Release Notes

TestBase
Version 2.5.0
April 30, 2003

1. <u>Overview</u>

This release includes the following main items:

- TestBase installation package, including a complete set of sample files, documentation and tutorial slides
- Redistributables, including:
    - TYX License Manager
    - Third-party run-time engines
    - Instrument drivers needed for running some of the samples
- Installation Guide

2. <u>Detailed Description</u>

2.1. Critical items

2.1.1. <u>Environment variable setting for LabVIEW Adapter</u>

The LabVIEW Adapter loads the LabVIEW engine when it is loaded by TestBase, and unloads the engine when it is unloaded. It appears that the repeated unloading and reloading of the LabVIEW engine causes unreliable operation. Thus, for a reliable operation of the LabVIEW test procedures it is recommended to set the value of the environment variable **TB_TCA_UNLOAD** to 0 (zero). See the Help System for details.

2.1.2. <u>IDE execution error caused by an Internet Explorer security upgrade</u>

After installing one of the new security updates for Internet Explorer (also included in Internet Explorer Service Pack 1), you might encounter the following behavior: a “File download” dialog will be displayed upon trying to open TestBase, asking if the users wants to Open or Save a file with the .vsd extension.

In order to prevent this behavior, you will have to manually re-associate files having “vsd” extension with file type “Microsoft Visio Drawing” by executing the following operations under Windows 2000 and Windows XP:

- In Windows Explorer, select menu **Tools | Folder Options…**, than click the **File Types** tab.
- Select the “VSD” extension from the list of “Registered File Types”. Click **New**. In the dialog that appears, type “vsd” in the “File Extension” input field and click **Advanced**.
- A Drop-down list will be displayed (this may take some time). Select “Microsoft Visio Drawing” from this list and press **OK**. Press **Yes** on the message warning that the extension is already associated with the file type “Microsoft Visio Drawing”.

- Press **Close** on the initial **Folder Options** dialog.

At this point we are not able to provide a similar procedure for Windows NT 4.0. We suggest using Internet Explorer versions earlier than 6.0 SP1, or upgrading to a newer Windows version.

2.1.3. Error when generating Diagnostic Procedure HTML Reports

When generating an HTML report for a diagnostic strategy, you may get the error message “Unable to export CFD”. The report files containing the Visio diagram are not generated. The report file containing the text description of the diagnostic strategy is generated, but its hyperlinks will not operate.

This problem occurs if Visio was installed on the system but its HTML export feature was never used. To fix the problem, open an existing flowchart in Visio or create a new one (menu **File | New**), then export it as HTML (menu **File | Save As**, select file type “HTML Files”, select a temporary folder and press **OK**).

2.1.4. Operating system compatibility

The product works properly with Windows NT 4.0 SP 6, Windows 2000 SP 2 and Windows XP.

2.1.5. Internet Explorer compatibility

The product works properly with Microsoft Internet Explorer 5.5 and 6.0. **It does not work with Internet Explorer 5.0**.

2.1.6. Microsoft Office compatibility

The display of Excel reports works properly with Microsoft Office 2000 Service Release 1 and Microsoft Office XP.

2.1.7. Microsoft Visio compatibility

The product works properly with Microsoft Visio 2000. **The current version of the product is not compatible with Visio 2002.**

2.1.8. Oracle compatibility

The MTI Controller distributed with the current version of the product was tested with the following combinations of versions :

- Oracle 8.1.5 and 8.1.7, under Windows NT 4.0
- Oracle 8.1.5 and 8.1.7, under Windows 2000
- Oracle 9.0.1.1, under Windows NT 4.0.

A known compatibility problem between the Oracle software and the Microsoft libraries used internally by TestBase may be fixed, for Oracle 8 versions, by configuring the system registry as described in the document Connectivity Issue with MDAC and Oracle8i.pdf. For Oracle 9, use the information provided for version 8.1 in the above document, performing the following replacements in the strings to be entered in the registry:

1. replace oraclient8.dll with oraclient9.dll
2. replace orasql8.dll with orasql9.dll

2.1.9. LabWindows/CVI compatibility

The Adapter for LabWindows/CVI works properly with the following versions: 5.0, 5.5 and 6.0. Minor changes to the sample test code must be performed in order to compile it under version 5.0. The TestBase installation package redistributes version 6.0 of the LabWindows/CVI Run Time Engine. You may obtain other versions of the Run Time Engine from National Instruments.

