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

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

For example, both 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 when no data is present is "Pass." The second formula is a better way to write this conditional because "Fail" is the False outcome and is reported as the default result when no data has been collected.

## Concatenation Operators

In SoftMax Pro software, two operators concatenate lists of numbers and arrays of numbers, and a third operator concatenates text strings. These operators allow you to perform complex data analyses using relatively simple formulas.

### Concatenating Text Strings

**Table 2-5** Concatenating Text Strings Descriptions

Operator	Description
+	Combines two text strings. "Soft"+ "Max" = "SoftMax"

### Concatenating Numbers

**Table 2-6** Concatenating Numbers Descriptions

Operator	Description
& (ampersand)	Combines several lists of numbers into a single list of numbers. X&Y&Z
~ (tilde)	Combines several lists of numbers into an array of numbers. X~Y~Z

The difference between ~ and & is illustrated briefly below and is then followed by examples from SoftMax Pro software.

**Table 2-7** Example 1

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

**Table 2-8** Example 2

A & B	A ~ B	
1	1	5
2	2	6
3	3	7
4	4	8
5		
6		
7		
8		

**Example: Using & in a Column Formula**

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

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

Max(!WellLm2 & !WellLm4)

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

**Example: Using ~ in a Group section (1)**

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

Max(AvgOD490 ~ AvgOD590 ~ AvgOD620 ~ AvgOD670)



**Example: Using ~ in a Group section (2)**

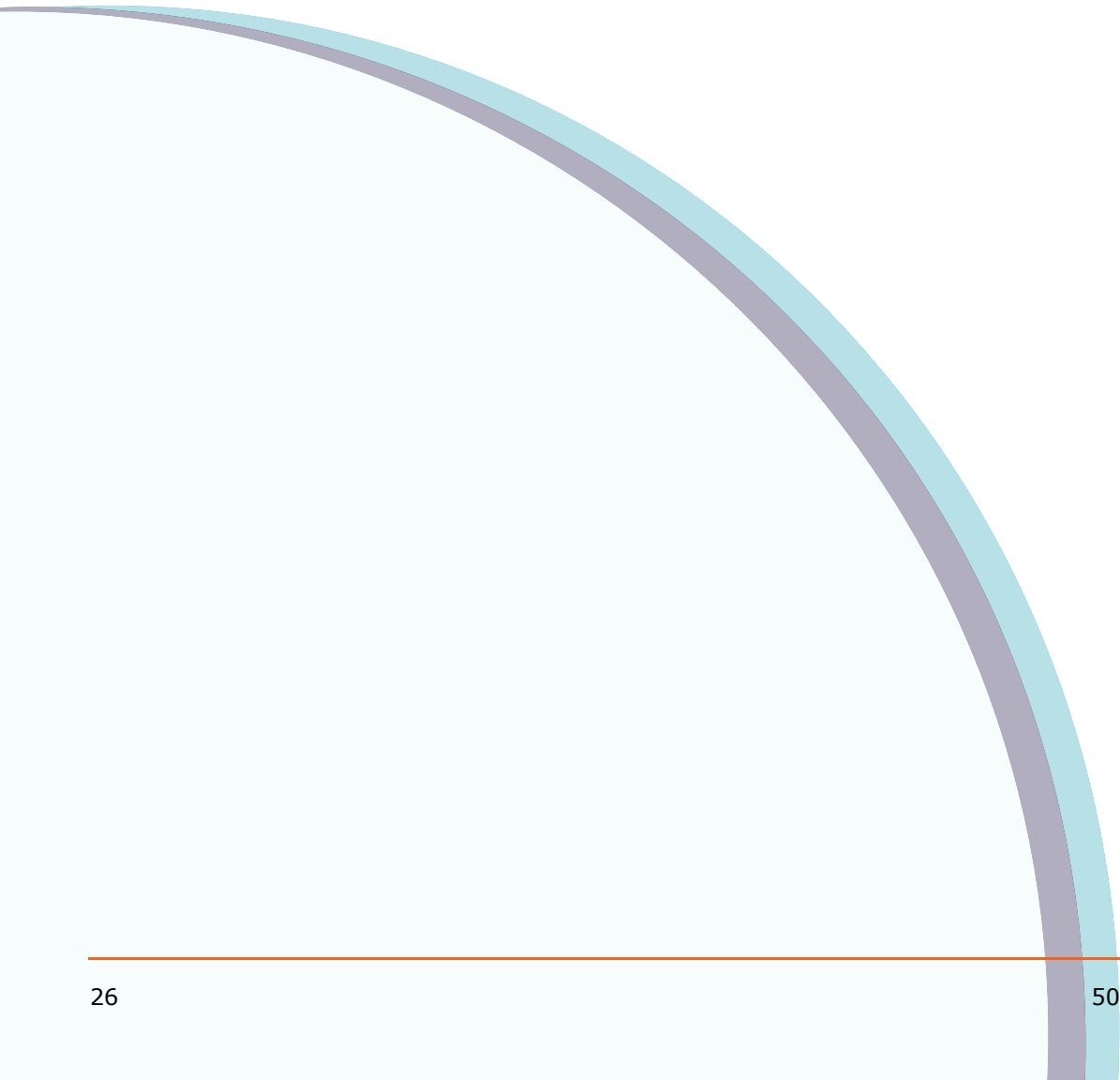
We can use the tilde operator to bring information into a Group section directly from the wells of the plate, bypassing the group structure setup in the template (i.e., the data are entered in well order, not group order). Consider the following formula:

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

The information from the Spectrum scans in these wells is averaged together on a wavelength-by-wavelength basis and is reported in a column (i.e., 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, etc.).

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

Formulas that bring 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.



# Functions

---

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

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

FunctionName(Parameter1, Parameter2, Parameter3)

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

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

SoftMax Pro software functions are divided into the following categories:

- Mathematical functions
- Statistical functions
- Reduction functions
- Interpolation functions
- Other functions
- NANs
- Text functions.

## 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 SoftMax Pro software are all common mathematical, algebraic, and trigonometric functions, and can be used alone or in combination with other functions, accessors, or operators to create more complex formulas.

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

**Table 3-1** Functions

Function	Description
Abs(Parameter)	Returns the absolute value of a number, list of numbers, or array of numbers. Abs(-10) = 10
Acos(Parameter)	Returns the arccosine of a number, list of numbers, or array of numbers. Acos(-0.25) = 1.82348 Acos(!CombinedPlot)
Asin(Parameter)	Returns the arcsine of a number, list of numbers, or array of numbers. Asin(-0.25) = 0.25268 Asin(ColumnX)
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(Summary#1)
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)
Atan(Parameter)	Returns the arctangent of a number, list of numbers, or array of numbers. Atan(0.25) = 0.24498 Atan(!Lm1)
Atan2(Parameter, Parameter)	Returns the arctangent from X- and Y- coordinates. Both 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. Returns a single number, list of numbers, or array of numbers from the two parameters entered. Atan2(1, 1) = 0.78540 Atan2(Values, Concentration)
Ceil(Parameter)	Returns a number, list of numbers, or array of numbers rounded up to the nearest integer. Ceil(1.9) = 2 Ceil(-1.9) = -1 Ceil(Results)

**Table 3-1** Functions

Function	Description
Cos(Parameter)	Returns the cosine of a number, list of numbers, or array of numbers. Cos(0.25) = 0.96891 Cos(Values)
Cosh(Parameter)	Returns the hyperbolic cosine of a number, list of numbers, or array of numbers. Cosh(4) = 27.30823 Cosh(!Wellm1)
Exp(Parameter)	Returns e raised to the power of a number, list of numbers, or array of numbers. Exp(3) = 20.08554 Exp(Summary#3)
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, beginning at 1 and ending at the specified number, i.e., the factorial of 4 is 1 x 2 x 3 x 4). Fact(4) = 24 Fact(Col#1)
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. Floor(1.9) = 1 Floor(-1.9) = -2 Floor(Results)
Fract(Parameter)	Returns the fractional part of a number, list of numbers, or array of numbers. Fract(34.567) = 0.567 Fract(Values)
Int(Parameter)	Returns the integer part of a number, list of numbers, or array of numbers. Int(6.9) = 6 Int(-6.9) = -6 Int(Results)
Ln(Parameter) or Log(Parameter)	Returns the natural logarithm of a number, list of numbers, or array of numbers. Ln(5) = 1.60944 Ln(Values)
Log10(Parameter)	Returns the logarithm base 10 of a number, list of numbers, or array of numbers. Log10(5) = 0.69897 Log10(!Lm1)

