UTSL Reference

Universal Test Specification Language

http://utsl.net/

Contents

Test Spec....5
Main Components....5
Test Step Components....6

Syntax....7
Concepts....7
Case Sensitivity....7
Whitespace....7
Identifiers....7
Class-Based Programming....7
Optional Function Parameters....8
Comments....8
Engineering Units....8
Multiplier....10
Scope....11
Possible Usage....11
Executable Code in Definition Fields....13
Reserved Words....13
Data Types....13
Basic Data Types....14
Pin....14
PinList....15
Site-aware Data Types....16
ConditionList....27
SerialDataFrame....29
SerialPort....29
SerialPortGen....30
Declaration....30
Constants....30
Variables....31
Arrays....31
Enumeration....33
Composite Data Type....34
Operators....35
Arithmetic Operators....35
Assignment Operators....35
Relational Operators....36
Flow Control....36
If/Else Statement....36
Site-Aware If/Else....37
Switch Statement....38
For Statement....39
While Statement....39
Procedure....40

Build-ins....42
Pins....42
Pins.Connect....42
Pins.ConnectAll....43
Pins.ConnectDib....43

Pins.Disconnect....43
Pins.DisconnectAll....44
Pins.DisconnectDib....45
Pins.Gate....45
Pins.Current....46
Pins.Digital....54
Pins.Signals....56
Pins.TesterSettings....57
Pins.Time....61
Pins.Voltage....77
Digital....88
Digital.ApplyLevels....88
Digital.ApplyLevelsTiming....89
Digital.ApplyTiming....89
Digital.LastVectorCount....90
Digital.LastVectorFailCount....90
Digital.Patterns....91
Digital.Ports....91
Spec....100
Spec.Author....100
Spec.DeviceName....100
Spec.Version....100
Spec.Test....101
Spec.Tests....102
Tester....104
Tester.Configure....104
Tester.CustomCode....105
Tester.Function....105
DIB....106
DIB.Application....106
Math....107
Math.Abs....107
Math.Log10....107
Math.Min....108
Math.Max....109
Math.Pow....111
Math.Sqrt....112
Math.Truncate....113
Evaluate....114
Wait....115

Enumerations....116

AlarmType....116
BandwidthRange....117
BitPolarity....117
ConnectType....118
DigitalLevel....118
DigitalState....119
DigitalStateWhen....119
InstrumentType....120
MeasureReadFormat....121
SerialBitOrder....122
SignalSlope....122
TimeHysteresis....123
TimeImpedance....124
TimeStartInput....125
WaitType....125

Environment Variables....126
Temperatures....126
Sequencer....126
Device Pins....127
Part Variations....127

Concepts....129
Parallel Test Concepts....129
Programming List of Pins....129
Site-aware Variables....129

Legend....130

Test Spec

UTSL is used in a document called the test specification (test spec).

This is a complete specification of the tests for a device or device family. The document consists of XML-based data objects which contain test language fragments at points where test information is best represented using a procedural language rather than data definitions.

Main Components

At the top level there are seven major components of the test spec.

Device Pins

This defines the list of pins to be controlled and measured throughout the test definitions.

Part Variations

This is the list of part types included in this spec, in cases of device families.

Test Steps

This defines the available test environments such as wafer and final test or testing at different temperatures. It specifies which environments are applicable to this test spec. The different environments may have different test lists, different limits and conditional actions in the runtime code.

Definitions and Setup Code

This is a UTSL code block which contains one or more functions which establish standard device setups to be used as the starting points of tests. Often there are pin setups which are the same for many tests; defining these here reduces errors compared to duplicating the setup in each test. This code also contains global data definitions common to all tests such as serial communication port definitions.

Functions

Defines global variables and functions which can be used in the whole test spec.

Test Steps

Tests are organized into test steps which each contain a list of individual tests.

Setdown Code

tbd

Related Links

Scope on page 11

Device Pins on page 127
List of all device pins.

Part Variations on page 127

List of all device parts.

Test Step Components

Each test step consists of four main parts.

Definitions Code

This code contains global variables and setup functions shared by all tests in this test step, but which are not accessible by other test steps.

Setup Code

This is a fragment of executable UTSL code which is run at the beginning of the test step. It establishes the proper setup prior to the first test and usually starts by calling a standard setup function in the global definitions area. The setup code is responsible for making a complete device setup regardless of starting conditions, so that test steps can be executed independently and in any order.

Tests

Each test step contains its own list of tests. Which contains conditions for measure and evaluate the results.

Setdown Code

tbd

Related Links

Scope on page 11

Syntax

UTSL syntax definitions

This section is an overall description of the UTSL syntax and major features. Overall these are designed to be consistent with the C, C++, and C# (Microsoft C Sharp) family of languages, with the addition of a class-based model for device pin stimulus and measurement.

Concepts

tbd

Case Sensitivity

The language is case-sensitive. This applies to both built-in language features and user-defined identifiers. For example, “DBus” and “dbus” are two different variables.

Whitespace

Whitespace in the language consists of space, \t (tab), and \n (newline character). Whitespace serves only to separate language tokens and is not significant in the semantics.

Identifiers

Identifiers (variable names) may contain only alphanumeric characters plus underscore. The first character may not be a numerical digit.

Class-Based Programming

Access to features for programming pin setups and spec information is done through a predefined class hierarchy using dot notation:

```
Pins(Pin1).Voltage.Force(1.0V, 0.5mA, 2.0V, 1.0mA);
```

The language itself does not support creating user-defined classes.

Class objects contain both functions and properties.

Functions

```
Pins(Pin1).Voltage.Force(1.0V, 0.5mA, 2.0V, 1.0mA);
result = Pins(Pl).Voltage.Meter.Read();
```

Properties

Properties provide a convenient way to define a parameter that can be both written and read back without requiring a separate readback function. A property is programmed like a C struct data field and may be read-only, write-only, or read/write depending on the property definition.

```
Pins(Pin1).Voltage.Value = 1.0V; // Sets the value
Lowlimit = Spec.Test.LowLimit;    // Gets the value
```

Optional Function Parameters

Parameters to functions may include optional parameters (indicated as "[optional]" in this documentation). The syntax is to use "NC" (meaning "No Change") in place of the omitted parameter. Using NC for a parameter which is not optional will cause an error. NCs at the end of the parameter list may be omitted.

```
Pins(Pin1).Voltage.Force(1.0V, 0.5mA, NC, NC);
Pins(Pin1).Voltage.Force(1.0V, 0.5mA);        // Same as previous line
Pins(Pin1).Voltage.Force(1.0V, NC, NC, 1.0mA);  // NCs are required
```

Comments

The language supports both C multi-line (/* */) and C++ single-line (//) comment styles.

Syntax single-line comment

// Comment

Syntax multi-line comment

/* Comment line 1
    Comment line 2*/

Example single-line comment

```
// Everything from here to the end of line is a comment
```

Example multi-line comment

```
/* This comment may continue across
   multiple lines */
```

Engineering Units

Readability of test specifications is greatly improved by including engineering units along with numerical constants.

UTSL allows numerical constants of type double to include a units code along with an optional one-character prefix denoting an engineering-notation *multiplier*.

Unit	Description
A	Amperes
B	Baud
bar	Barometric pressure
C or Cel	Degrees Celsius (temperature)
deg	Degrees (angle)
F	Farads

Unit	Description
g	Grams
H	Henries (inductance)
Hz	Hertz
J	Joules
K	Degrees Kelvin (temperature)
LSB	Least Significant Bit
m	Meters
N	Newtons (force)
Ohm	Ohms
Pa	Pascals (pressure)
rad	Radians
s	Seconds
W	Watts
V	Volts
dB	Decibel (no multipliers allowed)
%	Percent (no multipliers allowed)
A_per_V	Amperes per Volt
A_per_LSB	Amperes per LSB
Cel_per_s	Degrees Celsius per second
Cel_per_Cel_per_s	Degress C per degree second
F_per_Cel_ per_s	Farads per degree second
Hz_per_Vsqr	Hertz per Volt-squared
nv_V	Inverse-Volts (1/V)
K_per_W	Degrees Kelvin per Watt
LSB_per_V	LSBs per Volt
LSB_per_A	LSBs per Ampere
N_per_m	Newtons per meter (surface tension)
V_per_s	Volts per second
V_per_us	Volts per microsecond
V_per_ns	Volts per nanosecond
V_per_LSB	Volts per LSB
V_per_g	Volts per gram
Vsqr	Volts-squared (power times resistance)

Examples

```
x = 1.0V; // Units with no prefix
x = 1.0mV; // x is (1.0 * 10e-3)
```

Related Links

Multiplier on page 10
The prefix multiplies the value by the indicated factor

Basic Numeric Data Types on page 14
The basic numeric data types supported by UTSL

Multiplier

The prefix multiplies the value by the indicated factor

Prefix	Multiplier	Description
E	exa	10e18
P	peta	10e15
T	tera	10e12
G	giga	10e9
M	mega	10e6
K	kilo	10e3
c	centi	10e-2
m	milli	10e-3
u	micro	10e-6
n	nano	10e-9
p	pico	10e-12
f	femto	10e-15
a	atto	10e-18

Examples

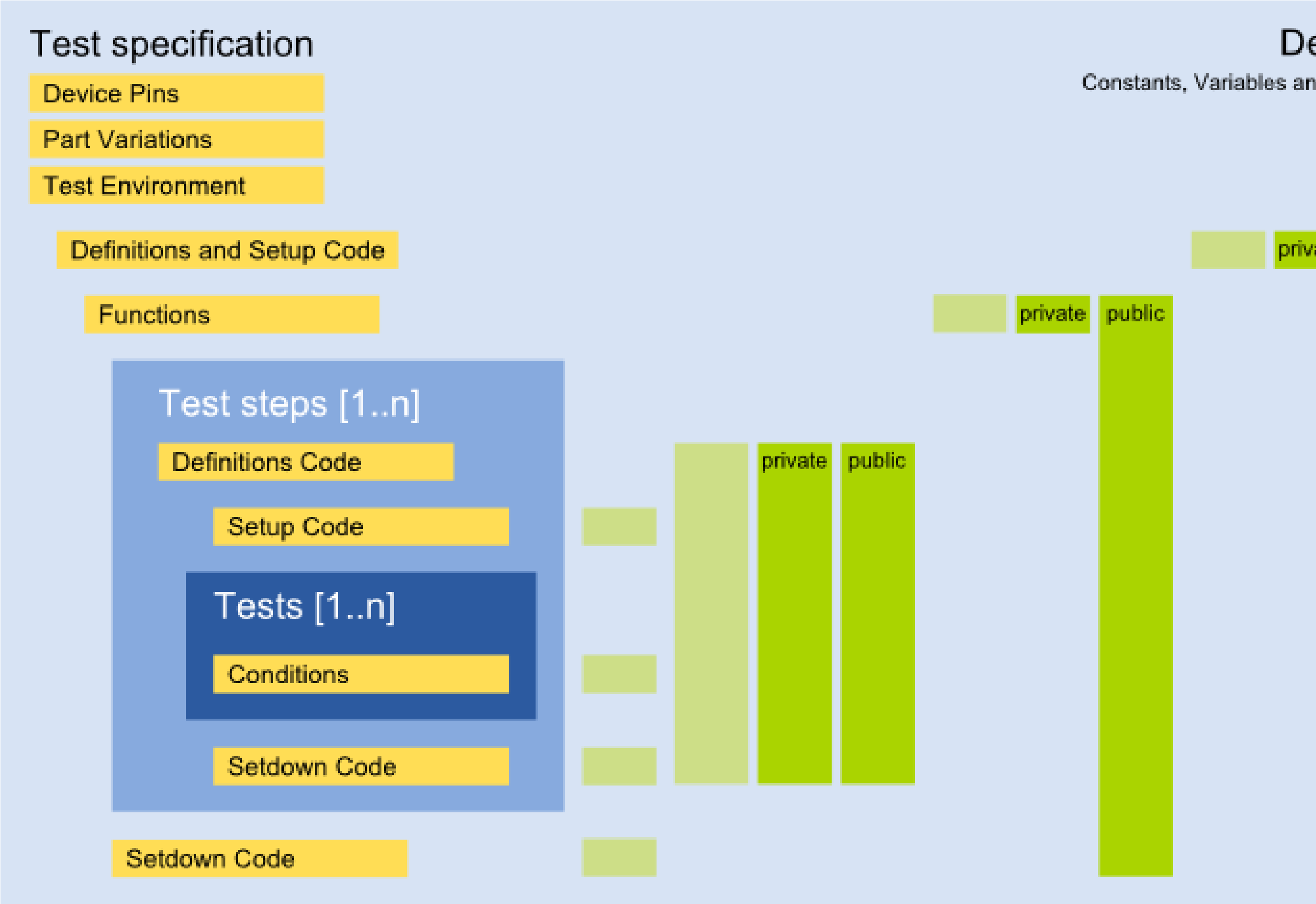
```
x = 1.0V; // Units with no prefix
x = 1.0mV; // x is (1.0 * 10e-3)
```

Related Links

Engineering Units on page 8

Readability of test specifications is greatly improved by including engineering units along with numerical constants.

Scope



Related Links

Main Components on page 5
At the top level there are seven major components of the test spec.

Test Step Components on page 6
Each test step consists of four main parts.

Possible Usage

Possible usage of declarations and executable code.

Header | Basic Setup/Setdown

	Declaration of public variables, constants and enums	Declaration of private variables	Declaration of private constants	Declaration of private enums	Assignment of values to variables	Declaration of public and private functions	Execution of build-ins and functions
Definitions and Setup Code	✓	✓	✓	✓		✓	

	Declaration of public variables, constants and enums	Declaration of private variables	Declaration of private constants	Declaration of private enums	Assignment of values to variables	Declaration of public and private functions	Execution of build-ins and functions
Setdown Code		✓			✓		✓

Header | Functions

	Declaration of public variables, constants and enums	Declaration of private variables	Declaration of private constants	Declaration of private enums	Assignment of values to variables	Declaration of public and private functions	Execution of build-ins and functions
Functions	✓	✓	✓	✓		✓	

Test Steps

	Declaration of public variables, constants and enums	Declaration of private variables	Declaration of private constants	Declaration of private enums	Assignment of values to variables	Declaration of public and private functions	Execution of build-ins and functions
Definition Code	✓	✓	✓	✓		✓	
Setup Code		✓			✓		✓
Setdown Code		✓			✓		✓

Tests

	Declaration of public variables, constants and enums	Declaration of private variables	Declaration of private constants	Declaration of private enums	Assignment of values to variables	Declaration of public and private functions	Execution of build-ins and functions
Conditions							

	Declaration of public variables, constants and enums	Declaration of private variables	Declaration of private constants	Declaration of private enums	Assignment of values to variables	Declaration of public and private functions	Execution of build-ins and functions
		✓			✓		✓

Related Links

Executable Code in Definition Fields on page 13

Executable Code in Definition Fields

Sometimes it is necessary to add executable code (ex. build-ins) in definition fields. Therefor the following syntax can be used:

```
[Tester.Configure("Setup1")]
```

Build-in surrounded by [ ] and no ; at the end.

Related Links

Possible Usage on page 11
Possible usage of declarations and executable code.

Reserved Words

The following identifiers are built-in language keywords or types and may not be used as variable names or other identifiers.

bool	break	case	ConnectType	Digital	double
else	enum	Evaluate	false	for	if
int	NC	Null	Optional	Pin	PinList
Pins	private	public	return	SerialBitField	SerialBitFieldMnemonic
SerialDataFrame	SerialPort	SerialPortGen	SignalSlope	SiteBool	SiteDouble
SiteInt	Spec	string	struct	switch	System
TimeHysteresis	TimeImpedance	true	ValueList	void	while

- All device pin names from the spec pinlist are reserved as predefined global Pin variables.
- Part variations from the spec parts list are reserved as predefined global bool variables.
- Sequencers and temperatures are reserved as predefined global bool variables.

Data Types

UTSL is strongly typed; this allows a greater extent of error-checking and produces runtime code more likely to work the first time. For mathematical operations, the result and operands must be of the same data type.

Related Links

Constants on page 30
Declarations of constants.

Variables on page 31

Declarations of variables.

Arrays on page 31

Composite Data Type on page 34
A composite data type is any data type which can be constructed using all data types and enumerations.

Basic Data Types

The basic data types supported by UTSL.

Basic Numeric Data Types

The basic numeric data types supported by UTSL

Data type	Description	Values	Example
bool	Boolean	true or false	true
int	32-bit signed integer	Decimal	42
		Octal	0xf354
		Hexadecimal	0776
double	Double-precision floating point number.	Decimal notation	0.00345
		Engineering notation	3.45e-3
		Engineering units	3.45mV

Related Links

Engineering Units on page 8
Readability of test specifications is greatly improved by including engineering units along with numerical constants.

Basic String Data Type

The basic string data type supported by UTSL

Data type	Description	Values	Example
string	Text string	Double-quotes	"text1"

Pin

The Pin type declares a variable which can be any pin.