2.1.10. Case sensitivity of Outcome comparisons

When a Test block is followed by decision blocks, the compiler compares the outcome values tested by the decision blocks against the set of possible outcomes returned by the called Test procedure. In previous releases, the above comparison was case insensitive. For consistency with other comparisons in TestBase, the comparison algorithm was changed to case sensitive, starting with release 2.4.0. This change does not affect the operation of existing projects and does not require their recompilation.

2.1.11. Rebuilding samples of Custom Data Type Editors and samples using Custom Data Type Editors

When rebuilding the Custom Data Type (CDT) Editor samples (or a modified version of these samples), overwrite the existing .OCX file (also included in the distribution). Do not delete the existing file before generating the new one. Overwriting the original file preserves the version used by Visual Basic to record binary compatibility. If you generate a new .OCX file (without overwriting), you will need to re-build all test procedures using that CDT Editor.

To rebuild the LabWindows/CVI test procedure samples that use Custom Data Type (CDT) Editors (or a modified version of these samples), you must first re-generate the wrapper files for the CDT Editor components. When re-generating, overwrite the original files.

2.2. Known Limitations

2.2.1. TestBase IDE

The “Undo” command used during flowchart editing does not operate properly in some situations. It is recommended to avoid using it.

The “Undo” command for the addition of an Off-Page Reference block does not delete the “pair” block that was automatically appended. Workaround: delete manually the “pair” block.

When running a test strategy in debug mode, with execution stopped between steps, if the user clicks on a different Control Flow Diagram (flowchart) then clicks the **Run**, **Step**, **SimulatedStep** or **SimulatedStepWithUI** buttons, an error message may be displayed: “Automation Error. Illegal to call out while inside message filter”. This is normal. Press **OK**, click on the Control Flow Diagram that is currently debugged and continue using the IDE. The error indicated before has no impact on subsequent operation.

In some situations accelerator keys (Ctrl-O for **File | Open**, F5 for **Debug | Go**, etc.) do not work. Use menus or toolbar buttons instead.

A General Protection Fault occurs when closing the IDE or a Diagnostic Controller user interface after an Abort operation. Because it occurs only when the application is terminated, this behavior does not have harmful effects.

2.2.2. LabWindows/CVI Adapter

A limited subset of scalar data types is currently supported in the TestBase Support Library for LabWindows/CVI. Use direct access to VARIANT fields for other data types.

The execution of samples that use Custom Data Type Editors as ActiveX controls requires a LabWindows/CVI installation (the run-time engine is not sufficient).

2.2.3. PAWS Adapter

Array parameters are not supported.

Debugging of ATLAS TPSs launched from TestBase is not supported. Workaround: debug TPSs in standalone execution mode.

2.2.4. LabVIEW Adapter

The documentation for developing test procedures with LabVIEW does not cover the use of Custom Data Type Editors as ActiveX controls on LabVIEW panels.

2.2.5. VEE Adapter

The documentation for developing test procedures with LabVIEW does not cover the use of Custom Data Type Editors.

2.2.6. Permissions for installation

The user performing the installation must have Administrator permissions for the Windows operating system. **After installing TestBase, this user must start the IDE once, to perform a Registry initialization.** After that, users with more restricted permissions can use TestBase modules. The user performing security administration from the IDE must have also Administrator permissions for the Windows operating system.

2.2.7. MTI error on Global Parameters Values with the same name

If Global Parameter Values with identical names are defined at multiple level of the Project tree and the storage of test results in MTI databases is enabled, an error will be generated at run-time. Workaround: avoid using identical names, until this problem is fixed.

2.2.8. Custom Data Types

The use of CDT Editors is currently not supported in test procedures developed with ATLAS and Agilent VEE.