**Table 3-1** Functions

Function	Description
Pi	Returns the value of pi (3.14159). No parameters. Summary#1 * Pi
Rand	Returns a random number between 0 and 1. Takes no parameters. Rand
RandNorm	Returns a random number, normally distributed around a mean of approximately 0, and a standard deviation of approximately 1.
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(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; and for zero it returns 0. Sign(25) = 1 Sign(-25) = -1 Sign (0) = 0
Sin(Parameter)	Returns the sine of a number, list of numbers, or array of numbers. Sin(2) = 0.90930 Sin(Column1)
Sinh(Parameter)	Returns the hyperbolic sine of a number, list of numbers, or array of numbers. Sinh(1) = 1.17520 Sinh(Column1)

**Table 3-1** Functions

Function	Description
Sqrt(Parameter)	Returns the positive square root of a number, list of numbers, or array of numbers. Sqrt(64) = 8 Sqrt(Column1)
Tan(Parameter)	Returns the tangent of a number, list of numbers, or array of numbers. Tan(0.25) = 0.25534 Tan(ParmB)
Tanh(Parameter)	Returns the hyperbolic tangent of a number, list of numbers, or array of numbers. Tanh(-2) = -0.96403 Tanh(ParmB)

The functions above can be used singly as shown in the examples above, or they can be used in combination with one another. For example:

Ceil(Abs(Values))

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

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

If you use Round, Int, Ceil, and Floor when SoftMax Pro software is set to display results to a set number of decimal places, and if the number of decimal places specified for SoftMax Pro software is larger than the number specified for the individual function, the extra decimal places are filled with zeroes. For example, if SoftMax Pro software is set to display 3 decimal places and you are rounding a number to two decimal places, the number is displayed as X.YZ0. Similarly, integers are displayed as X.000.

# 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.

**Table 3-2** Statistical Functions

Statistical Function	Description
Average(Parameter)	Returns the average of a list or array of numbers.
AvgDev(Parameter)	Returns the average of the absolute deviation from the mean of a list or array of numbers.
Cv(Parameter)	Returns the coefficient of variation of a list or array of numbers.
Cvp(Parameter)	Returns the coefficient of variation (population) of a list or array of numbers.
FDist(F,df1,df2)	Returns the probability for a given F-statistic F, with df1 numerator degrees of freedom, and df2 denominator degrees of freedom. It is equivalent to the Excel formula with identical syntax, FDist(F,df1,df2).
FInv(p,df1,df2)	Returns the inverse of the preceding formula; the F-statistic, with the numerator degrees of freedom df1 and denominator degrees of freedom df2, for which the probability is p.  <b>Note:</b> p must satisfy $0 < p < 1$ .
Max(Parameter)	Returns the maximum value in a list or array of numbers.
MaxDev(Parameter)	Returns the absolute value of the maximum absolute deviation from the mean of a list or array of numbers.
Median(Parameter)	Returns the middle of list of numbers or array of numbers after the list is sorted.
Min(Parameter)	Returns the minimum value in a list or array of numbers.



**Table 3-2** Statistical Functions

Statistical Function	Description
StDev(Parameter)	Returns the standard deviation of a list or array of numbers.
StDevp(Parameter)	Returns the standard deviation (population) of a list or array of numbers.
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.
Sum(Parameter)	Returns the sum of a list or array of numbers.
TDist(t,df)	<p>Returns the two-tailed probability for a given Student's t-statistic t with df degrees of freedom. It is equivalent to the Excel formula TDIST(t, df, 2).</p> <hr/> <p><b>Note:</b> df must be a positive integer.</p> <hr/>
TInv(p,df)	<p>Returns the inverse of the preceding formula; the t-statistic, with df degrees of freedom, for which the probability is p. It is equivalent to the Excel formula with identical syntax, TINV(p,df). Commonly available tabulations of Student's 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.</p> <hr/> <p><b>Note:</b> p must satisfy <math>0 &lt; p &lt; 1</math>.</p> <hr/>

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

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

For example, if you want the maximum values in a Spectrum scan, you would use the following formula:

Max(!Lm1) in the Plate section reduction or

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 (see [Other Functions on page 44](#)) to find an optical density at a wavelength other than max lambda. In addition, the NthItem function can be used to report the OD at a selected wavelength in the scan (see [Other Functions on page 44](#) for more information on NthItem).

## **Interpolation Functions**

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

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

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

For detailed information on the curve-fit parameters in SoftMax Pro software, refer to the *SoftMax Pro Software User Guide*.

**Table 3-3** Functions

Function	Description
AreaUnder(Xvalues, Yvalues)	<p>Returns the trapezoidal area under a curve. This function is included as one of the default Kinetic/Spectrum reductions in Plate and CuvetteSet sections' Reduction dialog box. It can also be used in a Summary formula in a Group section.</p> <p>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:</p> <p style="text-align: center;">AreaUnder(Xvalues, Yvalues)</p>
AreaUnderFit (PlotName, Xstart, Xstop)	<p>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, then the areas of the trapezoids defined by the 'slices' are summed to determine the area under the curve.</p> <p>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.</p> <p style="text-align: center;">AreaUnderFit(Plotname@Graphname, 1, 1000)</p>
ChiSquared (PlotName@ Graph Name)	<p>Returns the value of the chi-squared statistic from the curve fit.</p> <p>PlotName@GraphName is the full name of the plot, including the name of the graph (for example, Plot#1@Graph#1, or Std@Standardcurve).</p>
ChiProbEx(ChiSquared (PlotName@ GraphSection), DegreesOfFreedom)	<p>Returns the value of the chi-squared probability distribution for a curve fit with the specified degrees of freedom.</p> <p>DegreesOfFreedom is the number of degrees of freedom of the fit: this is the number of points in the curve minus the number of fitted parameters in the fit equation.</p> <p style="text-align: center;">ChiProbEx(ChiSquared(PlotName@GraphSection), 6)</p>

**Table 3-3** Functions

Function	Description
ChiSquaredPLA (PlotName@ GraphSection)	<p>Returns the value of the chi-squared statistic for a reduced curve fit. This can only be used when Parallel Line Analysis is enabled in the Parameter Settings.</p> <p>The designation of a specific plot is somewhat arbitrary since the chi-squared value is calculated from all plots in the designated graph.</p> <p>ChiSquaredPLA(StandardCurve@PLAGraph)</p>
ChiProbabilityPLA (PlotName@ GraphSection)	<p>Returns the chi-squared probability distribution value for a reduced curve fit using the appropriate degrees of freedom.</p> <p>This can only be used when Parallel Line Analysis is enabled in the Parameter Settings.</p> <p>The designation of a specific plot is somewhat arbitrary since the chi-squared probability is calculated from all plots in the designated graph.</p> <p>ChiProbabilityPLA(StandardCurve@PLAGraph)</p>
FStatPLA (PlotName@ GraphSection)	<p>Returns the value of the F-test statistic for a reduced curve fit.</p> <p>This can only be used when Parallel Line Analysis is enabled in the Parameter Settings.</p> <p>The designation of a specific plot is somewhat arbitrary since the F-test statistic is calculated from all plots in the designated graph.</p>
Intercept(Xvalues, Yvalues)	<p>Returns the intercept of a linear regression line that was fit (<math>Y=mX+b</math>) to the X and Y values as identified in the function.</p> <p>Xvalues is the list of numbers to be used for the X values.</p> <p>Yvalues is the list of numbers to be used for the Y values.</p> <p>Intercept(Concentration, MeanValue)</p>
InterpX (PlotName@GraphName, Y values)	<p>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 (e.g., Plot#1@Graph#1, or Std@Standardcurve).</p> <p>InterpX(Std@StandardCurve, YValues)</p>