Related Links

Device Pins on page 127

List of all device pins.

Pin Operators

Operators

Operator		Description
!= / ==	bool PinA == PinB bool PinA != PinB	Checks whether pin A does or does not refer to the same pin as B.
+	PinList = PinA + PinB	Produces a new PinList which is the concatenation of the operand. Commonly used to build pinlists in initializations and function calls..

Example ==

```
if (pinx == VDD)
{
    Pins(pinx).Voltage.Value = 1V;
}
```

Example +

```
PinList plist = Pin1 + Pin2;
Pins(Pin1 + Pin2).Voltage.Value = 1V;
```

PinList

A PinList is an ordered list of device pins which is built by combining Pins and PinLists.

PinList.AddPin

Adds a pin to the end of a PinList.

Syntax

PinList.AddPin (P)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
P	The pin to add.	Pin	no	-	yes

Details

Type	Function
Spec relevant	yes

PinList.GetPinN

Finds the Nth pin in the list.

Syntax 1
Pin = PinList.GetPinN (Index)

Syntax 2
Pin = PinList[Index)]

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Index	Gets the Nth pin of a PinList (range 0 to PinList.Length-1).	int	no	-	yes

Details

Type	Function
Spec relevant	yes

PinList.Length

Counts the pins in the list.

Syntax
int = PinList.Length

Details

Type	Property
Spec relevant	yes

Site-aware Data Types

These data types store device results or parameters which may require multiple copies at runtime to support multi site test.

SiteBool

Site-aware version of basic type bool. Contains a boolean value per-site at runtime.
SiteBool Operators

Operators

Operator		Description

&&	SiteBool = SiteBool && SiteBool
	SiteBool = SiteBool &&
		Relational and operation with any combination of SiteBool and bool operands.

Operator		Description
	bool SiteBool = bool && SiteBool	

| |	SiteBool = SiteBool | | SiteBool
	SiteBool = SiteBool | | bool
	SiteBool = bool | | SiteBool
		Relational or operation with any combination of SiteBool and bool operands.

=	SiteBool = SiteBool
	SiteBool = bool
		Assignment from SiteBool or bool.

!	SiteBool = !SiteBool
	SiteBool = !bool
		Relational negation with SiteBool or bool operand.

SiteInt

Site-aware version of basic type bool. Contains a integer value per-site at runtime.

SiteInt Operators

Operators

Operator		Description

+	SiteInt = SiteInt + SiteInt
	SiteInt = SiteInt + int
	SiteInt = int + SiteInt
		Addition of any combination of SiteInt and int operands.

-	SiteInt = SiteInt - SiteInt
	SiteInt = SiteInt - int
	SiteInt = int - SiteInt
		Subtraction of any combination of SiteInt and int operands.

*	SiteInt = SiteInt * SiteInt
	SiteInt = SiteInt * int
	SiteInt = int * SiteInt
		Multiplication of any combination of SiteInt and int operands.

/	SiteInt = SiteInt / SiteInt	Division of any combination of SiteInt and int operands.

Operator		Description

	SiteInt = SiteInt / int
	SiteInt = int / SiteInt	

<<	SiteInt = SiteInt << int	Shift left.
>>	SiteInt = SiteInt >> int	Shift right.

&	SiteInt = SiteInt & SiteInt
	SiteInt = SiteInt & int
	SiteInt = int & SiteInt
		Bitwise-and of any combination of SiteInt and int operands.

|	SiteInt = SiteInt | SiteInt
	SiteInt = SiteInt | int
	SiteInt = int | SiteInt
		Bitwise-or of any combination of SiteInt and int operands.

^	SiteInt = SiteInt ^ SiteInt
	SiteInt = SiteInt ^ int
	SiteInt = int ^ SiteInt
		Bitwise-exclusive or of any combination of SiteInt and int operands.

==	SiteBool = SiteInt == SiteInt
	SiteBool = SiteInt == int
	SiteBool = int == SiteInt
		Equality operator with any combination of SiteInt and int operands.

<	SiteBool = SiteInt < SiteInt
	SiteBool = SiteInt < int
	SiteBool = int < SiteInt
		Less-than operator with any combination of SiteInt and int operands.

<	SiteBool = SiteInt <= SiteInt
	SiteBool = SiteInt <= int
	SiteBool = int <= SiteInt
		Less-than-or-equal operator with any combination of SiteInt and int operands.

Operator		Description

>	SiteBool = SiteInt > SiteInt
	SiteBool = SiteInt > int
	SiteBool = int > SiteInt
		Greater-than operator with any combination of SiteInt and int operands.

>=	SiteBool = SiteInt >= SiteInt
	SiteBool = SiteInt >= int
	SiteBool = int >= SiteInt
		Greater-than-or-equal operator with any combination of SiteInt and int operands.

!=	SiteBool = SiteInt != SiteInt
	SiteBool = SiteInt != int
	SiteBool = int != SiteInt
		Inequality operator with any combination of SiteInt and int operands.

=	SiteInt = SiteInt
	SiteInt = int
		Assignment from SiteInt or int.

+	SiteInt = +SiteInt
	SiteInt = +int
		Unary plus with SiteInt or int operand.

-	SiteInt = -SiteInt
	SiteInt = -int
		Unary minus with SiteInt or int operand.

~	SiteInt = ~SiteInt
	SiteInt = ~int
		One's complement with SiteInt or int operand.

SiteDouble

Site-aware version of basic type bool. Contains a double value per-site at runtime.

SiteDouble Operators

Operators

Operator		Description
+	SiteDouble = SiteDouble + SiteDouble	Addition of any combination of SiteDouble, SiteInt, double, and int operands.

Operator		Description

	SiteDouble = SiteDouble + SiteInt
	SiteDouble = SiteInt + SiteDouble
	SiteDouble = SiteDouble + double
	SiteDouble = double + SiteDouble
	SiteDouble = SiteDouble + int
	SiteDouble = int + SiteDouble	

-	SiteDouble = SiteDouble - SiteDouble
	SiteDouble = SiteDouble - SiteInt
	SiteDouble = SiteInt - SiteDouble
	SiteDouble = SiteDouble - double
	SiteDouble = double - SiteDouble
	SiteDouble = SiteDouble - int
	SiteDouble = int - SiteDouble
		Subtraction of any combination of SiteDouble, SiteInt, double, and int operands.

*	SiteDouble = SiteDouble * SiteDouble
	SiteDouble = SiteDouble * SiteInt
	SiteDouble = SiteInt * SiteDouble
	SiteDouble = SiteDouble * double
	SiteDouble = double * SiteDouble
	SiteDouble = SiteDouble * int
	SiteDouble = int * SiteDouble
		Multiplication of any combination of SiteDouble, SiteInt, double, and int operands.

/	SiteDouble = SiteDouble / SiteDouble
	SiteDouble = SiteDouble / SiteInt
	SiteDouble = SiteInt / SiteDouble
	SiteDouble = SiteDouble / double
	SiteDouble = double / SiteDouble
	SiteDouble = SiteDouble /
		Division of any combination of SiteDouble, SiteInt, double, and int operands.

Operator		Description
	int SiteDouble = int / SiteDouble	

==	SiteBool = SiteDouble == SiteDouble
	SiteBool = SiteDouble == SiteInt
	SiteBool = SiteInt == SiteDouble
	SiteBool = SiteDouble == double
	SiteBool = double == SiteDouble
	SiteBool = SiteDouble == int
	SiteBool = int == SiteDouble
		Equality operator with any combination of SiteDouble, SiteInt, double, and int operands.

<	SiteBool = SiteDouble < SiteDouble
	SiteBool = SiteDouble < SiteInt
	SiteBool = SiteInt < SiteDouble
	SiteBool = SiteDouble < double
	SiteBool = double < SiteDouble
	SiteBool = SiteDouble < int
	SiteBool = int < SiteDouble
		Less-than operator with any combination of SiteDouble, SiteInt, double, and int operands.

<=	SiteBool = SiteDouble <= SiteDouble
	SiteBool = SiteDouble <= SiteInt
	SiteBool = SiteInt <= SiteDouble
	SiteBool = SiteDouble <= double
	SiteBool = double <= SiteDouble
	SiteBool = SiteDouble <= int
	SiteBool = int <= SiteDouble
		Less-than-or-equal operator with any combination of SiteDouble, SiteInt, double, and int operands.

>	SiteBool = SiteDouble > SiteDouble
	SiteBool = SiteDouble > SiteInt
	SiteBool = SiteInt > SiteDouble
	SiteBool = SiteDouble >
		Greater-than operator with any combination of SiteDouble, SiteInt, double, and int operands.

Operator		**Description**

	double
	SiteBool = double > SiteDouble
	SiteBool = SiteDouble > int
	SiteBool = int > SiteDouble	

>=	SiteBool = SiteDouble >= SiteDouble
	SiteBool = SiteDouble >= SiteInt
	SiteBool = SiteInt >= SiteDouble
	SiteBool = SiteDouble >= double
	SiteBool = double >= SiteDouble
	SiteBool = SiteDouble >= int
	SiteBool = int >= SiteDouble
		Greater-than-or-equal operator with any combination of SiteDouble, SiteInt, double, and int operands.

!=	SiteBool = SiteDouble != SiteDouble
	SiteBool = SiteDouble != SiteInt
	SiteBool = SiteInt != SiteDouble
	SiteBool = SiteDouble != double
	SiteBool = double != SiteDouble
	SiteBool = SiteDouble != int
	SiteBool = int != SiteDouble
		Inequality operator with any combination of SiteDouble, SiteInt, double, and int operands.

=	SiteDouble = SiteDouble
	SiteDouble = SiteInt
	SiteDouble = double
	SiteDouble = int
		Assignment from SiteDouble, SiteInt, double, or int.

+	SiteDouble = + SiteDouble
	SiteDouble = + SiteInt
	SiteDouble = + double
	SiteDouble = + int
		Unary plus with SiteDouble, SiteInt, double, or int operand.

Operator		Description

-	SiteDouble = - SiteDouble
	SiteDouble = - SiteInt
	SiteDouble = - double
	SiteDouble = - int
		Unary minus with SiteDouble, SiteInt, double, or int operand.

ValueList

ValueList is a class which stores a SiteDouble value per pin for a PinList. This is commonly used as a return value for measurement functions which allow measuring a list of pins simultaneously.

ValueList.GetData

Finds the data stored for a specific pin.

Syntax

SiteDouble = ValueList.GetData (WhichPin [, InstType])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
WhichPin	Finds the data stored for pin in PinList	Pin	no	-	yes
InstType	The type of instrument to match	*InstrumentType*	yes	Default	yes

Overload

SiteDouble = ValueList.GetData (Index)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Index	Finds the data stored for the Nth pin in PinList (range 0 to PinList.Length-1).	int	no	-	yes

Details

Type	Function
Spec relevant	yes

Related Links

InstrumentType. on page 120

The tester resource map allows using more than one type of instrument within a single test setup. For these cases, this enum selects a single instrument when needed, such as when choosing which instrument to connect with the Connect() function.

ValueList.SetData

Sets the data stored for a specific pin to a uniform value for all sites.

Syntax

ValueList.SetData (WhichPin , Value)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
WhichPin	Sets the data for pin in PinList.	Pin	no	-	yes
Value	The value to set for WhichPin.	double	no	-	yes

Details

Type	Function
Spec relevant	yes

ValueList.SetDataN

Sets the data stored for the Nth pin to a uniform value for all sites.

Syntax

ValueList.SetDataN (Index , Value)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Index	Sets the data for the Nth pin in PinList (range 0 to PinList.Length-1).	Pin	no	-	yes
Value	The value to set for Index.	double	no	-	yes

Details

Type	Function
Spec relevant	yes

ValueList.SetDataN

Sets the data stored for the Nth pin to a uniform value for all sites.

Syntax

ValueList.SetDataN (Index , Value)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Index	Sets the data for the Nth pin in PinList (range 0 to PinList.Length-1).	Pin	no	-	yes
Value	The value to set for Index.	double	no	-	yes

Details

Type	Function
Spec relevant	yes

ValueList.Pins

The list of pins for which values are stored.

Syntax

PinList = ValueList.Pins

Details

Type	Property
Access	Read/Write
Spec relevant	yes

ValueList Operators

Operators

Operator		Description

+	ValueList = ValueList + ValueList
	ValueList = ValueList + SiteDouble
	ValueList = SiteDouble + ValueList
	ValueList = ValueList + SiteInt
	ValueList = SiteInt + ValueList
	ValueList = ValueList + double
	ValueList = double + ValueList
	ValueList = ValueList + int
	ValueList = int + ValueList
		Addition of any combination of ValueList, SiteDouble, SiteInt, double, and int operands.

-		Subtraction of any combination of ValueList, SiteDouble,

Operator		Description

	ValueList = ValueList - ValueList
	ValueList = ValueList - SiteDouble
	ValueList = SiteDouble - ValueList
	ValueList = ValueList - SiteInt
	ValueList = SiteInt - ValueList
	ValueList = ValueList - double
	ValueList = double - ValueList
	ValueList = ValueList - int
	ValueList = int - ValueList
		SiteInt, double, and int operands.

*	ValueList = ValueList * ValueList
	ValueList = ValueList * SiteDouble
	ValueList = SiteDouble * ValueList
	ValueList = ValueList * SiteInt
	ValueList = SiteInt * ValueList
	ValueList = ValueList * double
	ValueList = double * ValueList
	ValueList = ValueList * int
	ValueList = int * ValueList
		Multiplication of any combination of ValueList, SiteDouble, SiteInt, double, and int operands.

/	ValueList = ValueList / ValueList
	ValueList = ValueList / SiteDouble
	ValueList = SiteDouble / ValueList
	ValueList = ValueList / SiteInt
	ValueList = SiteInt / ValueList
	ValueList = ValueList / double
	ValueList = double / ValueList
	ValueList = ValueList / int
	ValueList = int / ValueList
		Division of any combination of ValueList, SiteDouble, SiteInt, double, and int operands.

Operator		Description

=	ValueList = ValueList
	ValueList = SiteDouble
	ValueList = SiteInt
	ValueList = double
	ValueList = int
		Assignment from ValueList, SiteDouble, SiteInt, double, or int.

+=	ValueList = +ValueList
	ValueList = +SiteDouble
	ValueList = +SiteInt
	ValueList = +double
	ValueList = +int
		Unary plus with ValueList, SiteDouble, SiteInt, double, or int operand.

-=	ValueList = -ValueList
	ValueList = -SiteDouble
	ValueList = -SiteInt
	ValueList = -double
	ValueList = -int
		Unary minus with ValueList, SiteDouble, SiteInt, double, or int operand.

ConditionList

A class containing a list of pins with evaluation limits.

Conditionlist Declaration

Sometimes the result of a test depends on the evaluation of multiple measurements across different pins, requiring an evaluation too complicated for the single set of test limits supported per-test in the test spec. This class allows conveniently expressing and evaluating a list of conditions without lengthy if-else code.

Syntax

```
ConditionList = (ConditionListName) {
{
   {Pin-1, {"=" | "<" | "<=" | ">" | ">="}, Condition-1},
   {Pin-2, {"=" | "<" | "<=" | ">" | ">="}, Condition-2},
   …
   {Pin-n, {"=" | "<" | "<=" | ">" | ">="}, Condition-n}
}
};
```

Example

```
ConditionList cl = {
    {{Pin1, "<", 5V},
      {Pin2, ">", 6V}}
      {Pin3, ">", 2.9V}}
    };
```

ConditionList.Checkresult

Verify that the values for each pin in Result satisfy the defined conditions. For each pin found in Result.PinList, look for conditions assigned to that pin name and check whether they are satisfied.

Syntax

SiteBool = ConditionList.Checkresult(Result)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Result	The data to verify. Pins found in Result.PinList but not defined in the ConditionList will cause an error.	ValueList	no	-	yes

Return Value

Description	Data Type
True if all applicable conditions were satisfied, false otherwise, based on the values of each site.	SiteBool

Details

Type	Function
Spec relevant	yes

Example

```
// Declare the measurement conditions
ConditionList cl = {
    {{ Pin1, "<", 5V },
     { Pin2, ">", 6V },
     { Pin3, "<", 2.9V }}
};
PinList plst = {Pin1, Pin2, Pin3};

// Read the values and check against conditions
ValueList result = Pins(plst).Voltage.Meter.Read();
SiteBool testresult = cl.CheckResult(result);
```

```
Evaluate(testresult);
```

SerialDataFrame

Defines a serial data frame for use with the functions in Digital.Ports().

Related Links

Digital.Ports. on page 91
Functions for digital serial data communication.

SerialDataFrame Declaration

tbd

Syntax

tbd

Example

```
tbd
```

SerialPort

Defines a serial communication port for use with Digital.Ports() functions.

The SerialPort class defines a port which includes a preexisting, user-created digital pattern for executing the communication. The class has parameters defining the pattern name and labels. At runtime, pattern writes are executed by modifying the pattern data at the indicated label.