The generic reports included with TestBase display the values of CDTs as follows:

- The Diagnostic Procedure report displays the generic text "<Value for Custom Data Type Editor with ProgID: ", followed by the ProgID of the CDT Editor.
- The sample Excel report "Sample_MTI_offline.xlt" displays the generic text "<Value for Custom Data Type Editor with ProgID: ", followed by the ProgID of the CDT Editor.

The values of Global Parameters with Custom Data Types are not stored in Oracle and Access MTI databases. You may retrieve these values from the Input Parameters to whom the Global Parameters are assigned.

2.2.9. Diagnostic Procedure HTML Reports

The internal links will not operate if the set of files generated for a report is moved to a different location in the file system or on a web server.

2.3 Enhancements

This section describes the features introduced since version 2.4.1.

2.3.1. Custom Data Types

This release introduces a new TestBase feature - support for Custom Data Types (CDTs). This feature enables users to develop and integrate Custom Data Type Editors, “plug-in” modules that support user input, display and programmatic access to custom data types of arbitrary complexity. Such data types may be application-specific (ex. UUT settings) or domain-specific (ex. analog signals, RF parameters, etc.).

CDTs are supported in the following TestBase modules:

- IDE: CDT Editors are displayed by the IDE when the user edits the value of a parameter having a custom data type, and when the user checks the value of such a parameter during debug.
- Production User Interface: The values of global parameters with Custom Data Types are displayed during execution.
- Test Procedures and Reports: CDTEs may be used in Test Procedures and Reports developed by users, providing programmatic access to data and, optionally, a graphical representation of data on test procedure panels or report sheets.
- MTI Database: Parameters with Custom Data Types are stored in relational databases and XML files.
- DiagML export and import. Parameters with Custom Data Types may be exported and re-imported.
- Documentation: The usage of CDT-related features is described in the Help System. The development of CDT Editors and their usage in test procedures and reports is described in the Developer’s Manual.
- Samples
    - Custom Data Type Editors (Visual Basic sources and the compiled files) may be found in subdirectories of <TestBase installation directory>\Samples\CDT Editors)
    - Test Procedures and Reports using CDTEs are used in test strategies implemented in Project files Sample.tdd and Demo.tdd.

Note: By default, this feature is disabled in the IDE. To enable it, select the menu **Tools | Development Options | Custom Data Types**. Select the menu again to disable the feature. A checkmark adjacent to the menu text indicates if the feature is currently enables.

The following new elements were added since the Beta2 release: COM category registration; documentation; enhancements in samples.

2.3.2. Test Procedure Adapter for National Instruments LabVIEW

This release includes a TestBase Adapter for LabVIEW.

- Note: You need a LabVIEW development license to develop test procedures with LabVIEW, or to run the LabVIEW samples provided with TestBase. An evaluation version of LabVIEW obtained from National Instruments may also be used.

The following new elements were added since the Beta2 release: support for callback functions; support for returning error and warning conditions; support for Custom Data Types; improved support for Abort (Abort is now instantaneous); documentation; enhancements in samples.

2.3.3. Test Procedure Adapter for Agilent VEE

This release includes a TestBase Adapter for Agilent VEE.

- Note: You need a VEE development license to develop test procedures with VEE, or to run the VEE samples provided with TestBase in debug mode. An evaluation version of VEE obtained from Agilent Technologies may also be used. You only need the VEE run-time to run the VEE samples provided with TestBase in non-debug mode. This run-time is redistributed with TestBase.

The following new elements were added since the Beta2 release: support for callback functions; support for returning error and warning conditions; improved support for Abort (Abort is now instantaneous); documentation; enhancements in samples.

2.3.4. Generic Procedures

This release includes a new TestBase IDE feature – Generic Procedures. These test procedures, included in the regular TestBase distribution, perform generic tasks such as: user input/output, comparisons and expression evaluation. They may be used to build test strategies, similar to the user-developed test procedures. The usage of Generic Procedures is illustrated in the test strategies located under UUT Model node “GenericTPs”, in Project file Sample.ttd. Utilization instructions are provided in the Help System of the TestBase IDE.