**Table 3-3** Functions

Function	Description
InterpY (PlotName@GraphName, Xvalues)	Returns the Y values of the specified plot interpolated using the X values and curve fit. InterpY(Plot#6@Graph#3, XValues)
ParmA (PlotName@GraphName)	Returns the A parameter value of the curve fit assigned to the specified plot.
ParmB (PlotName@ GraphName)	Returns the B parameter value of the 5-parameter logistic curve fit assigned to the specified plot.
ParmC (PlotName@GraphName)	Returns the C parameter value of the 5-parameter logistic curve fit assigned to the specified plot.
ParmD (PlotName@GraphName)	Returns the D parameter value of the 5-parameter logistic curve fit assigned to the specified plot.
ParmG (PlotName@GraphName)	Returns the G parameter value of the 5-parameter logistic curve fit assigned to the specified plot.
Rsquared (PlotName@GraphName)	Returns the square of the correlation coefficient of the curve fit assigned to the specified plot.
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.
RelativePotency (Standard PlotName, TestPlotName)	Returns the Relative Potency of a Standard compared to a Test sample. When using a 4-parameter curve fit, the Relative Potency is calculated as the ratio of the C values for the StandardPlotName curve to the TestPlotName curve. With a 5-parameter curve fit, the C values are no longer the mid-point between the max and min. SoftMax Pro software compensates by calculating the IC50 for each plot as: $C \times [2^{(1/G)} - 1]^{(1/B)}$ before taking the ratio.

**Table 3-3** Functions

Function	Description
ParmACILower (PlotName@GraphName)	Returns the lower limit of the confidence interval for the A parameter of a 4- or 5-parameter non-linear curve fit assigned to the specific plot. <sup>1</sup>
ParmBCILower (PlotName@GraphName)	Returns the lower limit of the confidence interval for the B parameter of a 4- or 5-parameter non-linear curve fit assigned to the specific plot. <sup>1</sup>
ParmCCILower (PlotName@GraphName)	Returns the lower limit of the confidence interval for the C parameter of a 4- or 5-parameter non-linear curve fit assigned to the specific plot. <sup>1</sup>
ParmDCILower (PlotName@GraphName)	Returns the lower limit of the confidence interval for the D parameter of a 4- or 5-parameter non-linear curve fit assigned to the specific plot. <sup>1</sup>
ParmGCILower (PlotName@GraphName)	Returns the lower limit of the confidence interval for the G parameter of a 4- or 5-parameter non-linear curve fit assigned to the specific plot. <sup>1</sup>
ParmACIUpper (PlotName@GraphName)	Returns the upper limit of the confidence interval for the A parameter of a 4- or 5-parameter non-linear curve fit assigned to the specific plot. <sup>1</sup>
ParmBCIUpper (PlotName@GraphName)	Returns the upper limit of the confidence interval for the B parameter of a 4- or 5-parameter non-linear curve fit assigned to the specific plot. <sup>1</sup>
ParmCCIUpper (PlotName@GraphName)	Returns the upper limit of the confidence interval for the C parameter of a 4- or 5-parameter non-linear curve fit assigned to the specific plot. <sup>1</sup>
ParmDCIUpper (PlotName@GraphName)	Returns the upper limit of the confidence interval for the D parameter of a 4- or 5-parameter non-linear curve fit assigned to the specific plot. <sup>1</sup>
ParmGCIUpper (PlotName@GraphName)	Returns the upper limit of the confidence interval for the G parameter of a 4- or 5-parameter non-linear curve fit assigned to the specific plot. <sup>1</sup>

**Table 3-3** Functions

Function	Description
RelPotCILowerPLA (PlotName@GraphName)	Returns the lower limit of the confidence interval for the relative potency as determined by the parallel Line Analysis feature. <sup>1</sup> .
RelPotCIUpperPLA (PlotName@GraphName)	Returns the upper limit of the confidence interval for the relative potency as determined by the parallel Line Analysis feature. <sup>1</sup> .
1. Calculations of confidence intervals including specification of the confidence level as a percentage must be set in the Fit Settings of the Graph section.	

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

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

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

**Example: Determining the Slope of the Optical Density versus Time<sup>2</sup> for a Kinetic Plot.**

Slope is one of the predefined choices in the list of Kinetic reductions in Plate/CuvetteSet sections' Reduction dialog. 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 (for example, rate) of the line can be calculated using the custom Kinetic reduction:

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

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

### **Example: Reporting Curve Fit Parameters in Group Section Summaries**

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

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

### **Example: Determining the Area Under a Plot in a Graph Section**

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

## **VMax® Reduction Functions**

You can determine Vmax® Rate or Time to Vmax application by using the functions in the Reduction dialog box. Alternatively, you can use Vmax reduction functions to customize Vmax reductions.

All Vmax reduction functions require three parameters: all Molecular Devices readers except the FlexStation® reader take parameters 1, 2, and 3; FlexStation instruments use parameters 1, 2, and 4.

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

**Second Parameter:** The number of Vmax Points to be used in the calculation. A constant (for example, 45) can be entered for the value of this parameter. If the constant specifies more Vmax Points than are contained in the list or array of numbers specified by the first parameter, then all points in the list/array will be used. Alternatively, the !VmaxPoints accessor can be used for this parameter. In this case, the number of Vmax Points specified in the Plate/CuvetteSet section's Reduction dialog box is used (see Chapter 4, [Accessors](#)).

**Third Parameter:** The read interval to be used in the calculation. A constant representing the number of seconds (for example, 120) can be used for this parameter. Alternatively, the !ReadInterval accessor can be used for this parameter, in which case the read interval specified in the Instrument Settings dialog is used (see Chapter 4, [Accessors](#)).



**Fourth Parameter:** The list of numbers to be used for the calculation. Specifies the time values, for example, !D3Lm1XVals.

**Table 3-4** Functions

Function	Description
TimeToVmax(Parameter1, Parameter2, Parameter3)	Returns the Time to Vmax for the list or array specified by the first parameter, based on the number of Vmax Points and the read interval specified in the second and third parameters, respectively.
TimeToVmaxEx (Parameter1, Parameter2, Parameter4) [FlexStation® reader only]	Returns the Time to Vmax for the list or array specified by the first parameter, based on the list or array of time values specified by the fourth parameter.
Vmax (Parameter1,Parameter2, Parameter3)	Returns the Vmax Rate 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. If you use all of the points in a Kinetic run, Vmaxa application is the same as the slope.
VmaxEx (Parameter1, Parameter2, Parameter4) [FlexStation reader only]	Returns the Vmax Rate for the list or array specified in the first parameter, based on the list or array of time values specified by the fourth parameter. If you use all of the points in a Kinetic run, VmaxEx is the same as the slope.
VmaxCorr (Parameter1, Parameter2, Parameter3)	Returns the correlation coefficient for the Vmax Rate of the list or array specified in Parameter1, based on the number of Vmax Points and read interval specified by the second and third parameters, respectively. It can be used to display this value as the reduced number in a Plate section or as a column in a Group section.

**Table 3-4** Functions

Function	Description
VmaxCorrEx (Parameter1, Parameter2, Parameter4) [FlexStation® reader only]	Returns the correlation coefficient for the Vmax Rate of the list or array specified in Parameter1, based on the list or array of time values specified by the fourth parameter. It can be used to display this value as the reduced number in a Plate section or as a column in a Group section.
VmaxPerSec (Parameter1, Parameter2, Parameter3)	Returns the Vmax Rate (in seconds) 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 (Parameter1, Parameter2, Parameter4) [FlexStation® reader only]	Returns the Vmax Rate (in seconds) for the list or array specified by the first parameter, based on the list or array of time values specified by the fourth parameter. If you use all the points in a Kinetic run, VmaxPerSecEx is the same as the slope.
VmaxPtsUsed (Parameter1, Parameter2, Parameter3)	Returns the number of points that will be used to calculate the Vmax for the list or array specified in the first parameter, based on the number of Vmax Points and the read interval specified in the second and third parameters, respectively.
VmaxPtsUsedEx (Parameter1, Parameter2, Parameter4) [FlexStation reader only]	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 list or array of time values specified by the fourth parameter.