Related Links

Digital.Ports. on page 91
Functions for digital serial data communication.

SerialPort Declaration

tbd

Syntax

tbd

Example

```
tbd
```

SerialPortGen

Defines a serial communication port for use with Digital.Ports() functions.

The SerialPortGen class includes communication parameters instead of a pattern. For these ports, the code generator will automatically produce a pattern based which assumes a simple 4-pin serial protocol.

Related Links

Digital.Ports. on page 91
Functions for digital serial data communication.

Declaration

Constants, variables and type declarations

Constants

Declarations of constants.

Syntax common

[public | private] readonly DataType ConstName = Value;

Example common

```
private readonly int iVar = 42;
```

Syntax List of Pins

[public | private] readonly PinList ConstName = Pin-1 + Pin-2 + … + Pin-n;
[public | private] readonly PinList ConstName = {Pin-1, Pin-2, …, Pin-n};

Example List of Pins

```
private readonly PinList plST1 = Pin1 + Pin2;
public readonly PinList plST2 = {Pin1, Pin2};
```

Details

Spec relevant	no

Related Links

Data Types on page 13

UTSL is strongly typed; this allows a greater extent of error-checking and produces runtime code more likely to work the first time. For mathematical operations, the result and operands must be of the same data type.

Variables

Declarations of variables.

Syntax common

[public | private] DataType VarName;

VarName = Value;

Example common

```
public double g_dDemo;
private string strDemo;

g_dDemo = 1.2;
strDemo = "Text";
```

Syntax List of Pins

[public | private] PinList PinListName;

PinListName = Pin-1 + Pin-2 + … + Pin-n;
PinListName = {Pin-1, Pin-2, …, Pin-n};

Example List of Pins

```
public PinList plst1;
private PinList plst2;

plst1 = Pin1 + Pin2 + Pin3;   // recommended
plst2 = {Pin1, Pin2};
```

Details

Spec relevant	no

Related Links

Data Types on page 13
UTSL is strongly typed; this allows a greater extent of error-checking and produces runtime code more likely to work the first time. For mathematical operations, the result and operands must be of the same data type.

Arrays

Related Links

Data Types on page 13

UTSL is strongly typed; this allows a greater extent of error-checking and produces runtime code more likely to work the first time. For mathematical operations, the result and operands must be of the same data type.

Array Declaration

Arrays of built-in types are declared by adding square brackets to the type. Arrays may have one dimension only; multidimensional arrays are not supported.

Syntax

[public | private] DataType[] ArrayName;

ArrayName.Length = Length;

ArrayName[0..Length-1] = Value;

[public | private] DataType[] ArrayName = {Value-1, Value-2, …, Value-n};

Example

```
int[] aiArray;
aiArray.Length = 4;
aiArray[0] = 42;

double[] adlist = {1.0, 2.0, 3.0};
```

Details

Spec relevant	no

Possible Data Types

Type	**Description**
bool[]	Array of booleans
int[]	Array of Integer
double[]	Array of Double
string[]	Array of String
SiteBool[]	Array of SiteBool
SiteInt[]	Array of SietInt
SiteDouble[]	Array of SiteDouble

Array Functions

Functions

Function	**Description**
SiteDouble = ArrayName.Clone	Creates a copy of the array.

Array Properties

Properties

Property	Description
bool = ArrayName.IsFixedSize	Gets a value indicating whether the array has a fixed size.

Enumeration

An enumeration is a data type consisting of a set of named values called members of the type.

Syntax

```
[public | private] enum Enum
{
   MemberName-1 [= Constant-1]
   MemberName-2 [= Constant-2]
   …
   MemberName-n [= Constant-n]
}

[public | private] Enum EnumName;
```

Example without constants

```
public enum Color
{
   blue
   red
   yellow
   green
   pink
}

public Color eColor;
```

Example with constants

```
public enum Color
{
   blue = 1
   red = 2
   yellow = 10
   green = 20
   pink = 1000
}

public Color eColor;
```

Related Links

Enumerations on page 116

Enumerations are defined for parameters of the built in UTSL functions.

Composite Data Type

A composite data type is any data type which can be constructed using all data types and enumerations.

Syntax

```
[public | private] struct CompDataType
{
   public {DataType | Enum}[ [] ] ElementName-1;
   public {DataType | Enum}[ [] ] ElementName-2;
   …
   public {DataType | Enum}[ [] ] ElementName-n;
}

[public | private] CompDataType CompDataTypeName;
```

Example

```
public struct Register
{
    public string sModuleName;
    public string sRegisterName;
    public double[] iaZahl;
    public Color eColor;
}

public Register scRegister;

scRegister.sModuleName = "Mod1";
scRegister.iaZahl[2] = 3;
scRegister.eColorValue = Color.red;
```

Details

Spec relevant	no

Related Links

Data Types on page 13

UTSL is strongly typed; this allows a greater extent of error-checking and produces runtime code more likely to work the first time. For mathematical operations, the result and operands must be of the same data type.

Operators

Arithmetic Operators

Operators

Operator	Description	Example
+	Addition and PinList concatenation	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus division (return remainder)	x % y
<<	Left shift	x << y
>>	Right shift	x << y
&	Bitwise AND	x & y
|	Bitwise OR	x | y
^	Bitwise exclusive OR	x ^ y
~	Bitwise complement	x ~ y

Assignment Operators

Operators

Operator	Description	Example
=	Assignment	
+=	Addition assignment (same as x = x + y)	x += y
-=	Subtraction assignment (same as x = x – y)	x -= y
*=	Multiplication assignment (same as x = x * y)	x *= y
/=	Division assignment (same as x = x / y)	x /= y
%=	Modulus assignment (same as x = x % y)	x %= y
>>=	Right-shift assignment (same as x = x >> y)	x >>= y

Operator	Description	Example
&=	Bitwise AND assignment (same as x = x & y)	x &= y
|=	Bitwise OR assignment (same as x = x | y)	x |= y
^=	Bitwise XOR assignment (same as x = x ^ y)	x ^= y

Relational Operators

Operators

Operator	Description	Example
<	Less than	x < y
<=	Less than or equal	x <= y
>=	Greater than	x > y
>=	Greater than or equal	x >= y
!=	Not equal	x != y
==	Equality	x = y
||	Locical OR	x || y
&&	Locical AND	x && y
!	Locical NOT	x ! y

Flow Control

Statements to control the flow of the program taken from the C/C++ languages.

If/Else Statement

Conditional execution of code blocks.

The if/else conditional statements are taken directly from the C language, except that to provide better error-checking, the condition must be a boolean expression. For non-boolean types, an explicit "==" must be used to check the value. This is a change made in C# which makes the statement less prone to coding errors.

The condition may evaluate to either bool or SiteBoolean. If bool, the statement is a regular scalar if-statement which executes the enclosing statements only if the condition is true.

Syntax

```
if (Condition) {
   [Statement sequence if;]
}
[else {
   Statement sequence else;
}]
```

Example

```
bool bvar = true;
int ivar = 43;
int x;

if (bvar) { // Ok, bvar is boolean
  x = 0;
}

if (ivar == 43) { // Ok, result of == is boolean
  x = 1;
}
else if (ivar) { // Error, ivar is not boolean
  x = 2;
}
```

Site-Aware If/Else

For operations on site-aware device results and device force/measurement, the if-statement condition may be a SiteBool. This means the enclosed statements will only affect sites which are true in the condition expression.

Example

```
// Read the fail count on two pins.
ValueList counts = Pins(Pin1+Pin2).Digital.PinFailCount();

// Extract the count for one pin.
// The counts may be different for different sites.
SiteDouble onepin = counts.GetData(Pin1);

// Evaluate pass/fail based on an expected fail count.
SiteBool result;
if (onepin == 43)
{
   result = true;     // Sets only sites for which onepin == 43
}
else
{
   result = false;    // Sets only sites for which onepin != 43
}
```

Statements may include else-if clauses, with a restriction that the condition types must match the original if-statement condition. If the original condition is bool, only bool is allowed in subsequent else-if clauses. Likewise of the original condition is SiteBool, only SiteBool is allowed in else-ifs.

If the if or else clause contains scalar statements such as math on scalar variables, these statements are executed unconditionally in each clause. Only site-aware operations have the conditional behavior. In a site-aware if statement, the conditional statements should be only device force and measure conditions plus operations on site-aware variables.

Switch Statement

The switch statement selects a code block based on the value of an integer parameter. This can replace a lenghty if/else sequence of statements for better readability.

Syntax

```
switch (Expression)
{
  case Constant-1:
    Statement sequence case-1;
    break;
  case Constant-2:
    Statement sequence case-2;
    break;
  …
  case Constant-n:
    Statement sequence case-n;
    break;
  [default:
    Statement sequence default;
    break;]
}
```

Example 1

```
switch (ivar)
{
  case 0:
  // code executed only when ivar == 0
  break;

  case 43:
  // code executed only when ivar == 43
  break;

  default:
  // code executed when ivar is none of the cases
  break;
}
```

Example 2

```
enum eStartModeChip
{
  WorkingMode
  SleepMode
  StandbyMode
  VDD5S_supply_only
};

eStartModeChip SModeChip;
int iValue;

switch (SModeChip)
{
  case eStartModeChip.WorkingMode:
  iValue = 3;
  break;
  case eStartModeChip.StandbyMode:
  iValue = 4;
  break;

  case eStartModeChip.SleepMode:
  iValue = 5;
  break;
};
```

For Statement

The for statement is most general purpose and takes three parameters: a statement to be executed before entering the loop, a condition specifying when looping continues, and a statement to be executed at the end of each iteration.

Syntax

```
for (Initialize; Condition; Update)
{
   Statement sequence;
}
```

Example

```
int i;
PinList plst;
ValueList result;
bool bAlreadyZapped;

plst = Pin1 + Pin2 + Pin3;
result = Pins(plst).Voltage.Meter.Read();

for (i=0; i < plst.Length; i++)
{
   if (result.GetDataN(i) > 2.5V)
   {
      bAlreadyZapped = true;
   }
}
```

While Statement

The while statement is a simple way to iterate on a single boolean condition.

Syntax

```
while (Condition)
{
   Statement sequence;
   [break;]
   Statement sequence;
}
```

Example

```
bool bLoopCond = true;
ValueList vlReading;

while (bLoopCond)
{
   Pins(Pin1).Voltage.Value = 1.0;
   vlReading = Pins(Pin2).Voltage.Meter.Read();
   if (vlReading > 0.5V)
   {
      bLoopCond = false;
   }
}
```

Procedure

Subprograms with parameters or optional parameters with or without return values.

Syntax

```
[Optional (Parameter-1 = Value-1, Parameter-2 = Value 2, ..., Parameter-m = Value-m)]
[public | private] {void | DataType} ProcedureName ([DataType  Parameter-1,
DataType Parameter-2, …, DataType Parameter-n])
{
[DataType VarName;]

   [Statement sequence];
   [return ReturnValue;]
   [Statement sequence];
}
```

Example void

```
public void regLowSupply(){
  Pins(VBP).Voltage.Force(8V, 200mA);
}
```

Example with parameters

```
public void regLowSupply(Pin PinName, double dVoltage){
  Pins(PinName).Voltage.Force(dVoltage, 200mA);
}
```

Example with optional parameters

```
[Optional(dIClamp = 200mA)]
public void regLowSupply(Pin PinName, double dVoltage, double
 dIClamp){
  Pins(PinName).Voltage.Force(dVoltage, dIClamp);
}
```

Example with return value

```
public bool IsEqual(int i, int j)
{
  if (i == j) {
    return true;
  }
  else {
    return false;
  }
}
```

Example with local variables

```
private int sum(int iValue1, int iValue2)
{
  int iSum;

  iSum = iValue1 + iValue2;

  return iSum;
}
```

Build-ins

UTSL functions and properties

Pins.

Classes for programming all types of pin stimulus, pin measurements and instrument parameters.

Pins.Connect

Connects the indicated instrument connection if not already connected.Connects only the primary instrument if it is XSet.

Syntax

Pins(Pins).Connect ([Type] [, InstType] [, DoDutConnect] [, ConnectVoltage])

Parameters

Name	Description	Data Type	Optional	Default Value
Pins	List of Pins	PinList	no	-
Type	The type of connection to make	*ConnectType*	yes	Default
InstType	The type of instrument to connect if the Tester Resource Map setup has more than one.	*InstrumentType*	yes	Default
DoDutConnect	If true, also set DUT connect relays for this connection, otherwise they are left unchanged.	bool	yes	true
ConnectVoltage	The voltage value based on connect types.	double	yes	-

Details

Type	Function
Spec relevant	no

Related Links

ConnectType. on page 118

The possible connection types for connect and disconnect. ATE instruments often have three-wire Kelvin connections and this enum provides explicit control over physical connections. The choice values can be combined with the OR operator to select more than one at once.

InstrumentType. on page 120
The tester resource map allows using more than one type of instrument within a single test setup. For these cases, this enum selects a single instrument when needed, such as when choosing which instrument to connect with the Connect() function.

Pins.ConnectAll

Connects all the instruments from the XSetSheet.

Syntax

Pins(Pins).ConnectAll()

Parameters

Name	Description	Data Type	Optional	Default Value
Pins	List of Pins	PinList	no	-

Details

Type	Function
Spec relevant	no

Pins.ConnectDib

Connects the secondary instruments from the XSetSheet.

Syntax

Pins(Pins).ConnectDib()

Parameters

Name	Description	Data Type	Optional	Default Value
Pins	List of Pins	PinList	no	-

Details

Type	Function
Spec relevant	no

Pins.Disconnect

Disconnects the indicated instrument connection if not already disconnected. Disconnects only the primary instrument if it is XSet.

Syntax

Pins(Pins).Disconnect ([Type] [, InstType] [, DoDutConnect])

Parameters

Name	Description	Data Type	Optional	Default Value
Pins	List of Pins	PinList	no	-
Type	The type of connection to disconnect.	*ConnectType*	yes	Default
InstType	The type of instrument to disconnect if the Tester Resource Map setup has more than one.	*InstrumentType*	yes	Default
DoDutConnect	If true, also set DUT connect relays for this disconnection, otherwise they are left unchanged.	bool	yes	true

Details

Type	Function
Spec relevant	no

Related Links

ConnectType. on page 118
The possible connection types for connect and disconnect. ATE instruments often have three-wire Kelvin connections and this enum provides explicit control over physical connections. The choice values can be combined with the OR operator to select more than one at once.

InstrumentType. on page 120
The tester resource map allows using more than one type of instrument within a single test setup. For these cases, this enum selects a single instrument when needed, such as when choosing which instrument to connect with the Connect() function.

Pins.DisconnectAll

Disconnects all the instruments from the XSetSheet.

Syntax

Pins(Pins).DisconnectAll()

Parameters

Name	Description	Data Type	Optional	Default Value
Pins	List of Pins	PinList	no	-

Details

Type	Function
Spec relevant	no

Pins.DisconnectDib

Disconnects the secondary instruments from the XSetSheet.

Syntax

Pins(Pins).DisconnectDib()

Parameters

Name	Description	Data Type	Optional	Default Value
Pins	List of Pins	PinList	no	-

Details

Type	Function
Spec relevant	no

Pins.Gate

If true, the instrument gate is on to provide the signal to the pin. If false, the gate is off. The electrical pin behaviour in the off state is tester-dependent.

Syntax

Pins(Pins).Gate = bool

Parameters

Name	Description	Data Type	Optional	Default Value
Pins	List of Pins	PinList	no	-

Property Values

Value	Description
true	Gate on
false	Gate off

Details

Type	Property
Access	Write-only
Spec relevant	no

Pins.Current.

Per-pin current stimulus and measurement.

Pins.Current.Force

Specifies a current forcing value with optional voltage clamp and range arguments. Puts the instrument in current forcing mode.

Syntax

Pins(Pins).Current.Force (I [, VClamp] [, IRange] [, VRange] [, VClamp2])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
I	Current value to force	double	no	-	yes
VClamp	Voltage clamp value	double	yes	No change from previous setting	yes
IRange	Instrument current range	double	yes	No change from previous setting	no
VRange	Instrument current range for VClamp and VClamp2	double	yes	No change from previous setting	no
IClamp2	Second voltage clamp value. VClamp and VClamp2 can be used to separately set positive and negative current clamps (subject to physical instrument capability).	double	yes	No change from previous setting	yes

Details

Type	Function
Spec relevant	yes

Pins.Current.ForceHIZ

This property sets the instrument into high impedance mode for zero current voltage metering. This command will automatically gate Off the instrument and disconnect the force line.

Syntax

Pins(Pins).Current.ForceHiZ = bool

Parameters

Name	Description	Data Type	Optional	Default Value
Pins	List of Pins	PinList	no	-

Property Values

Value	Description
true	Set to high impedance mode

Details

Type	Property
Access	?
Spec relevant	no

Pins.Current.Range

The current forcing instrument range.