Note: The Generic Procedure “EvaluateExpression” is not documented in the Help System. The documentation is included in Section **Error! Reference source not found.** of these Release Notes.

The following features were added since the Beta2 release: new procedure “EvaluateExpression”; samples.

2.3.5. New Compiler

This release includes a new compiler for the TestBase IDE. This compiler is up to 12 times faster than the original compiler (the actual improvement depends on the particularities of compiled Test Strategies) and has more powerful verification capabilities.

Note: The new compiler is active by default. To activate the old compiler, change the value of the environment variable TB_COMPILER_VER (automatically created by the installer) to 1. Change the value back to 2 to activate the new compiler.

The Help System distributed with the current release describes the original compiler and the new compiler.

The main differences between the old compiler and the new compiler are described below:

- New flowchart drawing rules:

Note: These rules were introduced to speed up the operation of the compiler. Previously developed Test Procedures may need minor adjustments to comply with the new rules. These rules are expected to impact a small percentage of existing Test Procedures.

  - If a Test or DP Call block is followed by a decision block, these two blocks must be directly connected (i.e., not through an off-page reference). This rule was introduced to speed up the operation of the compiler. Previously developed Test Procedures may need minor adjustments to comply with the rule. The rule is expected to impact a small percentage of existing Test Procedures.
  - START blocks must be connected directly to Test, Display or DP Call blocks.

- The new compiler detects the situation when the data type of an Input Parameter was changed in the Test database (.ttd), while one or more immediate values for the parameter were specified in a Diagnostic database (.tdd). The above situation has the potential of causing errors at run-time. The immediate values are reset to their defaults and a warning is issued to inform the user about this automatic change.
- The new compiler detects the case when the Test Procedure assigned to a Test has no Outcome Values defined, while the Test block is followed by decision block(s). A similar behavior is provided for DP Call blocks. This change does not affect the operation of previously developed Test Procedures (the above case, detected at development time by the new compiler, caused a run-time error with the original compiler).
- The new compiler detects the case when the Test Procedure assigned to a Test has Outcome Values not handled by the decision block(s) that follow the Test. A similar behavior is provided for DP Call blocks. This change does not affect the operation of previously developed Test Procedures (the above case, detected at development time by the new compiler, caused a run-time error with the original compiler).
- The case when the decision block(s) that follow the Test handle additional Outcome Values, besides those of the Test Procedure assigned to the Test, causes a warning. A similar behavior is provided for DP Call blocks. These situations were reported as errors by the original compiler. This change does not affect the operation of previously developed Test Procedures (the above case can not produce run-time errors).
- The new compiler issues a warning when detected Test, Document or DP Call blocks that are not connected to the control flow.
- The new compiler does not issue a warning when the beginning of a connector is not connected to a block. This change does not affect the operation of existing Test Procedures.
- The new compiler does not issue a warning when connectors are partially overlapped. This change does not affect the operation of previously developed Test Procedures.

2.3.6. Speed improvements

The implementation of existing TestBase components was modified to increase execution speed. The modifications do not change the operation of components. The following improvements are implemented:

- The speed of the original compiler was increased several times (the actual improvement depends on the particularities of compiled Test Strategies), to support users who do not want to switch to the new compiler (see Section **Error! Reference source not found.**).

- The time required to load Projects in the Diagnostic Controller and Functional Test Controller user interfaces was reduced to almost half (the actual improvement depends on the particularities of Test Procedures characterized in the Test Database assigned to the Project).

2.3.7. “Simulate” flag in IDE

This feature enables the user to change the value of the Simulate flag in IDE. In previous releases, simulation control was available only through the toolbar buttons **Step** and **Simulated Step**. To set the Simulate flag, select the IDE menu **Tools | Run-time options...,** click the **Flags** tab, then click the appropriate radio button in the **Simulate** frame.

Note: The value set by the user for the Simulate flag is sampled by the IDE at the beginning of each Test Strategy execution. Changes made during execution are NOT reflected in the current execution.