### **Example: Viewing Vmax Rate Determined Using Different Numbers of Vmax Points in Columns of a Group Section**

You can calculate Vmax Rate 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 Vmax Points set to 45, 25, 10, or 5, respectively. !WellCombinedPlot specifies the list of numbers to be used to calculate the VmaxRate (see [Plate Data Accessors on page 64](#) for a complete discussion of the !WellLx accessors), and the !ReadInterval accessor specifies that the read interval defined in the Instrument Settings dialog is used in the calculation.

Using fewer Vmax Points allows the SoftMax Pro software to calculate the linear regression of subsets of the data in each well using creeping iteration, then report the fastest rate (i.e., the steepest slope) in each well. For a complete description of how Vmax Points are used, and how Vmax Rate is calculated, please see the *SoftMax Pro Software User Guide*.

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

To display the Vmax Correlation Coefficient as a plate reduction:

- 1.** In the Display dialog box choose to plot raw data with reduced number.
- 2.** In the Reduction dialog box, select Custom for the kinetic reduction.
- 3.** 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**

If Kinetic data is collected in a Plate section whose Reduction dialog box was used to set the reduction to Vmax Rate, using the accessor !WellValues can bring the reduced number from the Plate section into the Group section.

The Vmax Correlation Coefficient has the custom formula:

VmaxCorr(!WellCombinedPlot, !VmaxPoints, !ReadInterval)

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

The !WellLm1, !VmaxPoints, and !ReadInterval accessors are discussed in detail in Chapter 4, [Accessors](#).

### **Example: Viewing the Number of VMax Points Used to Calculate Each VMax Rate in a Group Section Column**

If a number is used to specify how many VMax Points should be used when calculating the VMax Rate, that number of VMax Points is used if that many data points are visible in the well/cuvette plot in the Plate/CuvetteSet section.

However, if the number of visible data points is less than the specified number of VMax Points, only the number of points that are visible are used (for a complete discussion of how VMax Rate is calculated, see the *SoftMax Pro Software User Guide*.)

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

VmaxPtsUsed(!WellLm1, !VmaxPoints, !ReadInterval)

## **Other Functions**

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

**Table 3-5** Other Functions

Function	Description
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(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(Parameter)	Reports the first array in a list of arrays. The returned values can be numbers, text, or booleans. FirstArray(!WellLm1)
FirstItem(Parameter)	Reports the first item in a list or array of numbers (e.g., the first OD in a Kinetic run, the first OD in a Spectrum scan, the first value in a column of numbers). FirstItem(!CombinedPlot)
Index	Returns an index for the samples in a group, going from 1 to n, for n samples. Index
IndexListFor(Parameter) or ItemIndex(Parameter)	Results in an index for a list or array of numbers (i.e., for the list 123, 125, 130, 127... the index is 1=123, 2=125, 3=130, 4=127...) IndexListFor(Values)
IndexofMax(Parameter)	Reports the index of the maximum of a list or array of numbers. IndexofMax(Results)
IndexofMin(Parameter)	Reports the index of the minimum of a list or array of numbers. IndexofMin(Results)
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 (Parameter1, Parameter2)	Parameter1 is a list of numbers. Parameter2 is a numerical value. The function returns the value of the specified item in the list. Item(Values,2)=25

**Table 3-5** Other Functions

Function	Description
ItemCount(Parameter) or Count(Parameter)	Returns the count (or number of items) of items in a list or array of numbers, text items, or booleans. The function does not count empty or error values in the list. ItemCount(Values)
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(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(Parameter1, N) or Item(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(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/array of numbers to be acted on, Parameter2 is a number indicating at which item in the list/array to begin omitting, and Parameter3 is a number indicating at which item in the list/array to stop omitting. Items between the points indicated by parameters 2 and 3 will be omitted. NullBetween(!WellLm1, 50, 100)
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/array of numbers to be acted on, Parameter2 is a number indicating at which item in the list/array to stop omitting, and Parameter3 is a number indicating at which item in the list/array to resume omitting. Items not between the points indicated by parameters 2 and 3 will be omitted. NullOutside(!WellLm1, 50, 100)
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)

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

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

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

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

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

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

The NullOutside function excludes items in the list below the value indicated by the second parameter and above the value indicated by the third parameter. Thus, in the formula for column WellA1, the first parameter is !A1Lm1 (the optical densities collected at each time point in well A1), the second parameter is 10 (i.e., the 10th time point collected) and the third parameter is 20 (i.e., the 20th time point collected). Column WellA1 therefore includes points 11-19 only, while WellA2 includes points 16-19 only.

The NullBetween function excludes all items in the list between (and including) the second and third parameters. Thus, in the formula for column WellA3, the list of numbers is !A3Lm1 (the optical densities collected at each time point in well A3) and data points 10 through 34 are excluded from the data analysis.

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

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

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

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

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

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

The !StepSweep and !StartSweep accessors are discussed in detail in Section 4.2., "Kinetic and Spectrum Data Accessors".

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

Generally, when calculating the reaction rate of a Kinetic run (the Vmax Rate), you are determining the slope of the line that results from plotting change in OD vs. Time. However, in some situations you may wish to calculate the slope of the line that results when you plot the delta (i.e., the rate of change) in the ODs vs. 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/MakeErrs

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

NANs propagate through most operations and functions in SoftMax Pro.

SoftMax Pro software uses some of the NANs to report errors/other messages in Plate or CuvetteSet sections, Group sections, and Graph sections.

Other NANs are only used by the user when creating custom formulas.

**Table 3-6** NAN (Not A Number) Class

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 or CuvetteSet 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 SoftMax Pro software 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® Pathlength measurement technology is selected and the pre-read has been performed, but the normal read hasn't yet been performed. Also seen if SoftMax Pro software was unable to perform the PathCheck calculation after the normal read.

**Table 3-6** NAN (Not A Number) Class

<b>NAN</b>	<b>Returns</b>
MakeErr(112)	"Open?" Cuvette door was open.
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.

The following list contains the functions that can be used to manipulate NANs:

**Table 3-7** NAN Manipulators

Function	Description
NoNum	Takes no parameters. Returns an "empty" number. It is the same as MakeErr(101).
IsEmpty(Parameter)	Returns true if a number, list of numbers, or array of numbers is empty; otherwise returns false.
IsErr(Parameter)	Returns true if a number, list of numbers, or array of numbers is a NAN; otherwise returns false.
WhatErr(Parameter)	Returns which error the NAN represents (see list in previous table) or 0 if the number, list of numbers, or array of numbers is not a NAN.
MakeErr(Parameter)	Creates a value corresponding to the list of NANs above.

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

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

Using NANs allows you to combine text with numbers in a column. In this example, we create a column that reports either the optical density or "Fail", depending on the value of the optical density. Since SoftMax Pro software treats NANs as if they are numbers, further calculations can be performed on this column, even though it appears to contain a string. The formula is a simple conditional statement:

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

### 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)))

## Text Functions

The following functions allow you to manipulate text strings in SoftMax Pro software.