Syntax

Pins(Pins).Current.Range = double

Parameters

Name	Description	Data Type	Optional	Default Value
Pins	List of Pins	PinList	no	-

Property Values

Value	Description
Current	Range

Details

Type	Property
Access	Write-only
Spec relevant	no

Pins.Current.Value

The site-aware current forcing value. This is used to modify values based on measured device values which at runtime may be different for different sites. All other instrument settings will not be changed.

Syntax

Pins(Pins).Current.Value = SiteDouble

Parameters

Name	Description	Data Type	Optional	Default Value
Pins	List of Pins	PinList	no	-

Property Values

Value	Description
Current	Forcing value

Details

Type	Property
Access	Write-only
Spec relevant	yes

Pins.Current.Meter.

Current measurement functions.

Pins.Current.Meter.GetSample

Retrieves a number of samples equal to the sample size from memory. Mostly, used to get samples from strobes in a pattern.

Syntax

ValueList = Pins(Pins).Current.Meter.GetSample ([SampleSize] [, DataFormat] [, StoreLocation] [, OfflineValue])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
SampleSize	Number of measurements to average. When SampleSize is > 1, the return value is the average of the samples.	int	yes	1	no

Name	Description	Data Type	Optional	Default Value	Spec relevant
DataFormat	Format in which samples are returned	*MeasureReadFormat*	yes	Average	no
SettlingTime	Setting time	double	yes	0	no
StoreLocation	Specifies from which location to read back the stored result	int	yes	0	no
OfflineValue	Assign an offline value for program running in offline	double	yes	-9999.0	no

Return Value

Description	Data Type
Current value for each pin	ValueList

Details

Type	Function
Spec relevant	yes

Related Links

MeasureReadFormat. on page 121
Declares how the measure read format should be.

Pins.Current.Meter.Read

Reads the current value for each pin using the meter instrument.

Syntax

ValueList = Pins(Pins).Current.Meter.Read ([Range [, SampleSize] [, SampleRate] [, DataFormat] [, SettlingTime] [, StoreLocation] [, OfflineValue])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
Range	Instrument metering range	double	yes	No change from previous setting	no
SampleSize	Number of measurements to average. When SampleSize is > 1, the return value is the	int	yes	1	no

Name	Description	Data Type	Optional	Default Value	Spec relevant
	average of the samples.				
SampleRate	Frequency at which to make measurements when SampleSize is > 1	double	yes	The maximum rate of the physical instrument	no
DataFormat	Format in which samples are returned	*MeasureReadFormat*	yes	Average	no
SettlingTime	Setting time	double	yes	0	no
StoreLocation	Specifies from which location to read back the stored result	int	yes	0	no
OfflineValue	Assign an offline value for program running in offline mode	double	yes	-9999.0	no

Return Value

Description	Data Type
Current value for each pin	ValueList

Details

Type	Function
Spec relevant	yes

Related Links

MeasureReadFormat. on page 121
Declares how the measure read format should be.

Pins.Current.Meter.Range

The current metering instrument range.

Syntax

Pins(Pins).Current.Meter.Range = double

Parameters

Name	Description	Data Type	Optional	Default Value
Pins	List of Pins	PinList	no	-

Property Values

Value	Description
Current	Range

Details

Type	Property
Access	Write-only
Spec relevant	no

Pins.Current.Threshold.

Threshold measurement functions.

Pins.Current.Threshold.Reset

Provides option to reset the threshold test.

Syntax

Pins(Pins).Current.Threshold.Reset (TriggerPin)

Parameters

Name	Description	Data Type	Optional	Default Value
Pins	List of Pins	PinList	no	-
TriggerPin	Name of the trigger pin that is used for the measurement on the meter pin in Pins() pinlist	PinList	no	-

Details

Type	Function
Spec relevant	no

Pins.Current.Threshold.RunAndRead

Makes a threshold measurement by running a previously defined ramp and looking for a previously defined trigger. The Pins() pin is the pin to measure.

Syntax

ValueList = Pins(Pins).Current.Threshold.RunAndRead (RampName [, MeterRange] [, RampPin] [, RampToEnd] [, OfflineValue])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
RampName	Name of the ramp	string	no	-	yes

Name	Description	Data Type	Optional	Default Value	Spec relevant
	defined in SetupRamp()				
MeterRange	Instrument current metering range	double	yes	No change from previous setting	yes
RampPin	Pin on which to source the ramp, which is allowed to be different from the pin being measured.	PinList	yes	Pins being measured	yes
RampToEnd	If true, force the ramp to run to completion after triggering. If false, halt the ramp after triggering.	bool	yes	false	yes
OfflineValue	Assign an offline value for program running in offline mode	double	yes	-9999.0	yes

Return Value

Description	Data Type
Reading measured on the pin(s)	ValueList

Details

Type	Function
Spec relevant	yes

Pins.Current.Threshold.SetupRamp

Creates the specified current ramp for a threshold measurement. The ramp will be sourced on the Pins() pins.

Syntax

Pins(Pins).Current.Threshold.SetupRamp (RampName , RampStartI , RampEndI , RampSize [, RampFreq] [, Bandwidth])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
RampName	The name for referencing	string	no	-	yes

Name	Description	Data Type	Optional	Default Value	Spec relevant
	this ramp in other functions.				
RampStartI	Starting current	string	no	-	yes
RampEndI	Ending current	string	no	-	yes
RampSize	Number of equal-sized steps in the ramp.	int	no	-	yes
RampFreq	Frequency of the ramp (if it were periodic between RampStartI and RampEndI)	double	yes	1 kHz	yes
Bandwidth	Bandwidth of the ramp to be used for controlling physical instrument parameters	double	yes	5000 Hz	yes

Details

Type	Function
Spec relevant	yes

Pins.Current.Threshold.SetupTrigger

Sets up a trigger pin for use in a threshold measurement. The Pins() pin is the trigger pin for a measurement on.

Syntax

Pins(Pins).Current.Threshold.SetupTrigger (MeterPin , TriggerLevel , TriggerSlope [, TriggerRange])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
MeterPin	Pin that will be measured	PinList	no	-	yes
TriggerLevel	Current at which to trigger	double	no	-	yes
TriggerSlope	Positive or negative slope of the trigger	*SignalSlope*	no	-	yes

Name	Description	Data Type	Optional	Default Value	Spec relevant
TriggerRange	Range of the trigger current hardware	double	yes	No change from previous setting	yes

Details

Type	Function
Spec relevant	yes

Related Links

SignalSlope. on page 122
The positive or negative expected slope of a signal.

Pins.Digital.

Per-pin digital data control.

The global Digital class controls digital patterns, which affect all digital pins of the device together. Digital patterns are referred to by a name which is passed through to the tester-specific program; UTSL does not include the definition of digital pattern data (because there are well-developed third-party tools for translating digital patterns from design tools directly to tester-specific formats).

This class also includes control of digital levels and timing setups through the ApplyLevels, ApplyTiming, and ApplyLevelsTiming functions. These are used once per test step to specify the setups to be used for running patterns. They may also be used in global setup functions.

Pins.Digital.PinFailCount

Number of failures found on each pin in the Pins() during the last pattern run.

Syntax

ValueList = Pins(Pins).Digital.PinFailCount ([OfflineValue])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
OfflineValue	Assign an offline value for program running in offline mode.	double	yes	0	no

Return Value

Description	Data Type
Number off failures found	ValueList

Details

Type	Function
Spec relevant	yes

Pins.Digital.SetLevel

Reprograms the selected digital level.

Syntax

Pins(Pins).Digital.SetLevel (WhichLevel , Value)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
WhichLevel	Selects the level to program	*DigitalLevel*	no	-	yes
Value	The new level value	double	no	-	yes

Details

Type	Function
Spec relevant	yes

Related Links

DigitalLevel. on page 118
Choice of different types of digital levels.

Pins.Digital.SetState

Sets the digital pin state immediately and/or at future pattern starts.

Syntax

Pins(Pins).Digital.SetState (State , When)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
State	The pin state to apply	*DigitalState*	no	-	yes
When	When the new state will be applied	*DigitalStateWhen*	yes	ImmedateAndP	no

Details

Type	Function
Spec relevant	yes

Related Links

DigitalState. on page 119

The drive state of a digital pin.

DigitalStateWhen. on page 119
Whether a digital pin DigitalState is to be forced immediately, at pattern start, or both.

Pins.Signals.

Per-pin signal shape definition and control.

Pins.Signals.Source

Functions for source signals.
Pins.Signals.Source.Clock
Starts a clock signal at the specified frequency and levels.

Syntax

Pins(Pins).Signals.Source.Clock (SignalName , Frequency , LowLevel , HighLevel)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
SignalName	A name assigned to the signal so it can be started and stopped later without respecifying the other parameters.	string	no	-	no
Frequency	The frequency of the clock signal in Hz	double	no	-	yes
LowLevel	The lower voltage level of the signal	double	no	-	yes
HighLevel	The higher voltage level of the signal	double	no	-	yes

Details

Type	Function
Spec relevant	yes

Pins.TesterSettings.

Per-pin settings that affect different tester instrument setup.

Pins.TesterSettings.AlarmClear

Clears alarms for instruments connected to the pins so that any previous alarm conditions will not cause failures. This avoids reporting alarms due to expected events such as instrument setup, device settling or transition, or DIB switching while alarms are turned on.

Syntax

Pins(Pins).TesterSettings.AlarmClear ([InstType])

Parameters

Name	Description	Data Type	Optional	Default Value
Pins	List of Pins	PinList	no	-
InstType	The type of instrument	*InstrumentType*	yes	Default

Details

Type	Function
Spec relevant	no

Related Links

InstrumentType. on page 120

The tester resource map allows using more than one type of instrument within a single test setup. For these cases, this enum selects a single instrument when needed, such as when choosing which instrument to connect with the Connect() function.

Pins.TesterSettings.AlarmOn

Turns on the specified alarm, which provides real-time error checking for setup errors on the pin's instrument.

Syntax

Pins(Pins).TesterSettings.AlarmOn (Type [, InstType])

Parameters

Name	Description	Data Type	Optional	Default Value
Pins	List of Pins	PinList	no	-
Type	Alarm type	*AlarmType*	no	-
InstType	The type of instrument to connect if the Tester Resource Map setup has more than one.	*InstrumentType*	yes	DC

Details

Type	Function
Spec relevant	no

Related Links

InstrumentType. on page 120
The tester resource map allows using more than one type of instrument within a single test setup. For these cases, this enum selects a single instrument when needed, such as when choosing which instrument to connect with the Connect() function.

AlarmType. on page 116
The alarm choices for controlling alarms in TesterSettings.

Pins.TesterSettings.AlarmOff

Turns off the specified alarm. This masks any runtime errors that the alarm might cause, which is sometimes necessary during temporary conditions like switching instrument connections.

Syntax

Pins(Pins).TesterSettings.AlarmOff (Type [, InstType])

Parameters

Name	Description	Data Type	Optional	Default Value
Pins	List of Pins	PinList	no	-
Type	Alarm type	*AlarmType*	no	-
InstType	The type of instrument to connect if the Tester Resource Map setup has more than one.	*InstrumentType*	yes	Default

Details

Type	Function
Spec relevant	no

Related Links

InstrumentType. on page 120
The tester resource map allows using more than one type of instrument within a single test setup. For these cases, this enum selects a single instrument when needed, such as when choosing which instrument to connect with the Connect() function.

AlarmType. on page 116

The alarm choices for controlling alarms in TesterSettings.

Pins.TesterSettings.ComplianceSettleWait

Ensures that an instrument is ready for use after a ComplianceRangeNegative or CompliancePositive change.

Syntax

Pins(Pins).TesterSettings.ComplianceSettleWait()

Parameters

Name	Description	Data Type	Optional	Default Value
Pins	List of Pins	PinList	no	-

Details

Type	Function
Spec relevant	no

Pins.TesterSettings.ComplianceRangeNegative

The suggested instrument negative compliance range value. This may affect performance of some instruments. After changing this value, ComplianceSettleWait() should be called prior to using the instrument.

Syntax

Pins(Pins).TesterSettings.ComplianceRangeNegative = double

Parameters

Name	Description	Data Type	Optional	Default Value
Pins	List of Pins	PinList	no	-

Property Values

Value	Description
?	Negative compliance range value

Details

Type	Property
Access	?
Spec relevant	no

Pins.TesterSettings.ComplianceRangePositive

The suggested instrument positive compliance range value. This may affect performance of some instruments. After changing this value, ComplianceSettleWait() should be called prior to using the instrument.

Syntax

Pins(Pins).TesterSettings.ComplianceRangePositive = double

Parameters

Name	Description	Data Type	Optional	Default Value
Pins	List of Pins	PinList	no	-

Property Values

Value	Description
?	Positive compliance range value

Details

Type	Property
Access	?
Spec relevant	no

Pins.TesterSettings.Bandwidth

Tester setting parameters related to signal path bandwidth. These assist the tester code generator in determining the details of instrument setup.

Pins.TesterSettings.Bandwidth.Value

The expected numerical bandwidth of the pin's signal path. Affects the setting of AC signal source and capture instruments.

Syntax

Pins(Pins).TesterSettings.Bandwidth.Value (Value [, InstType])

Parameters

Name	Description	Data Type	Optional	Default Value
Pins	List of Pins	PinList	no	-
Value	?	double	no	-
InstType	The type of instrument to connect if the Tester Resource Map setup has more than one.	*InstrumentType*	yes	DC

Details

Type	Function

Spec relevant	no

Related Links

InstrumentType. on page 120
The tester resource map allows using more than one type of instrument within a single test setup. For these cases, this enum selects a single instrument when needed, such as when choosing which instrument to connect with the Connect() function.

Pins.TesterSettings.Bandwidth.Range

An approximate range of bandwidth, when specifying a precise number is not necessary. This is used most commonly to control the response time of DC instruments.

Syntax

Pins(Pins).TesterSettings.Bandwidth.Range (Range [, InstType])

Parameters

Name	**Description**	**Data Type**	**Optional**	**Default Value**
Pins	List of Pins	PinList	no	-
Range	?	BandwidthRange	no	-
InstType	The type of instrument to connect if the Tester Resource Map setup has more than one.	*InstrumentType*	yes	DC

Details

Type	Function
Spec relevant	no

Related Links

BandwidthRange. on page 117
The bandwidth choices for Pins.TesterSettings.Bandwidth.Range. The numerical meanings of each choice are instrument-dependent.

InstrumentType. on page 120
The tester resource map allows using more than one type of instrument within a single test setup. For these cases, this enum selects a single instrument when needed, such as when choosing which instrument to connect with the Connect() function.

Pins.Time.

Per-pin time measurement.

Pins.Time.ReadTimeOut

Time to wait for a valid reading in a .Read function before aborting and raising an error.

Syntax

Pins(Pins).Time.ReadTimeOut = double

Parameters

Name	Description	Data Type	Optional	Default Value
Pins	List of Pins	PinList	no	-

Property Values

Value	Description
?	Time to wait

Details

Type	Property
Access	?
Spec relevant	no

Pins.Time.DutyCycle.

Duty cycle measurement functions.

Pins.Time.DutyCycle.Read

Returns the duty cycle measurement result.

Syntax

ValueList = Pins(Pins).Time.DutyCycle.Read (TriggerName [, OfflineValue])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
TriggerName	Name of the trigger to read	string	no	-	yes
OfflineValue	Assign an offline value for program running in offline	double	yes	-9999.0	no

Return Value

Description	Data Type
?	ValueList

Details

Type	Function
Spec relevant	yes

Pins.Time.DutyCycle.Setup

Sets up a duty cycle measurement.

Syntax

Pins(Pins).Time.DutyCycle.Setup (TriggerLevel , TriggerRange [, SampleSize] [, Slope] [, Hysteresis] [, InputImpedance])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
TriggerLevel	Pin trigger level	double	no	-	yes
TriggerRange	Pin trigger range	double	no	-	no
SampleSize	Number of measurements to average	int	yes	1	no
Slope	Expected signal slope	*SignalSlope*	Positive	-	no
Hysteresis	Instrument hardware hysteresis optimization to apply	*TimeHysteresis*	yes	Off	no
InputImpedance	Impedance of the input path	*TimeImpedance*	yes	IHighZ	no

Details

Type	Function
Spec relevant	yes

Related Links

SignalSlope. on page 122
The positive or negative expected slope of a signal.

TimeHysteresis. on page 123
Controls windowing of the trigger threshold of a time measurement to prevent "chatter" caused by crossing the threshold multiple times.

TimeImpedance. on page 124
Declares the expected input signal impedance for a time measurement.

Pins.Time.DutyCycle.Trigger

Triggers the duty cycle measurement.

Syntax

Pins(Pins).Time.DutyCycle.Trigger (TriggerName)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
TriggerName	User-defined name which identifies the trigger when calling .Read()	string	no	-	yes

Details

Type	Function
Spec relevant	yes

Pins.Time.FallTime.

Fall time measurement functions.