2.3.8. “Break on FAIL, Error and Warning” in Functional Test Controller

The controlled execution of test strategies in the Functional Test Controller has a new capability: break (interrupt execution) when a test procedure returns an error condition, a warning condition or a FAIL condition (represented by returning False in the “Pass” flag). This functionality may be configured from the menu **Tools | Options | UI-FT Specific**. It is documented in the “Functional Test Controller “ section of the Help System.

2.3.9. Diagnostic Procedure HTML Reports

The **Display DP Report** toolbar button in IDE generates reports in HTML format (Excel reports were generated by previous versions of TestBase). This change eliminates the need to have Excel installed for users that do not need the Excel Adapter. In addition, the new HTML reports offer a dual graphical/textual representation.

2.3.10. Support for LabWindows/CVI 6.0

The LabWindows Adapter supports the use of LabWindows/CVI 6.0 for developing and executing test procedures. The samples distributed with TestBase are built using LabWindows/CVI 6.0. The LabWindows/CVI 6.0 run-time engine in included in the distribution.

2.3.11. Support for MANUAL INTERVENTION in the PAWS Adapter

The PAWS Adapter now offers support for the execution of ATLAS programs that contain MANUAL INTERVENTION and MONITOR statements.

The support provided by the adapter must be complemented by additional support in the production user interface. In principle, the user interface should include an indicator that is active when a MANUAL INTERVENTION statement is reached and a button that is clicked by the user in order to resume execution.

The same functionality supports the MONITOR statement. While the MONITOR statement is repeatedly, the indicator is active. When the button is pressed, execution continues with the following statement.

The development of user interfaces that implement the above functionality is described in detail in the Developer’s Manual.

2.3.12. Import and export of array values, in DiagML and MS Excel formats

This release adds new functionality to the export/import capabilities of TestBase, enabling the export and re-import of parameter values of type array, using the DiagML format. This functionality, available in the Array Editor window of the TestBase IDE, is documented in the Help System.

This release enables the import of parameter values of type array from Excel spreadsheets. This feature allows the generation of array data outside TestBase, using mathematical formulas or other algorithms implemented in Excel. This functionality, available in the Array Editor window of the TestBase IDE, is documented in the Help System.

2.3.13. Dynamic resizing of drop-down lists in IDE

The drop-down lists displayed in the grid (table) area of the IDE are now dynamically resized, according to the height of the grid area. This improves accessibility to list entries.

2.3.14. Memorization of grid column widths in IDE

When the user changes the widths of a grid column, the new width is memorized and used each time a column with the same name is displayed, in the current session and in future sessions.

2.3.15. Opening database files directly from the Windows Explorer

Starting with this version, the TestBase files (extensions .tdd, .ttd and .ted) are associated to the TestBase application. You can open these files by double-clicking their icons in the Windows Explorer.

2.3.16. Improved installation procedure

The installation procedure is guided by the Welcome Screens, a set of HTML pages include a brief description of the installation steps and provide access to various installation packages, for TestBase and redistributables. Details are available in the Installation Guide.

The location of redistributables was moved on the installation CD, reducing the requirements for hard drive space during installation.

Starting with this version, the TestBase installation procedure supports the use of licensing via hardware keys. Details are available in the Installation Guide.

2.3.17. Improved documentation

All developer-oriented documentation is now accessible in a unique document, the Developer’s Manual. The development of test procedures and reports is exemplified using a common set of samples, for all test languages and report formats.

The “Walkthrough” section of the Help System was improved and extended, to provide a solid introduction to the usage of all TestBase user interface modules. The samples used in the “Walkthrough” are consistent with those described in the Developer’s Manual.

2.4 Problem Reports

This section addresses problem reports received for releases 2.4.1, 2.5.0 Beta and 2.5.0 Beta2.

03007 - “Unable to export CFD" error when generating DP report

This appears to be caused by a peculiar behavior of the Visio application. A workaround solution is provided in Section 2.1.3.

03008 - Long lists not showing last items (under Windows XP)

This was corrected by dynamically resizing list height. See Section 2.3.13 for details.

03009 - Abort does not stop TP in VEE

Abort occurs only at the end of the test procedure.