**Table 3-8** Text Functions

Function	Description
ASCII("Text")	Returns the ASCII value for the first character in the text string. ASCII("Average") = 65
Concat("Text1", "Text2"... "TextN")	Concatenates a series of text strings together. This may also be done using the + operator (see <a href="#">Mathematical Operators on page 13</a> for more on the + operator). Concat("Molecular", "Devices ", "Corporation") = Molecular Devices Corporation
Exact("Text1", "Text2") 	Returns true if the two text strings are identical and false if they are not.  <b>Note:</b> This function is case sensitive. To do a case insensitive Comparison, use the "=" operator, Exact("Softmax Pro", "SoftMax Pro") = False, Exact("SoftMax Pro", "SoftMax Pro") = True.
Left("Text", N)	Returns the first N characters from the left end of a text string. Left("SoftMax Pro", 4) = "Soft"
Right("Text", N)	Returns the first N characters from the right end of text string. Right("SoftMax Pro", 3) = "Pro"
Mid("Text", N1, N2)	Returns the text between the specified start and stop characters. Mid("SoftMax Pro", 5, 3) = "Max"
Len("Text")	Returns the length of (or number of characters in) the text string. Len("Molecular Devices") = 17
Lower("Text")	Returns the text in all lower-case characters. Lower ("Molecular Devices") = "molecular devices"
Upper("Text")	Returns the text in all uppercase characters. Upper ("SoftMax Pro") = "SOFTMAX PRO"
Text (NumericalParameter)	Converts numbers to text—returns a number, list of numbers, or array of numbers as text strings. Text(26.00) = "26"

**Table 3-8** Text Functions

Function	Description
Val("Text")	Returns a number corresponding to the numerical value of the beginning of the text string. If the beginning 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
Now	Takes no parameters. Returns the current time of day.  <b>Note:</b> Every time you open or recalculate the file, Now will be updated to the current time.  Now = 9:05:30 AM
Today	Returns the current day and date.  <b>Note:</b> Every time you open or recalculate the file, Today will be updated to the current day and date.  Today = Friday, April 29, 2005

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

In some situations, you may wish to flag certain results in a column and need to have the flag and the numerical results in the same column. One way to accomplish this is by converting the numbers to text.

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

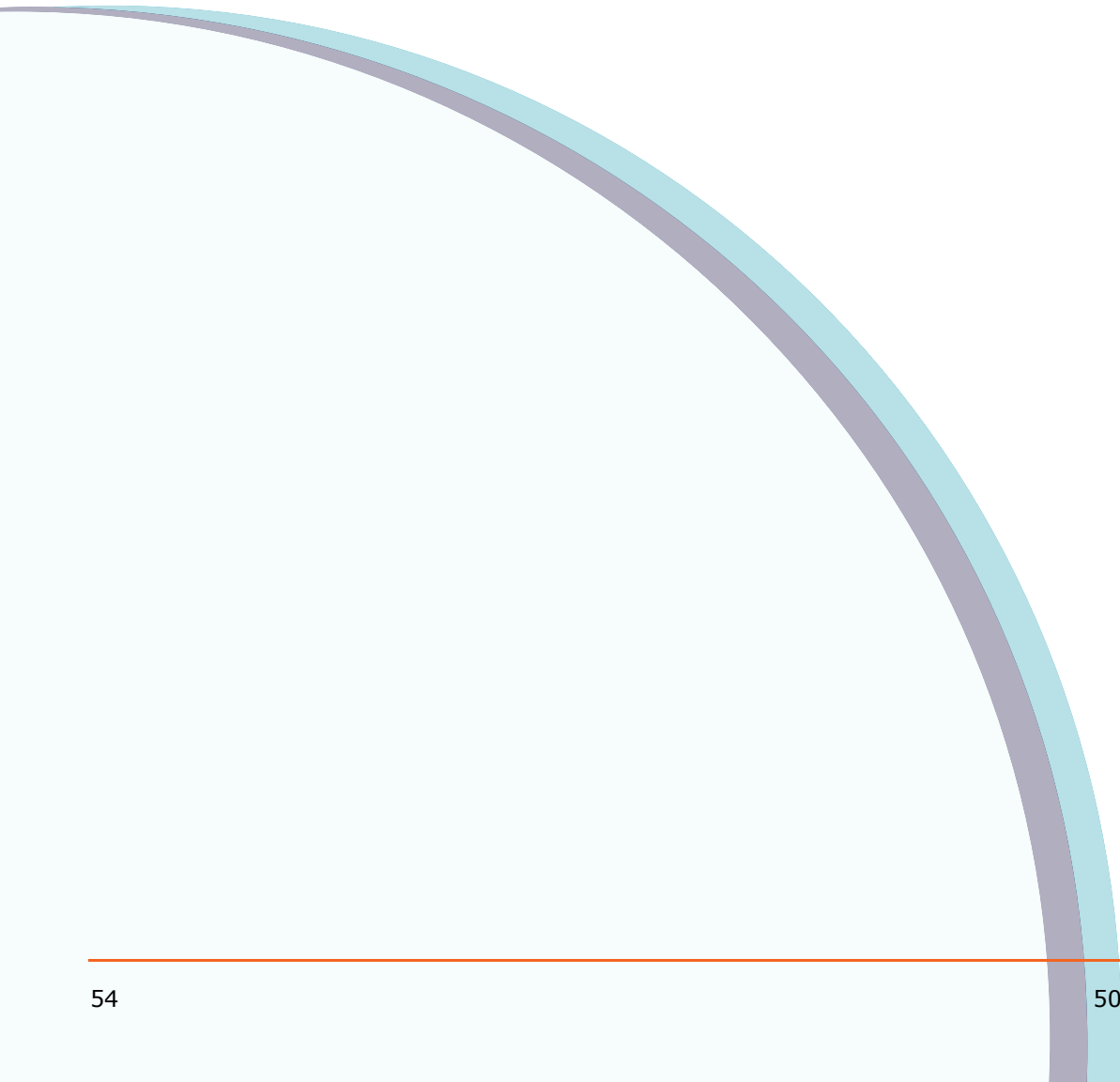
We can construct a new column from the column "Adj.Result" as follows:

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



# Accessors

---

Accessors are special functions that provide access to other specific information in the SoftMax® Pro Software program (for example, the number of VMax Points used to calculate the VMax Rate). Unlike functions, accessors do not have parameters associated with them. Accessors are always preceded by an exclamation point (!, also known as a “bang”).

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

Accessors can be divided into several broad categories:

- Plate/CuvetteSet section information accessors
- Kinetic/Spectrum information accessors
- Plate/CuvetteSet data accessors
- Group information accessors
- Well information accessors
- Blank data accessors.

## Using “Well” as an Accessor Prefix

Accessors can refer to the information in Plate/CuvetteSet sections within SoftMax Pro software in two ways: in read order (well A1, A2, A3, A4...) or in template or group order (wells A1, B1, A2, C2 from Plate #1 and wells G1, H1, G2, H2 on Plate #2 if they are part of the same group). When the word “Well” is used as a prefix between the exclamation point and the accessor, the information is reported in group order:

!Accessor

!WellAccessor

!WellAccessor is used ONLY in Group sections and !Accessor is almost never used in Group section tables.

The “Well” prefix is not valid with all accessors (accessors that can be used with Well are noted in the sections below).

## Plate/CuvetteSet Setup Information Accessors

Plate/CuvetteSet section information accessors provide information about Plate or CuvetteSet section settings. Following the listing of these accessors and their descriptions is a series of examples showing how to use these accessors.

**Table 4-1** Accessor Descriptions

Accessor	Description
!AutoCutoffOn !WellAutoCutoffOn [Fluorescence mode only.]	Returns True if the emission cutoff filter is set to Automatic in the Instrument Settings dialog and reports False if the emission cutoff filter is set to manual.
!AvgReadTemperature !WellAvgReadTemperature	Returns (as a number) the average temperature measured while the plate or cuvette was read.
!EmCutoffs !WellEmCutoffs [Fluorescence mode only.]	Reports the cutoff filter wavelengths set in the Instrument Settings dialog for Endpoint, Kinetic, and Spectrum reads.
!EmFixedWavelength !WellEmFixedWavelength [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 !WellEmWavelengths [Fluorescence mode only.]	Reports the emission wavelength(s) chosen in the Instrument Settings dialog for Endpoint, Kinetic, and Spectrum reads. All (i.e., white light) is the default emission wavelength for luminescence assays and is returned unless a specific emission wavelength is chosen.
!ExFixedOn !WellExFixedOn [Fluorescence mode only.]	Reports True if the instrument setting is set to perform a Spectrum scan with fixed excitation and variable emission wavelengths.
!ExFixedWavelength !WellExFixedWavelength [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 !WellExWavelengths [Fluorescence mode only.]	Reports the excitation wavelengths specified in the Instrument Settings dialog for Endpoint, Kinetic, and Spectrum reads.
!FirstColumn !WellFirstColumn	Returns (as a number) the first column read in a specified Plate section. Useful when reading only selected columns on a plate.
!Gfactor	Returns the current Gfactor value set in the Display dialog box.