Pins.Time.FallTime.Read

Returns the fall time measurement result.

Syntax

ValueList = Pins(Pins).Time.FallTime.Read (TriggerName [, OfflineValue])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
TriggerName	Name of the trigger to read	string	no	-	yes
OfflineValue	Assign an offline value for program running in offline	double	yes	-9999.0	no

Return Value

Description	Data Type
?	ValueList

Details

Type	Function
Spec relevant	yes

Pins.Time.FallTime.Setup

Sets up a fall time measurement.

Syntax

Pins(Pins).Time.FallTime.Setup (Threshold1 , Threshold2 , ThresholdRange [, SampleSize] [, Hysteresis] [, InputImpedance])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
Threshold1	Voltage threshold which starts the measurement	double	no	-	yes
Threshold2	Voltage threshold which stops the measurement	double	no	-	no
ThresholdRange	Pin threshold voltage range	double	no	-	no
SampleSize	Number of measurements to average	int	yes	1	no
Hysteresis	Instrument hardware hysteresis optimization to apply	*TimeHysteresis*	yes	Off	no
InputImpedance	Impedance of the input path	*TimeImpedance*	yes	IHighZ	no

Details

Type	Function
Spec relevant	yes

Related Links

TimeHysteresis. on page 123
Controls windowing of the trigger threshold of a time measurement to prevent "chatter" caused by crossing the threshold multiple times.

TimeImpedance. on page 124
Declares the expected input signal impedance for a time measurement.

TimeHysteresis. on page 123
Controls windowing of the trigger threshold of a time measurement to prevent "chatter" caused by crossing the threshold multiple times.

TimeImpedance. on page 124

Declares the expected input signal impedance for a time measurement.

Pins.Time.FallTime.Trigger
Triggers the fall time measurement.

Syntax
Pins(Pins).Time.FallTime.Trigger (TriggerName)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
TriggerName	User-defined name which identifies the trigger when calling .Read()	string	no	-	yes

Details

Type	Function
Spec relevant	yes

Pins.Time.Frequency.

Frequency measurement functions.
Pins.Time.Frequency.Read
Returns the frequency measurement result.

Syntax
ValueList = Pins(Pins).Time.Frequency.Read (TriggerName [, OfflineValue])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
TriggerName	Name of the trigger to read	string	no	-	yes
OfflineValue	Assign an offline value for program running in offline	double	yes	-9999.0	no

Return Value

Description	Data Type
?	ValueList

Details

Type	Function
Spec relevant	yes

Pins.Time.Frequency.Setup

Sets up a frequency measurement.

Syntax

Pins(Pins).Time.Frequency.Setup (TriggerLevel , TriggerRange [, SampleSize] [, Slope] [, Hysteresis] [, InputImpedance])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
TriggerLevel	Pin trigger level	double	no	-	yes
TriggerRange	Pin trigger range	double	no	-	no
SampleSize	Number of measurements to average	int	yes	1	no
Slope	Expected signal slope	*SignalSlope*	Positive	-	no
Hysteresis	Instrument hardware hysteresis optimization to apply	*TimeHysteresis*	yes	Off	no
InputImpedance	Impedance of the input path	*TimeImpedance*	yes	IHighZ	no

Details

Type	Function
Spec relevant	yes

Related Links

SignalSlope. on page 122
The positive or negative expected slope of a signal.

TimeHysteresis. on page 123
Controls windowing of the trigger threshold of a time measurement to prevent "chatter" caused by crossing the threshold multiple times.

TimeImpedance. on page 124

Declares the expected input signal impedance for a time measurement.

Pins.Time.Frequency.Trigger
Triggers the frequency measurement.

Syntax
Pins(Pins).Time.Frequency.Trigger (TriggerName)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
TriggerName	User-defined name which identifies the trigger when calling .Read()	string	no	-	yes

Details

Type	Function
Spec relevant	yes

Pins.Time.Period.

Period measurement functions.
Pins.Time.Period.Read
Returns the period measurement result.

Syntax
ValueList = Pins(Pins).Time.Period.Read (TriggerName [, OfflineValue])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
TriggerName	Name of the trigger to read	string	no	-	yes
OfflineValue	Assign an offline value for program running in offline	double	yes	-9999.0	no

Return Value

Description	Data Type
?	ValueList

Details

Type	Function
Spec relevant	yes

Pins.Time.Period.Setup

Sets up a period measurement.

Syntax

Pins(Pins).Time.Period.Setup (TriggerLevel , TriggerRange [, SampleSize] [, Slope] [, Hysteresis] [, InputImpedance])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
TriggerLevel	Pin trigger level	double	no	-	yes
TriggerRange	Pin trigger range	double	no	-	no
SampleSize	Number of measurements to average	int	yes	1	no
Slope	Expected signal slope	*SignalSlope*	Positive	-	no
Hysteresis	Instrument hardware hysteresis optimization to apply	*TimeHysteresis*	yes	Off	no
InputImpedance	Impedance of the input path	*TimeImpedance*	yes	IHighZ	no

Details

Type	Function
Spec relevant	yes

Related Links

SignalSlope. on page 122
The positive or negative expected slope of a signal.

TimeHysteresis. on page 123
Controls windowing of the trigger threshold of a time measurement to prevent "chatter" caused by crossing the threshold multiple times.

TimeImpedance. on page 124

Declares the expected input signal impedance for a time measurement.

Pins.Time.Period.Trigger
Triggers the period measurement.

Syntax

Pins(Pins).Time.Period.Trigger (TriggerName)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
TriggerName	User-defined name which identifies the trigger when calling .Read()	string	no	-	yes

Details

Type	Function
Spec relevant	yes

Pins.Time.PropDelay.

Propagation delay measurement functions.

Pins.Time.PropDelay.Read
Returns the propagation delay measurement result.

Syntax

ValueList = Pins(Pins).Time.PropDelay.Read (OutPins , TriggerName [, OfflineValue])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	-
OutPins	Out pins	PinList	no	-	yes
TriggerName	Name of the trigger to read	string	no	-	yes
OfflineValue	Assign an offline value for program running in offline	double	yes	-9999.0	no

Return Value

Description	Data Type
?	ValueList

Details

Type	Function
Spec relevant	yes

Pins.Time.PropDelay.Setup

Sets up a propagation delay measurement between two pins. The Pins() parameter is the delay input pin which starts the measurement while the OutPin parameter is the output pin which stops it.

Syntax

Pins(Pins).Time.PropDelay.Setup (OutPin , TriggerLevelIn , TriggerLevelOut , TriggerRangeIn , TriggerRangeOut [, SampleSize] [, SlopeIn] [, SlopeOut] [, HysteresisIn] [, HysteresisOut] [, InputImpedanceIn] [, InputImpedanceOut])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
TriggerLevelIn	Trigger level which starts the measurement	double	no	-	yes
TriggerLevelOut	Trigger level which stops the measurement.	double	no	-	yes
TriggerRangeIn	Voltage range for TriggerLevelIn.	double	no	-	no
TriggerRangeOut	Voltage range for TriggerLevelOut	double	no	-	no
SampleSize	Number of measurements to average	int	yes	1	no
SlopeIn	Expected signal slope of the start pin	*SignalSlope*	yes	Positive	yes
SlopeOut	Expected signal slope of the stop pin	*SignalSlope*	yes	Positive	yes
HysteresisIn	Instrument hardware hysteresis optimization to apply on the start pin	*TimeHysteresis*	yes	Off	no
HysteresisOut	Instrument hardware hysteresis optimization	*TimeHysteresis*	yes	Off	no

Name	Description	Data Type	Optional	Default Value	Spec relevant
	to apply on the stop pin				
InputImpedanceIn	Impedance of the input pin path	*TimeImpedance*	yes	IHighZ	no
InputImpedanceOut	Impedance of the output pin path	*TimeImpedance*	yes	IHighZ	no

Details

Type	Function
Spec relevant	yes

Related Links

SignalSlope. on page 122
The positive or negative expected slope of a signal.

TimeHysteresis. on page 123
Controls windowing of the trigger threshold of a time measurement to prevent "chatter" caused by crossing the threshold multiple times.

TimeImpedance. on page 124
Declares the expected input signal impedance for a time measurement.

Pins.Time.PropDelay.Trigger

Triggers the propagation delay measurement.

Syntax

Pins(Pins).Time.PropDelay.Trigger (TriggerName)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
TriggerName	User-defined name which identifies the trigger when calling .Read()	string	no	-	yes

Details

Type	Function
Spec relevant	yes

Pins.Time.PulseWidth

Pulsewidth measurement functions.

Pins.Time.PulseWidth.Read

Returns the pulsewidth measurement result.

Syntax

ValueList = Pins(Pins).Time.PulseWidth.Read (TriggerName [, OfflineValue])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
TriggerName	Name of the trigger to read	string	no	-	yes
OfflineValue	Assign an offline value for program running in offline	double	yes	-9999.0	no

Return Value

Description	Data Type
?	ValueList

Details

Type	Function
Spec relevant	yes

Pins.Time.PulseWidth.Setup

Sets up a pulsewidth measurement.

Syntax

Pins(Pins).Time.PulseWidth.Setup (TriggerLevel , TriggerRange [, SampleSize] [, Slope] [, Hysteresis] [, InputImpedance])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
TriggerLevel	Pin trigger level	double	no	-	yes
TriggerRange	Pin trigger range	double	no	-	no
SampleSize	Number of measurements to average	int	yes	1	no

Name	Description	Data Type	Optional	Default Value	Spec relevant
Slope	Expected signal slope	*SignalSlope*	Positive	-	no
Hysteresis	Instrument hardware hysteresis optimization to apply	*TimeHysteresis*	yes	Off	no
InputImpedance	Impedance of the input path	*TimeImpedance*	yes	IHighZ	no

Details

Type	Function
Spec relevant	yes

Related Links

SignalSlope. on page 122
The positive or negative expected slope of a signal.

TimeHysteresis. on page 123
Controls windowing of the trigger threshold of a time measurement to prevent "chatter" caused by crossing the threshold multiple times.

TimeImpedance. on page 124
Declares the expected input signal impedance for a time measurement.

Pins.Time.PulseWidth.Trigger

Triggers the pulsewidth measurement.

Syntax

Pins(Pins).Time.PulseWidth.Trigger (TriggerName)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
TriggerName	User-defined name which identifies the trigger when calling .Read()	string	no	-	yes

Details

Type	Function
Spec relevant	yes

Pins.Time.RiseTime

Rise time measurement functions.

Pins.Time.RiseTime.Read

Returns the rise time measurement result.

Syntax

ValueList = Pins(Pins).Time.RiseTime.Read (TriggerName [, OfflineValue])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
TriggerName	Name of the trigger to read	string	no	-	yes
OfflineValue	Assign an offline value for program running in offline	double	yes	-9999.0	no

Return Value

Description	Data Type
?	ValueList

Details

Type	Function
Spec relevant	yes

Pins.Time.RiseTime.Setup

Sets up a rise time measurement.

Syntax

Pins(Pins).Time.RiseTime.Setup (Threshold1 , Threshold2 , ThresholdRange [, SampleSize] [, Hysteresis] [, InputImpedance])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
Threshold1	Voltage threshold which starts the measurement	double	no	-	yes
Threshold2	Voltage threshold which	double	no	-	no

Name	Description	Data Type	Optional	Default Value	Spec relevant
	stops the measurement				
ThresholdRange	Pin threshold voltage range	double	no	-	no
SampleSize	Number of measurements to average	int	yes	1	no
Hysteresis	Instrument hardware hysteresis optimization to apply	TimeHysteresis	yes	Off	no
InputImpedance	Impedance of the input path	TimeImpedance	yes	IHighZ	no

Details

Type	Function
Spec relevant	yes

Pins.Time.RiseTime.Trigger

Triggers the rise time measurement.

Syntax

Pins(Pins).Time.RiseTime.Trigger (TriggerName)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
TriggerName	User-defined name which identifies the trigger when calling .Read()	string	no	-	yes

Details

Type	Function
Spec relevant	yes

Pins.Voltage.

Per-pin voltage stimulus and measurement.

Pins.Voltage.CompareLevel

Measures a pin voltage and converts to a digital value relative to the VoltageLow and VoltageHigh arguments. Returns integer values of 0 for below VoltageLow, 1 for above VoltageHigh, and 2 for between the values (midband). ForceCurrent specifies a load during the measurement.

Syntax

ValueList = Pins(Pins).Voltage.CompareLevel (VoltageLow , VoltageHigh , ForceCurrent [, VoltageClamp] [, CurrentRange] [, VoltageRange] [, SettlingTime] [, MeasureRange] [, OfflineValue])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
VoltageLow	High compare level	double	no	-	yes
VoltageHigh	Low compare level	double	no	-	yes
ForceCurrent	Load to apply prior to measurement	double	no	-	yes
VoltageClamp	Voltage clamp for ForceCurrent	double	yes	No change from last current force setting on this pin.	no
CurrentRange	Range for ForceCurrent	double	yes	No change from last current force setting on this pin.	no
VoltageRange	Range for VoltageClamp	double	yes	No change from last current force setting on this pin.	no
SettlingTime	Time to wait between applying ForceCurrent and reading the voltage.	double	yes	0.0s	no
MeasureRange	Range for VoltageLow and VoltageHigh	double	yes	No change from last voltage meter setting on this pin.	no

Name	Description	Data Type	Optional	Default Value	Spec relevant
OfflineValue	Assigne an offline value for program running in offline mode.	int	yes	0	no

Return Value

Description	Data Type
Integer values of 0 for below VoltageLow, 1 for above VoltageHigh, and 2 for between the values (midband).	ValueList

Details

Type	Function
Spec relevant	yes

Pins.Voltage.CompareLevelHiZ

Measures a pin voltage and converts to a digital value relative to the VoltageLow and VoltageHigh arguments. Returns integer values of 0 for below VoltageLow, 1 for above VoltageHigh, and 2 for between the values (midband). The measurement is taken with no load (Hi-Z).

Syntax

ValueList = Pins(Pins).Voltage.CompareLevelHiZ (VoltageLow , VoltageHigh [, SettlingTime] [, MeasureRange] [, OfflineValue])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
VoltageLow	High compare level	double	no	-	yes
VoltageHigh	Low compare level	double	no	-	yes
SettlingTime	Time to wait between applying ForceCurrent and reading the voltage.	double	yes	0.0s	no
MeasureRange	Range for VoltageLow and VoltageHigh	double	yes	No change from last voltage meter setting on this pin.	no
OfflineValue	Assigne an offline value for program	int	yes	0	no

Name	Description	Data Type	Optional	Default Value	Spec relevant
	running in offline mode.				

Return Value

Description	Data Type
Integer values of 0 for below VoltageLow, 1 for above VoltageHigh, and 2 for between the values (midband).	ValueList

Details

Type	Function
Spec relevant	yes

Pins.Voltage.Force

Specifies a voltage forcing value with optional current clamp and range arguments. Puts the instrument in voltage forcing mode.

Syntax

Pins(Pins).Voltage.Force (V [, IClamp] [, VRange] [, IRange] [, IClamp2])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	-
V	Voltage value to force	double	no	-	yes
IClamp	Current clamp value	double	yes	No change from previous setting	yes
VRange	Instrument voltage range	double	yes	No change from previous setting	no
IRange	Instrument current range for IClamp and IClamp2	double	yes	No change from previous setting	no
IClamp2	Second current clamp value. IClamp and IClamp2 can be used to separately set positive and negative current clamps (subject to physical	double	yes	No change from previous setting	yes

Name	Description	Data Type	Optional	Default Value	Spec relevant
	instrument capability).				

Details

Type	Function
Spec relevant	yes

Pins.Voltage.Range

The voltage forcing instrument range.

Syntax

Pins(Pins).Voltage.Range = double

Parameters

Name	Description	Data Type	Optional	Default Value
Pins	List of Pins	PinList	no	-

Property Values

Value	Description
?	Range

Details

Type	Property
Access	Write-only
Spec relevant	no

Pins.Voltage.Value

The site-aware voltage value. This is used to modify values based on measured device values which at runtime may be different for different sites. All other instrument settings will not be changed.

Syntax

Pins(Pins).Voltage.Value = SiteDouble

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes

Details

Type	Property
Access	Write-only

Spec relevant	yes

Pins.Voltage.Diffmeter.

Differential voltage measurement functions

Pins.Voltage.DiffMeter.Read

Reads the differential voltage value between the Pins() pin and LowPin.

Syntax

ValueList = Pins(Pins).Voltage.DiffMeter.Read ([Range [, SampleSize] [, SampleRate] [, DataFormat] [, OfflineValue])

Parameters

Name	**Description**	**Data Type**	**Optional**	**Default Value**	**Spec relevant**
Pins	List of Pins	PinList	no	-	yes
Range	Instrument metering range	double	yes	No change from previous setting	no
SampleSize	Number of measurements to average. When SampleSize is > 1, the return value is the average of the samples.	int	yes	1	no
SampleRate	Frequency at which to make measurements when SampleSize is > 1	double	yes	The maximum rate of the physical instrument	no
DataFormat	Format in which samples are returned	*MeasureReadFormat*	yes	Average	no
OfflineValue	Assign an offline value for program running in offline	double	yes	-9999.0	no

Return Value

Description	**Data Type**
Voltage value for each pin	ValueList

Details

Type	Function

Spec relevant	yes

Related Links

MeasureReadFormat. on page 121
Declares how the measure read format should be.