Corrected. See Section 2.3.3 for details.

03010 - Type libraries not found

The type libraries necessary when developing COM test procedures are not found in the location indicated in the developer's documentation.

The documentation was corrected.

03011 - MTI storage with VEE Adapter

Output parameters of type array are not supported in conjunction with MTI storage. The Problem Report is actually mislabeled, the problem occurring for the LabVIEW Adapter.

The MTI storage was corrected.

03043 - LabVIEW Test Procedures cause random abort after long time

This problem is caused by a memory management problem in the LabVIEW engine. Corrected using a fix provided by National Instruments.

Execution of successive LabVIEW Test Procedures fails randomly

This was corrected by changing the lifetime management of the LabVIEW engine. See Critical Item in Section 2.1.1 for important configuration instructions.

Storage of unsigned values in DiagML

In previous versions, the values of parameters with unsigned integer data types where shifted, when stored in DiagML files (similar to the shifting used when storing such values in MTI databases).

The behavior was changed in release 2.5.0 - the values stored in DiagML files are identical to the values entered in IDE.

The above change will cause a problem when re-importing DiagML files exported with previous versions, if these files contain parameters with unsigned integer types. The original files may be easily corrected, to avoid such problems. Please contact TYX for instructions.

Abort of Global Parameter Value Acquisition Procedures

Aborting Global Parameter Value Acquisition Procedures causes fatal errors under some circumstances.

Corrected.

Abort from Functional Test Controller

Aborting execution from the Functional Test Controller Global Parameter causes fatal errors under some circumstances.

Corrected.

2.5. Additional Documentation

2.5.1. Generic Procedure "EvaluateExpression"

_EvaluateExpression – calculates the result of an expression

Input Parameters

- Arg0…Arg9 – expression arguments; have the type "double"
- Expression – string representing the expression to be evaluated; details are provided in the following

Expression syntax

The expression string can contain:

- immediate numeric constants, as described below
- operators, as described below
- functions, as described below
- the arguments Arg0 ... Arg9

Spaces can be freely introduced to visually separate expression elements. Parentheses '(' and ')' can be used to override operator precedence as necessary.

Numeric constants

You can specify constant numbers exactly like in programming languages, using the "e" or the "E" to separate mantissa from exponent and the point "." as decimal separator. Do not use the comma "," as decimal separator as it's used as parameter separator. Do not use the space " " as decimal separator as it could represent implied multiplication. Examples:

Correct	**Incorrect**
1	1,000,000
1.0	1,5
1e5	1e+0.5
1000000	1 000 000
1e+5	.05

Operators

Operator	**Name**	**Description**
& AND	logical AND operators	Combine multiple conditions formed using relational or equality expressions. Return the value 1 if both operands are nonzero; otherwise, return 0.
| OR	logical OR operators	Combine multiple conditions formed using relational or equality expressions. Return the value 1 if both operands are nonzero; otherwise, return 0.
NOT	logical negation	Returns the value 0 if its operand is nonzero; returns the value 1 if its operand is 0.
> < = >= <=	binary relational and equality operators	Compare their first operand to their second operand to test the validity of the specified relationship. Return 1 if the tested relationship is true and 0 if it is false.
<> >< !=	binary inequality operators	Compare their first operand to their second operand to test whether they are different. Return 1 if they are not equal, 0 if they are equal.
+	binary addition operator	Causes its two operands to be added.
-	binary subtraction operator	Subtracts the second operand from the first.
–	unary minus operator	Causes the following operand to undergo a change in sign.
*	binary multiplication operator	Causes its two operands to be multiplied.
/	binary division operator	Causes the first operand to be divided by the second.
^	binary power operator	Calculates the first operand raised to the power of the second operand.
!	unary factorial operator	Calculates the factorial of the operand.

The table below contains all the operators, in order of increasing evaluation precedence:

&, AND, |, OR
NOT
>=, <=, >, =, <, <>, ><, !=
+, -
-
*, /
^
!