**Table 4-1** Accessor Descriptions

Accessor	Description
!IntegrationEnd !WellIntegrationEnd [Time Resolved Fluorescence mode only]	Returns (as a number) the integration end time.
!IntegrationStart !WellIntegrationStart [Time Resolved Fluorescence mode only]	Returns (as a number) the integration start time.
!LastColumn !WellLastColumn	Returns (as a number) the last column read in a specified Plate section. Useful when reading only selected columns on a plate.
!NumWells !WellNumWells [Fluorescence mode only.]	Returns the number of wells in the plate type specified in the Instrument Settings dialog.
!PlateName !WellPlateName	Returns the name of the Plate/CuvetteSet section as a text string. This accessor is very useful when Group sections contain samples from multiple plates in which the samples are in the same wells since it reports which plate the sample was in.
!PlateType !WellPlateType	Returns the text string "Endpoint", "Kinetic", or "Spectrum" depending on the instrument setup.
!PMTSetting !WellPMTSetting [SpectraMax Gemini, M2, M2e, M5 and M5e in Fluorescence mode only.]	Returns the text string "Automatic", "High", "Medium" or "Low" depending on the PMT Setting chosen in the Instrument Settings dialog.
!PreMixDuration !WellPreMixDuration	Returns the number of seconds Automix has been set to mix prior to reading in the Instrument Settings dialog. For Kinetic runs, it reports the Automix duration prior to the first read. It does not report the Automix duration inbetween reads ( <a href="#">Kinetic and Spectrum Data Accessors on page 59</a> ).
!PreMixOn !WellPreMixOn	Returns True if Automix has been turned on in the Instrument Settings dialog box and False if Automix has not been turned on. (For Kinetic runs, this applies to Automix prior to the first read. For Automix between reads, see <a href="#">Kinetic and Spectrum Data Accessors on page 59</a> ).

**Table 4-1** Accessor Descriptions

Accessor	Description
!PreReadOn !WellPreReadOn	Returns True if the Pre-Read Instrument setting has been checked in the Instrument Settings dialog box; returns False if it has not been turned on.
!ReadsPerWell !WellReadsPerWell [Fluorescence mode only.]	Returns (as a number) the reads per well specified in the Instrument Settings dialog box.
!ReadType !WellReadType [Not available on all instruments]	Returns the text string "Absorbance", "Fluorescence", "Luminescence", or "Time Resolved", depending on the instrument setup.
!SectionName	Returns the name of the section containing the formula.
!TemperatureSetPoint !WellTemperatureSetPoint	Returns (as a number) the incubator set point (from the Instrument Settings dialog box) at the time the plate or cuvette was read.
!TimeOfRead !WellTimeOfRead	Returns the time and date the plate or cuvette was read (this information can be found in the time/date stamp displayed in Plate/CuvetteSet sections after a reading).
!TimeResolvedOn !WellTimeResolvedOn [Time Resolved Fluorescence mode only.]	Returns True if the Read Mode in the Instrument Settings dialog box is set to Time Resolved. Reports False if the Read Mode is not set to Time Resolved.
!Wavelengths !WellWavelengths	Returns as a text string (e.g., "405 650") the wavelength(s) at which a Plate/CuvetteSet section was read.

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

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

!TimeOfRead@PlateX

!TimeOfRead@PlateY.

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

## Kinetic and Spectrum Data Accessors

The accessors in the following table are specific to Kinetic and/or Spectrum scan data. They allow you to:

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

**Table 4-2** Kinetic and Spectrum Data Accessor Descriptions

Accessor	Description
!NumPoints !WellNumPoints	Returns the number of data points (number of time points to be collected in a Kinetic run or number of wavelengths to be read in a Spectrum scan) specified in the Instrument Settings dialog Box.
!NumPointsRead !WellNumPointsRead [SpectraMax® Plus, SpectraMax 190, SpectraMax 340 PC, VersaMax, 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 Instrument Settings.  This accessor returns both numbers and NaNs and, thus, it may 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.
!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 (selected in the Instrument Settings dialog box) is read.  The information reported by this accessor is invalid for Endpoint readings if PathCheck has been selected in the Instrument Settings dialog box.
!ReadInterval !WellReadInterval	Returns the number of seconds specified in the Instrument Settings dialog box for a Kinetic run read interval.

**Table 4-2** Kinetic and Spectrum Data Accessor Descriptions

Accessor	Description
!RunTime !WellRunTime	Returns the time (in seconds) that it will take to read a Kinetic plate, as calculated from the information in the Instrument Settings dialog box. If the run is terminated before it can be completed, this accessor will still report the calculated time to finish the run, not the actual run time (however, !WellNumPoints could be used to calculate the actual number of data points collected).
!StartSweep !WellStartSweep	Returns (as a number) the wavelength at which to begin a Spectrum scan as specified in the Instrument Settings dialog box.
!EndSweep !WellEndSweep	Returns (as a number) the wavelength at which to end a Spectrum scan as specified in specified in the Instrument Settings dialog box.
!StepSweep !WellStepSweep	Returns (in nm) the step size of a Spectrum scan as specified in the Instrument Settings dialog box.
!AutomixOn !WellAutomixOn	Returns (as a text string) True if Automix between reads is checked for Kinetic plates in the Instrument Settings dialog box and False if it is not checked.
!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.

**Table 4-2** Kinetic and Spectrum Data Accessor Descriptions

Accessor	Description
!TimeRun !WellTimeRun	<p>!TimeRun returns a list of read times in a vertical column.</p> <p>!WellTimeRun returns a list of read times in a horizontal array. Valid for use with Kinetic data only, this accessor returns a list of numbers corresponding to the times at which Kinetic data points are collected.</p> <p>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 (e.g., 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 will return 9 and then 12 for the read times). All of the cuvettes in a CuvetteSet 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.</p>
!WavelengthRun !WellWavelengthRun	<p>Returns (as a list of numbers) the wavelengths at which each of the Spectrum scan data points are collected.</p> <p>This accessor is valid for Spectrum scans only.</p>
!MinLimit !WellMinLimit	<p>Returns (as a number) the Min OD set in the Reduction dialog for a Spectrum scan or Kinetic run. Any data collected below the Min OD will be reported with the NAN MakeErr(113) "Limits-" and will not be used in any reduction.</p>
!MaxLimit !WellMaxLimit	<p>Returns (as a number) the Max OD set in the Reduction dialog for a Spectrum scan or Kinetic run. Any data collected above the Max OD will be reported with the NAN MakeErr(114) "Limits+" and will not be used in any reduction.</p>
!StartLimit !WellStartLimit	<p>Returns (in seconds) the lag time for Kinetic run data analysis and the start wavelength for Spectrum scans (both set in the Reduction dialog box). Any data collected before the lag time or below the starting wavelength is reported with the NAN MakeErr(113) "Limits-" and is not used in any reduction.</p>

**Table 4-2** Kinetic and Spectrum Data Accessor Descriptions

Accessor	Description
!EndLimit !WellEndLimit	Returns the end time set for Kinetic run data analysis and the end wavelength set for Spectrum scan data analysis (both set in the Reduction dialog box). Any data collected after the end time or above the end wavelength is reported with the NAN MakeErr(114) "Limits+" and is not used in any reduction.
!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.
!OnsetValue !WellOnsetValue [Previously called !OnsetOD, !WellOnsetOD]	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.
!XVals	Returns the values associated with the X-axis of a Kinetic or Spectrum read. This is the same as !TimeRun for Kinetic data or !WavelengthRun for Spectrum data.

**Example: Accessing Kinetic Run Settings Information**

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

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

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

We 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 (i.e., 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 CuvetteSet section is present in the Experiment section you need to identify the Plate/CuvetteSet section in the formulas; for example, !StartSweep@Plate#1 or !EndSweep@CuvetteSet#1.