Pins.Voltage.DiffMeter.Setup

Sets up a differential measurement. The Pins() pinlist is the high-side pin and the LowPin parameter is the low-side pin. The high and low pinlists must contain only a single pin.

Syntax

ValueList = Pins(Pins).Voltage.DiffMeter.Setup (LowPin)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	High-side pin, must contain only a single pin.	PinList	no	-	yes
LowPin	Low-side pin to measure, must contain only a single pin.	PinList	no	-	yes

Details

Type	Function
Spec relevant	yes

Pins.Voltage.Meter.

Single-ended voltage measurement functions

Pins.Voltage.Meter.GetSample

Retrieves a number of samples equal to the sample size from memory. Mostly, used to get samples from strobes in a pattern.

Syntax

ValueList = Pins(Pins).Voltage.Meter.GetSample ([SampleSize] [, DataFormat] [, StoreLocation] [, OfflineValue])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
SampleSize	Number of measurements to average. When SampleSize is > 1, the return value is the	int	yes	1	no

Name	Description	Data Type	Optional	Default Value	Spec relevant
	average of the samples.				
DataFormat	Format in which samples are returned	*MeasureReadFormat*	yes	Average	no
SettlingTime	Setting time	double	yes	0	no
StoreLocation	Specifies from which location to read back the stored result	int	yes	0	no
OfflineValue	Assign an offline value for program running in offline	double	yes	-9999.0	no

Return Value

Description	Data Type
Current value for each pin	ValueList

Details

Type	Function
Spec relevant	yes

Related Links

MeasureReadFormat. on page 121
Declares how the measure read format should be.

Pins.Voltage.Meter.Read

Reads the voltage value for each pin using the meter instrument.

Syntax

ValueList = Pins(Pins).Voltage.Meter.Read ([Range [, SampleSize] [, SampleRate] [, DataFormat] [, SettlingTime] [, StoreLocation] [, OfflineValue])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
Range	Instrument metering range	double	yes	No change from previous setting	no
SampleSize	Number of measurements to average. When SampleSize is	int	yes	1	no

Name	Description	Data Type	Optional	Default Value	Spec relevant
	> 1, the return value is the average of the samples.				
SampleRate	Frequency at which to make measurements when SampleSize is > 1	double	yes	The maximum rate of the physical instrument	no
DataFormat	Format in which samples are returned	*MeasureReadFormat*	yes	Average	no
SettlingTime	Setting time	double	yes	0	no
StoreLocation	Specifies from which location to read back the stored result	int	yes	0	no
OfflineValue	Assign an offline value for program running in offline	double	yes	-9999.0	no

Return Value

Description	Data Type
Voltage value for each pin	ValueList

Details

Type	Function
Spec relevant	yes

Related Links

MeasureReadFormat. on page 121
Declares how the measure read format should be.

Pins.Voltage.Meter.Range

The voltage metering instrument range.

Syntax

Pins(Pins).Voltage.Meter.Range = double

Parameters

Name	Description	Data Type	Optional	Default Value
Pins	List of Pins	PinList	no	-

Property Values

Value	Description
?	Range

Details

Type	Property
Access	Write-only
Spec relevant	no

Pins.Voltage.Threshold.

The standard measurement functions include setting up and measuring voltage and current thresholds. A ramp is sourced which causes a trigger on a different pin. The meter reading measures the ramp value at the time of triggering.

Pins.Voltage.Threshold.Reset

Provides option to reset the threshold test.

Syntax

Pins(Pins).Voltage.Threshold.Reset (TriggerPin)

Parameters

Name	Description	Data Type	Optional	Default Value
Pins	List of Pins	PinList	no	-
TriggerPin	The name of the trigger pin that is used for the measurement on then meter pin in Pins() pinlist.	PinList	no	-

Details

Type	Function
Spec relevant	no

Pins.Voltage.Threshold.RundAndRead

Makes a threshold measurement by running a previously defined ramp and looking for a previously defined trigger. The Pins() pin is the pin to measure.

Syntax

ValueList = Pins(Pins).Current.Threshold.RunAndRead (RampName [, MeterRange] [, RampPin] [, RampToEnd] [, OfflineValue])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes

Name	Description	Data Type	Optional	Default Value	Spec relevant
RampName	Name of the ramp defined in SetupRamp()	string	no	-	yes
MeterRange	Instrument voltage metering range	double	yes	No change from previous setting	yes
RampPin	Pin on which to source the ramp, which is allowed to be different from the pin being measured.	PinList	yes	Pins being measured	yes
RampToEnd	If true, force the ramp to run to completion after triggering. If false, halt the ramp after triggering.	bool	yes	false	yes
OfflineValue	Assign an offline value for program running in offline	double	yes	-9999.0	yes

Return Value

Description	Data Type
Reading measured on the pin(s)	ValueList

Details

Type	Function
Spec relevant	yes

Pins.Voltage.Threshold.SetupRamp

Creates the specified voltage ramp for a threshold measurement. The ramp will be sourced on the Pins() pin.

Syntax

Pins(Pins).Current.Threshold.SetupRamp (RampName , RampStartV , RampEndV , RampSize [, RampFreq] [, Bandwidth])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes

Name	Description	Data Type	Optional	Default Value	Spec relevant
RampName	The name for referencing this ramp in other functions.	string	no	-	yes
RampStartV	Starting voltage	string	no	-	yes
RampEndV	Ending voltage	string	no	-	yes
RampSize	Number of equal-sized steps in the ramp.	int	no	-	yes
RampFreq	Frequency of the ramp (if it were periodic between RampStartV and RampEndV)	double	yes	1 kHz	yes
Bandwidth	Bandwidth of the ramp to be used for controlling physical instrument parameters	double	yes	5000 Hz	yes

Details

Type	Function
Spec relevant	yes

Pins.Voltage.Threshold.SetupTrigger

Sets up a trigger pin for use in a threshold measurement. The Pins() pin is the trigger pin for a measurement on.

Syntax

Pins(Pins).Voltage.Threshold.SetupTrigger (MeterPin , TriggerLevel , TriggerSlope [, TriggerRange])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Pins	List of Pins	PinList	no	-	yes
MeterPin	Pin that will be measured	PinList	no	-	yes
TriggerLevel	Voltage at which to trigger	double	no	-	yes

Name	Description	Data Type	Optional	Default Value	Spec relevant
TriggerSlope	Positive or negative slope of the trigger	*SignalSlope*	no	-	yes
TriggerRange	Range of the trigger voltage hardware	double	yes	No change from previous setting	yes

Details

Type	Function
Spec relevant	yes

Related Links

SignalSlope. on page 122
The positive or negative expected slope of a signal.

Digital.

Control of digital patterns, which affect all digital pins of the device together.

The global Digital class controls digital patterns, which affect all digital pins of the device together. Digital patterns are referred to by a name which is passed through to the tester-specific program; UTSL does not include the definition of digital pattern data (because there are well-developed third-party tools for translating digital patterns from design tools directly to tester-specific formats).

This class also includes control of digital levels and timing setups through the ApplyLevels, ApplyTiming, and ApplyLevelsTiming functions. These are used once per test step to specify the setups to be used for running patterns. They may also be used in global setup functions.

Digital.ApplyLevels

Apply a digital levels setup to this test step. This function should normally be used in test step or global setup code and may only be used once per test step. If more than one levels setup is applied, only the last one encountered will be in effect.

Syntax

Digital.ApplyLevels (LevelsNames)

Parameters

Name	Description	Data Type	Optional	Default Value
LevelsName	The name of a digital levels setup defined in the spec.	string	no	-

Details

Type	Function
Spec relevant	no

Digital.ApplyLevelsTiming

Apply both digital timing and levels setups to this test step. This function should normally be used in test step or global setup code. Only one levels and timing setup may be applied per test step. If more than one timing or levels setup is applied, only the last one encountered will be in effect.

Syntax

Digital.ApplyLevelsTiming (LevelsNames , TimingName)

Parameters

Name	Description	Data Type	Optional	Default Value
LevelsName	The name of a digital levels setup defined in the spec.	string	no	-
TimingName	The name of a digital timing setup defined in the spec.	string	no	-

Details

Type	Function
Spec relevant	no

Digital.ApplyTiming

Apply a digital timing setup to this test step. This function should normally be used in test step or global setup code and may only be used once per test step. If more than one timing setup is applied, only the last one encountered will be in effect.

Syntax

Digital.ApplyLevels (LevelsNames)

Parameters

Name	Description	Data Type	Optional	Default Value
TimingName	The name of a digital timing setup defined in the spec.	string	no	-

Details

Type	Function
Spec relevant	no

Digital.LastVectorCount

The number of vectors executed by the most recent pattern run. Returns zero if no pattern has been run.

Syntax

SiteInt = Digital.LastVectorCount ([OfflineValue])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
OfflineValue	Assign an offline value for program running in offline mode.	double	yes	0	yes

Return Value

Description	Data Type
Number of vectors executed	SiteInt

Details

Type	Property
Access	Read-only
Spec relevant	yes

Digital.LastVectorFailCount

The number of failed vectors executed by the most recent pattern run. Returns zero if no pattern has been run.

Syntax

SiteInt = Digital.LastVectorFailCount ([OfflineValue])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
OfflineValue	Assign an offline value for program running in offline mode.	double	yes	0	yes

Return Value

Description	Data Type
Number of vectors executed	SiteInt

Details

Type	Property
Access	Read-only
Spec relevant	yes

Digital.Patterns.

Functions for running digital patterns.

Digital.Patterns.Run

Starts the specified digital pattern and waits for completion.

Syntax

Digital.Patterns(PatternName).Run (Label)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
PatternName	The name of the pattern file to run. No file extension should be included since this is tester-dependent while filenames are universal.	string	no	-	yes
Label	The vector label within the file at which to start.	string	yes	-	yes

Details

Type	Function
Spec relevant	yes

Digital.Ports.

Functions for digital serial data communication.

Related Links

SerialDataFrame on page 29
Defines a serial data frame for use with the functions in Digital.Ports().

SerialPort on page 29
Defines a serial communication port for use with Digital.Ports() functions.

SerialPortGen on page 30

Defines a serial communication port for use with Digital.Ports() functions.

Digital.Ports.GetFieldRead

Extracts and returns the value of one serial bitfield from data which is a SiteInt array returned by SerialRead(). The result is suitable for passing to the Evaluate() function for datalogging.

Syntax

SiteInt = Digital.Ports(PortName).GetFieldRead (RawData , Frame , FieldName)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
PortName	The name of the a SerialPort defined in the spec.	*SerialPort*	no	-	yes
RawData	Array of integer readback data with two-bit per value encoding.	SiteInt[]	no	-	yes
Frame	A SerialDataFrame defined in the test spec.	*SerialDataFrame*	no	-	yes
FieldName	The name of a field defined in frame.	string	no	-	yes

Return Value

Description	Data Type
Result	SiteInt

Details

Type	Function
Spec relevant	yes

Digital.Ports.GetFieldWrite

Extracts and returns the value of one serial bitfield from data which is an integer array suitable for writing with SerialWrite().

Syntax

int = Digital.Ports(PortName).GetFieldWrite (RawData , Frame , FieldName)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
PortName	The name of the a SerialPort defined in the spec.	*SerialPort*	no	-	yes
RawData	Array of integer data.	int[]	no	-	yes
Frame	A SerialDataFrame defined in the test spec.	*SerialDataFrame*	no	-	yes
FieldName	The name of a field defined in frame.	string	no	-	yes

Return Value

Description	Data Type
Result	int

Details

Type	Function
Spec relevant	yes

Digital.Ports.JTAGRead

Reads and returns raw numeric data from a JTAG port using a two-bit encoding format to indicate a state of 0, 1, Midband or Glitch for each readback bit. The function GetFieldRead() can be used on the result to extract individual bitfield values from the raw data.

Syntax

SiteInt[] = Digital.Ports(PortName).JTAGRead (Frame [, UseDssc])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
PortName	The name of the a SerialPort defined in the spec.	*SerialPort*	no	-	yes
Frame	A SerialDataFrame defined in the test spec.	*SerialDataFrame*	no	-	yes
UseDssc	UseDssc flag	bool	yes	-	yes

Return Value

Description	Data Type
Raw numeric data	SiteInt[]

Details

Type	Function
Spec relevant	yes

Digital.Ports.JTAGReadWrite

Writes formatted data to the device through the specified JTAG port.

Syntax

SiteInt[] = Digital.Ports(PortName).JTAGReadWrite (Frame , Data [, Offline] [, UseDssc])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
PortName	The name of the a SerialPort defined in the spec.	*SerialPort*	no	-	yes
Frame	A SerialDataFrame defined in the test spec.	*SerialDataFrame*	no	-	yes
Data	Array of value to write.	int[]	no	-	yes
Offline	Offline value.	string	yes	-	yes
UseDssc	UseDssc flag	bool	yes	-	yes

Return Value

Description	Data Type
Raw numeric data	SiteInt[]

Details

Type	Function
Spec relevant	yes

Digital.Ports.JTAGReadWriteString

Writes formatted data to the device through the specified JTAG port.

Syntax

SiteInt[] = Digital.Ports(PortName).JTAGReadWriteString (Frame , DataString [, Offline] [, UseDssc])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
PortName	The name of the a SerialPort defined in the spec.	*SerialPort*	no	-	yes
Frame	A SerialDataFrame defined in the test spec.	*SerialDataFrame*	no	-	yes
DataString	Array of value to write.	int[]	no	-	yes
Offline	Offline value.	string	yes	-	yes
UseDssc	UseDssc flag	bool	yes	-	yes

Return Value

Description	Data Type
Raw numeric data	SiteInt[]

Details

Type	Function
Spec relevant	yes

Digital.Ports.JTAGWrite

Writes raw numeric data to a JTAG port using the number of bits defined for Frame. This function writes all frame databits and is not affected by the frame's bitfield definitions.

Syntax

Digital.Ports(PortName).JTAGWrite (Frame , Data)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
PortName	The name of the a SerialPort defined in the spec.	*SerialPort*	no	-	yes
Frame	A SerialDataFrame defined in the test spec.	*SerialDataFrame*	no	-	yes
Data	Array of values to write.	int[]	no	-	yes

Details

Type	Function
Spec relevant	yes

Digital.Ports.JTAGWriteString

Writes formatted data to the device through the specified JTAG port.

Syntax

Digital.Ports(PortName).JTAGWriteString (Frame , DataString)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
PortName	The name of the a SerialPort defined in the spec.	*SerialPort*	no	-	yes
Frame	A SerialDataFrame defined in the test spec.	*SerialDataFrame*	no	-	yes
DataString	Value to write, consisting of a list of pairs of bitfield names and values.	string	no	-	yes

Details

Type	Function
Spec relevant	yes

Digital.Ports.SerialRead

Reads and returns raw numeric data from a serial port using a two-bit encoding format to indicate a state of 0, 1, Midband or Glitch for each readback bit. The function GetFieldRead() can be used on the result to extract individual bitfield values from the raw data.

Syntax

SiteInt[] = Digital.Ports(PortName).SerialRead (Frame [, Offline] [, Alarm] [, UseDssc])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
PortName	The name of the a SerialPort defined in the spec.	*SerialPort*	no	-	yes
Frame	A SerialDataFrame	*SerialDataFrame*	no	-	yes

Name	Description	Data Type	Optional	Default Value	Spec relevant
	defined in the test spec.				
Offline	Offline value.	string	yes	-	yes
Alarm	Alarm value.	bool	yes	-	yes
UseDssc	UseDssc flag	bool	yes	-	yes

Return Value

Description	Data Type
Raw numeric data	SiteInt[]

Details

Type	Function
Spec relevant	yes

Digital.Ports.SerialWrite

Writes raw numeric data to a serial port using the number of bits defined for Frame. This function writes all frame databits and is not affected by the frame's bitfield definitions.

Syntax

Digital.Ports(PortName).SerialWrite (Frame , Data)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
PortName	The name of the a SerialPort defined in the spec.	*SerialPort*	no	-	yes
Frame	A SerialDataFrame defined in the test spec.	*SerialDataFrame*	no	-	yes
Data	Array of values to write.	int[]	no	-	yes

Details

Type	Function
Spec relevant	yes

Digital.Ports.SerialWriteString

Writes formatted data to the device through the specified port.