Functions

Function	**Description**
ABS(x)	Returns the absolute value of x. Example: ABS(-5) returns 5.
ACOS(x)	Returns the arc-cosine of x. The return value is an angle specified in radians.
ASIN(x)	Returns the arc-sine of x. The return value is an angle specified in radians.
ATAN(x)	Returns the arc-tangent of x. The return value is an angle specified in radians.
CEIL(x)	Returns a floating-point value representing the smallest integer that is greater than or equal to x. Example: CEIL(4.3) returns 5.

COS(x)	Returns the cosine of x. The x value is an angle expressed in radians.
COSH(x)	Returns the hyperbolic cosine of x. The x value is an angle expressed in radians.
DIV(x,y)	Returns the quotient of the integer division between x and y. The x and y values must be integer values, otherwise unpredictable results will be generated.
MOD(x,y)	Returns the remainder of the integer division between x and y. The x and y values must be integer values, otherwise unpredictable results will be generated.
EXP(x,y)	Returns the exponential of x, that is to say the neperian constant *e* raised to the x-th power.
FLOOR(x,y)	Returns a floating-point value representing the largest integer that is less than or equal to x. Example: FLOOR(6.7) returns 6.
HYPOT(x,y)	Returns the hypotenuse of a right triangle with the given cateti x and y. Example: HYPOT(3,4) returns 5.
LN(x)	Returns the natural logarithm of x. If x is negative, returns an indefinite. If x is 0, returns INF (infinity).
LOG(x)	Returns the base-10 logarithm of x. If x is negative, returns an indefinite. If x is 0, returns INF (infinity).
MAX(x,y)	Returns the maximum of x and y.
MIN(x,y)	Returns the minimum of x and y.
RAND(x)	Returns a pseudorandom integer in the range 0 to x.
SIN(x)	Returns the sine of x. The x value is an angle expressed in radians.
SINH(x)	Returns the hyperbolic sine of x. The x value is an angle expressed in radians.
SQRT(x)	Returns the square root of x.
TAN(x)	Returns the tangent of x. The x value is an angle expressed in radians.
TANH(x)	Returns the hyperbolic tangent of x. The x value is an angle expressed in radians.

Internal number representation

The evaluation algorithm makes uses the type *double* (8 bytes) for internal number representation. The range for this data type is +/-1.7E308, with at least 15 digits of precision.

Examples

Arg1*(Arg1-Arg3)

1.2*SIN(Arg3)

<u>Output Parameters</u>

- Result – contains the expression evaluation result; has the type "double"

<u>Outcome Values</u>

- PASS – returned when the expression was successfully evaluated
- FAIL – returned when expression evaluation is impossible, although the expression's syntax is correct; this may occur, for example, in the case of a division by zero

<u>Error conditions</u>

An error condition is returned if the expression has an incorrect syntax. The following error messages may be returned:

Error Message	**Description**

Empty expression	This error arises when the expression string is empty, or composed of white spaces only
Wrong number of arguments	This error arises when you call a function with a bad number of arguments, for example when you try to evaluate the expression: sin(45, 2), when the sin() is known to accept one only argument.
Wrong use of reserved word	This error occur when you try to use a name of a function as if it were a variable or a constant or something else. In practice this is true whenever the parentheses which indicate a function call are not used. For example the following expressions generate this type of error: 1 + sin (COS + TAN) sin 45
An argument was expected	This error usually occurs when an expression is truncated or when you forget an operand or an argument; this error is also generated when you forget a parenthesis at the end of a correct expression, rendering it an unusable sub-expression. For example the following expressions generate this sort of error: Sin((1+5)/(4+) (1+5)/(4+5
Undefined function	This error is reported when you use an identifier name (a name that can be used to name a variable) immediately before an open parenthesis: it seems to be a function name but it is not (probably because you misspelled the name, or perhaps because you intended to multiply a variable by something that is inside the parenthesis). Examples: x(5+4)+y - you intended: x*(5+4)+y sine(3.14) - you intended: sin(3.14)
Undefined identifier	This error is reported when you use an identifier name (a name that can be used to name a variable) different from Arg0 ... Arg9, probably because you misspelled the name.