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

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

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

## Plate Data Accessors

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

**Table 4-3** Plate Data Accessor Descriptions

Accessor	Description
!LmX !A1LmX !WellLmX	Reports (as a number, in Endpoint readings, or list of numbers, in Kinetic/Spectrum scans) the raw OD values (before the wavelength combination reduction is applied) from wells or cuvettes. LmX specifies the wavelength (i.e., Lm1, Lm2, etc.). !LmX accesses the raw values from all wells or cuvettes in the section at the given wavelength. !A1LmX accesses the raw value from the specified well or cuvette at the specified wavelength (for example, !H2Lm1, or !B12Lm3). !WellLmX accesses the raw optical densities reported in a Group section in template order. Optical densities at each Kinetic run time point or each Spectrum scan wavelength are reported in a horizontal array.
!WellLmXSRaw (For Fluorescence Polarization readings only)	Reports the raw values (before any other reduction is applied) from wells or cuvettes for perpendicular polarized data of a specific wavelength (e.g., Lm1 or 1st wavelength set) for a given template group (in a Group section). !WellLmXSRaw accesses the raw values reported in a Group section in template order.
!WellLmXPRaw (For Fluorescence Polarization readings only)	Reports the raw values (before any other reduction is applied) from wells or cuvettes for parallel polarized data of a specific wavelength (e.g., Lm2 or 2nd wavelength set) for a given template group (in a Group section). !WellLmXPRaw accesses the raw values reported in a Group section in template order.
!WellIDLmXSRaw (For Fluorescence Polarization readings only)	Returns perpendicular (or an array of perpendicular) data for the given well (e.g., A2) and specified wavelength (e.g., Lm1 or 1st wavelength set).
!WellIDLmXPRaw (For Fluorescence Polarization readings only)	Returns parallel (or an array of parallel) data for the given well (e.g., A2) and specified wavelength (e.g., Lm1 or 1st wavelength set).



**Table 4-3** Plate Data Accessor Descriptions

Accessor	Description
!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 (i.e., Lm1, Lm2, etc.). In Plate sections, !A1LmXXVals returns the read time (timetagged) 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 will be reported in a horizontal array. This accessor is valid only for FlexStation.
!AllWavelengths !WellAllWavelengths	Returns (as a list of numbers) the raw optical densities for each wavelength read in a given well. This accessor can only be used with multiple-wavelength Endpoint reads.
!Values !WellValues	Returns the reduced number from each well of a Plate section or cuvette in a CuvetteSet section. !WellValues is the default formula given to the default Group sections column named Values.
!MValue !Pvalue (Not available on all instruments)	Returns the reduced number from each well of a Plate section or cuvette in a CuvetteSet section. !MValue is returned as the default for readings taken after an Minjection; !PValue is returned as the default for readings taken after an P-injection.
!AllValues	Returns a list of the reduced numbers from all wells in a Plate section or all cuvettes in a CuvetteSet section.
!PreReadLmX !A1LmXPreRead !WellPreReadLmX	Reports (as a number, in Endpoint readings, or list of numbers, in Kinetic/Spectrum scans) the pre-read values from wells or cuvettes. LmX specifies the wavelength (i.e., Lm1, Lm2, etc.). In Plate sections !PreReadLmX returns the pre-read value(s) for each well at the specified wavelength. !A1LmXPreRead returns the pre-read value for the specified well at the indicated wavelength, and !WellPreReadLmX returns the pre-read values for the specified wavelength in template order. In CuvetteSet sections, the !PreRead accessor can be used to display the REF value.

**Table 4-3** Plate Data Accessor Descriptions

Accessor	Description
!CombinedPlot !WellCombinedPlot	This accessor can be used in Plate or CuvetteSet sections, for Kinetic runs and Spectrum scans. Returns the result of the wavelength combination reduction formula for each well or cuvette (!LmX or !WellLmX returns information before the wavelength combination reduction is applied).
!StdDevPlateBlank	Returns the standard deviation value of the plate blank group.
!StdDevPlateBlankLmX	Specifies the wavelength LmX (e.g., Lm1, Lm2) of the plate blank group.
!PlateBlankS !PlateBlankLmXS	Returns perpendicular (or S) plate blank raw value(s) for the given wavelength LmX in a Plate section (e.g., Lm2 or the 2nd wavelength set).
!PlateBlankP !PlateBlankLmXP	Returns parallel (or P) plate blank raw value(s) for the given wavelength LmX in a Plate section (e.g., Lm1 or the 1st wavelength set).

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

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

!Lm1!/A1Lm1

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

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

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

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

!WellLm1

!WellLm2

!WellLm3

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

$(\text{OD260}-\text{OD320})/(\text{OD280}-\text{OD320})$

Last, we can calculate the average pathlength, in centimeters, for each sample:

Average(!WellPathlength)

### **Example: Viewing the REF Values for a CuvetteSet**

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

!WellPreReadLm1

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

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

!WellPreReadLm2

!WellPreReadLm3

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

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

In addition, the display of optical densities from a Kinetic run in the Group section is affected by whether or not the Absolute Values checkbox in the Reduction dialog is enabled. When the box is not checked (disabled by default), optical densities are displayed normalized to 0: the first value for each well is 0.000

When the box is checked (enabled), absolute optical densities are displayed: the first OD for each well is no longer zero.

If you want to graph the optical densities from a Kinetic run or Spectrum scan, you need to bring them into the Group section in a columnar format (independent of the template) instead of a horizontal array (as shown in the example above).

## PathCheck® Accessors

PathCheck® 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 may be used in Plate sections or in column formulas in Group sections or in Summary formulas. These accessors are not used in CuvetteSet sections.

**Table 4-4** Pathcheck® Accessor Descriptions

Accessor	Description
!Pathlength !A1Pathlength !WellPathlength	Returns the pathlength determined by PathCheck® accessors. These accessors can only be applied 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.
!PathCheckLm900	Returns a number representing either the cuvette reference value read at 900 nm or the value for the Water Constant at 900 nm, depending on whether the PathCheck instrument settings are set to use a cuvette reference or the Water Constant.
!PathCheckLm1000	Returns a number representing either the cuvette reference value read at 1000 nm or the value for the Water Constant at 1000 nm, depending on whether the PathCheck instrument settings are set to use a cuvette reference or the Water Constant.
!Lm900 !WellLm900	Returns (as a list of numbers) the PathCheck values read from the plate at 900 nm.
!Lm1000 !WellLm1000	Returns (as a list of numbers) the PathCheck values read from the plate at 1000 nm.
!PathCorrectionOn	Returns True if the Plate section has been set to use pathlength correction and False if it has not.
!PathCheckOn	Returns True if PathCheck has been selected in the Instrument Settings dialog box and returns False if it has not.
!PathBackground!PathBackgroundLm1,... !PathBackgroundLm6	Returns the plate background constant that has been set for the specified wavelength(s).
!PathCheckApplied	Returns True if the PathCheck Reduction setting has been applied to a Plate section.

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

If Pathlength correction is enabled, the !WellValues accessor brings 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.

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

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

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

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

## Group and Well Information Accessors

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

**Table 4-5** Group and Well Information Accessor Descriptions

Accessor	Description
!SampleDescriptor !Concentration !Factor	Returns the numerical value assigned by the user in the Sample Descriptor field in the Template Editor, usually concentration for standards or a dilution factor for unknowns. The sample descriptor value can be used with any numerical value you want to associate with a sample (for example, if it is one of a series of samples drawn over time, time could be used as a sample descriptor).
!SampleNames	Returns (as a text string) the sample name assigned in the Template Editor.
!SampleNumber	Returns the number of the sample.

**Table 4-5** Group and Well Information Accessor Descriptions

Accessor	Description
!Units	Returns (as a text string) the units assigned to the sample descriptor in the New Group or Edit Group dialog box in the Template Editor.
!WellIDs	Returns the well identifier (A1 to H12) for each sample replicate.
!WellNumbers	Returns the well number of each sample replicate in a group to the Group section as a list of numbers (Well A1 = 1...H12 = 96).