Syntax

Digital.Ports(PortName).SerialWriteString (Frame , DataString)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
PortName	The name of the a SerialPort defined in the spec.	*SerialPort*	no	-	yes
Frame	A SerialDataFrame defined in the test spec.	*SerialDataFrame*	no	-	yes
DataString	Value to write, consisting of a list of pairs of bitfield names and values.	string	no	-	yes

Details

Type	Function
Spec relevant	yes

Digital.Ports.SetFieldRead

Sets the specified serial bitfield to Value in raw data which is a SiteInt array returned by SerialRead().

Syntax

Digital.Ports(PortName).SetFieldRead (RawData , Frame , FieldName , Value)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
PortName	The name of the a SerialPort defined in the spec.	*SerialPort*	no	-	yes
RawData	Array of integer data.	SiteInt[]	no	-	yes
Frame	A SerialDataFrame defined in the test spec.	*SerialDataFrame*	no	-	yes

Name	Description	Data Type	Optional	Default Value	Spec relevant
FieldName	The name of a field defined in Frame.	string	no	-	yes
Value	?	SiteInt	no	-	yes

Details

Type	Function
Spec relevant	yes

Digital.Ports.SetFieldWrite

Sets the specified serial bitfield to Value in raw data which is an integer array suitable for writing with SerialWrite().

Syntax

Digital.Ports(PortName).SetFieldWrite (RawData , Frame , FieldName , Value)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
PortName	The name of the a SerialPort defined in the spec.	*SerialPort*	no	-	yes
RawData	Array of integer data.	int[]	no	-	yes
Frame	A SerialDataFrame defined in the test spec.	*SerialDataFrame*	no	-	yes
FieldName	The name of a field defined in Frame.	string	no	-	yes
Value	?	int	no	-	yes

Details

Type	Function
Spec relevant	yes

Spec.

Properties of the spec.

Spec.Author

The author name field in the test spec.

Syntax

string = Spec.Author

Return Value

Description	Data Type
Author name	string

Details

Type	Property
Access	Read-only
Spec relevant	no

Spec.DeviceName

The device name string from the test spec.

Syntax

string = Spec.DeviceName

Return Value

Description	Data Type
Device name	string

Details

Type	Property
Access	Read-only
Spec relevant	yes

Spec.Version

The device version code from the test spec.

Syntax

string = Spec.Version

Return Value

Description	Data Type
Spec version	string

Details

Type	Property
Access	Read-only
Spec relevant	no

Spec.Test.

Properties of the currently executing test.

Accessing properties of the current test (.Test) is only valid in individual test code and not in global or test module setup code.

Spec.Test.HighLimit

The test's high limit in effect at runtime.

Syntax

double = Spec.Test.HighLimit

Return Value

Description	Data Type
High limit	double

Details

Type	Property
Access	Read-only
Spec relevant	yes

Spec.Test.LowLimit

The test's low limit in effect at runtime.

Syntax

double = Spec.Test.LowLimit

Return Value

Description	Data Type
Low limit	double

Details

Type	Property

Access	Read-only
Spec relevant	yes

Spec.Test.Result

Result given to the last Evaluate() function for this test.

Syntax

SiteDouble = Spec.Test.Result

Return Value

Description	Data Type
Result	SiteDouble

Details

Type	Property
Access	Read-only
Spec relevant	yes

Related Links

Evaluate on page 114

The runtime system compares the value to the test limits and datalogs the result. The result is stored for possible future access using Spec.Tests(TestNumber).Result.

Spec.Tests.

Properties of a specific test identified by number.

Accessing by test number (.Tests(490)) can appear in any code block, but there will be a runtime error if no result has yet been logged for the requested test. For test numbers outside of the current Test Step, the optional parameter (.Tests(490, “TestStep2”)) can be provided.

Spec.Tests.HighLimit

The test's high limit in effect at runtime.

Syntax

double = Spec.Tests (TestNumber [, TestStep]).HighLimit

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
TestNumber	The test number defined in the spec	int	no	-	yes
TestStep	The test step for the corresponding test number if	string	yes	-	yes

Name	Description	Data Type	Optional	Default Value	Spec relevant
	different from current test step.				

Return Value

Description	Data Type
High limit	double

Details

Type	Property
Access	Read-only
Spec relevant	yes

Spec.Tests.HighLimit

The test's high limit in effect at runtime.

Syntax

double = Spec.Tests (TestNumber [, TestStep]).HighLimit

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
TestNumber	The test number defined in the spec	int	no	-	yes
TestStep	The test step for the corresponding test number if different from current test step.	string	yes	-	yes

Return Value

Description	Data Type
Low limit	double

Details

Type	Property
Access	Read-only
Spec relevant	yes

Spec.Tests.Result

Result given to the last Evaluate() function for this test.

Syntax

SiteDouble = Spec.Tests(TestNumber[, TestStep]).Result

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
TestNumber	The test number defined in the spec	int	no	-	-
TestStep	The test step for the corresponding test number if different from current test step.	string	yes	-	yes

Return Value

Description	Data Type
Result	SiteDouble

Details

Type	Property
Access	Read-only
Spec relevant	yes

Related Links

Evaluate on page 114

The runtime system compares the value to the test limits and datalogs the result. The result is stored for possible future access using Spec.Tests(TestNumber).Result.

Tester.

Access tester-specific functions.

The UTSL language is designed to be independent of specific test system, however to produce a working ATE program from a test spec, it may be necessary to add tester-specific programming statements. The Tester class include two functions for executing tester-specific code when necessary.

Tester.Configure

Configure tester instrument usage according to the named setup.

Sometimes a test module requires defining multiple instrument setups, for example if a different DC instrument is required. For this case, the UTSL statement Tester.Configure() is provided. This takes an argument of a setup name defined in the Tester Resource Map and applies this to subsequent lines of code in the module.

Syntax

Tester.Configure (SetupName)

Parameters

Name	Description	Data Type	Optional	Default Value
SetupName	The name of a setup defined in the tester resource map	string	no	-

Details

Type	Function
Spec relevant	no

Tester.CustomCode

Pass through an arbitrary tester-specific language code string.

The argument to this is a string which is just passed through to the test program without any error checking. These calls will typically need to be replaced in order to generate a test program for a different test platform, and so should only be used when absolutely necessary in order to preserve the tester-independence of the test spec.

Syntax

Tester.CustomCode (Code)

Parameters

Name	Description	Data Type	Optional	Default Value
Code	Tester-specific language code	string	no	-

Details

Type	Function
Spec relevant	no

Tester.Function

Call a custom tester-specific function.

The argument to this is a function call, which may contain arguments. The function call is validated for function syntax and arguments. The argument will be generated as a tester function call in the test program, and it is assumed tester code resolves the function. This method of using custom tester code is still tester-independent, in that each target test system just needs to provide a definition of the function.

Syntax

Tester.Function (FunctionName(Parameters))

Parameters

Name	Description	Data Type	Optional	Default Value
FunctionName(Parameters)	Function call with parameters	string	no	-

Details

Type	Function
Spec relevant	no

DIB.

Access tester-independent features of device interface.

DIB.Application

Set the named DIB application hardware on or off. The relays controlled by on and off are defined in the DIB Application Table.

Often a test requires a condition such as “connect Pin1 to a 10K resistor to ground”, which is done by custom hardware on the Device Interface Board (DIB). The physical hardware setup for this is tester-specific, but a test language statement called DIB.Application() allows including this specification in a tester-independent way.

This function works by assigning a string name to each different application required on the DIB. This string is passed to the DIB.Application function along with a status of whether it should be turned on or off. The code generator for a tester platform provides a mapping of these names to physical switches in a table called the DIB Application Map.

Syntax

DIB.Application (AppName , Value)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
AppName	Name from the DIB Application Table	string	no	-	yes
Value	True to turn the feature on, false to turn it off.	bool	no	-	yes

Details

Type	Function
Spec relevant	yes

Math.

Math library with commonly used functions.

Math.Abs

Returns the absolute value of a number for all numerical data types.

Syntax

ReturnValue = Math.Abs (Value)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Value	Value to analyze	see overloads	no	-	yes

Return Value

Description	Data Type
Absolute value	see overloads

Overloads

ValueList = Math.Abs (ValueList)
SiteDouble = Math.Abs (SiteDouble)
SiteInt = Math.Abs (SiteInt)
double = Math.Abs (double)
int = Math.Abs (int)

Details

Type	Function
Spec relevant	yes

Math.Log10

Returns the base 10 logarithm of a number for all numerical data types.

Syntax

ReturnValue = Math.Log10 (Value)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Value	Value to analyze	see overloads	no	-	yes

Return Value

Description	Data Type
Base 10 logarithm of value	see overloads

Overloads

ValueList = Math.Log10
(ValueList)
SiteDouble = Math.Log10
(SiteDouble)
SiteDouble = Math.Log10
(SiteInt)
double = Math.Log10
(double)
double = Math.Log10 (int)

Details

Type	Function
Spec relevant	yes

Math.Min

Returns the smaller of two values for all combinations of numerical data types.

Syntax

ReturnValue = Math.Min (Value1 , Value2)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Value1	First value to compare	see overloads	no	-	yes
Value2	Second value to compare	see overloads	no	-	yes

Return Value

Description	Data Type
Smaller value	see Overloads

Overloads

ValueList = Math.Min (ValueList,
ValueList)
ValueList = Math.Min (SiteDouble,
ValueList)
ValueList = Math.Min (ValueList,
SiteDouble)
ValueList = Math.Min (ValueList,
SiteInt)

ValueList = Math.Min (SiteInt,
ValueList)
ValueList = Math.Min (double,
ValueList)
ValueList = Math.Min (ValueList,
double)
ValueList = Math.Min (ValueList,
int)
ValueList = Math.Min (int,
ValueList)
SiteDouble = Math.Min (SiteDouble,
SiteDouble)
SiteDouble = Math.Min (SiteDouble,
SiteInt)
SiteDouble = Math.Min (SiteInt,
SiteDouble)
SiteDouble = Math.Min (SiteDouble,
double)
SiteDouble = Math.Min (double,
SiteDouble)
SiteDouble = Math.Min (SiteDouble,
int)
SiteDouble = Math.Min (int,
SiteDouble)
SiteInt = Math.Min (SiteInt,
SiteInt)
SiteInt = Math.Min (SiteInt,
double)
SiteInt = Math.Min (double,
SiteInt)
SiteInt = Math.Min (int,
SiteInt)
SiteInt = Math.Min (SiteInt,
int)
double = Math.Min (double,
double)
double = Math.Min (double,
int)
double = Math.Min (int,
double)
int = Math.Min (int, int)

Details

Type	Function
Spec relevant	yes

Math.Max

Returns the larger of two values for all combinations of numerical data types.

Syntax

ReturnValue = Math.Max (Value1 , Value2)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Value1	First value to compare	see overloads	no	-	yes
Value2	Second value to compare	see overloads	no	-	yes

Return Value

Description	Data Type
Larger value	see overloads

Overloads

ValueList = Math.Max (ValueList,
ValueList)
ValueList = Math.Max (SiteDouble,
ValueList)
ValueList = Math.Max (ValueList,
SiteDouble)
ValueList = Math.Max (ValueList,
SiteInt)
ValueList = Math.Max (SiteInt,
ValueList)
ValueList = Math.Max (double,
ValueList)
ValueList = Math.Max (ValueList,
double)
ValueList = Math.Max (ValueList,
int)
ValueList = Math.Max (int,
ValueList)
SiteDouble = Math.Max (SiteDouble,
SiteDouble)
SiteDouble = Math.Max (SiteDouble,
SiteInt)
SiteDouble = Math.Max (SiteInt,
SiteDouble)
SiteDouble = Math.Max (SiteDouble,
double)
SiteDouble = Math.Max (double,
SiteDouble)
SiteDouble = Math.Max (SiteDouble,
int)
SiteDouble = Math.Max (int,
SiteDouble)
SiteInt = Math.Max (SiteInt,
SiteInt)
SiteInt = Math.Max (SiteInt,
double)
SiteInt = Math.Max (double,
SiteInt)
SiteInt = Math.Max (int,
SiteInt)

SiteInt = Math.Max (SiteInt, int)
double = Math.Max (double, double)
double = Math.Max (double, int)
double = Math.Max (int, double)
int = Math.Max (int, int)

Details

Type	Function
Spec relevant	yes

Math.Pow

Returns Value1 raised to the Value2 power for all combinations of numerical data types.

Syntax

ReturnValue = Math.Pow (Value1 , Value2)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Value1	Base	see overloads	no	-	yes
Value2	Exponent	see overloads	no	-	yes

Return Value

Description	Data Type
Exponentiation	see overloads

Overloads

ValueList = Math.Pow (ValueList, ValueList)
ValueList = Math.Pow (SiteDouble, ValueList)
ValueList = Math.Pow (ValueList, SiteDouble)
ValueList = Math.Pow (ValueList, SiteInt)
ValueList = Math.Pow (SiteInt, ValueList)
ValueList = Math.Pow (double, ValueList)
ValueList = Math.Pow (ValueList, double)
ValueList = Math.Pow (ValueList, int)
ValueList = Math.Pow (int, ValueList)
SiteDouble = Math.Pow (SiteDouble,

SiteDouble)
SiteDouble = Math.Pow (SiteDouble, SiteInt)
SiteDouble = Math.Pow (SiteInt, SiteDouble)
SiteDouble = Math.Pow (SiteDouble, double)
SiteDouble = Math.Pow (double, SiteDouble)
SiteDouble = Math.Pow (SiteDouble, int)
SiteDouble = Math.Pow (int, SiteDouble)
SiteInt = Math.Pow (SiteInt, SiteInt)
SiteInt = Math.Pow (SiteInt, double)
SiteInt = Math.Pow (double, SiteInt)
SiteInt = Math.Pow (int, SiteInt)
SiteInt = Math.Pow (SiteInt, int)
double = Math.Pow (double, double)
double = Math.Pow (double, int)
double = Math.Pow (int, double)
int = Math.Pow (int, int)

Details

Type	Function
Spec relevant	yes

Math.Sqrt

Returns the square root of a number for all numerical data types.

Syntax

ReturnValue = Math.Sqrt (Value)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Value	Value to analyze	see overloads	no	-	yes

Return Value

Description	Data Type
Square root	see overloads

Overloads

ValueList = Math.Sqrt (ValueList)
SiteDouble = Math.Sqrt (SiteDouble)
SiteDouble = Math.Sqrt (SiteInt)
double = Math.Sqrt (double)
double = Math.Sqrt (int)

Details

Type	Function
Spec relevant	yes

Math.Truncate

Returns the integral part of a double value.

Syntax

ReturnValue = Math.Truncate (Value)

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Value	Value to analyze	see overloads	no	-	yes

Return Value

Description	Data Type
Integral part	see overloads

Overloads

ValueList = Math.Truncate (ValueList)
SiteDouble = Math.Truncate (SiteDouble)
double = Math.Truncate (double)

Details

Type	Function
Spec relevant	yes

Evaluate

The runtime system compares the value to the test limits and datalogs the result. The result is stored for possible future access using Spec.Tests(TestNumber).Result.

Syntax

Evaluate (Result [, FormatString])

Parameters

Name	Description	Data Type	Optional	Default Value	Spec relevant
Result	The site-aware result to log	see overloads	no	-	yes
FormatString	The format specifier (ANSI C format string) for the output value - e.g. %5.4f	string	yes	Testsystem depended	yes

Overloads

Evaluate (SiteDouble [,
string])
Evaluate (SiteInt [,
string])
Evaluate (SiteBool [, string])

ANSI C format string

%[Width][.Precision][Char]

Parameters

Name	Description
Width	Minimum filed width of the output field.
Precision	Number of digits to the right of the decimal point in a floating point value.
Char	Conversion character

Char

Value	Description
i	int
f	double

Details

Type	Function
Spec relevant	yes

Related Links

Spec.Test.Result on page 102
Result given to the last Evaluate() function for this test.

Spec.Tests.Result on page 104
Result given to the last Evaluate() function for this test.

Wait

Halts execution of the test until the indicated time has elapsed.

A test often needs to execute a wait until a signal settles before taking a measurement; the Wait() function supports this. A test should always have an explicit wait if one is required rather than assuming the tester statements will take a certain amount of time to execute, otherwise the test may start to fail in the future if executed on testers with faster computers or instruments.

Syntax

Wait (Time [, Type])

Parameters

Name	**Description**	**Data Type**	**Optional**	**Default Value**	**Spec relevant**
Time	Time to wait in seconds	double	no	-	-
Type	The type of Wait	*WaitType*	no	-	yes/no (depends on WaitType)

Details

Type	Function
Spec relevant	yes/no (depends on WaitType)

Related Links

WaitType. on page 125
The possible wait types for the Wait statement.

Enumerations

Enumerations are defined for parameters of the built in UTSL functions.

Syntax
enumeration.member

Some enumerations have options which can be combined with the addition operator + to select more than one member at once, for example: ConnectType.Force + ConnectType.Sense.

Related Links

Enumeration on page 33
An enumeration is a data type consisting of a set of named values called members of the type.

AlarmType.

The alarm choices for controlling alarms in TesterSettings.