## Blank Accessors

Blank accessors allow you to access and manipulate plate blank and group blank information. The average of the plate blank is displayed in the Plate section for Endpoint reads. The average of the group blank is displayed at the top of each Group section.

**Table 4-6** Blank Accessor Descriptions

Accessor	Description
!GroupBlank@GroupName	Returns (as a number) the average of all wells that have been designated as group blanks for the specified group.
!GroupBlankValues@GroupName	Returns (as a list of numbers) the values of the individual wells that have been designated as group blanks for the specified group.
!PlateBlank	Returns the average plate blank value (as a number).
!StdDevPlateBlank	Returns the standard deviation of the plate blank group.

**Table 4-6** Blank Accessor Descriptions

Accessor	Description
!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.
!PlateBlanks !PlateBlankSLmXS	Returns perpendicular (or S) plate blank raw values for the given wavelength LmX in a Plate section. !PlateBlankLm1S
!PlateBlankP !PlateBlankLmXP	Returns parallel (or P) plate blank raw values for the given wavelength LmX in a Plate section. !PlateBlankLm1P

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, etc., use the specific well accessor (e.g., !A1LmX) in conjunction with a concatenation operator (& or ~) and the appropriate statistical function (Stdev, Min, Max, etc.).

### Example: Accessing Group Blank Information

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

Average group blank:

!GroupBlank

Standard deviation of the group blank:

Stdev(!GroupBlankValues)

Maximum of group blank values:

Max(!GroupBlankValues)

Minimum of group blank values:

Min(!GroupBlankValues)

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

If you want to subtract the same plate blank from multiple groups on different plates, you can perform the subtraction in the Plate sections. In this example, the template from first Plate section includes the plate blank. In the second Plate section (containing samples that continue the group from the first Plate section), the reduction formula is set to be the optical density minus the plate blank from Plate #1. The custom formula for this reduction is:

`!Lm1-!PlateBlank@Plate#1`

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

To subtract a plate blank from a single group that has samples on multiple plates, you must perform 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 box, disable the Use Plate Blank check box so that the plate blank is not subtracted automatically from the group. It is very important to do this or the blank will be subtracted twice from the samples on this plate—once in the Plate section and again in the Group section.
3. Define the samples belonging to the group in the subsequent Plate sections.
4. Create two custom columns called “Values” and “BlankSbtVal” using the following formulas respectively:
  - ♦ `!WellValues`
  - ♦ `Values-!PlateBlank@Plate#1`

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



# Index

---

---

## Symbols

16  
- 14  
^ 14  
\* 13  
/ 13  
& (ampersand) 23  
+ 13, 23  
= 15  
> 15  
>= 15  
~ (tilde) 23

---

## A

Abs 28  
Accessors 8  
Acos 28  
And 17  
AntiLog 28  
antilog 28  
AntiLog10 28  
arccosine 28  
arcsine 28  
arctangent 28  
AreaUnder 34  
AreaUnderFit 34, 39  
array of numbers 31  
arrays  
    calculations on 9  
ASCII 52  
Asin 28  
Atan 28

Atan2 28  
Average 32  
AvgDev 32

---

## B

blank group information 71  
boolean operators 16

---

## C

Ceil 28, 31  
ChiProbabilityPLA 35  
ChiProbEx 35  
ChiSquared 34  
ChiSquaredPLA 35  
comparison operators  
    example 16  
Concat 52  
concatenating  
    text strings 23  
concatenation operators 23  
conditional operators 16  
Cos 29  
Cosh 29  
cosine 29  
Count 45  
Cv 32  
Cvp 32

---

## D

data  
    raw 5  
DegreesOfFreedom 35  
Delta 45

---

## E

Exact 52  
examples  
    calculating the first derivative 48  
    column formulas 24  
    combination formula 21  
    combining text with numbers 53  
    comparison operator 16  
    determining the slope 39  
    interpolation functions 39  
    interpolation functions 39  
    mathematical operators 15  
    reporting blank group informaton  
        71  
    statistical functions 33  
    subtracting the same plate blank  
        72  
    using a tilde 24  
    using IndexOfMax 48  
    using NaNs 51  
    using NullBetween 47  
    usingInterpX 38  
    viewing VMax rate 43  
    VMax 43, 44  
Exp 29

---

## F

Fact 29  
factorial 29

False 17  
FDist 32  
FInv 32  
FirstArray 45  
FirstItem 45  
Floor 29, 31  
formula building blocks  
    Accessors 8  
    Functions 8  
    NaNs 8  
    Operators 8  
formulas  
    conditional 18  
    conditional formulas tips 22  
    netsted conditionals 18  
    reduction 5  
    summary 7  
Fract 29  
FStatPLA 35  
functions 8, 40  
    interpolation 33  
    mathematical 27  
    other 44  
    statistical 31  
    text 52  
    VMax reduction 40

---

## H

hierarchical naming structure 9  
hyperbolic 29  
hyperbolic sine 30  
hyperbolic tangent 31

---

## I

If 17  
Index 45  
IndexListFor 45

IndexOfMax 48  
IndexofMax 45  
IndexofMin 45  
IndexofNearest 45  
Int 29, 31  
Intercept 35  
interpolation  
    functions 33  
InterpX 34, 36  
InterpY 34, 36  
IsEmpty 51  
IsErr 51  
Item 45  
ItemCount 46

---

## L

Left 52  
Len 52  
list of numbers 31  
Ln 29  
Log 29  
Log10 29  
Lower 52

---

## M

MakeErr 49, 51  
MakeErrs 49  
manipulators 51  
mathematical  
    functions 27  
    operators 14  
mathematical operators 13  
    example 14  
    using 14  
Max 32  
MaxDev 32

Median 32  
Mid 52  
Min 32  
Mod 14

---

## N

NANs 49, 51  
NearestTo 46  
NoNum 51  
Not 17  
Not A Number (NAN) 8  
Now 53  
NthArray 46  
NthItem 46  
NullBetween 47  
NullOutside 46, 47, 48  
numbers  
    calculations on 9

---

## O

operators 8, 15  
    boolean 16  
    comparison 15  
    concatneation 23  
    conditional 16  
    logical 15  
    mathematical 13  
Or 17

---

## P

ParmA 36  
ParmACILower 37  
ParmACIUpper 37

ParmB 36  
ParmBCILower 37  
ParmBCIUpper 37  
ParmC 36  
ParmCCILower 37  
ParmCCIUpper 37  
ParmD 36  
ParmDCILower 37  
ParmDCIUpper 38  
ParmG 36  
ParmGCILower 37  
ParmGCIUpper 38  
Pi 30  
PlotName 34  
positive square root 31

---

## R

Rand 30  
RandNorm 30  
ReadInterval 40, 43, 44, 48  
RelativePotency 37  
Right 52  
Round 30, 31  
Rsquared 36

---

## S

Sideways 46  
Sign 30  
Sin 30  
Sinh 30  
Slope 34, 36  
special characters 11  
Sqrt 31  
StartSweep 48

statistical  
    functions 31  
StdErr 32  
StDev 32  
StDevp 32  
StepSweep 48  
Sum 33

---

## T

Tan 31  
tangent of a number 31  
Tanh 31  
TDist 33  
Text 52  
text strings  
    concatenating 23  
TimeToVMax 41  
TimeToVMaxEx 41  
TInv 33  
Today 53  
True 17

---

## U

Upper 52

---

## V

Val 53  
VMax 41  
    determining rate 40  
    reduction functions 40  
VMax reduction 40  
VMaxCorr 41  
VMaxCorrEx 41

VMaxEx 41  
VMaxPerSec 42  
VMaxPerSecEx 42  
VMaxPoints 40, 43, 44, 48  
VMaxPtsUsed 42, 44  
VMaxPtsUsedEx 42  
VMaxRate 43

---

## W

Wavelength Combination 5  
WellCombinedPlot 44  
WellLx 43  
WellValues 44  
WhatErr 51

---

## X

Xstart 34  
Xstop 34