The choice values can be combined by addition to select more than one. For example:
AlarmType.OpenLoop + AlarmType.OverRange

Members

Name	Description
All	All possible alarms
OpenLoop	Indicates a DCVI control loop cannot be satisfied to achieve the programmed values.
OpenKelvin	Means the force and sense lines are not connected. This is either a Connect() programming error or a wiring error on the DIB.
Force	Means the instrument is programmed to force current but is hitting the voltage clamp, or vice-versa. The instrument output is not achieving its programmed value.
Guard	Indicates the guard wire is connected incorrectly; usually this is a wiring error on the DIB.
SourceSink	Means the VI instrument is forcing current in the opposite direction of the programmed value. This usually means a programming error with the sign of the voltage or current value.
OverRange	Occurs when a meter reading is greater than the full-scale programmed range. The readback value is limited by the range rather than measuring the true signal value.

Details

Type	Enumeration
Can be combined	yes

Spec relevant	no

Related Links

Pins.TesterSettings.AlarmOn on page 57
Turns on the specified alarm, which provides real-time error checking for setup errors on the pin's instrument.

Pins.TesterSettings.AlarmOff on page 58
Turns off the specified alarm. This masks any runtime errors that the alarm might cause, which is sometimes necessary during temporary conditions like switching instrument connections.

BandwidthRange.

The bandwidth choices for Pins.TesterSettings.Bandwidth.Range. The numerical meanings of each choice are instrument-dependent.

Members

Name	**Description**
Low	?
Medium	?
High	?

Details

Type	Enumeration
Can be combined	no
Spec relevant	no

Related Links

Pins.TesterSettings.Bandwidth.Range on page 61
An approximate range of bandwidth, when specifying a precise number is not necessary. This is used most commonly to control the response time of DC instruments.

BitPolarity.

Whether a digital dignal is high or low true.

Members

Name	**Description**
LowTrue	
HighTrue	

Details

Type	Enumeration
Can be combined	no
Spec relevant	yes

ConnectType.

The possible connection types for connect and disconnect. ATE instruments often have three-wire Kelvin connections and this enum provides explicit control over physical connections. The choice values can be combined with the OR operator to select more than one at once.

Members

Name	Description
Default	Make a full connection of force, sense, and guard. This is the default when an optional connect parameter is omitted.
Force	Select the force wire only
Guard	Select the guard wire only
Sense	Select the sense wire only
Kelvin	Select the local kelvin connection only
Safe	Safe connect using ConnectVoltage

Details

Type	Enumeration
Can be combined	yes
Spec relevant	no

Related Links

Pins.Connect on page 42
Connects the indicated instrument connection if not already connected.Connects only the primary instrument if it is XSet.

Pins.Disconnect on page 43
Disconnects the indicated instrument connection if not already disconnected. Disconnects only the primary instrument if it is XSet.

DigitalLevel.

Choice of different types of digital levels.

Members

Name	Description
Vih	Drive high voltage
Vil	Drive low voltage
Voh	Compare high voltage
Vol	Compare low voltage
Vt	Termination voltage
Ioh	Output high load current. Typically a negative value.

Name	Description
Iol	Output low load current. Typically a positive value.

Details

Type	Enumeration
Can be combined	no
Spec relevant	yes

Related Links

Pins.Digital.SetLevel on page 55
Reprograms the selected digital level.

DigitalState.

The drive state of a digital pin.

Members

Name	Description
High	Force the pin to the programmed digital high level.
Low	Force the pin to the programmed digital low level.
Off	No forcing is done, allowing the device to drive.

Details

Type	Enumeration
Can be combined	no
Spec relevant	yes

Related Links

Pins.Digital.SetState on page 55
Sets the digital pin state immediately and/or at future pattern starts.

DigitalStateWhen.

Whether a digital pin DigitalState is to be forced immediately, at pattern start, or both.

Members

Name	Description
Immediate	Force the state immediately as the statement is executed.

Name	Description
PatternStart	Force the state at the beginning of each future pattern start.
ImmediateAndPatternStart	Force the state immediately and also at the beginning of each future pattern start.

Details

Type	Enumeration
Can be combined	no
Spec relevant	yes

Related Links

Pins.Digital.SetState on page 55
Sets the digital pin state immediately and/or at future pattern starts.

InstrumentType.

The tester resource map allows using more than one type of instrument within a single test setup. For these cases, this enum selects a single instrument when needed, such as when choosing which instrument to connect with the Connect() function.

UTSL has no knowledge of specific tester instruments, but it does have a concept of generic instrument types: DC, time measurement, digital, AC. These are defined by the InstrumentType enum. Many UTSL statements require the assignment of a specific insturment type. For example, Pins.Voltage requires a DC instrument and Pins.Time requires a time measurement instrument. Using these statements without assigning an instrument of the proper type will cause a program generation error. A single Tester Resource Map setup may have multiple instrument types per pin, but only one instrument of a type. For example, a pin in a setup may have DC and time measurement instruments, but not two DC instruments; otherwise the Pins.Voltage statement would not know which DC instrument to program. Because of this, defining more than one instrument of the same type on one pin in one setup is not allowed and causes a program generation error. To use a different instrument of the same type, a different Tester Resource Map setup must be defined with the new instrument assignment.

Members

Name	Description
Default	Select the default instrument, for use when an optional InstrumentType argument is left out. When the active Tester Resource Map setup has more than one instrument type selected, the default is the first one listed in the map. When the setup has a single instrument, the default chooses that one regardless of type.
Digital	Select the digital instrument
DC	Select the DC force and measure instrument, or select the high side for a differential instrument.
DClo	Select the low side of a differential DC force and measure instrument
DiffMeter	Select the differential DC voltmeter instrument

Name	Description
Time	Select the time measurement
AC	Select the AC signal source and capture instrument

Details

Type	Enumeration
Can be combined	no
Spec relevant	no

Related Links

Pins.Connect on page 42
Connects the indicated instrument connection if not already connected.Connects only the primary instrument if it is XSet.

Pins.Disconnect on page 43
Disconnects the indicated instrument connection if not already disconnected. Disconnects only the primary instrument if it is XSet.

Pins.TesterSettings.AlarmClear on page 57
Clears alarms for instruments connected to the pins so that any previous alarm conditions will not cause failures. This avoids reporting alarms due to expected events such as instrument setup, device settling or transition, or DIB switching while alarms are turned on.

Pins.TesterSettings.AlarmOn on page 57
Turns on the specified alarm, which provides real-time error checking for setup errors on the pin's instrument.

Pins.TesterSettings.AlarmOff on page 58
Turns off the specified alarm. This masks any runtime errors that the alarm might cause, which is sometimes necessary during temporary conditions like switching instrument connections.

Pins.TesterSettings.Bandwidth.Value on page 60
The expected numerical bandwidth of the pin's signal path. Affects the setting of AC signal source and capture instruments.

Pins.TesterSettings.Bandwidth.Range on page 61
An approximate range of bandwidth, when specifying a precise number is not necessary. This is used most commonly to control the response time of DC instruments.

ValueList.GetData on page 23
Finds the data stored for a specific pin.

MeasureReadFormat.

Declares how the measure read format should be.

Members

Name	Description
Average	?
ArrayData	?

Details

Type	Enumeration
Can be combined	no
Spec relevant	yes

Related Links

Pins.Current.Meter.Read on page 49
Reads the current value for each pin using the meter instrument.

Pins.Current.Meter.GetSample on page 48
Retrieves a number of samples equal to the sample size from memory. Mostly, used to get samples from strobes in a pattern.

Pins.Voltage.Meter.Read on page 83
Reads the voltage value for each pin using the meter instrument.

Pins.Voltage.Meter.GetSample on page 82
Retrieves a number of samples equal to the sample size from memory. Mostly, used to get samples from strobes in a pattern.

Pins.Voltage.DiffMeter.Read on page 81
Reads the differential voltage value between the Pins() pin and LowPin.

SerialBitOrder.

The direction in which to send or receive serial digital data.

Members

Name	**Description**
LSBFirst	Least significant bit first
MSBFirst	Most significant bit first

Details

Type	Enumeration
Can be combined	no
Spec relevant	yes

SignalSlope.

The positive or negative expected slope of a signal.

Members

Name	**Description**
Positive	Rising signal slope
Negative	Falling signal slope

Details

Type	Enumeration
Can be combined	no
Spec relevant	yes

Related Links

Pins.Current.Threshold.SetupTrigger on page 53
Sets up a trigger pin for use in a threshold measurement. The Pins() pin is the trigger pin for a measurement on.

Pins.Voltage.Threshold.SetupTrigger on page 87
Sets up a trigger pin for use in a threshold measurement. The Pins() pin is the trigger pin for a measurement on.

Pins.Time.DutyCycle.Setup on page 63
Sets up a duty cycle measurement.

Pins.Time.Frequency.Setup on page 67
Sets up a frequency measurement.

Pins.Time.Period.Setup on page 69
Sets up a period measurement.

Pins.Time.PropDelay.Setup on page 71
Sets up a propagation delay measurement between two pins. The Pins() parameter is the delay input pin which starts the measurement while the OutPin parameter is the output pin which stops it.

Pins.Time.PulseWidth.Setup on page 73
Sets up a pulsewidth measurement.

TimeHysteresis.

Controls windowing of the trigger threshold of a time measurement to prevent "chatter" caused by crossing the threshold multiple times.

Members

Name	Description
Off	No hysteresis correction, for digital signals or other signals with fase edges.
On	Correction for both positive and negative signal slopes. Typically used for duty cycle, period, frequency, and pulsewidth measurements with symmetrical rising and falling edges.
OnPositive	Optimize threshold hysteresis for signals with a positive slope, such as a rise time measurement.
OnNegative	Optimize threshold hysteresis for signals with a negative slope, such as a fall time measurement.

Details

Type	Enumeration
Can be combined	no
Spec relevant	yes

Related Links

Pins.Time.DutyCycle.Setup on page 63
Sets up a duty cycle measurement.

Pins.Time.FallTime.Setup on page 65
Sets up a fall time measurement.

Pins.Time.Frequency.Setup on page 67
Sets up a frequency measurement.

Pins.Time.Period.Setup on page 69
Sets up a period measurement.

Pins.Time.PropDelay.Setup on page 71
Sets up a propagation delay measurement between two pins. The Pins() parameter is the delay input pin which starts the measurement while the OutPin parameter is the output pin which stops it.

Pins.Time.PulseWidth.Setup on page 73
Sets up a pulsewidth measurement.

Pins.Time.FallTime.Setup on page 65
Sets up a fall time measurement.

TimeImpedance.

Declares the expected input signal impedance for a time measurement.

Members

Name	**Description**
INone	No impedance matching
I50Ohm	50Ohm impedance matching
IHighZ	High-Z impedance matching

Details

Type	Enumeration
Can be combined	no
Spec relevant	yes

Related Links

Pins.Time.DutyCycle.Setup on page 63
Sets up a duty cycle measurement.

Pins.Time.FallTime.Setup on page 65
Sets up a fall time measurement.

Pins.Time.Frequency.Setup on page 67
Sets up a frequency measurement.

Pins.Time.Period.Setup on page 69
Sets up a period measurement.

Pins.Time.PropDelay.Setup on page 71
Sets up a propagation delay measurement between two pins. The Pins() parameter is the delay input pin which starts the measurement while the OutPin parameter is the output pin which stops it.

Pins.Time.PulseWidth.Setup on page 73

Sets up a pulsewidth measurement.

Pins.Time.FallTime.Setup on page 65
Sets up a fall time measurement.

TimeStartInput.

Declares how a time measurement is triggered: by explicit command or by crossing the trigger threshold on the input pin.

Members

Name	Description
Command	Wait for a language-controlled trigger command.
InputPin	Trigger when the input pin signal crosses the programmed threshold at the specified slope.

Details

Type	Enumeration
Can be combined	no
Spec relevant	yes

WaitType.

The possible wait types for the Wait statement.

Members

Name	Description	Spec relevant
DUT	Select DUT type only.	yes
Tester	Select Tester type only.	no
Screening	Select Screening type only.	yes
Misc	Select Misc type only.	no

Details

Type	Enumeration
Can be combined	no
Spec relevant	yes/no (depends on WaitType)

Related Links

Wait on page 115
Halts execution of the test until the indicated time has elapsed.

Environment Variables

Bosch-specific environment variables.

The test spec can be configured with environment variables representing different test conditions at runtime, for example wafer or final test, hot or cold temperature, or production or characterization mode. The exact list of features is customized for each customer's production environment.

The test language supports conditional code based on the runtime setting of the environment variables. Each variable is predefined in the language as a global variable of type bool. These variables can be used in any code block.

Temperatures

Temperature	**Description**
HT	High temperature
RT	Room temperature
CT	Cool temperature

Details

Datatype	bool
Spec relevant	yes

Sequencer

Sequencer	**Description**
EWS	Wafer sort production phase
EWSHT	Wafer sort production phase, high temperature
EWSRT	Wafer sort production phase, room temperature
EWSCT	Wafer sort production phase, cold temperature
FT	Final test production phase
FTHT	Final test production phase, high temperature
FTRT	Final test production phase, room temperature
FTCT	Final test production phasee, cold temperature
EWS2	Wafer2 sort production phase
EWS2HT	Wafer2 sort production phase, high temperature
EWS2RT	Wafer2 sort production phase, room temperature
EWS2CT	Wafer2 sort production phase, cold temperature
FT2	Final test2 production phase
FT2HT	Final test2 production phase, high temperature

Sequencer	Description
FT2RT	Final test2 production phase, room temperature
FT2CT	Final test2 production phasee, cold temperature
Extended	Extended test

Details

Datatype	bool
Spec relevant	yes

Device Pins

List of all device pins.

Details

Datatype	Pin
Spec relevant	yes

Related Links

Pin on page 14
The Pin type declares a variable which can be any pin.

Main Components on page 5
At the top level there are seven major components of the test spec.

Part Variations

List of all device parts.

A single test spec can support testing multiple parts in a family. The spec includes the list of parts which are supported. The test language allows conditional code execution based on the part type being tested at runtime. Each part name defined in the spec is predefined in the language as a global variable of type bool. These variables can be used in any code block.

Details

Datatype	bool
Spec relevant	yes

Example

```
if (Part1)
{
    Pins(Pin1).Voltage.Value = 1.12V;    // Special for Part1
  }
if (Part1 || Part2)
{
    Pins(Pin2).Voltage.Value = 0.5V;    // Special for Part1 or Part2
}
```

Related Links

Main Components on page 5
At the top level there are seven major components of the test spec.

Concepts

Description of important UTSL concepts.

Parallel Test Concepts

UTSL code specifies tests for a single device. However, to boost throughput in modern production testing, multiple devices are tested in parallel by a single execution of the test program. In general, producing the parallel test production program from the single-device test UTSL code is the responsibility of the code generator for specific test equipment. However, there are two points where the test code writer is affected by parallel test considerations: the order in which pins are programmed in pinlists and using site-aware variables.

Programming List of Pins

In statements which program lists of pins, the order in which the pins are actually programmed is not defined by UTSL; this is an implementation detail of the tester code generator. Testers usually have the capability of broadcast-programming to multiple pins in parallel, and a code generator should be expected to take advantage of this.

Consider the following setup statement:

```
Pins(Pin1+Pin2).Voltage.Value = 0.5V;
```

Here Pin1 and Pin2 may be set up in either order or simultaneously. If a specific setup order is required for the test, this should be explicitly coded by using separate statements:

```
Pins(Pin1).Voltage.Value = 0.5V;
Pins(Pin2).Voltage.Value = 0.5V;
```

Site-aware Variables

When reading device values such as voltage measurements, the test is coded as though there is a single value which can be manipulated mathematically and logged as a result. However, in a parallel test environment, there is a separate copy of each measured value for each device being tested. Representing this data requires data types known as "site-aware".

This is handled in UTSL by the special data types SiteDouble, SiteInt, and SiteBool. These are similar to double, int, and bool values in how they appear in the test code, but they are implemented by the code generator as having a different value at runtime for each device site being tested.

The ValueList type used as a return value from measurement functions is also a site-aware type. The test code programmer must keep in mind that these are not interchangable with the basic types.

UTSL does not support accessing or looping through the individual site components of sitewise data types, but this can be accomplished if necessary by calling a function in tester-specific code using the Tester.Function or Tester.CustomCode features.

Legend

Description of fonts and font styles used in the syntax diagrams and colors used in the parameter and detail tables.

Legend Syntax diagram

Commands
Variable items
[ ] optional items
{ } a set of choices
| individual choices
... repeatable arguments

Legend Parameter Table

Name	**Description**	**Data Type**	**Optional**	**Default Value**	**Spec relevant**
Parameter 1	Parameter 1 is optional and non spec relevant	Data type of the value	yes	Default value	no
Parameter 2	Parameter 2 of not optional and spec relevant	Data type of the value	no		yes

Legend Detail Table

Type	Enumeration, Function or Property
Spec relevant	background color if not spec relevant